**NULLCLASS INTERNSHIP REPORT 1**

**PROJECT 1 TITLE: Sign Language Recognition**

1. Introduction

Sign Language is an essential form of communication for people having difficulties with hearing.  Automated systems that recognise and interpret sign gestures close the communication gap, and enable people with hearing loss to interact more successfully with the community.A system for sign language recognition has been created in this project. Through the conversion of sign gestures into textual information, sign language recognition is a useful tool that helps people with difficulties with hearing.Convolutional Neural Networks (CNNs) are used in this system, and it is enhanced with an appealing Graphical User Interface (GUI) to facilitate user-friendly interaction.

1. Background

The projects's basis is the use of convolutional neural networks in image recognition. The ability of CNNs to identify patterns in images regardless of their placement is advantageous for classifying signs in sign language since signs can appear in many places within an image. CNNs therefore are well-suited for image identification tasks, which makes them perfect for deciphering sign language actions that have been captured in a picture. The dataset used to train the model includes pictures of different sign language symbols, each connected to a distinct alphabet. Individuals are able to interact and test different images with the trained model with the help of Graphical User Interface.

1. Learning Objectives

* To Become proficient on the fundamental concepts underlying Convolutional Neural Networks and how they are employed in image classification applications.
* To recognise the value of normalising pixel values and reshaping image data to match the model input shape in order to guarantee reliable and efficient training.
* Dropout layers are used to increase the model's randomness and prevent overfitting.
* Recognise the convolutional, pooling, flattening, and fully linked layers that constitute to form Convolutional Neural Network’s architecture.
* Acquire the knowledge of building the model by utilising the proper optimizer and loss functions.
* Employ Tkinter, a Python GUI toolkit, and become acquainted with its features.
* Learn how to load pre-trained Convolutional Neural Networks model, which will make it less challenging to integrate Convolutional Neural Networks model into graphical user interface.
* Acknowledge the wider social significance of sign language recognition, particularly to facilitate effective communication for those with disability of hearing.

1. Activities and Tasks

A: Examining the  sign language dataset.

• Use Pandas to load the datasets for testing and training.

• Examine dataset data with `info()` and `describe()` to comprehend its features and organisation.

B: Data reshaping and normalisation.

• Reshape the feature set to have the proportions suitable for a CNN model (28x28x1).

• Normalize pixel values in the feature set to be in the range [0, 1].

C: Augmenting the training dataset using ImageDataGenerator.

•  For data augmentation , create a `ImageDataGenerator}.

• Define augmentation parameters, such as shift, rotation, shear, zoom, and flip horizontally.

D: Convolutional Neural Network (CNN) development and training.

• Use Keras for building a sequential CNN model.

• Add convolutional layers, pooling layers, activation functions.

• Include dense layers with the proper activation functions along with  a flatten layer.

• Use the Adam optimizer with  and categorical crossentropy loss.

E: Model training.

• Train the model using the training dataset .

• Specify validation data for training. and epochs.

• Allow data shuffles to improve data randomness and keep the model from overfitting.

F: Saving onto the trained model.

• To aid in its integration with the Graphical User Interface, save the trained model in a file called "action.h5".

G: Assessing the model using the test dataset and making predictions.

• Assess the accuracy of the model utilising the testing dataset.

•Predict the letter through generating predictions on a sample input image.

H: Developing a graphical user interface application to recognise sign language.

• Load 'action.h5' file which is the trained model.

• Make a GUI based on Tkinter that has buttons and labels.

• Set the ability for the GUI to load and display images.

• As a result, display the predicted letter.

1. Skills and Competencies

* Capacity to manipulate data using libraries such as Pandas and NumPy.
* Understanding of Keras for creating and training neural network models employing with layers, activation functions, and optimizers.
* Knowledge of CNN architecture, including convolutional layers and pooling layers.
* The ability to normalise and reshape images for model input.
* Knowledge of model compilation with optimizers, loss functions, and metrics.
* The ability to train models using predetermined parameters, such as validation data, epochs, and batch size.
* Expertise in utilising Tkinter to create graphical user interfaces.
* Ability to load pre-trained Convolutional Neural Networks model and interact with Graphical User Interface.

1. Feedback and Evidence

Feedback

• Verify that the code is arranged and well-commented for a better understanding.

• Assess the accuracy of the model using the test data.

•Examine to ensure that the GUI set up displays images and predicts accurately.

Evidence

• Logs demonstrating accuracy and loss when training the model.

• The model file that was saved after training .

• Visual evidence displaying the GUI application by showing image loading, prediction, and result display.

• Comments embedded into the code that describe logic and functionality of the code.

1. Challenges and Solutions

• Challenge: Before entering the raw image data into the neural network, it must be preprocessed.

Solution: The code uses ImageDataGenerator to reshape, normalise, and perform data augmentation to correctly handle data preprocessing.

• Challenge: Overfitting must be managed and proper hyperparameter tweaking is required while training a deep learning model.

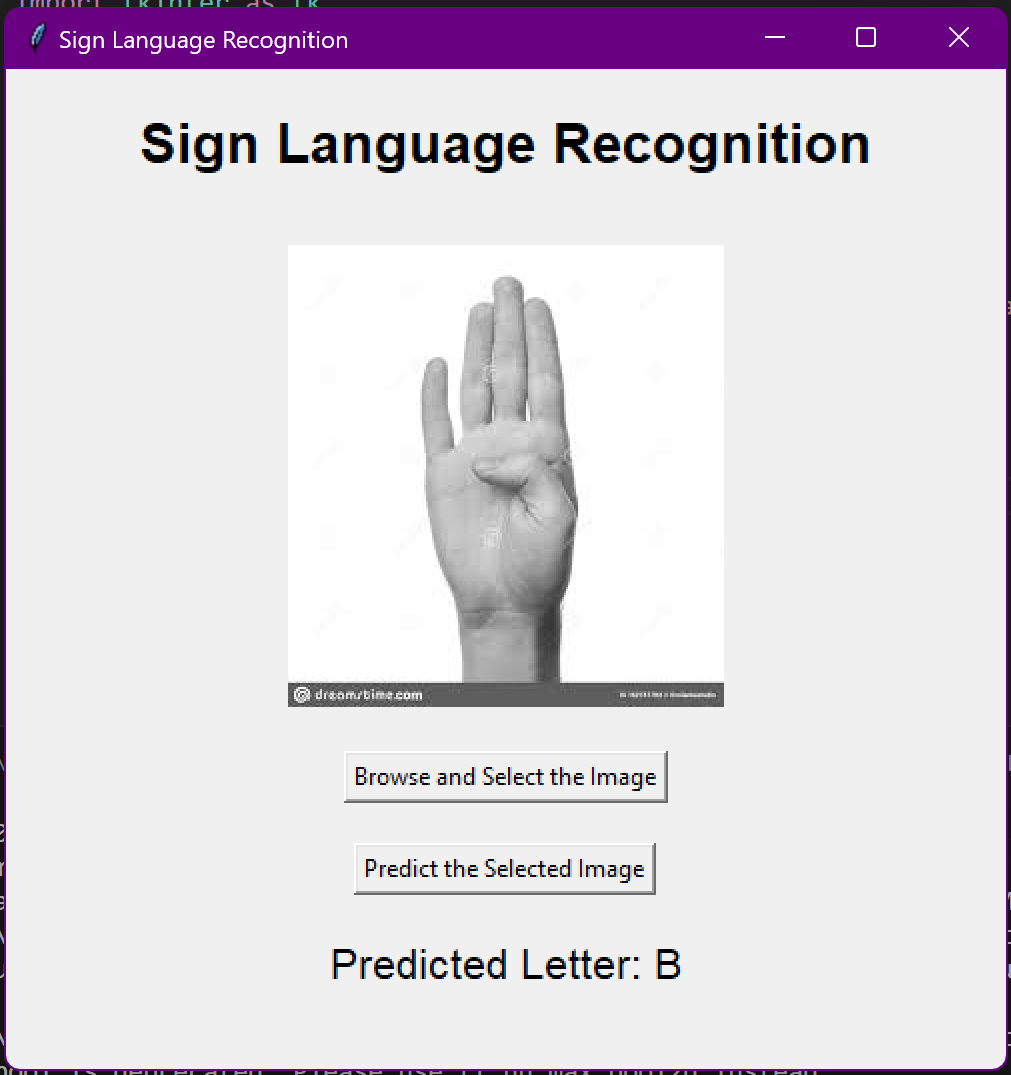
Solution: The algorithm uses the Adam optimizer, category crossentropy loss, and leverages dropout to prevent overfitting. To increase generalisation, the  model is trained with augmented data.

• Challenge:  Evaluating the model's performance and saving it for integrate with Graphical User Interface

Solution: Metrics including loss and accuracy are obtained by evaluating the model with the evaluate() function. Furthermore, the model is readily available with no requirement for retraining as it is saved in an  action.h5 for .

1. Outcomes and Impact

* The evaluation determines  that the trained model can accurately recognise gestures in sign language with model accuracy equal to 99.55 percent.
* To assess the model's ability for generalisation, test set is used for evaluation.
* For individuals who have an interest in sign language recognition, the GUI makes the model usable and accessible by enabling users to import, visualise, and receive predictions for an image.
* In the image below Graphical User Interface is shown which is integrated with our model and we can see that the image is showing gesture of letter B and our model is predicting the gesture correctly.



1. Conclusion

The developed model performs accurately when it attempts recognising gestures in sign language.The approach can be comprised into applications designed to help people with hearing loss communicate more effectively.

With its accessible interface, the GUI application improves user interaction with the model.The code serves as a foundation for future enhancements, including model optimisation, the addition of more varied datasets, and the enlargement of application functionality.More reliable sign language recognition systems with more uses can result from ongoing research and development.