**Steps To Preprocess your image dataset**  
you guys can use this as sample steps and take help from it

The document includes the following steps:

1. **Understanding the Dataset:**
   * Overview of the dataset structure and file formats.
   * Insights on inspecting images for quality and resolution.
2. **Environment Setup:**
   * Necessary libraries and packages to install.
3. **Preprocessing Steps:**
   * Resizing images for uniformity.
   * Normalizing pixel values.
   * Converting images to grayscale (if applicable).
   * Applying data augmentation techniques (rotation, flipping, noise addition, etc.).
   * Removing noise using filters (Gaussian or bilateral).
   * Applying thresholding for OCR-specific tasks.
4. **Organizing the Data:**
   * Splitting the dataset into training, validation, and test sets.
5. **Saving Preprocessed Data:**
   * Guidelines for saving processed images for future use.
6. **Verification:**
   * Visualizing preprocessed images to ensure correctness.

**1. Understand the Dataset**

* Review the dataset structure (e.g., folders, labels).
* Check the number of images and their resolution.
* Identify the file formats (e.g., JPEG, PNG).

**2. Set Up the Environment**

* **Install necessary libraries:**
* pip install numpy pandas opencv-python scikit-image matplotlib pillow tensorflow
* **Import libraries in Python:**
* import os
* import cv2
* import numpy as np
* from skimage import io, transform
* import matplotlib.pyplot as plt

**3. Load and Inspect the Data**

* **Read the images and visualize a few to understand their quality.**
* data\_path = "path/to/dataset"
* for img\_file in os.listdir(data\_path):
* img\_path = os.path.join(data\_path, img\_file)
* image = cv2.imread(img\_path)
* plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))
* plt.show()
* break

**4. Preprocessing Steps**

**a) Resize Images**

**Standardize the image size for consistency:**

def resize\_image(image, size=(224, 224)):

return cv2.resize(image, size)

**b) Normalize Pixel Values**

**Normalize pixel values to a range of [0, 1] (important for neural networks):**

def normalize\_image(image):

return image / 255.0

**c) Convert to Grayscale (if needed)**

**Convert images to grayscale if the model doesn’t need color channels:**

def to\_grayscale(image):

return cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

**d) Data Augmentation**

**To increase the dataset’s diversity, apply transformations such as:**

* **Rotation**
* **Flipping (horizontal/vertical)**
* **Adding noise**
* **Scaling**
* from tensorflow.keras.preprocessing.image import ImageDataGenerator
* datagen = ImageDataGenerator(
* rotation\_range=15,
* width\_shift\_range=0.1,
* height\_shift\_range=0.1,
* shear\_range=0.1,
* zoom\_range=0.1,
* horizontal\_flip=True,
* fill\_mode="nearest",
* )
* # Example for augmenting one image
* augmented\_image = next(datagen.flow(np.expand\_dims(image, axis=0)))[0]
* plt.imshow(augmented\_image)
* plt.show()

**e) Remove Noise**

**Use Gaussian blur or bilateral filtering to reduce noise:**

def denoise\_image(image):

return cv2.GaussianBlur(image, (5, 5), 0)

**f) OCR-Specific Preprocessing**

* **Thresholding (binary or adaptive):**
* def threshold\_image(image):
* return cv2.adaptiveThreshold(
* cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY), 255,
* cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 11, 2

)

**5. Organize Data**

* **Split data into training, validation, and test sets:**
* from sklearn.model\_selection import train\_test\_split
* train\_images, test\_images = train\_test\_split(images, test\_size=0.2, random\_state=42)
* train\_images, val\_images = train\_test\_split(train\_images, test\_size=0.25, random\_state=42)

**6. Save Preprocessed Data**

**Save the processed images for later use:**

output\_path = "path/to/save/preprocessed"

os.makedirs(output\_path, exist\_ok=True)

for i, image in enumerate(processed\_images):

cv2.imwrite(os.path.join(output\_path, f"image\_{i}.jpg"), image)

**7. Verify Preprocessed Data**

* Visualize some preprocessed images to ensure correctness.
* Check for class balance in the dataset.