CHAPTER 1

INTRODUCTION AND PROBLEM STATEMENT

1.1 PROJECT SUMMARY

- The world is hardly live without communication, no matter whether it is in form of textual, voice or visual expression.
- The communication among the deaf and dumb people is carried by text or visual expression.
- Gesture communication is always in the scope of confidential and secure communication.
- Hands and facial parts are immensely influential to express Face detection and the thoughts of human in confidential communication.
- Sign language is learned by deaf and dumb, and usually it is not known to normal people, so it becomes a challenge for communication between a normal and hearing impaired person.
- Its strike to our mind to bridge the between hearing impaired and normal people to make the communication easier.
- Sign language recognition (SLR) system takes an input expression from the hearing impaired person gives output to the normal person in the form text or voice.

1.2 OBJECTIVE

- Communication is always having a great impact in every domain and how it is considered the meaning of the thoughts and expressions that attract the researchers to bridge this gap for every living being.
- The objective of this project is to identify the symbolic expression through images so that the communication gap between a normal and hearing impaired person can be easily bridged.

1.3 PROBLEM STATEMENT

- To Identify the Hand Signs Gestures or Symbol from an Image to Recognition the Alphabets.

CHAPTER 2

Tools, Technologies and Principles used

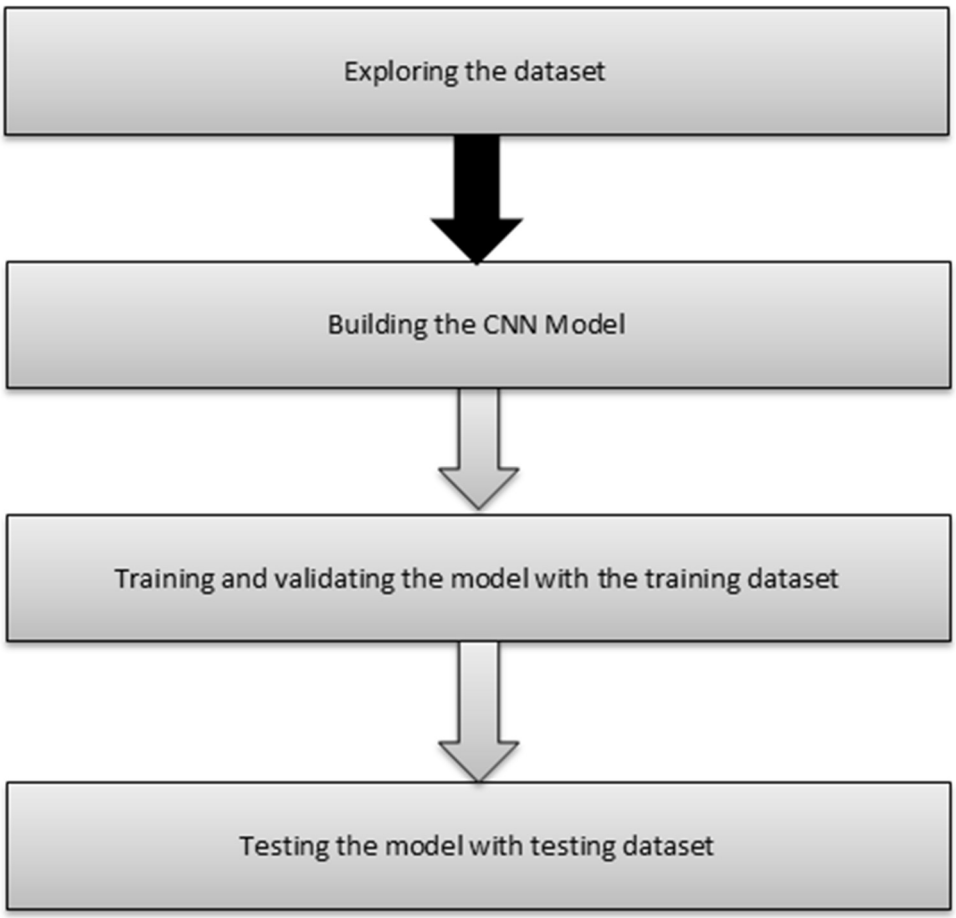
2.1 Tools, Technologies and Principles used

1. **Sign Language Recognition using Python Programming Language**.
2. **Application :** Console Application
3. **Required Software**
    a. VS Code
    b. Jupyter Notebook
4. **Required Libraries:**
    a. Pandas
    b. Numpy
    c. Matplotlib
    d. OpenCV
    e. Imutils
    f. PIL
    g. Tenserflow
    h. Keras
5. **Hardware Specification**
    a. Intel Core I3 Processor 1.6 GHz
    b. RAM : 4 Gigabyte
    c. ROM : 480 Gigabyte
6. **Operation System**
    a. Windows 10 Pro

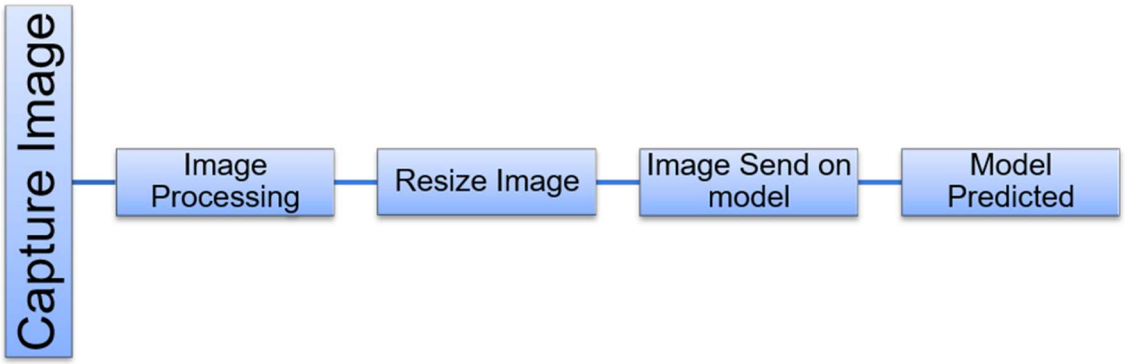
CHAPTER 3

METHODOLOGY

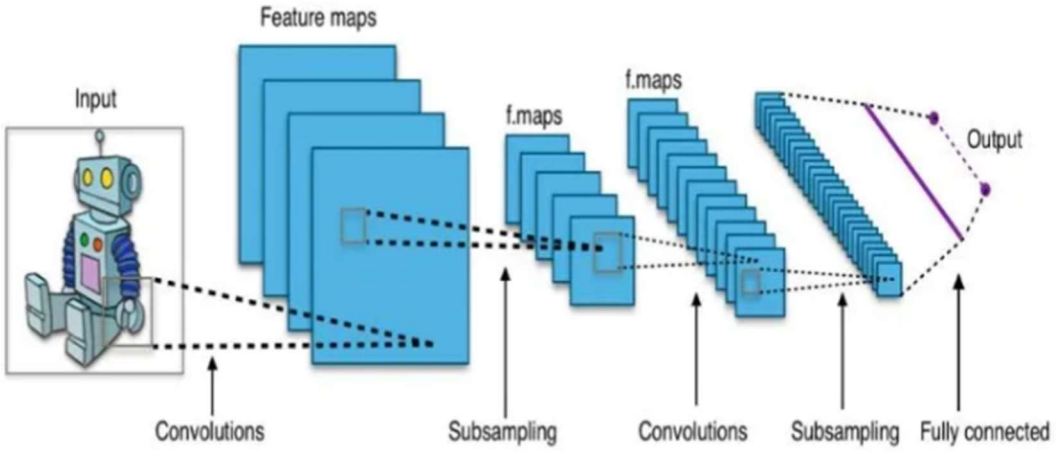
3.1 TRAINING PROCESS

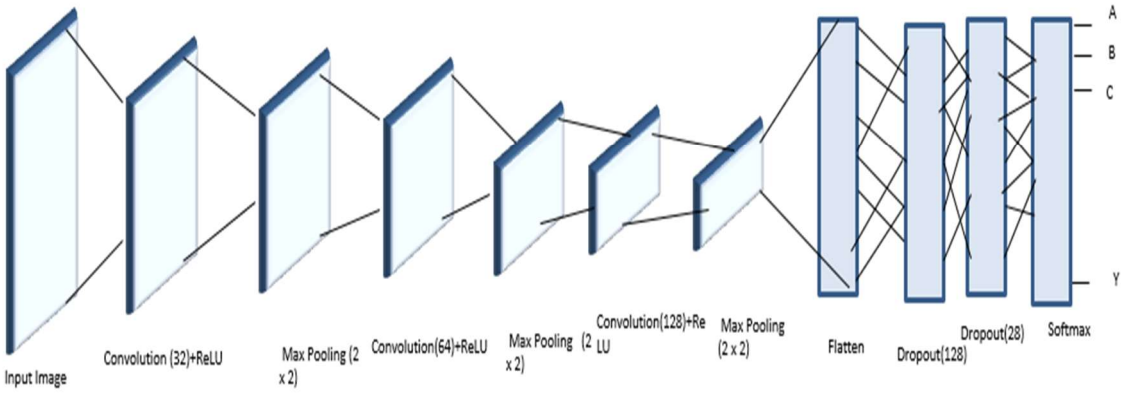


3.2 DETECTION PROCESS

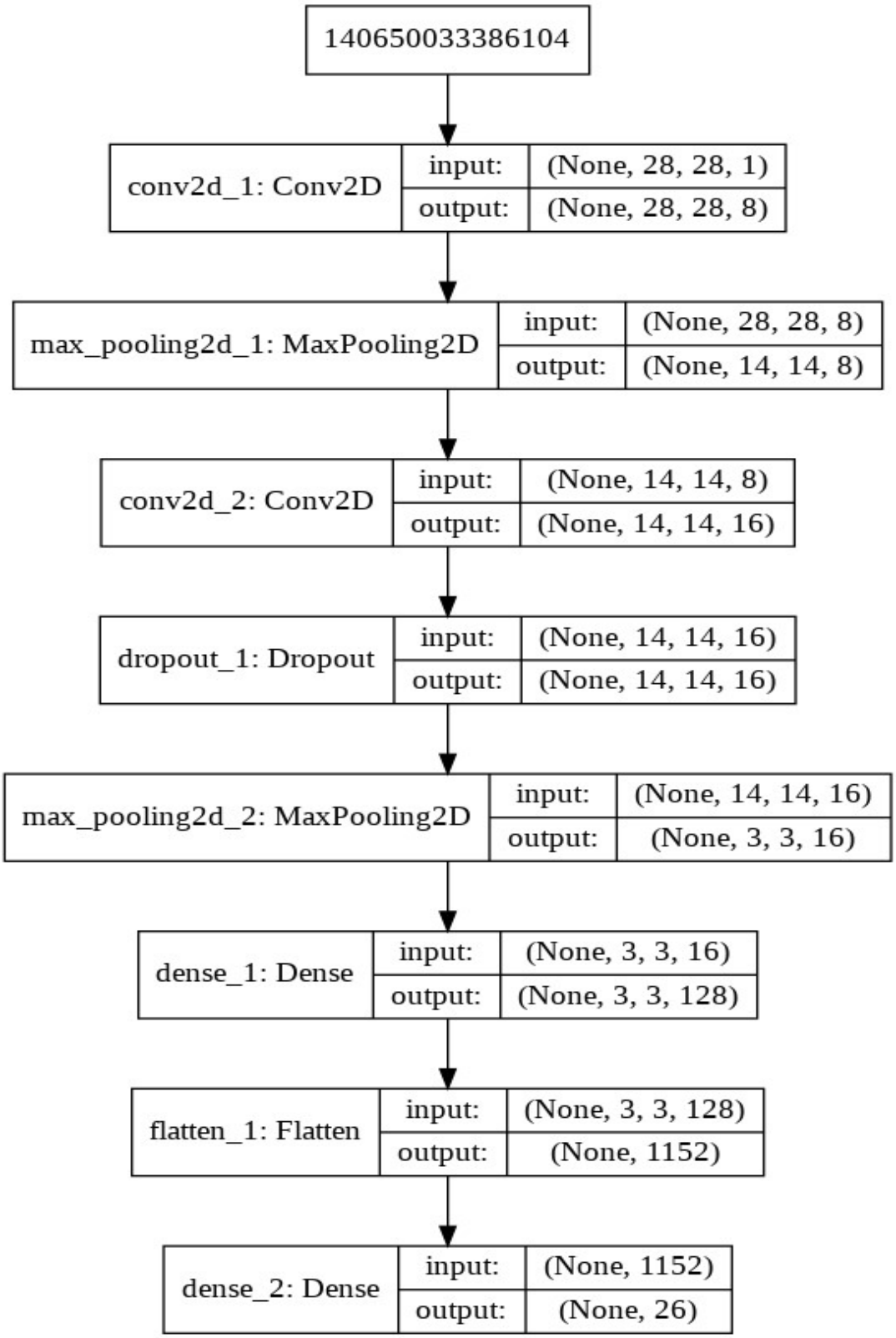


3.3 CNN Model Architecture





3.4 CNN Model Architecture with Input and Output

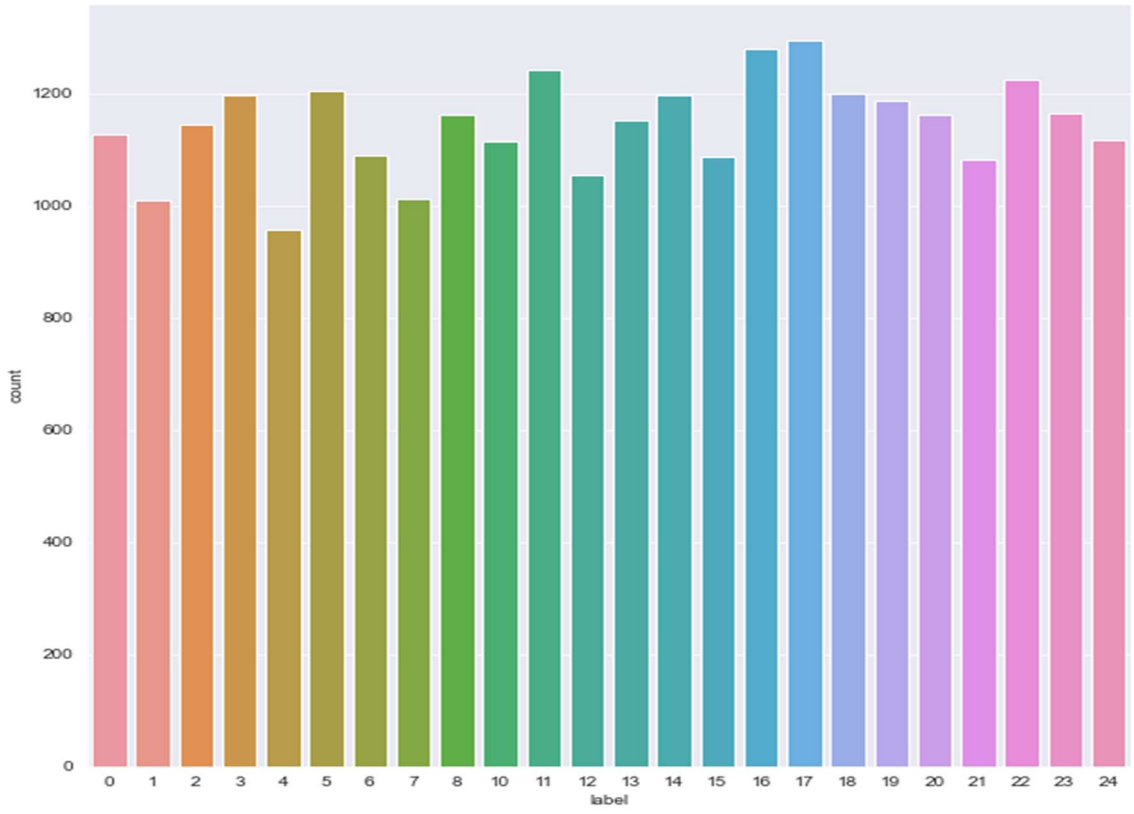


CHAPTER 4

DATASET

4.1 Data Set that is used for Training the Model

4.2 Dataset Exploration



- The dataset format is patterned to match closely with the classic MNIST.
- Each training and test case marks a label from 0 to 25 as a matched map for every alphabetic letter A-Z (and no cases for 9=J or 25=Z due to gesture motions).
- Each class label is a set of sign images of the English alphabet.
- The training data (27,455 samples) and test data (7172 samples) are approximately half the size of the standard MNIST but otherwise similar with a header row of the label, pixel1, pixel2…. pixel784 which represents a single 28x28 pixel image with grayscale values ranging 0-255.
- The dataset seems balanced as for each training label, and enough training examples exist shown in Figure on the left side that we obtained during the exploration of the dataset.

CHAPTER 5
IMPLEMENTATION DETAILS

5.1 IMPLEMENTATION DETAILS

1. **Required Libraries:**

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import tensorflow
from tensorflow import keras
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

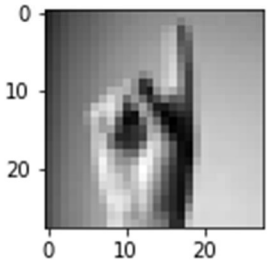
2. **Load Train and Test Data:**

```
train = pd.read_csv('dataset\sign_mnist_train\sign_mnist_train.csv')
test = pd.read_csv('dataset\sign_mnist_train\sign_mnist_train.csv')
```

3. **Plot 1 random image from Training Data:**

```
i = random.randint(1,train.shape[0])
fig1, ax1 = plt.subplots(figsize=(2,2))
plt.imshow(train_data[i,1:].reshape((28,28)), cmap='gray')
print("Label for the image is: ", class_names[int(train_data[i,0])])
```

```
Label for the image is:  K
```



4. **Normalize / Scale Data Set Values**

```
X_train = train_data[:, 1:] /255.
X_test = test_data[:, 1:] /255.
```

5. Defining the Convolutional Neural Network (CNN) Model 1

```
#Model 1

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape = (28,28,1), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation = 'relu'))
model.add(Dense(25, activation = 'softmax'))
```

6. Defining the Convolutional Neural Network (CNN) Model 2

```
#Model2

model2 = Sequential()

model2.add(Conv2D(32, (3, 3), input_shape = (28,28,1), activation='relu'))
model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size = (2, 2)))

model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size = (2, 2)))

model2.add(Conv2D(128, (3, 3), activation='relu'))
model2.add(Conv2D(25, (1,1)))

model2.add(Flatten())

model2.add(Dense(25, activation = 'softmax'))
```

7. Defining the Convolutional Neural Network (CNN) Model 3

```
# Model 3
classifier = Sequential()

classifier.add(Conv2D(filters=8, kernel_size=(3,3),strides=(1,1),padding='same',input_shape=(28,28,1),activation='relu', dat
classifier.add(MaxPooling2D(pool_size=(2,2)))

classifier.add(Conv2D(filters=16, kernel_size=(3,3),strides=(1,1),padding='same',activation='relu'))
classifier.add(Dropout(0.5))

classifier.add(MaxPooling2D(pool_size=(4,4)))
classifier.add(Dense(128, activation='relu'))

classifier.add(Flatten())

classifier.add(Dense(26, activation='softmax'))
classifier.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
```

8. Model Summary

```
1 classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 8)         80
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 16)        1168
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 16)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 16)          0
_________________________________________________________________
dense_1 (Dense)              (None, 3, 3, 128)         2176
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_2 (Dense)              (None, 26)                29978
=================================================================
Total params: 33,402
Trainable params: 33,402
Non-trainable params: 0
_________________________________________________________________
```

9. Capture Video and Image Processing

```
cam_capture = cv2.VideoCapture(0)
while (cam_capture.isOpened()):

    _, image_frame = cam_capture.read()

    im2 = crop_image(image_frame, x,y,width,height)

    image_grayscale = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY)

    image_grayscale_blurred = cv2.GaussianBlur(image_grayscale, (15,15), 0)
    im3 = cv2.resize(image_grayscale_blurred, (28,28), interpolation = cv2.INTER_AREA)

    im4 = np.resize(im3, (28, 28, 1))
    im5 = np.expand_dims(im4, axis=0)
```

10. Predicted the Class of a Sing Symbole

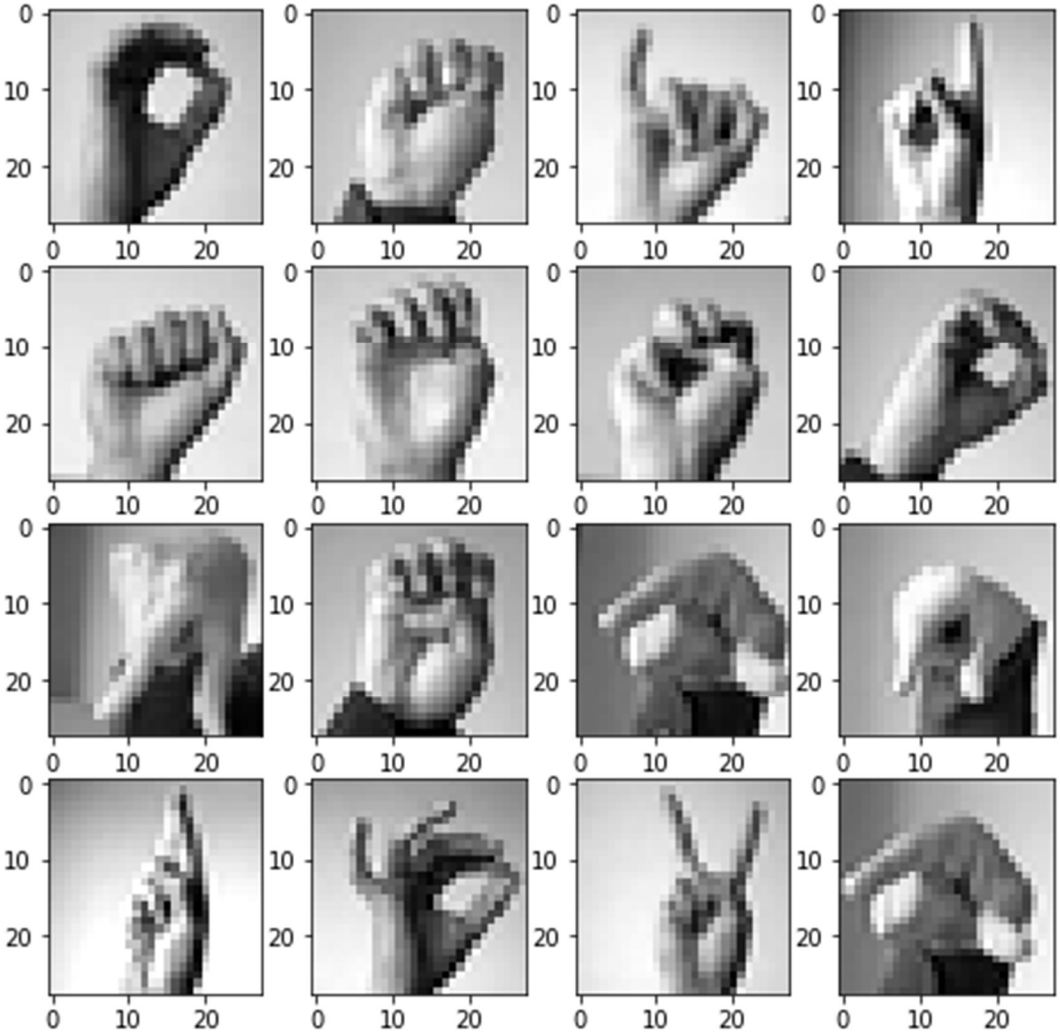
```
pred_probab, pred_class = keras_predict(model, im5)

curr = prediction(pred_class)
print(curr,pred_probab)
if pred_probab > 0.95:
    cv2.putText(image_frame, curr, (250, 300), cv2.FONT_HERSHEY_COMPLEX, 2.0, (255, 255, 0), 2,lineTyp
```

11. Predicted Function to predict the Class

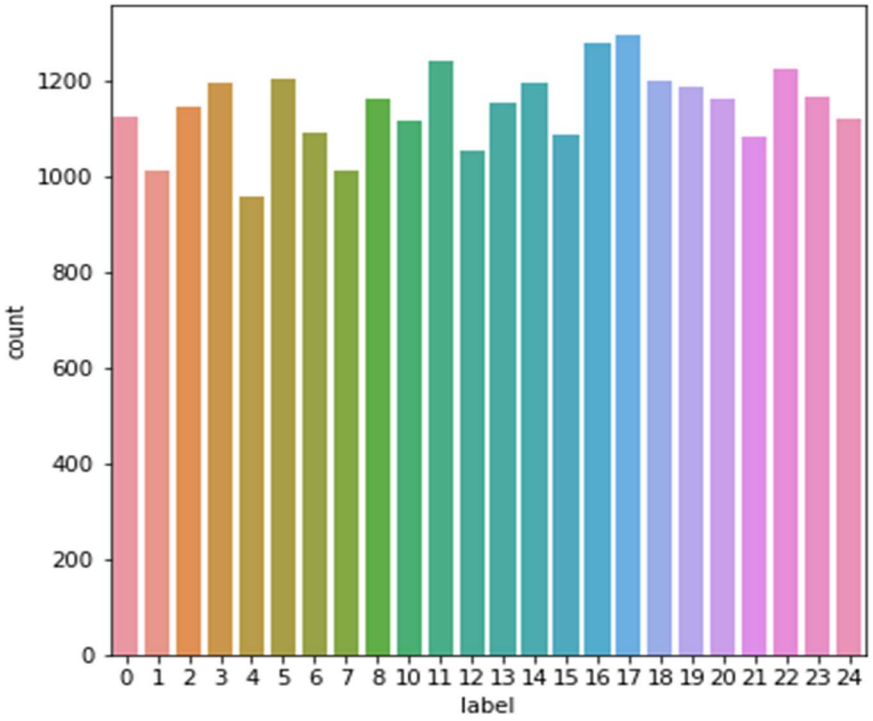
```
def keras_predict(model, image):
    data = np.asarray( image, dtype="int32" )
    pred_probab = model.predict(data)[0]
    pred_class = list(pred_probab).index(max(pred_probab))
    return max(pred_probab), pred_class
```

CHAPTER 6
RESULT

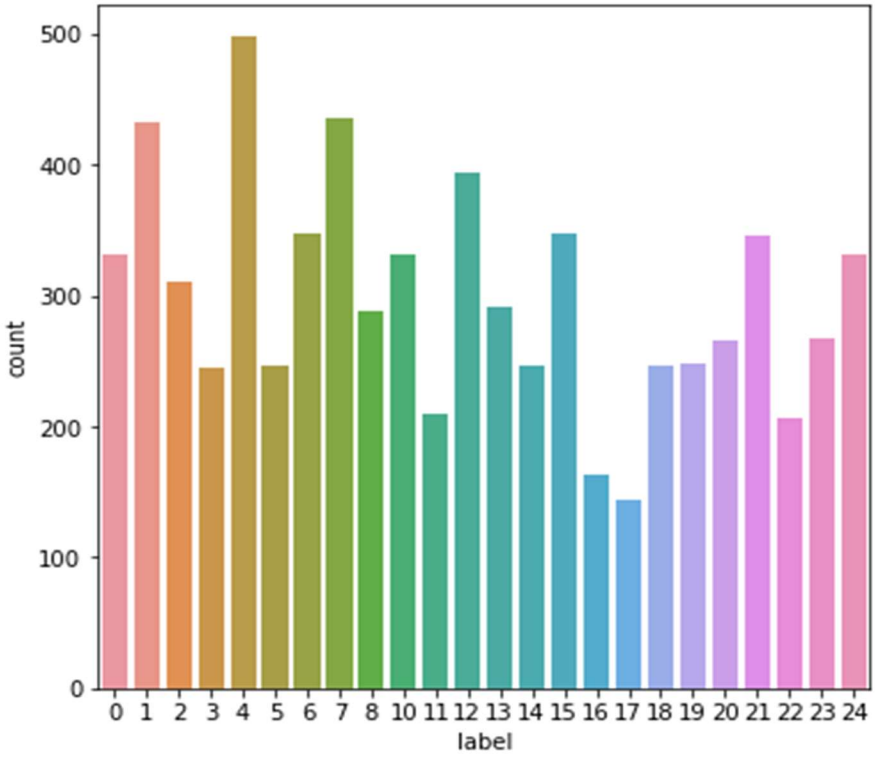
1. Data Set Images



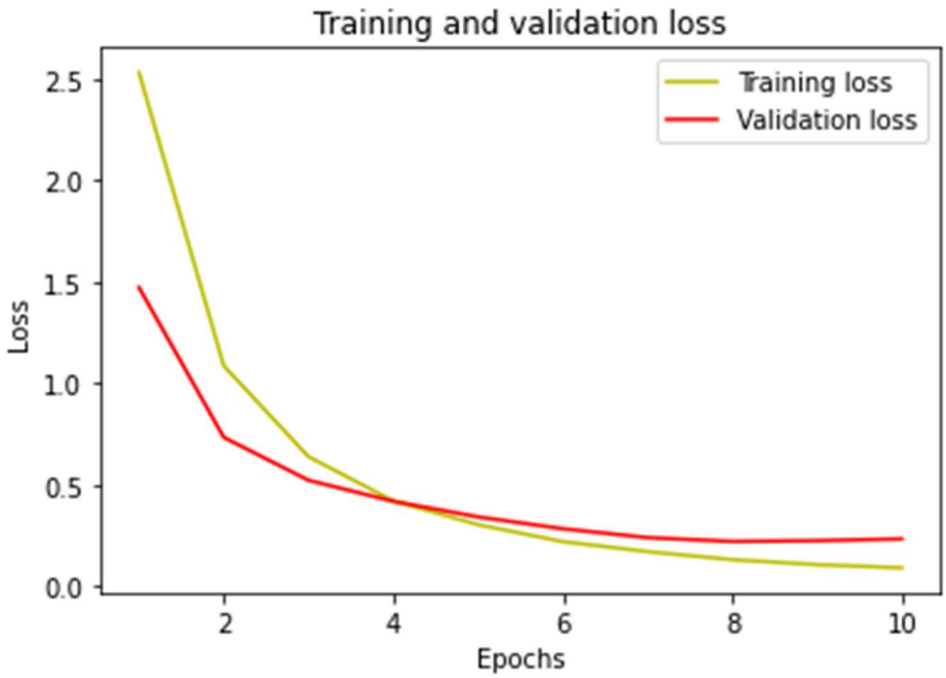
2. Train Data Distribution Visualization



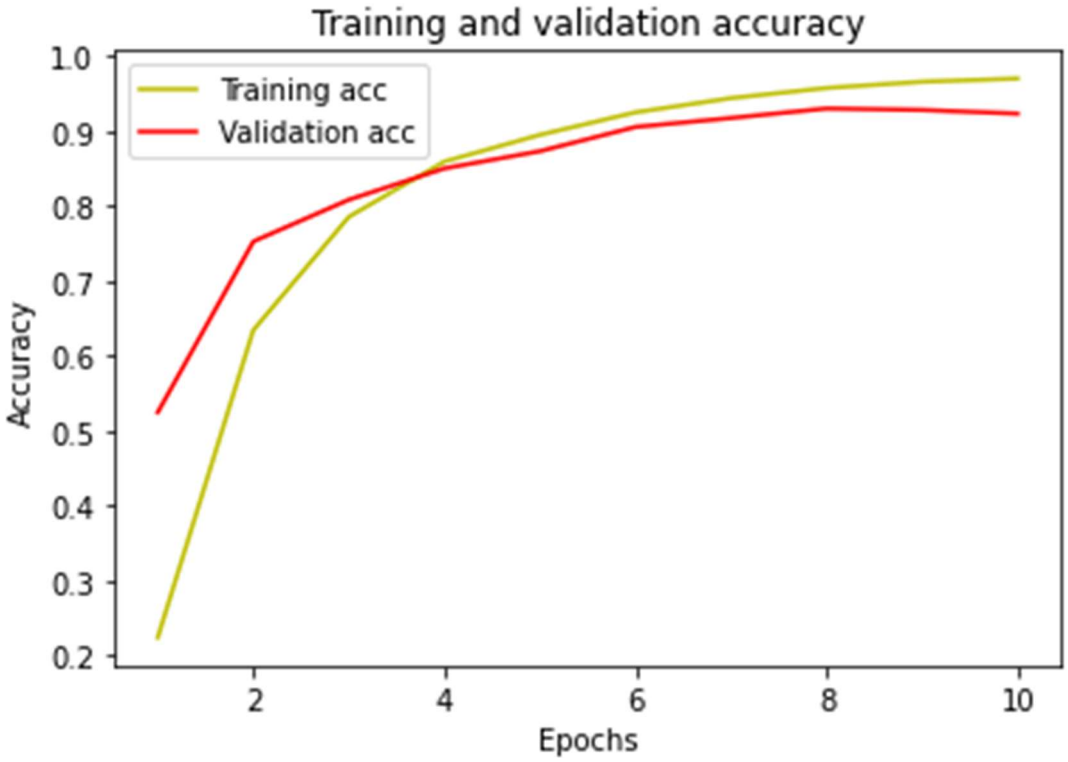
3. Test Data Distribution Visualization



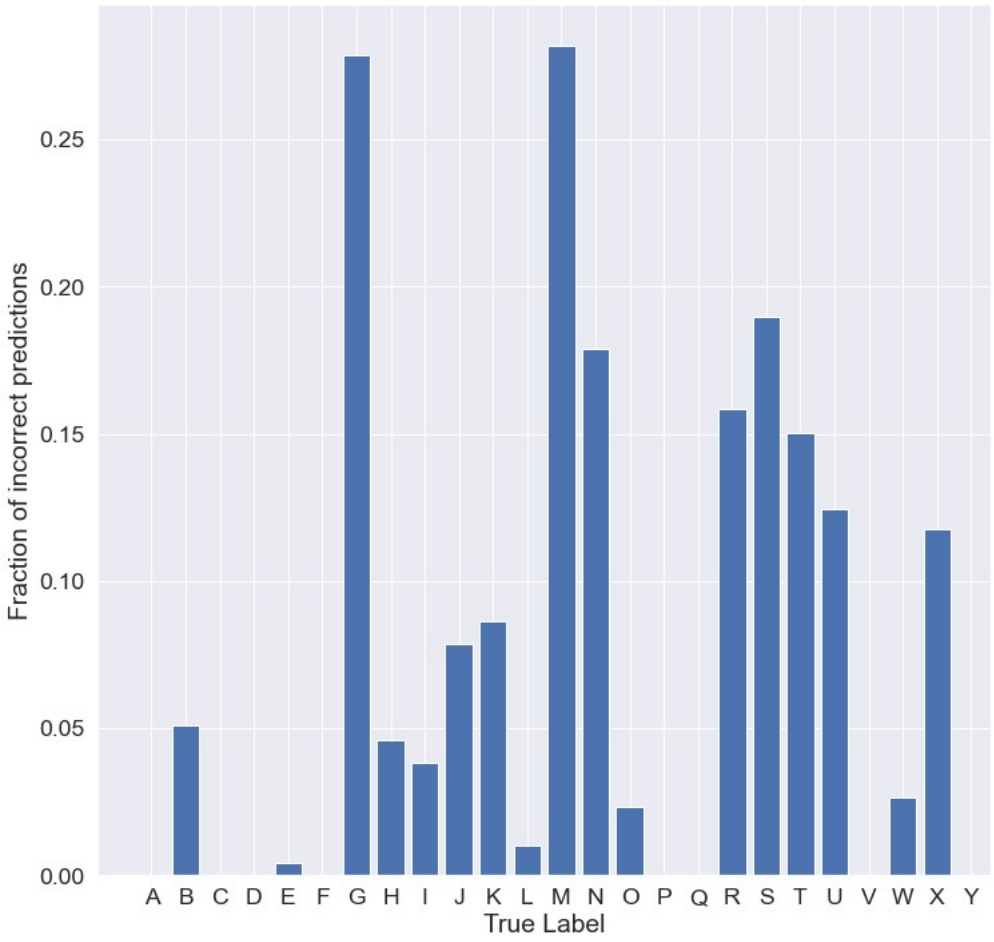
4. **Training and validation loss**



5. **Training and validation accuracy**

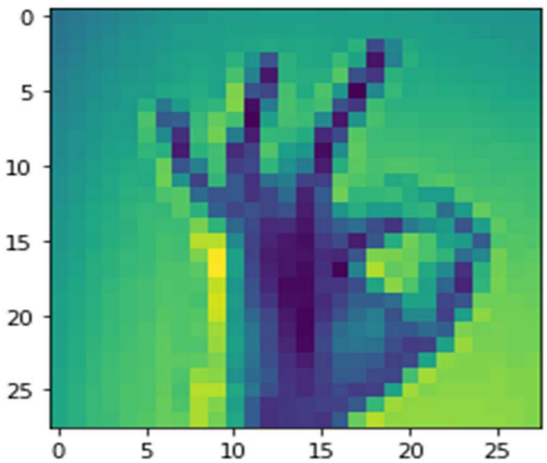


6. Plot fractional incorrect misclassifications



7. Predict On Testing Data

```
Predicted Label:  F
True Label:  F
```

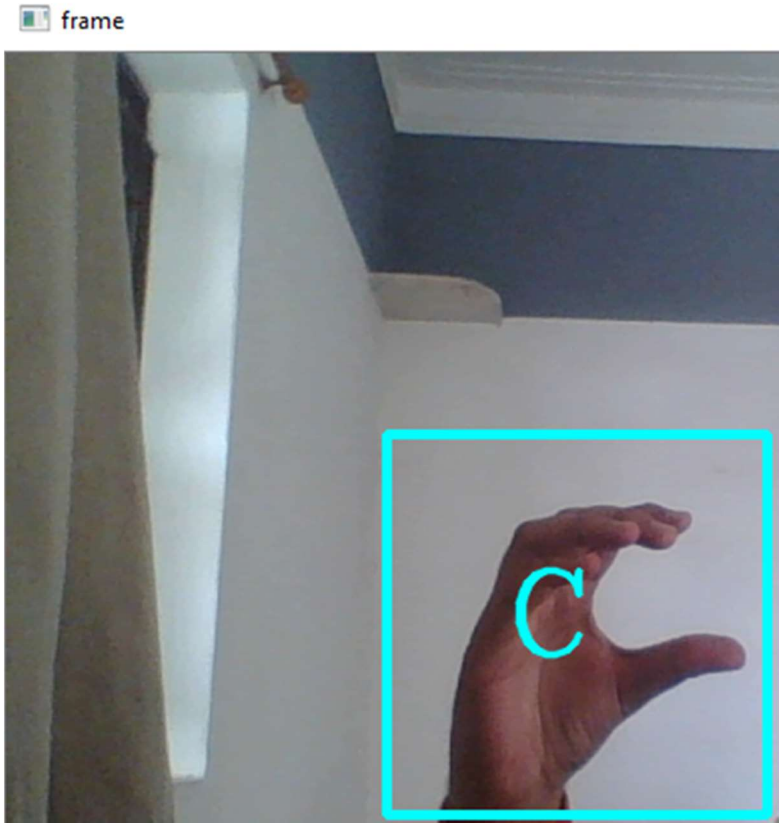


8. Accuracy Score

```
1 from sklearn.metrics import accuracy_score
2 prediction = model.predict_classes(X_test)
3 accuracy = accuracy_score(y_test, prediction)
4 print('Accuracy Score = ', accuracy)
```

```
Accuracy Score =  0.923452314556609
```

9. Predict Using Camera



CHAPTER 7
CONCLUSION

- Here, I am using Deep learning and computer vision techniques to classify Sign Language.
- I am using python programming and its different libraries for Deep learning and computer vision.
- My system has extract different feature of the Sign Languages so based on that features we can easily classify Alphabet.

CHAPTER 8
FUTURE ENHANCEMENT

- Future study may extend our work to accept video frames to include letters J and Z in the classification so that more varied inputs can be processed and understood by the network.
- Furthermore, there is a need for a large public dataset for automatic sign language recognition to utilize new deep learning techniques and a better way to benchmark performance

CHAPTER 9
REFERENCES

References

[1] "Indian Sign Language Character Recognition," [Online]. Available: https://www.cse.iitk.ac.in/users/cs365/2015/_submissions/vinsam/report.pdf.

[2] "MIE324 Final Report: Sign Language Recognition," [Online]. Available: https://www.eecg.utoronto.ca/~jayar/mie324/asl.pdf.

[3] tecperson, "Kaggle MINIST Data Set," [Online]. Available: https://www.kaggle.com/datasets/datamunge/sign-language-mnist.

[4] M. Wurangian. [Online].