Ensemble LUT classification for degraded document enhancement

Tayo Obafemi-Ajayi, Gady Agam, Ophir Frieder

Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616

ABSTRACT

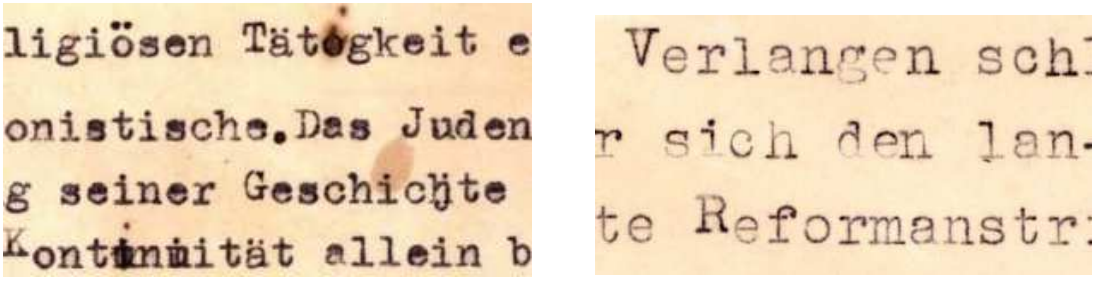
The fast evolution of scanning and computing technologies have led to the creation of large collections of scanned paper documents. Examples of such collections include historical collections, legal depositories, medical archives, and business archives. Moreover, in many situations such as legal litigation and security investigations scanned collections are being used to facilitate systematic exploration of the data. It is almost always the case that scanned documents suffer from some form of degradation. Large degradations make documents hard to read and substantially deteriorate the performance of automated document processing systems. Enhancement of degraded document images is normally performed assuming global degradation models. When the degradation is large, global degradation models do not perform well. In contrast, we propose to estimate local degradation models and use them in enhancing degraded document images. Using a semi-automated enhancement system we have labeled a subset of the Frieder diaries collection.[1] This labeled subset was then used to train an ensemble classifier. The component classifiers are based on lookup tables (LUT) in conjunction with the approximated nearest neighbor algorithm. The resulting algorithm is highly efficient. Experimental evaluation results are provided using the Frieder diaries collection.[1]

Keywords: image enhancement, historical documents, document degradation models, ensemble classification, document image analysis

1. INTRODUCTION

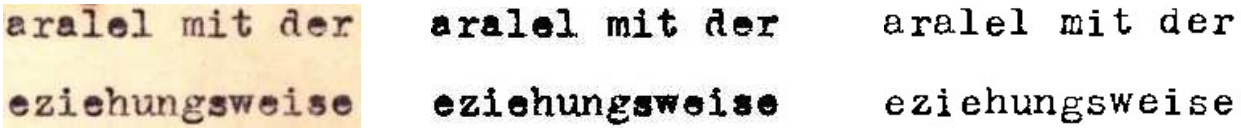
The enhancement of old typewritten historical documents is very essential and needful for preservation and continuation of information. They currently exist electronically as scanned document images. Not only is the quality of the typewritten text poor and non-uniform, many of these documents have also deteriorated due to age of paper and ink used. The characteristics of the deterioration include noisy background, paper discoloration, creases, blurred, merged and faint text.[2] Usually, typewritten text contains non-uniform characters, some darker or faint than others, depending on the amount of force used in striking the typewriter keys[3] while some of the characters may be blotted such as the 'e's, as illustrated in Fig. 1. The degradation of the text hinder the readability of these documents, as seen in Fig. 1, and the level and type of degradation vary from document to document. Thus, there is need for an adaptable automated system to enhance these documents to improve their readability.

The existing state of the art document enhancement systems for processing historical documents focus primarily on segmentation techniques which involves foreground-background separation. The text in the documents is classified as foreground while everything else is rendered as background. While these systems perform well in obtaining a relatively uniform background, they are unable to effectively correct the distortions in the foreground such as blotted text, broken characters, or overwritten characters. Sometimes, the text in the document are further degraded during the foreground-background process. Our proposed approach goes beyond the current state of the art systems in its



(a) "blotted/filled characters" (b)"fainted text"

Figure 1. Different Degradations in Typewritten Documents



(a) Original distorted document image (b) Binary form of distorted image (c) Ground Truth image obtained manually

Figure 2. Example of the binary image format and the ground truth data derived from an original distorted document image using the interactive document enhancement software

ability to enhance text degradations in typewritten documents, beyond the foreground-background separation phase, to improve the readability of the documents. We present an automated adaptive system, based on look up table (LUT) training and classification algorithms, which learns the patterns of text degradation and the corresponding enhancements in the document images. We train on real degraded historical documents obtained from a subset of the Yad Vashem Holocaust museum document collection.[1] The ground truth data for these degraded document images is generated manually by a human expert using an interactive document enhancement software (a continuation of the existing work of Agam et al.[4]). The software allows the human expert to manually correct the distortions in a document character by character to generate the ideal uniform clean text document image of the degraded image, as illustrated in Fig. 2. We evaluate the performance of our system by applying it to a set of test data also obtained from the collection. The performance of our system is measured both quantitatively (Pixel Accuracy) and qualitatively (enhanced readability) in comparison to the ground truth data. Our system is able to perform the task of enhancing a single document image in less than 1 minute thus making it a more efficient way to correct a large set of documents quickly compared to the manual process using the interactive software which can take up to 5 hours per document.

Our main contributions are (1) the simplicity and novelty of the design of degraded document image enhancement LUT classifiers; and (2) an efficient system that can process multiple documents in one pass. Our LUT classifier system can be used as an add-on to existing foreground-background separation systems to further improve their results. In the subsequent sections, we describe the proposed approach fully and then present the experimental results obtained to validate our approach. We also compare our work to related systems in Sec. 2.

2. RELATED WORK

Some work has been done on conversion of historical documents to a logically indexed, searchable form by Antonacoupoulos et al. in.[2] Their approach is based on content extraction using semantic information which involves the expert knowledge of a historian/archivist. In contrast, our approach does not entail having knowledge of the underlying information contained in the document. Antonacoupoulos et al. in[5] attempt to enhance these documents to prepare them for optimal OCR performance using an off-shelf OCR package. They attempt to enhance the documents by individually segmenting and enhancing each character while our proposed approach learns degradation patterns of the characters in the context of an entire document image.

The existing foreground-background separation based systems for enhancing degraded historical documents includes the work done by Gatos et al. in[6] and Agam et al. in.[4] The system developed by Gatos et al. binarizes historical documents based on adaptive threshold segmentation and various pre- and post-processing steps. An iterative approach for segmenting degraded document images is described by Kavallieratou et al.[7] The work done by Agam et al. is based on probabilistic models utilizing the expectation maximization (EM) algorithm. Our proposed system goes beyond this class of system as we focus on correction of degradations in the foreground. We handle the foreground-background separation during the preprocessing stage of our system. Our system can be employed as an add-on to these systems. This is beneficial, as such systems sometimes introduce additional distortions in the foreground during the process of background removal. Our classifier can be applied to the binary image outputs to correct any additional foreground distortions incurred during the segmentation process.

Molton et al. in[8] apply pattern recognition concepts of illumination and shadowing to enhancement of incised documents. Their work deals with tablets, which are a special class of historical documentary source, unlike our work which focuses on typewritten documents. Andra et al. in[9] train classifiers to detect styles of pattern in documents in order to classify documents from similar sources. The context and focus of their work differs from our proposed approach as we focus on learning degradation patterns with the goal of enhancement, not to determine the source. As-Sadhan et al. in[10] did a comparative study of applying different algorithms such as Support Vector Machine

(SVM), Principal Component Analysis (PCA), and Single-Nearest-Neighbor Method (1-NNM) to distorted-character recognition for OCR-based techniques.

Zheng et al. in[11] train classifiers to restore document images based on morphological degradation models. They build a look up table, similar to our approach, using a 3×3 filter. However, their look up table consist of a matrix mapping each entry to at most 512 possible outputs, unlike our approach that maps each entry to two possible outputs. We also use real degraded document images during our training phase which differs from their approach of utilizing synthetic images generated using the Kanungo morphological degradation model.[12] Their degradation model is well suited for uniform text document images corrupted during document generation and copying processes but unable to handle the degradation characteristics of historical typewritten document images, the core of our work. We discuss more extensively the detail of the comparison of our approach to Zheng et al.'s restoration algorithm based on Kanungo's degradation model in Sec. 4.

3. ENSEMBLE LUT CLASSIFICATION

The core of our work is the design of effective classifiers that enhance the readability of historical typewritten documents by learning the patterns of degradation and enhancement from the training data set. The training data set consists of pairs of a binary degraded document image and the corresponding ground truth image. The goal of the training phase is to build the look up table (LUT) which is utilized during the enhancement phase to correct the degradations in the document image, as we describe in detail in Sec. 3.1 and Sec. 3.2.

The LUT classifier processes binary document images consisting of only black (foreground) and white (background) pixels. Historical documents are currently stored electronically as scanned color or grayscale document images. Thus, we preprocess the document images to convert each scanned degraded document image to a binary image by separating the foreground from the background. The binary image can then be effectively processed by our classifier. The preprocessing phase attempts to remove the background degradations to generate a uniform background. The nature of degradation of the backgrounds varies from document to document for example, some have a really dark streaky background while others have blots of ink stains, wrinkling, etc.[2] There are different segmentation algorithms that are adaptive to the varied nature of degradation of background in the document images which can be employed to obtain a binary document image, as discussed in Sec. 2. We utilize the adaptive Min-Max threshold algorithm[4] (a Bremen segmentation technique) in the segmentation process because of its known efficiency.[4] Our LUT classifier system is portable in that it is adaptable to other foreground-background separation systems. If there is an existing binary document image, obtained using another segmentation technique, the distortions in the text in the binary image can still be enhanced by feeding it directly into our system, bypassing our preprocessing stage.

3.1 Training Phase: Building the Look Up Table (LUT)

Suppose we have an image pair in our training data set $T = \{(D, G)\}$, where D is the binary degraded document image and G, the corresponding ground truth image. Let N represent an arbitrary $w \times w$ neighborhood bit pattern in D with p_i representing its center pixel located at position (x, y) while p_o denote the pixel at same position (x, y) in G. Let $p(x, y)$ represent the pixel value at (x, y) and $b(x, y)$ represent the binary code for the neighborhood N centered at (x, y). The i-th bit of $b(x, y)$, where $i \in [0, w^2-1]$, is denoted by $b_i(x, y)$. We have $b_i(x, y) = p(x+L_x(i), y+L_y(i))$ where $L(i) \equiv (L_x(i), L_y(i))$ is the relative displacement of the i-th pixel in the neighborhood with respect to (x, y). The relative displacements are given by $L_x(i) = i\%w - \lfloor w/2 \rfloor$ and $L_y(i) = i/w - \lfloor w/2 \rfloor$. E.g. for a 3×3 neighborhood: L_x = [-1 0 1 -1 0 1 -1 0 1] and L_y = [-1 -1 -1 0 0 0 1 1 1]. Using $b_i(x, y)$ as above, the binary code $b(x, y)$ is given by: $b(x, y) = \sum_{i=0}^{w^2-1} b_i * 2^i$.

Let $P(p_o|N)$ be the conditional probability of the output center pixel p_o at (x, y) in D given the neighborhood information N centered at (x, y) in D. The goal of the training phase is to obtain the data needed to estimate $P(p_o|N)$ for all neighborhood patterns found in D. For each occurrence of a N in D, we obtain its frequency set $\{F(1|N), F(0|N)\}$, defined as the number of times p_o is a *foreground* pixel, and *background* pixel respectively, for all occurrences of $N \in D$. (We represent *foreground* pixels as 1 and *background* pixels as 0). We estimate $P(p_o|N)$ using its frequency set information. The LUT is a mapping of all the unique patterns of N existing in D to its frequency set $\{F(1|N), F(0|N)\}$, as illustrated in Fig. 3. The neighborhood size the LUT considers, $w \times w$, can also be viewed as the dimensionality of its filter window.

To build the LUT, we scan each pixel p in D to obtain its corresponding N except for two sets of pixels which we consider *not-relevant*. The first *not-relevant* set are all pixels for we cannot obtain a complete neighborhood

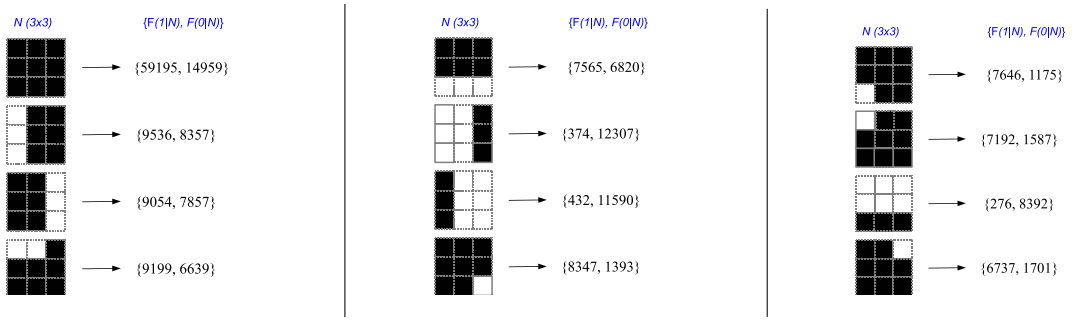


Figure 3. An example of the 12 most occurring entries in a look up table (LUT) generated using a 3x3 filter window. Each entry consists of unique neighborhood bit patterns N found in the degraded document image and the corresponding frequency set of $F(1|N)$ and $F(0|N)$.

pattern in D, i.e., the boundary pixels located at position $\{(x,y)|x < w/2 \vee x > n - w/2 \vee y < w/2 \vee y > m - w/2\}$ where (m,n) are the image dimensions of D. This does not diminish the effectiveness of the classifier, as there are generally no foreground data contained in the border region of document images. The second set are all pixels having a neighborhood of only white pixels, i.e. pixels at position $\{(x,y)|b(x,y) = 0\}$. These pixels are also overlooked since N has no foreground data. This greatly reduces the number of pixels we have to process in D. The algorithm to build a $LUT(w,T)$, given $T = \{(D,G)_i, i = 1,\ldots,t\}$, is detailed in algorithm 1.

Algorithm 1 Build-*LUT*

Build-LUT(w, $T = \{(D,G)_i, i = 1,\ldots,t\}$)

$F(1|N) = 0$; $F(0|N) = 0$ //initialize frequency sets to zero
for all $(D,G)_i \in T$ **do**
  for all *relevant* $p_i(x,y) \in D$ **do**
    obtain $N = b(x,y)$
    if $p_o(x,y) = 1$ **then**
      $F(1|N) + 1$
    else $\{p_o(x,y) = 0\}$
      $F(0|N) + 1$
    end if
  end for
end for

end buildLUT

3.2 Enhancement Phase: LUT Classification

During the enhancement phase, we apply the LUT classifier to a given degraded document image $D \notin T$ (i.e. the training data set) to obtain its enhanced image $\hat{G}$. The basic LUT classifier is an ensemble of two classifiers: (i) ANN cluster classifier, and (ii) Maximum Likelihood (M-L) decision classifier. To enhance D given a $LUT(w,T)$ we scan each pixel $p(x,y) \in D$, using the filter window size w, to obtain its corresponding $N = b(x,y)$. (The same set of pixels ignored during the training phase are also overlooked during the enhancement phase). There are two main steps in the enhancement process: the first step is the *lookup operation* of N, handled by the ANN cluster classifier, while the second step is the pixel classification decision of the output center pixel, performed by the M-L decision classifier. Both steps are described in detail below.

ANN Cluster Classifier

During the enhancement process, it is important that our LUT can generalize well to be able to process unseen samples (i.e. values of N not encountered during the training phase). For example if we have a 5×5 neighborhood, even a small difference in one pixel out of the 25 total pixels can cause the lookup operation of N in the LUT to fail, if the slight variation was never trained for. To overcome this, we perform the lookup operation using an ANN

cluster classifier which utilizes the k-Nearest Neighbors Search Algorithm by ANN[13] to search for similar entries to the unseen sample. ANN performs approximate nearest neighbor searching, based on the use of standard and priority search in kd-trees and balanced box-decomposition (bbd) trees. The ANN classifier returns the frequency set of N, if N is found in the LUT, or the frequency sets for k most similar entries of N found in the LUT. This output is passed on to the M-L Classifier to make a pixel classification decision. Thus, the lookup operation classifies each pattern of degradation i.e. N to exactly the same or k most similar patterns of N existing in the LUT.

Each entry N in a given LUT is represented using its binary code $b(x, y)$, as described in Sect. 3.1, that are preprocessed by ANN into a kd-tree[14] data structure. To compute the similarity distance for any two entries, ANN uses the Euclidean distance between their binary codes. For any query point $N \notin$ LUT the ANN classifier is able to report the k nearest entries with ϵ approximation to N efficiently. The ϵ specifies the maximum approximation error bound, which permits us to control the tradeoff between accuracy and running time. We show the impact of both ANN parameters, k and ϵ, on the running time and accuracy of our LUT classifier in Sec. 4.

Maximum Likelihood Classifier

The M-L classifier makes a pixel classification decision by estimating the conditional probability of the output pixel $P(p_o|N)$, as defined in Sect. 3.1, using the frequency set information of N obtained from the ANN classifier.

$$p_o(x, y) = \underset{p \in \{0,1\}}{\operatorname{argmax}} P(p|N) \tag{1}$$

The computation of the value of output center pixel given its neighborhood information N is essentially the maximum likelihood estimate of $p_o(x, y)$ being a foreground or a background pixel using the conditional probability obtained from the associated frequency set $\{F(1|N), F(0|N)\}$. Given the frequency of occurrence of $p_o(x, y) \in G$ being 1 or 0 for N during training, we estimate the value of $p_o(x, y) \in \hat{G}$ to be 1 if $F(1|N) > F(0|N)$, and vice-versa for 0. If $F(1|N) = F(0|N)$, we take no action: $p_o(x, y) = p(x, y)$.

The ANN classifier may determine k neighbors. It sends the frequency set information for a set $\{N_i, i = 1, \ldots, k\} \subset LUT(w, T)$ to the M-L classifier. If $k > 1$, the pixel classification decision of $p_o(x, y)$ is based on the majority vote over the set $\{N_i, i = 1, \ldots, k\}$. For each N in the set, we obtain its estimate of $p_o(x, y)$ using equation (1) and then compute the majority vote over the individual estimates obtained. If there is no majority vote, then no action is taken i.e. $p_o(x,y) = p(x,y)$.

The enhancement process is summarized in Algorithm 2.

3.3 Performance of LUT Classifier

Theoretically, the size of the LUT ($\|\{N\}\|$) is bounded by $\boldsymbol{O}(2^{w^2})$ as $N = b(x, y)$ has a length of w^2. This implies an exponential memory requirement which will translate to a very inefficient system. For example, using a *w-5* filter for an LUT would require a memory storage of about 33MB (2^{25}) while for *w-7* filter 524288GB (2^{49})! Intuitively, the actual bound of the LUT will be much less given that not all possible pixel pattern configurations will exist in typewritten document images. To validate this assumption we measured the number of different neighborhoods occurring in actual documents images. We used a set of 25 document image pairs to observe the size of the LUT for $w = 5, 7, 9$. From the experimental results, we saw that a small percentage of all the possible bit patterns exist in document images. The percentage of entries to the total number of theoretically possible entries actually decreased exponentially as w increased. Therefore the bound on the size of the LUT is $\ll 2^{w^2}$. Our experiments, as discussed in Sect. 4, demonstrate that a small set of images is sufficient to learn the degradation and enhancement patterns. To improve the performance of the LUT, we utilize the map container data structure. The performance of lookup operation for each N is $\boldsymbol{O}(\log(||T||))$. For a given entry N in the LUT, we define the frequency marginal difference as the difference between $\{F(1|N)$ and $F(0|N)\}$ in its corresponding frequency set. Usually in a LUT, there are some entries that have very little or no marginal differences for example, frequency sets such as $\{1, 0\}, \{245, 247\}$. This implies that the probability for choosing foreground or background as the output pixel when we encounter the pattern N during the enhancement phase is almost equal. Thus, if we eliminate these entries that have very small marginal difference from our LUT, we may improve the performance of our classifier by trimming away trivial entries. This process is referred to as 'Pruning' the LUT. The pruning threshold PT is defined as the minimum absolute marginal difference allowed for the frequency set of each N retained in the LUT. We present and discuss experimental results of pruning on the performance of the classifier in Sec. 4.

Algorithm 2 Enhance-D to obtain $\hat{G}$

Enhance-D(w, LUT, ϵ, k)

arrange LUT into ANN structure with parameters ϵ and k
for all *relevant* $p_i(x,y) \in D$ **do**
  obtain $N = b(x,y)$
  if $N \in$ LUT **then**
    ANN Classifier returns $\{F(1|N), F(0|N)\}$ for N
    set $p_o(x,y) \in \hat{G} = 0 \backslash 1 \backslash p_i(x,y)$ using equation 1
  else $\{N \notin$ LUT$\}$
    $vote0 = 0; vote1 = 0$ //counters for majority voting
    ANN Classifier returns $\{F(1|N), F(0|N)\}$ for $\{N_i, i = 1, \ldots, k\}$
    for $i = 1$ to k **do**
      (using equation 1 based on N_i)
      if $p_o(x,y) = 0$ **then**
        $vote0 + 1$
      else if $p_o(x,y) = 0$ **then**
        $vote1 + 1$
      end if
    end for
    //time to take majority vote to set $p_o(x,y) \in \hat{G}$
    if $vote0 > vote1$ **then**
      $p_o(x,y) = 0$
    else if $vote1 > vote0$ **then**
      $p_o(x,y) = 0$
    else $\{vote1 = vote0\}$
      $p_o(x,y) = p_i(x,y)$
    end if
  end if
end for

end enhance-D: output $\hat{G}$

3.4 Cascade LUT Classification

To further improve the performance of our LUT classifiers, we propose a method of applying the classifiers in a cascaded configuration. When we train a basic LUT classifier, as described in Section 3.1, we compare a degraded binary document image D to its ground truth image G, given $T = \{(D,G)_i, i = 1, \ldots, t\}$, to produce a single LUT. In the cascade LUT classifier configuration, we produce multiple LUTs during the training phase from the same training data set T. Let LUT_1 denote the first LUT obtained by comparing each $D \in T$ to its corresponding G. We apply LUT_1 on each $D \in T$ to obtain its estimated enhanced image $\hat{G}$. We then build LUT_2 using $T' = \{(\hat{G}, G)_i, i = 1, \ldots, t\}$. We compare the output image $\hat{G}$ resulting from applying LUT_1 on the degraded binary image D to the ground truth image G again to obtain another LUT. A two-stage cascade LUT classifier comprises of LUT_1 and LUT_2. To enhance a document image $D \notin T$, we apply LUT_1 and LUT_2 in the same sequential order as they were built. Thus we apply LUT_1 initially to D to get $\hat{G}_1$, then we apply LUT_2 on $\hat{G}_1$ to obtain $\hat{G}$, which is the final enhanced image of D given by the cascade configuration.

The goal of the cascade is that, with each stage, the next LUT improves on the work done by the previous LUT. Each stage in the cascade attempts to correct the more difficult points to classify in the original document. There is an additional overhead cost of increased training and execution time - twice the cost of training a single LUT. We can generalize the cascade LUT classifier to comprise of m LUTs with a cost equivalent to m times the cost of training and using one LUT. The cascaded LUT is a different variant of the ensemble LUT classifier - it consists of a set of m classifiers applied in sequential order. While building a m-cascade LUT classifier, the process is terminated if during the iterations of training new LUTs, we obtain an LUT_{i+1} that yields no more improvement on the training data compared to the former LUT_i. The performance of the cascaded LUT classifier is discussed in Sec. 4.

4. EXPERIMENTAL RESULTS AND ANALYSIS

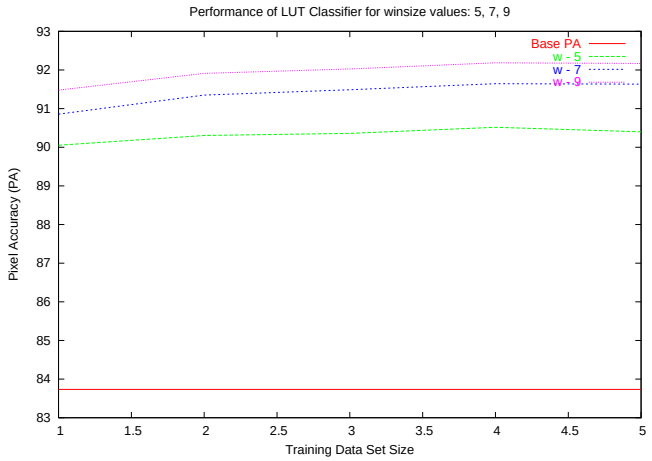
Evaluation of our proposed approach is done by comparing the resultant images obtained to the ground truth images generated by human expert, as explained in Sec. 1. To quantitatively measure the performance of LUT classifier, we use Pixel Accuracy PA as the performance measure. PA is defined as $(M/P) \times 100$, where M is the number of pixels in the output image $\hat{G}$ that match with the ground truth image G and P is the number of pixels in the original binary degraded image D. Given that our goal is to improve the readability of these documents and the ground truth image is a perfect standard of readability, based on human judgment, we relate the pixel accuracy to readability. Usually, an improvement in pixel accuracy to the known truth implies improvement in readability. We also perform a qualitative analysis of the results obtained by observing them visually to validate that there is actually an improvement in the readability. The efficiency of the classifier is measured by its execution time in seconds.

The *base PA* is the value of *PA* obtained by comparing the binary image (obtained after preprocessing) to its ground truth image before we apply the classifier to the image. This is the effect of applying a classifier that does nothing to the image beyond the background removal stage. The *base PA* enable us to quantify how much improvement is obtained by our LUT classifiers beyond the foreground-background separation systems. We show preliminary results obtained thus far on six document images in our ground truth data set. Each document image is approximately 1200 by 1750 pixels in size and contain 2400 character instances on the average, bringing the total number of characters to roughly 15,000. We performed character segmentation on each document image prior to applying the filter to ensure that as we scan the document image pixel by pixel, the filter window does not overlap neighboring characters. It ignores any neighboring character's pixel information contained in its window.

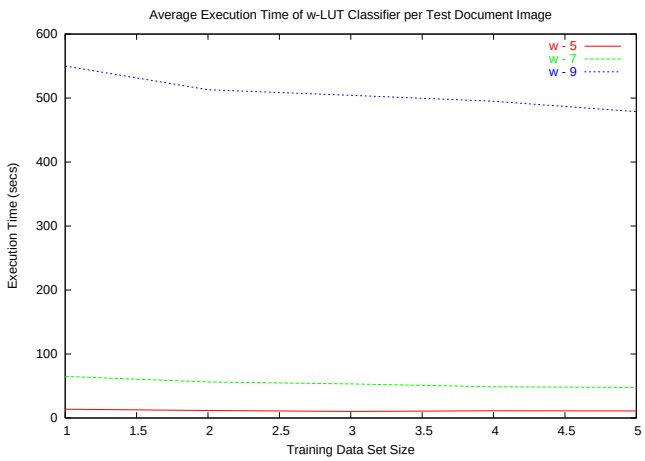
Figure 4a illustrates the performance of the LUT classifier for three different *winsize* values { 5, 7, 9 } as a function of the size of training data set T. From Fig. 4a , we observe that the *w-9* classifier attains the best performance on enhancement of the degraded images. This implies that the larger the size of locality of neighborhood considered by the filter, the better the enhancement. A quantitative result is shown in Fig. 5. We can observe that the output image of the *w-5* filter, as shown in Fig. 5d, is blurred compared to the results of the others filters. The characters in the output image of *w-9* filter are much clearer and distinct though some are still slightly broken. Increase in the *winsize* of the filter implies a greater complexity cost which affects the execution time of the classifier, as shown in Fig. 4b. The average execution time per document image using a *w-9* filter when the training set size T is one is 550s while for a *w-5* filter, it is 13.7s. As we increase the training set size T, the performance of the classifiers generally improves though the marginal improvement decreases. The PA using a LUT based on T of size 5 is actually less than that of $T = 4$. This implies that very large training set is not needed to enhance the degraded document images. What is more important is that the document images in T have very similar degradation patterns to the test document images. A few images, used during the training phase, is sufficient for the classifier to learn the patterns of degradation and enhancement. We can also observe, from Fig. 4b, that he execution time generally decreases as T increases. The execution time is mainly affected by the number of times the ANN classifier has to search for similar entries to N. As the size of T increases, the probability of locating the exact N in the LUT is increased so the frequency of searching for similar entries is decreased which results in lower execution time.

Figure 6 demonstrate the effect of pruning on the accuracy of the LUT classifier for the three *winsize* values. For *w-5* and *w-7* filters, the best result is obtained when the pruning threshold PT is set to 1. After that, the accuracy begins to diminish. Pruning however greatly diminishes the performance of the *w-9* filter. This is because the LUT is very sparse, given the large window size, and so most of the entries do have very low marginal differences therefore the seemingly trivia entries actually do matter. We lose a lot of information by pruning the *w-9* LUT. We can conclude that for the smaller *winsizes* of 5 and 7, pruning with PT set to 1, does help improve the performance of the classifier. When we prune the LUT, we lose a lot of information compared to the *w-5* classifier. Pruning increases the running time of the LUT, as can be observed in Fig. 6, because the probability of having to resort to search for similar entries using ANN structure increases and more entries are pruned from the LUT.

Figures 7 and 8 show the performance of the *w-5* LUT classifier as a function of ANN parameters. As can be observed in Fig. 7b, as we increase the number of neighbors k, we obtain a higher PA at an increased cost of execution time. As shown in Fig. 8, the distance approximation error bound parameter ϵ does not impact the performance of the classifier significantly. Using these graphs, we fix k and ϵ at a value that ensure a reasonable execution time and accuracy for our experiments.

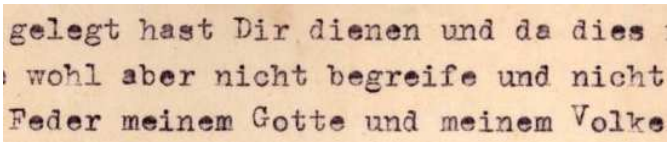


(a)Pixel Accuracy of LUT classifier for winsize 5, 7 and 9

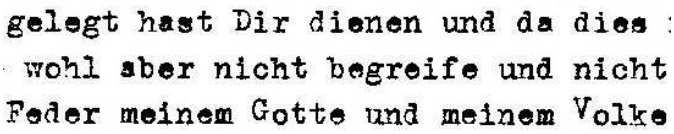


(b)Execution Time

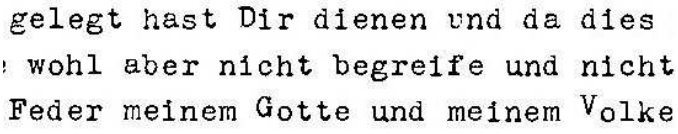
Figure 4. Performance of LUT classifier for different winsizes (5, 7, 9). The size of the training data set T was varied from 1 to 5.



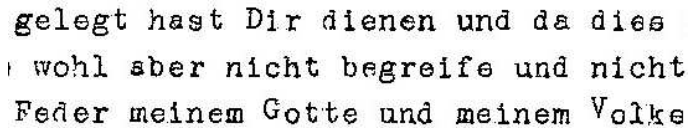
(a) Original distorted document image in color



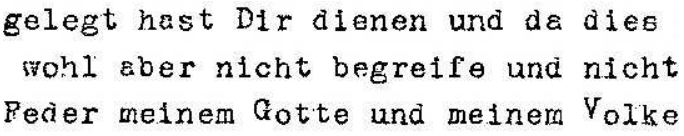
(b) Binarized version of distorted image



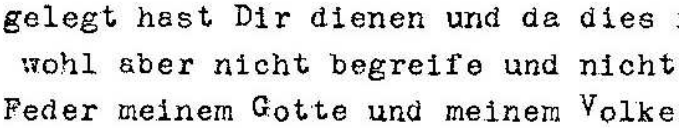
(c) Ground Truth version



(d) Output image using *w-5*filter

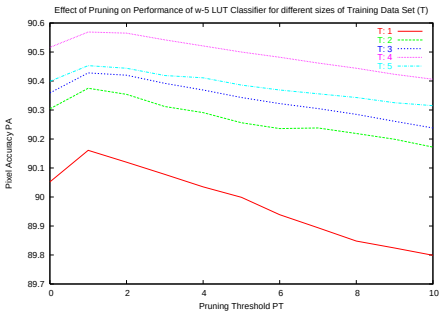


(e) Output image using *w-7*filter

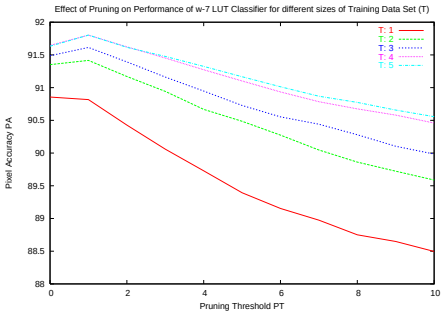


(f) Output image using *w-9*filter

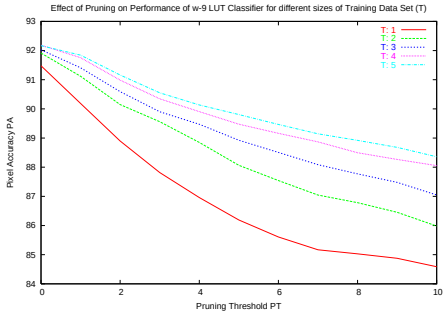
Figure 5. Result of applying LUT classifier of different *winsizes* on a test document image. The size of the training data set T used is 1.



(a)w - *5* LUT

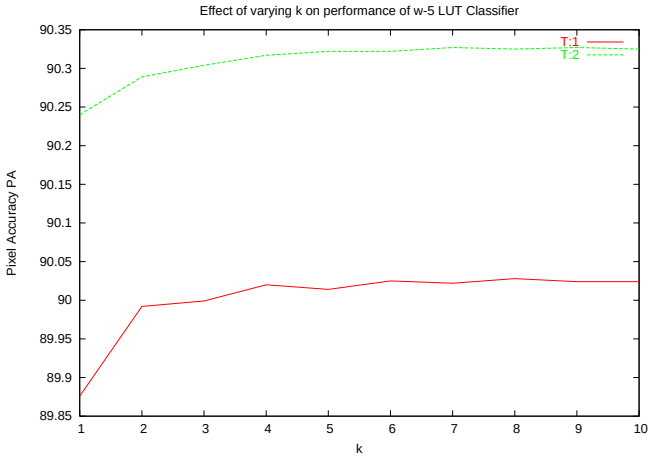


(b))w - *7* LUT

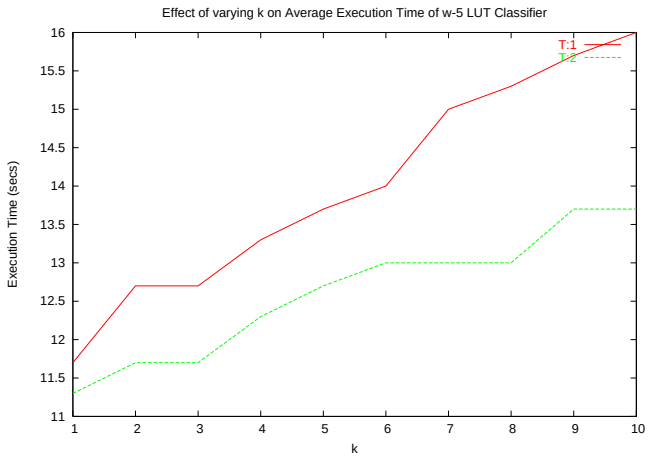


(c))w - *9* LUT

Figure 6. Effect of pruning on LUT performance for winsizes 5, 7, 9 for different training data set T sizes

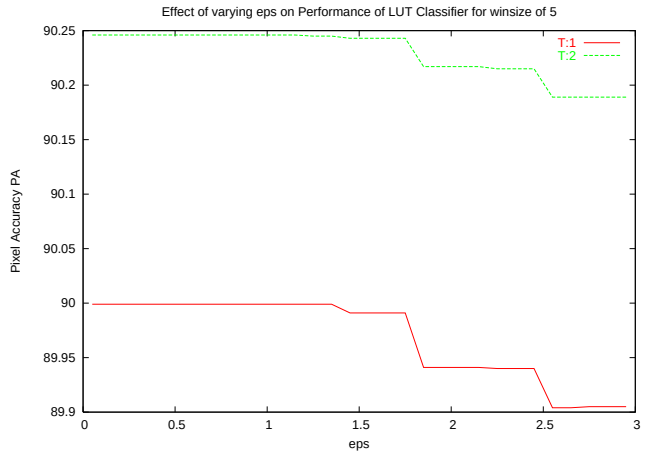


(a) Accuracy of classifier as we vary k

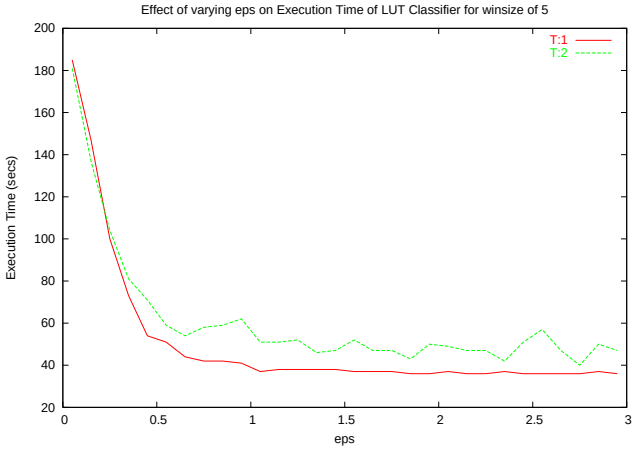


(b)Execution Time of classifier as we vary k

Figure 7. Performance of the w-5 LUT classifier as a function of k (ANN parameter) with ϵ fixed at 1.25

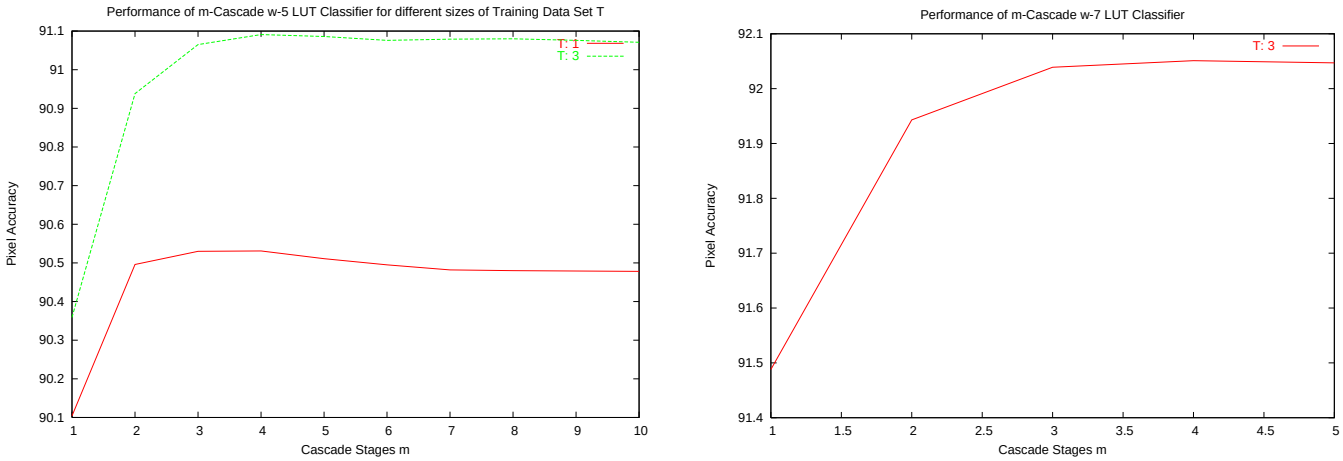


(a) Accuracy of classifier as we vary ϵ (eps)



(b)Execution Time of classifier as we vary ϵ (eps)

Figure 8. Performance of the w-5 LUT classifier as a function of ϵ (ANN parameter) with k fixed at 3



(a) (b)

Figure 9. Performance of Cascaded LUT

Performance of Cascade LUT Classifiers

The performance of the m-stage cascaded LUT classifiers results in an improved performance compared to using a single stage classifier, as shown in Fig. 9. We observe, however, that there is a bound on the number of stages m that results in improved performance. This is because during training, as more stages are added, when the *winsize* value increases, the resulting image obtained is almost the same as the ground truth image. The classifier is able to learn the training data images almost perfectly. The resulting LUT_i thus provides little or no information in correction of the degradation. Improved performance is guaranteed up to the optimal value of $m = 3$ beyond which the PA decreases.

Comparison to Kanungo's method

As mentioned in Sec. 2, Zheng et al. design a LUT table using a 3×3 filter window to perform restoration of degraded documents using their morphological degradation model. Their LUT is a 512×512 matrix. During training, for each 3×3 neighborhood pattern N in the degraded image, all possible occurrences of the corresponding output in the ideal image are stored. During restoration, each patch in the degraded image is replaced with the most occurring output pattern encountered during training. We applied this LUT to degraded typewritten document images and the performance was much worse, compared to our proposed algorithm. (Note that from the algorithm documentation in,[11] it is not clear how to determine the starting pixel for placing the filter window). In our study, we use real degraded typewritten document images which have degradation patterns that are more pronounced compared to the degraded images generated by their degradation models. A 3×3 filter window size is too small to learn the degradation patterns. In contrast to the Kanungo's method in which a complete window is replaced with a complete window, our approach of correcting a pixel at a time, taking into account the neighborhood pixel information, gives more accurate results. We also applied our proposed algorithm to the morphological degraded images using Kanungo's algorithm[15] and obtained finer results which suggest that our algorithm is capable of learning and correcting the degradations produced by the Kanungo model. The Kanungo degradation model is suitable for small perturbations[16] encountered during photocopying and scanning of uniform text documents but not to large degradations found in old typewritten documents.

5. CONCLUSION

We present a novel method for enhancing the degradation in historical typewritten documents using LUT ensemble classifiers that have been trained to learn the corrections of degradation patterns in the document images. Currently, our basic LUT classifier processes an entire document image in less than 1 minute using a *w-5* filter. The effectiveness of the LUT classifier can be further improved by pruning and arranging the LUT classifiers in a cascade configuration. In future work we plan to combine the effectiveness of these classifiers using more complex ensemble of cascade configurations to improve performance and exploring non-square filter window size options.

REFERENCES

1. "The diaries of Rabbi Dr. Avraham Abba Frieder." http://ir.iit.edu/collections/.
2. A. Antonacopoulos and D. Karatzas, "Semantics-based content extraction in typewritten historical documents," in *Proc. International Conference on Document Analysis and Recognition ICDAR'05*, 2005.
3. A. Antonacopoulos and D. Karatzas, "A complete approach to the conversion of typewritten historical documents for digital archives," in *Proc. IAPR International Workshop on Document Analysis Systems DAS'04*, pp. 90–101, 2004.
4. G. Agam, G. Bal, G. Frieder, and O. Frieder, "Degraded document image enhancement," in *Document Recognition and Retrieval XIV*, X. Lin and B. A. Yanikoglu, eds., *Proc. SPIE* **6500**, pp. 65000C–1 – 65000C–11, 2007.
5. A. Antonacopoulos and D. Karatzas, "Document image analysis for world war ii personal records," in *Proc.International Workshop on Document Image Analysis for Libraries DIAL'04*, 2004.
6. B. Gatos, I. Pratikakis, and S. J. Perantonis, "An adaptive binarization technique for low quality historical documents," in *Int'l Workshop Document Analysis Systems (DAS)*, pp. 102–113, 2004.
7. E. Kavallieratou and E. Stamatatos, "Improving the quality of degraded document images," in *Int'l Conf. Document Image Analysis for Libraries DIAL'06*, 2006.
8. N. Molton, X. Pan, M. Brady, A. Bowman, C. Crowther, and R. Tomlin, "Visual enhancement of incised text," *Pattern Recognition* **36**, pp. 1031–1043, April 2003.
9. S. Andra and G. Nagy, "Combining dichotomizers for map field classification," in *Proc. 18th International Conference on Pattern Recognition ICPR'06*, pp. 210–214, 2006.
10. B. As-Sadhan, Z. A. Bawab, A. E. Seed, and M. Noamany, "Comparative evaluation of different classifiers for robust distorted character recognition," in *Proc. SPIE '06*, 2006.
11. Q. Zheng and T. Kanungo, "Morphological degradation models and their use in document image restoration," in *International Conference on Image Processing*, pp. 193–196, 2001.
12. Q. Zheng and T. Kanungo, "Estimation of morphological degradation model parameters," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '01*, pp. 1961–1964, 2001.
13. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM* (45), pp. 891–923, 1998.
14. Friedman, Bentley, and Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software* **3**(3), pp. 209–226, 1977.
15. T. Kanungo, *Document Degradation Models and a Methodology for Degradation Model Validation*. PhD thesis, University of Washington, 1996.
16. H. Baird, "Document image quality: Making fine discriminations," in *Proc., Int'l Conf. on Document Analysis and Recognition*, 1999.