AN ENHANCED CONVOLUTIONAL NEURAL NETWORK MODEL FOR TRANSLATING KENYAN SIGN LANGUAGE INTO TEXT IN ENGLISH

MUTHUI NANCY NJOKI

A Thesis Submitted to the Graduate School in Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science in Computer Science of Chuka University

CHUKA UNIVERSITY

NOVEMBER 2024

DECLARATION AND RECOMMENDATION

Declaration

This thesis is my original work and has not been submitted for examination in any other university.

Signature: _____ Date: _____

Muthui Nancy Njoki,
SM22/45894/21.

Recommendation

This thesis has been examined, passed, and submitted with our approval as the university supervisors.

Signature: _____ Date: _____

Dr. Edna Chebet Too, PhD
Chuka University.

Signature: _____ Date: _____

Prof. David Gitonga Mwathi, PhD
Chuka University.

COPYRIGHT

DEDICATION

This is dedicated to my love, Brian Wanjiru; my parents, Mr. Lawrence Muthui and Mrs. Jane Mbogo; and my siblings, Jackline Njeri, Mercy Wanjiru, Lucy Wairimu, Pauline Wamaitha, James Kamau, Esther Wangari and Samson Mbogo, for their endless love, encouragement and support throughout the journey. I love you all.

ACKNOWLEDGMENT

I would like to thank God, whose strength, grace, and guidance have made this journey possible. My faith has been a constant source of inspiration and resilience through every challenge and achievement.

I extend my deepest gratitude to my supervisors, Dr. Edna Too, PhD and Prof. David G. Mwathi, PhD, whose knowledge, patience, and unwavering support have been instrumental in completing this work. Your guidance has encouraged me to grow academically and personally. May God bless you abundantly.

I'm grateful to my love and family for their love and encouragement. Your support has been a foundation and my motivation throughout the journey.

To my friends Lavender, Logova, Elizabeth, and my colleagues, thank you for your advice, understanding, and humor whenever I needed it. Your companionship has made this journey enjoyable.

Lastly, I thank everyone who contributed their resources, time, and insights to this study. Your generosity and encouragement cannot be taken for granted.

ABSTRACT

Most people communicate effectively and socialize through verbal means, such as talking. However, mute and deaf people cannot interact with society through speech. So, they use the non-verbal modes of communication. Non-verbal communication is a sort of usual body movements, hand gestures, and facial expressions like sign language, and this needs translation according to the specific patterns that the gestures and facial expressions or positioning of the hands, fingers, and arms carry with them during sign language. While it bridges a gap between those who can hear and those who cannot, it is by no means universally comprehended, thus standing as a barrier that leads to frustration and social exclusion of deaf people. As such, a translation tool may help convert sign language into easily understandable written language that will facilitate smooth communication between hearing and hard-of-hearing persons. While lots of research is going on in the area, little attention has been given to translating Kenyan Sign Language into some of the commonly spoken languages in Kenya. Besides, most translation tools face several challenges due to changing environmental conditions and the movement of a person while performing sign language, leading to changes in background lighting. This work translates KSL into English text through the experimental approach using a deep learning CNN model, DenseNet121, preprocessed by Contrast-Limited Adaptive Histogram Equalization. This architecture has been developed, trained, and tested on the dataset provided by the Kenyan Sign Language Classification Hackathon with an accuracy of 91.5%. The proposed model will bridge communication gaps and help include people who are hard of hearing in educational, health, and employment opportunities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AHE	Adaptive Histogram Equalization
AI	Artificial Intelligence
ASL	America Sign Language
BSL	British Sign Language
CNN	Convolutional Neural Networks
CLAHE	Contrast Limited Adaptive Histogram Equalization
CPU	Central Processing Unit
DNN	Deep Neural Network
HCI	Human-Computer Interaction
ISL	Indian Sign Language
KSL	Kenyan Sign Language
LSTM	Long – Short Term Memory
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue
RNN	Recurrent Neural Network
SL	Sign Language
SVM	Support Vector Machine
USL	Ugandan Sign Language

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Artificial intelligence is a branch of computer science that develops machines capable of doing things that generally require human intelligence, such as solving day-to-day problems and learning. AI depends on building "intelligent agents"-devices that perceive their environment and take actions that increase the possibility of successfully achieving their goals. AI comes into play when a machine can do things people commonly do. Machine learning is a field of AI that enables computers to learn without being specifically coded. It is most valuable in computational tasks for which designing and implementing exact methods would be hard or impossible. Development in machine learning has risen computers and brought forth the development of deep learning.

Neural network architectures are used in the machine learning subgroup of deep learning with many layers to support large parameter sizes for data processing using multi-nonlinear transformations. Examples of deep learning approaches are autoencoders, Deep Belief Networks, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Recursive Neural Networks, and Direct Deep Reinforcement Learning (Michael, 2015). They are widely used in natural language processing, computer vision, and predictive analytics. Applications such as these involve the creation of algorithms that are able to forecast and learn from data, shifting from static programming to data-driven decision-making.

The aim of computer vision is to transfer to a computer the ability or understanding of visual information contained in a digital image or video in ways that are considerably similar to human interpretation of the image. Computer vision uses knowledge of image formation, camera characteristics, and physical world knowledge to infer useful information from visual inputs. Advanced deep learning explains how computer vision works; hence, it performs successful applications that allow hearing and non-hearing persons to communicate with each other through sign translation systems.

Communication, in daily life, allows people to interact with each other through words and actions. Verbal actions can be written or spoken, although spoken languages remain the primary communication medium in the hearing world. Most communication models support both written and spoken languages. The methods of translations have fantastically connected individuals worldwide into different nationals by allowing cross-cultural communications to occur. However, persons with hearing and speech impairments are disadvantaged in not speaking or hearing the spoken language. Therefore, their social and economic successes are impeded in many cases as they cannot join in conversations, which is typical for people with hearing and speech impairments (Sigit & Kartika, 2016). They may resort to hearing aids, but these work best only if one is hearing ability is not too poor. For the severely impaired, communication is necessary through other techniques such as sign language, visual language, lip reading, and text-based. Deaf and mute persons rely highly on sign languages and written text; most only have their first language as a sign language.

Sign language, taught to deaf or mute persons early in life, depends upon gestures rather than sounds to give meaning. It has evolved like any other language, with various regional dialects all over the globe. Even though signing represents the native communication method for the deaf or hard of hearing, it is not even minimally comprehended by the hearing segment of society (Sharma, 2020). It comprises three salient aspects: dynamic gestures, static gestures, and facial expressions. Static gestures are those that demand only holding the hands in specific ways but without movement. Dynamic gestures are those that necessitate circular or linear hand motions. Facial expressions and body language further bring emotional shades into the communication. Most of these signs, however, have information conveyed through hand gestures, which consist of the formation and movement of arms and hands. These depend critically on hand shape, movement, location of the body, and hand orientation.

Hand gesture and sign language recognition approaches could be subdivided into conventional and deep learning-based methods. Traditional methods depend on predefined knowledge to extract features, including skin color, to segment the hand from other objects,

as in Dogra et al. 2018. In contrast, deep learning-based modern methods can perform robust modeling, capturing rich features and intuitive learning with the recent advancement of computer vision by deep learning. For example, CNNs can achieve high accuracy in recognizing gestures, as done by Jayadeep et al., 2020. Still, it is also learned by hearing people wanting to either communicate or interpret it. It is not the universal language, and each country has its form honed and patterned according to the shared experiences of the citizens in being mute or deaf.

This has been the causation for the development of languages such as American Sign Language, BSL, and specifically in Kenya, KSL, which is widely used within the country, though the hearing population often has little understanding of it

Most of the research on translating sign languages into text and speech has targeted American and Indian sign languages, while little attention is paid to KSL. Statistics from the 2019 Kenyan census show that approximately 250,000 Kenyans use KSL as their first language. Because KSL is not that well known, the hearing community relies on available KSL resources or interpreters, who are not always available. This gap calls for a machine learning-based KSL translation tool that would impart the deaf and mute community with continuous communication capabilities (Initiatives, 2020). Cheok et al. (2019) reviewed earlier studies on sign language and hand gesture recognition by explaining the process step-by-step, from data collection and preprocessing to segmentation, feature extraction, and classification. They also noticed several issues that resulted in some challenges concerning models' accuracy; for instance, illumination change of background and occlusion of hands bring high computational cost. The improvement in the performance of CNN models is related to these obstacles, which are very valuable, among which is the dataset's quality.

For example, the illumination in the background will decrease the contrast of images and lower the available information for feature extraction. Mustafa and Abdul Kader (2018) emphasized on the role of preprocessing in training the CNN models with image enhancement techniques that can increase recognition accuracy. Image enhancement can

emphasize desired features, suppress irrelevant ones, and provide excellent contrast among various objects. Enhancement of images refers to refining the quality of an image to facilitate information extraction or interpretation, aligning it with human visual perception standards. Enhancement techniques can generally be classified into two main types: frequency domain methods and spatial domain methods. The image is dealt with as a two-dimensional signal in the former, enhanced based on its Fourier transform. In contrast, the spatial domain methods involve enhancing the contrast of the image and emphasizing local features. It includes techniques like histogram equalization, successive mean quantization transforms, and gamma correction. Among these, HE has been really helpful in improving the quality of images and thus enhancing the model recognition performance. This study focused on improving the performance of the sign language translation model by reducing background illumination effects. An experimental research design was followed, where a CNN enhanced through histogram equalization was developed, trained, and tested using data from the Kenyan Sign Language Classification Hackathon.

1.2 Statement of the Problem

Despite a growing emphasis on inclusivity and increasing advancements in technology, the hard of hearing continues to face significant discrimination and barriers in various aspects of life. The hearing-impaired community encounters challenges such as access to education, employment opportunities, effective communication with hearing individuals, and social integration. As technology advances, there is a need to bridge the gap and ensure that people who are hard of hearing can converse naturally with hearing people and participate fully in all spheres of society. Less work has been done on the translation of Kenyan Sign Language (KSL) into English text. Also, background illumination often affects most of the existing tools designed to facilitate communication between hearing and hearing-impaired. This hinders the natural flow of conversation and creates frustration for hearing-impaired as they attempt to communicate seamlessly. This study aimed to develop a translation tool for KSL into easily understandable verbal language with the ability to overcome background illumination. An experimental research design was used in which a CNN model enhanced with histogram equalization was developed, trained, tested, and validated using a Kenyan Sign Language Classification Hackathon dataset. The

performance of the developed CNN model was assessed based on the model's accuracy, precision, and recall parameters.

1.3 Objectives of the Study

1.3.1 General Objective

To develop an enhanced Convolutional Neural Network model to translate Kenyan Sign Language into text in English.

1.3.2 Specific Objectives

The study met the following specific objectives:

i. To develop a Convolutional Neural Network model to translate Kenyan Sign Language into text in English.
ii. To enhance the Convolutional Neural Network model with histogram equalization to reduce the background illumination effect.
iii. To evaluate the performance of the enhanced Convolutional Neural Network model in translating Kenyan Sign Language into text in English.

1.4 Research Questions

i. How can Convolutional Neural Networks be modeled to translate Kenyan Sign Language into a textual format in English?
ii. How can we use histogram equalization to reduce the background illumination effect that affects sign language recognition?
iii. How does the developed Convolutional Neural Network model perform in translating Kenyan Sign Language into English text?

1.5 Significance of the Study

The hearing impaired, deaf or mute, use sign language, visual language, lip reading, or text to communicate with the hearing. In order to properly lip-read to a deaf person, hearing people should ensure their mouths are open while they talk. They should speak at a natural tempo, neither too fast nor too sluggish, and use short, straightforward words to improve comprehension of the topic of the discussion. Visual language involves conveying information through photographs, charts, drawings, sketches, art, and graphs. This type of

communication requires special skills from the hearing and hearing impaired to decode the message being passed. This leaves the majority of the hearing-impaired community with the option of using sign language to converse with the hearing.

The hearing, however, has little or no comprehension of the sign language. Therefore, mute/deaf people mostly depend on sign language interpreters to communicate successfully with the hearing community. Unfortunately, most hearing people are unwilling to learn sign language, leading to a scarcity of interpreters. However, the successful application of deep learning in sign language translation has made deaf people multilingual and independent. Therefore, developing a Kenyan sign language translation model can aid in minimizing the communication problem between people who can hear and those who cannot hear by enabling the hearing impaired to use their native language (sign language), which can be translated into English text.

1.6 Scope of the Study

The study mainly focused on sign language from Kenyan nationality, which was translated into English text. A convolutional Neural Network model with a histogram equalization pre-processing method was trained, tested, and verified using a static sign language dataset from the Kenyan Sign Language Classification Hackathon. The developed model detected, equalized and recognized the hand gestures used in Kenyan sign language and gave the output of the hand gesture in the form of text in English based on the trained dataset.

1.7 Assumptions of the Study

i. The deaf/mute can operate and use the Kenyan sign language
ii. The hearing person can understand written Language in English.

1.8 Operational Definition of Terms

Convolutional Neural Network	It is a deep-learning method for processing and identifying images.
Deaf	Refers to a person who is unable to hear
Hearing impaired	Refers to people with hearing loss which can range from mild to profound including those who are deaf and hard of hearing.
Hearing	Refers to people with the ability to perceive sound
Histogram equalization	This refers to the image processing method for enhancing an image by adjusting the intensity levels of pixels in an image so that the distribution of intensities is more evenly spread out.
Sign Language (SL)	Refers to non-verbal type of communication used by the deaf and mute for the purpose of message passing amongst themselves or with the hearing people.

CHAPTER TWO
LITERATURE REVIEW

2.1 Overview of Kenyan Sign Language

In Kenya, many deaf individuals and some hearing people use Kenyan Sign Language (KSL) as their native form of communication. KSL, the country's official sign language, originated in the early 1960s at two prominent schools for the deaf, Mumias and Nyang'oma, in western Kenya. Founded by the Catholic Church, these schools became vital centers for the development of KSL. The language also draws from East African gestural traditions, and as the deaf education system expanded, KSL evolved into a unified language used nationwide.

Initially, sign language was not part of the educational approach in Mumias and Nyang'oma. Instead, oralism—teaching lip-reading and spoken language—was the primary method. Despite this focus on oralism, students began to create their sign language, which later expanded as more schools for the deaf opened in central Kenya and along the coast. This growth continued with the establishment of secondary and other major town-based deaf schools.

According to a 2019 study, sign language relies on specific hand and arm movements to enable communication for individuals with hearing impairments. It includes four main manual elements: hand movement, shape, position relative to the body, and orientation. Since sign language can vary by region and community, KSL provides a standardized system in Kenya, fostering communication between hearing and deaf individuals nationwide.

Dynamic movements, immobile (static) gestures, and facial expressions are the three main parts of signing. A static gesture is one in which the hands do not move while making the sign. In the contrary, dynamic gestures involve moving the hand while signs are being made. Body language and facial expression are more closely tied to the sentiment that underlies the sign. It is important to consider how one person's body language differs from another since this affects their signing. In other words, two people may sign the same

message differently due to variations in their behaviors and body language. People with hearing loss can communicate using hand-based gestures involving arm and hand motions to identify most signs (He, 2019).

According to Xu (2017), hand gestures are a natural communication method mostly used by people with different disabilities, including those with speaking and hearing problems and those who are paralyzed, to communicate and fulfill their basic needs. Hand movement recognition has drawn more attention from researchers mainly because of its growing ability in human-computer interaction, virtual reality, sign language recognition, and robotics, Chen *et al*. (2014). It enables users to interact with a device without touching or tapping it. It falls into two categories: static, where the hand's position denotes a sign, and dynamic, in which hand movement conveys some messages. According to Mantecón et al. (2019), hand movement recognition analyzes hand motion to determine a user's intent.

Two main methods can be used to recognize hand movement: vision-based approaches and sensor-based. The sensor-based procedure uses flex sensors together with Inertial Measurement Units (IMUs) to determine the orientation of the hand and the flexion of the fingers. It is based on professional, wearable electromagnetic devices with sensors, like gloves. This is mainly used in the film industry, where a user wears an additional device with many cables. It performs well, but it is costly and unusable in some environments. The other one makes use of computer vision. It involves image processing. An image-capturing sensor, such as a camera or infrared sensor, is required, independent of the user, Xu (2017).

Various studies have been carried out on vision-based hand movement recognition as the area is widely expanding. Some studies proposed different techniques involving machine learning and deep learning methods to implement hand movement experiments. For image processing, there are many tools to extract features of an input image, as well as Artificial intelligence, which has various classifiers to classify the different types of input data, Abdulbaki (2014).

According to Initiatives (2020), the 2019 census analysis by domain shows that 150,000 of Kenyan's population have a hearing disability that ranges from mild to severe hearing loss. It also shows that around 100,000 Kenyans have a communication disability. Due to the large number of hearing and communication disability, the KSL bill was raised, as reported by the Republic of Kenya (2021). The aim is to promote the inclusion of sign language within the education curriculum and support its use in legal proceedings. The Senate approved the bill, but the president has not signed it. Once it becomes a law, the KSL will be the third National Language after English and Kiswahili. Therefore, any individual whose preferred or first language is KSL will be entitled to use Kenyan signed English language, and an interpreter will be required during all legal proceedings to translate the KSL to spoken or written language or from spoken to Kenya signed English language.

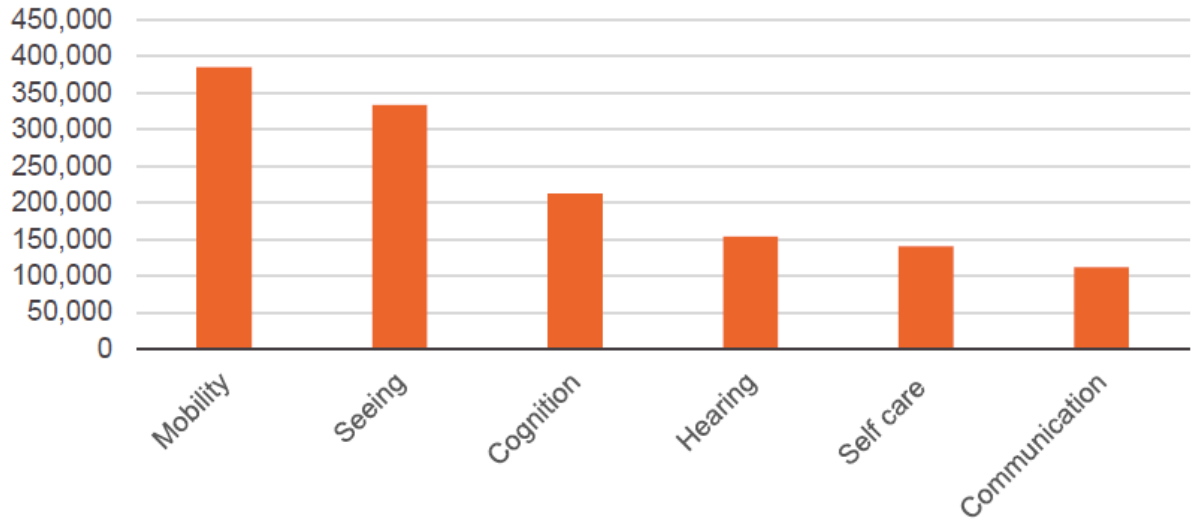

Figure 1: Number of Kenyan People with a disability, by domain, Initiatives (2020).

2.2 Machine Learning

Fundamentally, artificial intelligence is the study of agents that examine their surroundings and take actions to accomplish predetermined goals based on these observations. This field crosses several domains, including artificial intelligence, computer vision, machine learning, and natural language processing, and it draws from a variety of disciplines, comprising logic, philosophy, linguistics, mathematics, likelihood, and decision theory.

Machine learning is a subfield of AI, and its goal is to have machines learn from data analysis so that their work improves with time. The concept evolved as a subset of AI research in pattern recognition and computational learning, wherein with machine learning, algorithms learn from data and forecast using them (Ongsulee, 2018). It finds particular application in complex computational problems where developing specific algorithms to deliver assured performance is tricky, for example, in data mining, computer vision, and predictive analytics.

Machine learning procedures might differ due to data and problem complexity, and the methodology a researcher usually adopts firmly depends on the nature of the task. The major ones include supervised, unsupervised, reinforcement, semi-supervised, ensemble, and neural networks, all of which apply to different problems and goals.

2.2.1 Types of Machine Learning

a) Supervised Learning

In supervised learning, algorithms are trained on labeled data, where the inputs and expected outputs are known. During training, the algorithm compares its predictions to the actual outcomes, allowing it to learn from errors and adjust accordingly. The dataset is typically cleaved into two parts: training and testing sets. The training set comprises of output variables that the model learns to forecast or categorize, and the patterns identified from this data are then applied to the test set to refine or classify new instances. This approach is commonly used in applications that forecast future events based on historical data.

b) Unsupervised Learning

Label less data is utilized for unsupervised learning. The algorithm must determine what is displayed because it is unaware of the intended outcome. Unsupervised algorithms are designed to investigate data and identify patterns in it. From the given data, it learns a few features; when additional data is added, it uses the features it has already learned to categorize the new data. Feature extraction and grouping are the primary applications of unsupervised learning. Al-Habsi and Khan (2020).

c) Semi-supervised Learning

In semi-supervised machine learning, an algorithm is trained using unlabeled and labeled data; usually, a significant amount of unlabeled data is combined with a small amount of labeled data. This type of learning employs regression, classification, and prediction techniques. It is helpful when the expense of labeling prevents a completely labeled training procedure. Al-Habsi and Khan (2020).

d) Reinforcement Learning

The algorithm learns by trials and mistakes in reinforcement learning, which maximizes reward. It consists of three parts: actions (what the agent can do), the environment (everything the agent interacts with), and the agent (who makes decisions). The aim of reinforcement learning is to choose behaviors that boost the predicted reward over a certain period of time. The agent achieves the objective more quickly if it adheres to sound policy. Reinforcement learning, therefore, seeks to determine the optimal policy, Gad (2018).

e) Ensemble learning

Ensembles represent one of the current popular machine-learning techniques. This methodology essentially involves generating and combining several models in order to solve a certain computational challenge. This approach would boost the performance of models or reduce the risk associated with dependence upon a single model that could be far from the optimum. The advantages of ensemble learning lie in incrementality of learning, correction of errors, dynamic changes in data (non-stationarity learning), and feature selection.

f) Neural Networks

The neural network comprises a suite of algorithms influenced by the workings of the human brain, designed to recognize relationships and patterns from data. Neural networks are constituted of interconnected neurons, which can be biological or artificial, allowing this highly adaptive system to optimize its output without changing its structure. The neural network comprises three layers: an input layer that receives data, a hidden layer where data processing takes place, and an output layer, which depicts the final result after the processing of data. Neural networks operate under three paradigms: unsupervised learning, supervised learning, and reinforcement learning. The expected output for each input is known in neural networks with supervision. Therefore, the network can modify its parameters because of the error between the forecast and the actual outcome.

These networks are usually implemented in feed-forward neural architectures. Conversely, unsupervised neural networks do not possess prior knowledge of output; instead, they classify data by input similarities to spot patterns or correlations. This category includes reinforcement-based neural networks, which, in reinforcement learning, are goal-oriented algorithms that learn through trial and error. The algorithm is rewarded for making a correct decision, allowing it to focus more on desired outcomes.

Within machine learning techniques, deep learning has recently gained much prominence in representational learning, where data is structured with multiple abstraction levels. This has become a key enabler in higher-order application areas such as natural language processing and computer vision, whereby models learn to represent data in ways similar to the human brain, capturing much more complex structures in large-scale data. Deep learning originated by combining various unsupervised and supervised feature learning methods, hierarchical probabilistic models, and neural networks.

These deep learning algorithms are built and widely applied to extract insight from diversified data. Different architectures apply to different data characteristics, including recurrent neural networks, convolutional neural networks, and support vector machines, each suited for different applications, particularly in natural language processing and computer vision. This research implements a CNN model in gesture interpretation within Kenyan Sign Language and converts it into English.

2.2.2 Convolutional Neural Network

A convolutional Neural Network is a deep learning algorithm, a multilayer perceptron variation with one or more convolutional layers that may be entirely connected or pooled, according to Dubey et al. (2019). This neural network computational model receives an image as an input and then assigns different weights and biases to almost every part of it so that they will be differentiable from each other, Pathak et al. (2018). When the parts can be divided, the convolution neural network model applies numerous activation functions to complete some tasks in the image handling area, including image categorization and recognition of things, faces, etc. Sometimes, the input image is marked, and on other occasions, the input image is unmarked, depending on the form of classification used. The two classification algorithms are the Supervised Algorithm used for the images, such as in the CNN, where images are labeled, and the unsupervised Algorithm, where the image does not have any given label, Al-Qureshi et al. (2021).

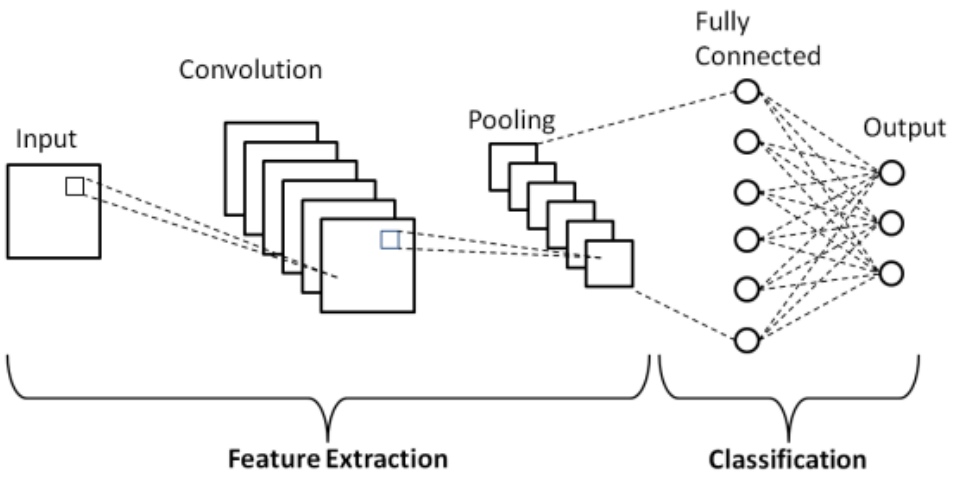

Figure 2: Layers of Convolutional Neural Network.

According to Wadhawan and Kumar's (2020) research, a CNN model comprises several layers to process input pictures and provide an output. These layers—the input layer, convolutional layer, and pooling layer—identify fundamental elements in the input picture as feature extraction tools. The fully connected layer and output layer then use these extracted characteristics to categorize the input picture into a particular group.

An input image is passed to the CNN model in the form of an array of matrices through the input layer of the model. It then steers to the convolutional layer, where the mathematical convolution method is done. Here, the image is multiplied with another two-dimensional array known as the filter or kernel and gets an output image called the activation map by sliding the filter from the top left pixel of the input image until the end and then from the top left of the next row of the input image to the correct new output images are obtained which give information about the base level features of the image, Oleinikov et al. (2018). The pooling layer then downsizes the size of the activation map to attain additional computation savings. This is achieved through some types of Pooling, including average Pooling, max pooling, and sum pooling. In the fully connected layer, the label prediction is performed with the help of activation functions like Sigmoid, etc.; the results are provided in the output layer by Gupta et al. (2021).

2.3 Image Enhancement

Image enhancement is a preprocessing mechanism that highlights an image's general or specific characteristics, clarifies an unclear image, emphasizes certain features of interest, suppresses uninteresting features, and increases the contrast between the characteristics of various items in the images. It can boost picture recognition and interpretation, improve image quality, add information, satisfy the demands of specific specialized analyses, and make images better suited to human visual systems and experiences, Pitaloka *et al*. (2017).

According to Wadhawan and Kumar's (2020) research, a CNN model comprises several layers that process input pictures and provide an output. As feature extraction tools, these layers—comprising the input, convolutional, and pooling layers—identify fundamental elements in the input picture. The fully connected output layers then use these extracted characteristics to categorize the input picture into a particular group. The enhancement will improve the dynamic range of the chosen features rather than the original information richness of the data (Rao, 2020).

Image enhancement has several strategies, and it can be divided into two groups:

a) Spatial Domain Enhancement Method

Spatial domain approaches manipulate the details of a single image pixel. It includes Neighborhood improvement algorithms and point arithmetic operations. Gray-level transformation and histogram equalization are two examples of point arithmetic operations. Linear, logarithmic, and power law transformations are all included in the gray-level transform. An image's negative transformation or identity may result from a linear transformation. The darker pixels of a picture may be enlarged via a logarithmic transformation while higher values are compressed, producing an improved image. N^{th} root or n^{th} power transformation may be a part of power law transformation. Depending on the value of n, the degree of amplification may be modified. The consistent distribution of gray levels could be ensured through histogram equalization (Vijayalakshmi et al., 2020).

Image sharpening and smoothing algorithms are examples of neighborhood enhancement techniques. Picture sharpening emphasizes the edges while lowering contrast, while image smoothing removes image noise.

b) Frequency Domain Enhancement method

Enhancement is done by first converting the image into the frequency domain through the Fourier transform, filtering the frequency components, and then inverse Fourier transforms the filtered frequency components back into the spatial frequency domain. Normally, homomorphic, low-pass, and high-pass filtering are this operation's most applied filtering methods. Low-pass filtering eliminates high-frequency noise and keeps only low-frequency components. In such a case, Butterworth low-pass filters can eliminate sharp edges caused by noise; hence, they may denoise and keep the main outlines of interest intact. Typically, homomorphic filters are used to separate between the illumination and reflectance components. High-pass filters suppress low-frequency elements and amplify high-frequency details.

2.4 Enhanced Convolutional Neural Network with Histogram Equalization

An illustration of the distribution of pixel intensities in a picture is called a histogram. The X-axis shows intensity values, and the Y-axis shows the frequency of each intensity, usually on a scale from 0 to 255, and depicts the range of tones in a digital picture. This gives information on the image's brightness, contrast, and general tonal distribution. A grayscale image contains one histogram, while an RGB color image contains three different histograms, each matching the three-color channels. A basic contrast enhancement method called histogram equalization modifies the pixel intensity distribution to produce a more consistent level. By expanding the dynamic range of pixel values, this technique enhances the visibility of features in both the image's brighter and darker regions. It transforms the original grayscale levels of an image, denoted, into enhanced levels via some transformation function, which emanates from the accumulated distribution of pixel intensities (Shukla, 2017).

With that in mind, Pandya et al. (2018) developed a deep-learning fingerprint recognition that incorporated preprocessing steps such as histogram equalization, Gabor filtering, and fingerprint thinning to reinforce the details in the texture. Further, a deep convolutional neural network is adopted for classifying the preprocessed fingerprints, and they had 98.21% accuracy with a loss rate of 0.9, while earlier results on the very same dataset were at 77% accuracy.

In Pitaloka et al. 2017, a CNN was further enhanced to identify six basic emotions: disgust, fear, anger, sadness, and happiness. The authors applied various preprocessing methods, and their effect on CNN was investigated. Thus, methods used included face detection, resizing, cropping, noise addition, and normalizations like local and global contrast normalization and histogram equalization. The system based only on face detection achieved an accuracy of 86.08%, while the proposed combined methodology increased the performance of CNNs by up to 97.06%.

Abdel-Salam et al. (2022) proposed a real-time method, Real-Time Image Enhanced CNN(RIECNN), for traffic sign recognition in intelligent vehicles. This methodology adapted any issues regarding brightness and contrast through preprocessing steps: resizing all images to 60x60 pixels, enhancing contrast, and using the Multi-scale Retinex algorithm to enhance color consistency. Further, histogram equalization was applied to sharpen edges and amplify the contrast of the image before the images were turned to grayscale. RIECNN outperformed other models in recognition and speed.

2.5 Sign Language Translation

Sign language is the main mode of communication for those who are mute or deaf. However, since most hearing people do not learn sign language, contact between those who can hear and sign language users is frequently restricted. This obstacle emphasizes the necessity of technology and solutions to address this communication gap. Typically, only close relatives and friends of deaf or mute individuals learn sign language, often necessitating a human interpreter for communication (Albrecht, 2014). Unfortunately, interpreters are not always available, creating a communication barrier.

Various sign language recognition and translation technologies have been developed to address this. This study focuses on translating Kenyan Sign Language (KSL) into speech, though existing technology is predominantly geared toward American, British, and Indian sign languages.

Chandra (2019) developed a machine learning-based prototype to convert sign language gestures into speech, supporting both ISL and ASL, enabling deaf or mute individuals to communicate in multiple languages. The prototype featured three main functions: sensing, signal processing, and running a machine learning algorithm. It included a glove with flex sensors, accelerometers, and gyroscopes that captured the user's real-time gestures. An Arduino Nano microcontroller collected data from these sensors and transmitted it to a PC via Bluetooth, where a SVM algorithm classified the gestures. Using ASL and ISL datasets, the SVM classifier achieved a 100% accuracy rate for ISL and 98.91% for ASL. However, the prototype used only one glove, making it challenging to detect two-handed gestures.

Hasan et al. (2020) proposed a deep CNN for perceiving 24 stationary ASL alphabet gestures. The model architecture comprises six convolutional layers, all using ReLU as an activation function, followed by three Max Pooling layers. While the first two convolutional layers consisted of 64 filters, the middle layers increased to 128, while the filter size rose to 256 in the last layers; all the layers have been Vancouver with a 3x3 filter size. Features extracted then passed through a fully connected layer comprising 256 neurons, while the output layer consisted of 24 nodes that utilized the SoftMax function for classifying the ASL alphabets. It yielded an accuracy rate of 97.62%, but its limitation was only static gesture recognition.

He proposed a sign language detection system integrating a 3D CNN and an LSTM network. Capable of detecting and tracking gesture areas, the system leveraged 3D CNNs for feature extraction and then utilized LSTM encoding-decoding networks to recognize gestures from sequences. It consisted of 10,000 pictures of the 40 most commonly used words, split 80% for training and 20% for validation, with an achieved recognition rate of 99.0%. Even though the model's accuracy is very high, its vocabulary is limited because the model has focused on certain gestures and the precision of the algorithm.

Accordingly, Bantupalli (2018) developed a vision-based software that could translate sign language gestures into texts to foster communication between non-signers and signers. In this, CNN and RNN models were connected to interpret the recorded ASL gestures as continuous videos, aiming at bridging the communication gap. The CNN model, Inception v3, used transfer learning, with its global pooling layer producing a 2048-dimensional feature vector. This was fed into an LSTM model, allowing recognition of gestures with longer temporal dependencies. The model achieved 93% accuracy, but its performance declined with variations in skin tones, the presence of faces, and changes in clothing, highlighting areas for further refinement.

The decision tree algorithm proposed by Johnny and Nirmala (2022) helped translate the sign language used by deaf people to their corresponding words. The first phase involved capturing both moving and static gestures using a webcam, after which the image is processed with a tf–pose-estimation algorithm in tensor-flow, which basically maps out the skeleton of the signed gesture. The processed images are then appended to the Decision tree algorithm for prediction. The decision tree achieved an accuracy of 80% on stomach ache and headache gestures and 70% on dancing and studying gestures. The system works well but only recognizes the gestures and displays their corresponding words. The system also does not recognize gestures signed using fingers.

Dogra et al. (2018) developed a sign language interpreter to enable computers to classify ASL signs and change these signs into text. This system employs machine learning and Human-Computer Interaction (HCI), utilizing a camera to capture the input gesture image from an ASL user. The system first detects whether the input is a hand gesture before performing image recognition through a Support Vector Machine (SVM) classifier. The recognized gesture is then converted into text, allowing effective interaction between the mute and deaf community and computers, bridging the gap in communication with the hearing community. However, the system is limited to recognizing only stationary gestures, as it cannot interpret dynamic gestures that require video input.

Abraham (2019) developed a gesture-sensing glove to translate Indian Sign Language (ISL) into speech in real time. The glove integrated gyroscopes, flex sensors, and accelerometers to capture dynamic movements and hand gesture data. Data collected by the glove was transmitted wirelessly via Bluetooth to a processing device, where it was classified into speech and text outputs using LSTM networks. The model was trained on 26 ISL words, with 40 samples per word, achieving a 98% accuracy on test data. However, this glove was limited to hand gestures and could not recognize gestures involving other body parts.

Rao et al. (2018) introduced a CNN-based model for recognizing Indian Sign Language gestures through selfie videos. These videos were taken using a mobile front camera, with the user holding a selfie stick in one hand while signing with the other. This selfie mode allowed users with hearing impairments to independently work with the sign language recognition application. Since no public dataset of ISL selfie-mode videos existed, a dataset was created with 200 commonly used ISL words recorded by five native ISL users from five different angles. They used the dataset to train and test the CNN model, achieving an accuracy of 92.88% on videos with consistent backgrounds.

Wadhawan and Kumar (2020) introduced a paradigm based on CNN to recognize static signs in ISL. The CNN model included convolutional layers, ReLU activations, and max-pooling layers with varying filter sizes to improve recognition accuracy and processing speed. A dataset of 35,000 images representing 100 stationary ISL signs was collected under diverse environmental circumstances using a webcam and used for training, validation, and testing. The model achieved a training accuracy of 99.90% on grayscale images and 99.12% on colored images, showing strong performance based on metrics like recall, precision, and F-score. However, additional training is needed for the model to recognize dynamic signs accurately.

Chong and Lee (2018) developed a prototype for recognizing ASL using a Leap Motion Controller (LMC) designed to interpret 26 letters and 10 digits. Their research also focused on extracting gesture features to distinguish between dynamic and static signs. The LMC, a compact device known for accurately tracking hand and finger movements, was used to collect data, which was then preprocessed to extract relevant features. To evaluate performance, this data was classified using both SVM and DNN models. The study realized that DNN outperformed SVM in classifying ASL gestures and analyzing gesture similarities, with DNN achieving an accuracy of 88.79% compared to SVM's 72.79%. However, the prototype had limitations, as it could not compute hand and finger orientation or detect finger overlaps.

Table 1: Classification of various sign language translation and recognition techniques.

Authors	Year of Publication	Algorithm or notation used	Accuracy	Limitations
Chandra	2019	Support Vector Machine (SVM) algorithm	100% on ASL and 98.91% on ISL	One glove was designed for the prototype making detecting two hand sign language gestures difficult
Hasan et al.	2020	Deep Convolutional Neural Network (Deep CNN)	97.62% on ASL Alphabet	The developed model could classify static gestures.
He	2019	3D CNN and LSTM	99.0% on 10,000 sign images	Used a limited dataset, and the focus of the research was on the accuracy of the algorithm. The actual time performance of the model was not considered.
Bantupalli	2018	Inception v3 and LSTM	93% on ASL data set	The accuracy of the model dropped on: i. Testing the model with different skin tones ii. Inclusion of faces iii. Variation in clothing
Johnny and Nirmala	2022	Decision Tree	80% on stomach ache and headache gestures 70% on dancing and studying gestures	The system only recognizes the gestures and displays their corresponding words. The system doesn't recognize gestures signed using fingers.
Dogra et al.	2018	SVM		The interpreter uses only static gesture recognition
Abraham	2019	LSTM	98% on 26 Indian Sign Language words	To better categorize the signals, the hand's orientation and relative position to the body are not taken into account.

Rao et al.	2018	CNN	92.88%	The model is limited to constant video backgrounds.
Wadhawan and Kumar	2020	CNN	99.90% on grayscale images and 99.12% on colored images	The model cannot recognize dynamic ISL signs
Chong and Lee	2018	SVM and Deep Neural Network (DNN)	72.79% for SVM and 88.79% for DNN	The model does not: i Compute the hand and finger orientation ii Detect overlapping between fingers

CHAPTER THREE
METHODOLOGY

3.1 Study Site

The experiments were carried out at Chuka University, Tharaka Nithi County, with geographic coordinates of 0.3195°S and 37.6575°E. An improved Convolutional Neural Network model was developed and trained, testing and validating translation into English text from Kenyan Sign Language within this university's computer laboratory. The model leveraged computations of Google Cloud Platform and Kaggle to process the model.

3.2 Research Design

A research design is a plan of action that directs a study and helps with data gathering, analysis, and observational interpretation. To properly address the questions guiding the research, researchers can utilize a research design as a guide to assist them in making a good decision on the instruments and methods they can employ for information collecting and evaluation, Kamiri and Mariga (2021).

The research design adopted for this study was an experimental design, which involved the planning and running an experiment to arrive at a result. The flow of the experiment generally is summarized in Figure 3. The one important algorithm considered for the processing of gestures was the Convolutional Neural Network due to its high accuracy in tasks that involve image recognition and object classification. The hand-based gesture inputs fed into the CNN model were from the Kenyan Sign Language Classification Hackathon. These were pre-processed input images to improve quality and make them suitable for model training. When the CNN models were being trained, they were validated with a labeled KSL dataset, where comparisons betwixt the model's prediction and actual labels were performed. The final output from the models was translated into English text.

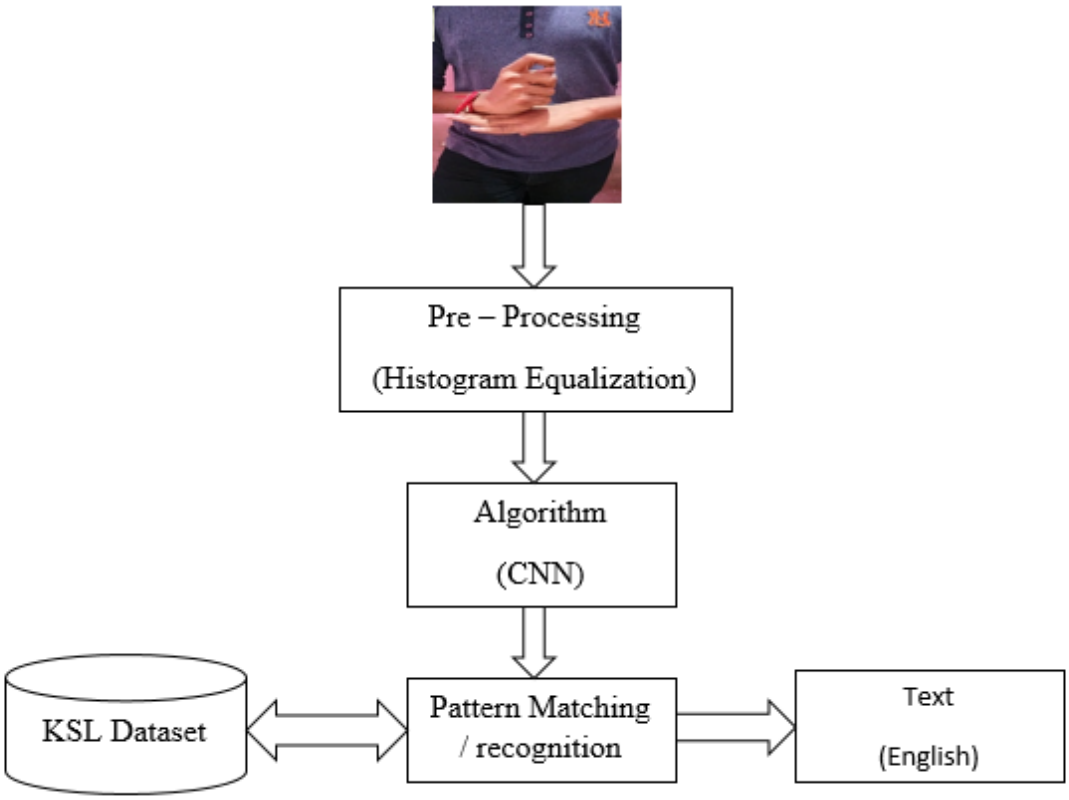

Figure 3: General Research Design.

3.3 Data Collection

The study utilized secondary data from the Kenyan Sign Language Classification Hackathon dataset. The dataset is a static hand gesture dataset collected from sign language users in Kenya representing signs in Kenyan Sign Language. It constituted 8928 images categorized into 9 different classes. The classes are Love, Temple, Seat, Me, Enough/Satisfied, you, Church, Mosque, and Friend. The dataset was used because it is the only public and freely available Kenyan Sign Language dataset. Figure 4 illustrates samples of the hand gestures in the Kenyan Sign language dataset.

Figure 4: Examples of KSL signs available in the dataset.

Before training the CNN models, we needed to verify the number of images the Kenyan Sign language dataset contained. It was also important to validate that the dataset had all nine classes of hand gestures. Figure 5 summarizes the Kenyan Sign Language dataset, indicating all the classes and the number of images per class. From the figure, we have all nine classes in our dataset, with classes Mosque, Enough/Satisfied, and Seat having 695 images per class and Friend, Me, Love, Church, You, and Temple having 694 images per class.

Figure 5: Summary of the KSL dataset.

3.3.1 Data Loading

This is an important step whereby the dataset is read into memory and transformed into a form suitable for model training. We used the 'ImageDataGenerator' class in Keras, a deep learning library, to load and augment the data. It is essential for training convolutional neural networks (CNNs) on image data. Data augmentation helps boost the training dataset's diversity and size by applying transformations to the images. This is usually done to reduce overfitting and help improve model performance by exposing it to various altered versions of the same image. Some transformations applied were brightness and contrast adjustments, which helped randomly change the contrast of the images, and width and height shifts that randomly translated images vertically or horizontally.

3.4 Model Construction

Model construction is the model's design, development, and testing that will be used within this study. It outlines the methodical procedure of processing the image dataset to eventually result in a fully trained classifier for KSL. Indeed, an improved CNN model with histogram equalization was used to recognize KSL hand-sign gestures and their translations into text in English. The model identifies hand-signed words itself and converts them into English text. Figure 6 shows the workflow of the developed model.

Our model used a secondary dataset from the Kenyan Sign Language (KSL) classification hackathon that consisted of a collection of images categorized into nine classes and their corresponding labels. The deep learning procedure used for our study is a CNN with its various architectures trained on the KSL dataset and their performance compared. The CNN architectures used in our study include ResNet50, Inception Resnet, InceptionV3, DenseNet121, and VGG16. Before the models were trained, data processing was done. This involved preparing the KSL images for training purposes by removing unnecessary portions of the images to enable the models to focus on the relevant areas, ensuring that all KSL images have consistent dimensions for input into the models, converting categorical labels into numerical representations suitable for the CNN algorithms for use as output classes for training, and applying techniques to improve the quality of our KSL images and address variations in lighting. Herein, CLAHE and AHE were used for image

enhancement. During model training, careful choices of hyperparameters like an optimizer, learning rate, batch size, and number of epochs had to be made to gain maximum accuracy on the validation set.

Lastly, after training, the model performance was checked on the validation dataset by classification reports, precision-recall curves, and training/validation loss curves to evaluate the models' effectiveness comprehensively.

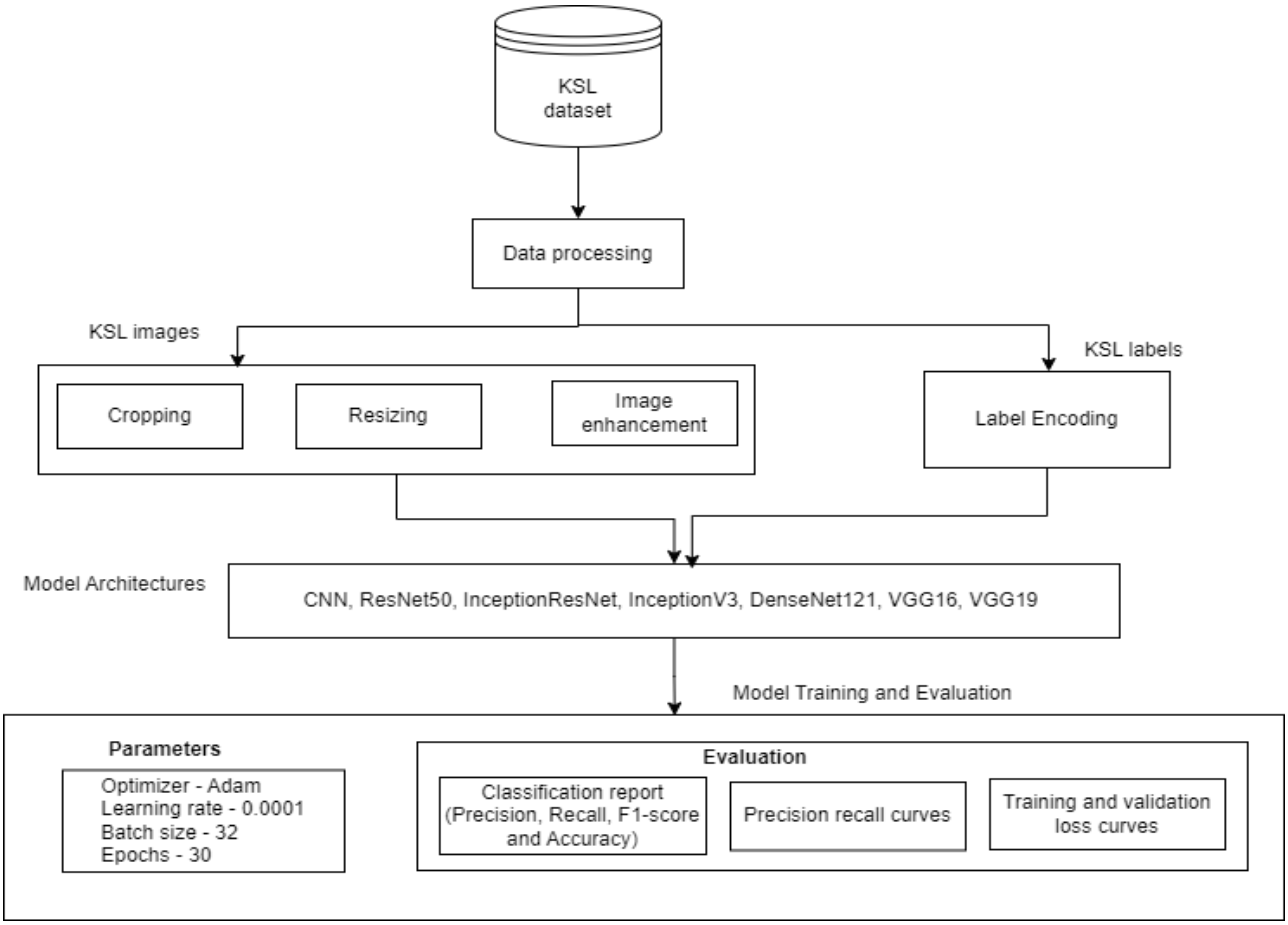

Figure 6: Pipeline of the developed models.

3.4.1 Data Pre-processing

This involves data normalization, noise reduction, cleaning, and augmentation where necessary. Data normalization involves scaling data to ensure consistency between training and testing datasets, enhancing the model's overall efficiency. Noise reduction, on the other hand, eliminates outliers or inconsistent data items that could hinder model performance. To improve the efficiency of the CNN models in this study, images underwent both normalization and noise reduction processes in the KSL dataset, which were also preprocessed using histogram equalization techniques to remove the background illumination effect. The two main techniques employed were:

Adaptive Histogram Equalization (AHE): AHE improves the contrast of images in a local context rather than the entire image, making it particularly useful for images with varying lighting conditions. This technique was implemented using exposure.

Contrast Limited Adaptive Histogram Equalization (CLAHE): CLAHE is an extension of AHE that prevents noise overamplification by limiting the contrast enhancements in localized regions. This technique was applied using OpenCV's 'cv2.createCLAHE' function. CLAHE is particularly effective in improving the visibility of gestures in poorly lit images, which is important for accurate gesture recognition. Label encoding was also done during the pre-processing stage to convert the categorical KSL label into a numerical format that the models can process.

3.4.2 Model Development and Training

CNN is a type of supervised deep learning; therefore, the model must be trained, tested, and validated to get a high-accuracy model. Several deep-learning architectures were used for training and verification to identify the well-performing model on the Kenyan Sign Language dataset. They include: DenseNet121, ResNet50, Inceptionv3, InceptionResNetV2 and VGG16. The model's development, training, and validation were conducted on the Kaggle platform, which offers a robust, integrated environment ideal for model building. Kaggle provides the computational resources and tools necessary for efficient development, making it suitable for this study's requirements.

During training, an adaptive learning rate optimization algorithm, the Adam optimizer, was used to minimize the loss function. Adam was well-suited for this task as it adapts the learning rate for each parameter, enhancing training efficiency and helping the model converge more quickly. It was adopted for its wide use in image classification tasks because of its efficiency and performance on complex data and its ability to adapt learning rates based on the first and second moments of gradients. A learning rate of 0.001 was set to control the step size at each iteration, guiding the model toward minimizing the loss function. This rate allowed for stable convergence while avoiding overshooting and balancing learning speed and accuracy during training. The lower learning rate was chosen to ensure slow but steady convergence, reducing the risk of overshooting the optimal parameters during backpropagation. 40 epochs were used per model with a batch Size of 32. The batch size was selected to balance memory efficiency and training speed, providing sufficient data for each update without overwhelming the system memory.

3.4.3 Model Validation

The developed CNN models were verified to ensure they generalized well to new Kenyan Sign Language images. The study aimed to enhance a CNN model using a histogram equalization technique to reduce background illumination effects and translate the Kenyan Sign Language into a textual format in English. The KSL dataset was divided into training and validation sets. The training set was used to train the models, and the validation set was used to verify that the models behaved as expected. 30% of the image dataset validated the model. This percentage was used to provide the models with a reasonable amount of data for validation, allowing a reliable assessment of their performance.

3.5 Model Performance Evaluation

Several measures were applied to assess the model's performance, including training and validation loss curves and a classification report. Kamiri and Mariga (2021) identify precision as an important metric that evaluates the ratio of properly categorized positive samples to the total number of samples projected as positive (including both right and inaccurate predictions). This statistic evaluates how well the model detects actual positives among all the positive predictions.

$$Precision = \frac{TP}{TP+FP} \quad \text{..... (1)}$$

Where: TP → True Positive

FP → False Positive

Recall is the ratio of correctly classified positive samples to the total number of positive samples. This metric reflects the model's capacity to identify all relevant instances within a dataset.

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (2)}$$

Where: TP → True Positive

FN → False Negative

Precision refers to how sure the model is that the sample is positive, while recall refers to the model's capacity to identify the presence of actual positive samples. A PR curve shows a balance between precision and recall across different thresholds and informs of the different sensitivities of models regarding the detection of true positives.

3.6 Hardware/ Software Requirements

3.6.1 Hardware Requirements

The experiments were conducted on a laptop with an Intel Core i7- 8550U CPU, 16GB RAM, and a minimum of 256 GB internal storage. This robust hardware configuration ensured optimal performance during the various stages of the research. An external hard disk with a minimum storage capacity of 500 GB was deemed essential for secure backup of project codes and documentation.

3.6.2 Software Requirements

A 64-bit Windows operating system was required to ensure seamless machine functioning and compatibility with the chosen software tools. Google Chrome or Firefox browsers were used for compatibility and efficient web navigation capabilities. Google Cloud Platform and Kaggle were used to train, test, and evaluate the performance of the enhanced CNN model in translating Kenyan Sign Language into text in English. At the same time, Kaggle provided a free GPU P100 version, which was utilized for seamless execution and training of the model. Various Python libraries, including NumPy, Keras, tensorflow, Scikit-learn, and Pandas, were instrumental in loading, pre-processing, and modeling. These libraries provided essential tools for evaluating and enhancing model performance.

3.7 Ethical Considerations

The researcher sought approval of the research proposal from the Chuka University Ethics Committee, followed by a research permit from the National Commission for Science, Technology, and Innovation (NACOSTI). These approvals ensured that the study complied to ethical standards and regulatory requirements. The research was conducted professionally, and the researcher ensured the referenced work was cited accordingly to avoid plagiarism. The study's findings will be published properly to aid in the progress of Artificial intelligence in Kenyan Sign Language translation.

CHAPTER FOUR
RESULTS AND DISCUSSIONS

4.1 Introduction

The findings are presented in three sections, corresponding to the particular objectives stated in Chapter One. Section A gives the results for Objective 1, relating to developing a Convolutional Neural Network model for translating Kenyan Sign Language into English text. The second section discusses the results concerning Objective 2, which involves enhancing the proposed CNN model by adding histogram equalization as a preprocessing step to reduce background illumination artifacts. The results of Objective 3 about the performance evaluation of the enhanced CNN model in translating KSL into English are discussed in the last section.

The Kenyan Sign Language dataset was used to train several CNN models: DenseNet121, ResNet50, Inceptionv3, InceptionResNetV2, and VGG16. These are pre-trained model architectures on large-scale datasets so that a researcher can use transfer learning to improve performance with limited training data. These architectures have shown excellent performance in computer vision applications and are, therefore, quite appropriate for Sign Language Translation since they can capture complicated patterns that might be necessary due to the variations of hand gestures.

All the models were trained with original images and enhanced images using AHE and CLAHE techniques. The performance results of each model were recorded for further comparative analysis. The performance of DenseNet121 during training with no enhancement of the images can be viewed in Figure 7, with AHE enhancement in Figure 8 and CLAHE enhancement in Figure 9. Among the other models, DenseNet121 stood at the forefront regarding performance on the KSL dataset.

```
Epoch 7: val_accuracy improved from 0.75938 to 0.79581, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 31s 189ms/step - accuracy: 0.7369 - loss: 0.8304 - val_accurac
y: 0.7958 - val_loss: 0.6604
Epoch 8/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.7887 - loss: 0.6784
Epoch 8: val_accuracy improved from 0.79581 to 0.82340, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 31s 190ms/step - accuracy: 0.7888 - loss: 0.6782 - val_accurac
y: 0.8234 - val_loss: 0.5947
Epoch 9/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.8167 - loss: 0.6010
Epoch 9: val_accuracy improved from 0.82340 to 0.84106, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 31s 190ms/step - accuracy: 0.8168 - loss: 0.6008 - val_accurac
y: 0.8411 - val_loss: 0.5367
Epoch 10/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.8509 - loss: 0.4900
Epoch 10: val_accuracy did not improve from 0.84106
165/165 ━━━━━━━━━━━━━━━━━━━━ 30s 185ms/step - accuracy: 0.8509 - loss: 0.4900 - val_accurac
y: 0.8389 - val_loss: 0.5145
Epoch 11/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 172ms/step - accuracy: 0.8690 - loss: 0.4297
Epoch 11: val_accuracy improved from 0.84106 to 0.85099, saving model to /kaggle/working/m
odel.keras
```

Figure 7: DenseNet121 Model without enhancement training process.

```
Epoch 7: val_accuracy improved from 0.78477 to 0.83554, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 32s 196ms/step - accuracy: 0.7587 - loss: 0.7490 - val_accurac
y: 0.8355 - val_loss: 0.5955
Epoch 8/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - accuracy: 0.8148 - loss: 0.6125
Epoch 8: val_accuracy improved from 0.83554 to 0.85210, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 32s 195ms/step - accuracy: 0.8148 - loss: 0.6124 - val_accurac
y: 0.8521 - val_loss: 0.5406
Epoch 9/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - accuracy: 0.8451 - loss: 0.5172
Epoch 9: val_accuracy improved from 0.85210 to 0.86093, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 32s 195ms/step - accuracy: 0.8451 - loss: 0.5171 - val_accurac
y: 0.8609 - val_loss: 0.4698
Epoch 10/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 0.8626 - loss: 0.4382
Epoch 10: val_accuracy improved from 0.86093 to 0.86534, saving model to /kaggle/working/m
odel.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 33s 197ms/step - accuracy: 0.8627 - loss: 0.4381 - val_accurac
y: 0.8653 - val_loss: 0.4602
Epoch 11/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - accuracy: 0.8756 - loss: 0.3885
Epoch 11: val_accuracy improved from 0.86534 to 0.87859, saving model to /kaggle/working/m
odel.keras
```

Figure 8: DenseNet121 Model with AHE enhancement training process.

```
Epoch 8: val_accuracy improved from 0.82781 to 0.84106, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 33s 201ms/step - accuracy: 0.8018 - loss: 0.6421 - val_accurac
y: 0.8411 - val_loss: 0.5334
Epoch 9/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 183ms/step - accuracy: 0.8413 - loss: 0.5543
Epoch 9: val_accuracy improved from 0.84106 to 0.86203, saving model to /kaggle/working/mo
del.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 33s 201ms/step - accuracy: 0.8414 - loss: 0.5541 - val_accurac
y: 0.8620 - val_loss: 0.4829
Epoch 10/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 183ms/step - accuracy: 0.8638 - loss: 0.4678
Epoch 10: val_accuracy improved from 0.86203 to 0.87638, saving model to /kaggle/working/m
odel.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 33s 201ms/step - accuracy: 0.8638 - loss: 0.4677 - val_accurac
y: 0.8764 - val_loss: 0.4259
Epoch 11/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - accuracy: 0.8746 - loss: 0.4177
Epoch 11: val_accuracy improved from 0.87638 to 0.87859, saving model to /kaggle/working/m
odel.keras
165/165 ━━━━━━━━━━━━━━━━━━━━ 33s 200ms/step - accuracy: 0.8746 - loss: 0.4175 - val_accurac
y: 0.8786 - val_loss: 0.4331
Epoch 12/40
165/165 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - accuracy: 0.8909 - loss: 0.3717
Epoch 12: val_accuracy improved from 0.87859 to 0.88300, saving model to /kaggle/working/m
```

Figure 9: DenseNet121 Model with CLAHE enhancement training process.

After the model training process was completed, a summary of the DenseNet121 model was extracted which is indicated in Table 2.

Table 2: DenseNet121 Model summary.

Layer name	Output shape	Number of parameters
conv5_block16_concat	(None, 7, 7, 1024)	0
bn	(None, 7, 7, 1024)	4096
relu	(None, 7, 7, 1024)	0
Average_pooling2d	(None, 1, 1, 1024)	0
flatten	(None, 1024)	0
dense	(None, 256)	262400
dropout	(None, 256)	0
Dense_1	(None, 128)	32896
Dropout_1	(None, 128)	0
Dense_2	(None, 9)	1161
Model parametres Summary		
Total params; 7,333,961		
Trainable params: 7,333, 961		
Non-trainable params: 0		

ResNet50, Inceptionv3, InceptionResNetV2, and VGG16 Convolutional Neural Network architectures did not perform well on our KSL dataset compared to DenseNet121. When all the models were trained, their model summaries were also extracted, as shown in Tables 3, 4, 5, and 6, respectively. The summaries showed the total number of learnable parameters, all trainable parameters, and non-trainable parameters in the models.

Table 3: ResNet50 Model Summary.

Layer name	Output shape	Number of parameters
conv5_block3_3_bn	(None, 7, 7, 2048)	8192
conv5_block3_add	(None, 7, 7, 2048)	0
conv5_block3_out	(None, 7, 7, 2048)	0
Average_pooling2d_1	(None, 1, 1, 2048)	0
flatten	(None, 2048)	0
Dense_3	(None, 256)	524544
Dropout_2	(None, 256)	0
Dense_4	(None, 128)	32896
Dropout_3	(None, 128)	0
Dense_5	(None, 9)	1161
Model parametres Summary		
Total params; 24,146,313		
Trainable params: 24,146, 313		
Non-trainable params: 0		

Table 4: Inceptionv3 Model Summary.

Layer name	Output shape	Number of parameters
Concatenate_3	(None, 5, 5, 2048)	0
Activation_187	(None, 5, 5, 2048)	0
Mixed10	(None, 5, 5, 2048)	0
Average_pooling2d_19	(None, 1, 1, 2048)	0
flatten	(None, 2048)	0
Dense_3	(None, 256)	524544
Dropout_2	(None, 256)	0
Dense_4	(None, 128)	32896
Dropout_3	(None, 128)	0
Dense_5	(None, 9)	1161
Model parametres Summary		
Total params; 22,361,385		
Trainable params: 22,361, 385		
Non-trainable params: 0		

Table 5: InceptionResNetV2 Model summary.

Layer name	Output shape	Number of parameters
Conv_7b	(None, 5, 5, 1536)	3194880
Conv_7b_bn	(None, 5, 5, 1536)	4608
Conv_7b_ac	(None, 5, 5, 1536)	0
Average_pooling2d_19	(None, 1, 1, 1536)	0
flatten	(None, 1536)	0
Dense_27	(None, 256)	393472
Dropout_18	(None, 256)	0
Dense_28	(None, 128)	32896
Dropout_19	(None, 128)	0
Dense_29	(None, 9)	1161
Model parametres Summary		
Total params; 54,764,256		
Trainable params: 54,764, 265		
Non-trainable params: 0		

Table 6: VGG16 Model summary.

Layer name	Output shape	Number of parameters
Block5_ conv2	(None, 14, 14, 512)	3194880
Block_conv3	(None, 14, 14, 512)	4608
Block5_pool	(None, 7, 7, 512)	0
Average_pooling2d_1	(None, 1, 1, 512)	0
flatten	(None, 512)	0
Dense_3	(None, 256)	393472
Dropout_2	(None, 256)	0
Dense_4	(None, 128)	32896
Dropout_3	(None, 128)	0
Dense_5	(None, 9)	1161
Model parametres Summary		
Total params; 14,880,073		
Trainable params: 14,880, 073		
Non-trainable params: 0		

4.2 Enhanced Convolutional Neural Network Model for Translating Kenyan Sign Language into Text in English

Image enhancement was used as a preprocessing step before images were fed to the proposed Convolutional Neural Network model. It minimizes background illumination so the model can focus on the essential features. This study used AHE and CLAHE to enhance the images.

Figures 10, 11, and 12 show images without any enhancement, with AHE enhancement, and with CLAHE enhancement, respectively. Note that the input images were adjusted to 224 × 224 pixels, the general size for most CNN models, and reduced to RGB from whatever original format.

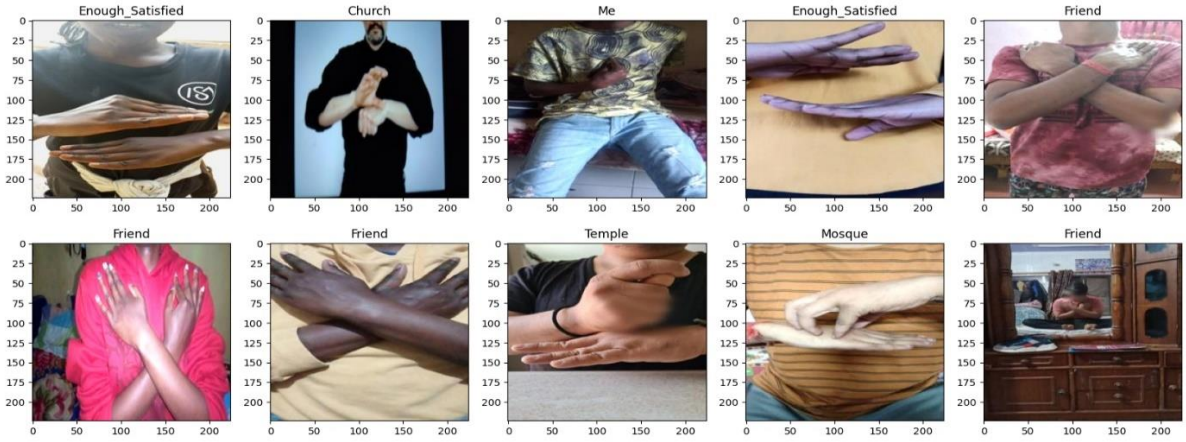

Figure 10: Images without enhancement.

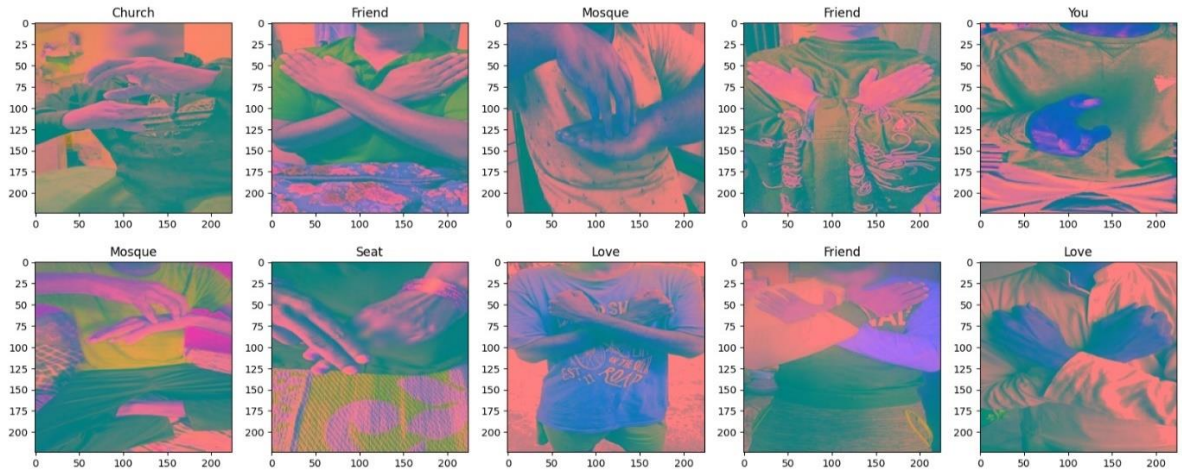

Figure 11: Images with AHE enhancement.

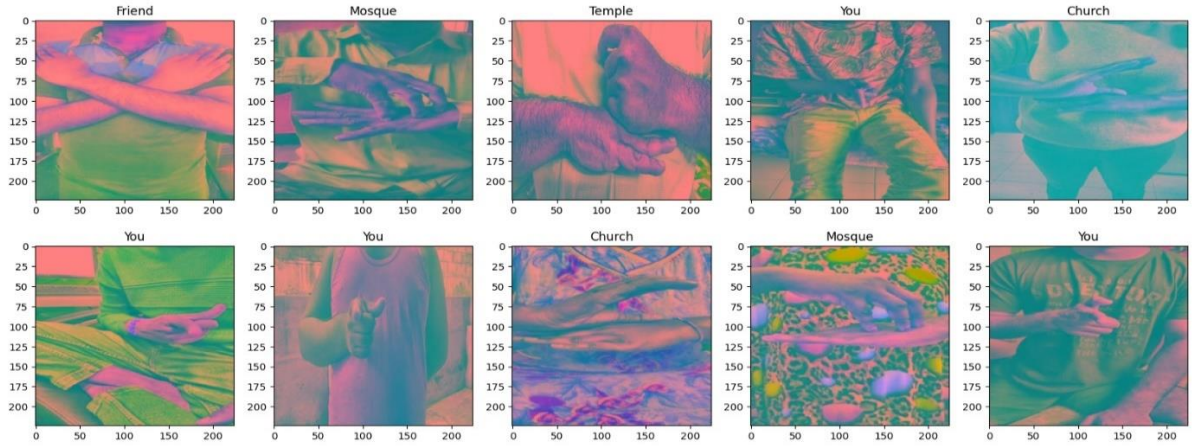

Figure 12: Images with CLAHE enhancement.

4.2.1 Model Training and Validation Loss

These are some of the most critical indicators in order to verify the performance of a model during its training, with the sole purpose of evaluating how well it learns. Training loss is a measure of the error rate of a model over a set of used training data, while validation loss can be regarded as the performance of a model on an unseen validation dataset. Figures 13, 14, and 15 show the training-validation loss curves of DenseNet121 at different epochs when images with no enhancement AHE enhanced, and CLAHE enhancement, respectively. In all cases, the training and validation loss decreased as the number of epochs increased; this means the model is learning something valuable from the data to improve its performance.

Figure 15 shows when enhanced by CLAHE, it generalized to the unseen data better than if enhanced by AHE or not enhanced at all; this would prove that CLAHE helped the model focus more on the relevant features.

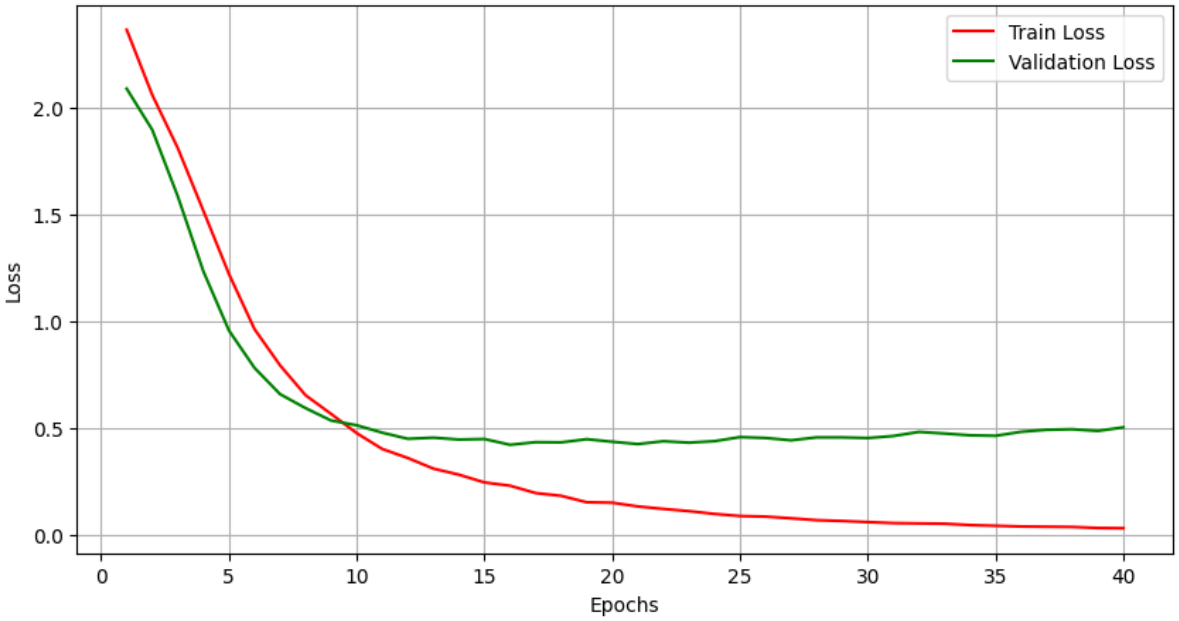

Figure 13: DenseNet121 without enhancement model training and validation loss.

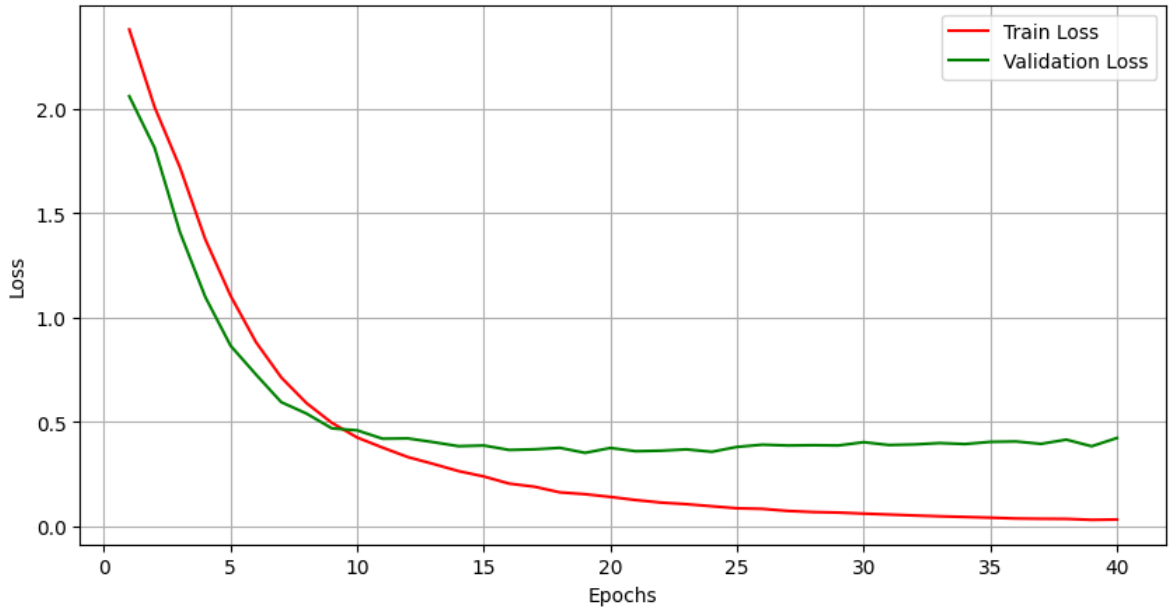

Figure 14: DenseNet121 with AHE enhancement model training and validation loss.

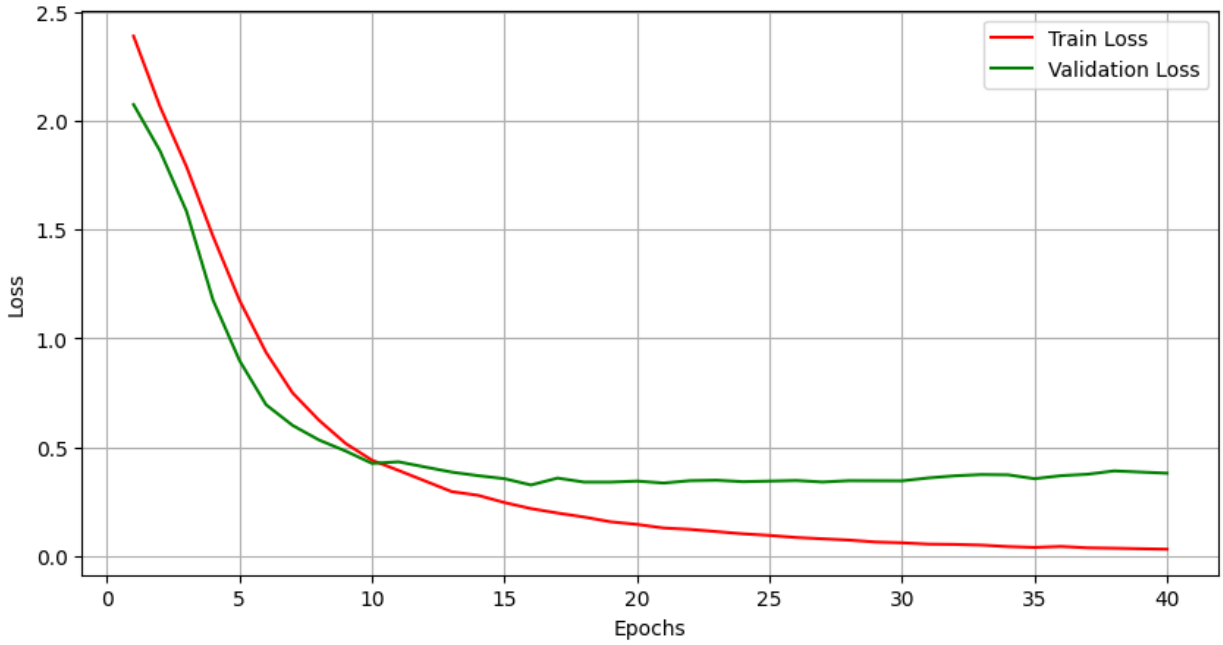

Figure 15: DenseNet121 with CLAHE enhancement model training and validation loss.

4.2.2 Model Training and Validation Accuracy

The Accuracy on the Training Dataset refers to the proportion of the correct predictions that the system makes on the training dataset, while the validation accuracy is the accuracy on the unseen validation dataset; therefore, these parameters can be considered some of the prime parameters for checking the performance of the model corresponding to machine learning during the training of the model.

In this regard, Figures 16, 17, and 18 depict the training and validation accuracy curves when the DenseNet121 model was trained without enhancement, with AHE CLAHE, respectively. The charts, in fact, reflect some improvements in both training and validation accuracy with respect to the number of epochs, which means there is consistency in learning and performance enhancement throughout. This indicated that the model managed to learn from the training data, and it improved its performance. However, Figure 18 illustrates that when CLAHE enhancement is applied, there's a better generalization performance compared to when AHE enhanced is used and when no enhancement is used.

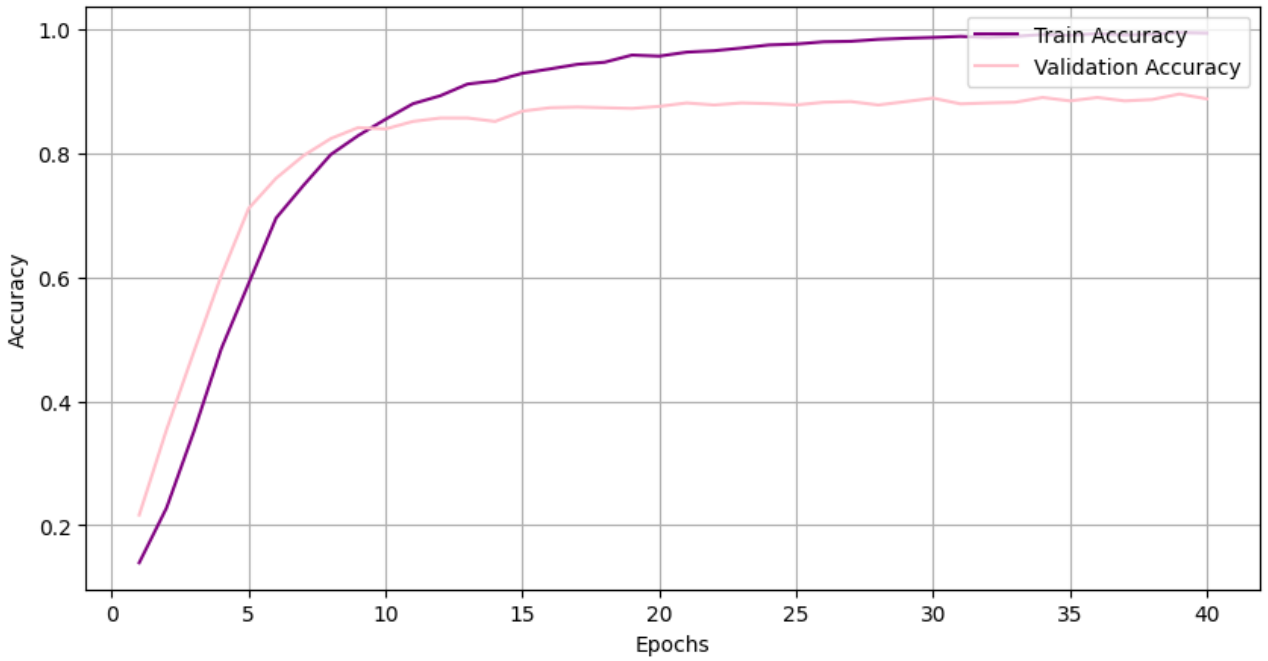

Figure 16: DenseNet121 without enhancement model training and validation accuracy.

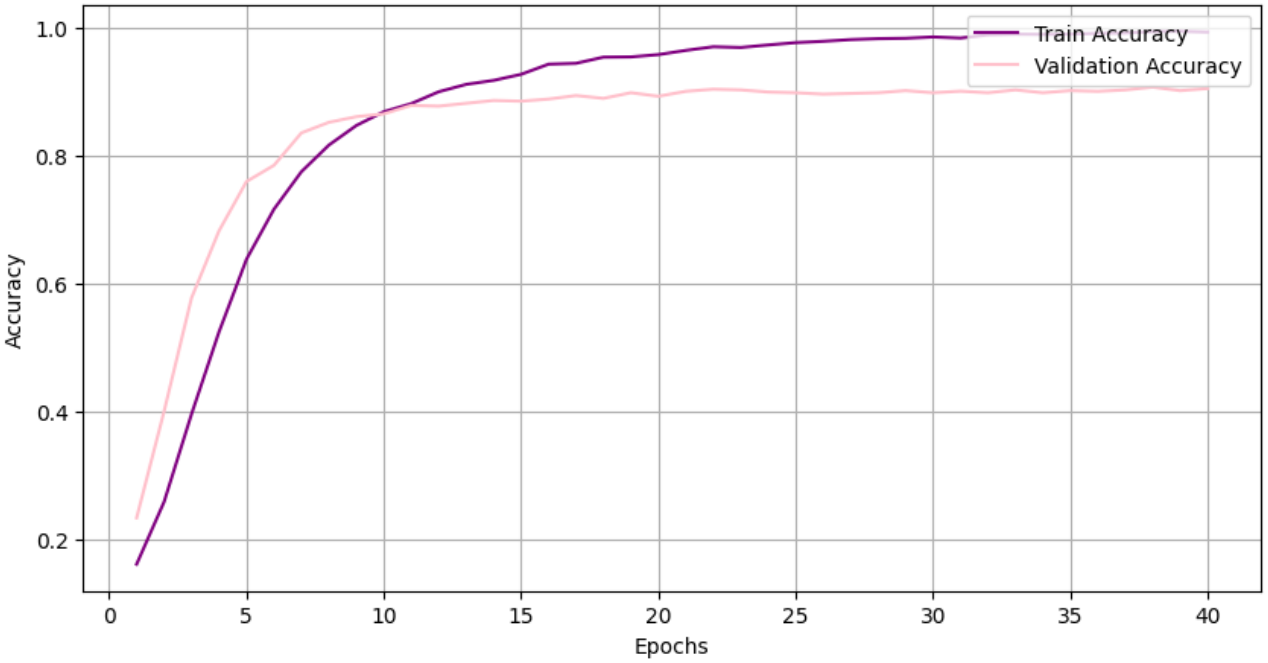

Figure 17: DenseNet121 with AHE enhancement model training and validation accuracy.

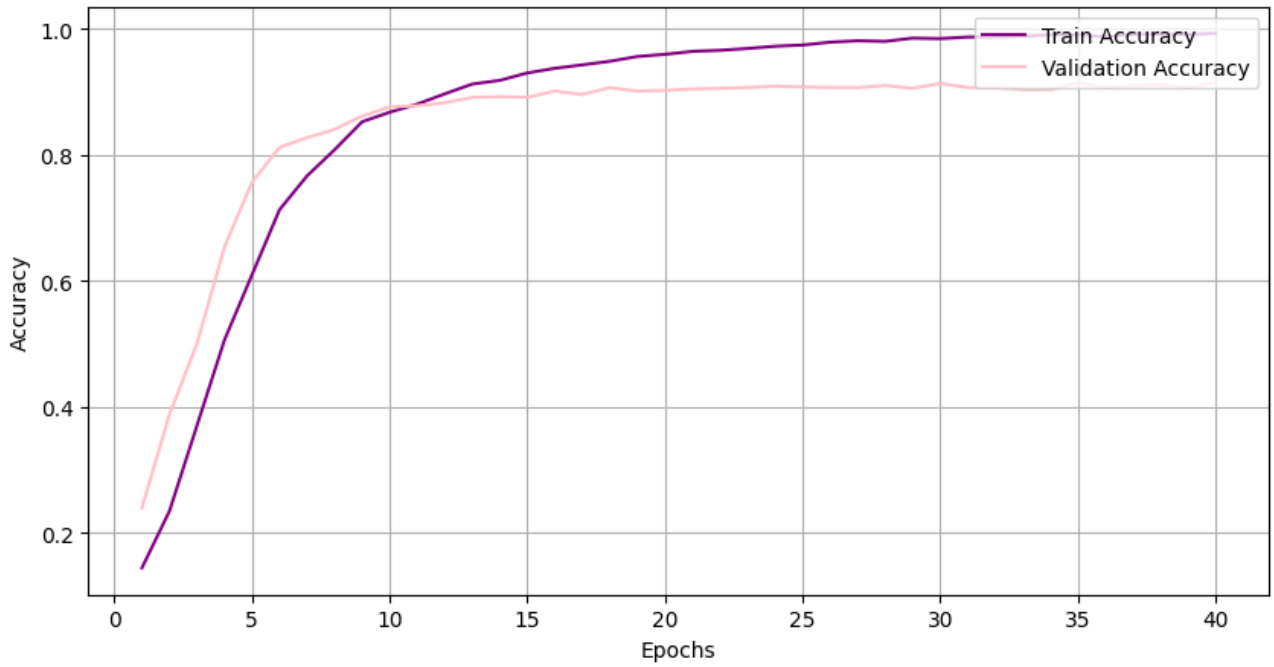

Figure 18: DenseNet121 with CLAHE enhancement model training and validation accuracy.

4.3 Model's Performance Results and Discussions

To evaluate DenseNet121's performance metrics, the researcher used the Classification Report, which handles the multi-class classification problem presented by an imbalanced dataset. A classification report summarizes model performance on all nine classes regarding precision, recall, F1-score, and support.

Recall determines the percentage of accurate forecasts for each class compared to all forecasts for that class, whereas precision measures the percentage of right predictions for each class compared to all actual instances of that class. As the consonant mean of precision and recall, the F1-score offers a fair measure that takes into account both completeness and accuracy in categorization performance. To provide information about how each class is represented during evaluation, support shows how many instances of each class are in the validation or test dataset.

Tables 7, 8, and 9 show the classification report for DenseNet121 without enhancement, with AHE and CLAHE, respectively. These figures indicate solid model performance, with accuracy scores between 0.8894 and 0.9154. Precision and recall were high across most classes in all figures, indicating effective classification. The model maintained consistent performance across all classes, including those with fewer samples. Notably, Table 9 highlights DenseNet121's robust performance on the CLAHE-enhanced Kenyan Sign Language dataset, with an overall accuracy of 0.9154, a macro average for precision, recall, and F1-score of 0.9154, 0.9136, and 0.9137, respectively, and a weighted average of precision, recall, and F1-score of 0.9154, 0.9136, and 0. 9136.

Table 7: DenseNet121 without enhancement model's classification report.

Class	Precision	Recall	F1-Score	Support
Church	0.8617	0.7788	0.8182	104
Enough - Satisfied	0.8785	0.8952	0.8868	105
Friend	0.9490	0.8942	0.9208	104
Love	0.8962	0.9235	0.9048	104
Me	0.8611	0.8942	0.8774	104
Mosque	0.8679	0.8846	0.8762	104
Seat	0.8889	0.9143	0.9014	105
Temple	0.8868	0.9038	0.8952	104
You	0.9143	0.9231	0.9187	104
Accuracy	0.8891			938
Macro Average	0.8894	0.8891	0.8888	938
Weighted Average	0.8894	0.8891	0.8888	938

Table 8: DenseNet121 with AHE enhancement model's classification report.

Class	Precision	Recall	F1-Score	Support
Church	0.8763	0.8173	0.8458	104
Enough - Satisfied	0.9216	0.8952	0.9082	105
Friend	0.9684	0.8846	0.9246	104
Love	0.8534	0.9519	0.9000	104
Me	0.9485	0.8846	0.9154	104
Mosque	0.8991	0.9423	0.9202	104
Seat	0.9612	0.9429	0.9519	105
Temple	0.8762	0.8846	0.8804	104
You	0.8509	0.9327	0.8899	104
Accuracy	0.9041			938
Macro Average	0.9062	0.9040	0.9040	938
Weighted Average	0.9062	0.9041	0.9041	938

Table 9: DenseNet121 with CLAHE enhancement model's classification report.

Class	Precision	Recall	F1-Score	Support
Church	0.9388	0.8846	0.9109	104
Enough - Satisfied	0.9109	0.8762	0.8932	105
Friend	0.9381	0.8750	0.9055	104
Love	0.8860	0.9712	0.9266	104
Me	0.8919	0.9519	0.9209	104
Mosque	0.9524	0.9615	0.9569	104
Seat	0.9600	0.9143	0.9366	105
Temple	0.8421	0.9231	0.8807	104
You	0.9184	0.8654	0.8911	104
Accuracy	0.9136			938
Macro Average	0.9154	0.9137	0.9136	938
Weighted Average	0.9154	0.9136	0.9136	938

4.4 Comparison of DenseNet121 and other Convolutional Neural Network Models in Translating Kenyan Sign Language.

Translating Kenyan Sign Language into English text using Convolutional Neural Networks (CNNs) is challenging because of the entangled nature of sign language gestures and the limited availability of labeled datasets for Kenyan Sign Language. However, experiments have demonstrated that CNNs hold significant potential in this field. This comparison examines the superior performance of DenseNet121 over other CNN architectures, including ResNet50, Inceptionv3, InceptionResNetV2, and VGG16, in translating Kenyan Sign Language gestures into English text.

The models were trained on unenhanced input and enhanced images through AHE and CLAHE. Each model's performance was recorded and analyzed to evaluate its effectiveness in handling this translation task.

4.4.1 ResNet50 Model in Translating Kenyan Sign Language

One of the models trained and validated with the Kenyan Sign Language (KSL) dataset is ResNet50, which was chosen based on its excellent performance in image classification. This model's loss and accuracy are tracked during training and validation. Figures 19, 20, and 21 indicate the training and validation loss curves for ResNet50 when the network is trained using input images: raw, enhanced with AHE, and enhanced with CLAHE.

Above all, the training and validation loss curves in the three figures trend downwards; this says that every model learns something and improves with each epoch. Also, the apparent interval betwixt the training and validation loss curves shows that the model has to deal with a potential problem of effectively generalizing to new data.

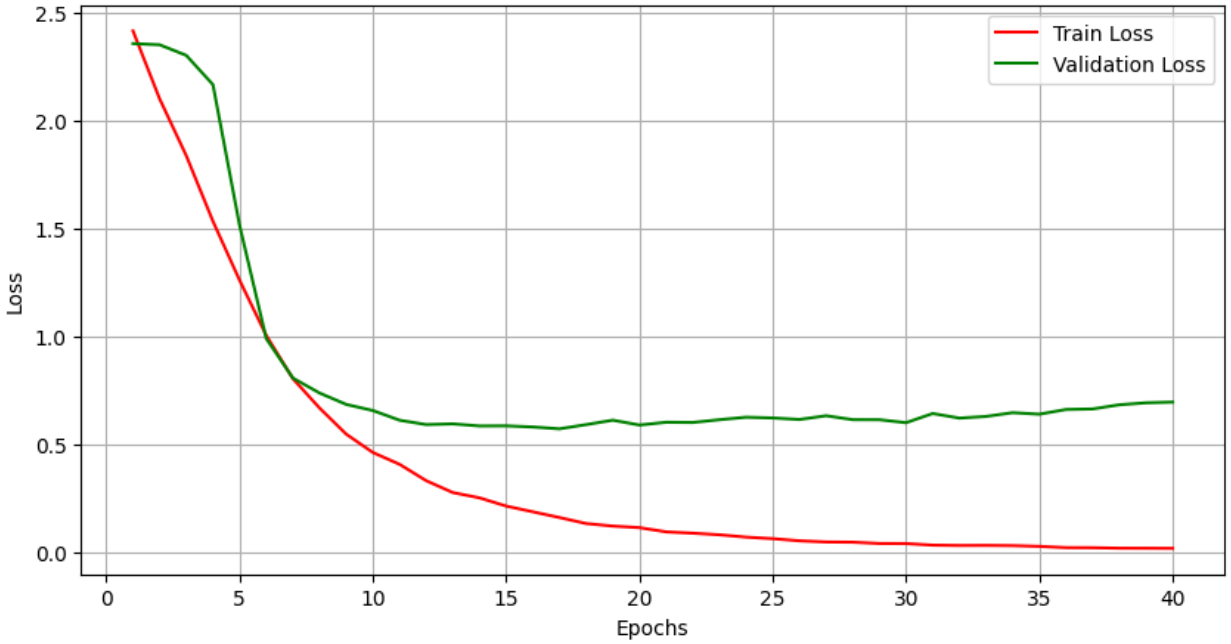

Figure 19: ResNet50 without enhancement model training and validation loss.

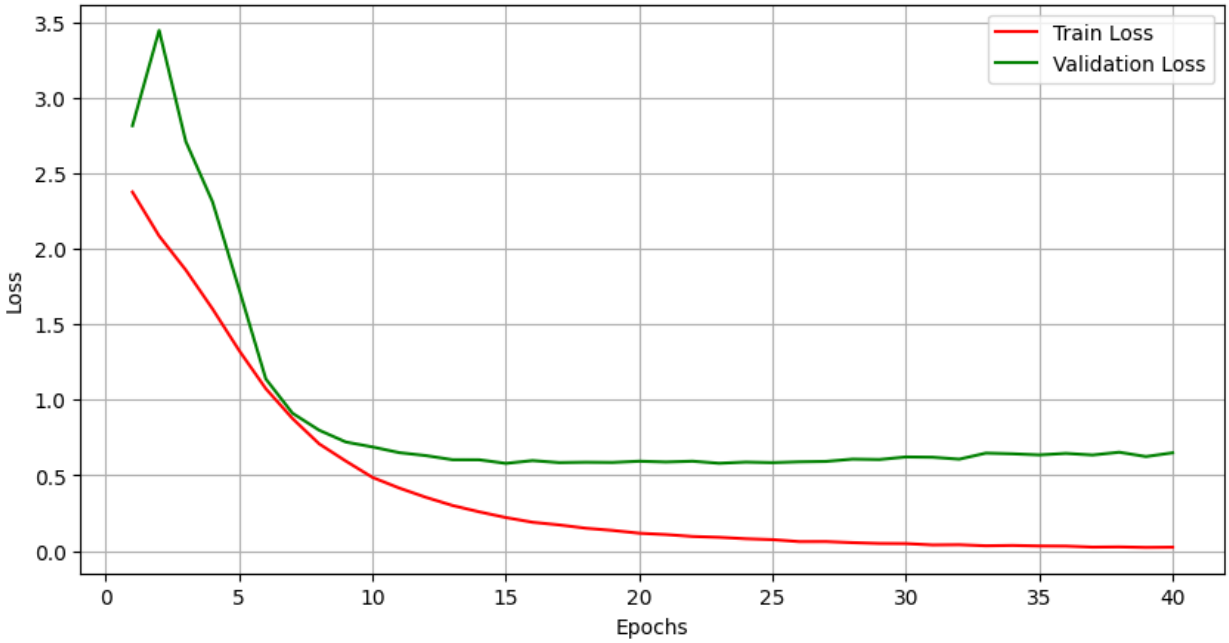

Figure 20: ResNet50 with AHE enhancement model training and validation loss.

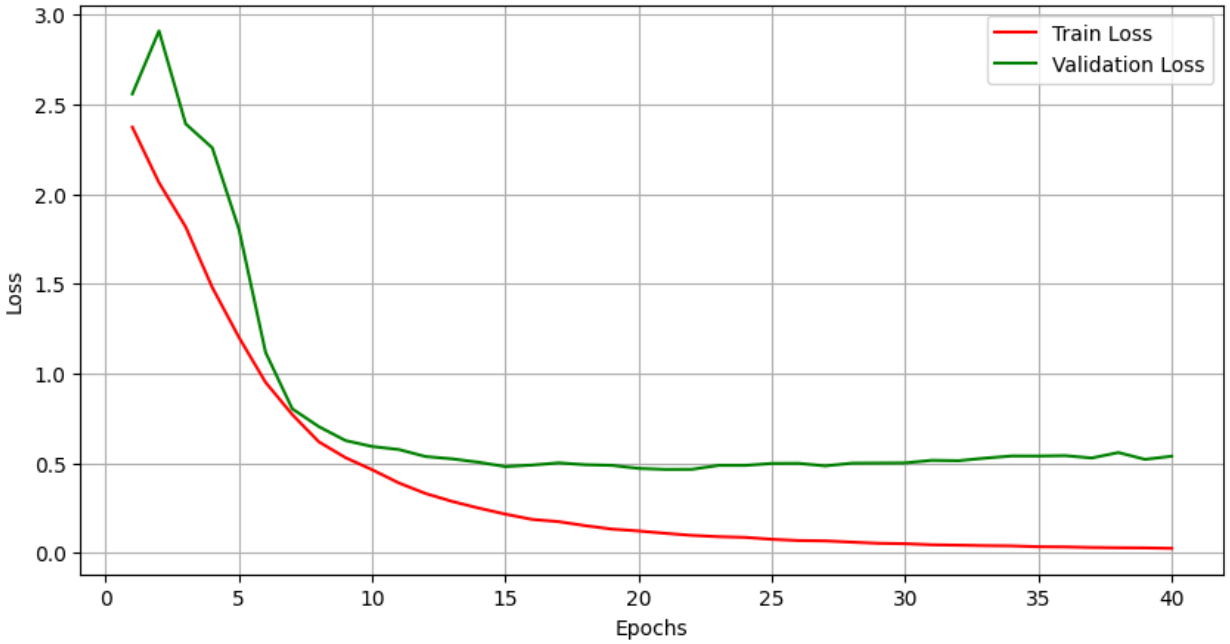

Figure 21: ResNet50 with CLAHE enhancement model training and validation loss.

The ResNet50 model's training and validation accuracy curves are displayed in Figures 22, 23, and 24, respectively, for three different training scenarios: no enhancement, AHE, and CLAHE. At first, the training and validation accuracy increased rapidly, indicating that the model successfully picked up basic patterns in the data. However, the validation accuracy plateaued at 0.84 for Figures 22 and 23. In contrast, Figure 24, which includes CLAHE enhancement, demonstrated better performance, achieving a validation accuracy of 0.87, suggesting that CLAHE helped the model generalize more effectively.

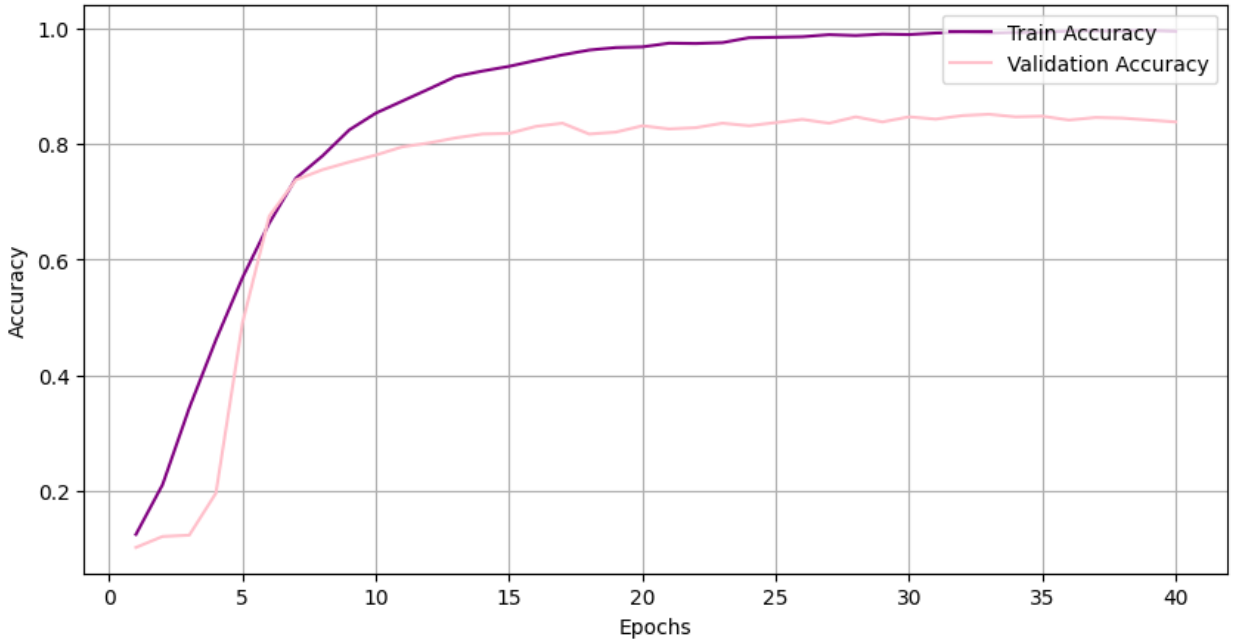

Figure 22: ResNet50 without enhancement model training and validation accuracy.

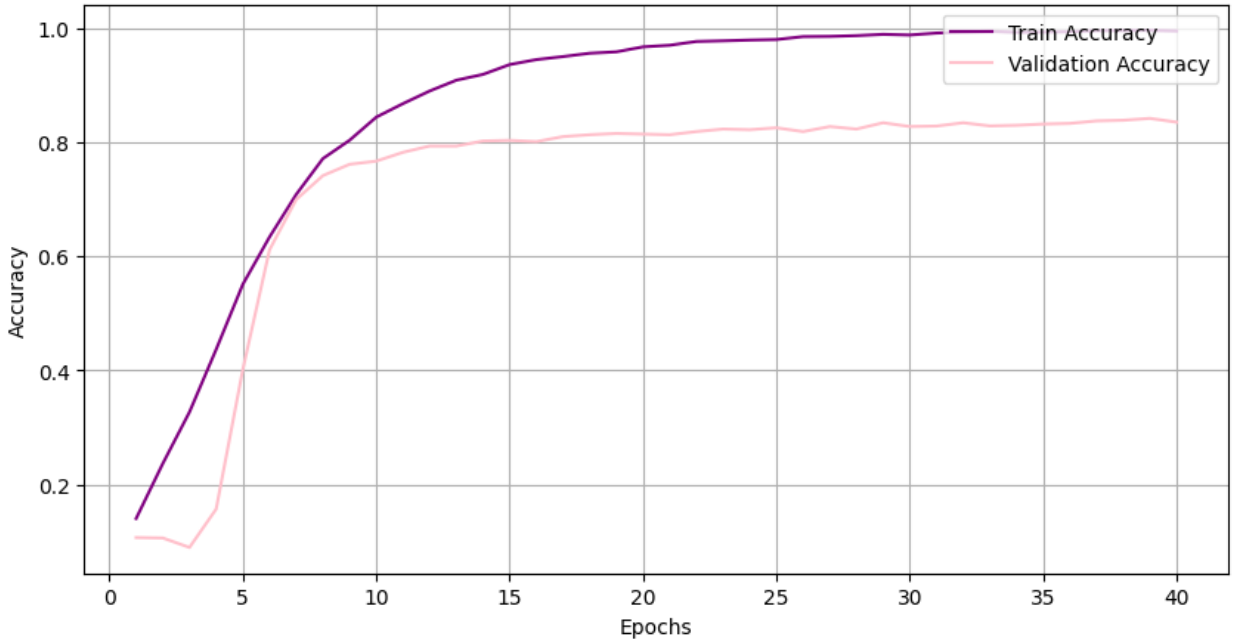

Figure 23: ResNet50 with AHE enhancement model training and validation accuracy.

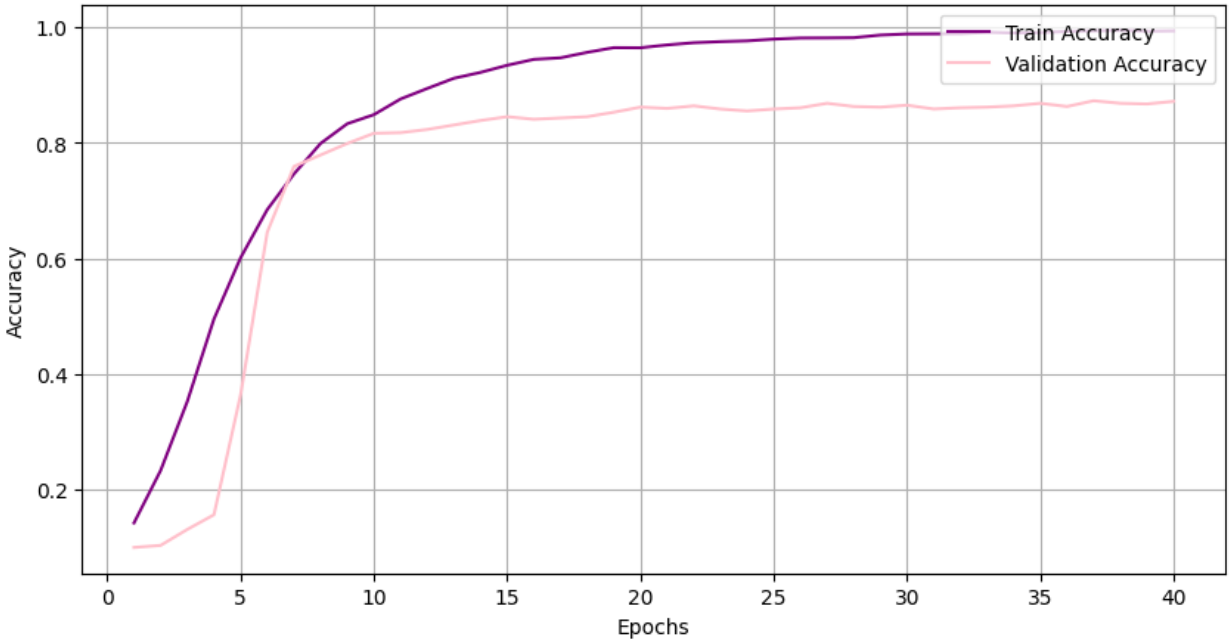

Figure 24: ResNet50 with CLAHE enhancement model training and validation.

4.4.2 InceptionV3 Model in Translating Kenyan Sign Language

InceptionV3 has inception modules that enable the model to learn multi-scale features using different convolutional filter sizes in parallel within the same layer. This architecture was chosen because it allows increased width and depth without significantly increasing computational complexity. Figures 25, 26, and 27 illustrate the training and validation loss curves for the InceptionV3 model throughout the training process, using images without enhancement, AHE, and CLAHE, respectively. These curves provide insight into the model's performance across different image preprocessing techniques.

As shown in the figures, InceptionV3 performed poorly on our Kenyan Sign Language dataset compared to the other models. An enormous disparity between the training and validation curves indicates that the model did not generalize well to unknown data.

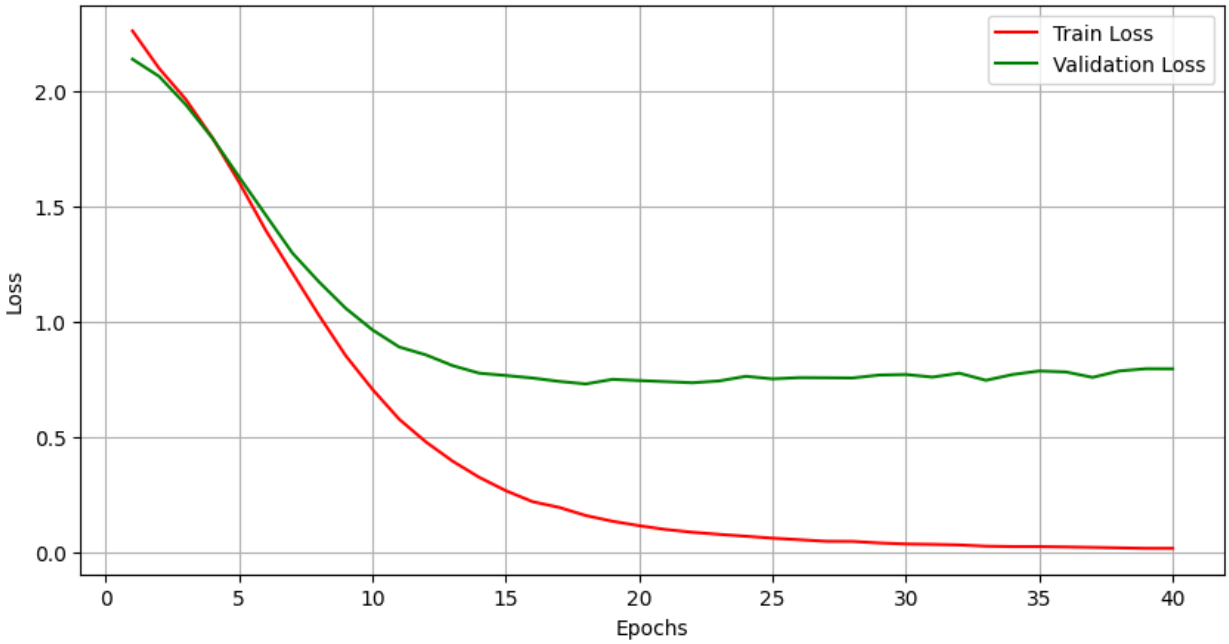

Figure 25: InceptionV3 without enhancement model training and validation loss.

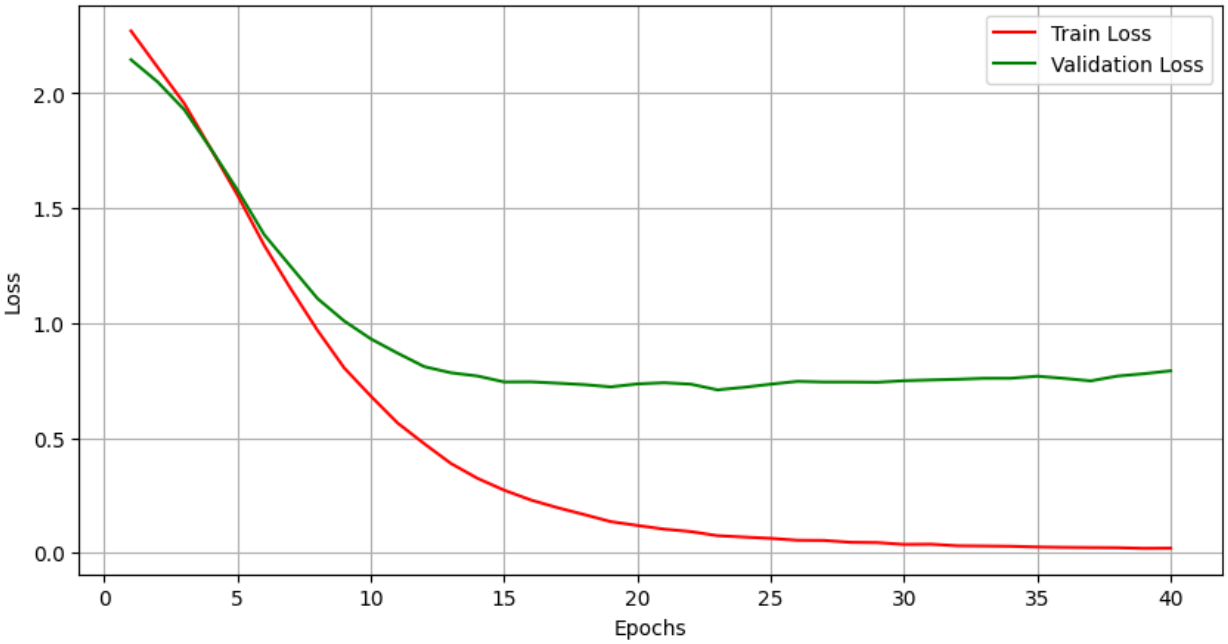

Figure 26: InceptionV3 with AHE enhancement model training and validation loss.

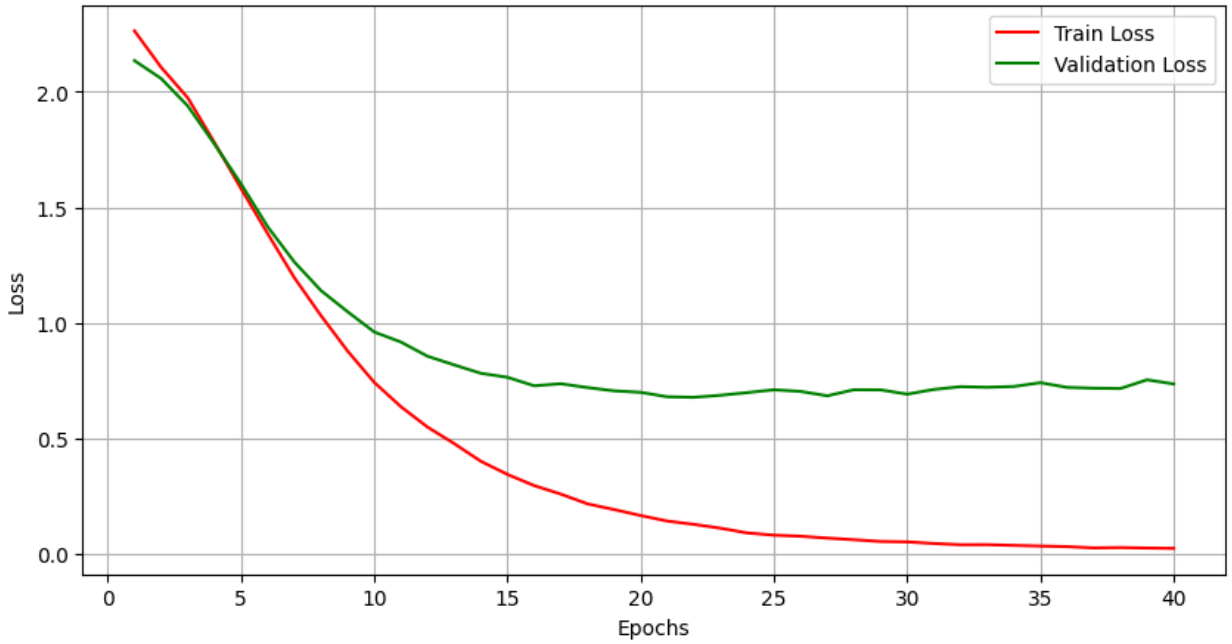

Figure 27: InceptionV3 without enhancement model training and validation loss.

Figures 28, 29, and 30 illustrate the training and validation accuracy curves for InceptionV3 during training without enhancement, with AHE, and with CLAHE, respectively.

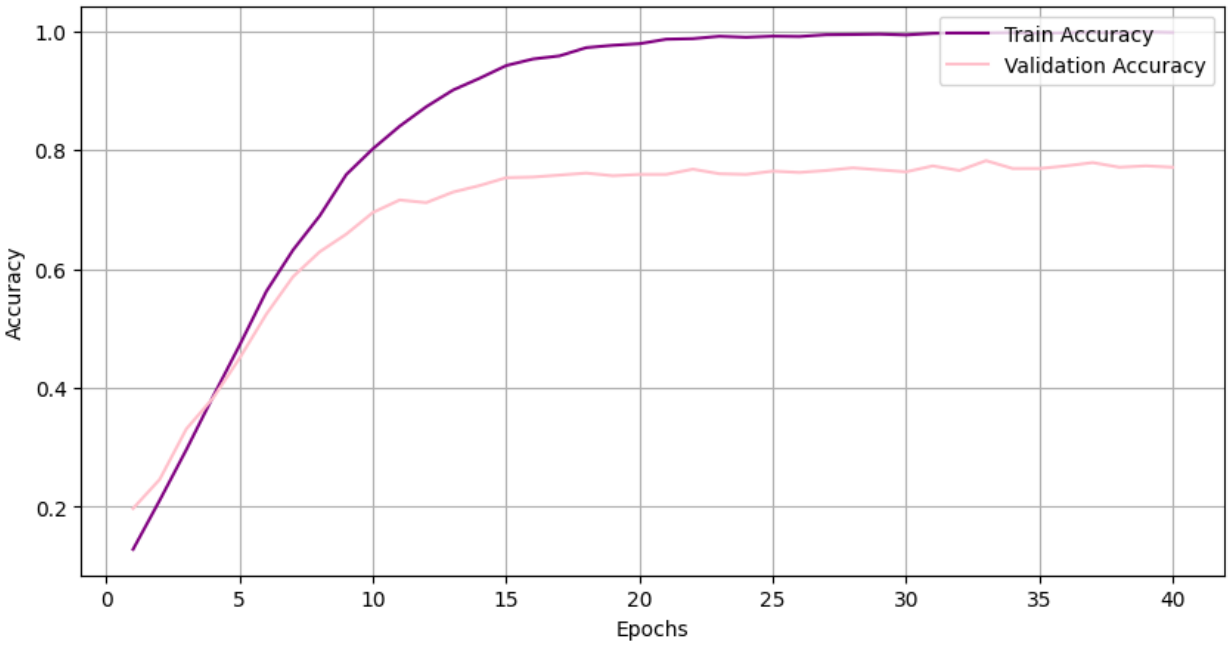

Figure 28: InceptionV3 without enhancement model training and validation accuracy.

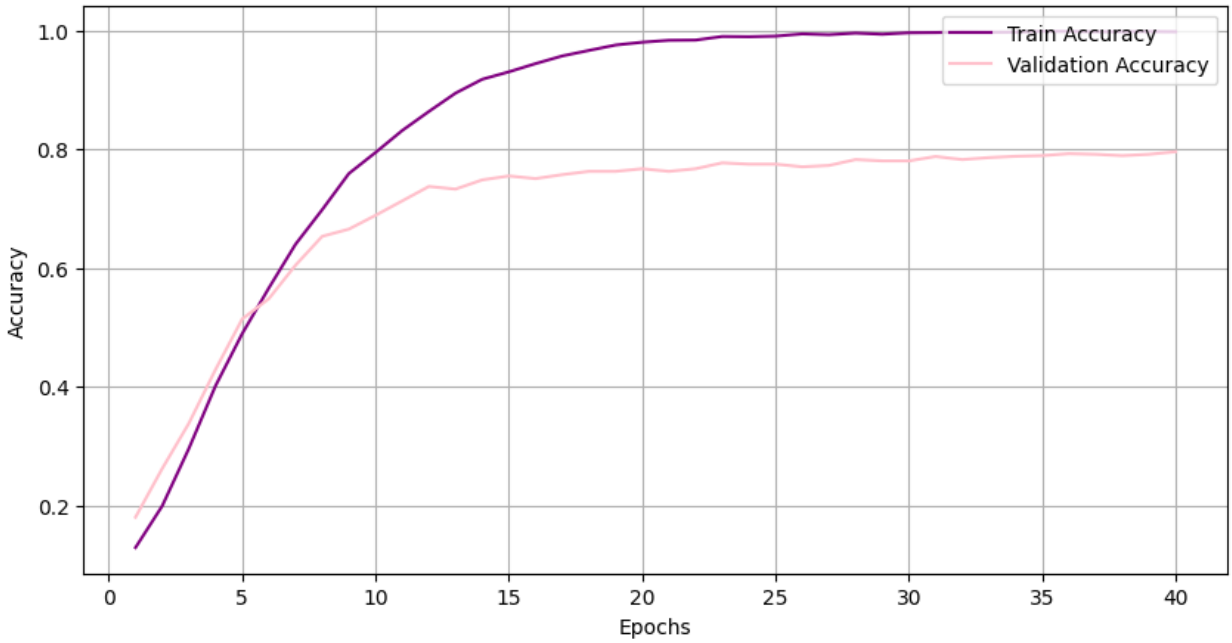

Figure 29: InceptionV3 with AHE enhancement model training and validation accuracy.

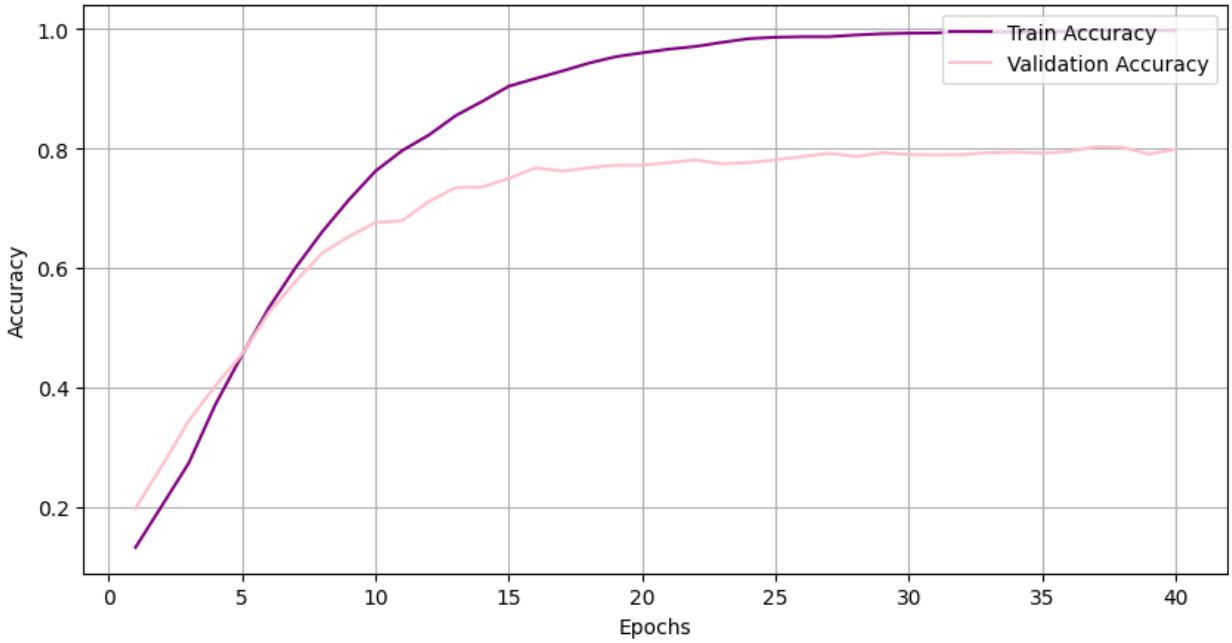

Figure 30: InceptionV3 with CLAHE enhancement model training and validation accuracy.

4.4.3 InceptionResNetV2 Model in Translating Kenyan Sign Language

InceptionResNetV2 is a hybrid model combining the strengths of both ResNet and Inception architectures. It was chosen because it merges residual connections with inception modules, enabling intense networks while maintaining computational efficiency. Figures 31, 32, and 33 show the training and validation loss curves for InceptionResNetV2 during the training process with images without enhancement, with AHE, and with CLAHE, respectively.

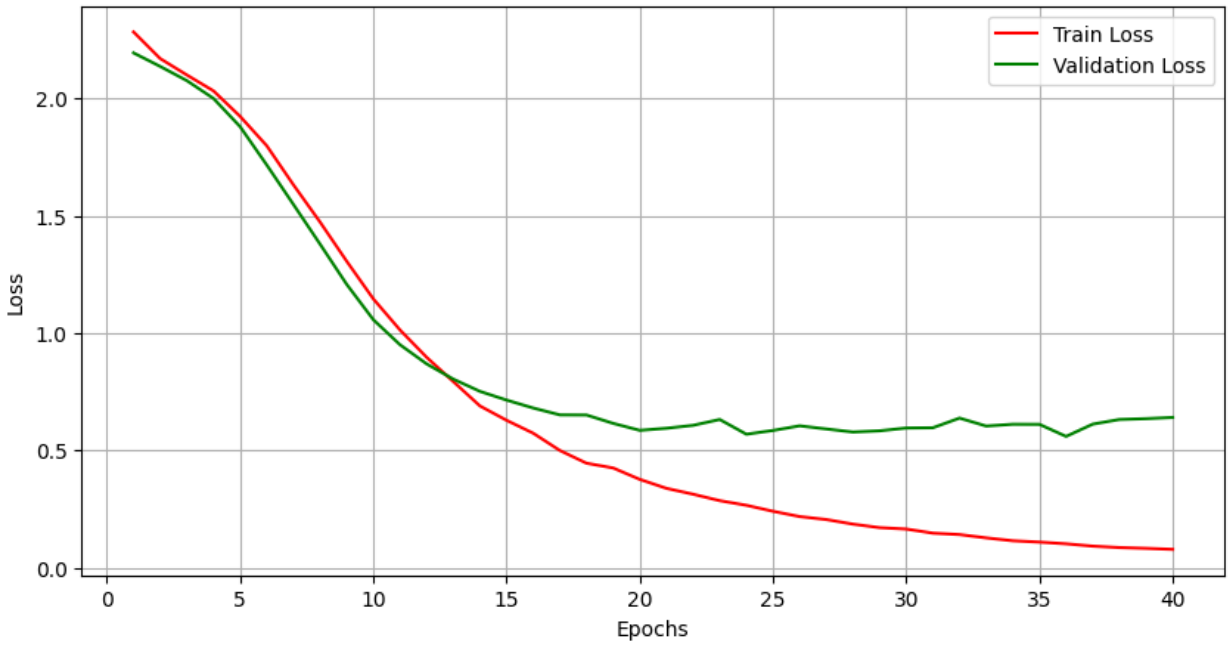

Figure 31: InceptionResNetV2 without enhancement model training and validation loss.

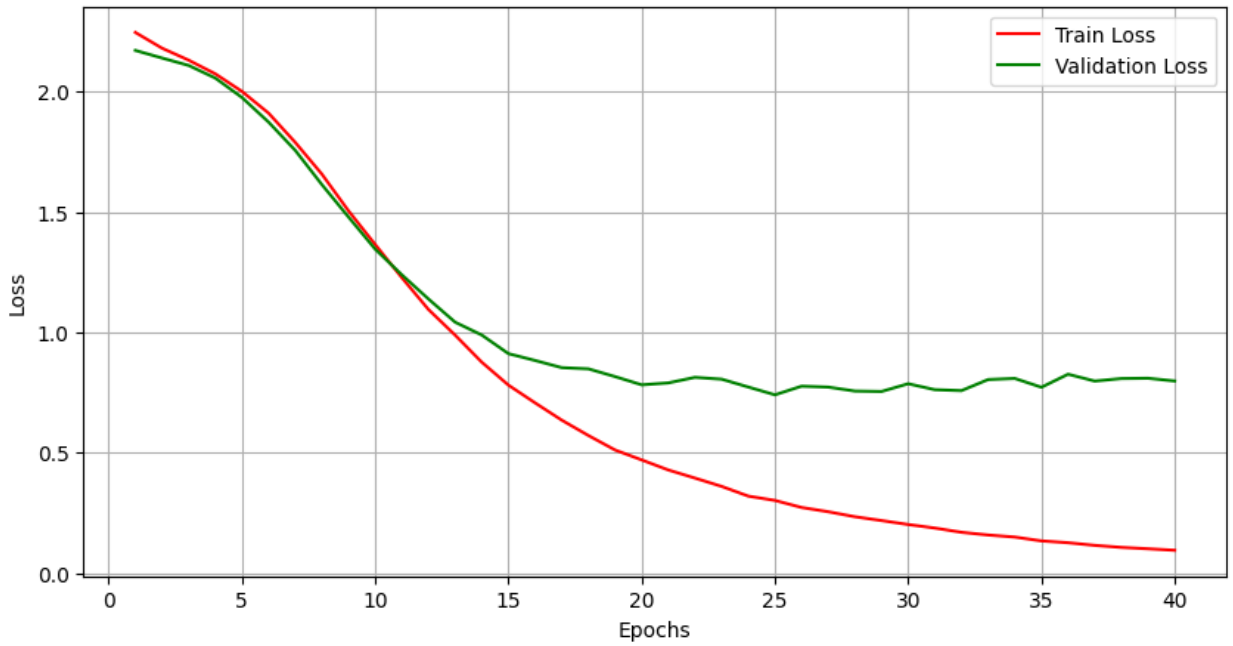

Figure 32: InceptionResNetV2 with AHE enhancement model training and validation loss.

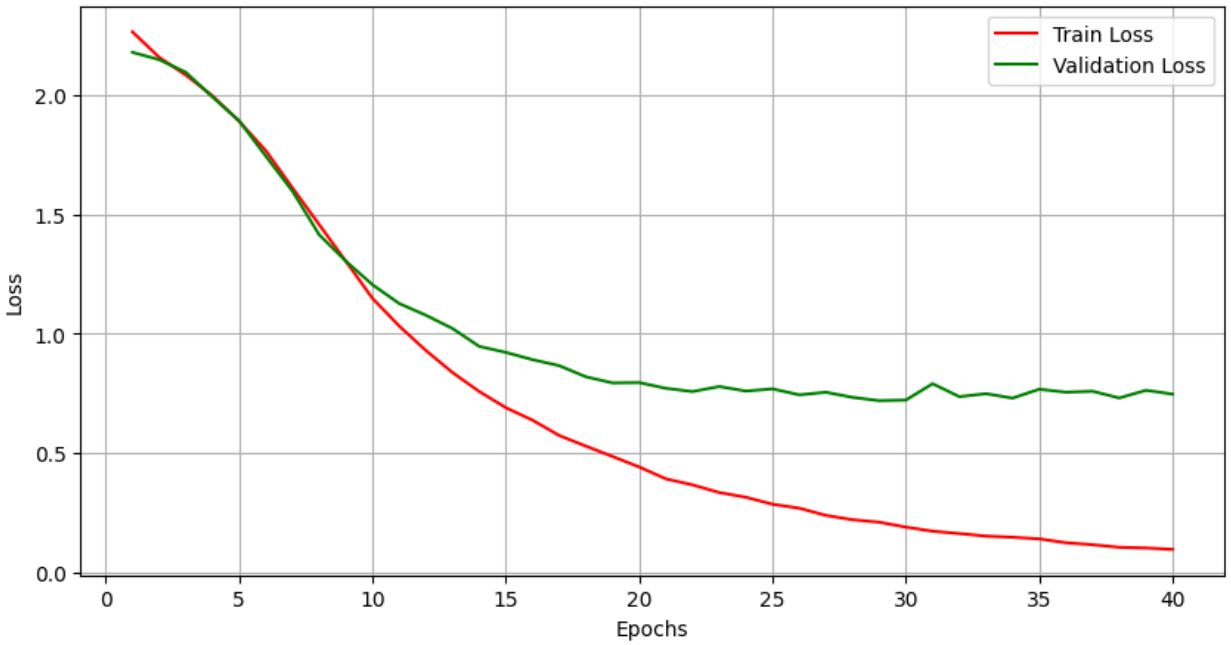

Figure 33: InceptionResNetV2 with CLAHE enhancement model training and validation loss.

Figures 34, 35, and 36 illustrate the training and validation accuracy curves for InceptionResNetV2 during training without enhancement, with AHE, and with CLAHE, respectively.

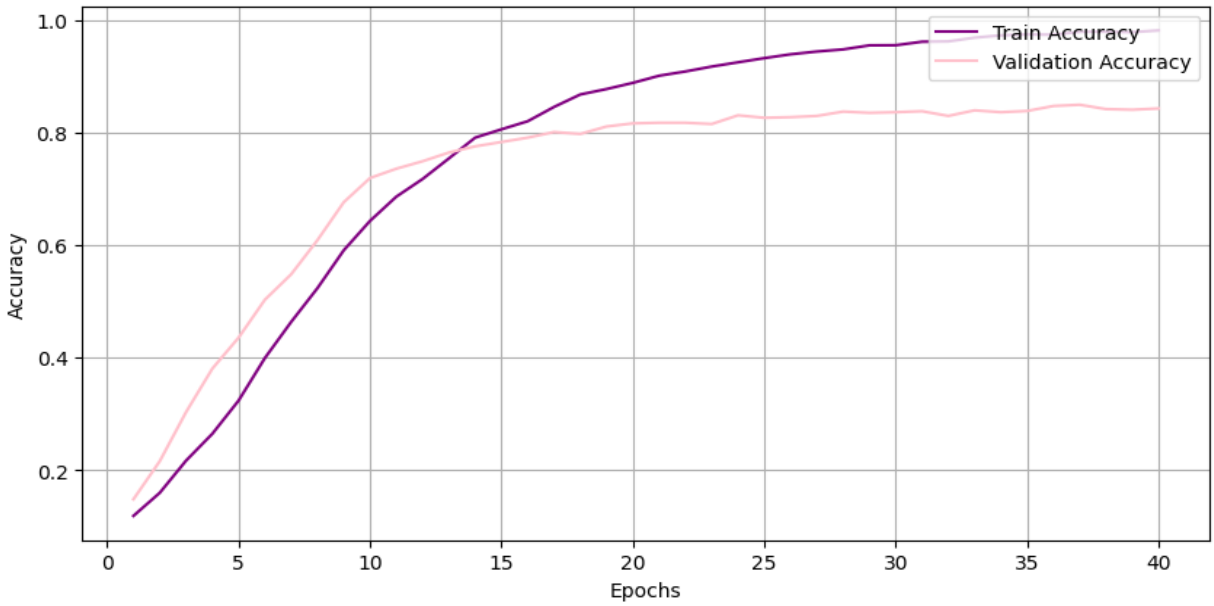

Figure 34: InceptionResNetV2 without enhancement model training and validation accuracy.

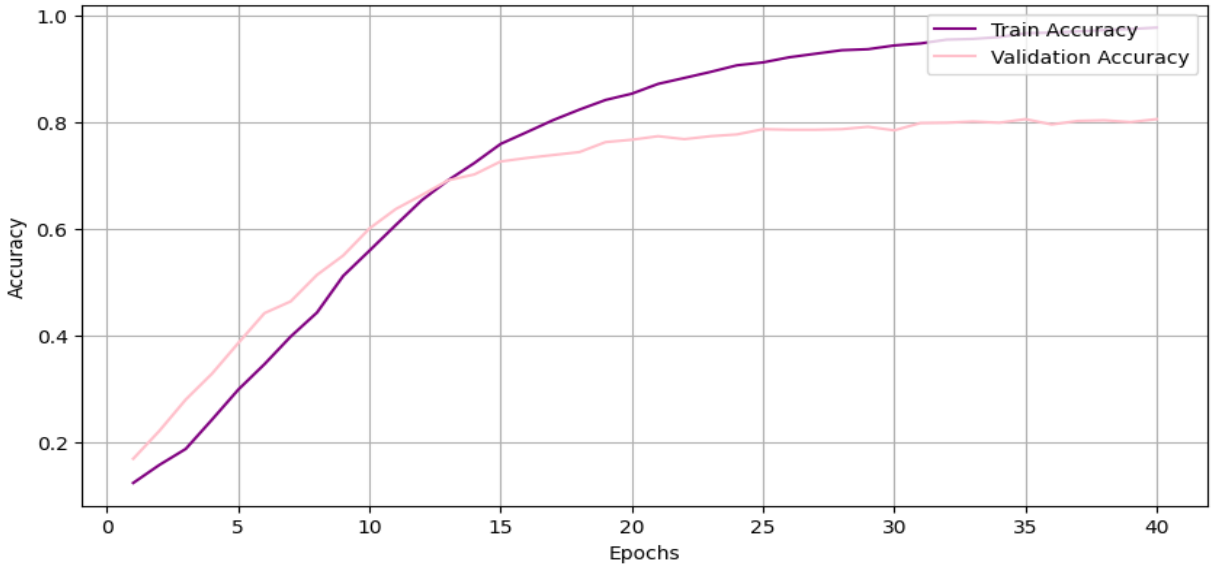

Figure 35: InceptionResNetV2 with AHE enhancement model training and validation accuracy.

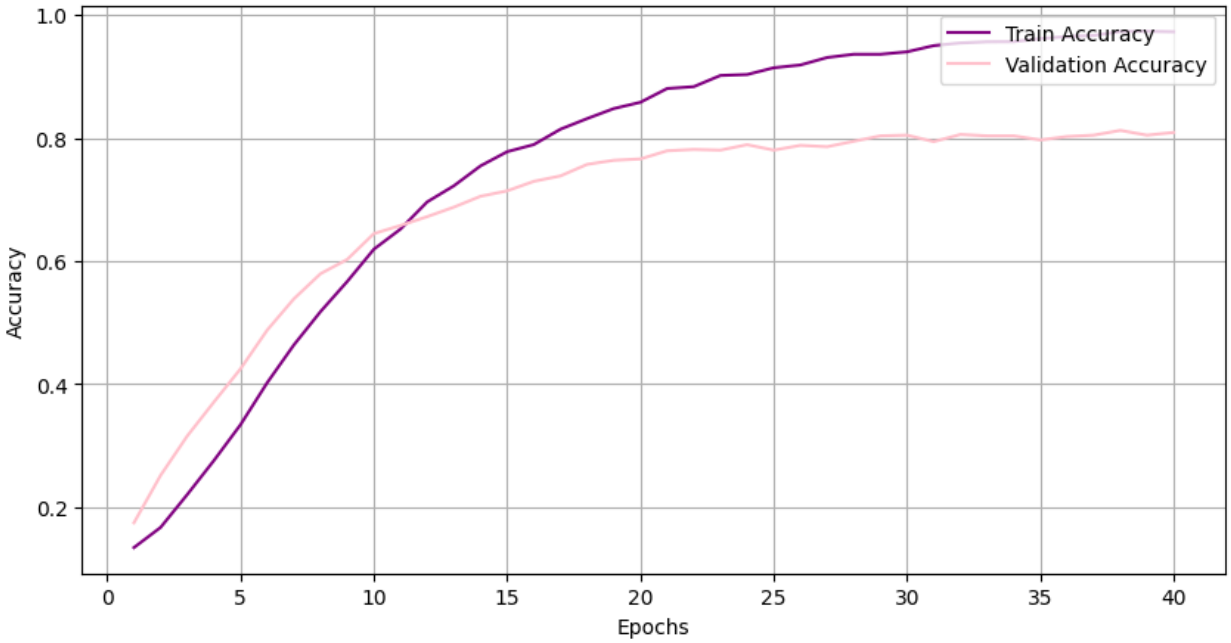

Figure 36: InceptionResNetV2 with CLAHE enhancement model training and validation accuracy.

4.4.4 VGG16 Model in Translating Kenyan Sign Language

VGG16 is a known Convolutional Neural Network Architecture for its depth and simplicity. It has been proven to produce high accuracy in image classification tasks. Figures 37, 38, and 39 show the training and validation loss curves for VGG16 during the training process with images without enhancement, with AHE enhancement, and with CLAHE enhancement, respectively.

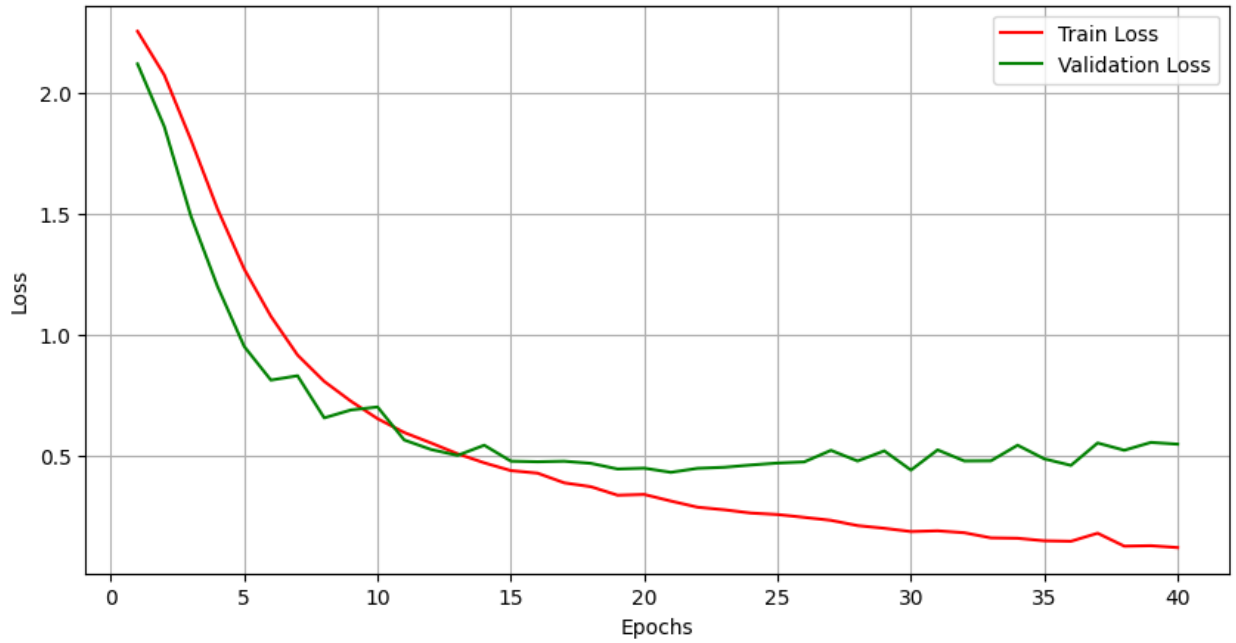

Figure 37: VGG16 without enhancement model training and validation loss.

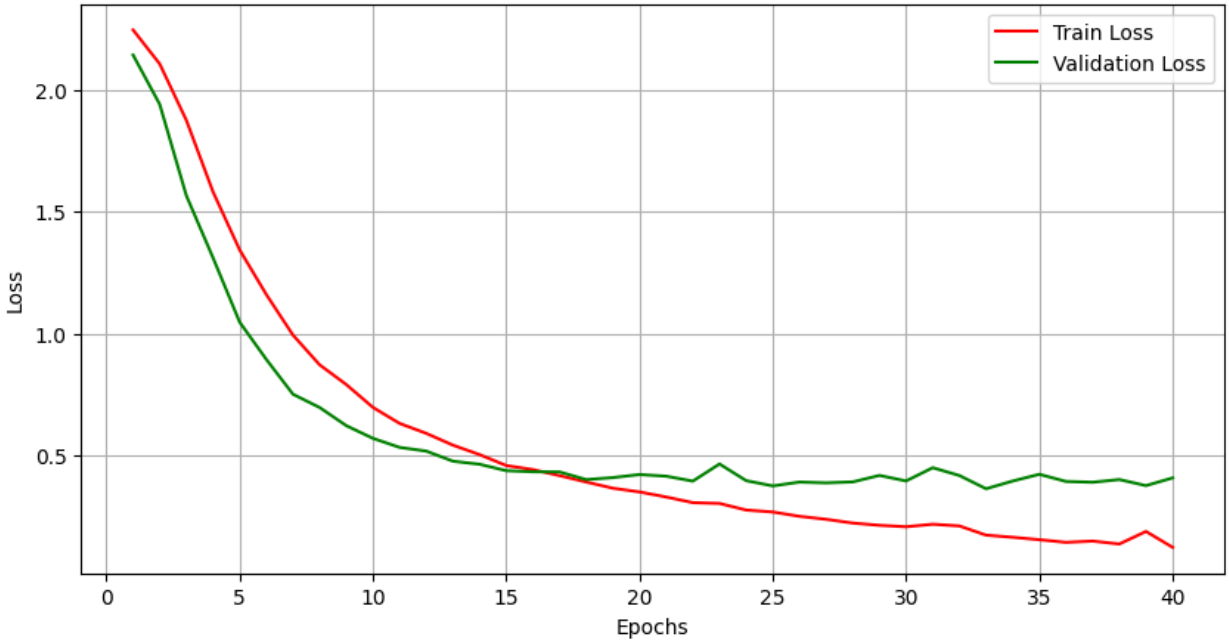

Figure 38: VGG16 with AHE enhancement model training and validation loss

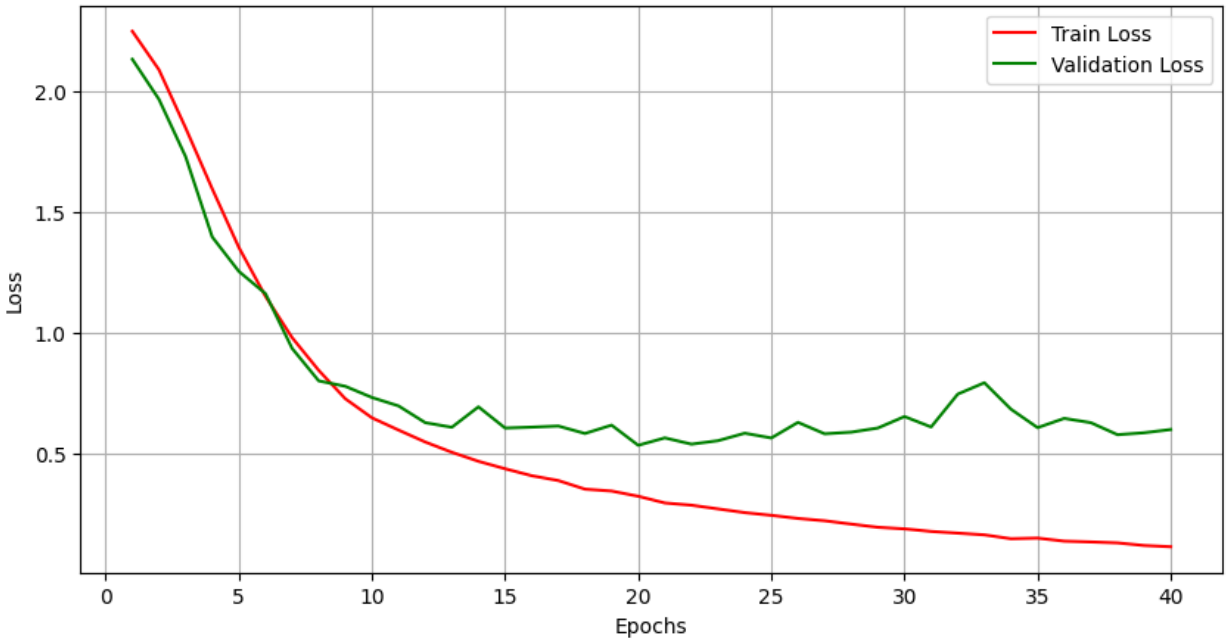

Figure 39: VGG16 with CLAHE enhancement model training and validation loss.

Figures 40, 41, and 42 illustrate the training and validation accuracy curves for VGG16 during training without enhancement, with AHE enhancement, and with CLAHE enhancement, respectively.

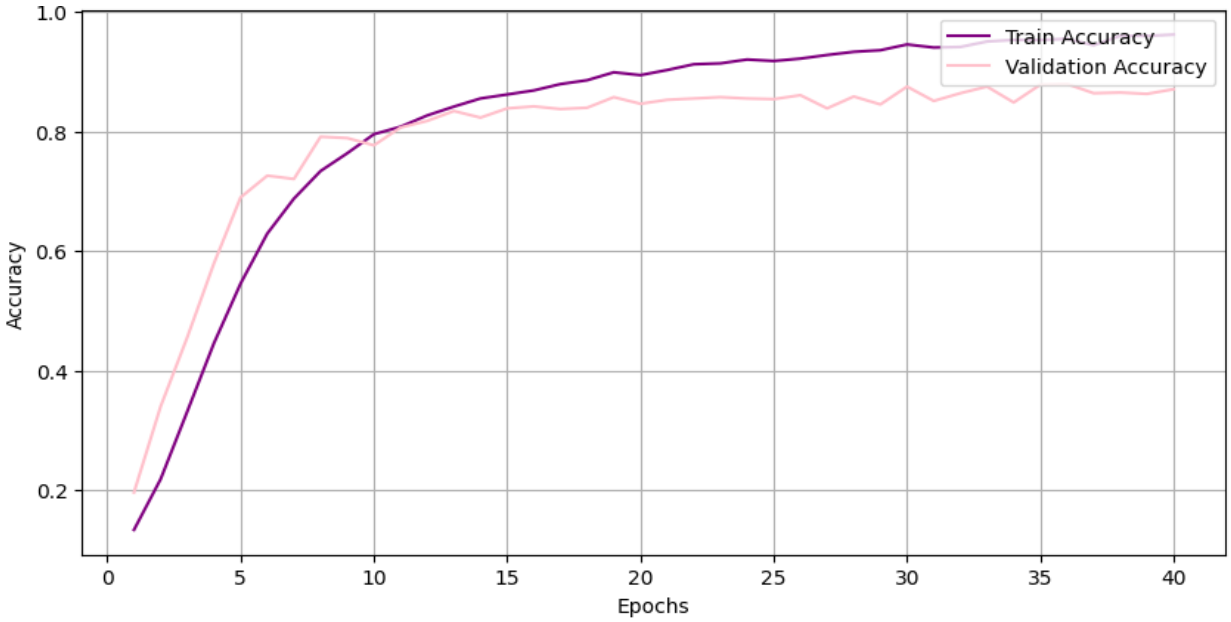

Figure 40:VGG16 without enhancement model training and validation accuracy.

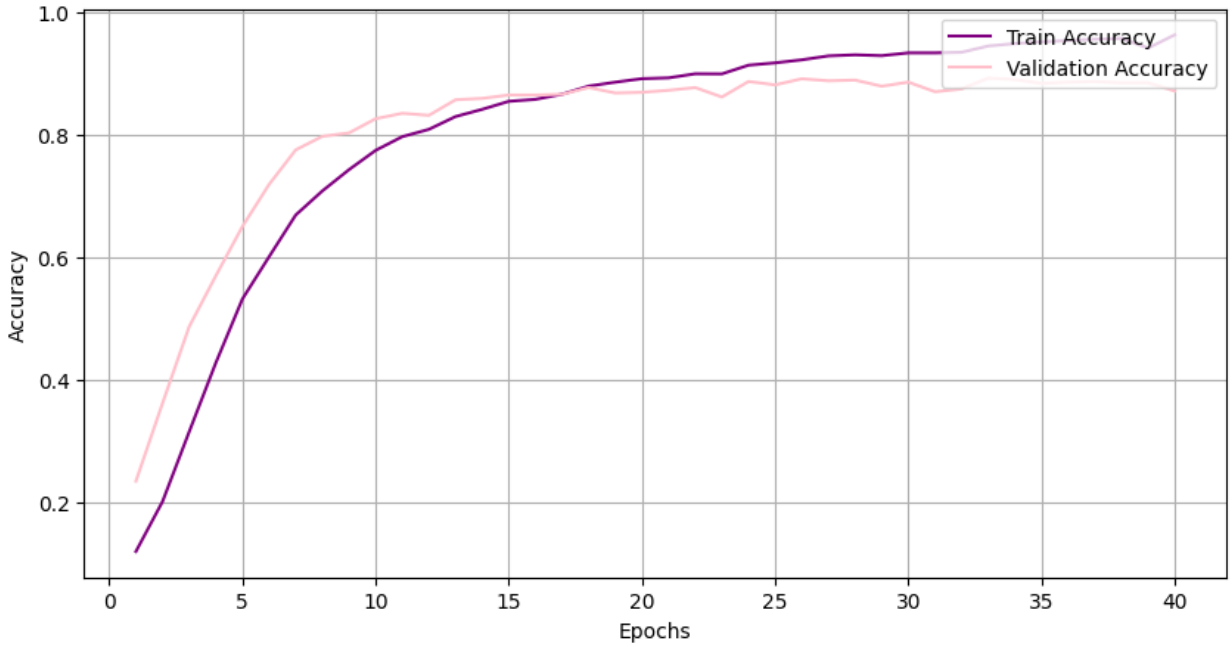

Figure 41: VGG16 with AHE enhancement model training and validation accuracy.

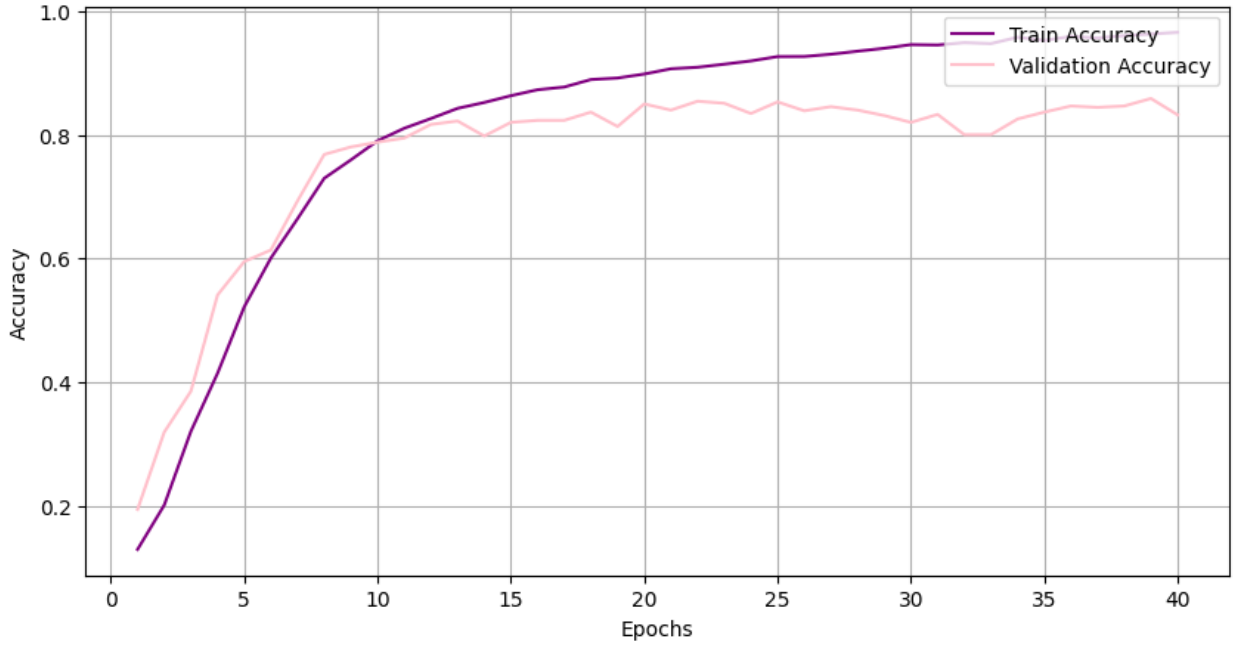

Figure 42: VGG16 with CLAHE enhancement model training and validation accuracy.

Table 10 compares the developed models based on accuracy and the macro average of precision, recall, and F1-score for all the models without enhancement, with AHE enhancement, and with CLAHE enhancement.

Table 10: Comparison of the developed models

Model	Accuracy			Macro Average		
	Without Enhancement	With AHE enhancement	With CLAHE enhancement	Without Enhancement	With AHE enhancement	With CLAHE enhancement
DenseNet121	0.8894	0.9062	0.9154	0.8888	0.9040	0.9136
ResNet50	0.8442	0.8401	0.8738	0.8430	0.8368	0.8722
Inceptionv3	0.7765	0.8022	0.8015	0.7745	0.8003	0.7987
InceptionResNetV2	0.8439	0.8120	0.8140	0.8413	0.8035	0.8079
VGG16	0.8775	0.8755	0.8415	0.8747	0.8715	0.8340

We used a clustered column chart to verify the performance of the models based on their accuracy and macro average on Kenyan Sign Language images without, with AHE enhancement and with CLAHE enhancement. Figure 43 compares the developed models based on the accuracy level. From the figure, it is evident that the impact of the enhancement varies across models, and the accuracy of the models generally improves, suggesting that the preprocessing was beneficial. DenseNet121 typically performed well across all enhancement techniques, favoring our study.

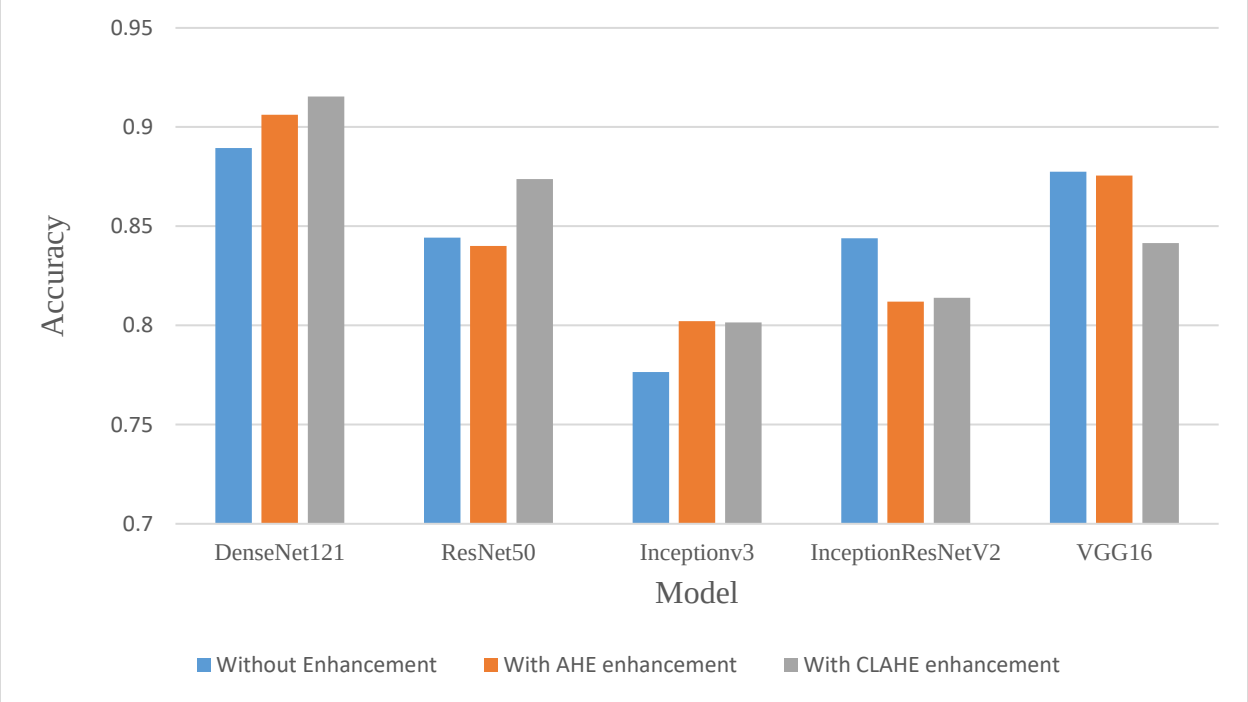

Figure 43: Comparison of accuracy of the developed models.

Figure 44 contrasts the developed models based on the macro average level. DenseNet121 and ResNet50 had greater macro average accuracy compared to other models.

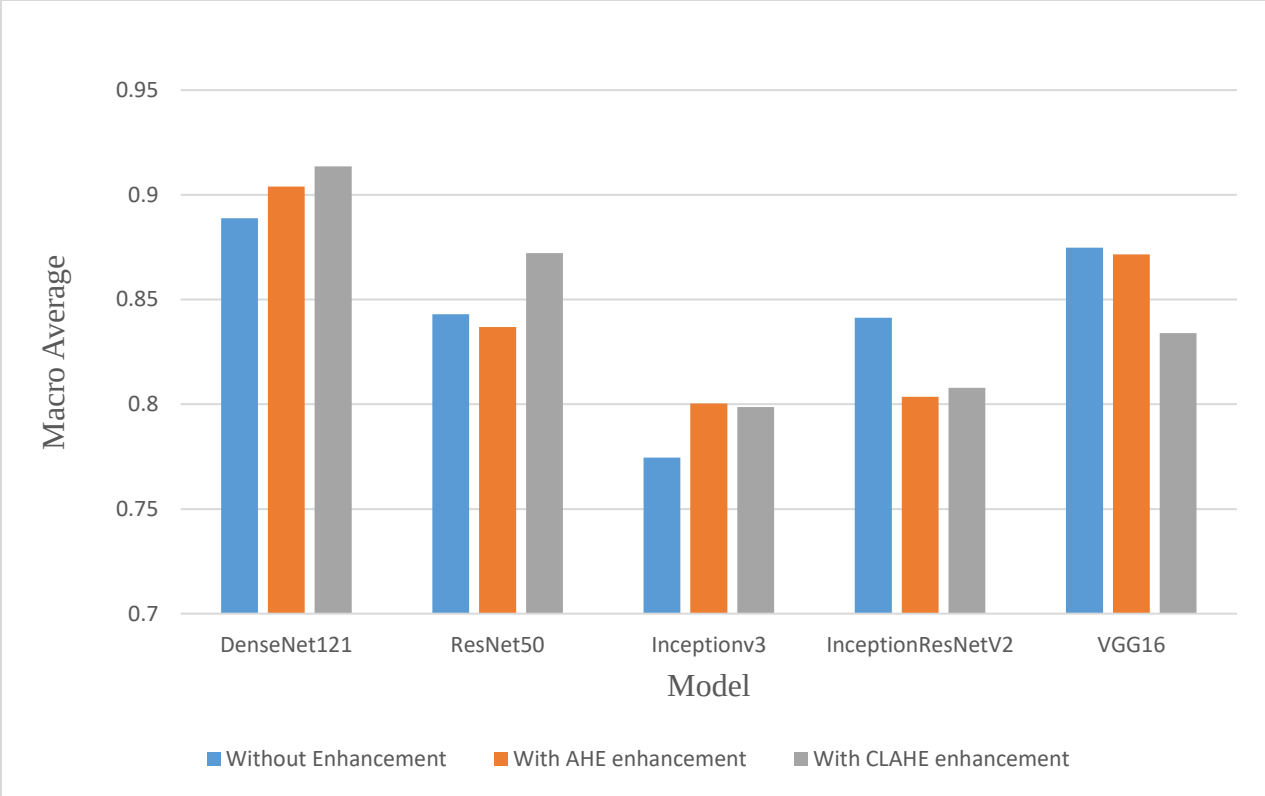

Figure 44: Comparison of Macro Average of the developed models.

4.5 Prediction of the Developed Enhanced DenseNet121 with Contrast Limited Adaptive Histogram Equalization (CLAHE)

After the development, training, and subsequent validation of DenseNet121, ResNet50, Inceptionv3, InceptionResNetV2, and VGG16 models using three different input image types (enhanced with CLAHE, enhanced with AHE, and without enhancement), DenseNet121 with CLAHE enhancement gave the best results. It performed better, surpassing all others in regard to accuracy, precision, recall, and F1-score among all categories. Further tests were done to establish the model's capacity to classify images of Kenyan Sign Language correctly.

The KSL dataset contained nine classes labeled initially with corresponding integer class IDs from 0 to 8 for each class. The input images were converted into RGB format for preprocessing and then resized to 224 × 224 to fit the model format during training. CLAHE enhancement was performed to reduce background illumination and highlight the

features. These preprocessed and enhanced images were used as input for the DenseNet121 model, which classified an input image based on the highest predicted probability. Finally, the label corresponding to the expected class ID was converted, displaying the input image and its predicted class label for verification.

Figure 45 shows the output of the model prediction, where an input image of a class mosque was correctly classified.

```
test_img_mosq_path = '/kaggle/input/mos-img/WhatsApp Image 2024-10-14 at 18.10.38.jpeg'
predicted_label_mos = predict_image_class(model, test_img_mosq_path)

print(f"Predicted label: {predicted_label_mos}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 148ms/step
[[1.93865776e-06 9.40849265e-09 9.99982953e-01 1.37056119e-07
  3.20091522e-06 2.56740513e-08 2.00546594e-07 1.14287541e-05
  1.06746334e-07]]
Predicted label: Mosque
```

+ Code + Markdown

```
df_test.loc[0, 'Predicted'] = predicted_label_mos
```

```
df_test
```

	img_IDS	Label	Predicted
0	ImageID_016X4GBI.jpg	mosque	Mosque

```
# Display the test image
display_image_with_prediction(test_img_mosq_path)
```

Figure 45: CLAHE Enhanced DenseNet121 Model Prediction.

4.6 Discussion of results in comparison to related work

The study's findings indicate that the proposed improved Convolutional Neural Network (CNN) model for Kenyan Sign Language to English text translation works effectively. As discussed, the DenseNet121 model, further optimized with Contrast Limited Adaptive Histogram Equalization (CLAHE), performed way better than other models, recording an accuracy rate of 91.5%. This model successfully classifies previously unseen KSL images,

a vital step toward bonding with the deaf/dumb community and the hearing population by lessening their dependency on human interpreters. The study's performance correlates with findings from Pitaloka et al. (2017), which indicated that the performance of CNN can be increased by enhancing the images using histogram equalization techniques.

With their demonstrated very good performance in image classification and recognition, CNNs have recently attracted increased interest in translating sign languages (Myagila & Kilavo, 2022). Hasan et al. (2020) and Bantupalli (2018) developed CNN models for translating American Sign Language, and the models performed very well in the classification work. Rao et al. (2018) also introduced a CNN-based model for recognizing Indian Sign Language gestures through selfie videos, which performed very well, too. The performance of those models motivated the researcher to use CNN for Kenyan Sign Language Translation.

Another crucial factor that enhances translation in sign language is background illumination management. Image contrast adjustment is the redistribution of pixel intensity that improves the performance of CNNs, Hassan et al. 2024. One frequently used technique is histogram equalization, which tries to distribute the pixel intensity uniformly but greatly amplifies noise without discrimination. To resolve this issue, the variants were AHE and CLAHE. CLAHE provides a better edge definition without overamplifying any noise in areas of low contrast to maintain all critical image features.

The application of CLAHE enhances the CNN models in paying more attention to important information on an image, thus improving the model's capabilities in effectively translating images of variable contrast. This increases the model's performance by significant strides in classification tasks related to sign language translation.

CHAPTER FIVE
SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This study aimed to propose a better Convolutional Neural Network that can be used for translating Kenyan Sign Language to English text. This was achieved by developing DenseNet121, ResNet50, Inceptionv3, InceptionResNetV2, and VGG16 with good training and close validations using the Kenyan Sign Language dataset. Each was trained with enhanced input images by AHE; CLAHE) enhanced input images and without enhanced input images.

On the contrary, DenseNet121 enhanced with Contrast Limited Adaptive Histogram Equalization outperformed all the developed models in translating Kenyan Sign Language to text in English. That was an assurance that the background illumination effect was eliminated, hence reducing noise on the input images and allowing the model to pay attention to crucial aspects of the input images.

The evaluation performance metrics used on DenseNet121 enhanced with the CLAHE were precision, recall, accuracy, and F1-score. To visualize accuracy and loss over epochs, results were plotted on line graphs to show the model's performance as time progressed. The model was also compared to other developed models and summarized in tabulation form.

5.1 Conclusion

The study has shown the potential of CNN in Kenyan sign language gesture recognition. The best-performing model was later found to be DenseNet121 enhanced with CLAHE, further presenting an overall accuracy of 0.9154, a macro average of Precision, Recall, and F1-score of 0.9154, 0.9136, 0.9137, and weighted average of Precision, Recall, and F1-score of 0.9154, 0.9136 and 0.9136 respectively.

This work lays a good basis for future research activities in Kenyan sign language recognition, which has practical implications for developing assistive technologies for the deaf/mute community.

5.2 Recommendations

i. The study recommended adopting the enhanced Convolutional Neural Network model with the contrast-limited Adaptive Histogram Equalization technique for translating Kenyan Sign Language into English text, considering its accuracy, recall, and F1 Score performance.

ii. The large Kenyan Sign Language dataset used for training and validation may generalize better to unseen data and thus translate more accurately when targeting Kenyan Sign Language.

5.2 Suggestion for Future Research

i. Other advanced preprocessing techniques can be explored instead of ordinary CLAHE or AHE to obtain much higher accuracy for the model.

ii. Emphasis on increasing the dataset of Kenyan Sign Language and providing this dataset for public availability.

iii. Real-world application of developed Kenyan Sign Language translation models.

REFERENCES

Abdulbaki, A. S. (2014). *Hand Gestures Detection and Recognition Building System for Stroke Patients using Supervised Neural Networks*. *February*.

Abraham, E. (2019). *Real-Time Translation of Indian Sign Language using LSTM*. 1–5.

Albrecht, G. (2014). Sign Language Interpretation. *Encyclopedia of Disability*, *4*(10), 89–92. https://doi.org/10.4135/9781412950510.n724

Bantupalli, K. (2018). *American Sign Language Recognition using Deep Learning and Computer Vision*. 4896–4899.

Batta, M. (2020). Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJ*, *9*(1), 381-undefined. https://doi.org/10.21275/ART20203995

Chandra, M. M. (2019). *Sign Languages to Speech Conversion Prototype using the SVM Classifier*. 1803–1807.

Chen, Z.-H., Kim, J.-T., Liang, J., Zhang, J., & Yuan, Y.-B. (2014). *Real-Time Hand Gesture Recognition Using Finger Segmentation*. https://doi.org/10.1155/2014/267872

Chong, T. W., & Lee, B. G. (2018). American Sign Language recognition using leap motion controller with machine learning approach. *Sensors (Switzerland)*, *18*(10). https://doi.org/10.3390/s18103554

Dogra, A., Malik, K., & Chowdary, V. (2018). Sign Language Interpreter. *International Journal of Engineering & Technology*, *7*(3.12), 990. https://doi.org/10.14419/ijet.v7i3.12.17619

Dubey, K., Jha, A., Tiwari, A., & Narmatha, K. (2019). Hand Gesture Movement Recognition System Using Convolution Neural Network Algorithm. *International Research Journal of Computer Science (IRJCS)*, *6*(04), 154–160.

Gollapudi, S. (2019). Deep Learning for Computer Vision. *Learn Computer Vision Using OpenCV*, 51–69. https://doi.org/10.1007/978-1-4842-4261-2_3

Hasan, M., Srizon, A. Y., Sayeed, A., & Hasan, A. M. (2020). *Classification of Sign Language Characters by Applying a Deep Convolutional Neural Network*. *November*, 28–29.

He, S. (2019). *Research of a Sign Language Translation System Based on Deep Learning*. 392–396. https://doi.org/10.1109/AIAM48774.2019.00083

https://www.kaggle.com/madz2000/kenyan-sign-language-classification-hackathon.

Initiatives, D. (2020). *Status of disability in Kenya*. *May*, 1–18.

Johnny, S., & Nirmala, S. J. (2022). Sign Language Translator Using Machine Learning. *SN Computer Science*, *3*(1), 1–5. https://doi.org/10.1007/s42979-021-00896-y

Kamiri, J., & Mariga, G. (2021). Research Methods in Machine Learning: A Content Analysis. *International Journal of Computer and Information Technology (2279-0764)*, *10*(2), 78–91. https://doi.org/10.24203/ijcit.v10i2.79

Mantecó nID, T., del-Blanco, C. R., Jaureguizar, F., & García, N. (2019). *A real-time gesture recognition system using near-infrared imagery*. https://doi.org/10.1371/journal.pone.0223320

Mweri, G. J. (2016). The Acquisition of Kenyan Sign Language (KSL) and its Significance as a Mother Tongue and Medium of Instruction in Schools for the Deaf in Kenya. 5, 85–100.

Oleinikov, A., Abibullaev, B., Shintemirov, A., & Folgheraiter, M. (2018). Feature extraction and real-time recognition of hand motion intentions from EMGs via artificial neural networks. *2018 6th International Conference on Brain-Computer Interface, BCI 2018*, *2018-Janua*, 1–5. https://doi.org/10.1109/IWW-BCI.2018.8311527

Ongsulee, P. (2018). Artificial intelligence, machine learning and deep learning. *International Conference on ICT and Knowledge Engineering*, 1–6. https://doi.org/10.1109/ICTKE.2017.8259629

Rao, G. A., Syamala, K., Kishore, P. V. V, & Sastry, A. S. C. S. (2018). *Deep Convolutional Neural Networks for Sign Language Recognition*. 194–197.

Rastgoo, R., Kiani, K., & Escalera, S. (2020). Jou rna lP. In *Expert Systems With Applications*. Elsevier Ltd. https://doi.org/10.1016/j.eswa.2020.113794

Republic of Kenya. (2017). Kenya Gazette Supplement. *Finance Act, 2018*, *59*(59), 165. http://www.ilo.org/dyn/travail/docs/505/Employment Act 2007.pdf

Sharma, A. (2020). *Sign Language to Speech Translation*.

Abdel-Salam, R., Mostafa, R., & Abdel-Gawad, A. H. (2022). RIECNN: real-time image enhanced CNN for traffic sign recognition. *Neural Computing and Applications*, *34*(8), 6085–6096. https://doi.org/10.1007/s00521-021-06762-5

Al-qurishi, M., Khalid, T., & Souissi, R. (2021). Deep Learning for Sign Language Recognition : Current Techniques , Benchmarks , and Open Issues. *IEEE Access*, *9*, 126917–126951. https://doi.org/10.1109/ACCESS.2021.3110912

Cheok, M. J., Omar, Z., & Jaward, M. H. (2019). A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics*, *10*(1), 131–153. https://doi.org/10.1007/s13042-017-0705-5

Gad, A. F. (2018). Practical Computer Vision Applications Using Deep Learning with CNNs. In *Practical Computer Vision Applications Using Deep Learning with CNNs*. https://doi.org/10.1007/978-1-4842-4167-7

Gupta, M., Singh, G., & Yadav, A. (2021). CNN Based Speech and Text Translation Using Sign Language. *Proceedings - 2021 3rd International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2021*, 433–437. https://doi.org/10.1109/ICAC3N53548.2021.9725601

Hassan, M. A., Ali, A. H., & Sabri, A. A. (2024). Enhancing communication: Deep learning for Arabic sign language translation. *Open Engineering*, *14*(1). https://doi.org/10.1515/eng-2024-0025

Jayadeep, G., Vishnupriya, N. V, Venugopal, V., Vishnu, S., & Geetha, M. (2020). *Mudra : Convolutional Neural Network based Indian Sign Language Translator for Banks*. *Iciccs*, 1228–1232.

Kamiri, J., & Mariga, G. (2021). Research Methods in Machine Learning: A Content Analysis. *International Journal of Computer and Information Technology(2279-0764)*, *10*(2), 78–91. https://doi.org/10.24203/ijcit.v10i2.79

Khan, A. I., & Al-Habsi, S. (2020). Machine Learning in Computer Vision. *Procedia Computer Science*, *167*(2019), 1444–1451. https://doi.org/10.1016/j.procs.2020.03.355

Mustafa, W. A., & Abdul Kader, M. M. M. (2018). A Review of Histogram Equalization Techniques in Image Enhancement Application. *Journal of Physics: Conference Series*, *1019*(1). https://doi.org/10.1088/1742-6596/1019/1/012026

Myagila, K., & Kilavo, H. (2022). A Comparative Study on Performance of SVM and CNN in Tanzania Sign Language Translation Using Image Recognition. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2021.2005297

Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., & Shaalan, K. (2019). Speech Recognition Using Deep Neural Networks: A Systematic Review. *IEEE Access*, *7*(c), 19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880

Nelson, A., Price, K. J., Multari, R., Price, K. J., Nelson, A., & Multari, R. (2019). *SMU Data Science Review ASL Reverse Dictionary - ASL Translation Using Deep Learning ASL Reverse Dictionary ASL Translation Using Deep Learning*. *2*(1).

Papastratis, I., Chatzikonstantinou, C., Konstantinidis, D., Dimitropoulos, K., & Daras, P. (2021). Artificial intelligence technologies for sign language. *Sensors*, *21*(17). https://doi.org/10.3390/s21175843

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of Deep Learning for Object Detection. *Procedia Computer Science*, *132*(Iccids), 1706–1717. https://doi.org/10.1016/j.procs.2018.05.144

Pitaloka, D. A., Wulandari, A., Basaruddin, T., & Liliana, D. Y. (2017). Enhancing CNN with Preprocessing Stage in Automatic Emotion Recognition. *Procedia Computer Science*, *116*, 523–529. https://doi.org/10.1016/j.procs.2017.10.038

Rao, B. S. (2020). Dynamic Histogram Equalization for contrast enhancement for digital images. *Applied Soft Computing Journal*, *89*, 106114. https://doi.org/10.1016/j.asoc.2020.106114

Salman, H., Grover, J., & Shankar, T. (2018). *Hierarchical Reinforcement Learning for Sequencing Behaviors*. *2733*(March), 2709–2733. https://doi.org/10.1162/NECO

Shukla, K. N. (2017). A Review on Image Enhancement Techniques. *International Journal of Engineering and Applied Computer Science*, *02*(07), 232–235. https://doi.org/10.24032/ijeacs/0207/05

Vijayalakshmi, D., Nath, M. K., & Acharya, O. P. (2020). A Comprehensive Survey on Image Contrast Enhancement Techniques in Spatial Domain. In *Sensing and Imaging* (Vol. 21, Issue 1). Springer US. https://doi.org/10.1007/s11220-020-00305-3

Wadhawan, A., & Kumar, P. (2020). Deep learning-based sign language recognition system for static signs. *Neural Computing and Applications*, *32*(12), 7957–7968. https://doi.org/10.1007/s00521-019-04691-y

Wu, Q., Liu, Y., Li, Q., Jin, S., & Li, F. (2017). The application of deep learning in computer vision. *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, *2017-Janua*, 6522–6527. https://doi.org/10.1109/CAC.2017.8243952

Sigit, R., & Kartika, D. R. (2016). *3D Sign Language Translator Using Optical Flow*.

Tomaszewski, P., & Napoli, D. J. (n.d.). *From sign language to spoken language ? A new discourse of language development in deaf children*. https://doi.org/10.2478/plc-2019-0004

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*. https://doi.org/10.1155/2018/7068349

Wadhawan, A., & Kumar, P. (2020). Deep learning-based sign language recognition system for static signs. *Neural Computing and Applications*, *32*(12), 7957–7968. https://doi.org/10.1007/s00521-019-04691-y

Xu, P. (2017). *A Real-time Hand Gesture Recognition and Human-Computer Interaction System*.

APPENDICES

Appendix 1: Model Development Source code

```
import pandas as pd
import os
import cv2
import random
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import tensorflow as tf
from tensorflow.keras.utils import to_categorical

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout,
BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img
from keras.models import Model
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Conv2D, AveragePooling2D,MaxPooling2D,
Dropout, Flatten, Activation, Input, Add

from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from PIL import Image
from tqdm import tqdm
```

```
from IPython.display import clear_output

from tensorflow.keras.applications.densenet import DenseNet121
from tensorflow.keras.optimizers import SGD

print(os.listdir("../input/sign-language"))

images_path = '../input/sign-language/KSL-dataset/Images'
test_path = '../input/sign-language/KSL-dataset/Test.csv'
train_path = '../input/sign-language/KSL-dataset/Train.csv'

train_df = pd.read_csv(train_path)
train_df.head()

images = [f for f in os.listdir(images_path) if f.endswith(('.jpg'))]
random_images = random.sample(images, 5)

plt.figure(figsize=(15, 10))
for i, image_file in enumerate(random_images):
    img_path = os.path.join(images_path, image_file)
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    plt.subplot(1, 5, i + 1)
    plt.imshow(img)
    plt.title(f'Image: {image_file}')
    plt.axis('on')
plt.tight_layout()
plt.show()
```

```
train_df['img_path'] = train_df['img_IDS'].apply(lambda x: os.path.join(images_path, f"{x}.jpg"))

def load_image(img_path, target_size=(224, 224)):
    img = Image.open(img_path)
    img = img.convert('RGB')
    img = img.resize(target_size)
    img = np.array(img)
    return img

main_dir = '/kaggle/working/ksl_ds'
if not os.path.exists(main_dir):
    os.makedirs(main_dir)

classes = train_df['Label'].unique()
classes

     'You', 'Friend', 'Seat'], dtype=object)
for cls in classes:
    # Sanitize class names to create valid directory names
    cls_folder = os.path.join(main_dir, cls.replace('/', '_').replace(' ', '_'))
    if not os.path.exists(cls_folder):
        os.makedirs(cls_folder)

# save images to kaggle/working
def process_and_save_images(df, img_path_column, target_size=(224, 224)):
    for index, row in df.iterrows():
        img_path = row[img_path_column]
        cls = row['Label']  # Replace 'label' with your actual column name for class labels
        cls_folder = os.path.join(main_dir, cls.replace('/', '_').replace(' ', '_'))
```

```
        img = load_image(img_path, target_size)

        # Create a file name and save the image
        file_name = os.path.basename(img_path)
        save_path = os.path.join(cls_folder, file_name)
        img_pil = Image.fromarray(img)
        img_pil.save(save_path)

process_and_save_images(train_df, 'img_path')
dir_path = '/kaggle/working/ksl_ds'
os.listdir(dir_path)

images = []
labels = []

for label in os.listdir(dir_path):
    label_dir = os.path.join(dir_path, label)
    for img in os.listdir(label_dir):
        img_path = os.path.join(label_dir, img)
        images.append(img_path)
        labels.append(label)
    images, labels = shuffle(images,labels)
pd.set_option('display.max_colwidth', None)
df_ksl = pd.DataFrame({'path':images, 'label':labels})
df_ksl.head()

df_ksl['label'].value_counts()

plt.rcParams.update({'font.size': 13})
plt.figure(figsize=(7,7))
sns.countplot(x = df_ksl['label'])
```

```
plt.xlabel('Class')
plt.xticks(rotation=45)
plt.ylabel('Number of Images')
plt.title('Number of Images for Each Class')
plt.show()

ksl_train_df, ksl_val_df = train_test_split(df_ksl, test_size=0.15, stratify=df_ksl['label'],
random_state=42)

print("Size of training set: {}".format(len(ksl_train_df)))
print("Size of validation set: {}".format(len(ksl_val_df)))

class configurations:
    IMAGE_SIZE = 224
    BRIGHTNESS = (0.64, 1.37) # (MIN, MAX)
    CONTRAST = (0.64, 1.37) # (MIX, MAX)
    BATCH_SIZE = 32
    EPOCHS = 40
    LEARNING_RATE = 0.001

def augment_image_with_CLAHE(image):
    """
    Images with Contrast Limited Adaptive Histogram Equalization (CLAHE)
    """
    datagen = ImageDataGenerator(
        brightness_range=configurations.BRIGHTNESS,
        rescale=1./255
    )
    # Convert input image from numpy to PIL Image
    image = np.expand_dims(image, axis=0)
    # Apply Transformations
```

```
    augmented_image = datagen.flow(image, batch_size=1)[0][0]

    # Convert to YUV color space
    img_yuv = cv2.cvtColor((augmented_image * 255).astype(np.uint8),
cv2.COLOR_RGB2YUV)
    img_y_channel = img_yuv[..., 0].astype(np.uint8)

    # Apply CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    img_yuv[..., 0] = clahe.apply(img_y_channel)

    # Convert back to RGB - to be used
    equalized_image = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2RGB)
    equalized_image = equalized_image / 255.0
    return equalized_image

"""
Display images with CLAHE enhancement
"""
def open_images(paths, augment=True):
    '''
    Given a list of paths to images, this function returns the images as arrays, and
conditionally augments them
    '''
    images = []
    for path in paths:
        image = load_img(path,
target_size=(configurations.IMAGE_SIZE,configurations.IMAGE_SIZE))
        if augment:
            image = augment_image_with_CLAHE(image)
        image = np.array(image)
```

```
        image = image/image.max()
        images.append(image)
    return np.array(images)

"""
Images with CLAHE enhancement
"""
k = random.randint(0,5000)
image_paths = list(ksl_train_df.path[k:k+10])
labels = list(ksl_train_df.label[k:k+10])
images = open_images(image_paths, augment=True)
plt.rcParams.update({'font.size': 10})
fig = plt.figure(figsize=(20, 8))

for i in range(0, 10):
    fig.add_subplot(2, 5, i+1)
    plt.imshow(images[i])
    plt.axis('on')
    plt.title(labels[i])
plt.show()
```

“”” DENSENET121 MODEL”””

```
LABELS =
['Seat','Enough_Satisfied','Mosque','You','Love','Friend','Me','Church','Temple']
label_encoder =
{'Seat':0,'Enough_Satisfied':1,'Mosque':2,'You':3,'Love':4,'Friend':5,'Me':6,'Church':7,'Te
mple':8}
label_Decoder =
{0:'Seat',1:'Enough_Satisfied',2:'Mosque',3:'You',4:'Love',5:'Friend',6:'Me',7:'Church',8:'T
emple'}
```

```
"""
Generator function with CLAHE equalization
"""
def data_generator(df, batch_size=configurations.BATCH_SIZE, augment=True,
epochs=configurations.EPOCHS):
    for e in range(epochs):
        for x in range(0,len(df), batch_size):
            image_paths = df.path[x:x+batch_size]
            images = open_images(image_paths, augment=augment)
            labels = df.label[x:x+batch_size]
            labels = [label_encoder[label] for label in labels]
            yield images, np.array(labels)

"""
With CLAHE equalization
"""
train_data_generator = data_generator(ksl_train_df,
batch_size=configurations.BATCH_SIZE, augment=True,
epochs=configurations.EPOCHS)
train_steps = int(len(ksl_train_df)/configurations.BATCH_SIZE)

val_data_generator = data_generator(ksl_val_df,
batch_size=configurations.BATCH_SIZE, augment=False,
epochs=configurations.EPOCHS)
val_steps = int(len(ksl_val_df)/configurations.BATCH_SIZE)

def model_with_clahe_eq():
    """
    Model with CLAHE equalization
    """
    sgd_optimizer = SGD(configurations.LEARNING_RATE)
```

```
    densenet_model= DenseNet121(weights='imagenet', include_top=False,
input_tensor=Input(shape=(224,224,3)))
    for layer in densenet_model.layers[:-10]:
        layers.trainable=False
    hm = densenet_model.output
    hm = AveragePooling2D(pool_size=(4,4))(hm)
    hm = Flatten(name = 'flatten')(hm)
    hm = Dense(256, activation = 'relu')(hm)
    hm = Dropout(0.3)(hm)
    hm = Dense(128, activation = 'relu')(hm)
    hm = Dropout(0.2)(hm)
    hm = Dense(9,activation = 'softmax')(hm)

    model = Model(inputs=densenet_model.input, outputs = hm)
    model.compile(optimizer=sgd_optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

"""
model
"""
model = model_with_clahe_eq()



"""
Model summary
"""

from tabulate import tabulate
```

```
def print_last_10_layers(model):
    total_layers = len(model.layers)
    print(f"Total layers in the model: {total_layers}\n")

    # Prepare headers and data
    headers = ["Layer Name", "Output Shape", "Number of Parameters"]
    data = []

    # Gather details for the last 10 layers
    for layer in model.layers[-10:]:
        try:
            output_shape = layer.output.shape if hasattr(layer, 'output') else 'No output shape
available'
        except AttributeError:
            output_shape = 'No output shape available'

        data.append([layer.name, str(output_shape), layer.count_params()])

    # Print the table with continuous borders
    print(tabulate(data, headers=headers, tablefmt="fancy_grid"))

    # Print total, trainable, and non-trainable parameters
    total_params = model.count_params()
    trainable_params = sum([layer.count_params() for layer in model.layers if
layer.trainable])
    non_trainable_params = total_params - trainable_params

    print("\nModel Parameters Summary:")
    print(f"Total params: {total_params:,}")
    print(f"Trainable params: {trainable_params:,}")
    print(f"Non-trainable params: {non_trainable_params:,}")
```

```
model = model_with_clahe_eq()
print_last_10_layers(model)

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath='/kaggle/working/model.keras',
    save_weights_only=False,
    monitor='val_accuracy',
    mode='max', verbose=1,
    save_best_only=True)

history = model.fit(train_data_generator, epochs=configurations.EPOCHS,
steps_per_epoch=train_steps,
        validation_data=val_data_generator, validation_steps=val_steps,
callbacks=[model_checkpoint_callback])

"""
Training and Validation accuracy With CLAHE equalization
"""
epochs = range(1, len(history.history['accuracy']) + 1)
plt.figure(figsize=(10, 5))
plt.plot(epochs, history.history['accuracy'], color='purple', label='Train Accuracy')
plt.plot(epochs, history.history['val_accuracy'], color='pink', label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend(loc="upper right")
plt.grid(True)
plt.show()

"""
```

```
Training and Validation loss With CLAHE equalization
"""

plt.figure(figsize=(10, 5))
plt.plot(epochs, history.history['loss'], color='red', label='Train Loss')
plt.plot(epochs, history.history['val_loss'], color='green', label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend(loc="upper right")
plt.grid(True)
plt.show()
"""
Classification Report
"""

val_data_generator= data_generator(ksl_val_df,
batch_size=configurations.BATCH_SIZE, augment=False, epochs=1)
val_steps = int(len(ksl_val_df)/configurations.BATCH_SIZE)

y_pred = []
y_true = []
for x,y in tqdm(val_data_generator, total=val_steps):
    pred = model.predict(x)
    pred = np.argmax(pred, axis=-1)
    for i in pred:
        y_pred.append(label_Decoder[i])
    for i in y:
        y_true.append(label_Decoder[i])
clear_output()
print(classification_report(y_true, y_pred, digits=4
```

Appendix 1I: Model Testing Source Code

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from PIL import Image
import cv2

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

path = '/kaggle/input/test_densenet/keras/default/1/model.keras'
model = load_model(path)

# Define label encoders
label_Decoder = {0: 'Seat', 1: 'Enough_Satisfied', 2: 'Mosque', 3: 'You', 4: 'Love', 5:
'Friend', 6: 'Me', 7: 'Church', 8: 'Temple'}

# Load and preprocess the image
def load_image(img_path, target_size=(224, 224)):
    """
    Load and preprocess the image: Convert to RGB and resize.
    """
    img = Image.open(img_path)
    img = img.convert('RGB')
```

```
    img = img.resize(target_size)
    img = np.array(img)
    return img

def augment_image_with_CLAHE(image):
    """
    Apply CLAHE (Contrast Limited Adaptive Histogram Equalization) on the image.
    """
    datagen = ImageDataGenerator(
        brightness_range=(0.64, 1.37),
        rescale=1./255
    )

    # Convert input image from numpy to PIL Image
    image = np.expand_dims(image, axis=0)

    # Apply Transformations
    augmented_image = datagen.flow(image, batch_size=1)[0][0]

    # Convert to YUV color space
    img_yuv = cv2.cvtColor((augmented_image * 255).astype(np.uint8),
cv2.COLOR_RGB2YUV)
    img_y_channel = img_yuv[..., 0].astype(np.uint8)

    # Apply CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    img_yuv[..., 0] = clahe.apply(img_y_channel)

    # Convert back to RGB
    equalized_image = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2RGB)
    equalized_image = equalized_image / 255.0
```

```
    return equalized_image

def preprocess_test_image(img_path, target_size=(224, 224)):
    """
    Preprocess the image for model prediction: Load, resize, apply CLAHE, and normalize.
    """
    img = load_image(img_path, target_size=target_size)
    img = augment_image_with_CLAHE(img)
    img = np.array(img)
    img = img / img.max()  # Normalize to range [0, 1]
    img = np.expand_dims(img, axis=0)

    return img

# Predict the class of the sign language image
def predict_image_class(model, img_path):
    """
    Given a model and image path, preprocess the image and predict the class.
    """
    # Preprocess the image
    processed_image = preprocess_test_image(img_path)
    predicted_class = model.predict(processed_image)
    print(predicted_class)

    """
    Argmax is most commonly used in machine learning
    for finding the class with the largest predicted probability.
    """
    predicted_class_id = np.argmax(predicted_class, axis=-1)[0]
```

```
    predicted_label = label_Decoder[predicted_class_id]

    return predicted_label

# Display the test image with predicted label
def display_image_with_prediction(img_path, predicted_label):
    """
    Display the image with its predicted label as the title.
    """
    # Load and display the image
    test_img = load_img(img_path, target_size=(224, 224))
    plt.imshow(test_img)
    plt.title(f"Predicted label: {predicted_label}")
    plt.axis('off')
    plt.show()
# Example usage with a new test image
test_image_path = '/kaggle/input/test-d/test.png'
test_you = '/kaggle/input/you-test/You.jpg'

# Predict and display the result
predicted_label = predict_image_class(model, test_you)

print(f"Predicted label: {predicted_label}")

# Display the test image with its predicted label
display_image_with_prediction(test_image_path, predicted_label)

df_test = pd.read_csv("/kaggle/input/test-csv-mos/test.csv")
df_test
```

```
test_img_mosq_path = '/kaggle/input/mos-img/WhatsApp Image 2024-10-14 at 18.10.38.jpeg'
predicted_label_mos = predict_image_class(model, test_img_mosq_path)

print(f"Predicted label: {predicted_label_mos}")

df_test.loc[0, 'Predicted'] = predicted_label_mos

df_test
```

Appendix 1II: Chuka University Introductory Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

OFFICE OF THE DIRECTOR
BOARD OF POSTGRADUATE STUDIES

Telephones: 020-2310512/18
Direct Line: 020-268 7625
postgraduate@chuka.ac.ke
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

REF: SM22/45894/21 **8th August, 2024**

Director
National Commission for Science Technology and Innovation
Off Waiyaki Way, Upper Kabete
P O Box 30623, 00100
Nairobi.

Dear Sir / Madam,

MUTHUI NANCY NJOKI

The above-named person is a *bona fide* student of Chuka University pursuing MSC in Computer Science proposal titled**: An Enhanced Convolutional Neural Network Model for Translating Kenya Sign Language into Text in English.**

Ms. Njoki has defended at the Faculty level and is now expected to conduct research. Any assistance accorded will be highly appreciated.

Yours sincerely,

08 AUG 2024

Prof. Moses Muraya, Ph.D.

DIRECTOR
BOARD OF POSTGRADUATE STUDIES

Appendix IV: Ethics Review Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

CHUKA UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE

Telephones: 020-2310512/18
Direct Line: 0772894438
Email: info@chuka.ac.ke,
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

5th June, 2024

REF: CUIERC/ NACOSTI/553
TO: **Muthui Nancy Njoki**

RE: An Enhanced Convolutional Neural Network Model for Translating Kenya Sign Language into Text in English

This is to inform you that *Chuka University IERC* has reviewed and approved your above research proposal. Your application approval number is *NACOSTI/NBC/AC-0812.* The approval period is 5th June, 2024 – 5th June, 2025.

This approval is subject to compliance with the following requirements;

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by *Chuka University IERC*.
iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to *Chuka University IERC* within 72 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to *Chuka University IERC* within 72 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to *Chuka University IERC*.

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely

Dr. Benjamin Kanga
SECRETARY

Appendix V: NACOSTI License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **639073**

Date of Issue: **22/August/2024**

RESEARCH LICENSE

This is to Certify that Miss.. Nancy Njoki Muthui of Chuka University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Tharaka-Nithi on the topic: AN ENHANCED CONVOLUTIONAL NEURAL NETWORK MODEL FOR TRANSLATING KENYAN SIGN LANGUAGE INTO TEXT IN ENGLISH for the period ending : 22/August/2025.

License No: **NACOSTI/P/24/39038**

639073

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions