Document Image Denoising Using Autoencoders

1. Introduction

The goal of this project is to denoise noisy document images using a Convolutional Autoencoder (CAE). Noisy document images can have significant amounts of noise, which hinders readability. Traditional filtering techniques such as Gaussian blur can smooth images but often result in loss of detail. Autoencoders, a class of neural networks designed for unsupervised learning, offer a more powerful approach by learning patterns in the data to remove noise while preserving critical details.

In this report, we will explain the process followed to build and train the autoencoder, including the data preprocessing steps, model architecture, and the results.

2. Data Description

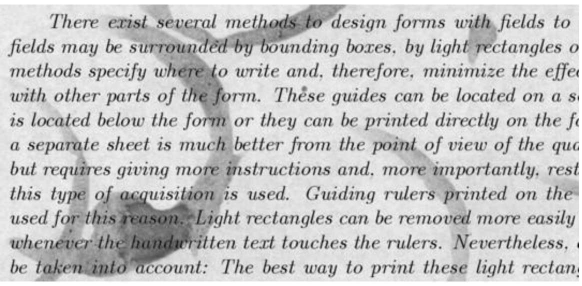
The dataset consists of noisy document images (train set) and their corresponding clean versions (train_cleaned set). There is also a test set of noisy images for which we will generate denoised versions using the trained autoencoder. The data is provided in a compressed format and must be extracted before processing.

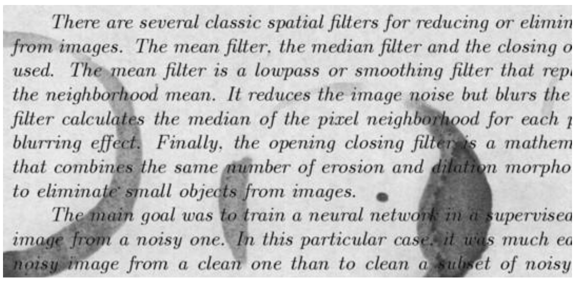
Dataset details:

- Train set: Contains noisy document images.

- Train_cleaned set: Contains clean document images (ground truth).

- Test set: Contains noisy document images for which we will generate predictions.

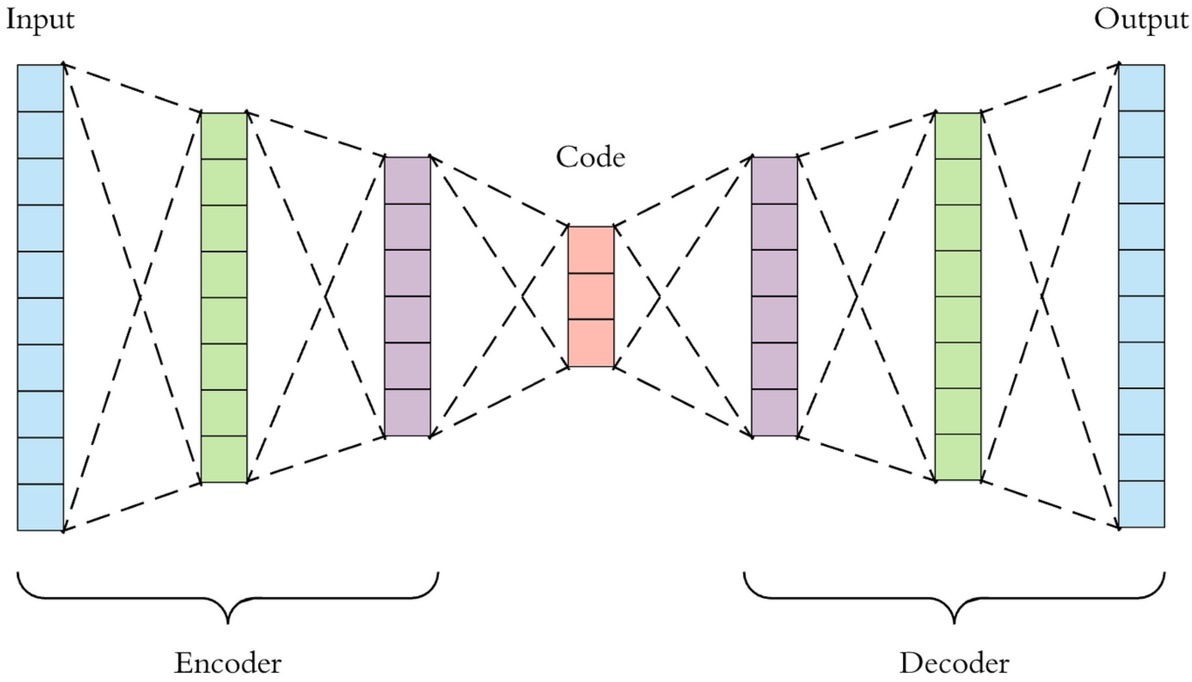

There exist several methods to design forms with fields to
fields may be surrounded by bounding boxes, by light rectangles o
methods specify where to write and, therefore, minimize the effe
with other parts of the form. These guides can be located on a s
is located below the form or they can be printed directly on the f
a separate sheet is much better from the point of view of the qu
but requires giving more instructions and, more importantly, rest
this type of acquisition is used. Guiding rulers printed on the
used for this reason. Light rectangles can be removed more easily
whenever the handwritten text touches the rulers. Nevertheless,
be taken into account: The best way to print these light rectan


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 rep
the neighborhood mean. It reduces the image noise but blurs the
filter calculates the median of the pixel neighborhood for each
blurring effect. Finally, the opening closing filter is a mathem
that combines the same number of erosion and dilation morpho
to eliminate small 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 ea
noisy image from a clean one than to clean a subset of noisy

Sample Images

3. Autoencoders: How They Work

Autoencoders are unsupervised learning models designed to compress input data into a smaller representation (latent space) and then reconstruct it back to its original form. In the context of this project, a Convolutional Autoencoder (CAE) is employed, which is especially suitable for image-based tasks due to the convolutional layers' ability to capture spatial information.



Autoencoder Architecture Components:

1. Encoder: Compresses the input into a smaller dimension (latent space) by applying convolutional operations and pooling layers to reduce spatial dimensions while capturing key features.

2. Latent Space: A bottleneck layer that forces the model to retain only the most significant features of the input.

3. Decoder: Expands the compressed representation back to the original input size using upsampling and convolutional layers, attempting to reconstruct the clean image.

Denoising with Autoencoders:

In this project, noisy images are passed through the encoder, which learns key features and eliminates noise. The decoder then reconstructs the clean image from the compressed representation, minimizing the loss between the reconstructed and ground-truth clean images.

4. Data Preprocessing

The noisy and clean images were preprocessed before training the autoencoder:

- Images were resized to a uniform dimension of 540x420 pixels.

- They were converted to grayscale (single-channel), normalized, and reshaped to fit the expected input format for the autoencoder.

The following steps were taken to load and preprocess the images:

```python
def process_image(path):
    img = cv2.imread(path)
    img = np.asarray(img, dtype="float32")
    img = cv2.resize(img, (540, 420))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = img / 255.0
    img = np.reshape(img, (420, 540, 1))
    return img
```

We also split the training set into training and validation subsets to monitor the model's performance on unseen data during training.

5. Autoencoder Model Architecture

The model is a Convolutional Autoencoder built using the Keras framework. It consists of convolutional layers for feature extraction, followed by max-pooling layers to reduce the spatial dimensions in the encoding step, and upsampling layers to reconstruct the image in the decoding step. Dropout layers are used to prevent overfitting, and Batch Normalization is applied to speed up training and ensure stability.

Key Layers:

- Encoder: Two convolutional layers with `ReLU` activations, followed by max-pooling to compress the input.
- Decoder: Two convolutional layers with `ReLU` activations, followed by upsampling layers to reconstruct the image.

```python
def model():
    input_layer = Input(shape=(420, 540, 1))
```

```
    Encoding layers
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(input_layer)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), padding='same')(x)
    x = Dropout(0.5)(x)

    Decoding layers
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = UpSampling2D((2, 2))(x)

    output_layer = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

    model = Model(inputs=[input_layer], outputs=[output_layer])
    model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])

    return model
```

6. Training the Autoencoder

The model was trained using the Mean Squared Error (MSE) loss function, which measures the reconstruction error between the noisy input and the cleaned ground truth image. The optimizer used was Adam, known for fast convergence in training deep neural networks.

Training was conducted with early stopping to avoid overfitting, halting training once the loss did not improve for a specified number of epochs.

```python
callback = EarlyStopping(monitor='loss', patience=30)
history = model.fit(X_train, Y_train, validation_data=(X_val, Y_val), epochs=600, batch_size=24, verbose=0, callbacks=[callback])
```

7. Results and Evaluation

The loss and mean absolute error (MAE) were monitored over epochs to evaluate how well the model was learning:

```python
 Plot the evolution of loss and MAE
epoch_loss = history.history['loss']
epoch_val_loss = history.history['val_loss']
epoch_mae = history.history['mae']
epoch_val_mae = history.history['val_mae']

plt.figure(figsize=(20,6))
plt.subplot(1,2,1)
plt.plot(range(0,len(epoch_loss)), epoch_loss, 'b-', linewidth=2, label='Train Loss')
plt.plot(range(0,len(epoch_val_loss)), epoch_val_loss, 'r-', linewidth=2, label='Val Loss')
plt.legend()

plt.subplot(1,2,2)
plt.plot(range(0,len(epoch_mae)), epoch_mae, 'b-', linewidth=2, label='Train MAE')
plt.plot(range(0,len(epoch_val_mae)), epoch_val_mae, 'r-', linewidth=2, label='Val MAE')
plt.legend()
plt.show()
```

The model successfully learned to reduce noise in the images. Sample results show significant improvements from the noisy input images to the denoised versions.

8. Denoising Results

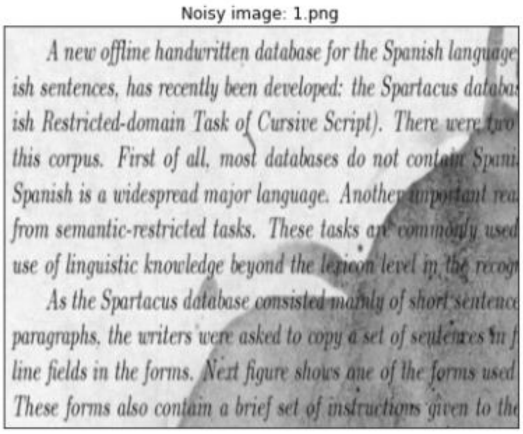
The trained autoencoder was used to predict clean versions of the noisy test images:

```python
Y_test = model.predict(X_test, batch_size=16)
```

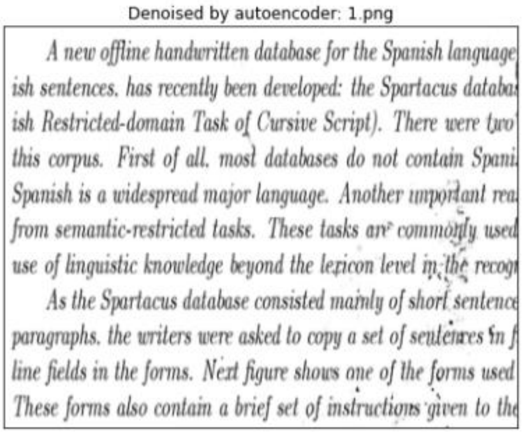
Visualizing the results shows that the autoencoder effectively removes noise from the document images, preserving important text information.

```python
plt.figure(figsize=(15,25))
for i in range(0,8,2):
    plt.subplot(4,2,i+1)
    plt.imshow(X_test[i][:,:,0], cmap='gray')
    plt.title('Noisy image')

    plt.subplot(4,2,i+2)
    plt.imshow(Y_test[i][:,:,0], cmap='gray')
    plt.title('Denoised by autoencoder')
plt.show()
```



A new offline handwritten database for the Spanish language
ish sentences, has recently been developed: the Spartacus databa
ish Restricted-domain Task of Cursive Script). There were two
this corpus. First of all, most databases do not contain Spani
Spanish is a widespread major language. Another important rea
from semantic-restricted tasks. These tasks are commonly used
use of linguistic knowledge beyond the lexicon level in the recog
As the Spartacus database consisted mainly of short sentence
paragraphs, the writers were asked to copy a set of sentences in f
line fields in the forms. Next figure shows one of the forms used
These forms also contain a brief set of instructions given to the



A new offline handwritten database for the Spanish language
ish sentences, has recently been developed: the Spartacus databa
ish Restricted-domain Task of Cursive Script). There were two
this corpus. First of all, most databases do not contain Spani
Spanish is a widespread major language. Another important rea
from semantic-restricted tasks. These tasks are commonly used
use of linguistic knowledge beyond the lexicon level in the recog
As the Spartacus database consisted mainly of short sentence
paragraphs, the writers were asked to copy a set of sentences in f
line fields in the forms. Next figure shows one of the forms used
These forms also contain a brief set of instructions given to the

9. Conclusion

This project demonstrates the power of Convolutional Autoencoders in denoising document images. The autoencoder learned to reconstruct clean images from noisy ones by capturing essential features in the latent space. This approach is highly effective in removing noise while preserving fine details in the documents. With further refinement, the model could be applied to real-world denoising tasks, such as restoring scanned documents or improving OCR accuracy.