Contents

1 Problem Statement

Nowadays, most newly created documents are stored electronically, and various digital countermeasures are in place to protect them from deterioration or corruption. These measures include error detection schemes and cloud storage, which ensure that electronic documents remain identical in form and easily accessible for years to come. However, this is not the case for physical documents, such as historical texts, research papers, and other hard copies that predate the digital age. These paper documents are vulnerable to corruption or damage due to exposure to the elements. To preserve their information, they must be converted into digital documents before their physical forms deteriorate. However, simply scanning and converting physical documents to digital copies is insufficient because noise that exists on physical documents due to wear-and-tear (folds, stains, smudges, shadows) will be carried over to any digital recreations.

To address this issue, several computer vision and machine learning techniques have been developed for document restoration and noise removal. The goal is to denoise dirty documents and create higher-fidelity digital recreations of the original documents. This report explores several such techniques using a dataset of noisy documents from the UC Irvine NoisyOffice Data Set.

The report evaluates, discusses, and compares the results of several denoising methods applied to a test dataset. The techniques explored in this report are Image Thresholding, High Pass Filtering, Autoencoding, and Linear Regression. The report explains each of these techniques and their specific variations used in this study in the following section. By exploring and comparing these methods, the report aims to identify the most effective techniques for denoising physical documents and creating high-quality digital recreations.

2 Datasets

We have two sets of images, train and test. These images contain various styles of text, to which synthetic noise has been added to simulate real-world, messy artifacts. The training set includes the test without the noise (train cleaned). You must create an algorithm to clean the images in the test set.

- UC Irvine Noisy Office Dataset.
- 200 train and 200 test images.

Example of Training Data set

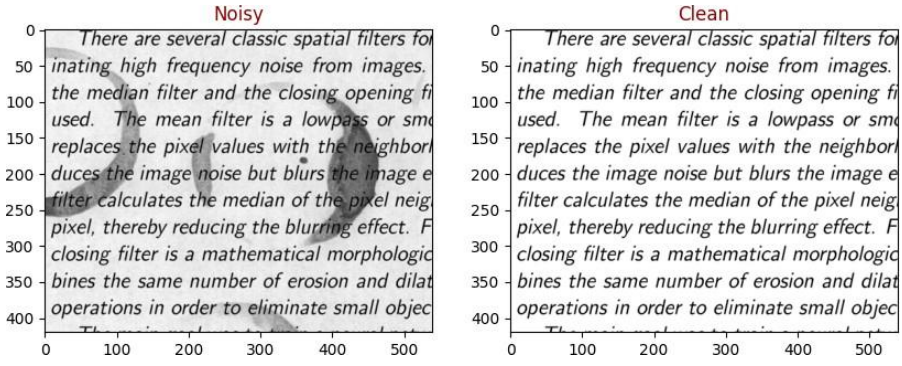


fig 1: **Noisy Image and it's corresponding clean image**

3 Methods :- Denoising Techniques

Several techniques were applied to document denoising and their results were evaluated in this report. The following subsections explain each in detail.

3.1 Image Thresholding

Image thresholding is a type of image segmentation that divides the foreground from the background in an image. In this technique, the pixel values are assigned corresponding to the provided threshold values. In computer vision, thresholding is done in grayscale images.

In **Simple Thresholding**, the matter is straight-forward. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value.

Adaptive Thresholding comes handy if an image has different lighting conditions in different areas. In that case, adaptive thresholding can help. Here, the algorithm determines the threshold for a pixel based on a small region around it. So we get different thresholds for different regions of the same image which gives better results for images with varying illumination.

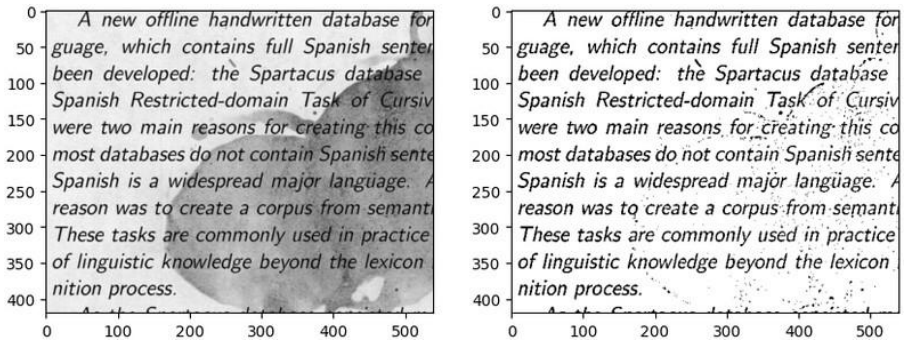


fig 2: **Filtered image by Image Thresholding**

Image thresholding is mainly used for image segmentation and does have some limitations. Choosing an appropriate threshold value can be a challenging task. Slight variations in the threshold can lead to significant changes in the resulting binary image. Thresholding relies solely on pixel intensity values and for it to denoise documents, the text needs to be darker than noise. It does not consider other essential image attributes like texture, shape, or spatial relationships. If an image pixel intensity is above a threshold value then we set the image pixel to 1 or 0. So this technique sometimes can not perform denoising at all.

3.2 High Pass Filtering

A high pass filter is the basis for most sharpening methods. An image is sharpened when contrast is enhanced between adjoining areas with little variation in brightness or darkness. A high pass filter tends to retain the high-frequency information within an image while reducing the low-frequency information.

The input image is transformed into the frequency domain using the 2D Fast Fourier Transform(FFT) operation. This converts the image from the spatial domain to the frequency domain.

Then we iterate over the rows and columns of the frequency domain representation respectively and then check whether the distance of the frequency component from the centre is below a specified threshold(0.03 in this case). If the distance is below the threshold, it is considered a low-frequency component and set to zero. This step effectively removes the low-frequency components from the image.

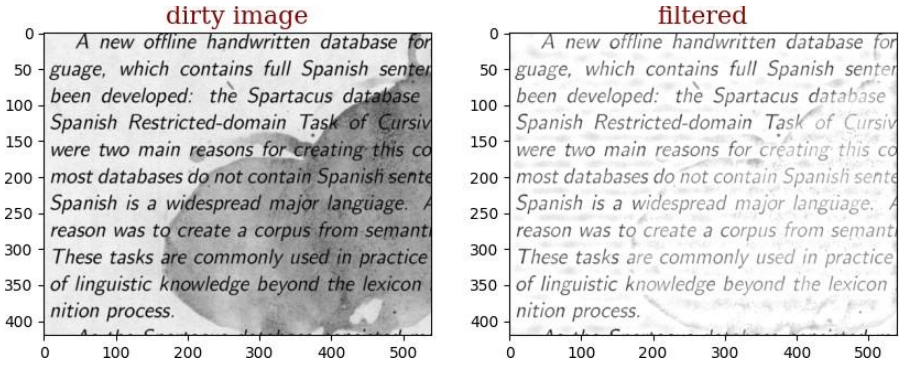


fig 3: **Filtered image by High Pass Filtering**

After applying the high-pass filter in the frequency domain, the filtered image is obtained by taking the inverse Fourier Transform(IFFT) of the modified frequency domain representation.

In summary, the high pass filter model applies a high-pass filter to an input image by manipulating its frequency domain representation. Frequencies below a specified threshold are set to zero, effectively removing low-frequency components from the image. The result is a filtered image that emphasizes high frequency components which is given in the figure below.

3.3 Linear Regression

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between a dependent variable and one or more independent features.

When the number of the independent feature, is 1 then it is known as **Univariate Linear regression**

$$y_i = wx + b$$

And in the case of more than one feature, it is known as **multivariate linear regression**.

$$y_i = \sum_{j=1}^{n} w_j x_{ij} + b$$

The goal of the algorithm is to find the best linear equation that can predict the value of the dependent

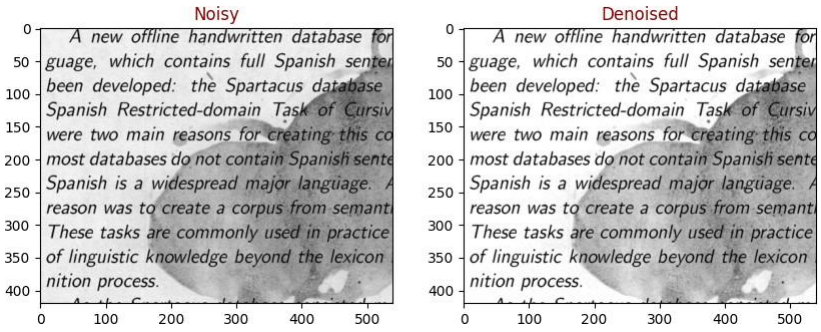


fig 4: **Predicted image by Linear Regression model**

variable based on the independent variables. The equation provides a straight line that represents the

relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).

We used linear regression to model the relationship between dependent variables and independent variables. In our case, the dependent variables are the pixel values of the clean image and the independent variables are the pixel values of the noisy image. Initially, we used only the pixel values as a feature for our Linear Regression model and we trained it and we got the following results.

As we can see in figure 4 the image produced by the linear regression model. It removed minor noises but failed miserably in removing noises like coffee stains. For this model, We trained using one noisy image and one corresponding clean image and a pixel value was used as a feature to train the model.

To improve the result, we shifted to Multiple Linear Regression model And in this, we used a total of 200 features to train the Multiple Linear Regression Model. We got these features by creating a window of 20 by 20 around the current pixel then we took that window and flattened it and used it as features. we tried different window sizes and we started with 2 by 2, then 4 by 4 then we reached 20 by 20. In each progression, our results were getting better. A window size of 20 by 20 is giving the desired result when it comes to removing coffee stains.

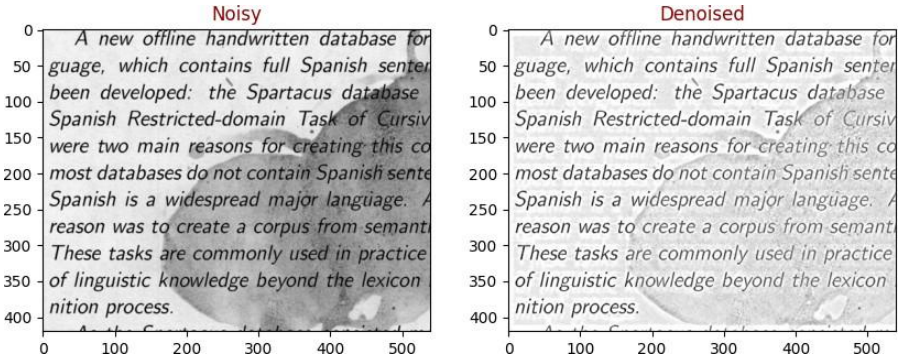


fig 5: **Predicted image by Multiple Linear Regression model**

As we can see in figure 5, the Multiple Regression Model significantly reduced the intensity of coffee stains. In order to remove clean it further we trained a separate Multiple Regression model where we used a different pair of a noisy image and a clean image to train the model. The nosy image selected for the training has noises of a different pattern where the intensity of the noise and the text are not close to each other.

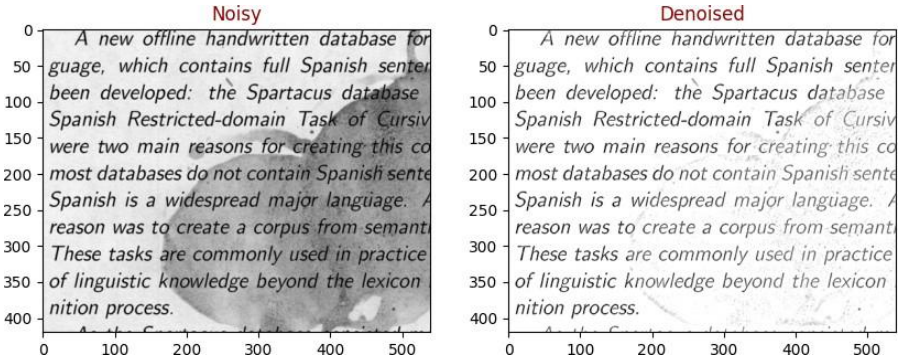


fig 6: **Predicted image by using two separate Multiple Linear Regression model**

For the figure 6 we have used two models of Regression one to explicitly remove coffee stains and

another one to remove minor noises spread throughout the image. And different images were used to train each of these models. As it is using 200 features to generate de-noised images it does take some time for computation if the test image is large.

3.4 Auto Encoder

Autoencoders are neural networks composed of an encoder and a decoder. The encoder compresses the input data into a lower-dimensional representation. The decoder reconstructs the representation to obtain an output that mimics the input as closely as possible. In doing so, the autoencoder learns the most salient features of the input data. Autoencoders are closely related to principal component analysis (PCA). If the activation function used within the autoencoder is linear within each layer, the latent variables present at the bottleneck (the smallest layer in the network, aka. code) directly correspond to the principal components from PCA.

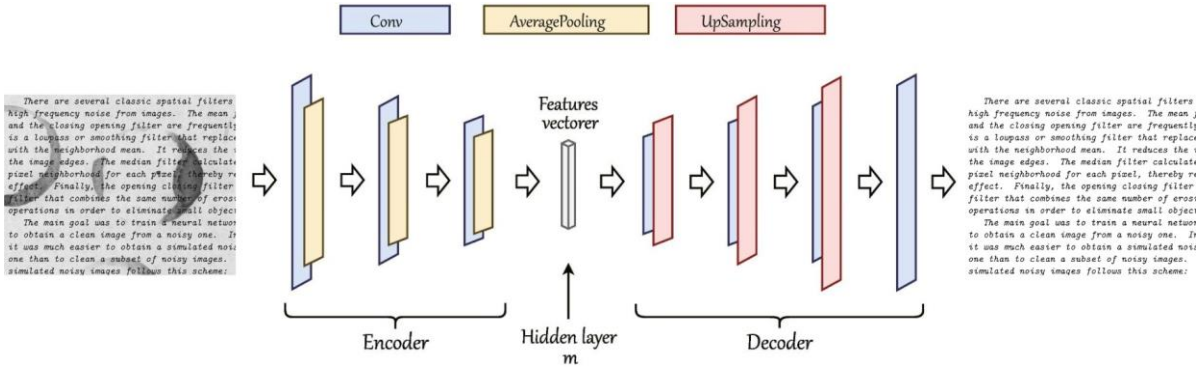


fig 7: **Auto Encoder architecture**

The network is composed of 5 convolutional layers to extract meaningful features from images. To execute a convolution, a convolutional kernel slides over the input. At each location, matrix multiplication is performed between the kernel and the overlapping region of the input, to produce the feature map passed to the next layer. The values of the kernel matrix are learned during training, using backpropagation with gradient descent. These layers are well-suited to image inputs as they successfully capture spatial dependencies.

In the first four convolutions, we use 64 kernels. Each kernel has different weights, performs different convolutions on the input layer, and produces a different feature map. Each output of the convolution, therefore, is composed of 64 channels. Thus, in the first convolution, each kernel has dimension 3x3x1, while in the next ones, the kernels are of dimension 3x3x64 in order to convolve every channel. The last convolution uses a single 3x3x64 kernel to give the singlechannel output. During convolutions, we use the same padding. We pad with zeros around the input matrix, to preserve the same image dimensions after convolution. To contain non-linearity in our model, the result of the convolution is passed through the Leaky ReLU activation function. The encoder uses max-pooling for compression. A sliding filter runs over the input image, to construct a smaller image where each pixel is the max of a region represented by the filter in the original image. The decoder uses up-sampling to restore the image to its original dimensions, by simply repeating the rows and columns of the layer input before feeding it to a convolutional layer.

Batch normalization layers are included to improve the speed, performance, and stability of the model. We normalize values from the previous layer by subtracting the batch mean and dividing by the batch stan-

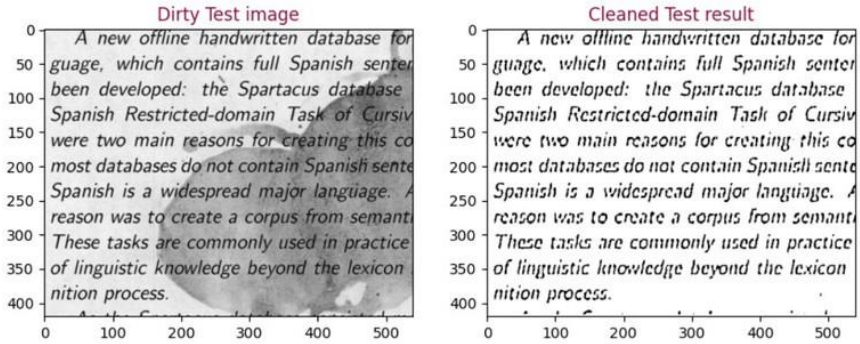


fig 8: **Predicted image by Auto Encoder**

dard deviation. Batch normalization reduces covariance shift, that is the difference in the distribution of the activations between layers and allows each layer of the model to learn more independently of other layers.

3.5 GANs - Generative Adversarial Networks

A generative adversarial network (GAN) is a machine learning (ML) model in which two neural networks compete with each other by using deep learning methods to become more accurate in their predictions. GANs typically run unsupervised and use a cooperative zero-sum game framework to learn, where one person's gain equals another person's loss.

The two neural networks that make up a GAN are referred to as the generator and the discriminator. The generator is a convolutional neural network and the discriminator is a deconvolutional neural network.

- The goal of the generator is to artificially manufacture outputs that could easily be mistaken for real data.
- The goal of the discriminator is to identify which of the outputs it receives have been artificially created.

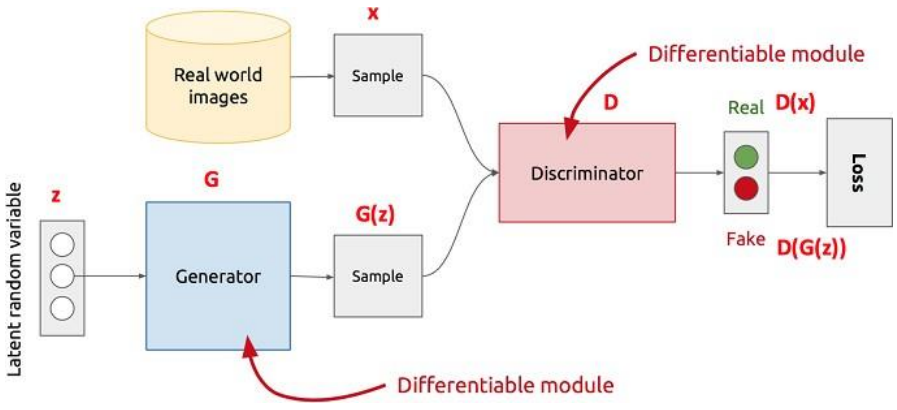


fig 9: **GAN architecture**

While GANs are capable of generating highly realistic and high-quality samples, the model can become very complex and training GANs can be a computationally intensive and time-consuming task. In our case

the input image size is big. So the generator, which takes random noise equal to the size of the image becomes too big to be handled by GPU. So, some convolutional layers are added to the generator to decrease the size of the input image. But that makes the generator too complex and training becomes very time-consuming. We tried to train the model using less number of epochs but this resulted in black images or no meaningful images. **We ran the model with 50 epochs using a batch size of 8 but the resultant image was just random noise.We were unable to run the model for more epochs due to insufficient computational resources.**

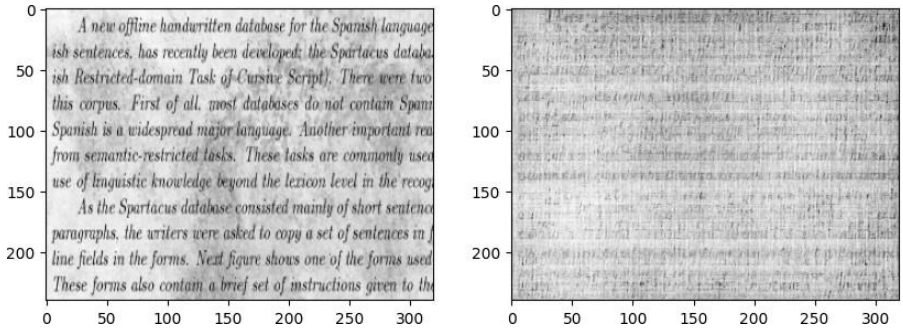


fig 10: **Predicted image by GAN**

3.6 Quantitative Analysis

3.6.1 Root Mean Square Error - RMSE

RMSE quantifies error using the following formula to compare an output denoised image K with a ground truth the clean image I.

Here, m and n represent the dimensions of the denoised and ground truth images.

$$RMSE = \frac{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j)-K(i,j)]^2}}{m*n}$$

3.6.2 Peak Signal-to-Noise Ratio - PSNR

PSNR quantifies error using the following two formulas. First the mean squared error (MSE) is determined. Then MSE and MAX, the maximum value of any pixel, can be used to determine PSNR.

Here, I represents the ground truth image and K represented the image being evaluated. m and n represent the dimensions of both images.

$$PSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j)-K(i,j)]^2}{m*n}$$

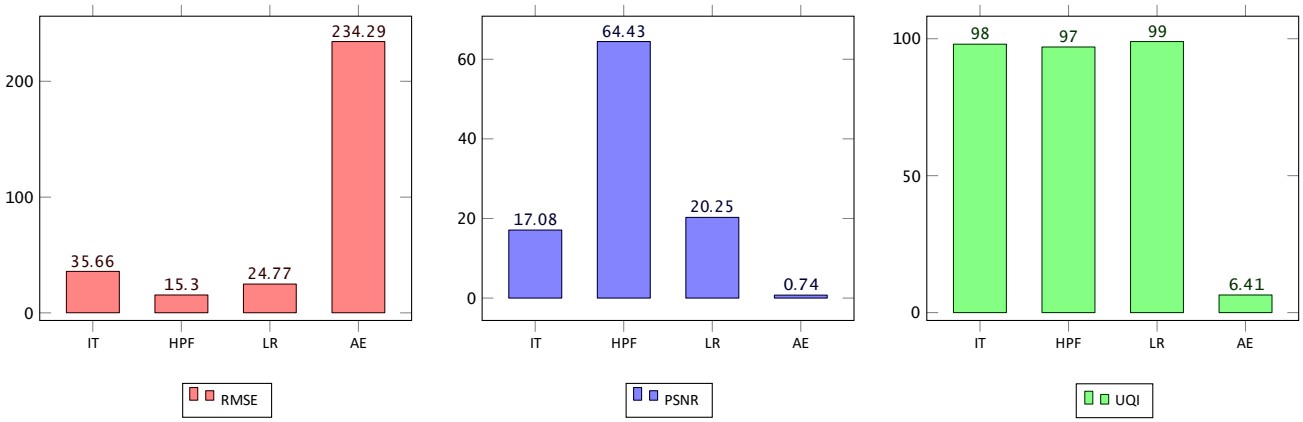
$$PSNR = 10 * \log_2\left(\frac{MAX}{MSE}\right)$$

3.6.3 Universal Quality Index - UQI

UQI is an objective image quality index first proposed as a method of quantifying image distortion by Z. Wang and A.C. Bovik in 2002. It factors loss of correlation, luminance distortion, and contrast distortion in calculating its metric. It avoids the error summation methods used in both RMSE and PSNR evaluations.

4 Results

Denoising Techniques	Quantitative Metrics		
	RMSE	PSNR	UQI
Image Thresholding	35.66	17.08	98
High Pass Filtering	15.30	64.43	97
Linear Regression	24.77	20.25	99
Auto Encoder	234.29	0.735	6.409



1. **IT:** Image Thresholding 2. **HPF:** High Pass Filtering 3. **LR:** Linear Regression 4. **Ae:** Auto Encoder

The process of denoising documents involves the use of different techniques, each with its own strengths and weaknesses. No single technique is perfect for denoising documents, and each method had a different profile among the three metrics.

These profiles are distinguishable by a person, and some techniques excel in certain areas of denoising while leaving behind tell-tale artifacts of their specific denoising strategy.

For example, **High Pass filtering** is a useful technique that is simple to implement and capable of mostly denoising documents. However, it lacks robustness in resolving the original features whenever stains overlap with document elements.

Image thresholding seems to be the best computer vision method, with good metrics and high UQI, producing legible documents with small specks as artifacts. Linear regression is an all-around strong denoising method with legible outputs, low error, and high UQI, but it is limited in capturing locality information for dark text on a light background and for folded documents.

Linear Regression also works pretty fine when it comes to removing minor noises such as folds, grainy noises etc. But struggles a little bit with coffee stains. It removes a majority of the coffee stains but in the process, it also makes the affected area lighter than the rest of the text. It has a low error to high UQI ratio when compared to other methods.

Auto Encoder is complex and has many nuances. It is the best at removing stains, but it also ”over-cleans” the document and ”whitens” the text in the process. One reason for the text whitening could be that the training dataset was small, consisting of only a few noisy and clean images. To generate more images, synthetic noise and linear transformations were added, but these newly generated images were not accurate enough for their corresponding cleaned versions, resulting in discrepancies in the output of the autoencoder. The ”whitening” of the images could also be due to the number of epochs that the autoencoder was trained to compute in the resultant images. If a large, quality dataset is provided as input for the autoencoder, there is a possibility of a significant improvement in the corresponding denoised versions of noisy images.

Overall, the selection of a denoising technique is dependent on the specific application and the type of

document being processed. It is important to consider the strengths and limitations of each technique, as well as the artifacts that may be introduced during the denoising process. By carefully choosing the appropriate denoising method for the given application, it is possible to achieve high-quality, legible documents while minimizing the impact of any introduced artifacts.

5 Conclusion

Two image processing techniques and two machine learning techniques used are Image Thresholding and High Pass Filtering and Linear Regression and Auto Encoder respectively. Each gave acceptable results but as they work in different ways to produce images each method has its fair share of faults. How well the image will be denoised depends on the way noise is present in the image.

Therefore depending on the noise present different methods give different results and out of the given results, the user may choose the most cleaned one.

The autoencoder technique appears to be the most effective in removing stains or noise, leaving no background stains in the resultant image. Nevertheless, it also has the drawback of ”whitening” the text.

The performance of the autoencoder technique is determined by the number of epochs the neural net is trained on, making hyperparameter tuning a crucial part of generating the best output from the dataset. Due to insufficient data for the CNN, we augmented our dataset with synthetic noise and linear transformations. If sufficient ”good” data and a certain number of GPUs are available, the autoencoder is expected to yield the best results.