# **Hand-Finger Gesture Recognition**

This project focuses on distinguishing numbers represented by hand gestures using deep learning techniques for image classification tasks.

In this project, I explore various preprocessing techniques, including morphological transformations, and delve into data augmentation using data generators to enhance diversity of training dataset.

* **Dataset Preparation: Frame Extraction from Video**

To create a dataset for training my model, I extracted individual frames from videos. Each video corresponds to a specific digit (0, 1, ..., 9), and the extracted frames are saved in subfolders based on the digit class.

This step ensures a sufficient number of labeled images for training while preserving the context of the dataset creation process.

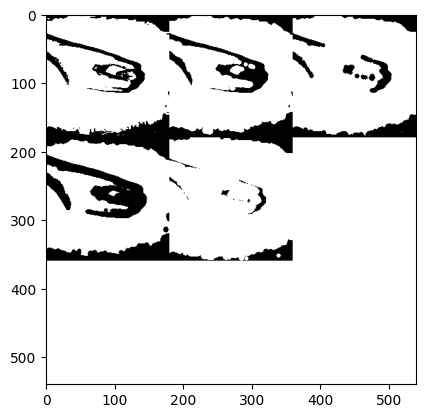
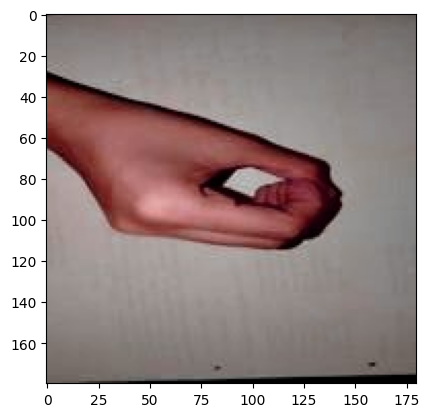
After running this code, I successfully extracted frames and stored them in organized subfolders.

Frames for digit 0 are stored in /test/0. Frames for other digits will follow the same approach, ensuring a **well-organized dataset**.

*A total of 10 videos were processed, each representing a digit class. Approximately 500 frames were extracted per video, resulting in a total dataset of 5,000 images*

* **Morphological Transformations**

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called a structuring element or kernel which decides the nature of operation.



All these can be done using the skimage.morphology module. Utilities that operate on shapes in images. These operations are particularly suited for binary images, although some may be useful for images of other types as well. I am going to use .disk().Using a circular disk of a spesific size (3) that moves across the image to perform these morphological transformations.

# **Data Augmentation**

Apply transformations like rotations, flips, zooms, etc. using ImageDataGenerator or other augmentation libraries. I use the Albumentations library for my augmentation process. Albumentations is a fast and flexible image augmentation library. The library is widely used in industry, deep learning research, machine learning competitions, and open source projects.

*RandomRotate90 and ShiftScaleRotate address spatial variability, ensuring the model doesn't overfit to a fixed orientation of the hand gestures.*

*HueSaturationValue and RandomBrightnessContrast improve the model's ability to generalize under varying lighting conditions.*

**

**Normalization** is the most crucial step in the pre-processing part. There are multiple ways to normalise images. Normalization technique where you divide pixel values by 255, which just lowers the range from 0-255 to 0-1, but the spread of pixel values over the range is kept the same. For this project, the image pixel values were normalized to the range [0, 1].

**Order of Preprocessing Steps**

As you can see, I end the preprocessing pipeline with augmentation, resize, and finally normalization.

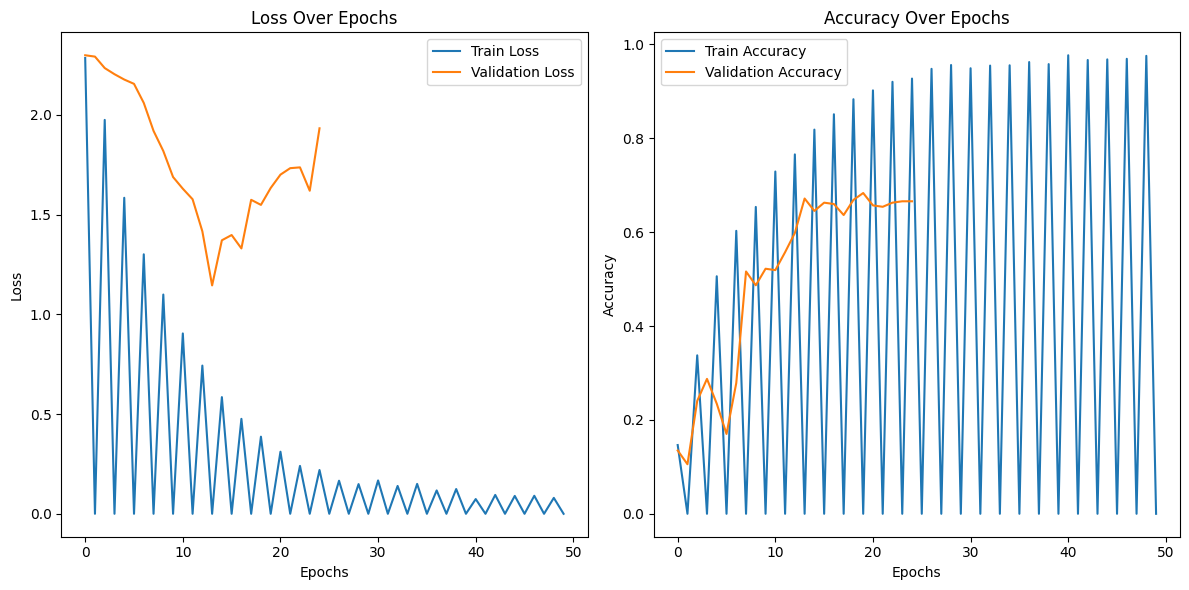
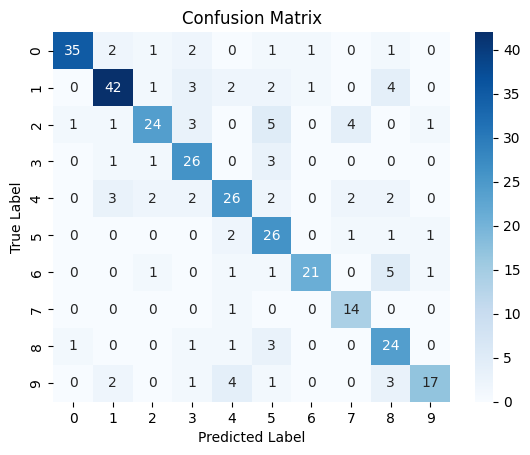
I apply augmentation to the first high-quality images. If I had done the resize first, some of the augmentations would have degraded the image quality.

I apply normalization last, obtaining stable and efficient pixel values. This is done last to avoid interfering with the augmentations that depend on the raw pixel values.

Test Loss: 1.6152

Test Accuracy: 0.7537

The model correctly classified 75% of the test images.



**References**

[**https://explore.albumentations.ai/**](https://explore.albumentations.ai/)

[**https://docs.opencv.org/4.x/d9/d61/tutorial\_py\_morphological\_ops.html**](https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html)

[**https://imagekit.io/blog/image-resizing-in-python/**](https://imagekit.io/blog/image-resizing-in-python/)

[**https://medium.com/@ricodedeijn/image-classification-computer-vision-from-scratch-pt-1-d02033edd70d**](https://medium.com/@ricodedeijn/image-classification-computer-vision-from-scratch-pt-1-d02033edd70d)