

Denoising Document

Patel Ronakkumar J.¹, Prof. Ajaykumar T. Shah²

Department of Computer Engineering

¹ Alpha College of Engineering & Technology, Gujarat, India

²H.O.D, Alpha College of Engineering & Technology, Gujarat, India

Abstract- Optical Character Recognition (OCR) is the process of getting typed or handwritten documents into a digitized format. The motivation of converting to a digitized format is to ensure security, accessibility, edit-ability and ease of searching and sharing.

Also, digital documents don't get dirty and cannot be ruined by coffee stains. [2] Unfortunately, a lot of documents eager for digitization are being held back. Coffee stains, faded sun spots, dog-eared pages, and lot of wrinkles are keeping some printed documents offline and in the past.

The image resulting from the scanning process may contain a certain amount of noise. Depending on the resolution on the scanner and the success of the applied technique for thresholding, the characters may be smeared or broken. Some of these defects, which may later cause poor recognition rates, can be eliminated by using a pre-processor to smooth the digitized characters.

Keywords- Denoising Document, Machine Learning, OCR, Neural Network

I. INTRODUCTION

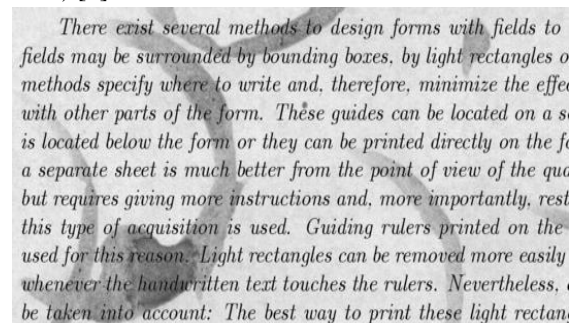
This Project "DENOISING DOCUMENT" is a pre-processor program which will process the image input of OCR.

Denoising Document project provide the solution of problem mentioned above. In this I train the machine learning model by giving it noisy document portion and corresponding clean document portion. To maximize the amount of training data, instead of simply cropping, i could "slide" a window across the picture, and train on batches of windows. This also allows us to clean documents of any size. At test time, I perform the same sliding operation, "clean" each windows, and stich them together to reconstruct the clean version of the original photo. For a pixel overlapped by different windows, i take the average of the corresponding pixels in each window as the final prediction.

Given a dataset of images of scanned text (synthetic images) that are "noisy" with stains and wrinkles, I propose to clean up the noise and help with the digitization process.

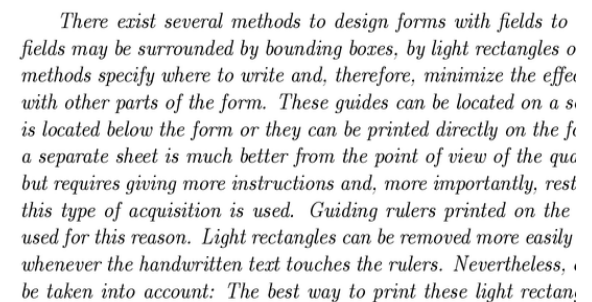
Dataset:

Dataset Kaggle provided a dataset which consists of 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 documents. The dirty images contain stains as well as creased paper. The training set also includes the cleaned up images of those found in the test file (train cleaned) [2].



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 fo a separate sheet is much better from the point of view of the que 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, i be taken into account: The best way to print these light rektan,

Original Image



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 fo a separate sheet is much better from the point of view of the que 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, i be taken into account: The best way to print these light rektan,

Cleaned Image

II. LITERATURE REVIEW

There are many places where OCR can't work properly because of the noise to overcome this problem I train the model according to previous data to predict the clean data according to noisy data.

We divide the document portion to small pieces. This pieces contain some pixels. To create this pieces we use sliding operation.

At test time, we perform the same sliding operation, "clean" each windows, and stich them together to reconstruct

the clean version of the original photo. For a pixel overlapped by different windows, we take the average of the corresponding pixels in each window as the final prediction. This also has an "ensembling effect".

The main model that we use to denoise is an autoencoder-like neural network.

$$f_0: R^{30 \times 30} \rightarrow R^{30 \times 30} \text{ given by,}$$

$$f_\theta = f_1 \circ f_2$$

where f1 is a convolutional encoder, f2 is a decoder (feedforward or deconvolutional), and θ is the parameters vector of f. f1 has the purpose of projecting the original image into a new space of lower dimension, whereas f2 attempts to restore the original image from this representation. A certain degree of information will be lost in the process, which hopefully is the noise we want to remove.

There is two methods which I can use to fulfil our requirements.

Random Forest

Here I propose a purely machine learning technique without any pre-processing whatsoever. The basic idea is to use a random forest regressor model to predict the pixel intensity based on neighbouring pixels.

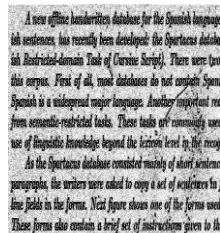
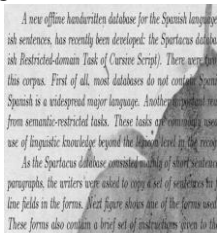
Algorithm:

Pad out each image by an extra 2 pixels (i.e.) $N \times N$ becomes $(N + 2) \times (N + 2)$.

Run a 3×3 sliding window on the image. Please note that every pixel of the original image will at least become the centre of the sliding window once.

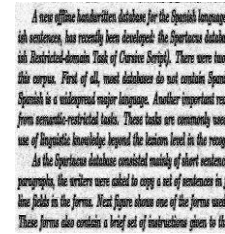
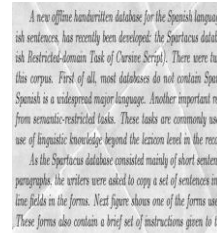
Use all 9 pixels within the sliding window as predictors for the pixel in the centre of the sliding window (i.e.) All the pixels in the sliding window of the dirty image acts as a feature to predict the centre pixel of the window for the cleaned pixel.

Use a Random Forest regressor model to predict the pixel brightness.



(a) Original Image

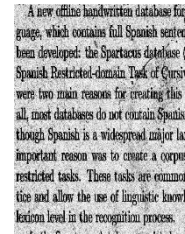
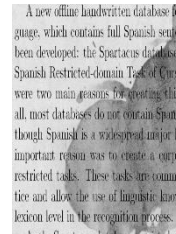
(b) Cleaned Image



(c) Original Image

(d) Cleaned Image

Figure 1: Using Random Forest a)



(a) Original Image

(b) Cleaned Image

Figure 2: Using Random Forest b)

While this method succeeds in removing the stains [5], it does not work very well with dog-ears and creases [4], in fact random forest just makes it worse. It looks as if random forest takes the stain and sprinkle it across the entire image so that the stains are not concentrated in one particular spot but more milder but widespread. This, as one can see from the cleaned image, is not conducive for reading and thus will not help us in our goal of converting to a digitized format for future use.

The RMSE score in Kaggle is 0.32492.

Challenges Faced

Fitting the training data to the model was gigantic task. I initially tried partial fitting but the results obtained were just random noises. The entire data-set had to be loaded simultaneously to get at least a proper output. Also training the model took around half an hour as I were unsure how to use GPU for this computation. To facilitate easier understanding I opted to go with IPython which is a very powerful interactive python shell. This helped us in saving the trained models and tracking variables without re-doing the entire thing.

Neural Network

I create a simple feed-forward neural network that denoises one pixel at a time. This neural network has one hidden layer. Each layer contains a weight matrix W and a bias vector b and computes the function:

$\text{act}(\text{input} * W + b)$ where act is typically some sort of sigmoid function.

The activation function of the input layer is the tanh function, while the activation function for the hidden layer is the clip function of Theano which clips the value based on the

```
def
clip (x , minx ,maxx) : if (x < min) :
return minx
el if (x > maxx) :
return maxx
return x
```

given minimum and maximum value (i.e.)

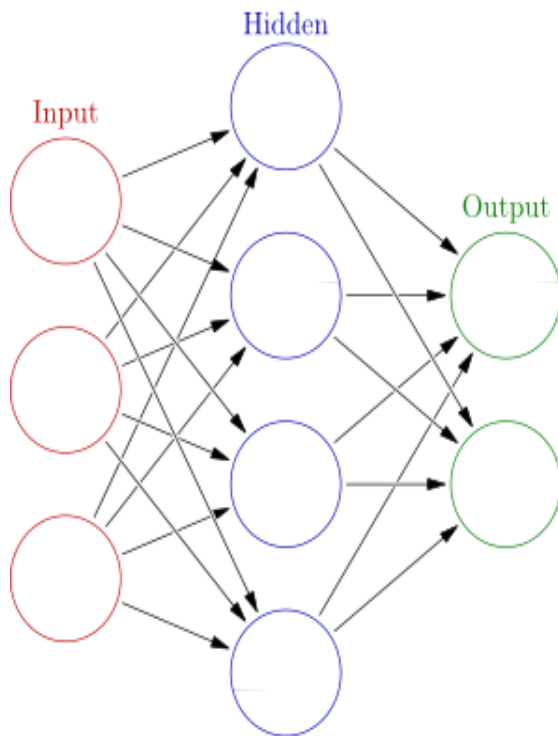


Figure 3: Artificial Neural Network [6]

The hidden layer contains 10 neurons, the no. of neurons for the input is 29 (which is the no. of feature vectors) and output layers has one neuron which is the pixel brightness.

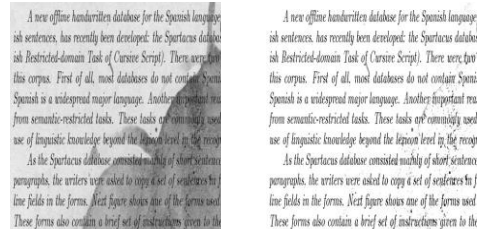
Before passing the images to the neural network, we first calculate the features of the image. I consider neighbouring pixels of the centre pixel using a 5x5 window as boundary as features. So for each pixel I have a feature vector containing 25 feature points. Also I do some initial image processing on these image and take the output as features for the neural network. I use median blur with kernel size 5 and kernel size 25. Using the

Sobel operative I calculate the first and second derivative of the images. For each pixel of the image, I have 4 image processing outputs, the median blur with kernel size 5, the median blur with kernel size 25, first Sobel derivative and second derivative. These are then added to the already existing 25 feature points making the total to 29 feature points for each pixel. The feature vectors are combined together to create a feature matrix for the image and given to the neural network.

Central Idea:

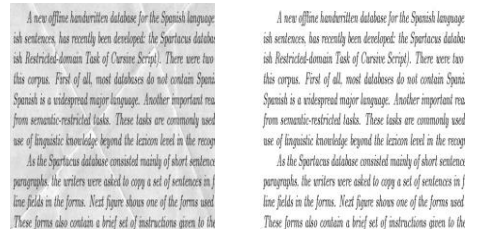
- Take a pixel from an image
- Calculate feature vector as mentioned above. It contains a total of 29 feature points.
- This is the input to Neural Network Model.
- Output is the de-noised pixel (i.e.) the intensity of the cleaned pixel.

I train the neural network using a naive gradient descent learning algorithm with the entire data-set.



(a) Original Image

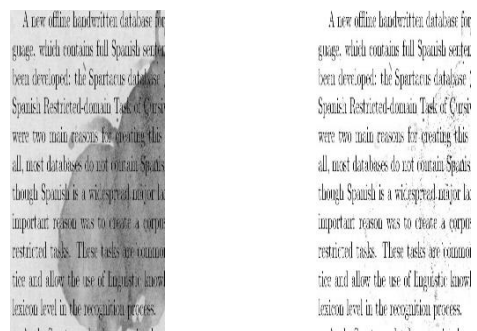
(b) Cleaned Image



(c) Original Image

(d) Cleaned Image

Figure 4: Neural Network a)



(a) Original Image

(b) Cleaned Image

Figure 5: Neural Network b)

As you can see from [4] and [5] the creases are pretty much invisible to the eye while the stains are faded to the point that only faint patches are visible.

The RMSE score in Kaggle is 0.03363.

Challenges Faced

I could not use the entire training data as our RAM was too small for it. I used only half the training data for this method. Ideally I should have trained this for at least 100 iterations(epochs) but due to low computational power i trained it only for 10 iterations(epochs) which took around 20 minutes in a GPU.

III. CONCLUSION

Comparing the results of all the methods listed I find that ANN works the best. It removes the stains & crevices and it is readable!! While the other methods remove stains, the text is quite hard to decipher as it is blurred or the ink is too thin. The RMSE values of the test data using our methods and the original image are listed in the table below:

Methods/Score	RMSE (%)
Fixed Thresholding	35.173%
Adaptive Thresholding	42.228%
Canny Edge (Dilation)	51.638%
Canny Edge (Erosion)	36.547%
Median Blur	55.096%
Random Forest Regressor	32.492%
Artificial Neural Network	3.363%

Table 1: Table with methods and their RMSE scores

FUTURE WORK

The images that I were able to clean are images of English texts. I plan on expanding this to cover texts in other languages, figures, combination of both facts and figures.

I have a tentative plan to create an android application that can remove stains and creases using the above mentioned methods.

A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In my project I can use RNN for handwriting reorganization and for predicting the missing character or words if they are too noisy to clean and can't clean by my algorithm.

REFERENCES

- [1] Colin blog. <http://tinyurl.com/gnptby6>.
- [2] Kaggle - denoising dirty documents. <http://tinyurl.com/z4ukatx>.
- [3] Kaggle blog. <http://tinyurl.com/gnedxjq>.
- [4] <https://github.com/Perseus14/>
- [5] Colin blog. <https://colinpriest.com/2015/09/07/denoising-dirty-documents-part-6/>.
- [6] Kaggle - denoising dirty documents. <http://tinyurl.com/z4ukatx>.
- [7] Glosser.ca. Artificial neural network. <https://commons.wikimedia.org/w/index.php?curid=24913461>.