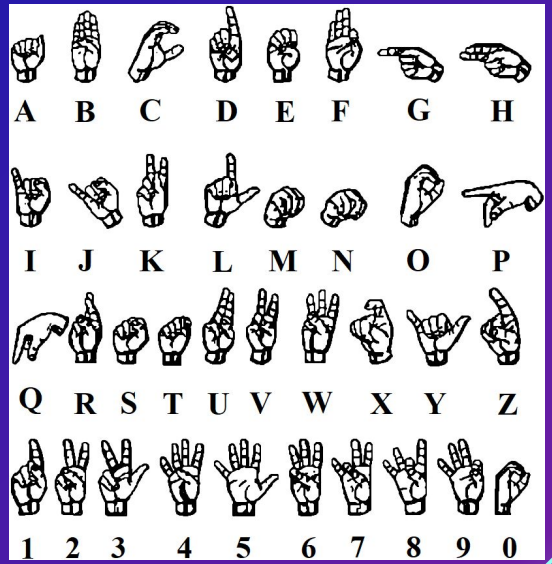
COMPUTER VISION
Sign Language Recognition

TABLE OF CONTENTS

01. Introduction

What is Sign Language?

- A highly versatile and expressive form of communication, used predominantly by the deaf and hard of hearing community.
- A combination of hand gestures, facial expressions, and body movements.
- Allow communication without relying on spoken language.



Our goal

- We realize that there is a communication barrier between Sign Language users & normal people.

➔ Bridge this gap by trying to interpret sign language, translating it into spoken or written language.

Expected Performance

- Input: An image of hand pose
- Output: The character that the pose represents

A

02. Preparation

Our Dataset

- American Sign Language Dataset | Kaggle
- 36 labels, correspond to 26 alphabet characters and 10 numbers (from 0 to 9).
- 70 images / label ⇨ 2520 samples total
- Divide into:
  - 1614 training samples (64%)
  - 403 validating samples (16%)
  - 503 evaluating samples (20%).

Pre-processing

- Use images' original size to train the model ⇨ too long
  - ➡ Resize to 64 by 64 pixels before training ⇨ accuracy not great
    - ➡ Resize to 256 by 256 pixels and apply Data Augmentation for potential performance improvement.

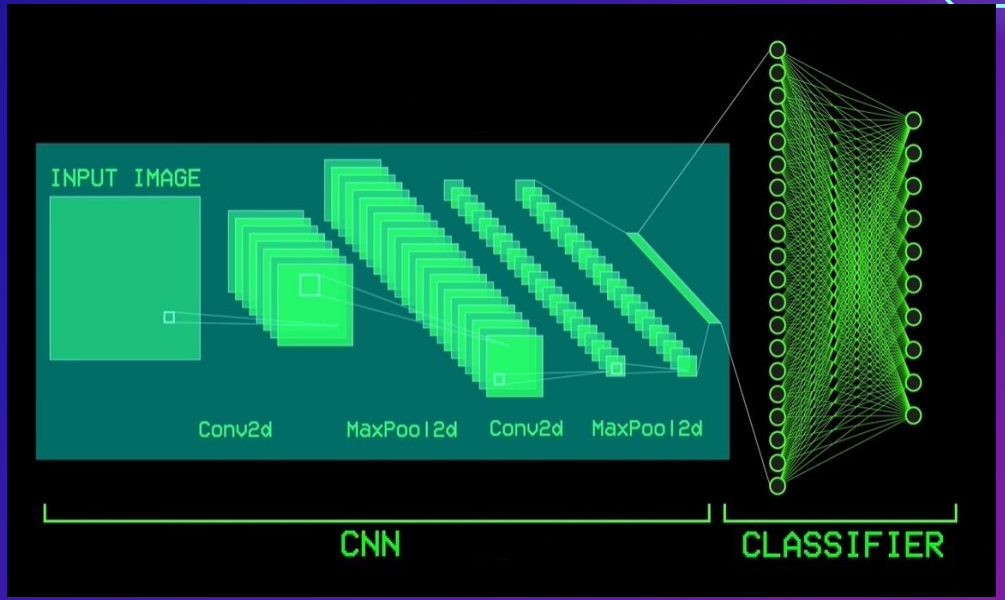
03. Model Training

Convolutional Neural Network

A type of deep neural network commonly used in image and video analysis tasks.

CNN consists of

- Convolutional layers.
- Pooling layers.
- Fully-connected layers.



Training

- **1 input layer:**

Input dimension = Shape of image (256 x 256 x 3 RGB channels)

- **7 hidden layers:**
  - **Conv2D layers:** Create a convolution kernel to produce a tensor of output
  - **Max Pooling layers:** Down sampling feature map
  - **Flatten layer:** Flattens the output of the previous layer into a 1D tensor
  - **Dense layer:** Classify image

- **1 output layer:**

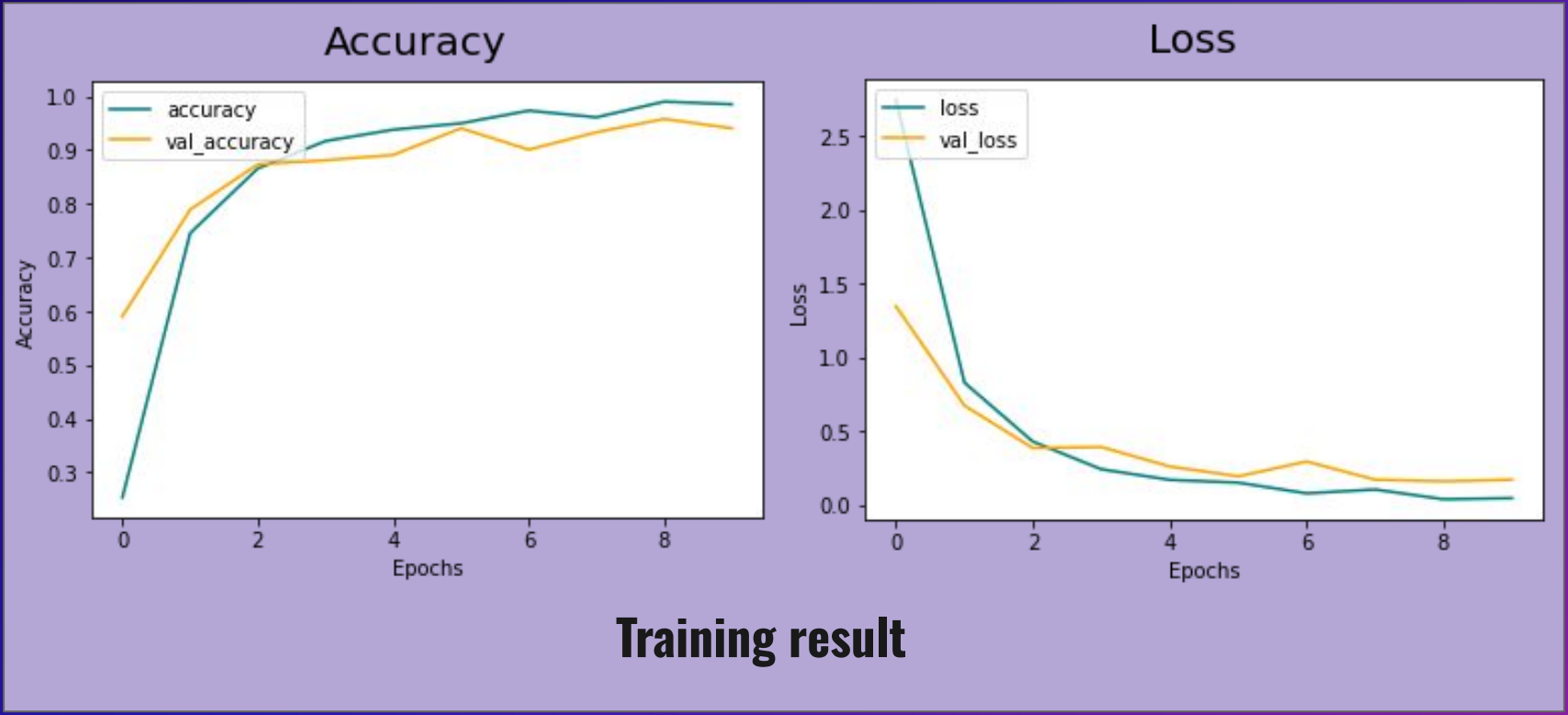
Probability of the input image belonging to one of the possible 36 classes.

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d_6 (MaxPooling 2D)	(None, 128, 128, 32)	0
conv2d_7 (Conv2D)	(None, 128, 128, 64)	18496
max_pooling2d_7 (MaxPooling 2D)	(None, 64, 64, 64)	0
conv2d_8 (Conv2D)	(None, 64, 64, 32)	18464
max_pooling2d_8 (MaxPooling 2D)	(None, 32, 32, 32)	0
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 256)	8388864
dense_5 (Dense)	(None, 36)	9252

Total params: 8,435,972
Trainable params: 8,435,972
Non-trainable params: 0

04. Result & Conclusion

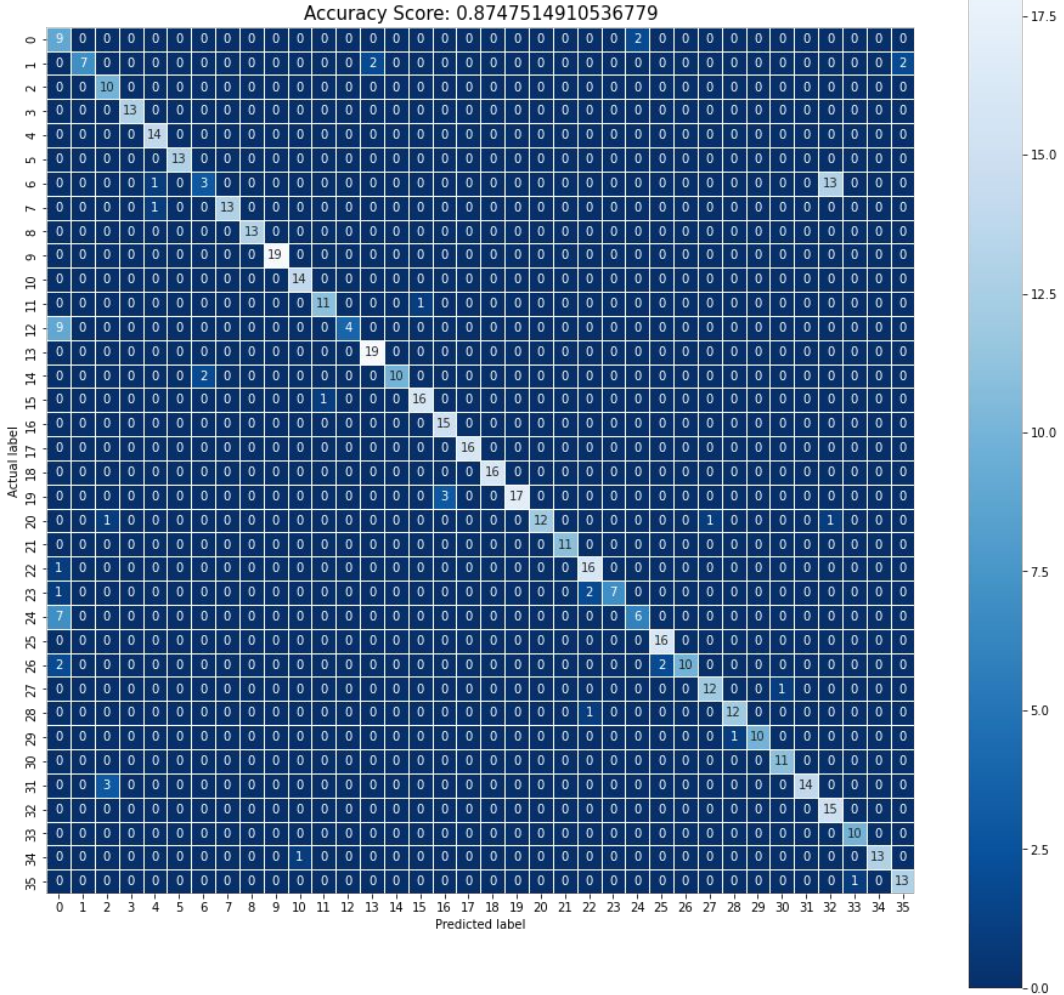
64x64 Version



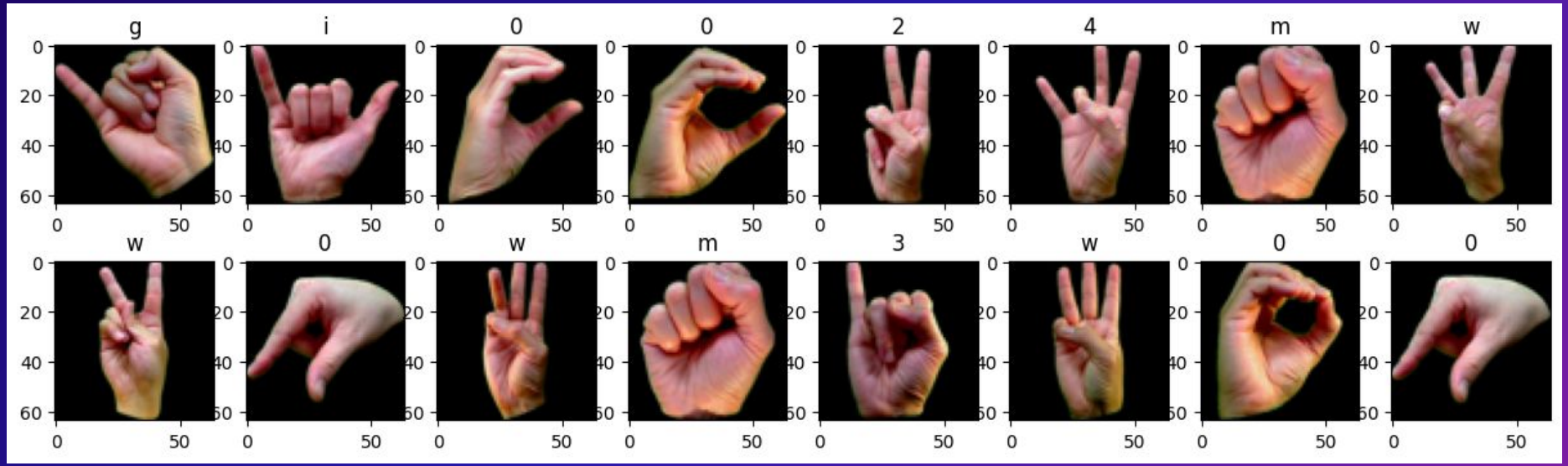
Training result

Evaluating result

Accuracy score: 87.5%

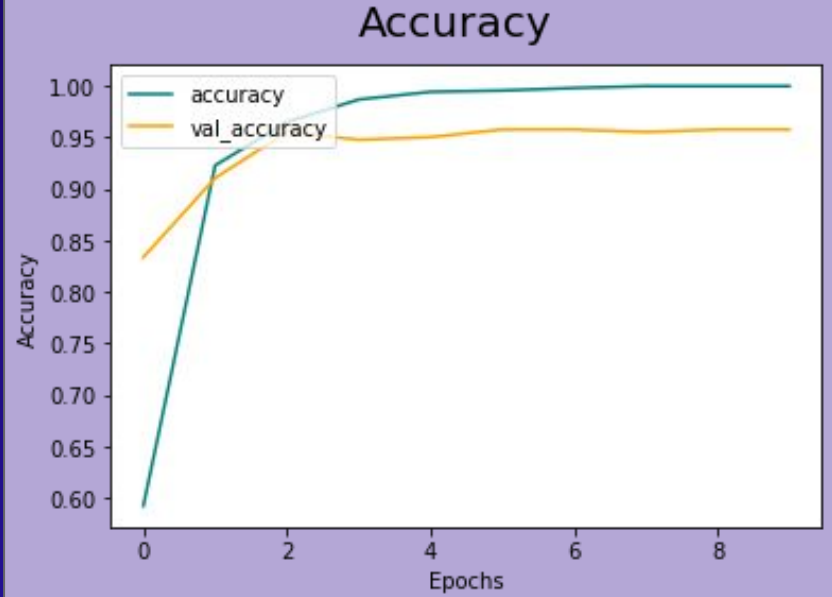


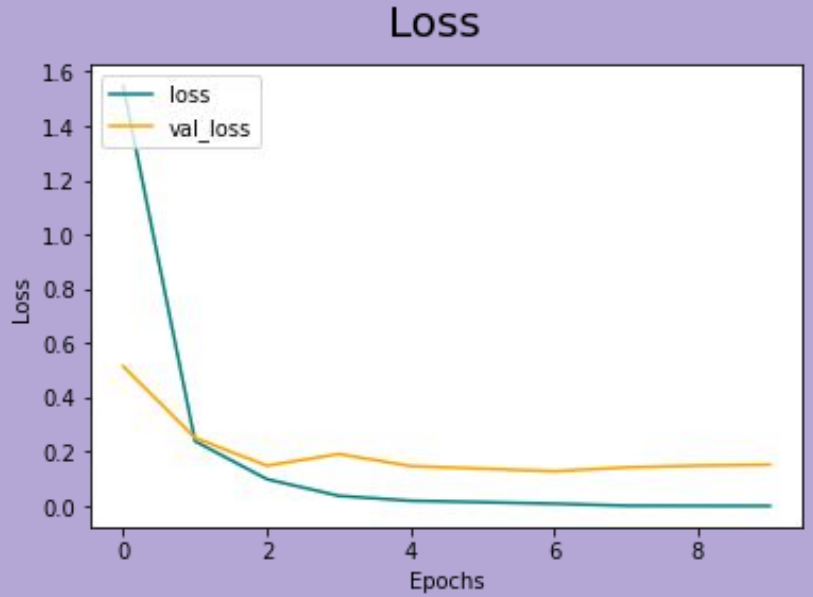
Some incorrectly predicted images (64x64)



There are totally 63 of them

256x256 Version

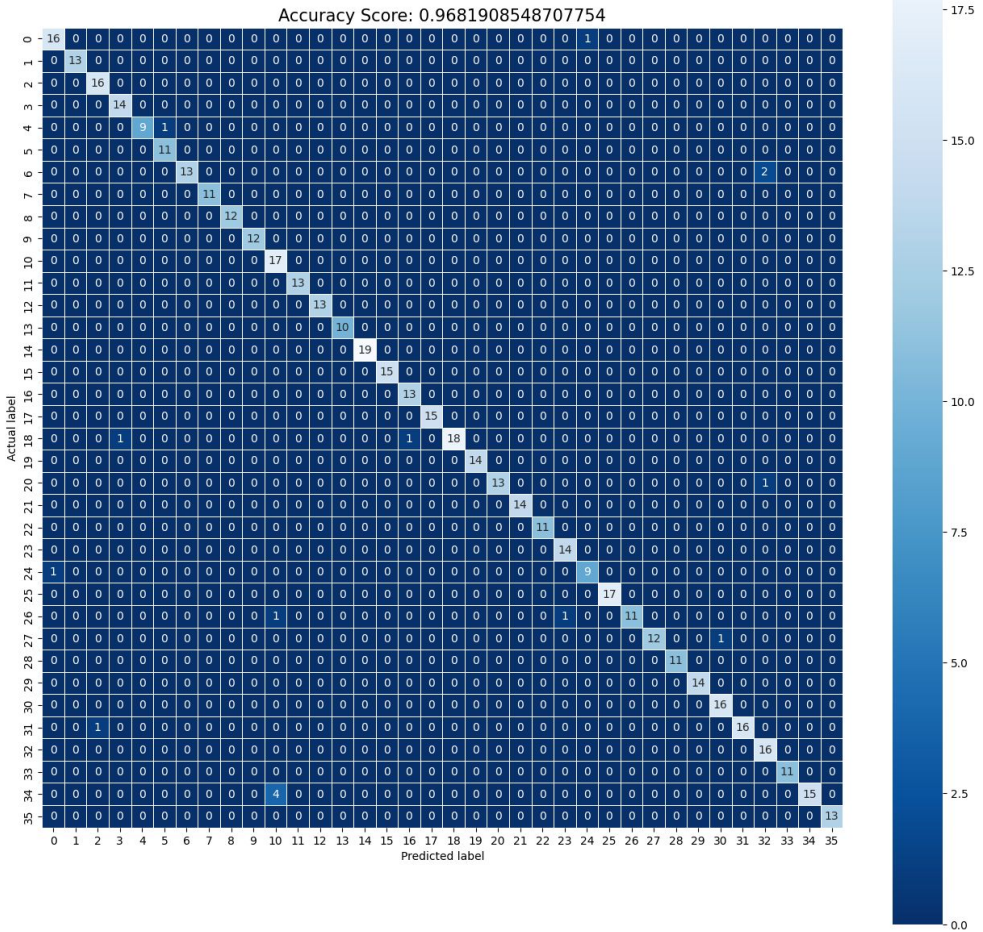




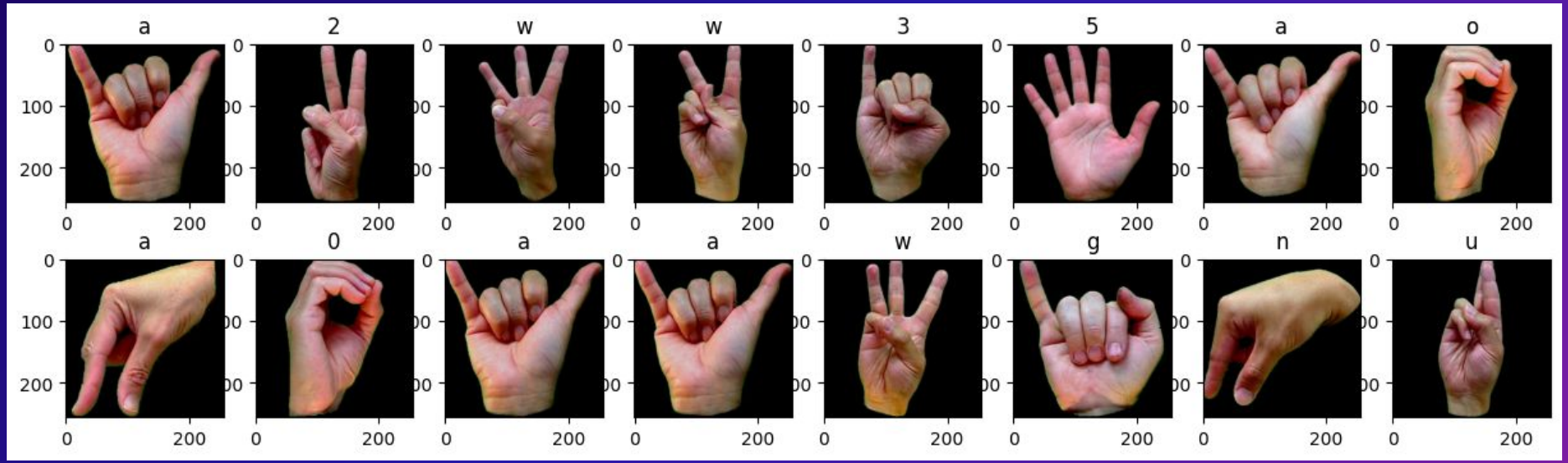
Training result

Evaluating result

Accuracy score: 96.8%

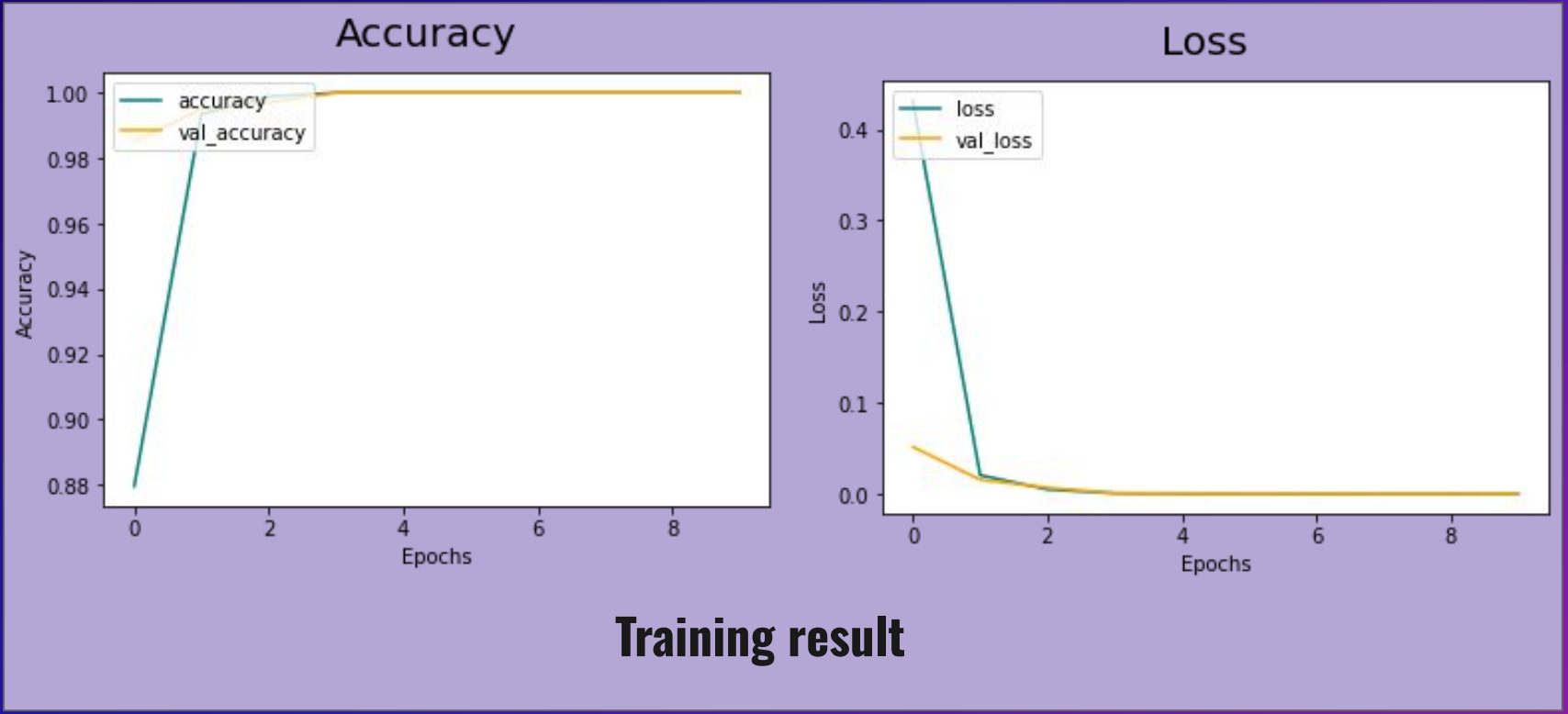


Incorrectly predicted images (256x256)



There are totally 16 of them

256x256 Version with Data Augmentation

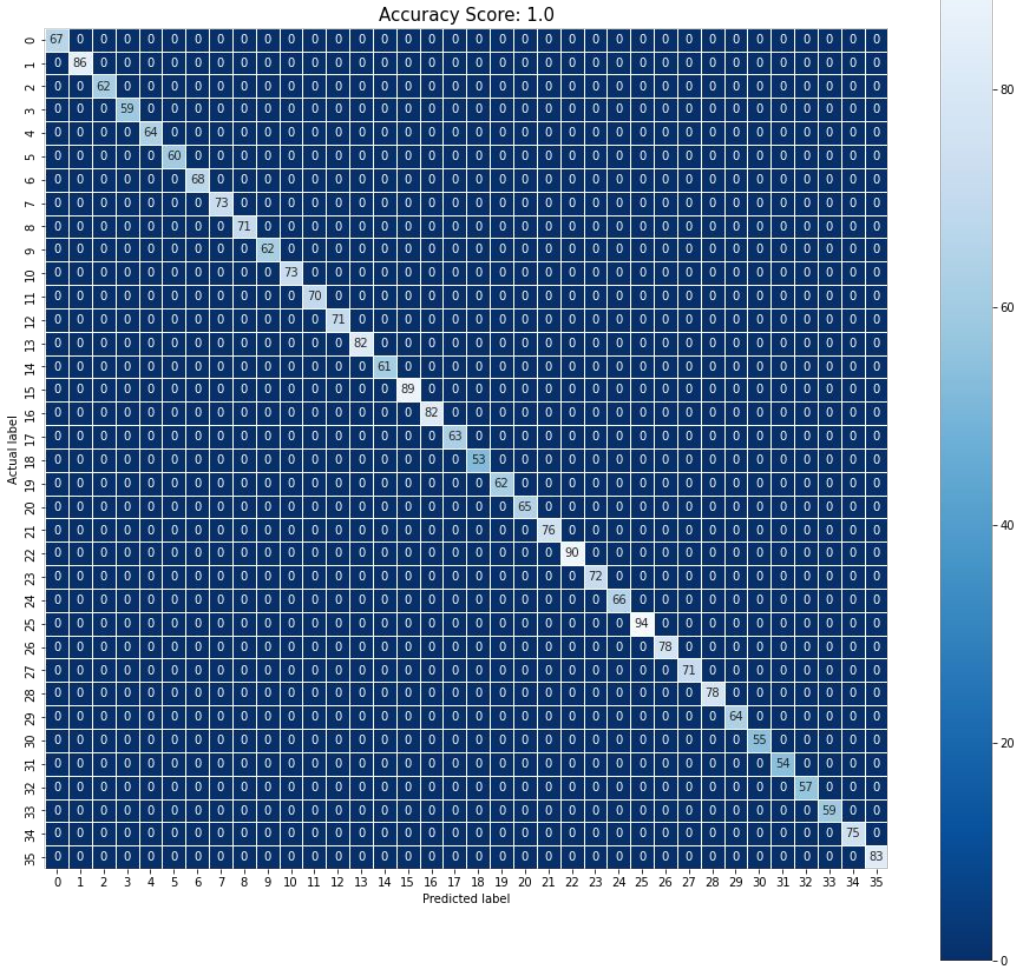


Training result

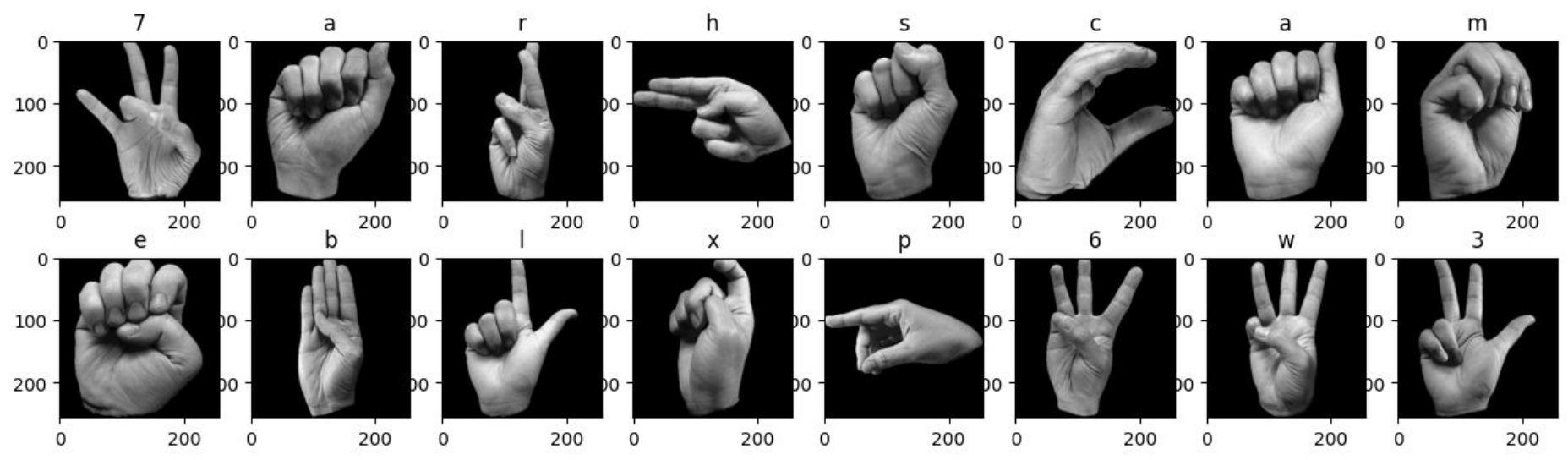
Evaluating result

Accuracy score: 100%!

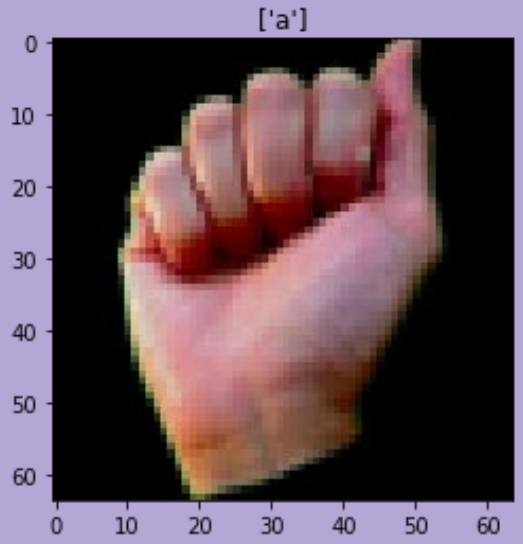
➔ Overfitting



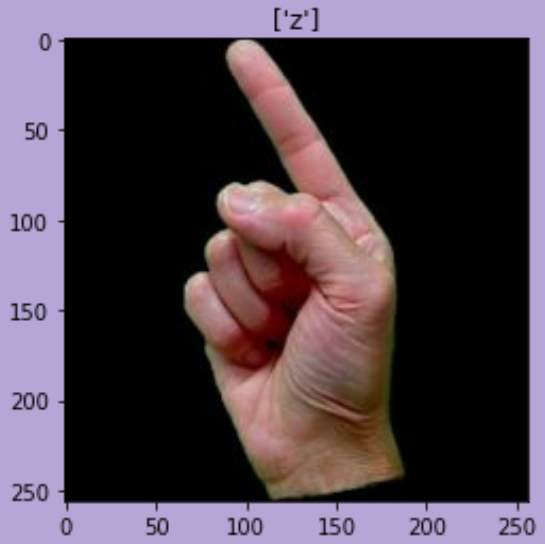
Some of the predicted images (256x256 with Data Augmentation)



Correct prediction

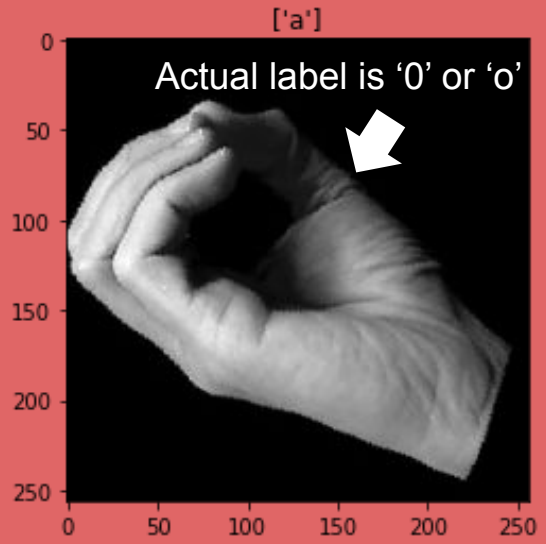


64x64



256x256

Incorrect prediction due to overfitting



256x256 with grayscale and rotation

Comparison to others models

Model	Accuracy	Source
Convolutional Neural Network	96.8%	Our model
K-Nearest-Neighbor	91.5%	Sign Classification Using sklearn libraries | Kaggle
Decision Tree	75.5%	Sign Classification Using sklearn libraries | Kaggle

Conclusion

- Resizing image to 256x256 (without Data Augmentation) and applying the CNN model gives us the best performance of 94.6% accuracy.
- Resizing the dataset images to a small size may badly affect the model accuracy, while Data Augmentation may cause overfitting.

Thank you for listening!

We would appreciate any question & comments!