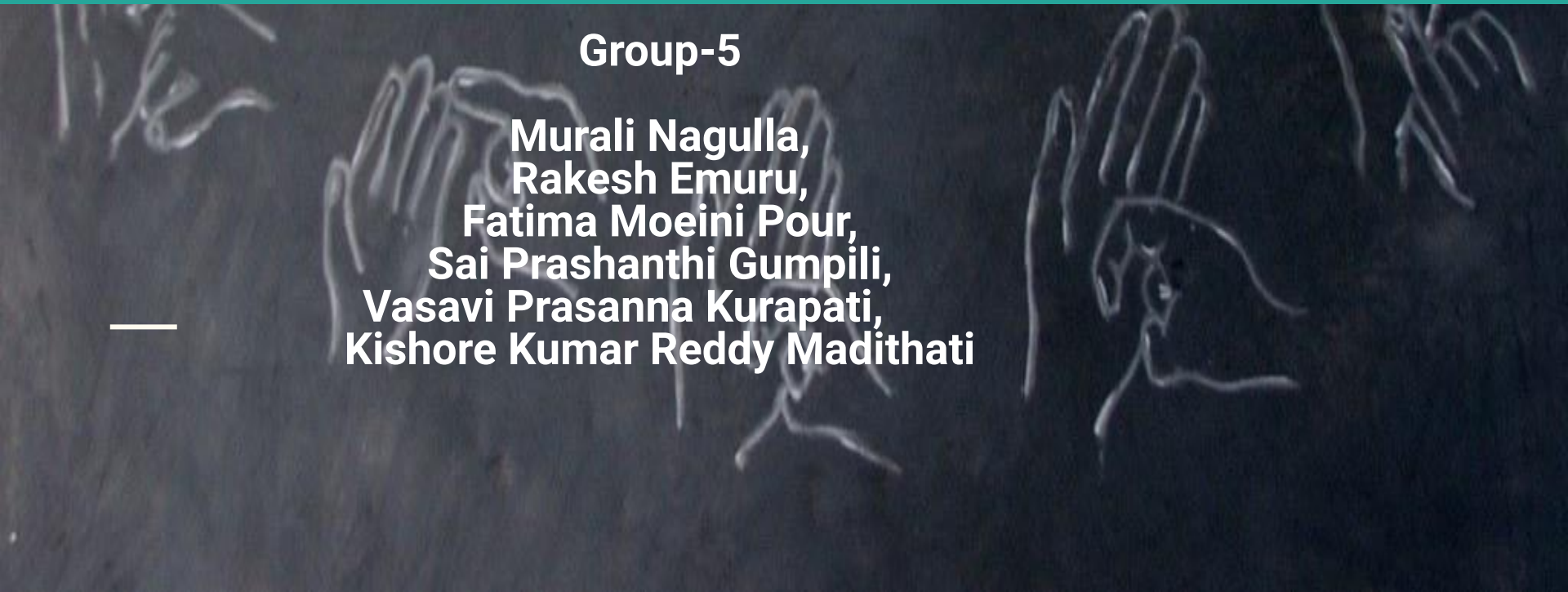
A Deep Learning-Based Algorithm for American Sign Language Recognition



Group-5

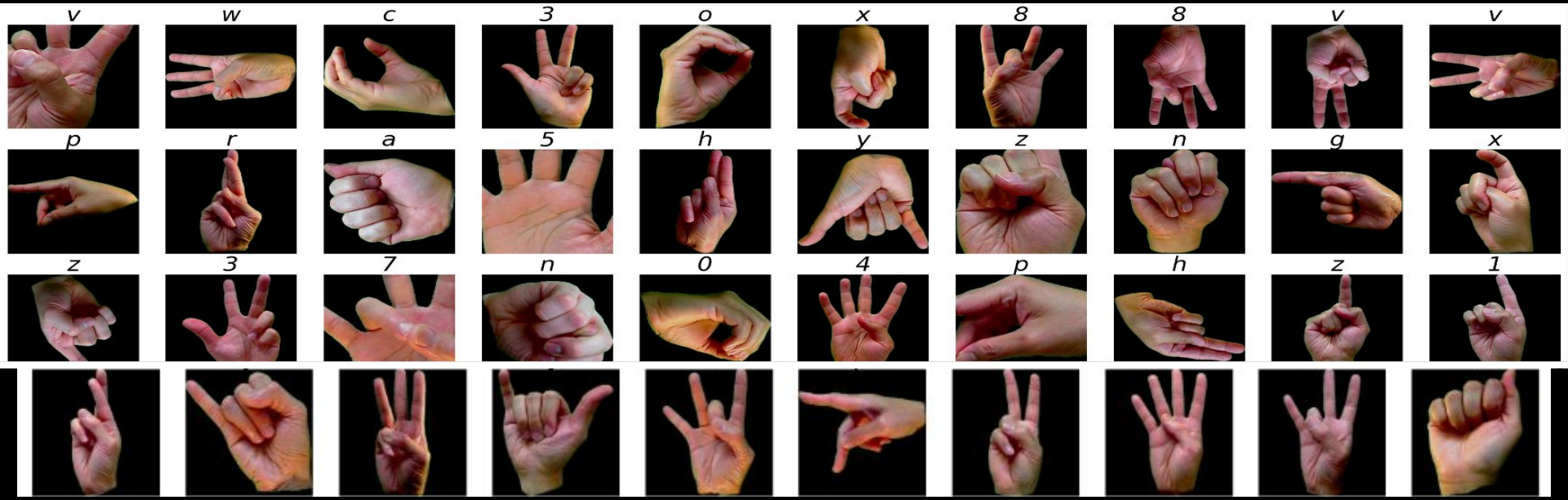
Murali Nagulla,
Rakesh Emuru,
Fatima Moeini Pour,
Sai Prashanthi Gumpili,
Vasavi Prasanna Kurapati,
Kishore Kumar Reddy Madithati

INTRODUCTION

- Speech-impaired individuals often rely on hand signs and gestures, or sign language, to communicate. People who are not familiar with sign language may find it difficult to understand these signs and gestures. This creates a communication barrier.
- In the US, approximately around 500,000 people use American sign language (ASL) as primary means of communication.
- With advancements in technology, particularly in the field of machine learning and artificial intelligence, it's now possible to develop a system that can recognize sign language. Such a system could translate sign language breaking down the communication barrier, promoting inclusivity and accessibility and making interactions smoother and more natural.
- Our project aims to develop a system that can classify sign language symbols using deep learning techniques. This system will be able to recognize different signs and gestures and translate them into understandable language.

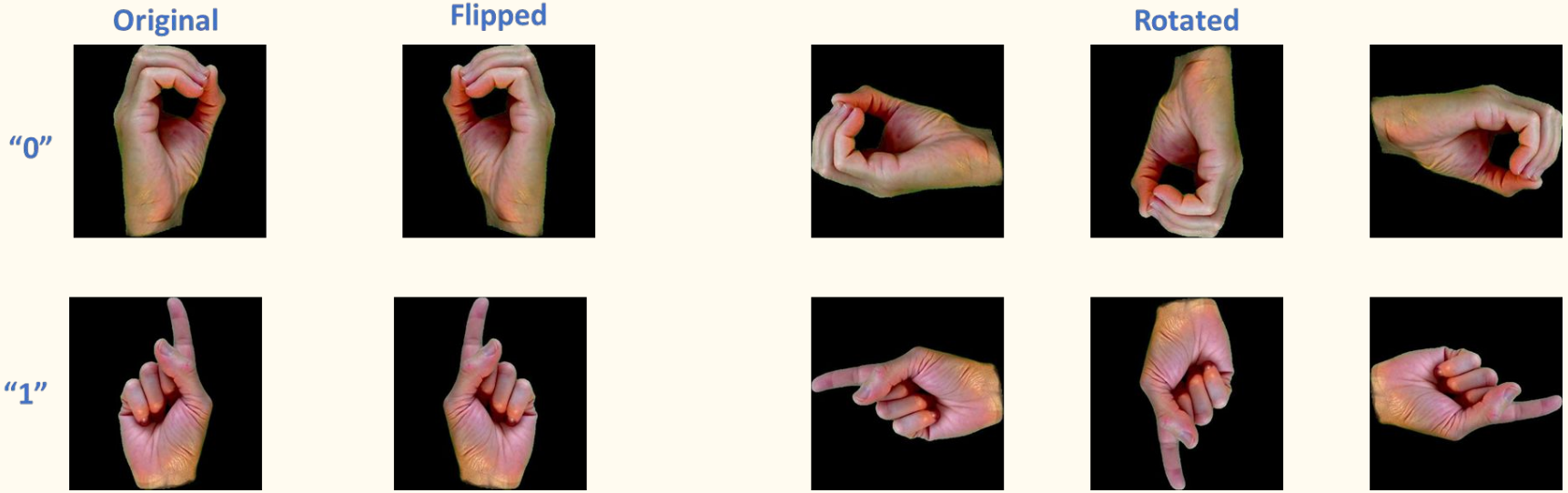
DATA

- The data set is a collection of images from the American Sign Language.
- There are 36 classes which include 0-9 and A-Z alphabets.



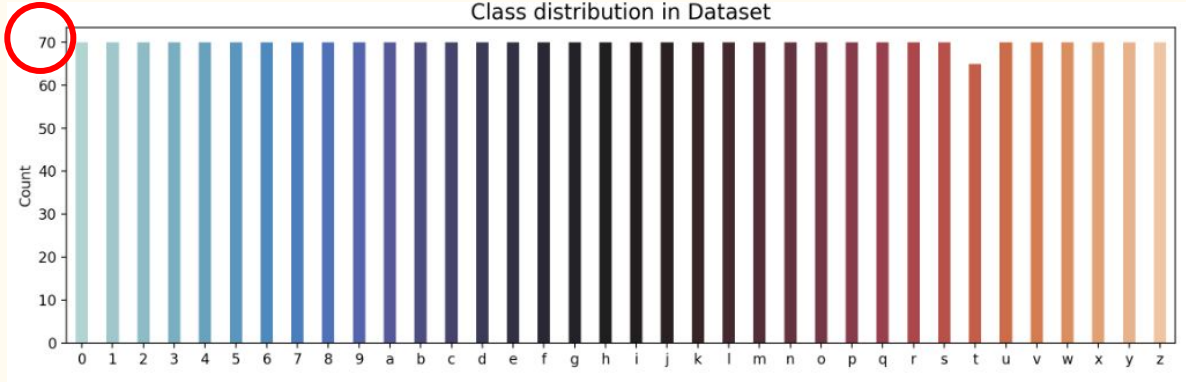
DATASET AUGMENTATION

- To make the dataset, the original image files are **resized**, **Mirrored,** and **rotated** using Image Data Generator.
- The dataset size is increased from 2,515 to **15,090**.

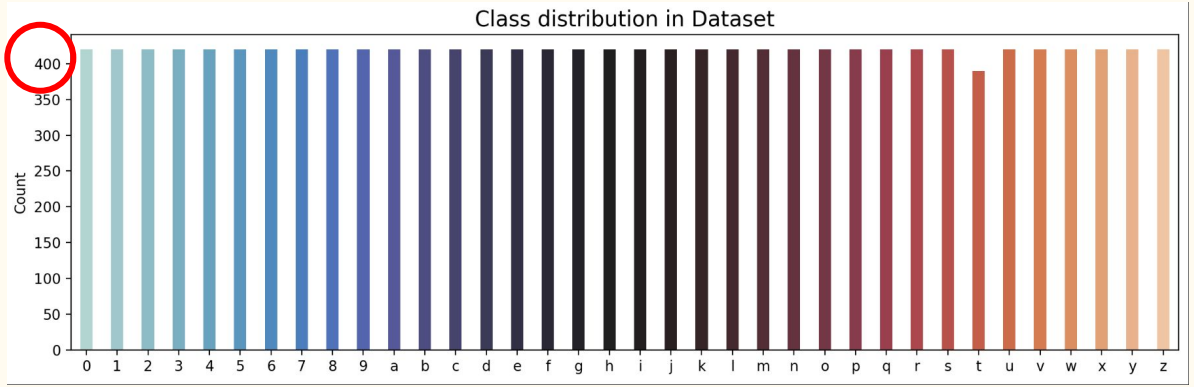


DATASET(EDA)

Original Dataset

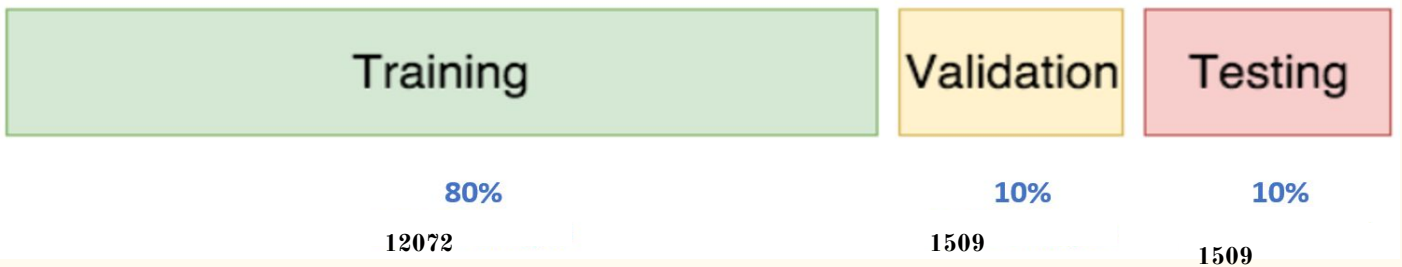


New Dataset



Dataset Preparation

- The new dataset is randomly **shuffled** and then **split** into the training, validation, and testing sets

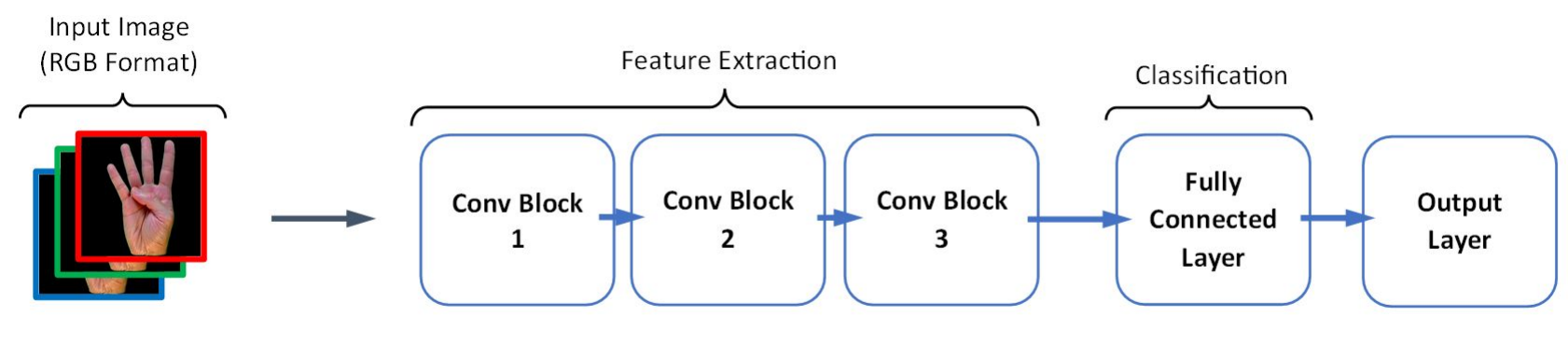


- Each image file is **200*200** pixels, and saved in an **RGB** format.
- The data is normalized.

DEEP LEARNING MODEL

The architecture of the CNN Classification model consists of the following Layers and components:

- The model contains the 3 blocks of convolution with increasing filters(32,64,128) and dropout (0.2,0.3,0.4) and activation function relu.
- Each convolution block contains Max pooling (pool_size = 2).
- The fully connected layers contain Flatten layer, Dense layer with 512 units and dropout layer with rate 0.2 and another dense layer with 128 units with dropout of 0.3 with relu activation.
- The output layer is a Dense layer with 36 units and softmax activation.



PARAMETERS (Hyper-Parameter Tuning)

Optimizer- Adam

Loss- Categorical Crossentropy

Activation_function- RELU

Output_Activation_Function - Softmax

Batch_size -32

Epochs- 50

Callbacks(Helps in controlling training process) -

ReduceLROnPlateau : Reduce learning rate when a metric(accuracy) has stopped improving.

EarlyStopping : Stop training when a monitored metric(loss) has stopped improving.

The process was EarlyStopped at Epochs=21

CONFUSION MATRIX

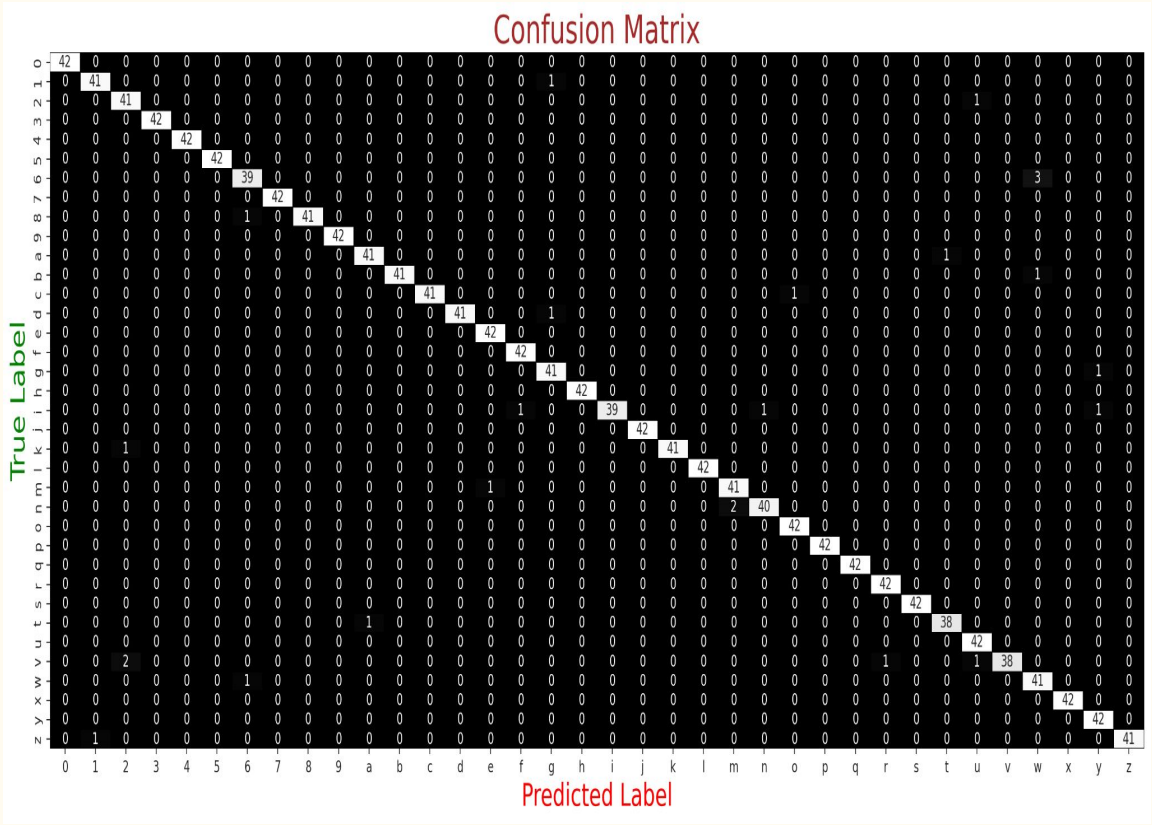
The confusion matrix shows the performance of a classification model on a held-out test set. The elements of the confusion matrix are:

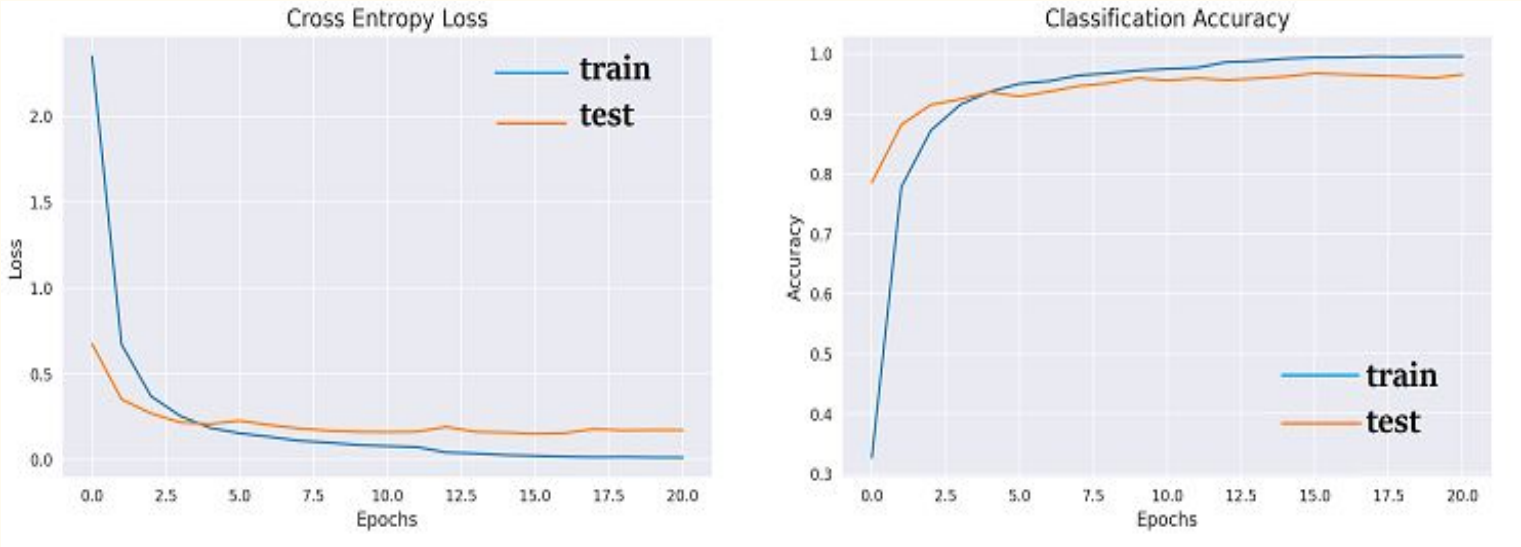
True positive(TP): The number of samples that are actually positive and are predicted as positive.

False positive(FP): The number of samples that are actually negative but are predicted as positive

True negative(TN): The number of samples that are actually negative and are predicted as negative.

False negative(FN): The number of samples that are actually positive but are predicted as negative.





Metric	Training	Validation	Test
Accuracy	100.00	96.75	98.34
Loss	0.0002	0.15	0.06

Classification Accuracy in Each Class

	precision	recall	f1-score	support
0	0.86	0.94	0.90	47
1	1.00	0.94	0.97	47
2	0.89	0.98	0.93	55
3	1.00	1.00	1.00	47
4	0.98	1.00	0.99	47
5	1.00	0.98	0.99	47
6	0.95	0.83	0.89	47
7	0.98	1.00	0.99	47
8	1.00	0.98	0.99	52
9	0.98	1.00	0.99	57
a	0.95	0.89	0.92	47
b	1.00	0.98	0.99	47
c	1.00	0.98	0.99	47
d	0.98	0.98	0.98	47
e	1.00	0.98	0.99	47
f	1.00	0.98	0.99	47
g	0.96	0.98	0.97	47
h	1.00	1.00	1.00	47
i	1.00	0.96	0.98	47
j	0.98	1.00	0.99	47
k	0.98	0.96	0.97	47
l	1.00	1.00	1.00	47
m	0.90	0.98	0.94	47

	precision	recall	f1-score	support
n	0.92	0.96	0.94	47
o	0.95	0.87	0.91	47
p	1.00	1.00	1.00	47
q	1.00	1.00	1.00	47
r	0.98	0.96	0.97	47
s	0.96	0.96	0.96	47
t	0.93	0.93	0.93	44
u	0.88	0.98	0.93	47
v	0.93	0.87	0.90	47
w	0.87	0.96	0.91	47
x	1.00	1.00	1.00	47
y	1.00	1.00	1.00	47
z	1.00	0.96	0.98	47
accuracy			0.97	1712
macro avg	0.97	0.97	0.97	1712
weighted avg	0.97	0.97	0.97	1712

$$\text{Precision} = \frac{TP}{TP + FP}$$

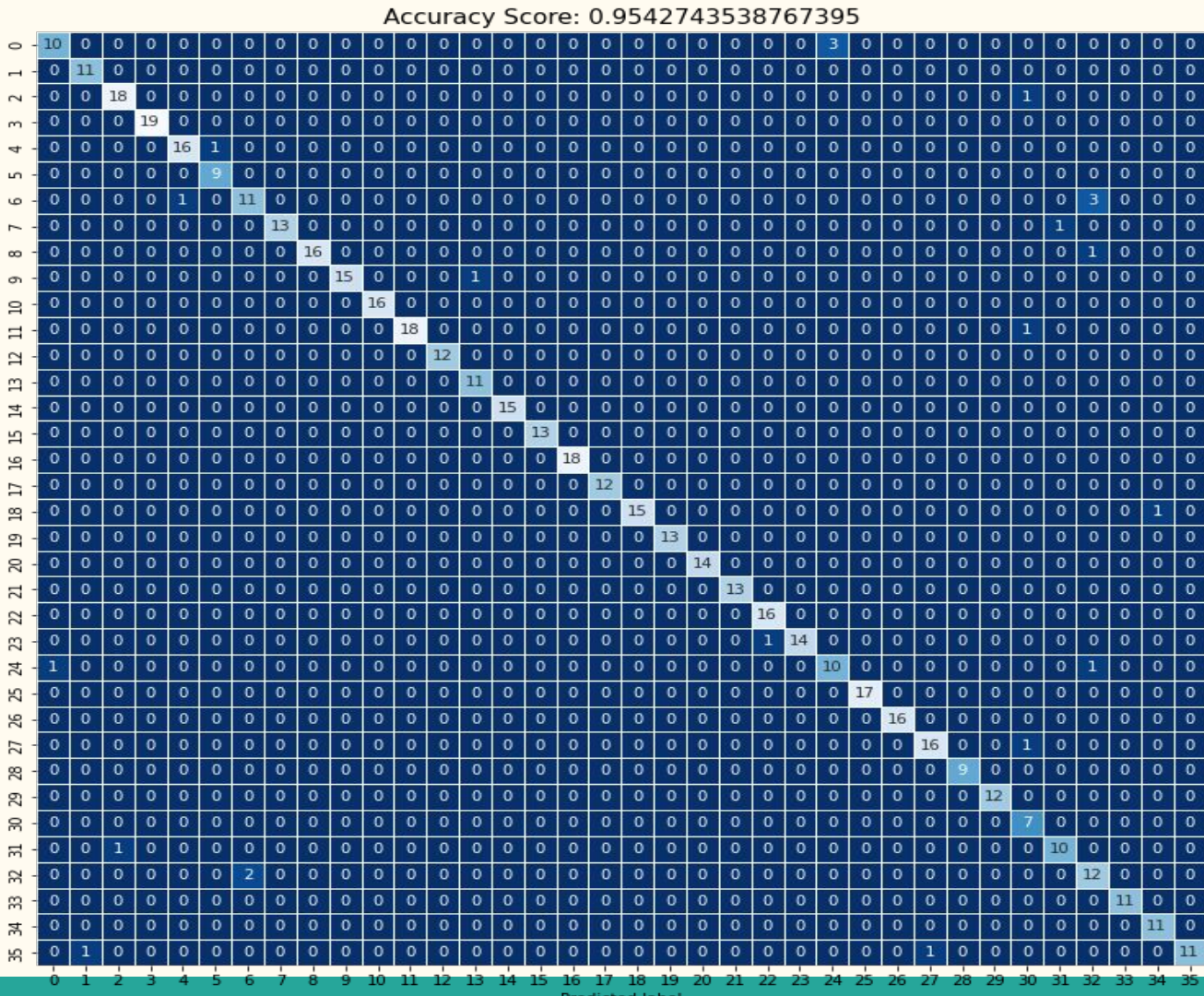
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

What all have we Tried?

1) **Run the model before expanding the dataset.**
2) Black and white, grayscale Images.
3) Using other less complex Models.
4) Hyper-Parameter tuning.
5) Using real time uncurated Hand signs

Before Expanding Dataset



What all have we Tried?

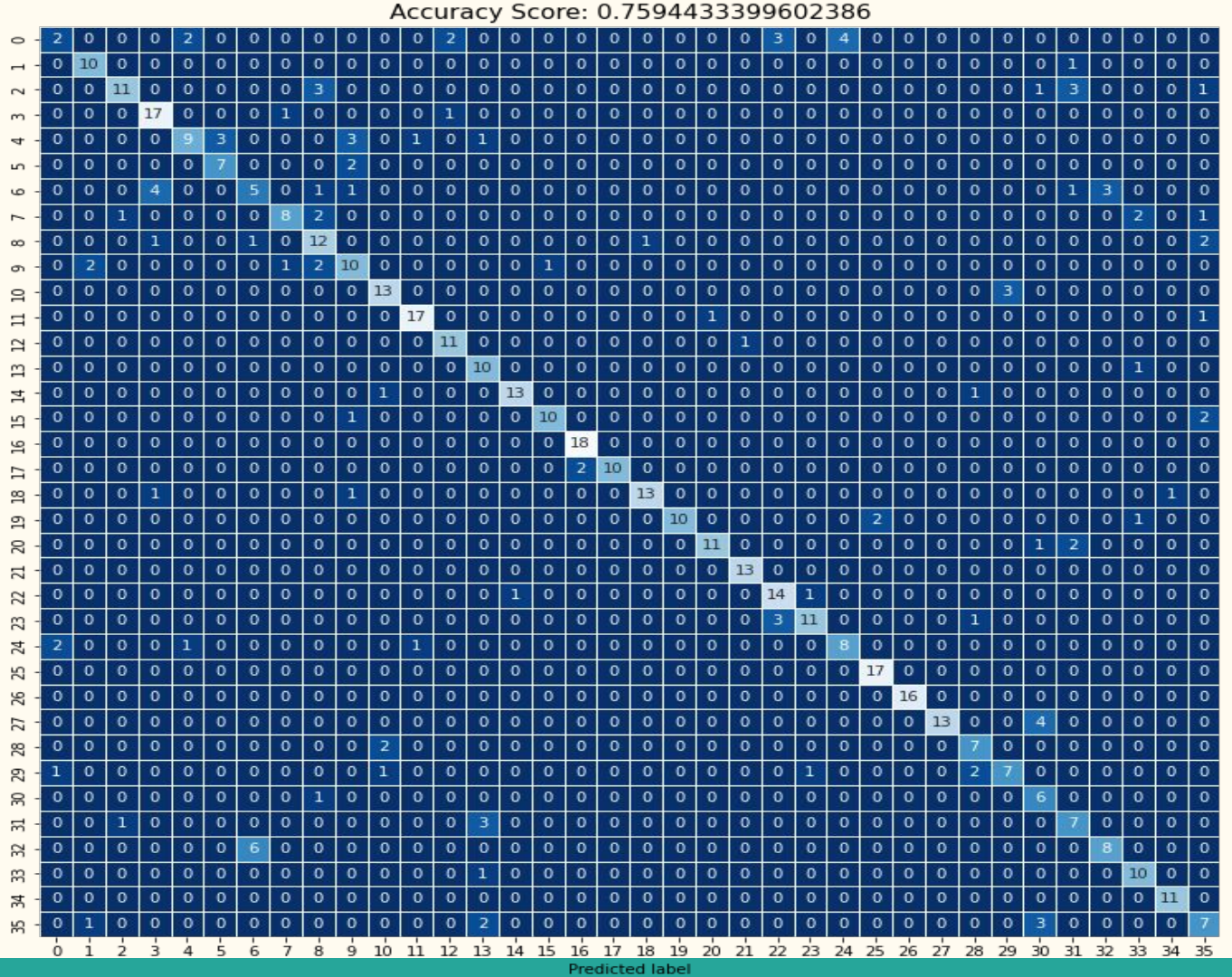
1) Run the model before expanding the dataset.
2) **Black and white, grayscale Images.**
3) Using other less complex Models.
4) Hyper-Parameter tuning.
5) Using real time uncurated Hand signs

What all have we Tried?

1) Run the model before expanding the dataset.
2) Black and white, grayscale Images.
3) **Using other less complex Models.**
4) Hyper-Parameter tuning.
5) Using real time uncurated Hand signs

We first tried.....

Decision Tree



What all have we Tried?

1) Run the model before expanding the dataset.
2) Black and white, grayscale Images.
3) Using other less complex Models.
4) **Hyper-Parameter tuning.**
5) Using real time uncurated Hand signs

What all have we Tried?

1) Run the model before expanding the dataset.
2) Black and white, grayscale Images.
3) Using other less complex Models.
4) Hyper-Parameter tuning.
5) **Using real time uncurated Hand signs**


T Y B D F S F P T H
O H N B W I X L Q S
U space A I B T E P Z space
T V G C U W nothing T E G
S Z X C P L G A U B
Q del N space nothing B G R W K
L I A space C Q O E O G
G Q E K E K W L A T
nothing E I U P A G R N O
G S C B R I M G S E

Thank You.