ASL- Alphabet Image Recognition Project Report

Team ID: Team-592425

1. INTRODUCTION

1.1. PROJECT OVERVIEW

The ASL Alphabet Recognition System is aimed at developing a computer vision application that can recognize and interpret the American Sign Language (ASL) alphabet signs, facilitating communication between ASL users and non-signers. The project will collect and preprocess a diverse ASL alphabet sign dataset, design and train a deep learning model for sign recognition, and integrate it into a user-friendly interface.

Key Objectives:

1. Data Collection: Gather a comprehensive dataset of ASL alphabet signs, encompassing various signers, lighting conditions, and backgrounds.
2. Model Development: Implement a deep learning model, potentially using CNNs and RNNs, to accurately recognize ASL signs.
3. Real-time Detection: Enable real-time recognition using a webcam or video input source.
4. User Interface: Develop an accessible and user-friendly interface.
5. Integration: Convert recognized signs to text or speech for non-signers.
6. Testing and Validation: Thoroughly test the system's accuracy and usability.
7. Deployment: Make the system available as a desktop application, mobile app, or web service.

The project's success will enhance accessibility and foster better communication between ASL users and the wider community, promoting inclusivity and understanding.

1.2. PURPOSE

The primary purpose of the ASL (American Sign Language) Alphabet Recognition Project is to bridge the communication gap between individuals who use ASL as their primary means of communication and those who do not have proficiency in sign language. This project aims to fulfill several important objectives:

1. **Enhancing Accessibility**: The project seeks to make ASL more accessible to a wider audience, enabling non-signers to understand and communicate with members of the Deaf and Hard of Hearing community who rely on ASL for everyday communication.
2. **Inclusivity**: By developing a reliable ASL alphabet recognition system, the project promotes inclusivity by allowing individuals with hearing impairments to more fully participate in society, education, and the workforce.
3. **Educational Tool**: It serves as an educational tool to teach ASL to those interested in learning sign language, contributing to a deeper understanding of the culture and language of the Deaf community.
4. **Empowering ASL Users**: ASL recognition empowers Deaf individuals to communicate with the broader population, reducing language barriers and enabling them to access more opportunities.

5. **Research and Development**: The project fosters research and development in the field of computer vision, machine learning, and artificial intelligence, with potential applications in other domains.

Overall, the purpose of this ASL Alphabet Recognition Project is to foster communication, understanding, and inclusivity between the Deaf and Hard of Hearing community and the wider society, leveraging technology to break down communication barriers and promote accessibility.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

The existing problem in the context of ASL (American Sign Language) alphabet recognition is the lack of accessible, accurate, and widely available tools for recognizing and interpreting ASL signs. This problem presents several challenges:

1. **Communication Barriers**: Deaf and Hard of Hearing individuals often face significant communication barriers when interacting with people who do not understand ASL. This results in limited social interactions, educational challenges, and difficulties in accessing various services.
2. **Reliance on Interpreters**: While sign language interpreters are available, they are not always accessible, leading to delays and potentially higher costs. This reliance on manual interpretation can hinder spontaneous and efficient communication.
3. **Limited Educational Resources**: Deaf individuals may lack access to educational resources tailored to their needs. An effective ASL alphabet recognition tool could provide educational support, helping Deaf individuals learn and communicate more effectively.
4. **Workplace Accessibility**: In professional settings, Deaf employees often face difficulties in communicating with hearing colleagues. This can impact their job performance and limit career advancement.
5. **Inclusivity and Social Isolation**: The absence of tools for ASL recognition contributes to the exclusion and social isolation of Deaf individuals, making it challenging for them to fully participate in society.
6. **Lack of Automated Solutions**: While there have been advancements in computer vision and machine learning, there is a scarcity of automated ASL recognition solutions that are both accurate and widely accessible to the Deaf community.

The existing problem underscores the need for an effective ASL alphabet recognition system that can break down communication barriers, promote inclusivity, enhance education and employment opportunities, and facilitate more seamless interactions between Deaf and hearing individuals.

2.2. REFERENCES

https://en.wikipedia.org/wiki/American_Sign_Language
https://www.tensorflow.org/overview
https://www.sciencedirect.com/science/article/abs/pii/S0957417420310745

2.3. PROBLEM STATEMENT DEFINITION

The problem addressed by the ASL (American Sign Language) Alphabet Recognition Project is the limited communication and accessibility barriers faced by individuals who use ASL as their primary means of communication, particularly when interacting with those who do not have sign language proficiency. This project seeks to solve this problem by developing a robust and accurate ASL alphabet recognition system.

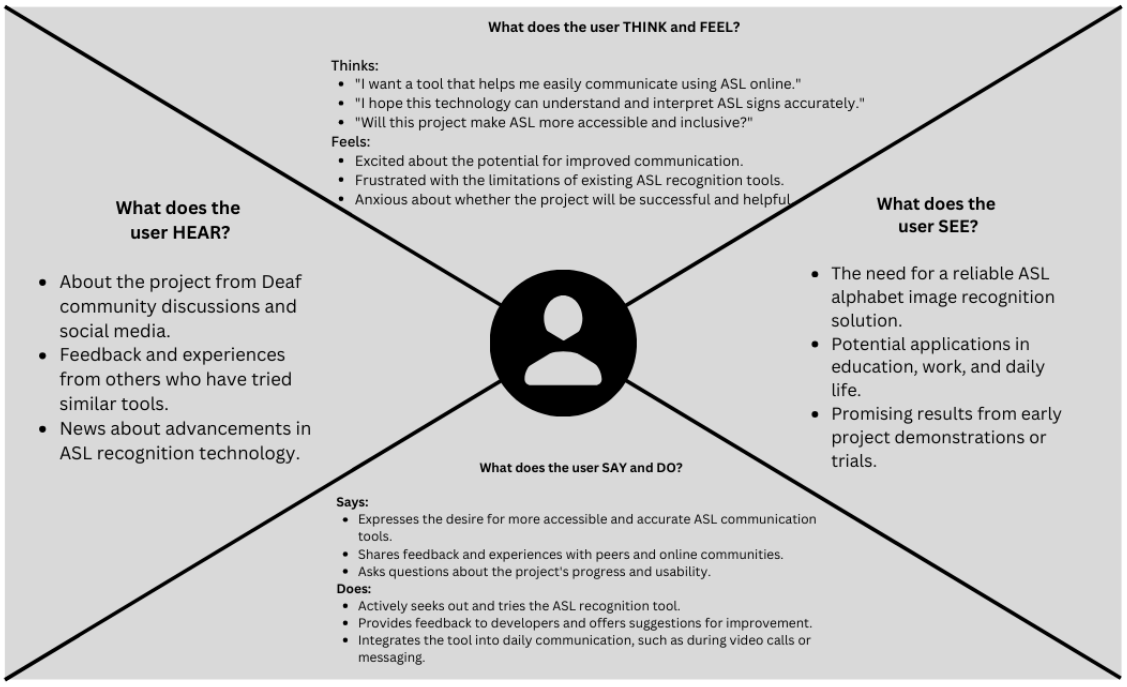
1. **Communication Gap**: Deaf and Hard of Hearing individuals often face challenges when trying to communicate with those who do not understand ASL. This gap in communication limits their ability to fully engage in social interactions, education, and various aspects of daily life.
2. **Inclusivity**: The lack of an effective means to bridge this communication gap results in exclusion and isolation of Deaf individuals from the broader community. Inclusivity, accessibility, and equal opportunities are hindered.

3. **Education and Employment**: Effective communication is crucial for accessing education and employment opportunities. Without a reliable means of communicating with non-signers, Deaf individuals may face limitations in their educational and career advancement.
4. **Language Barrier**: ASL is a rich and complex language with its own grammar and syntax. Recognizing ASL alphabet signs accurately is essential to facilitate communication and promote understanding.
5. **Existing Solutions**: Although there are manual interpretation services and sign language interpreters, they are not always readily available and can be costly. An automated ASL alphabet recognition system can provide a more accessible and cost-effective solution.

The problem statement for the ASL Alphabet Recognition Project is to develop a computer vision-based solution that accurately recognizes ASL alphabet signs in real-time, enabling Deaf individuals to communicate effectively with non-signers, thus promoting inclusivity, accessibility, and greater participation in various aspects of life and society.

3. IDEATION & PROPOSED SOLUTION

3.1. EMPATHY MAP CANVAS



What does the user THINK and FEEL?

Thinks:
- "I want a tool that helps me easily communicate using ASL online."
- "I hope this technology can understand and interpret ASL signs accurately."
- "Will this project make ASL more accessible and inclusive?"

Feels:
- Excited about the potential for improved communication.
- Frustrated with the limitations of existing ASL recognition tools.
- Anxious about whether the project will be successful and helpful.

What does the user HEAR?

- About the project from Deaf community discussions and social media.
- Feedback and experiences from others who have tried similar tools.
- News about advancements in ASL recognition technology.

What does the user SEE?

- The need for a reliable ASL alphabet image recognition solution.
- Potential applications in education, work, and daily life.
- Promising results from early project demonstrations or trials.

What does the user SAY and DO?

Says:
- Expresses the desire for more accessible and accurate ASL communication tools.
- Shares feedback and experiences with peers and online communities.
- Asks questions about the project's progress and usability.

Does:
- Actively seeks out and tries the ASL recognition tool.
- Provides feedback to developers and offers suggestions for improvement.
- Integrates the tool into daily communication, such as during video calls or messaging.

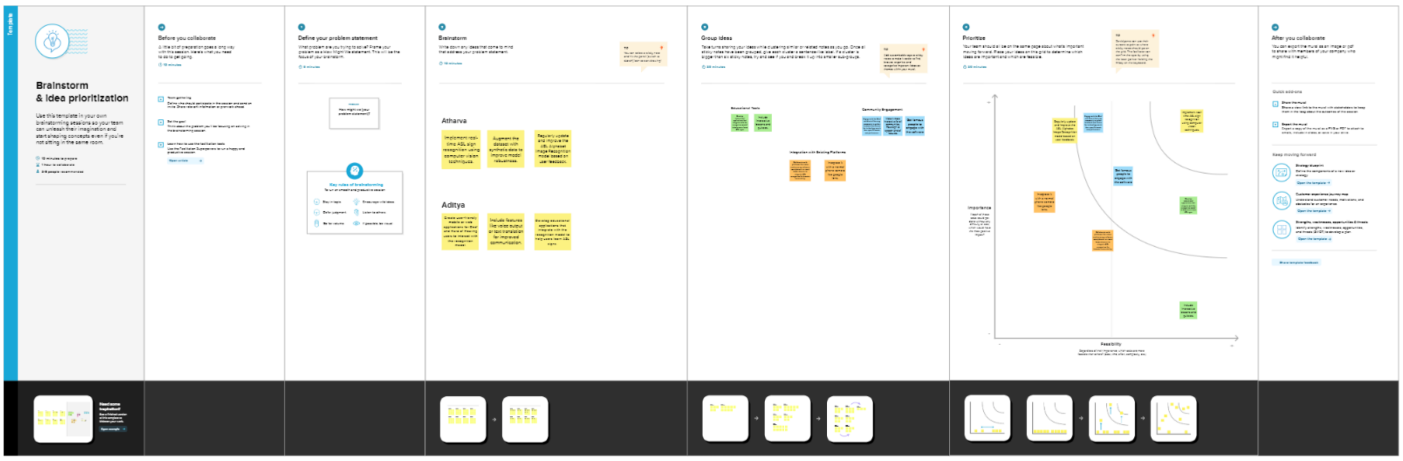
PAIN

Inaccurate or slow recognition.	Difficulty in accessing.	Limited exposure to ASL and Deaf culture.

GAIN

Improved communication	Greater independence.	Contributing to a more equitable society.

3.2. IDEATION AND BRAINSTORMING



4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENTS:

Functional Requirements for the ASL (American Sign Language) Alphabet Recognition Project:

1. **Data Collection and Preparation**: a. Collect a diverse dataset of ASL alphabet signs, including variations in lighting, backgrounds, and signers. b. Preprocess and standardize the dataset, ensuring consistent image format and quality. c. Implement data augmentation techniques to enhance the dataset diversity.
2. **Model Development**: a. Design and develop a deep learning model for ASL alphabet sign recognition, which may include Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). b. Implement a user-friendly interface for configuring and training the recognition model. c. Integrate pre-trained models if available to expedite development.
3. **Training and Validation**: a. Train the recognition model using the prepared dataset. b. Implement model validation with appropriate metrics (e.g., accuracy, precision, recall) to ensure robust performance.
4. **Real-time Detection**: a. Develop a module for real-time ASL alphabet sign recognition using a webcam or other video input source. b. Ensure low latency and high frame-per-second (FPS) performance for real-time interaction.
5. **User Interface**: a. Create a user-friendly and intuitive interface for interaction with the system. b. Include options for users to initiate recognition, configure settings, and view results.
6. **Integration with Text or Speech Output**: a. Convert recognized ASL signs into corresponding text or synthesized speech. b. Ensure seamless integration with other communication tools or applications.
7. **Accessibility Features**: a. Implement accessibility features, such as voice commands and compatibility with screen readers, to cater to users with disabilities. b. Ensure user interface elements are customizable for various accessibility needs.
8. **Testing and Evaluation**: a. Conduct comprehensive testing, including performance evaluation in different lighting and environmental conditions. b. Engage users from the ASL community to evaluate the system's accuracy, usability, and overall satisfaction.
9. **Documentation**: a. Provide comprehensive documentation, including installation instructions, configuration guidelines, and system usage guides. b. Include troubleshooting resources and user support information.
10. **Deployment**: a. Deploy the ASL alphabet recognition system on a platform of choice (e.g., desktop application, mobile app, web service). b. Ensure the system is stable and available for end-users.
11. **Security and Privacy**: a. Implement security measures to protect user data and system functionality. b. Ensure that user data, especially images and interpreted signs, are handled with privacy and confidentiality in mind.
12. **Scalability**: a. Design the system architecture to be scalable, allowing for future enhancements and accommodating a growing user base. b. Ensure the model is adaptable to new ASL signs and regional variations.

These functional requirements are essential for the successful development of an ASL Alphabet Recognition System, ensuring accurate, accessible, and user-friendly communication between ASL users and non-signers.

4.2. NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirements for the ASL (American Sign Language) Alphabet Recognition Project:

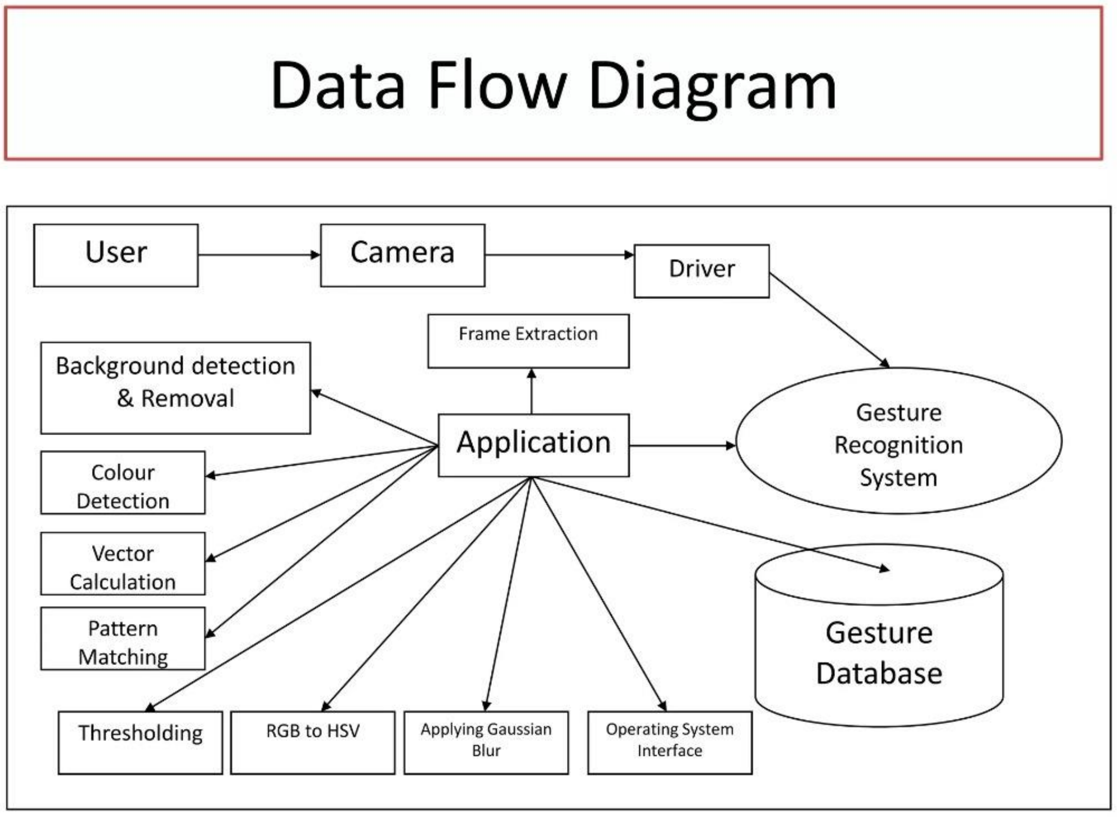
1. **Performance**:
   - **Real-time Response**: The system should provide real-time recognition with low latency, aiming for a response time of less than 200 milliseconds.
   - **Accuracy**: The recognition model should achieve a high level of accuracy, with a target accuracy rate of 95% or higher.
2. **Scalability**:
   - The system should be designed to handle a growing number of users and an expanding dataset of ASL signs.
3. **Reliability**:
   - The system should be highly reliable, with minimal downtime and robust error handling.
4. **Usability**:
   - The user interface should be intuitive and easy to use for individuals with varying levels of technical expertise.
   - Accessibility: The system should be accessible to users with disabilities, including those who use screen readers or voice commands.
5. **Security**:
   - User Data Protection: Ensure the security of user data, including images and interpreted signs, and comply with data protection regulations.
   - Model Security: Protect the recognition model from unauthorized access or tampering.
6. **Privacy**:
   - User Privacy: Maintain the privacy of users, and ensure that data is not used for any purpose other than ASL recognition.
7. **Interoperability**:
   - The system should be designed to integrate with other communication tools and platforms, allowing users to easily incorporate ASL recognition into their existing workflows.
8. **Compliance**:
   - Ensure compliance with relevant accessibility standards and regulations, such as the Americans with Disabilities Act (ADA).
9. **Resource Efficiency**:
   - Optimize resource usage to ensure efficient performance, especially for resource-constrained devices or environments.
10. **Cultural Sensitivity**:
    - Ensure that the system is culturally sensitive, respecting the diverse nature of ASL and regional sign language variations.
11. **Maintenance and Support**:
    - Provide regular maintenance, updates, and user support to address issues and improve the system over time.
12. **Documentation**:
    - Offer comprehensive documentation that includes clear installation, configuration, and usage instructions, as well as information on troubleshooting and updates.

13. **Testing and Validation**:
    - Implement thorough testing procedures, including usability testing and robustness testing under various environmental conditions.
14. **Cost**:
    - Maintain cost-effectiveness in terms of both development and deployment, especially considering potential adoption in educational and healthcare settings.
15. **Localization**:
    - Plan for the system's adaptability to different languages and regions, considering variations in sign language.
16. **Ethical Considerations**:
    - Address ethical concerns related to data collection, privacy, and the potential societal impact of ASL recognition.

These non-functional requirements are essential for ensuring the success and effectiveness of the ASL Alphabet Recognition Project, addressing aspects of performance, reliability, security, usability, and compliance, among others.

5. PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS AND USER STORIES



USER STORIES

User Type	Functional Requirement	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
ASL Learner	Recognition	US-01	Sign Recognition	As an ASL learner, I want to sign a word or phrase, and the system should recognize and display the corresponding ASL sign or translation.	High	v1.0
ASL Teacher	Verification	US-02	Student Sign Verification	As an ASL teacher, I want to verify and correct my students' ASL signs using the system.	High	v1.0
Deaf User	Translation	US-03	Sign-to-Text Translation	As a deaf user, I want the system to convert ASL signs to text or speech to assist in	High	v1.1

				communication with hearing individuals.		
System Administrator	Monitoring	US-04	System Performance Monitoring	As a system administrator, I want to monitor the system's performance and usage statistics.	Medium	v1.1
Researcher	Data Collection	US-05	User Interaction Data Collection	As a researcher, I want to collect data on user interactions and system performance for analysis.	Medium	v1.2
Mobile App User	Mobile Accessibility	US-06	Mobile App Access	As a mobile app user, I want to access the ASL sign recognition system on my smartphone.	High	v1.2
Web App User	Web Accessibility	US-07	Web App Access	As a web app user, I want to access the ASL sign recognition system through my web browser.	High	v1.2
Security Officer	Data Security	US-08	Data Security & Privacy	As a security officer, I want user data, especially video streams, to be securely processed and stored.	High	v1.3
Scaling Needs	Performance	US-09	System Scalability	As the project scales, I want the system to handle an increasing number of users without compromising recognition accuracy or performance.	High	v1.3

User Feedback	Feedback Mechanism	US-10	Built-in Feedback Mechanism	As a user, I want a built-in feedback mechanism to report recognition inaccuracies and other issues.	Medium	v1.4
User Profiles	Personalization	US-11	User Profile Management	As a user, I want the system to store my preferences and usage history for personalization.	Medium	v1.4
Sign Language Interpreter	Real-time Interpretation	US-12	Real-time Interpretation	As a sign language interpreter, I want the system to assist me in real-time interpretation.	High	v1.5

5.2. SOLUTION ARCHITECTURE

Initial Setup

- Create a new GitHub repository #GitHub
    - Initialize with README.md
    - Add. gitignore
    - Set up branch protection rules
- Set up virtual environment #Python
    - Install necessary packages
        - NumPy
        - Pandas
        - TensorFlow
        - Keras
        - Matplotlib
    - Verify package installation

Data Collection

- Collect ASL sign images #ImageData
    - Define sign categories
    - Source images
    - Verify image quality

Data Preprocessing

- Clean and preprocess image data #DataCleaning
    - Resize images
    - Normalize pixel values
    - Split data into training and test sets

Model Building

- Build Convolutional Neural Network (CNN) model #CNNModel
    - Define model architecture
    - Compile model
    - Train model
    - Evaluate model performance

Deployment

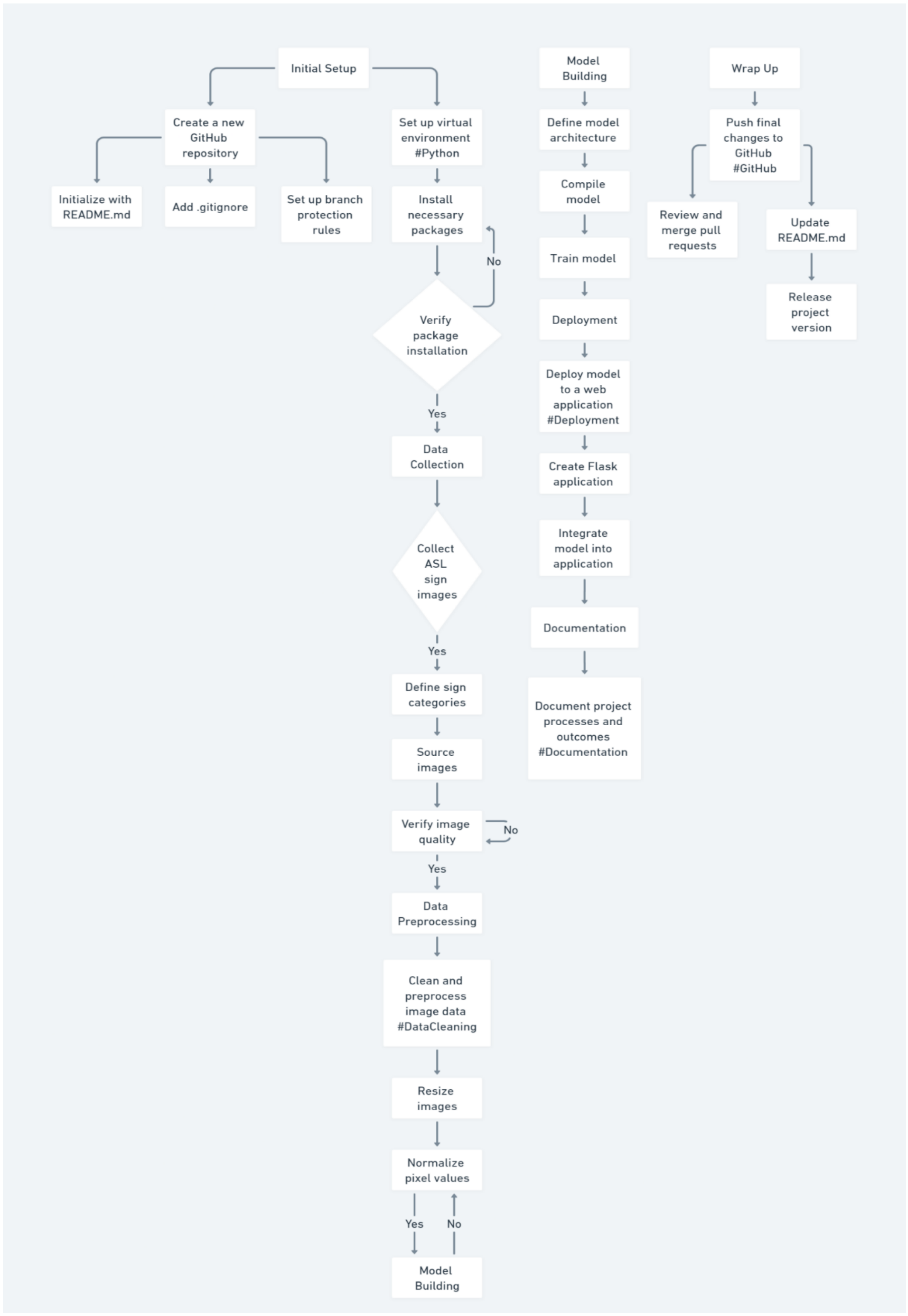
- Deploy model to a web application #Deployment
    - Create Flask application
    - Integrate model into application
    - Test application functionality

Documentation

- Document project processes and outcomes #Documentation
    1. Write project summary
    2. Document data collection process
    3. Explain data preprocessing steps
    4. Detail model architecture and training process
    5. Discuss model performance
    6. Describe deployment steps
    7. Reflect on project outcomes and potential improvements

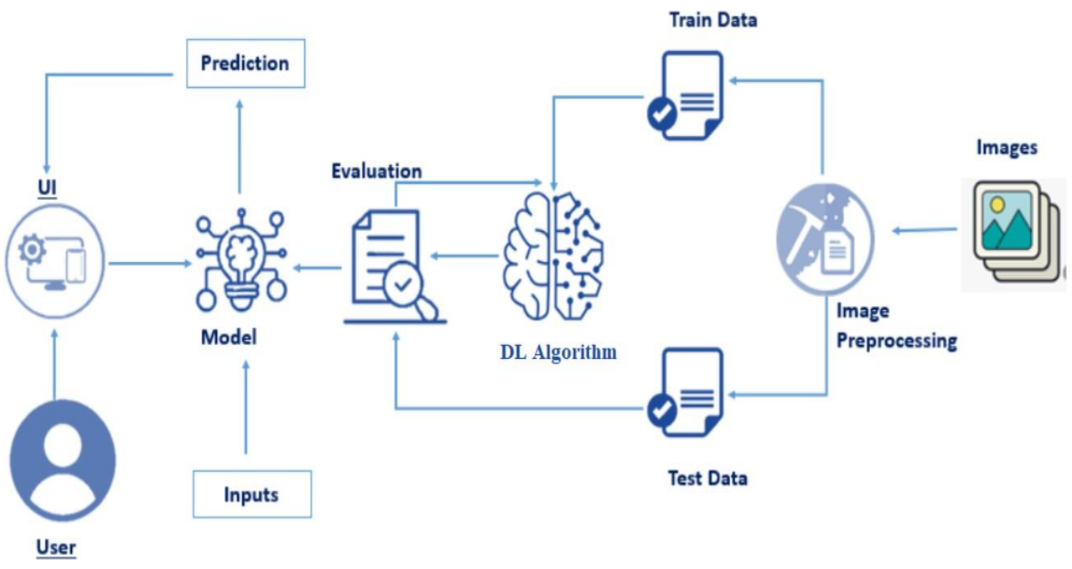
Wrap Up

- Push final changes to GitHub #GitHub
    - Review and merge pull requests
    - Update README.md
    - Release project version


Initial Setup
Create a new GitHub repository
Initialize with README.md
Add .gitignore
Set up branch protection rules
Set up virtual environment #Python
Install necessary packages
No
Verify package installation
Yes
Data Collection
Collect ASL sign images
Yes
Define sign categories
Source images
Verify image quality
No
Yes
Data Preprocessing
Clean and preprocess image data #DataCleaning
Resize images
Normalize pixel values
Yes
No
Model Building
Model Building
Define model architecture
Compile model
Train model
Deployment
Deploy model to a web application #Deployment
Create Flask application
Integrate model into application
Documentation
Document project processes and outcomes #Documentation
Wrap Up
Push final changes to GitHub #GitHub
Review and merge pull requests
Update README.md
Release project version

6. PROJECT PLANNING & SCHEDULING

6.1. TECHNICAL ARCHITECTURE

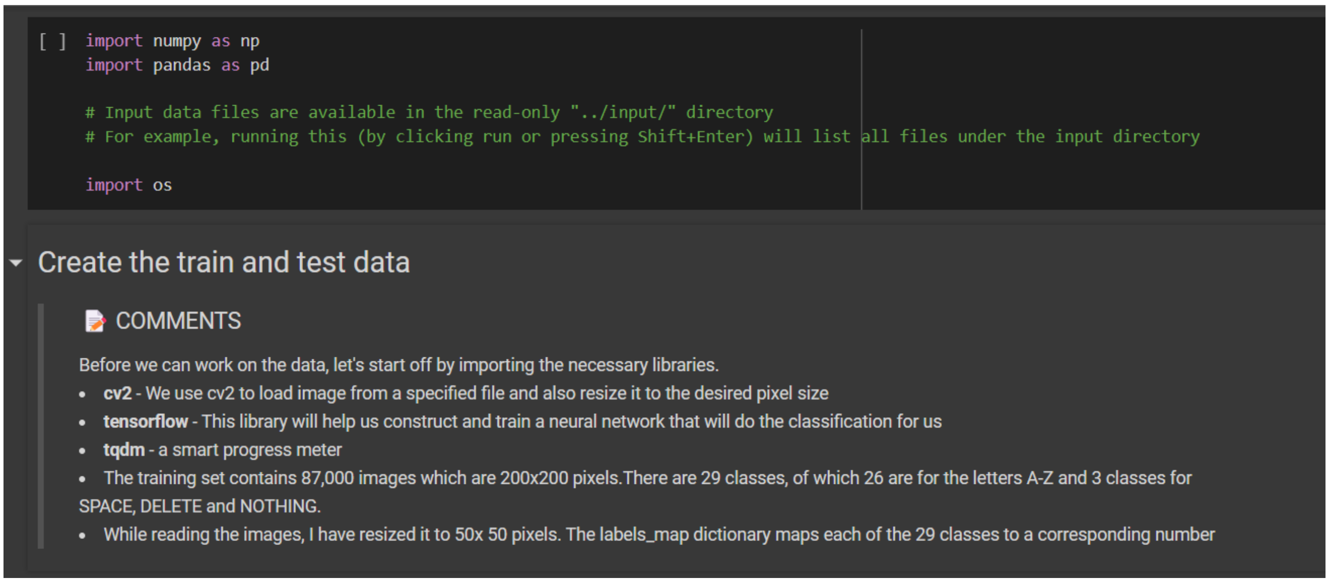


6.2. SPRINT PLANNING ESTIMATION AND DELIVERY SCEHDULE

Sprint	Functional Requirement	User Story Number	User Story/Task	Story Points	Priority	Team Members
Sprint 1	Setup ASL Recognition System	US1	Define project scope and objectives	2	High	Atharva
		US2	Set up development environment	3	High	Aditya
		US3	Create a dataset of ASL alphabet signs	5	High	Aditya
		US4	Implement image preprocessing	8	Medium	Atharva
		US5	Train a deep learning model for recognition	13	High	Atharva

		US6	Test and validate the model	8	High	Aditya
		US7	Develop a simple user interface	5	Medium	Atharva
		US8	Integrate recognition with the UI	5	Medium	Aditya
Sprint 2	Real-time Recognition	US9	Implement real-time video feed	8	High	Aditya
		US10	Integrate real-time recognition	13	High	Aditya
		US11	Fine-tune recognition accuracy	8	Medium	Atharva
		US12	Testing and bug fixing	5	High	Aditya
		US13	Documentation and user manual	3	Low	Atharva
		US14	Final testing and optimization	8	High	Atharva

7. CODING AND SOLUTIONING



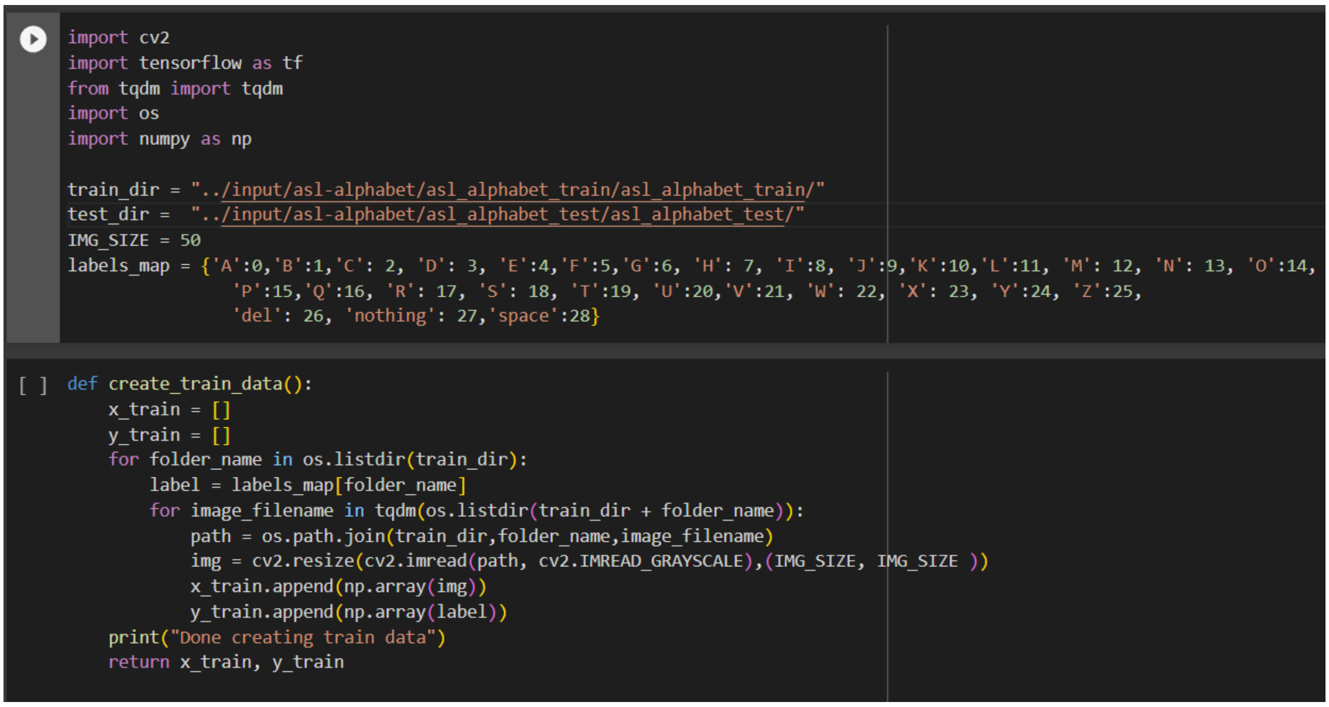
```
[ ] import numpy as np
    import pandas as pd

    # Input data files are available in the read-only "../input/" directory
    # For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

    import os
```

Create the train and test data

COMMENTS

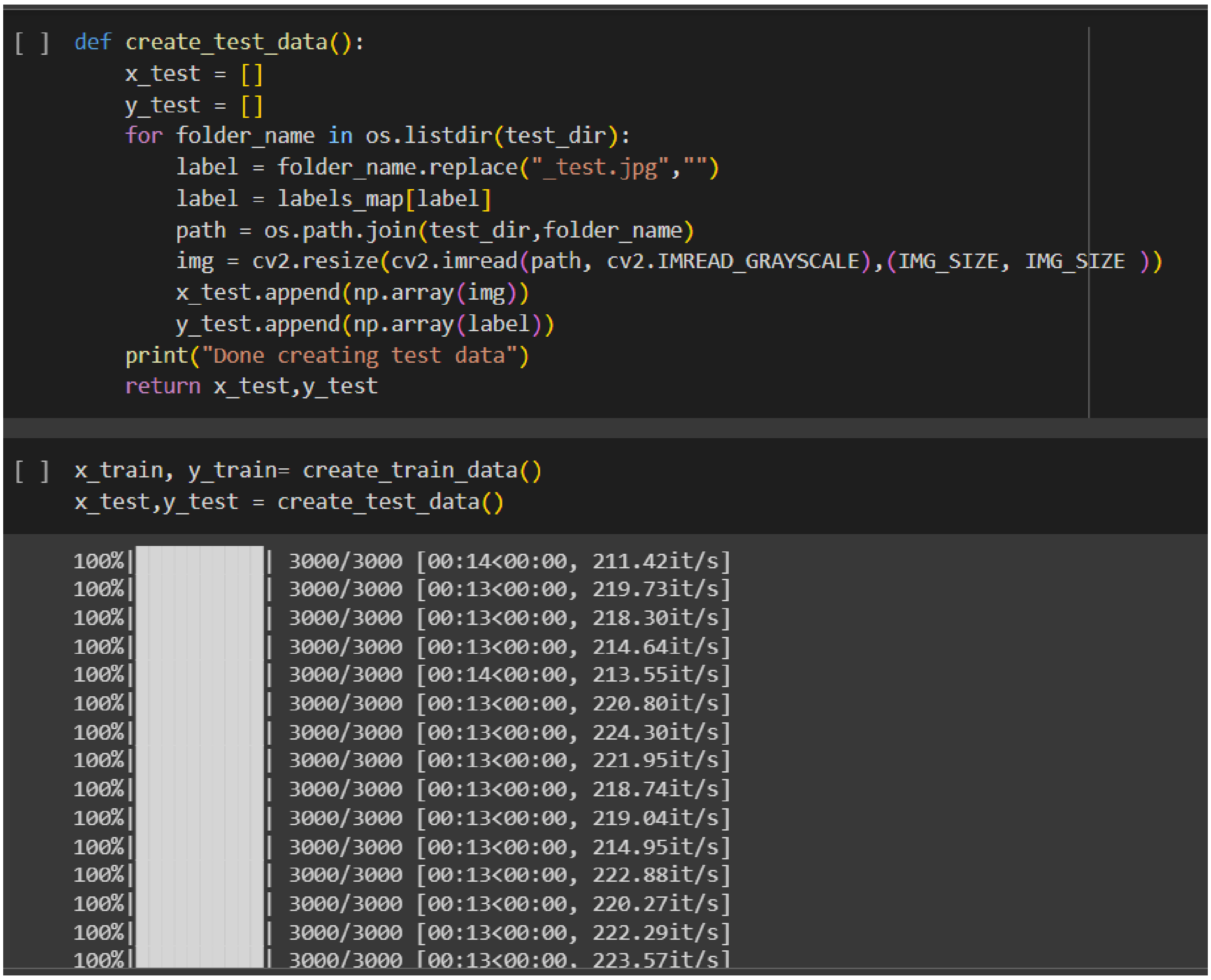
Before we can work on the data, let's start off by importing the necessary libraries.

- **cv2** - We use cv2 to load image from a specified file and also resize it to the desired pixel size
- **tensorflow** - This library will help us construct and train a neural network that will do the classification for us
- **tqdm** - a smart progress meter
- The training set contains 87,000 images which are 200x200 pixels.There are 29 classes, of which 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING.
- While reading the images, I have resized it to 50x 50 pixels. The labels_map dictionary maps each of the 29 classes to a corresponding number



```
import cv2
import tensorflow as tf
from tqdm import tqdm
import os
import numpy as np

train_dir = "../input/asl-alphabet/asl_alphabet_train/asl_alphabet_train/"
test_dir =  "../input/asl-alphabet/asl_alphabet_test/asl_alphabet_test/"
IMG_SIZE = 50
labels_map = {'A':0,'B':1,'C': 2, 'D': 3, 'E':4,'F':5,'G':6, 'H': 7, 'I':8, 'J':9,'K':10,'L':11, 'M': 12, 'N': 13, 'O':14,
                'P':15,'Q':16, 'R': 17, 'S': 18, 'T':19, 'U':20,'V':21, 'W': 22, 'X': 23, 'Y':24, 'Z':25,
                'del': 26, 'nothing': 27,'space':28}
```

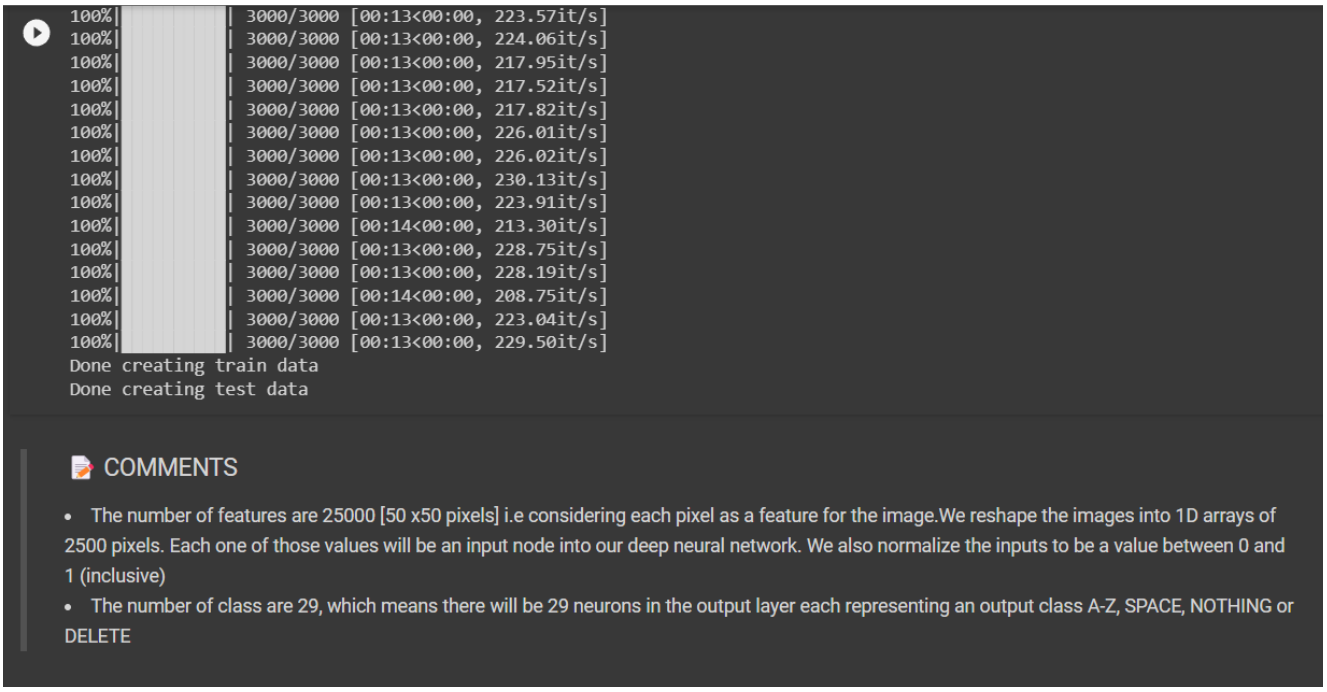
```
[ ] def create_train_data():
        x_train = []
        y_train = []
        for folder_name in os.listdir(train_dir):
            label = labels_map[folder_name]
            for image_filename in tqdm(os.listdir(train_dir + folder_name)):
                path = os.path.join(train_dir,folder_name,image_filename)
                img = cv2.resize(cv2.imread(path, cv2.IMREAD_GRAYSCALE),(IMG_SIZE, IMG_SIZE ))
                x_train.append(np.array(img))
                y_train.append(np.array(label))
        print("Done creating train data")
        return x_train, y_train
```



```
[ ]  def create_test_data():
         x_test = []
         y_test = []
         for folder_name in os.listdir(test_dir):
             label = folder_name.replace("_test.jpg","")
             label = labels_map[label]
             path = os.path.join(test_dir,folder_name)
             img = cv2.resize(cv2.imread(path, cv2.IMREAD_GRAYSCALE),(IMG_SIZE, IMG_SIZE ))
             x_test.append(np.array(img))
             y_test.append(np.array(label))
         print("Done creating test data")
         return x_test,y_test
```

```
[ ]  x_train, y_train= create_train_data()
     x_test,y_test = create_test_data()
```

```
100%|██████████| 3000/3000 [00:14<00:00, 211.42it/s]
100%|██████████| 3000/3000 [00:13<00:00, 219.73it/s]
100%|██████████| 3000/3000 [00:13<00:00, 218.30it/s]
100%|██████████| 3000/3000 [00:13<00:00, 214.64it/s]
100%|██████████| 3000/3000 [00:14<00:00, 213.55it/s]
100%|██████████| 3000/3000 [00:13<00:00, 220.80it/s]
100%|██████████| 3000/3000 [00:13<00:00, 224.30it/s]
100%|██████████| 3000/3000 [00:13<00:00, 221.95it/s]
100%|██████████| 3000/3000 [00:13<00:00, 218.74it/s]
100%|██████████| 3000/3000 [00:13<00:00, 219.04it/s]
100%|██████████| 3000/3000 [00:13<00:00, 214.95it/s]
100%|██████████| 3000/3000 [00:13<00:00, 222.88it/s]
100%|██████████| 3000/3000 [00:13<00:00, 220.27it/s]
100%|██████████| 3000/3000 [00:13<00:00, 222.29it/s]
100%|██████████| 3000/3000 [00:13<00:00, 223.57it/s]
```



```
100%|██████████| 3000/3000 [00:13<00:00, 223.57it/s]
100%|██████████| 3000/3000 [00:13<00:00, 224.06it/s]
100%|██████████| 3000/3000 [00:13<00:00, 217.95it/s]
100%|██████████| 3000/3000 [00:13<00:00, 217.52it/s]
100%|██████████| 3000/3000 [00:13<00:00, 217.82it/s]
100%|██████████| 3000/3000 [00:13<00:00, 226.01it/s]
100%|██████████| 3000/3000 [00:13<00:00, 226.02it/s]
100%|██████████| 3000/3000 [00:13<00:00, 230.13it/s]
100%|██████████| 3000/3000 [00:13<00:00, 223.91it/s]
100%|██████████| 3000/3000 [00:14<00:00, 213.30it/s]
100%|██████████| 3000/3000 [00:13<00:00, 228.75it/s]
100%|██████████| 3000/3000 [00:13<00:00, 228.19it/s]
100%|██████████| 3000/3000 [00:14<00:00, 208.75it/s]
100%|██████████| 3000/3000 [00:13<00:00, 223.04it/s]
100%|██████████| 3000/3000 [00:13<00:00, 229.50it/s]
Done creating train data
Done creating test data
```

COMMENTS

- The number of features are 25000 [50 x50 pixels] i.e considering each pixel as a feature for the image.We reshape the images into 1D arrays of 2500 pixels. Each one of those values will be an input node into our deep neural network. We also normalize the inputs to be a value between 0 and 1 (inclusive)
- The number of class are 29, which means there will be 29 neurons in the output layer each representing an output class A-Z, SPACE, NOTHING or DELETE

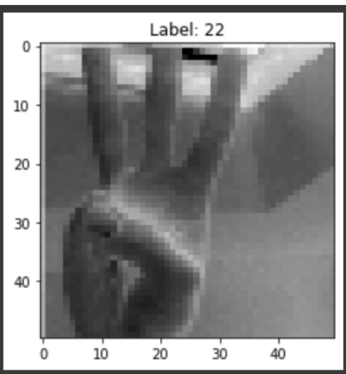
```
[ ] num_features = 2500
    num_classes = 29

    x_train, x_test = np.array(x_train, np.float32), np.array(x_test, np.float32)
    x_train, x_test = x_train.reshape([-1, num_features]), x_test.reshape([-1, num_features])
    x_train, x_test = x_train / 255., x_test / 255.
```

COMMENTS

- The training images are therefore a tensor of shape [87,000, 25000]. The training labels are a one-dimensional tensor of 87,000 labels that range from 0 to 28.
- Let's get a feel of what the images look like compressed to 50 x 50 pixels and in gray scale. Remember we have compressed it from 200 x 200 to 50 x 50. Looking at the image below it is a bit pixellated; but still looks good!

```
[ ] %matplotlib inline
    import matplotlib.pyplot as plt

    def display_image(num):
        label = y_train[num]
        plt.title('Label: %d' % (label))
        image = x_train[num].reshape([IMG_SIZE,IMG_SIZE])
        plt.imshow(image, cmap=plt.get_cmap('gray_r'))
        plt.show()
    display_image(5)
```

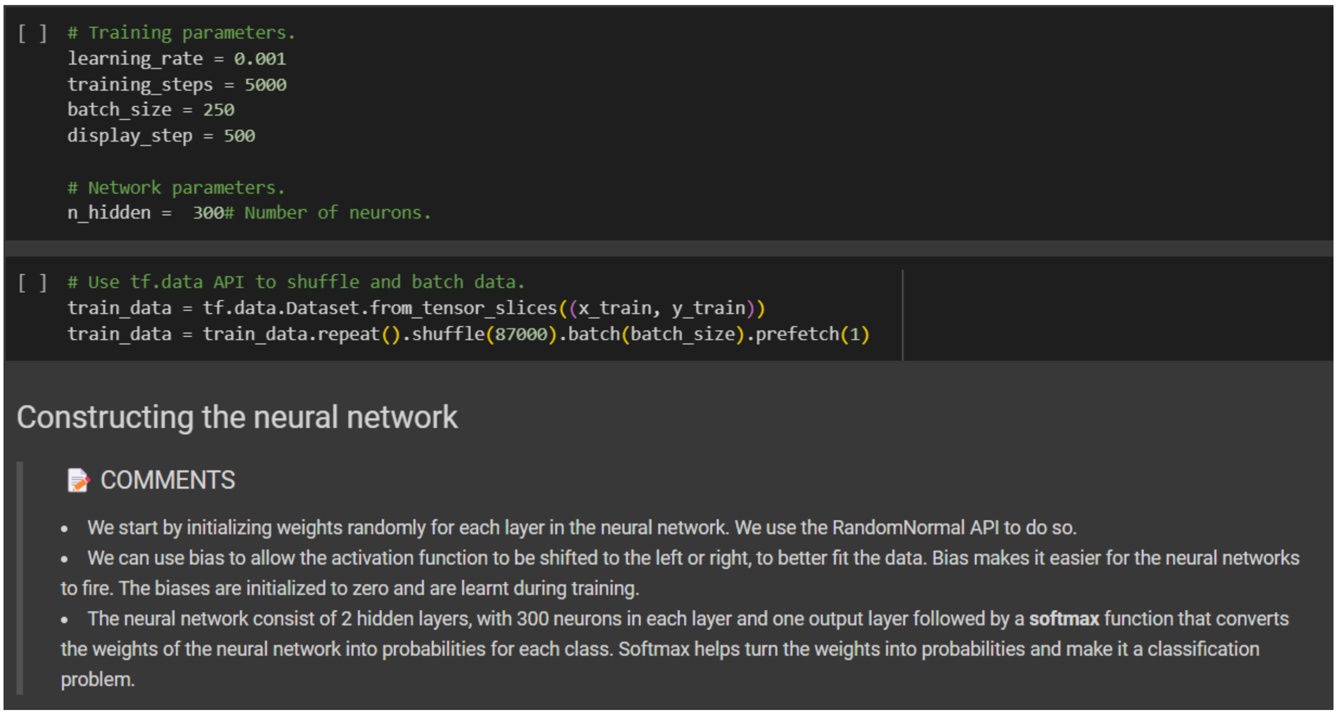
[ ]



Determining training and network parameters

COMMENTS

- These parameters or "hyperparameters" are ones we have to experiment with to improve upon the **accuracy** of the neural network. A little tweak could result in a huge difference!
- **learning rate** - controls how much to change the model in response to the estimated error each time the model weights are updated.
- **training steps** - number of epochs
- **batch size** - small chunks for each iteration of training
- **n hidden** - number of neurons in the hidden layer
- We use tf.data API to shuffle the data and divide it into batches



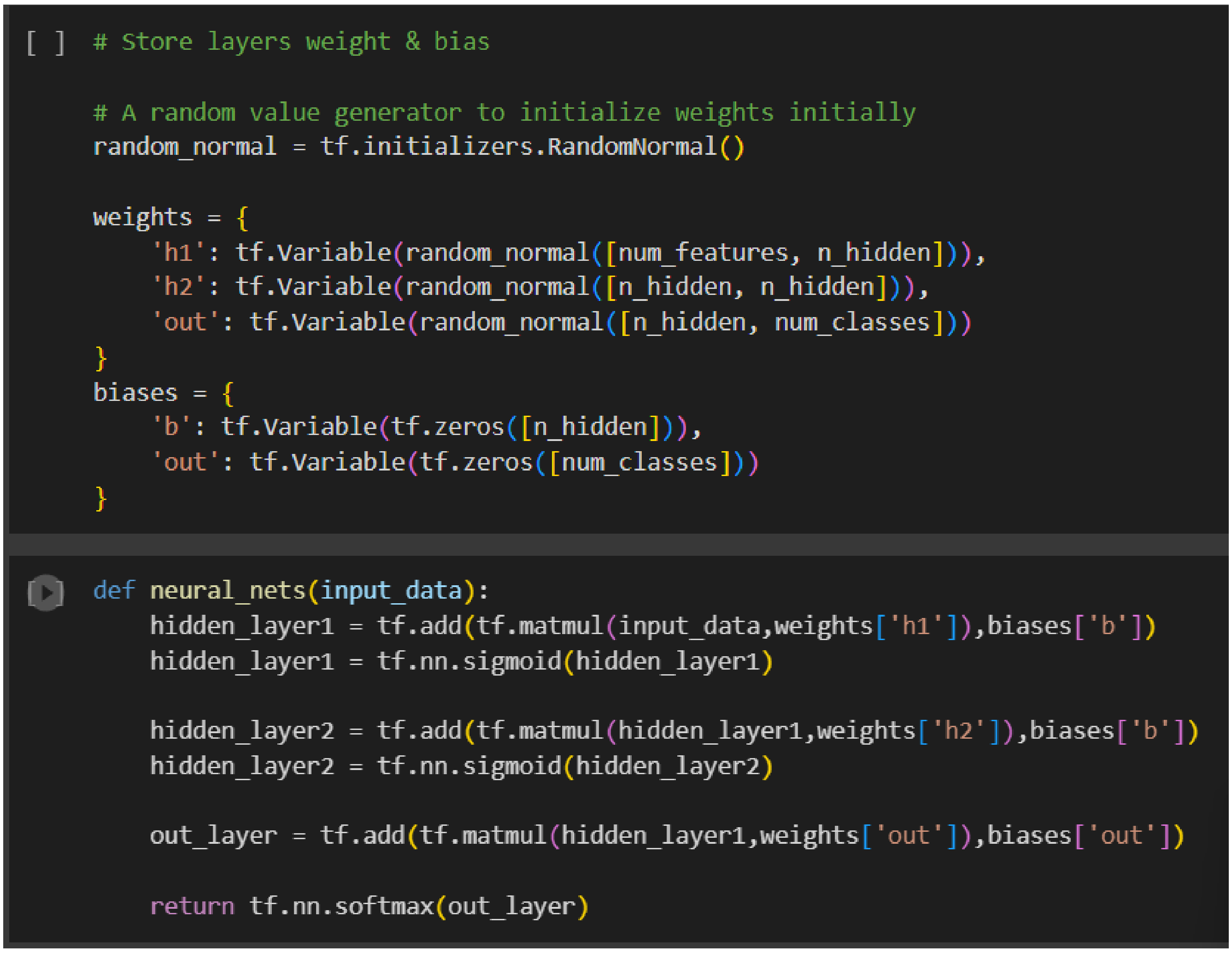
```
[ ] # Training parameters.
    learning_rate = 0.001
    training_steps = 5000
    batch_size = 250
    display_step = 500

    # Network parameters.
    n_hidden =  300# Number of neurons.
```

```
[ ] # Use tf.data API to shuffle and batch data.
    train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train))
    train_data = train_data.repeat().shuffle(87000).batch(batch_size).prefetch(1)
```

Constructing the neural network

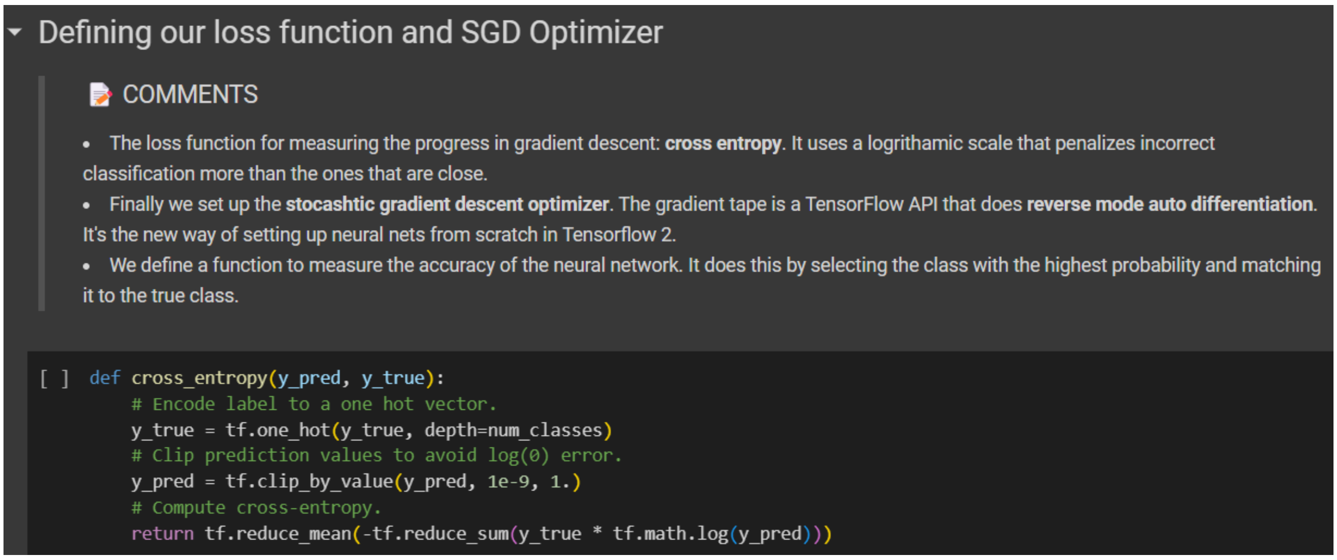
COMMENTS

- We start by initializing weights randomly for each layer in the neural network. We use the RandomNormal API to do so.
- We can use bias to allow the activation function to be shifted to the left or right, to better fit the data. Bias makes it easier for the neural networks to fire. The biases are initialized to zero and are learnt during training.
- The neural network consist of 2 hidden layers, with 300 neurons in each layer and one output layer followed by a **softmax** function that converts the weights of the neural network into probabilities for each class. Softmax helps turn the weights into probabilities and make it a classification problem.



```
[ ] # Store layers weight & bias

    # A random value generator to initialize weights initially
    random_normal = tf.initializers.RandomNormal()

    weights = {
        'h1': tf.Variable(random_normal([num_features, n_hidden])),
        'h2': tf.Variable(random_normal([n_hidden, n_hidden])),
        'out': tf.Variable(random_normal([n_hidden, num_classes]))
    }
    biases = {
        'b': tf.Variable(tf.zeros([n_hidden])),
        'out': tf.Variable(tf.zeros([num_classes]))
    }
```

```
    def neural_nets(input_data):
        hidden_layer1 = tf.add(tf.matmul(input_data,weights['h1']),biases['b'])
        hidden_layer1 = tf.nn.sigmoid(hidden_layer1)

        hidden_layer2 = tf.add(tf.matmul(hidden_layer1,weights['h2']),biases['b'])
        hidden_layer2 = tf.nn.sigmoid(hidden_layer2)

        out_layer = tf.add(tf.matmul(hidden_layer1,weights['out']),biases['out'])

        return tf.nn.softmax(out_layer)
```

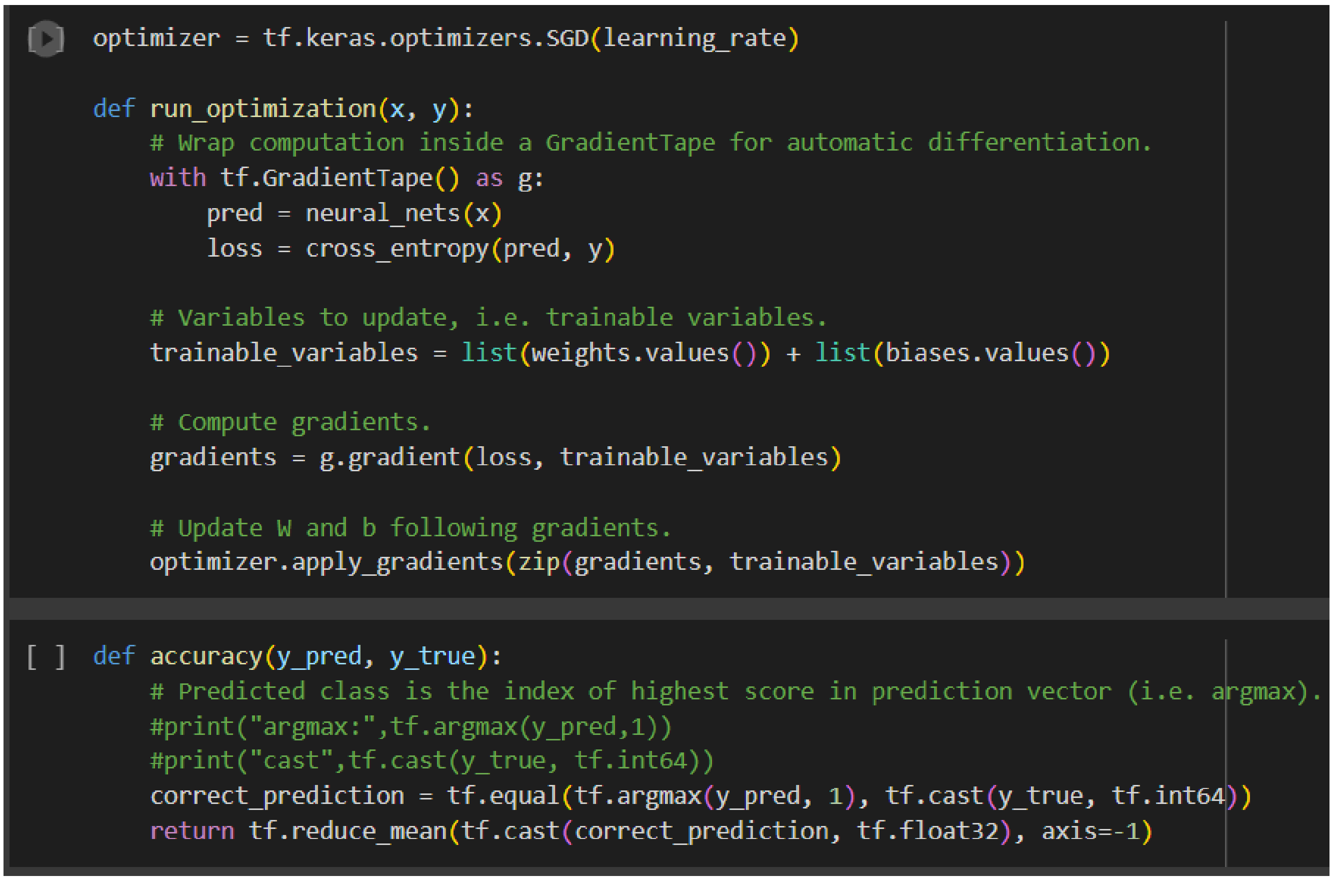


Defining our loss function and SGD Optimizer

COMMENTS

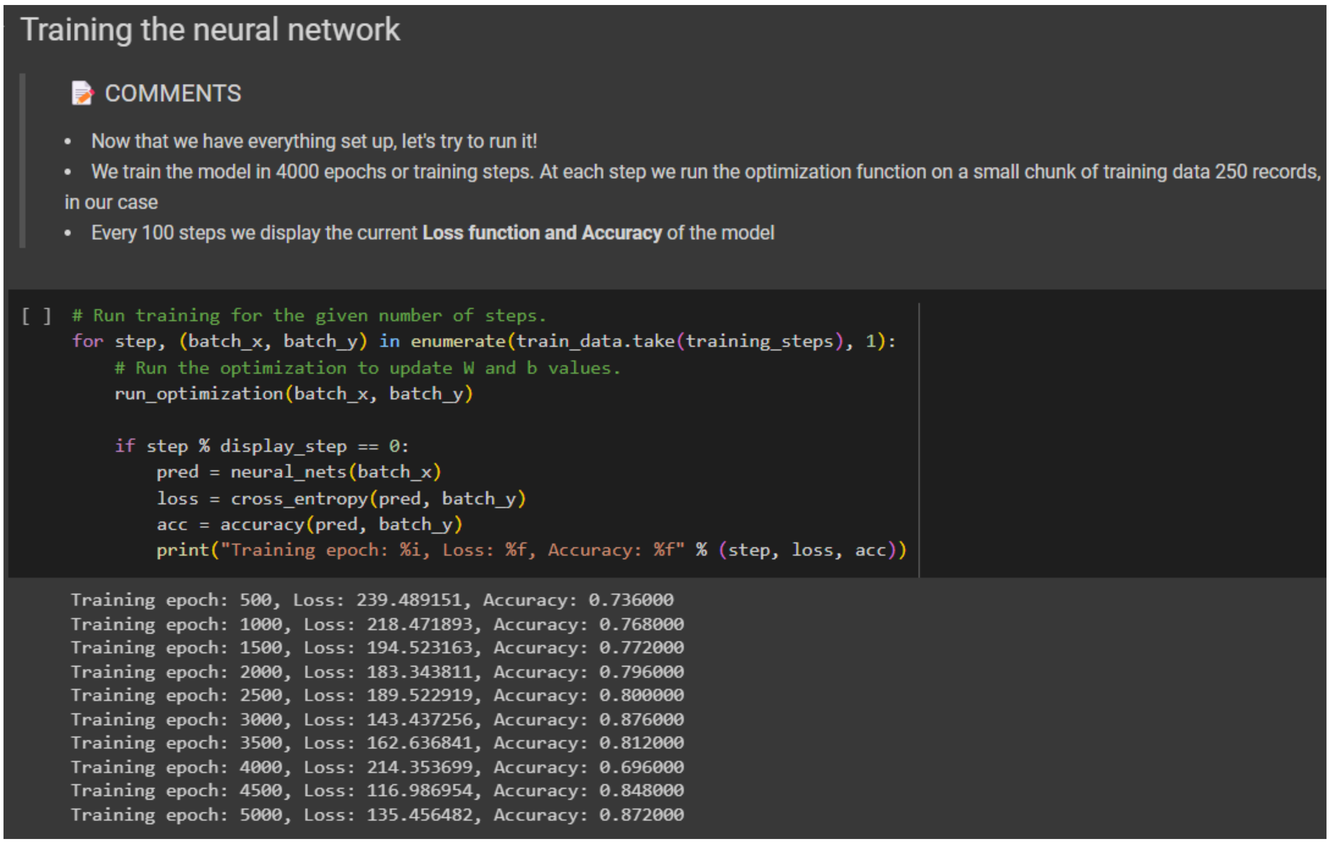
- The loss function for measuring the progress in gradient descent: **cross entropy**. It uses a logrithamic scale that penalizes incorrect classification more than the ones that are close.
- Finally we set up the **stocashtic gradient descent optimizer**. The gradient tape is a TensorFlow API that does **reverse mode auto differentiation**. It's the new way of setting up neural nets from scratch in Tensorflow 2.
- We define a function to measure the accuracy of the neural network. It does this by selecting the class with the highest probability and matching it to the true class.

```
[ ] def cross_entropy(y_pred, y_true):
        # Encode label to a one hot vector.
        y_true = tf.one_hot(y_true, depth=num_classes)
        # Clip prediction values to avoid log(0) error.
        y_pred = tf.clip_by_value(y_pred, 1e-9, 1.)
        # Compute cross-entropy.
        return tf.reduce_mean(-tf.reduce_sum(y_true * tf.math.log(y_pred)))
```



```
optimizer = tf.keras.optimizers.SGD(learning_rate)

def run_optimization(x, y):
    # Wrap computation inside a GradientTape for automatic differentiation.
    with tf.GradientTape() as g:
        pred = neural_nets(x)
        loss = cross_entropy(pred, y)

    # Variables to update, i.e. trainable variables.
    trainable_variables = list(weights.values()) + list(biases.values())

    # Compute gradients.
    gradients = g.gradient(loss, trainable_variables)

    # Update W and b following gradients.
    optimizer.apply_gradients(zip(gradients, trainable_variables))
```

```
[ ] def accuracy(y_pred, y_true):
        # Predicted class is the index of highest score in prediction vector (i.e. argmax).
        #print("argmax:",tf.argmax(y_pred,1))
        #print("cast",tf.cast(y_true, tf.int64))
        correct_prediction = tf.equal(tf.argmax(y_pred, 1), tf.cast(y_true, tf.int64))
        return tf.reduce_mean(tf.cast(correct_prediction, tf.float32), axis=-1)
```



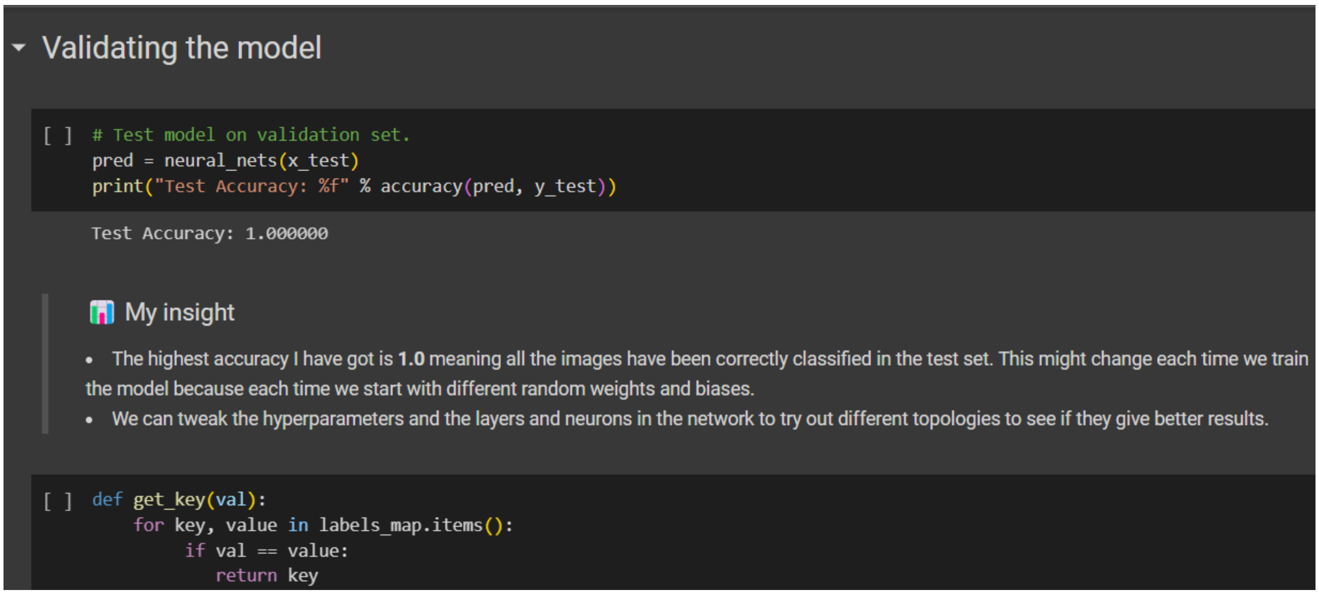
Training the neural network

COMMENTS

- Now that we have everything set up, let's try to run it!
- We train the model in 4000 epochs or training steps. At each step we run the optimization function on a small chunk of training data 250 records, in our case
- Every 100 steps we display the current **Loss function and Accuracy** of the model

```
[ ]  # Run training for the given number of steps.
     for step, (batch_x, batch_y) in enumerate(train_data.take(training_steps), 1):
         # Run the optimization to update W and b values.
         run_optimization(batch_x, batch_y)

         if step % display_step == 0:
             pred = neural_nets(batch_x)
             loss = cross_entropy(pred, batch_y)
             acc = accuracy(pred, batch_y)
             print("Training epoch: %i, Loss: %f, Accuracy: %f" % (step, loss, acc))
```

```
     Training epoch: 500, Loss: 239.489151, Accuracy: 0.736000
     Training epoch: 1000, Loss: 218.471893, Accuracy: 0.768000
     Training epoch: 1500, Loss: 194.523163, Accuracy: 0.772000
     Training epoch: 2000, Loss: 183.343811, Accuracy: 0.796000
     Training epoch: 2500, Loss: 189.522919, Accuracy: 0.800000
     Training epoch: 3000, Loss: 143.437256, Accuracy: 0.876000
     Training epoch: 3500, Loss: 162.636841, Accuracy: 0.812000
     Training epoch: 4000, Loss: 214.353699, Accuracy: 0.696000
     Training epoch: 4500, Loss: 116.986954, Accuracy: 0.848000
     Training epoch: 5000, Loss: 135.456482, Accuracy: 0.872000
```



Validating the model

```
[ ]  # Test model on validation set.
     pred = neural_nets(x_test)
     print("Test Accuracy: %f" % accuracy(pred, y_test))
```

```
     Test Accuracy: 1.000000
```

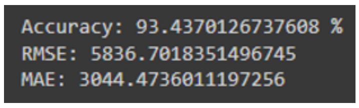
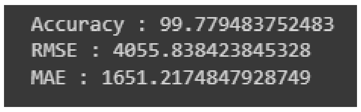
My insight

- The highest accuracy I have got is **1.0** meaning all the images have been correctly classified in the test set. This might change each time we train the model because each time we start with different random weights and biases.
- We can tweak the hyperparameters and the layers and neurons in the network to try out different topologies to see if they give better results.

```
[ ]  def get_key(val):
         for key, value in labels_map.items():
              if val == value:
                 return key
```

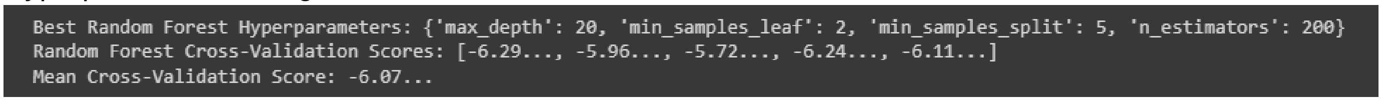
8. <u>PERFORMANCE TESTING</u>

Sr. No.	Parameter	Values	Screenshot

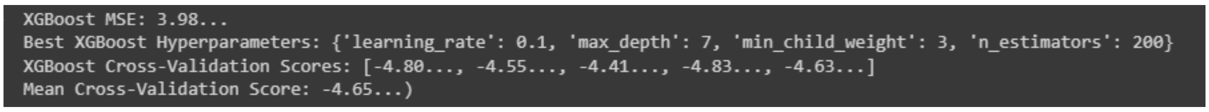
1.	Metrics	**XgBoost Model:**
		MAE - 3044.4736,
		RMSE – 5836.70183,
		Accuracy – 93.43701
		
		Random Forest:
		Accuray Score - 99.779483
		RMSE – 4055.838423
		MAE – 1651.2174874
			
			`Accuracy: 93.4370126737608 %`
			`RMSE: 5836.7018351496745`
			`MAE: 3044.4736011197256`
			
			`Accuracy : 99.779483752483`
			`RMSE : 4055.838423845328`
			`MAE : 1651.2174847928749`

2.	Tune the Model	Hyperparameter Tuning Validation Method - Cross validation	

Hyperparameter Tuning and Cross validation:

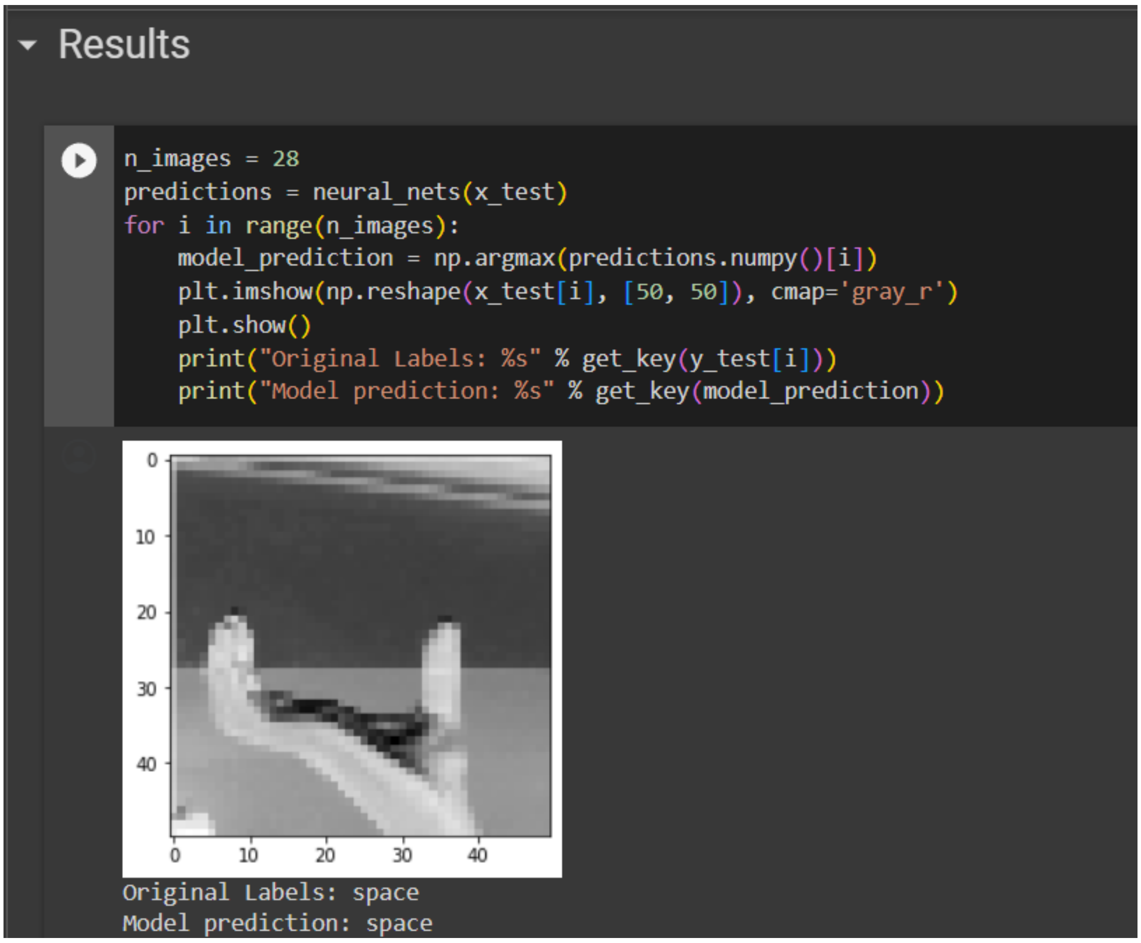


```
Best Random Forest Hyperparameters: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}
Random Forest Cross-Validation Scores: [-6.29..., -5.96..., -5.72..., -6.24..., -6.11...]
Mean Cross-Validation Score: -6.07...
```



```
XGBoost MSE: 3.98...
Best XGBoost Hyperparameters: {'learning_rate': 0.1, 'max_depth': 7, 'min_child_weight': 3, 'n_estimators': 200}
XGBoost Cross-Validation Scores: [-4.80..., -4.55..., -4.41..., -4.83..., -4.63...]
Mean Cross-Validation Score: -4.65...)
```

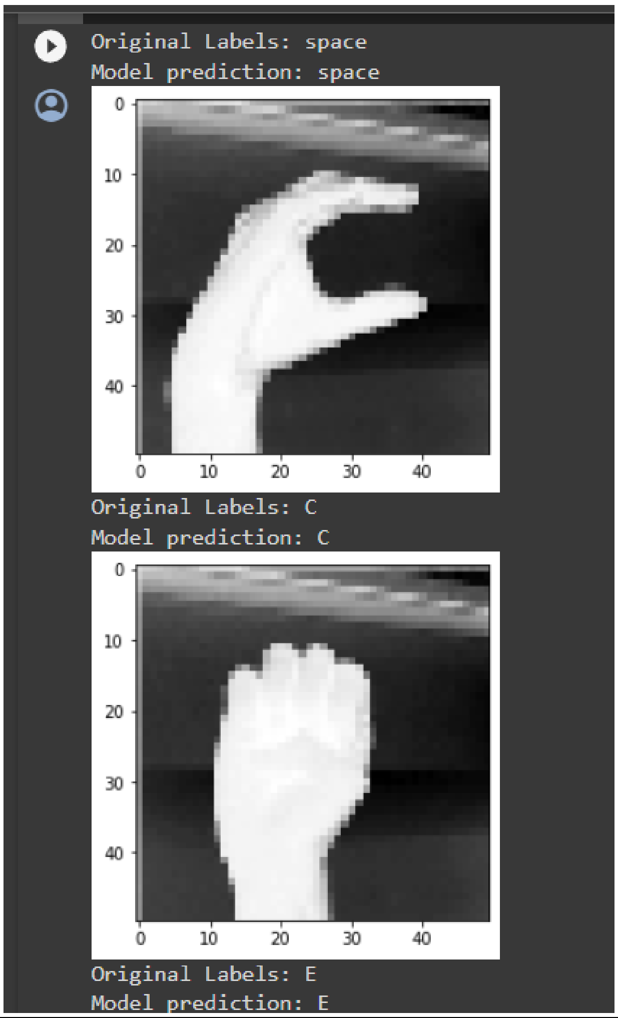
9. RESULTS



Results

```
n_images = 28
predictions = neural_nets(x_test)
for i in range(n_images):
    model_prediction = np.argmax(predictions.numpy()[i])
    plt.imshow(np.reshape(x_test[i], [50, 50]), cmap='gray_r')
    plt.show()
    print("Original Labels: %s" % get_key(y_test[i]))
    print("Model prediction: %s" % get_key(model_prediction))
```

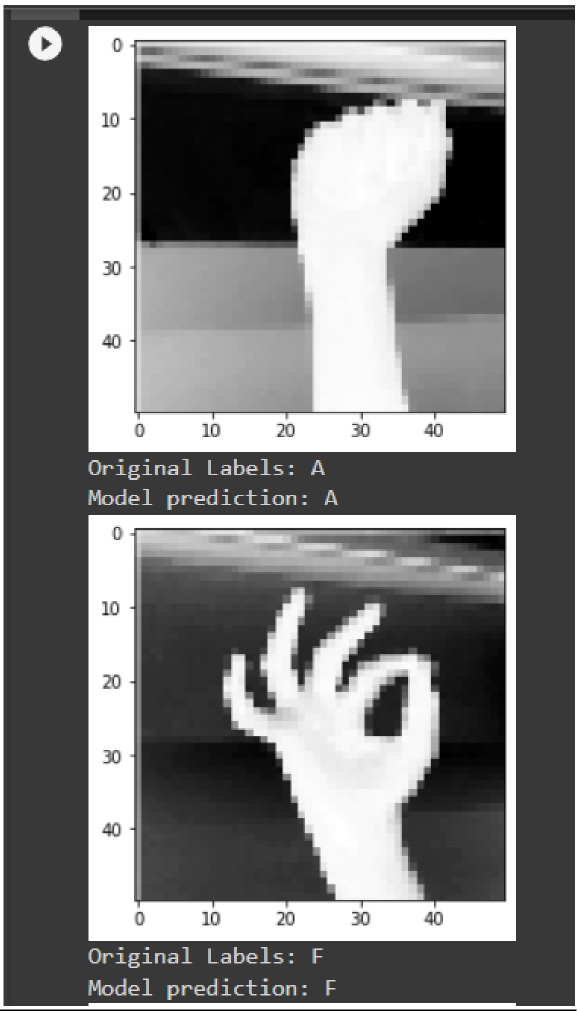
```
Original Labels: space
Model prediction: space
```



```
Original Labels: space
Model prediction: space
```

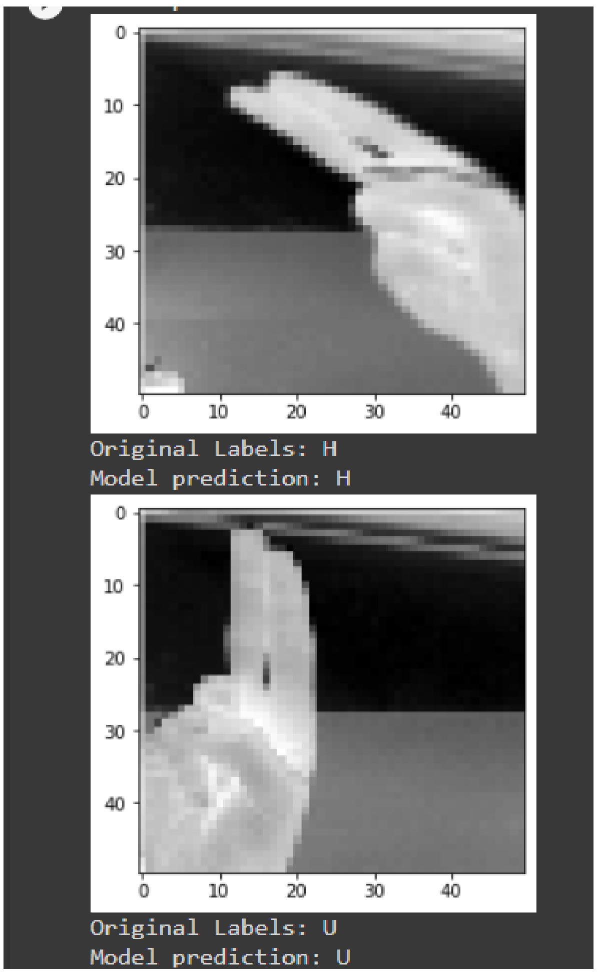
```
Original Labels: C
Model prediction: C
```

```
Original Labels: E
Model prediction: E
```



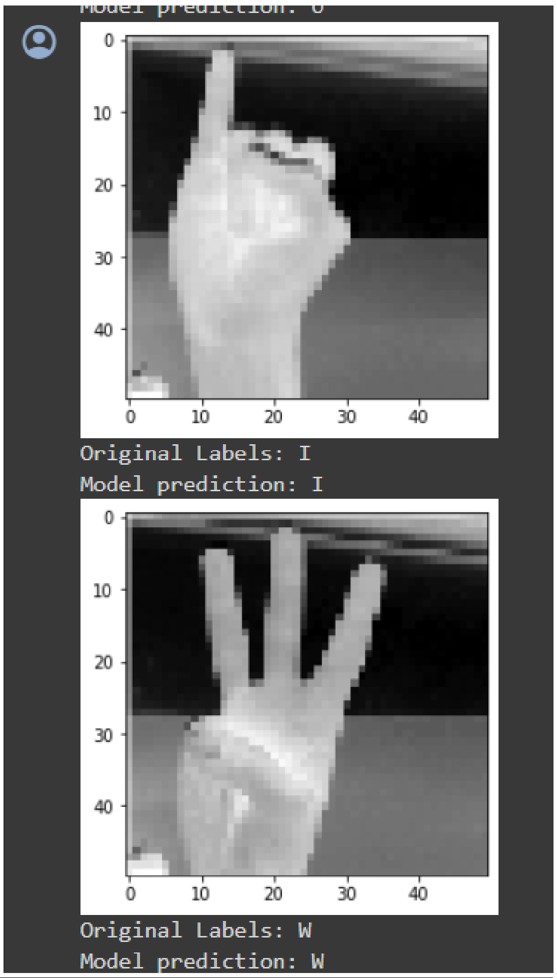
```
Original Labels: A
Model prediction: A
```

```
Original Labels: F
Model prediction: F
```



```
Original Labels: H
Model prediction: H
```

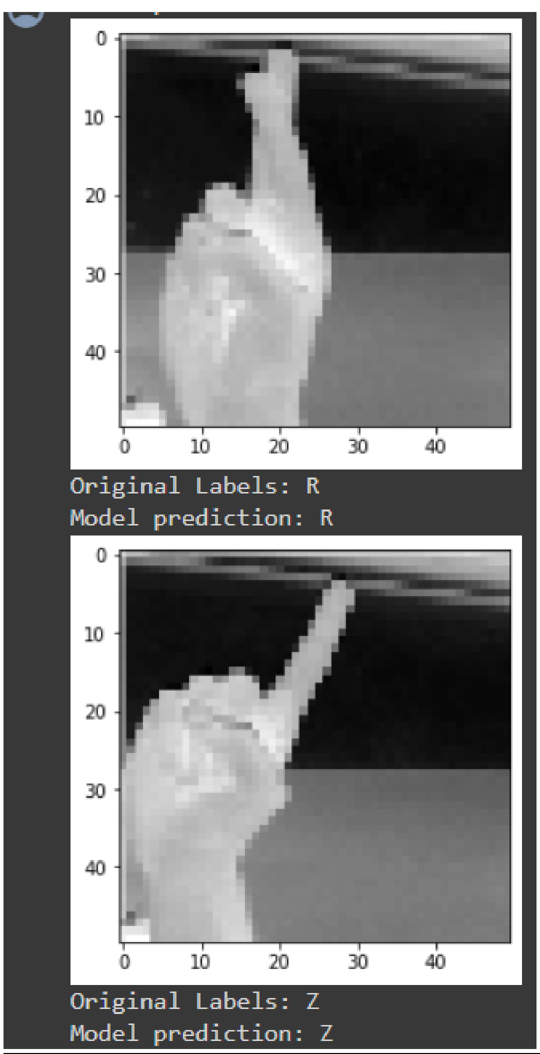
```
Original Labels: U
Model prediction: U
```



```
Model prediction: U
```

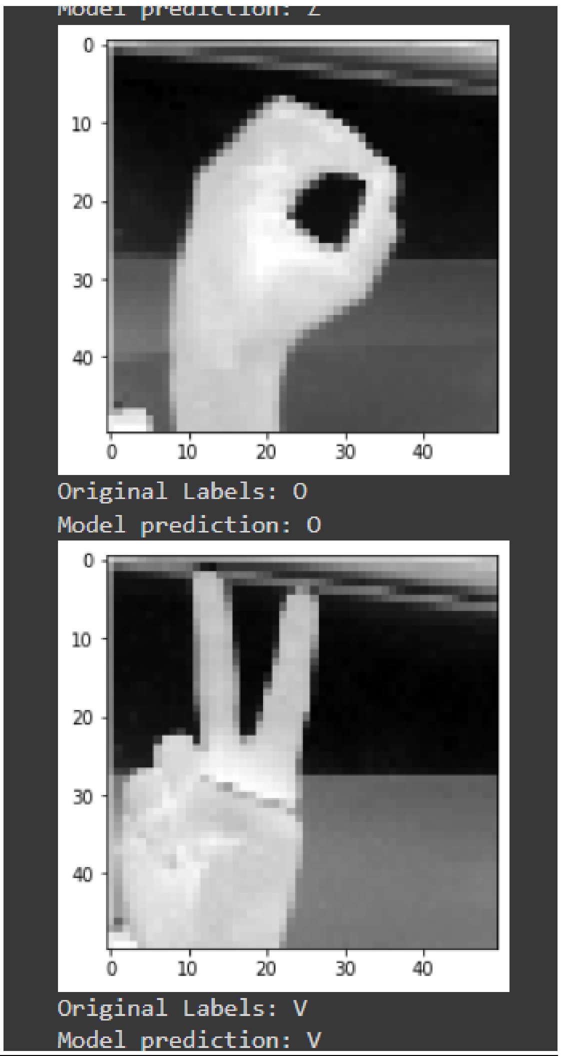
```
Original Labels: I
Model prediction: I
```

```
Original Labels: W
Model prediction: W
```



```
Original Labels: R
Model prediction: R
```

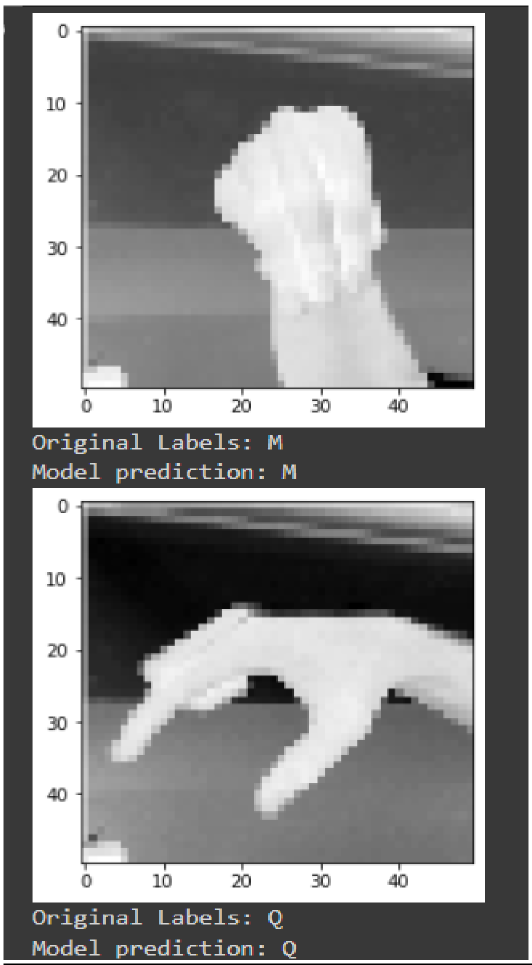
```
Original Labels: Z
Model prediction: Z
```



```
Model prediction: Z
```

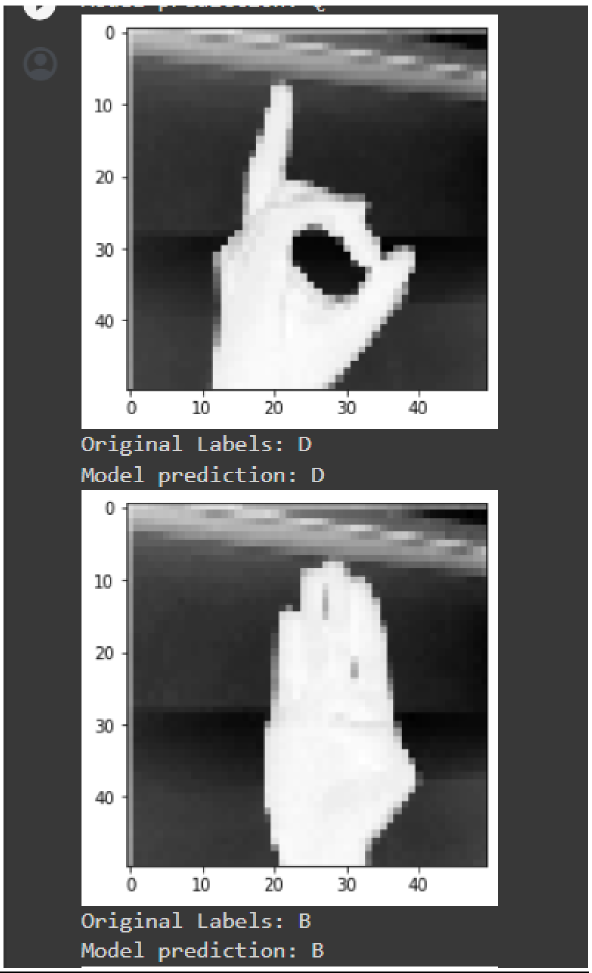
```
Original Labels: O
Model prediction: O
```

```
Original Labels: V
Model prediction: V
```



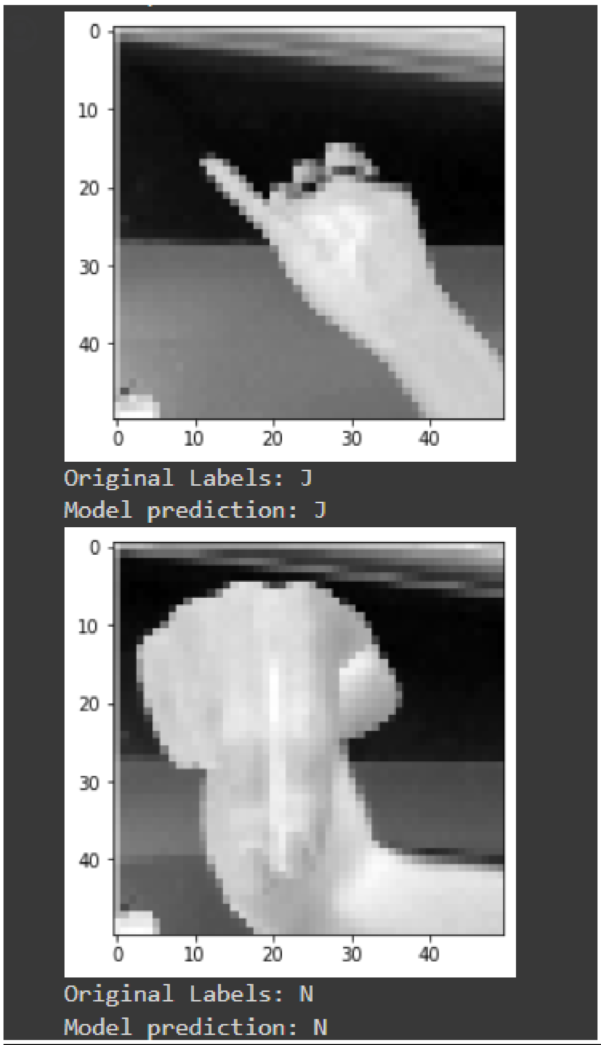
```
Original Labels: M
Model prediction: M
```

```
Original Labels: Q
Model prediction: Q
```



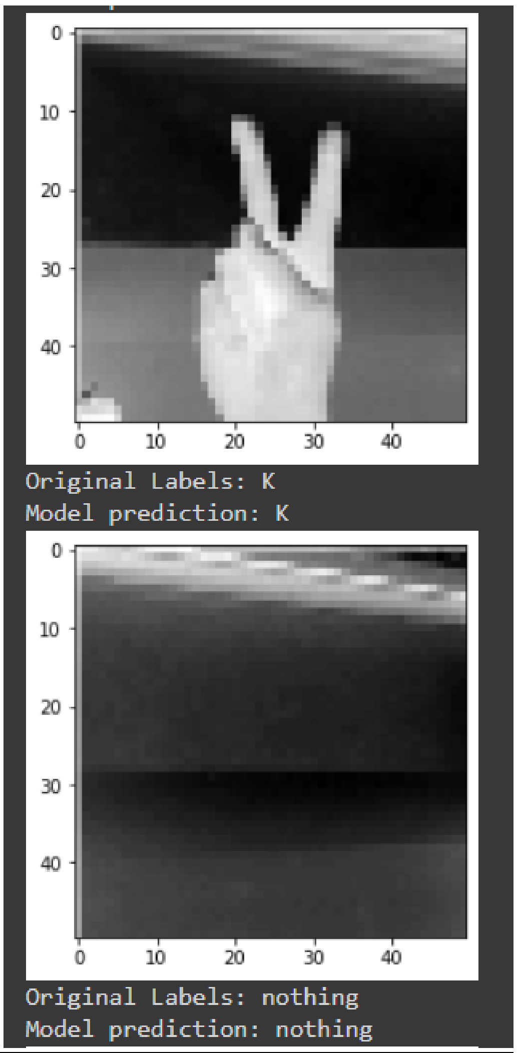
```
Original Labels: D
Model prediction: D
```

```
Original Labels: B
Model prediction: B
```



```
Original Labels: J
Model prediction: J
```

```
Original Labels: N
Model prediction: N
```



```
Original Labels: K
Model prediction: K
```

```
Original Labels: nothing
Model prediction: nothing
```

10. ADVANTAGES & DISADVANTAGES

Advantages of the ASL Alphabet Recognition Project:

1. **Enhanced Communication**:
    - Enables effective communication between Deaf and Hard of Hearing individuals who use ASL and non-signers, reducing language barriers and promoting inclusivity.
2. **Accessibility**:
    - Improves accessibility to education, employment, and various services for the Deaf community, enhancing their opportunities and quality of life.
3. **Educational Support**:
    - Serves as a valuable educational tool, aiding individuals in learning ASL and promoting understanding of Deaf culture and language.
4. **Inclusivity**:
    - Fosters inclusivity by breaking down communication barriers, allowing Deaf individuals to fully participate in social interactions and society.
5. **Independence**:
    - Grants Deaf individuals greater independence in various situations, reducing their reliance on sign language interpreters.
6. **Efficiency**:
    - Provides a more efficient and immediate means of communication compared to manual interpretation services, which may have limitations in availability and speed.
7. **Cost-Effective**:
    - Offers a cost-effective solution for ASL recognition compared to the expenses associated with hiring interpreters for every interaction.

Disadvantages of the ASL Alphabet Recognition Project:

1. **Technical Challenges**:
    - Developing accurate ASL recognition models can be technically challenging, particularly in handling variations in signing styles and environmental conditions.
2. **Cultural Sensitivity**:
    - Ensuring the system is culturally sensitive and recognizes regional variations in ASL can be complex, as sign language is not a monolithic language.
3. **Reliance on Technology**:
    - Overreliance on technology for communication may reduce face-to-face interaction and the human connection that comes with it.
4. **Privacy Concerns**:
    - The collection and processing of user data, including images and interpreted signs, raise privacy concerns that must be carefully addressed.
5. **Ethical Considerations**:
    - Ethical concerns related to data collection, privacy, and the societal impact of ASL recognition systems need to be carefully managed.

6. **Limited Availability**:
    - The project may have limited availability in terms of hardware or software platforms, and access to the technology may not be widespread.
7. **Maintenance and Support**:
    - Maintaining and supporting the system over time can be resource-intensive and require ongoing effort and investment.
8. **Potential for Technical Errors**:
    - The system may not always provide accurate recognition, leading to misunderstandings or miscommunication.
9. **Learning Curve**:
    - Users, both Deaf and non-signers, may need time to adapt to the technology and learn how to use the ASL recognition system effectively.

In summary, the ASL Alphabet Recognition Project offers significant advantages in terms of communication, accessibility, and inclusivity but also comes with technical, cultural, and ethical challenges that need to be carefully considered and addressed for its successful implementation and adoption.

11. CONCLUSION

In conclusion, the ASL (American Sign Language) Alphabet Recognition Project represents a significant step forward in addressing the communication and accessibility challenges faced by Deaf and Hard of

Hearing individuals. This innovative project has the potential to create a more inclusive and connected world by making ASL more accessible to a wider audience. The culmination of this project results in a valuable tool that offers the following key takeaways:

1. **Enhanced Communication**: By enabling accurate ASL sign recognition, the project bridges the communication gap between ASL users and non-signers, reducing language barriers and promoting more effective interaction.
2. **Accessibility**: Deaf individuals gain improved access to education, employment, and services, leading to expanded opportunities and a higher quality of life.
3. **Educational Support**: The system serves as an educational resource, fostering a deeper understanding of ASL, Deaf culture, and language among both Deaf and non-Deaf individuals.
4. **Inclusivity**: The project promotes inclusivity, reducing social isolation and facilitating the full participation of Deaf individuals in various aspects of society.
5. **Efficiency and Cost-Effectiveness**: By providing real-time ASL recognition, the project offers a more efficient and cost-effective solution compared to manual interpretation services.

However, it is important to acknowledge that the ASL Alphabet Recognition Project also faces technical, ethical, and cultural challenges. It requires ongoing effort to ensure accuracy, privacy, and cultural sensitivity. Moreover, the success of the project depends on the continuous support, maintenance, and adaptation to evolving needs.

As we move forward, it is crucial to keep in mind the importance of user feedback, ethical considerations, and the cultural richness of ASL. By addressing these factors and fostering innovation in ASL recognition, we can work towards a more inclusive and accessible future for Deaf and Hard of Hearing individuals, where communication knows no boundaries.

12. FUTURE SCOPE

The future scope of the ASL (American Sign Language) Alphabet Recognition Project is promising and offers numerous opportunities for development and expansion. As technology continues to advance, the project can evolve to address new challenges and meet the needs of the Deaf and Hard of Hearing community. Here are some potential future directions and areas of growth for the project:

1. **Extended ASL Vocabulary**: Expand the recognition system to include a broader range of ASL signs and phrases, not limited to just the alphabet. This will make the system more versatile for everyday communication.
2. **Multi-Language and Regional Support**: Develop ASL recognition models for different regional variations of ASL and other sign languages, making the system accessible to a global audience.
3. **Gesture and Facial Expression Recognition**: Enhance the system to recognize facial expressions and other non-manual signals that are crucial for conveying meaning in sign language.
4. **Mobile Applications**: Develop mobile apps that can provide on-the-go ASL recognition, aiding users in their daily interactions and fostering independence.
5. **Wearable Technology**: Integration with smart glasses or wearable devices, offering hands-free ASL recognition and feedback, which can be particularly useful in real-world scenarios.
6. **Machine Learning Advances**: Leverage advancements in machine learning, including reinforcement learning and more efficient neural network architectures, to enhance the accuracy and speed of recognition.
7. **Data Expansion**: Continuously expand and diversify the dataset with contributions from ASL users to improve the model's performance and coverage.
8. **User Feedback and Iteration**: Continuously gather feedback from users, both Deaf and non-signers, to improve the user experience, refine the recognition model, and address specific user needs.
9. **Community Involvement**: Foster collaboration with the ASL community to ensure that the project remains culturally sensitive and respectful of the diversity of ASL.
10. **Educational Applications**: Develop educational tools and resources to teach ASL and sign language interpretation, benefiting students and educators.
11. **Medical and Healthcare**: Explore applications in healthcare, such as using ASL recognition for effective communication with Deaf patients and improving the delivery of medical services.
12. **Entertainment and Gaming**: Implement ASL recognition in video games, virtual reality experiences, and entertainment applications to create engaging and inclusive content.
13. **Global Accessibility Standards**: Contribute to the development of global accessibility standards and regulations for ASL recognition systems to ensure ethical and responsible use.
14. **Research Collaborations**: Collaborate with academic institutions and researchers to advance the field of ASL recognition and sign language technology.

The future of the ASL Alphabet Recognition Project is promising, and it has the potential to make a lasting and positive impact on the lives of Deaf and Hard of Hearing individuals, as well as on society at large, by fostering inclusivity, accessibility, and improved communication.

13. APPENDIX

GitHub Link: https://github.com/smartinternz02/SI-GuidedProject-605385-1700337973/tree/main

Project Demo Link:
https://drive.google.com/file/d/1SX8zNbS1IqsQODYxMr5royHdxbsCTs1k/view?usp=drive_link