Denoising Dirty Documents: Document Restoration with Computer Vision and Machine Learning

Kartikeya Shukla
ks5173@nyu.edu

Chinmay Wyawahare
cnw282@nyu.edu

Michael Lally
mfl340@nyu.edu

***Abstract*—Numerous scientific papers, historical documentaries/artifacts, recipes, books are stored as papers be it handwritten/typewritten. With time, the paper/notes tend to accumulate noise/dirt through fingerprints, weakening of paper fibers, dirt, coffee/tea stains, abrasions, wrinkling, etc. There are several surface cleaning methods used for both preserving and cleaning, but they have certain limits, the major one being: that the original document might get altered during the process. The purpose of this project is to do a comparative study of traditional computer vision techniques vs deep learning networks when denoising dirty documents.**

I. Introduction

Most modern, newly created documents exist primarily as electronic files, protected from deterioration or corruption by a vast array of digital countermeasures. Error detection schemes and the proliferation of cloud storage ensure that documents created today will remain identical in form and easy to access for years to come. However, the same cannot be said for physical documents.

An extreme quantity of documents - historical texts, research papers, etc. - that predate the digital age exist only as hard copies. These paper documents are susceptible to corruption through mere existence and exposure to the elements. To preserve their information, they must be converted to digital documents before their physical forms deteriorate. Simply scanning and converting physical documents to digital copies in insufficient; noise that exists on physical documents as a result of wear-and tear (folds, stains, smudges, shadows) will be carried over to any digital recreations.

This report explores several computer vision and machine learning techniques for document restoration and noise removal as a solution to this problem. The data set of noisy documents if from the UC Irvine NoisyOffice Data Set [3]. Denoising dirty documents enables the creation of higher-fidelity digital recreations of original documents. Several methods for denoising documents are applied to a test dataset and their results are evaluated, discussed, and compared. The techniques explored in this report are **Median Filtering**, **Edge Detection, Dilation & Erosion**, **Adaptive Filtering**, **Autoencoding**, and **Linear Regression**. The techniques, and the variations on each specific to this report, are explained in the following section.

Also covered in this report is the integration of these methods into a web application. The purpose of this application is to allow users to upload noisy documents and qualitatively evaluate the denoised images resulting from the different techniques.

II. Denoising Techniques

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

A. Median Filtering

Median filtering is the simplest denoising technique and it follows two basic steps: first obtain the ”background” of an image using Median Filtering with a kernel size of 23 x 23, then subtract the background from the image. Only the ”foreground” will remain, clear of any noise that existed in the background. In this context, ”foreground” is the text or significant details of the document and ”background” is the noise, the white space between document elements.

B. Edge Detection, Dilation & Erosion (EDE Method)

Edge detection methods identify the points where the image brightness changes sharply, to organize them into edges. Canny edge detection is particularly helpful to extract edges. In the example below, we have already gotten rid of some of the coffee stains.

Before using the techniques, the images were processed to be cleaned away from the edges of noise. First apply dilation, which makes lines thicker by adding pixels to boundaries. Notice this results in “filling in” the text, while edges surrounding stains remain hollow. Then, by applying the reverse operation, erosion, one can completely remove thin lines while preserving the thicker lines.

C. Adaptive Thresholding

Another characteristic of the dirty images is that the text tends to be darker than the noise. Within dark noises, the text inside is even darker. Thus the objective is to preserve pixels that are the darkest locally.

Thresholding sets all pixels whose intensity is above a threshold to 1 (background), and the remaining pixels to 0

(foreground). During adaptive thresholding, there is no single global threshold: the threshold value is computed for each pixel. To determine the threshold, we use Gaussian Thresholding. The threshold value is the weighted sum of neighboring pixel intensities, where the weights are a gaussian window.

D. Linear Regression

Instead of modelling the entire image at once, we tried predicting the cleaned-up intensity for each pixel within the image, and constructed a cleaned image by combining together a set of predicted pixel intensities using linear regression [2]. After creating a vector of y values, and a matrix of x values, the simplest data set is where the x values are just the pixel intensities of the dirty image.

Except at the extremes, there is a linear relationship between the brightness of the dirty images and the cleaned images. There is a broad spread of x values as y approaches 1, and these pixels probably represent stains that need to be removed. The linear model has done a brightness and contrast correction. That's quite good performance for a simple least squares linear regression!

E. Autoencoding

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.

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 back propagation 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 single-channel 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 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 standard 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.

III. Quantitative Evaluations

Three metrics were used to quantitatively evaluate the outputs of the denoising techniques: Root Mean Square Error (**RMSE**), Peak Signal-to-Noise Ratio (**PSNR**), and Universal Quality Index (**UQI**). These evaulations, as they relate to this report, are explained in the following subsections.

A. Root Mean Square Error - RMSE

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

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

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

B. 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.

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

$$PSNR = 10 \times log_{10}\left(\frac{MAX}{MSE}\right)$$

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

C. 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 [1]. 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.

IV. Individual Results

The results of each method on the example ”Dirty Image” are compared against the ground truth ”Clean Image”. The following technique pipelines follow.

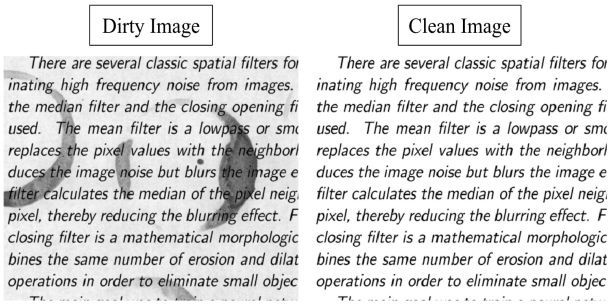

Fig. 1. Dirty Image used as input and Clean Image used as ground truth

A. Median Filtering

Median Filter considers the text we want to preserve was as “foreground” and rest as “background” of our image. In this example, the rings from stains of a coffee cup are still partly visible. Most of the noise from the dirty image has been removed, but artifacts still remain.

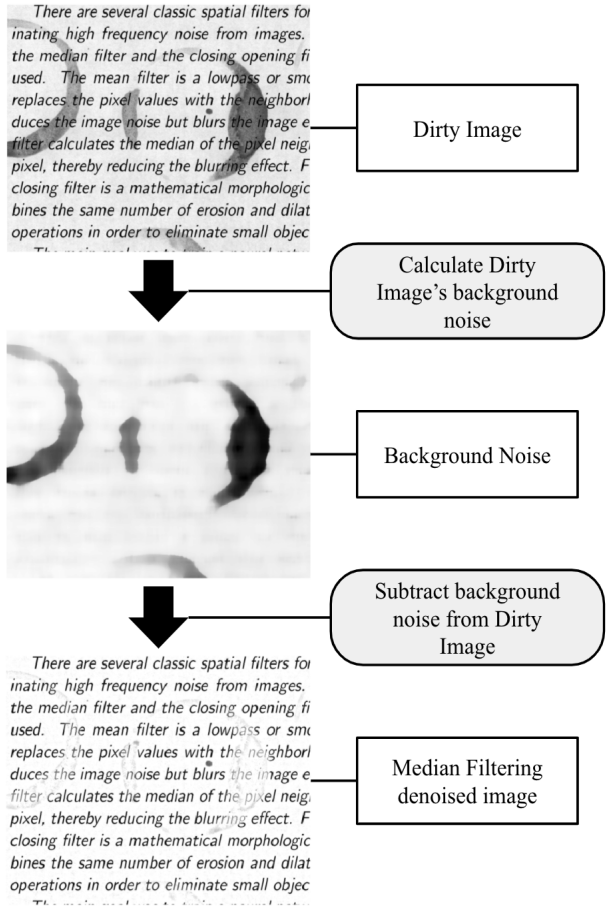

Fig. 2. Median Filtering pipeline for denoising a Dirty Image

B. Edge Detection, Dilation & Erosion (EDE Method)

While most of the noise has been removed, artifacts from dilation and erosion are clearly visible and make the text difficult to read. Narrow gaps between different letters have been filled in, joining some features together. Characters are much thinner than they should be While this technique does denoise, it also alters the resulting document.

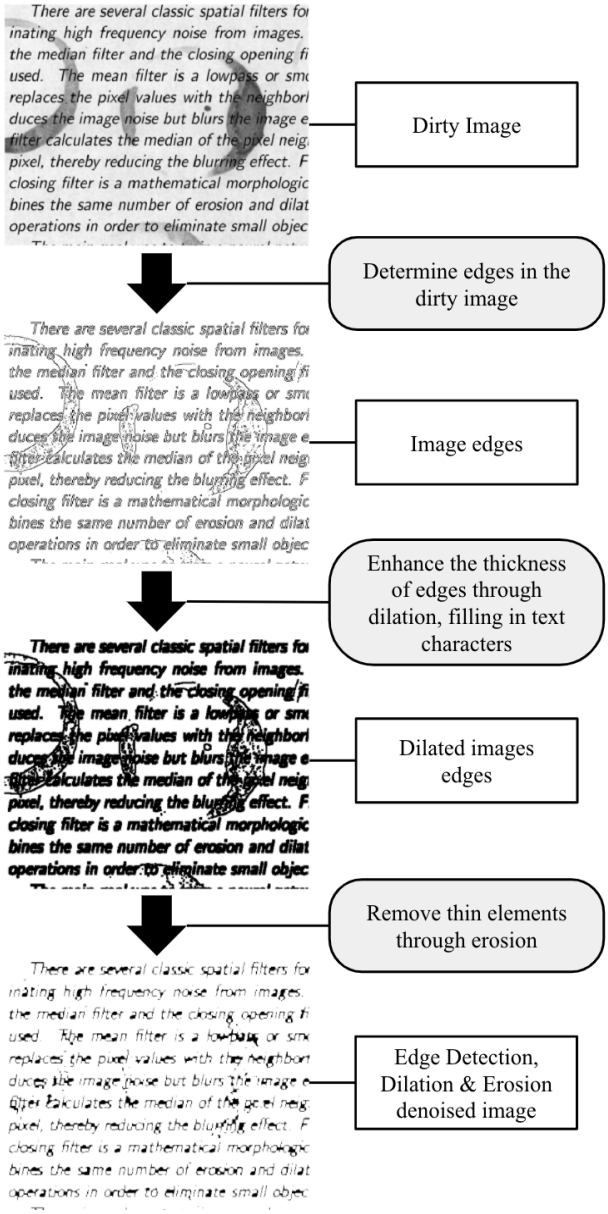

Fig. 3. Edge Detection, Dilation & Erosion pipeline for denoising a Dirty Image

C. Adaptive Thresholding

This method is both simple and very effective. Nearly all of the feature text is recovered and very few artifacts remain in the cleaned image. Adaptive Thresholding assumes that background noise will never be dark enough to avoid being

culled, but this is not always true. Some small, black artifacts are visible in the output. This is usually only in regions where there was lots of dark noise without any nearby darker text. It is easily distinguishable as an artifact, not impacting the legibility of the text.

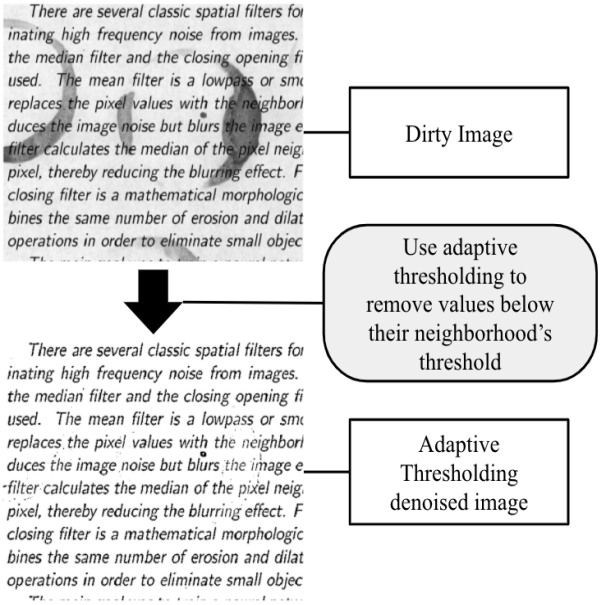

Fig. 4. Adaptive Thresholding pipeline for denoising a Dirty Image

D. Linear Regression

Linear regression is extremely effective at predicatively denoising the documents and leaving virtually no artifacts that indicate that a computer vision or machine learning processing pass has occurred.

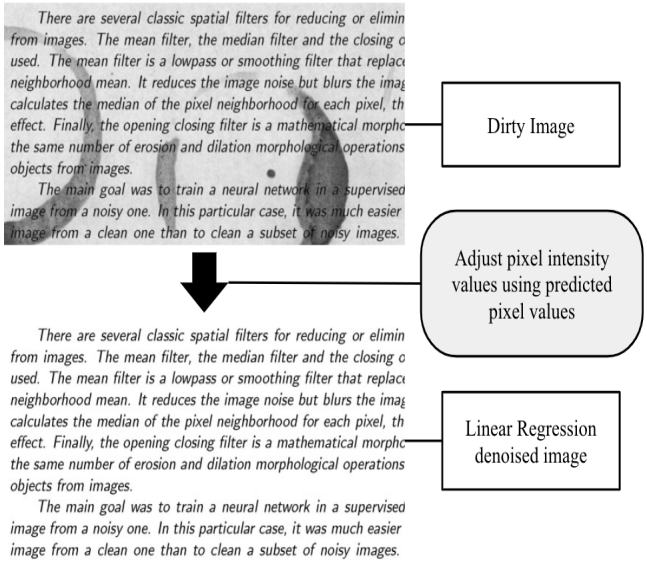

Fig. 5. Linear Regression pipeline for denoising a Dirty Image

E. Autoencoder

The autoencoder method removes background noise very effectively; text is largely restored as well. However, it can be seen that artifacts of compression exist on the text itself. While the results of compressing and up-sampling are visible in the output, the method is still generally successful at denoising.

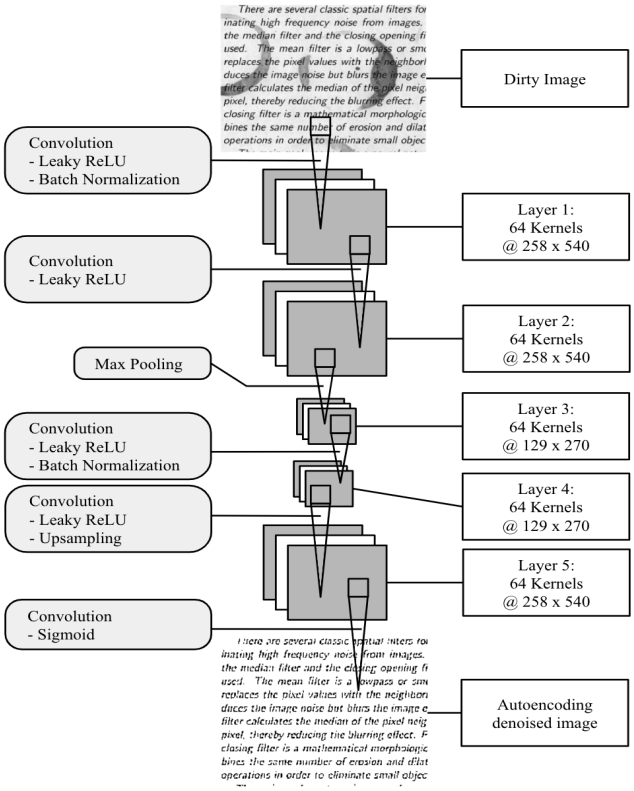

Fig. 6. Convolutional Neural Net (CNN) used to denoise with autoencoder

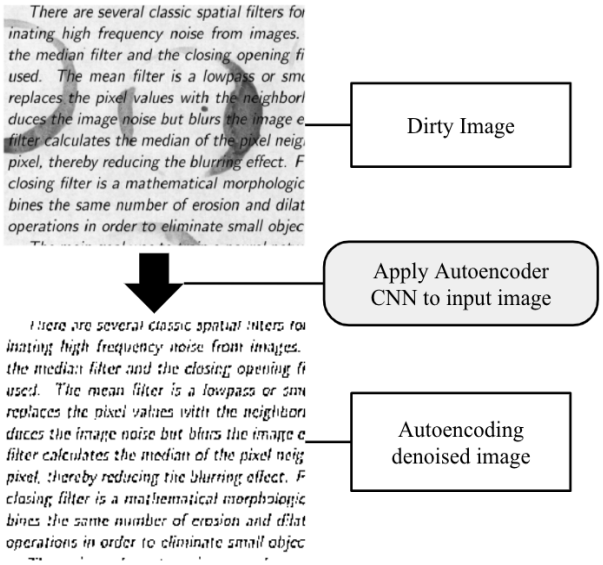

Fig. 7. Autoencoder pipeline for denoising a Dirty Image

V. Results Compared

TABLE I
DENOISING METHODS AND THEIR SCORING METRICS FOR TEST IMAGE

Denoising Techniques	Quantitative Metrics		
	RMSE	*PSNR*	*UQI*
Median Filtering	21	21	99
Edge Deteciton, Dilation & Erosion	63	12	94
Adaptive Thresholding	30	18	98
Linear Regression	34	21	87
Autoencoder	234	*	6

*Library difficulties prevented PSNR calculation for Autoencoder method.

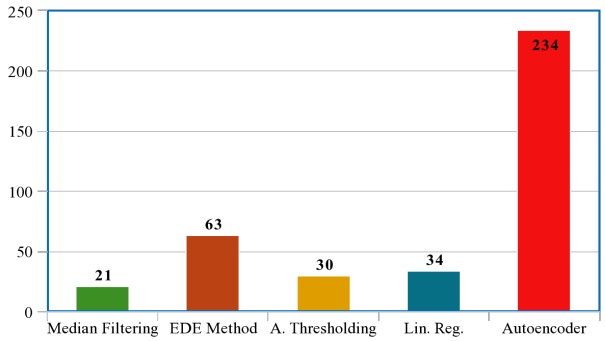

Fig. 8. RMSE of different denoising methods

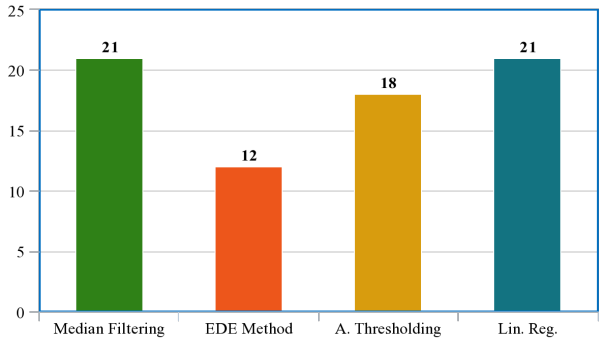

Fig. 9. PSNR of different denoising methods

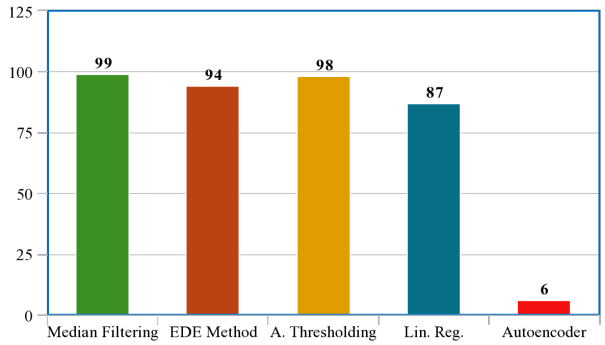

Fig. 10. UQI of different denoising methods

No single technique was perfect at denoising documents. Each denoising method had a different profile among the three metrics; similarly a person can distinguish the types of outputs between techniques. Some excelled in certain areas of denoising, while also leaving behind tell-tale artifacts of its given denoising strategy.

Median filtering is a useful technique; simple to implement, with consistent metrics, its capable of mostly denoising documents. However it lacks robustness; whenever stains overlap with document features, its unable to resolve the original features. EDE Method, though largely effective at removing stains, generates text outputs that can be illegible. Similarly, if stains have any texture to them, they may be restored along with the document elements. Adaptive thresholding seems to be the best base computer vision method; it has good metrics, a high UQI, and qualitatively produces legible documents. Artifacts produced from adaptive thresholding are small specks that passed through the method's thresholding calculation and don't affect the legibility of the document. It it very good at distinguishing text from within dark stains.

Linear regression also proved to be an all-around strong denoising method. Its outputs are legible and it has low error and high UQI. It was limited, however. In dirty images, the text is darker than the background. We haven't captured this locality information. For folded documents, the predicted image retains some of the crease lines. Remaining crease lines are narrower and not as dark as the writing. In dirty images, shadows next to the white centres of the crease remain as crease lines on the predicted image. This led us to hypothesize that the model for a single pixel needs to include information about the intensities of other pixels in the image.

The autoencoding method was complex and has many nuances. The autoencoder seems to "over clean" the document. Although it's the best at removing stains, it also "whitens" text in the process. One reason for the whitening could be that the training data set was small, consisting of only a few noisy and clean images. We added synthetic noise and applied linear transformations to generate more images. Since the newly generated images aren't accurate enough for their corresponding cleaned versions, discrepancies occurred in the output of autoencoder. "Whitening" of the images could also be due to the number of epochs that the autoencoder was trained to compute in the resultant images. Perhaps if we provided a large, quality dataset as an input for the autoencoder, there would be a significant improvement in the corresponding denoised versions of noisy images.

VI. Denoizer - Web Application

We have hosted this tool on AWS. A user logs into the system using user credentials and accesses the dashboard. In the dashboard, the user upload a noisy image, which will be stored on an S3 bucket on AWS. Then the user selects a technique to clean the noisy image. Techniques are presented in tabs on the navigation bar of the web tool.

After choosing the technique, the respective script and logic for each technique will run under the hood. Results

for each technique will be displayed in their respective tabs. Generated results are also stored on an S3 bucket, ensuring that users can access them at anytime.

A. Denoizer Architecture

Denoizer makes use of the following AWS services:

- **S3**: Store the uploaded image and results
- **API** Gateway: Develop login and upload APIs
- **Lambda**: Execute python scripts for each technique
- **Cognito**: Handle user management for providing access to AWS services
- **RDS**: Host login and image databases

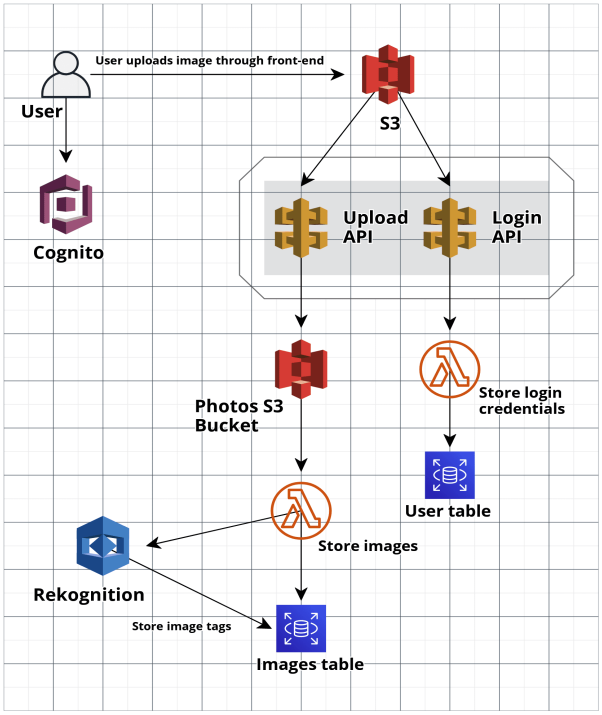

Fig. 11. AWS Architecture

B. Denoizer - Screens

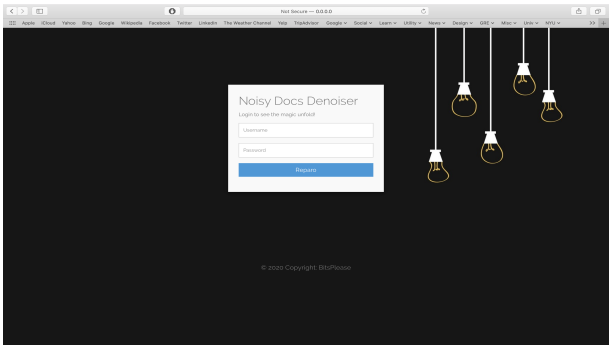

Fig. 12. Denoizer Login

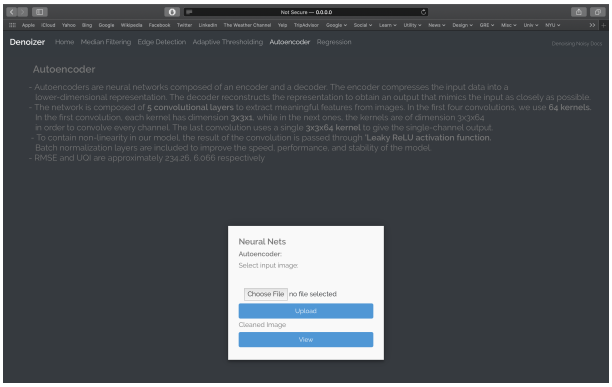

Fig. 13. Autoencoder

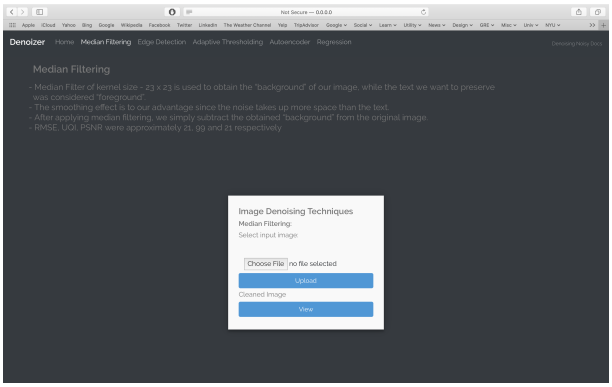

Fig. 14. Median Filtering

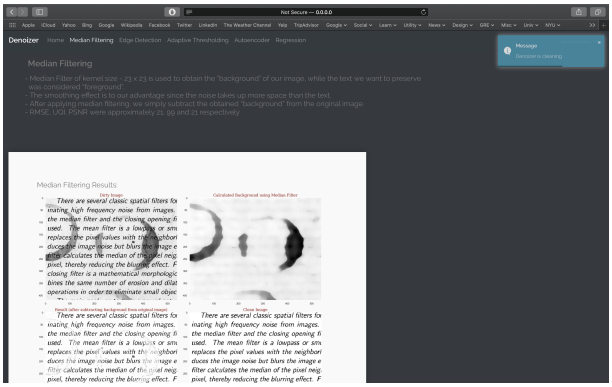

Fig. 15. Median Filtering Results

The code for the Denoizer application along with data sets and notebooks for the techniques covered in the report can be found at the following address:
https://github.com/gandalf1819/Denoise-docs-CV

VII. Conclusions & General Discussion

Without any machine learning, using only image processing techniques, three different insights were obtained on how the documents can be cleaned. However, the three methods are "imperfect" on their own. The output generated varies based on the noise of the input image which is why the choosing the best output out of all the generated results would be made solely by the user.

The autoencoder seems to be best at removing most "noise" (i.e. stains). As one can observe, there is no hint of any background stain left on the resultant image as previously described in the report. Although it's the best at removing stains, it's also "whitening" text in the process. The reasons being– the performance of Autoencoder technique is highly governed by the number of epochs the neural net is trained on. Hyperparameter tuning the neural nets becomes a crucial part in generating the best output from the dataset. Besides, we had insufficient data to feed to the CNN, and augmented our own dataset by adding synthetic noise and linear transformations. Given sufficient amount of "good" data and a certain number of GPUs, then presumably Autoencoder will give the best results. As for the CV algorithms– since the noise and features are "quite" similar we used the same kernel size/threshold value for all images. Thus a "learning" based kernel/threshold value estimator required.

For the final product, a user will upload a dirty image, and we will generate the respective clean versions of image using each technique. Thus, rendering measures such as RMSE, UQI PSNR will need to be moderated and the decision to compute the metric should be carefully chosen due to nature of variable input. At most we can measure the structural similarity using SSIM of a noisy and denoised image, if we don't have a clean reference.

On the software engineering front, this project is hosted as a flask web application on AWS EC2 instance. Currently we are hosting the web app under AWS Free Tier plan which caps us on the data storage limits and hosting the web app for longer hours. In order to fix this, we will be required to pay for a few AWS services like S3, RDS instances to keep the servers running 24x7. We tried creating a "live" dashboard/webapp using Tableau. However, Tableau seems to work with dataframes. The base framework of Tableau cannot handle/display images in the form of arrays.

A. Future Work

Each of the denoising algorithms provide different benefits for cleaning documents, with individual strengths and weaknesses. Future work could be done combining these techniques, enhancing their strengths and cancelling out their weaknesses.

Perhaps a "stacked" neural network could be composed of 6 convolutional layers with **5 input channels**:

- the original image
- the output from median filtering
- the detected edges
- the output from adaptive thresholding
- the output from the CNN autoencoder

The stacked model could use all five information sources to produce the final output. The autoencoder was trained for 10 epochs for the output presented in the report. We can try to hyperparameter tune the autoencoder and adjust the number of epochs to increase the learning for the model. Along with the autoencoder, we could add techniques like CycleGAN and compare the results between the autoencoder output and CycleGAN using RMSE, UQI and PSNR metrics. Also, we could use AWS rekognition to tag images, and extract text from images.

References

[1] Z. Wang, A. C. Bovik "A Universal Image Quality Index" IEEE Signal Processing Letters, Volume 9, Issue 3, Pages 81-84, August 2002.

[2] F. Zamora-Martinez, S. España-Boquera and M. J. Castro-Bleda, Behaviour-based Clustering of Neural Networks applied to Document Enhancement, in: Computational and Ambient Intelligence, pages 144-151, Springer, 2007.

[3] S. Kolwa (Thabs) "Noisy and Rotated Scanned Documents" Kaggle, 2020 (https://www.kaggle.com/sthabile/noisy-and-rotated-scanned-documents)

[4] Chollet, F. (2015) Keras https://keras.io

[5] Andrew Khalel Sewar http://sewar.readthedocs.io/