```
1 Start coding or generate with AI.
 1 import os
 2 import numpy as np
 3 import pandas as pd
 4 import cv2
 5 import matplotlib.pyplot as plt
  6 import seaborn as sns
 7 from PIL import Image
 9 import tensorflow as tf
10 from tensorflow import keras
11 from tensorflow.keras import layers, models
12 from tensorflow.keras.models import Model
13 from tensorflow.keras.layers import Conv2D, MaxPooling2D, UpSampling2D, Input
{\tt 14\ from\ tensorflow.keras.layers\ import\ BatchNormalization, Concatenate}
15 import tensorflow.keras.backend as K
16 from tensorflow.keras.preprocessing.image import ImageDataGenerator
17 from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPooling2D, UpSampling2D, Concatenate
18 from sklearn.metrics import mean_squared_error
 1 from google.colab import drive
 2 drive.mount('/content/drive')

→ Mounted at /content/drive
```

1. Brief description of the problem and data

Description

In this project, we aim to develop a machine learning model to remove noise from scanned text documents and restore them to their clean, readable versions. Optical Character Recognition (OCR) is widely used to digitize printed and handwritten documents, but real-world issues like stains, wrinkles, and smudges make OCR less effective. The goal of this project is to create a document enhancement model that improves text clarity by removing noise while preserving the original content.

The methodology will include the following stages:

- · Brief Description of the Problem and Data
- Data Preprocessing and Exploratory Data Analysis (EDA)
- · Model Architecture and Development
- Model Evaluation
- · Results and Analysis
- Conclusion

Data

The dataset consists of scanned text images with synthetic noise added to simulate real-world document degradation. It contains:

- train/: Noisy images used for training.
- train_cleaned/: Corresponding clean (ground truth) versions of the images.
- $\bullet\;$ test/: Noisy test images where the model must remove noise.
- sampleSubmission.csv: A sample submission file that shows the required format for Kaggle.

Each image is grayscale, with pixel intensity values ranging from 0 (black) to 1 (white). The submission format requires flattening the image into a list of pixel values, each assigned a unique ID (image_row_col).

```
1 # Define dataset paths
  2 dataset_path = "/content/drive/My Drive/DTSA5510/final/"
  3 train_path = os.path.join(dataset_path, "train")
  4 train_cleaned_path = os.path.join(dataset_path, "train_cleaned")
  5 test_path = os.path.join(dataset_path, "test")
  1 # Count images each folders
  2 def count_images(directory):
        return len(os.listdir(directory))
  5 print("Number of Train Images:", count_images(train_path))
  6 print("Number of Train Cleaned Images:", count_images(train_cleaned_path))
  7 print("Number of Test Images:", count_images(test_path))
Number of Train Images: 144
Number of Train Cleaned Images: 144
     Number of Test Images: 72
  1 # Inspect
  2 print("Train set sample files:", os.listdir(train_path)[:5])
  3 print("Train Cleaned set sample files:", os.listdir(train_cleaned_path)[:5])
  4 print("Test set sample files:", os.listdir(test_path)[:5])
Train set sample files: ['33.png', '8.png', '42.png', '45.png', '15.png']
Train Cleaned set sample files: ['56.png', '36.png', '60.png', '47.png', '8.png']
Test set sample files: ['16.png', '40.png', '79.png', '91.png', '13.png']
  1 # Inspect and display images
  \label{eq:constrain_path} 2 \ \mathsf{sample\_noisy} \ = \ \mathsf{os.path.join}(\mathsf{train\_path}, \ \mathsf{os.listdir}(\mathsf{train\_path})[\emptyset])
  3 sample_clean = os.path.join(train_cleaned_path, os.listdir(train_cleaned_path)[0])
  4 sample_test = os.path.join(test_path, os.listdir(test_path)[0])
  6 def display_images(img_paths, titles, figsize=(15, 5)):
         fig, axes = plt.subplots(1, len(img_paths), figsize=figsize)
```

```
8
      for ax, img path, title in zip(axes, img paths, titles):
           img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
10
           ax.imshow(img, cmap='gray')
11
           ax.set_title(title)
12
           ax.axis("off")
13
      plt.show()
14
15 display_images(
16
      [sample_noisy, sample_clean, sample_test],
       ["Noisy (Train)", "Clean (Train Cleaned)", "Noisy (Test)"]
17
18)
```

₹ Noisy (Train)

NOISY (IraIn)

There are several classic spatial filters for reducing or elimin from images. The mean filter, the median filter and the closing o used. The mean filter is a lowpass or smoothing filter that replace neighborhood mean. It reduces the image noise but blurs the image calculates the median of the pixel neighborhood for each pixel, the effect. Finally, the opening closing filter is a mathematical morphic has ame number of erosion and dilation morphological operations objects from images.

The main goal was to train a neural network in a supervised image from a noisy one. In this particular case, it was much easier image from a clean one than to clean a subset of noisy images.

Clean (Train Cleaned)

There exist several methods to design forms with fi instance, fields may be surrounded by bounding boxes, guiding rulers. These methods specify where to write the effect of skew and overlapping with other parts of can be located on a separate sheet of paper that is lo they can be printed directly on the form. The use of; is much better from the point of view of the quality or requires giving more instructions and, more importantly tasks where this type of acquisition is used. Guiding are more commonly used for this reason. Light rectang easily with filters than dark lines whenever the handw

Noisy (Test)

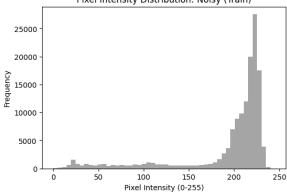
NOISY (1981)

A new offline handwritten database for the Spanish lang Spanish sentences, has recently been developed: the Spartacu for Spanish Restricted-domain Task of Cursive Script). There creating this corpus. First of all, most databases do not contai though Spanish is a widespread major language. Another impo a corpus from semantic-restricted tasks. These tasks are comm allow the use of linguistic knowledge beyond the lexicon level ir. As the Spartacus database consisted mainly of short sente long paragraphs, the writers were asked to copy a set of sentences one-line fields in the forms. Next figure shows one of the forn process. These forms also contain a brief set of instructions give

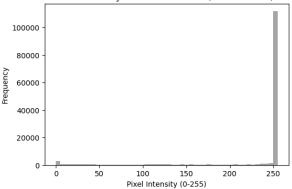
```
1 # Analyze Pixel Intensity Distribution
 2 def plot_histogram(img_path, title):
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        plt.figure(figsize=(6, 4))
        plt.hist(img.ravel(), bins=50, color='gray', alpha=0.7)
plt.title(f"Pixel Intensity Distribution: {title}")
        plt.xlabel("Pixel Intensity (0-255)")
        plt.ylabel("Frequency")
        plt.show()
10
11 \mbox{\tt\#} Plot histograms for noisy, cleaned, and test images
12 plot_histogram(sample_noisy, "Noisy (Train)")
13 plot_histogram(sample_clean, "Clean (Train Cleaned)")
14 plot_histogram(sample_test, "Noisy (Test)")
```

∓*

Pixel Intensity Distribution: Noisy (Train)



Pixel Intensity Distribution: Clean (Train Cleaned)



Pixel Intensity Distribution: Noisy (Test)

```
25000
20000
15000
```

```
1 # Image Size Consistency
2 def check_image_shapes(directory):
     shapes = []
      for img_name in os.listdir(directory):
          img_path = os.path.join(directory, img_name)
```

```
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
shapes.append(img.shape)
return set(shapes)

fterurn set(sha
```

EDA Summary

- Dataset Completeness: The dataset contains 144 training images with corresponding clean versions and 72 test images, ensuring full
 data availability for training and evaluation.
- Image Size Variability: The dataset has two unique image sizes, (420, 540) and (258, 540), requiring resizing for model consistency.
- Noise Characteristics: Noisy images contain dark artifacts, stains, and wrinkles, causing pixel intensity distributions to be more spread
 out compared to the clean images, which are mostly concentrated near 255 (white background).
- Train vs. Test Similarity: The test images exhibit similar noise patterns as the training set, confirming that denoising methods trained
 on the dataset should generalize well to test images.
- Preprocessing Needs: Due to varied noise types and intensity distributions, preprocessing should include grayscale conversion, normalization, and potential adaptive thresholding before applying denoising techniques.

3. Model Architecture and Development

Data Processing: Normalize pixel values and resize.

```
1 # Set image dimensions
 2 IMG_HEIGHT, IMG_WIDTH = 256, 256
 1 # Data Processing Functions
 2 def load_images(directory, resize_shape=(IMG_HEIGHT, IMG_WIDTH)):
       images = []
       for img_name in sorted(os.listdir(directory)):
           img_path = os.path.join(directory, img_name)
           img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
 6
           img = cv2.resize(img, resize_shape)
           img = img.astype("float32") / 255.0
           images.append(img)
10
       return np.array(images).reshape(-1, resize_shape[0], resize_shape[1], 1)
 1 # Load noisy and clean images
 2 train_noisy = load_images(train_path)
3 train_clean = load_images(train_cleaned_path)
 5 # Print dataset shape
 6 print(f"Train Noisy Shape: {train_noisy.shape}")
 7 print(f"Train Clean Shape: {train_clean.shape}")
→ Train Noisy Shape: (144, 256, 256, 1)
    Train Clean Shape: (144, 256, 256, 1)
Define Convolutional Autoencoder Model
 1 # Define the autoencoder architecture
 2 def build_autoencoder(input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)):
      input_img = Input(shape=input_shape)
       # Encoder
       x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
       x = MaxPooling2D((2, 2), padding='same')(x)
 8
       x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
       x = MaxPooling2D((2, 2), padding='same')(x)
10
       # Decoder
11
       x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
13
       x = UpSampling2D((2, 2))(x)
       x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
14
15
       x = UpSampling2D((2, 2))(x)
       output_img = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x) # Output layer
16
17
18
19
       autoencoder = Model(input_img, output_img)
20
       autoencoder.compile(optimizer='adam', loss='mse')
21
       return autoencoder
 1 # Instantiate the model
 2 autoencoder = build_autoencoder()
 1 # Print model summary
 2 autoencoder.summary()
```

→ Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 256, 256, 1)	0
conv2d (Conv2D)	(None, 256, 256, 32)	320
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_1 (Conv2D)	(None, 128, 128, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	36,928
up_sampling2d (UpSampling2D)	(None, 128, 128, 64)	0
conv2d_3 (Conv2D)	(None, 128, 128, 32)	18,464
up_sampling2d_1 (UpSampling2D)	(None, 256, 256, 32)	0
conv2d_4 (Conv2D)	(None, 256, 256, 1)	289

Total params: 74,497 (291.00 KB)

Train the Model

```
→ Epoch 1/30
                            - 11s 452ms/step - loss: 0.1628 - val_loss: 0.0590
    9/9 -
    Epoch 2/30
                             - 0s 42ms/step - loss: 0.0677 - val_loss: 0.0621
    9/9 -
    Epoch 3/30
                             - 0s 42ms/step - loss: 0.0697 - val loss: 0.0621
    9/9 -
    Epoch 4/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0700 - val_loss: 0.0621
    Epoch 5/30
                              0s 42ms/step - loss: 0.0696 - val_loss: 0.0621
    Epoch 6/30
    9/9
                              0s 42ms/step - loss: 0.0684 - val_loss: 0.0621
    Epoch 7/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0694 - val_loss: 0.0621
    Epoch 8/30
                             - 0s 42ms/step - loss: 0.0691 - val_loss: 0.0621
    9/9 -
    Epoch 9/30
                             - 0s 42ms/step - loss: 0.0694 - val_loss: 0.0621
    9/9 -
    Epoch 10/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0701 - val_loss: 0.0621
    Epoch 11/30
                            - 0s 41ms/step - loss: 0.0698 - val_loss: 0.0621
    Enoch 12/30
                              0s 42ms/step - loss: 0.0699 - val_loss: 0.0621
    Epoch 13/30
                             - 0s 42ms/step - loss: 0.0704 - val_loss: 0.0621
    9/9 -
    Epoch 14/30
                             - 0s 42ms/step - loss: 0.0700 - val loss: 0.0621
    9/9 -
    Epoch 15/30
                             - 0s 42ms/step - loss: 0.0712 - val_loss: 0.0621
    9/9 -
    Epoch 16/30
    9/9 -
                            - 0s 42ms/step - loss: 0.0690 - val_loss: 0.0621
    Epoch 17/30
    9/9 -
                            - 0s 42ms/step - loss: 0.0702 - val_loss: 0.0621
    Enoch 18/30
                              0s 42ms/step - loss: 0.0701 - val_loss: 0.0621
    Epoch 19/30
                              0s 43ms/step - loss: 0.0693 - val_loss: 0.0621
    Epoch 20/30
                             - 0s 42ms/step - loss: 0.0700 - val loss: 0.0621
    9/9 -
    Epoch 21/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0714 - val_loss: 0.0621
    Epoch 22/30
    9/9 -
                             - 0s 44ms/step - loss: 0.0698 - val_loss: 0.0621
    Epoch 23/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0696 - val_loss: 0.0621
    Epoch 24/30
                            - 0s 42ms/step - loss: 0.0700 - val_loss: 0.0621
    Epoch 25/30
    9/9
                             • 0s 42ms/step - loss: 0.0703 - val_loss: 0.0620
    Epoch 26/30
                              0s 42ms/step - loss: 0.0685 - val loss: 0.0554
    9/9 -
    Epoch 27/30
                              0s 42ms/step - loss: 0.0867 - val_loss: 0.0621
    9/9 -
    Epoch 28/30
    9/9 -
                             - 0s 42ms/step - loss: 0.0698 - val loss: 0.0621
    Epoch 29/30
                            - 0s 42ms/step - loss: 0.0698 - val_loss: 0.0621
```

```
1 # Plot training & validation loss
 2 plt.plot(history.history['loss'], label='Training Loss')
 3 plt.plot(history.history['val_loss'], label='Validation Loss')
 4 plt.xlabel("Epochs")
5 plt.ylabel("Loss (SSIM)")
 6 plt.legend()
7 plt.title("Training vs. Validation Loss")
₹
                                  Training vs. Validation Loss
         0.13
                                                                     Training Loss
                                                                     Validation Loss
         0.12
         0.11
      0.10
0.09
         0.08
         0.07
         0.06
                 ò
                            5
                                       10
                                                              20
                                                   15
                                                                         25
                                                                                    30
                                               Epochs
```

Evaluate Model Performance

```
1 # Predict
 2 test_noisy = load_images(test_path)
 3 denoised_images = autoencoder.predict(test_noisy)
<del>_</del>→ 3/3 —
                                 -- 4s 494ms/step
 1 # Display results
 2 def display_denoising_results(noisy, denoised, num_samples=3):
         fig, axes = plt.subplots(num_samples, 2, figsize=(10, 10))
         for i in range(num_samples):
    axes[i, 0].imshow(noisy[i].reshape(IMG_HEIGHT, IMG_WIDTH), cmap='gray')
    axes[i, 0].set_title("Noisy Image")
 5
 6
              axes[i, 0].axis("off")
             axes[i, 1].imshow(denoised[i].reshape(IMG_HEIGHT, IMG_WIDTH), cmap='gray')
axes[i, 1].set_title("Denoised Image")
axes[i, 1].axis("off")
10
11
12
13
         plt.show()
 1 # Show test results
 2 display_denoising_results(test_noisy, denoised_images)
```

Noisy Image

A new offine handaritten database for the Sparish language ish sentences, has recordly been developed: the Sparians database ish Bestriched-domain Task of Cursine Script). There were fast this corpus. First of all, most databases do not contact Span Spanish is a vireleymed major lasquage. Another pality potated in from senuation-restricted tasks. These leaks we do missingly use use of languistic leconicely begand the lessand week for the record As the Sparians database, consistentially of south-section purpopshs. The writers were acted to copy a set of senderees the line fields in the forms. Best figure shaws are of the forms were the test of the forms also contains a brief set of instructions were to it.

Denoised Image

Noisy Image

A new office hardwritten database for the Spariest knopuser side entirences, has recording been developed: the Sparience database its Restative/deficience Tasks of Consine Scripts. There were bon this corpus. Firstled all, most detabases do and condition Sparies for a micrograded in more dissipance. Another insportant rest, from acrossoft-projected testion. These testis are contravolal word use of linguistic browledge begand the leation level in the record As the Spariesian detabase consisted matrix of sin and wortener prographs. The writers were asked to copy is set of sentences in finite fields in the forms. Next figure shows use of the forms are the forms show the forms a best of sell controlled in the forms. The figure shows one of the forms are to this

Denoised Image

Noisy Image

A new offline handwritten database for guage, which contains full Spanish senter been developed: the Spartacus database is Spanish Restricted-domain Task of Cursiw were two main reasons for creating this co most databases do not contain Spanish sente Spanish is a widespread major language. A reason was to create a corpus from semanti These tasks are commonly used in practice of linguistic knowledge beyond the lexicon inition process.

Denoised Image

The first model, the performance is good. The RMSE is 0.26 which is okay. However, we still can improve the denoised images which appear blurred and less clear, which suggests that the model is:

- · Removing too much detail, affecting text sharpness.
- · Smoothing out noise, but at the cost of reducing readability.
- Not preserving fine edges, leading to loss of text structure.

SSIM Loss Function: SSIM (Structural Similarity Index) measures how similar two images are, focusing on text clarity rather than pixel-wise differences.

```
1 def ssim_loss(y_true, y_pred):
      return 1 - tf.reduce_mean(tf.image.ssim(y_true, y_pred, max_val=1.0))
 1 # Define the autoencoder architecture (same as before)
 2 def build_autoencoder(input_shape=(IMG_HEIGHT, IMG_WIDTH, 1)):
      input_img = Input(shape=input_shape)
      # Encoder
 6
      x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
      x = MaxPooling2D((2, 2), padding='same')(x)
      x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
 8
      x = MaxPooling2D((2, 2), padding='same')(x)
10
11
      # Decoder
      x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
12
      x = UpSampling2D((2, 2))(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
13
14
       x = UpSampling2D((2, 2))(x)
16
      output_img = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x) # Output layer
17
      # Compile model with SSIM loss
18
      autoencoder = Model(input_img, output_img)
19
20
      autoencoder.compile(optimizer='adam', loss=ssim_loss)
21
22
      return autoencoder
```

```
1 # Build the autoencoder
2 enhance_autoencoder = build_autoencoder()
3 enhance_autoencoder.summary()
```

→ Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 256, 256, 1)	0
conv2d_5 (Conv2D)	(None, 256, 256, 32)	320
max_pooling2d_2 (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_6 (Conv2D)	(None, 128, 128, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 64, 64, 64)	0
conv2d_7 (Conv2D)	(None, 64, 64, 64)	36,928
up_sampling2d_2 (UpSampling2D)	(None, 128, 128, 64)	0
conv2d_8 (Conv2D)	(None, 128, 128, 32)	18,464
up_sampling2d_3 (UpSampling2D)	(None, 256, 256, 32)	0
conv2d_9 (Conv2D)	(None, 256, 256, 1)	289

Total params: 74,497 (291.00 KB)

```
1 # Train the model with SSIM loss
 2 history = enhance autoencoder.fit(
       train_noisy, train_clean,
       epochs=100,
 5
       batch size=16.
       shuffle=True.
 6
       validation_split=0.1

→ Epoch 1/100
                             6s 316ms/step - loss: 0.7918 - val_loss: 0.6993
    Epoch 2/100
                              0s 46ms/step - loss: 0.7018 - val_loss: 0.5949
    9/9 -
    Epoch 3/100
                             0s 46ms/step - loss: 0.5820 - val loss: 0.5473
    9/9 -
    Epoch 4/100
    9/9 -
                             0s 46ms/step - loss: 0.5467 - val loss: 0.5300
    Epoch 5/100
    9/9 -
                            - 0s 47ms/step - loss: 0.5259 - val_loss: 0.5177
    Epoch 6/100
                              0s 46ms/step - loss: 0.5032 - val_loss: 0.4936
    Epoch 7/100
                              0s 47ms/step - loss: 0.4812 - val_loss: 0.4688
    Epoch 8/100
    9/9
                             0s 46ms/step - loss: 0.4557 - val_loss: 0.4542
    Epoch 9/100
                             0s 46ms/step - loss: 0.4413 - val_loss: 0.4501
    9/9 -
    Epoch 10/100
                             0s 46ms/step - loss: 0.4258 - val loss: 0.4228
    9/9 -
    Epoch 11/100
    9/9 -
                            - 0s 46ms/step - loss: 0.4091 - val_loss: 0.4213
    Epoch 12/100
    9/9 -
                             0s 46ms/step - loss: 0.3976 - val_loss: 0.4056
    Epoch 13/100
                              0s 46ms/step - loss: 0.3847 - val_loss: 0.4082
    Epoch 14/100
                             0s 47ms/step - loss: 0.3778 - val_loss: 0.3968
    9/9
    Epoch 15/100
                             - 0s 47ms/step - loss: 0.3721 - val_loss: 0.3770
    9/9 -
    Epoch 16/100
                            - 0s 47ms/step - loss: 0.3582 - val loss: 0.3761
    9/9 -
    Epoch 17/100
    9/9 -
                             0s 46ms/step - loss: 0.3579 - val_loss: 0.3658
    Epoch 18/100
    9/9 -
                             0s 46ms/step - loss: 0.3423 - val_loss: 0.3595
    Epoch 19/100
                              0s 46ms/step - loss: 0.3372 - val_loss: 0.3528
    Epoch 20/100
                              0s 46ms/step - loss: 0.3358 - val_loss: 0.3532
    Fnoch 21/100
                             0s 47ms/step - loss: 0.3277 - val loss: 0.3482
    9/9 -
    Epoch 22/100
                             - 0s 47ms/step - loss: 0.3197 - val_loss: 0.3346
    9/9
    Epoch 23/100
    9/9 -
                              0s 46ms/step - loss: 0.3161 - val_loss: 0.3354
    Epoch 24/100
    9/9 -
                             0s 47ms/step - loss: 0.3075 - val_loss: 0.3229
    Epoch 25/100
    9/9
                              0s 46ms/step - loss: 0.3047 - val_loss: 0.3133
    Epoch 26/100
                              0s 46ms/step - loss: 0.2897 - val_loss: 0.3119
    Epoch 27/100
                             0s 46ms/step - loss: 0.2831 - val loss: 0.2965
    9/9 -
    Epoch 28/100
    9/9 -
                            - 0s 46ms/step - loss: 0.2726 - val loss: 0.2856
    Epoch 29/100
                            - 0s 46ms/step - loss: 0.2571 - val_loss: 0.2775
    9/9 -
```

1 # Plot training & validation loss
2 plt.plot(history.history['loss'], label='Training Loss')
3 plt.plot(history.history['val_loss'], label='Validation Loss')
4 plt.xlabel("Epochs")
5 plt.ylabel("Loss (SSIM)")
6 plt.legend()

```
7 plt.title("Training vs. Validation Loss")
8 plt.show()
```

Training vs. Validation Loss 0.8 Training Loss Validation Loss 0.7 0.6 0.5 SSO 0.4 0.3 0.2 0.1 Ó 20 80 100 60 Epochs

```
1 # Predict
 2 test_noisy = load_images(test_path)
 3 enhance_denoised_images = enhance_autoencoder.predict(test_noisy)
<del>→</del> 3/3 -
                            - 1s 169ms/step
 1 # Display results
 2 def display_denoising_results(noisy, denoised, num_samples=3):
       fig, axes = plt.subplots(num_samples, 2, figsize=(10, 10))
       for i in range(num_samples):
           axes[i, 0].imshow(noisy[i].reshape(IMG_HEIGHT, IMG_WIDTH), cmap='gray')
 6
           axes[i, 0].set title("Noisy Image")
           axes[i, 0].axis("off")
10
           axes[i, 1].imshow(denoised[i].reshape(IMG_HEIGHT, IMG_WIDTH), cmap='gray')
11
           axes[i, 1].set_title("Denoised Image")
           axes[i, 1].axis("off")
12
13
       plt.show()
 1 # Show test results
 2 display_denoising_results(test_noisy, enhance_denoised_images)
```

⊋÷

Noisy Image

A wer offine handaritter database for the Spanish language ish sentences, has recently been developed: the Spartacus databasis Rostrieted-damain Task of Consire Script. There were be this corpus. First of all, mod databases do and contact Syan. Spanish is a windoppoud major danguage. Another generators from semantic-restricted tasks. These tasks up convengs, used use of languistic bearded beyond the damain level of the present database beyond the damain level of the present database were sensited under a first venture promptings the writter work asked to tropp a set of senderate the time fields in the forms. Jett figure shares must the first file fields in the forms. Jett figure shares must the forms also conclain a wirel set of institutions given to the

Noisy Image

A new offices handarriller distalone for the Sparies hanguage with sentences, has recently been diselected. The Sparies could be distributed and last of Conserv Series, I There were has this corpuse Friend, all most distalones do not contain Spani Spraish as wisdopmed in yole disquare. Another important the from semanticipatricted lesies. These this are consumity used use of linguistic household, beyond the leatons letted in the recognistic baselines and distalone consisted musty of start section propagates the writers were asked to capit use of sentences in fine fields in the prime. Next figure shows one of the form seed. These forms also contain a brief set of instructions given to this

Noisy Image

A new offline handwritten database for guage, which contains full Spanish senter been developed: the Spartacus database is Spanish Restricted-domain Task of Cursiv were two main reasons for creating this co most databases do not contain Spanish sente Spanish is a widespread major language. A reason was to create a corpus from semanti These tasks are commonly used in practice of linguistic knowledge beyond the lexicon inition process.

Denoised Image

A wer affine handeritten deinkoer for the Sparich knoppin isk mainens, hes remails from derelopelt the Speritures deithed isk fleshrideridennia Task of Ernine Seright. Then were two fives carpus. First of of, word detalance to not contribe New Sparich is a wisdegreat in upor language. Another important as from a countrie-contribed tasks. These leads are invariantly were not fragistist knowledge beyond the between best in fine from the lightly reasy. As the Sparitures delitives morbid in delity of what waters to give the first own asked to supply a 4 fast of waters to give policy in the forms. Next figure stams use of the form to the forms are stripf of a materialness spire to the

Denoised Image

A sera offine kurskeritete birdose for the Sporith kongrupe ide sentieren. Sen verwilig here denderedt the Sporitures detake ide Bedrieferd-dennier Task of Lenius Seright. There were two five serges. Pind of of weat defalaces to not enriche Spority serges in Sporiture dendered to sporiture in provincia in a visiogrand mayor incopany. Another trapartient are from securation exteriories testes. There hade are monormly used use of Supplicite homologic levels. There hade are monormly used use of Supplicite homologic levels. There hade are monormly used use of Supplicite homologic levels. There hade are monormly used use of Supplicit homologic levels. There hade are monormly used used of Supplicite the Post and the Supplicit the Supplicite Supplicit levels are the supplicit levels. The supplicit levels are the Supplicit levels.

These focus also contain a brief set of instructions given to the Denoised Image

A new offline handwritten database for guage, which contains full Spanish senter been developed: the Spartacus database Spanish Restricted-domain Task of Cursiv were two main reasons for creating this co most databases do not contain Spanish sents Spanish is a widespread major language. I reason was to create a corpus from semant These tasks are commonly used in practice of linguistic knowledge beyond the lexicon nicion process.

Results and Analysis

Please see the comparison between original, denoised with autoencoder and denoised with enhanced autoencoder model below.

- Convolutional Autoencoder performs well with SSIM loss and longer training, Enhanced Autoencoder. The denoised images show clear
 improvements in quality. It produces sharper, more readable text compared to the basic autoencoder, effectively reducing noise and
 improving contrast.
- The RMSE confirm that the enhacement model is better, RMSE 0.1123 comparing to RMSE, 0.26120 from previous model.
- The basic autoencoder struggles with fine text details, producing blurred outputs, while the enhanced model preserves more text structures and improves readability.
- Future improvements could explore GAN-based denoising or Transformer-based architectures to further enhance the model's ability to restore text clarity.

```
1 # Select random indices to visualize
 3 random_indices = np.random.choice(len(test_noisy), num_images, replace=False)
 5 # Plot comparison
 6 fig, axes = plt.subplots(num_images, 3, figsize=(10, 15))
 8 for i, idx in enumerate(random_indices):
      # Original test image
      test_img = test_noisy[idx]
10
11
      denoised_img = denoised_images[idx]
      enhanced_img = enhance_denoised_images[idx]
12
13
14
      # Plot original
      axes[i, 0].imshow(test_img, cmap='gray')
15
      axes[i, 0].set_title("Original Test Image")
16
      axes[i, 0].axis("off")
17
18
19
       # Plot autoencoder denoised image
20
      axes[i, 1].imshow(denoised_img, cmap='gray')
      axes[i, 1].set_title("Denoised (Autoencoder)")
axes[i, 1].axis("off")
21
22
23
24
      # Plot enhanced autoencoder denoised image
25
      axes[i, 2].imshow(enhanced_img, cmap='gray')
      axes[i, 2].set_title("Denoised (Enhanced Autoencoder)")
26
27
      axes[i, 2].axis("off")
28
29 plt.tight_layout()
```

Original Test Image

A new offline handwritten database for the which contains full Spanish sentences, has re the Spartacus database (which stands for Spar Task of Cursive Script). There were two main this corpus. First of all, most databases desentences, even though Spanish is a widespressimportant reason, was to create a corpus from tasks. These tasks are commonly used in pracuse of linguistic knowledge beyond the lexico process.

As the Spartacus database consisted mainly and did not contain long paragraphs, the writ a set of sentences in fixed places; dedicate the forms. Next figure shows one of the form process. These forms also contain a brief sa

Denoised (Autoencoder)

I per effices haderthop decides for fations consider tall french containers, but at the formers dedines felled events for file but at finester fampti. There were the mation corpus. From et al., northindayes a container, own though flowing is a temper important reason was no commit and his proture. These tasts are commity and his proture of limportal involving beyond the latter presson.

for the furtherner denders emplated south, and field are contain long prompage, the extra one of runnings in firms places before the forms. Best figure others one of the first process. These furth solar partitions that first in

Denoised (Enhanced Autoencoder)

A new offline handwritten database for the which cuntains full Spanish centencee, has a the Operacus database (which stands fur Spai Tash of Curaive Ocrapt). There were two main this carpus. First of all, most database de sentences, even though Spanish is a wideapre important reason was to create a corpus from tooks. These tasks are community used in process.

An the Spartacus detabase cummisted sminl; and did not contain long paragraphs, the writ a not of centences in fund places; dedicatthe forms. Next figure shows one of the form process. These forms also contain a brief or