# **Instructions for Collecting and Preparing the Air-Writing Dataset**

# These are the **step-by-step instructions** for **us** (the team) to **capture, process, and label air-written digits** for training our digit recognition model.

# 

## **Step 1: Define the Dataset Format**

# Each entry in our dataset will have:

# **Images** → **Grayscale 28x28 PNG images** of air-written digits.

# **Labels** → A **CSV file (labels.csv)** mapping each image filename to its digit.

# **Folder Structure**

# airwriting\_dataset/

# │— images/ # Folder where we store air-written digit images

# │ ├── 00001.png

# │ ├── 00002.png

# │ └── ...

# │— labels.csv # CSV file mapping filenames to digits

# 

# **Example of labels.csv**

# image,label

# 00001.png,3

# 00002.png,7

# 00003.png,0

# 

# 

## **Step 2: Capture Air-Written Digits**

# **Write a digit in the air** using our mobile app.

# **Take a screenshot** when the digit is complete.

# **Save the screenshot as a PNG file**.

# **Store all images in a folder** on the device.

# **Make sure to take at least 100 images per digit** so we have enough data for training.

# 

## **Step 3: Transfer Images to a Computer**

# Since we need to process and train on the dataset, we should **move all images from the mobile device to our computer**.

# 

## **Step 4: Resize Images to 28x28 Pixels**

# Since our model is trained on **28x28 images**, we must **resize all images** before training.

### **Run the Image Resizing Script**

# Once the images are on your computer:

# **Modify the script below** to set your dataset path.

# **Run the script** to resize all images to 28x28 pixels.

# **Resize Script (resize\_images.py)**

# import os

# import cv2

# 

# # ENTER YOUR PATHS HERE

# dataset\_path = "path/to/your/data" # Folder where raw images are stored

# output\_path = "path/to/output/folder" # Folder to save processed images

# 

# # Create output directory if it doesn't exist

# if not os.path.exists(output\_path):

# os.makedirs(output\_path)

# 

# # Loop through all images in the dataset folder

# for img\_file in os.listdir(dataset\_path):

# if img\_file.endswith(('.png', '.jpg', '.jpeg')): # Check if it's an image

# img\_path = os.path.join(dataset\_path, img\_file)

# 

# # Read image and convert to grayscale

# img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)

# 

# # Step 1: Increase contrast using adaptive histogram equalization

# clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))

# contrast\_img = clahe.apply(img)

# 

# # Step 2: Apply Canny Edge Detection

# edges = cv2.Canny(contrast\_img, threshold1=50, threshold2=150)

# 

# # Step 3: Thicken edges using dilation

# kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (2, 2)) # Adjust thickness as needed

# thick\_edges = cv2.dilate(edges, kernel, iterations=1)

# 

# # Step 4: Resize to 28x28 while preserving shape

# final\_output = cv2.resize(thick\_edges, (28, 28))

# 

# # Save the processed image

# cv2.imwrite(os.path.join(output\_path, img\_file), final\_output)

# 

# print(f" All images processed and saved in: {output\_path}")

# 

## **Step 5: Label the Images**

# Instead of manually entering labels, we will **generate the labels.csv file automatically**.

# **Labeling Script (generate\_labels.py)**

# import os

# import pandas as pd

# 

# # ENTER YOUR IMAGE DIRECTORY

# dataset\_path = "path/to/your/resized\_images"

# output\_csv = os.path.join(dataset\_path, "labels.csv")

# 

# # Create a list to store filenames and labels

# data = []

# 

# # Manually enter labels OR modify this to auto-detect digits later

# for img\_file in os.listdir(dataset\_path):

# if img\_file.endswith(".png"):

# digit\_label = input(f"Enter the digit for {img\_file}: ") # Ask the user to confirm

# data.append([img\_file, digit\_label])

# 

# # Save to CSV

# df = pd.DataFrame(data, columns=["image", "label"])

# df.to\_csv(output\_csv, index=False)

# print(f" Labels saved to: {output\_csv}")

# 

# 