Investigating coupling preprocessing with shallow and deep convolutional neural networks in document image classification

Yi Liu[a,*] **Leen-Kiat Soh**[a] **and Elizabeth Lorang**[b]
[a]University of Nebraska–Lincoln, Department of Computer Science and Engineering, Lincoln, Nebraska, United States
[b]University of Nebraska–Lincoln, University Libraries, Lincoln, Nebraska, United States

Abstract. Convolutional neural networks (CNNs) are effective for image classification, and deeper CNNs are being used to improve classification performance. Indeed, as needs increase for searchability of vast printed document image collections, powerful CNNs have been used in place of conventional image processing. However, better performances of deep CNNs come at the expense of computational complexity. Are the additional training efforts required by deeper CNNs worth the improvement in performance? Or could a shallow CNN coupled with conventional image processing (e.g., binarization and consolidation) outperform deeper CNN-based solutions? We investigate performance gaps among shallow (LeNet-5, -7, and -9), deep (ResNet-18), and very deep (ResNet-152, MobileNetV2, and EfficientNet) CNNs for noisy printed document images, e.g., historical newspapers and document images in the RVL-CDIP repository. Our investigation considers two different classification tasks: (1) identifying poems in historical newspapers and (2) classifying 16 document types in document images. Empirical results show that a shallow CNN coupled with computationally inexpensive preprocessing can have a robust response with significantly reduced training samples; deep CNNs coupled with preprocessing can outperform very deep CNNs effectively and efficiently; and aggressive preprocessing is not helpful as it could remove potentially useful information in document images.
 [DOI: 10.1117/1.JEI.30.4.043024]

Keywords: convolutional neural network; document image; historical newspapers; document image analysis; document classification; poetic content classification; image denoising.

Paper 210160 received Apr. 4, 2021; accepted for publication Aug. 3, 2021; published online Aug. 25, 2021; corrected Aug. 27, 2021.

1 Introduction

Convolutional neural networks (CNNs), inspired by biological visual processes, have been popularly and successfully applied as a type of deep learning network in image-related classification approaches for generic images (e.g., hyperspectral images,[1–3] scenes,[4,5] plant images,[6,7] and graphic images[8–12]) and image-related denoising approaches (e.g., Gaussian noise,[13,14] rain effects,[15] snow effects,[16] and general frameworks[17]). Indeed, there have been significantly more results and findings of CNN-based approaches on generic images (e.g., picture-based or graphic images) than on document images, facilitated by highly competitive challenges, such as CIFAR,[18] ImageNet,[19] and MNIST,[20] that comprehensively compared both deep and shallow CNN solutions focusing on generic images. Meanwhile, there have been relatively fewer comparative studies (e.g., RVL-CDIP[21]) for document image classification, even though document image classification also has recently seen an increased use of CNN-based approaches in category classification,[22–24] layout analysis,[25–30] binarization,[31,32] text line extraction,[33–35] and optical character recognition (OCR).[36–38] Findings on the application of CNNs on generic images do not necessarily generalize to document images[39–41] as these two types of images are very

*Address all correspondence to Yi Liu, yil@cse.unl.edu

different.[42] Note that document images consist primarily of texts.[43] Also many document images are monochrome. Hence, the role of color-based visual cues, which are used in approaches for generic images, is diminished. Another property often found in document images is denser structural layouts, which make document images more susceptible to degradations.

Meanwhile, spurred by the advent of digital libraries, researchers have access to historical documents at an unprecedented scale and with unprecedented speed.[44] The increased level of accessibility inevitably has led to increased needs for searchability and other information retrieval tasks. For example, as an important category of printed historical documents, historical newspapers are a popular format and source used by librarians, social scientists, humanities researchers, genealogists, and so forth to perform investigative research.[45] Digital libraries of historic newspapers have typically been built to facilitate a limited type of investigation: keyword search and human browsing. They often do not provide article-level information for newspapers' articles, such as article types (news, poetry, advertisements, obituaries, and drawings) or access to particular types of information or content at scale—thereby artificially limiting the types of research questions that scholars might pursue. This scenario creates a challenge for collecting certain content types from millions of newspaper pages, for example. This difficulty results in researchers resorting to manual inspection for identification and classification content, which is hardly scalable across tens of millions of pages. Therefore, document classification requires accurate and fast tools for diverse documents with wide-ranging properties. Furthermore, compared with the existing amount of document image collections (e.g., Chronicling America), the datasets with labeling information, which can be used for training deep CNNs, are much less prevalent both in amount and temporal coverage. According to the study of d'Andecy et al.,[46] deep learning models require exhaustive samples to be trained well compared with incremental classification for document type classification. Deep learning could be less robust on out-of-domain samples than the incremental classification when the training sample is reduced.

This paper focuses on our investigations on the use of preprocessing to improve performance of deep learning on tasks involving printed document images. These document images may be born-digital or digitized copies of original documents. For digitized copies (e.g., document scanning), various noise effects may be present in the images;[41] these features may be more widespread when the document images have been digitized from microphotographic copies—common in many historical newspaper collections—as compared with digitized from their physical originals. For example, unevenly distributed luminance (i.e., range effects), low contrast, or visible ink from the other side of the paper (i.e., bleed-through) may be present. To deal with these noisy digitized document images, a preprocessing step (e.g., binarization or text line consolidation) to clean up the images is often required.

Although layers of CNN can perform some preprocessing or achieve the effects of preprocessing, preprocessing techniques have also been used to prepare data before feeding it into a CNN.[47] In this paper, we investigate coupling conventional preprocessing algorithms with CNNs for the following two reasons. We recognize that preprocessing has been widely used in preparing data for CNN training. However, the impact of different levels of preprocessing on CNN performance has not been investigated. For example, note that light-level preprocessing (e.g., binarization) cleans images but modifies them only minimally (e.g., removing noise), whereas aggressive-level preprocessing (e.g., consolidation) cleans and modifies images significantly to the extent of enhancing cues so that they are visually easier to recognize. First, in terms of effectiveness, which level would be more appropriate for CNN for a classification task? How would the different levels of preprocessing impact CNN performance differently? Indeed, in document images, binarization is capable of removing noise from the background pixels but could also cause disconnected strokes in the object pixels (i.e., text pixels), whereas consolidation is capable of extracting more connected layout structures but could smear text areas causing loss of textual information. Thus one investigation is determining what level of preprocessing is adequate to improve CNN's performance. Then for preprocessing techniques that can improve CNN's performance, how much is the improvement? Can preprocessing techniques make a shallower CNN effectively outperform a deeper CNN? As reported later, our investigation (Sec. 4.2) shows that aggressive-level preprocessing could degrade CNN's performance even though visual cues of the image are better enhanced because of information loss as a result

of consolidation. Our investigation in Sec. 4.3 further demonstrates that preprocessing can improve a shallower CNN to outperform or match a deeper CNN's effectiveness even though deeper CNNs are computationally more capable of handling classification tasks. Second, in terms of efficiency, it is known that, while deeper CNNs are more computationally capable of handling classification tasks, they are also expensive to train in terms of both computational cost and the requirement of training samples. Preprocessing could highlight and summarize visual cues to help CNNs train faster. Thus another investigation is determining whether and how preprocessing would help CNN to overcome a smaller training set. As reported later, our investigation in Sec. 4.4 shows that preprocessing improves CNN performance with fewer data samples. But, contrary to our findings about its impact on effectiveness, we see that preprocessing is more beneficial in the challenging classification task than in the simpler task.

The remainder of this paper is as follows. Section 2 provides an overview of related work. Section 3 describes the design of our investigation in detail. Section 4 gives the analysis of two investigations and reports on the comparative results. Section 5 concludes and presents future work.

2 Related Works

2.1 *Preprocessing*

Binarization is an image processing technique to separate the pixels of an image into background and object pixels. Otsu's method[48] is one well-known histogram-based binarization technique. It is known to be effective and was used as a baseline to evaluate binarization for document images in ICDAR's competition on document image binarization (DIBCO),[49–52] which is one of the most popular competitions in the field and has a collection of state-of-the-art algorithms for document image binarization. In Otsu's method, the between-class variance evaluates every intensity level of the histogram to find the suitable intensity as the threshold to split the background and the foreground. There have been improvements[53,54] that provide better outcomes. Liu et al.[53] proposed taking the mean or median of immediate neighbors of the intensity value into the computation of the between-class variance to make the method more robust to noise. Nina et al.[54] proposed recursively calling Otsu's method to binarize the document image. Yildirim[52] proposed smoothing the image using the Wiener filter (a smoothing operator in the image frequency domain) and enhancing the contrast and brightness quality before applying Otsu's method. Otsu's method is a histogram-based binarization approach, whereas Howe's method[55] is a state-of-the-art document image binarization in DIBCO. Howe's method is based on modeling the image to an energy function. It leverages every pixel to build the energy function and identifies the best threshold for the document image as where the energy function has the lowest value.

Furthermore, deskewing and smoothing are two important preprocessing strategies to remove noise from document images. van Beusekom et al.[56] proposed a combined skew and orientation estimation algorithm; based on geometric modeling, the algorithm gives the skewness angle and its orientation by searching for text lines within a predefined angle range. Smoothing is used to remove texturized effects in the background of the document image. He et al.[57] proposed a filter operator called a guided filter to smooth the image while preserving edges in the image.

Meanwhile, text line consolidation is based on the intuition that if a region of text lines that contains the visual cues can be recognized, all pixels from outside the recognized region can be set to the background pixel value without causing loss of visual cues. Soh et al.[58] proposed a projection-based approach to aggressively clean up the background of digitized historical newspapers. In their approach, the position and height of the text line were recognized by observing peak values in the horizontal projection histogram. They, then, for each recognized text line, set every pixel into textual (foreground) pixels to highlight the recognized region.

2.2 *Image-Based Document Image Classification*

To extract information from digitized document images, one approach is to use OCR to extract the textual content, i.e., textual characters, from the images. However, OCR struggles with noisy

document images.[44,45] In Ref. 44, for example, lexicons were used to classify recipes in digitized historical newspapers, and the performance of the classifier dropped because those relatively clean lexicons could not address or cover the various distortions in the digital texts caused by noise. Similarly, Lansdall-Welfare et al.[45] sought to identify and extract words to classify and represent major historical British events in digitized historical newspapers. However, because of noise, some of the OCRed texts were ambiguous and, thus, discarded from being used for classification, which resulted in reduced accuracy and richness of the resulting collection of words. Meanwhile, another approach to document image classification is by analyzing visual layouts without directly extracting the textual content. This approach is known as image-based document image classification.[59–62] Hu et al.[59] proposed an approach to identify five different document types (i.e., 1-column and 2-column letters, 1-column and 2-column journals, and magazine pages) using structural page layout obtained via image-based visual analysis. Shin et al.[61] and Loia and Senatore[60] leveraged layouts such as textual to non-textual content ratio, column structure, and graphic content arrangement to identify document image types. Santosh[62] leveraged user-provided feature patterns such as text area information, word count, and metadata to obtain graph models to extract similar text areas from document images.

Further, there are two types of document images that are discussed separately due to their visual differences. One deals with handwritten manuscripts, and the other one deals with printed documents such as historical newspapers. Challenges for the classification of handwritten manuscripts are very different from those of printed documents. First, character sizes typically are more consistent in printed documents compared with those in handwritten manuscripts. Second, character strokes that belong to different text lines rarely touch each other in printed documents. Third, content layouts of printed documents are typically more complicated than those of handwritten manuscripts, with compound layouts such as multiple columns on a single page and graphic figures mixed with textual contents.

Finally, some types of articles have distinctive layouts or visual cues compared with others, which make them suitable for image-based document image classification. For example, poems published in printed historical documents (e.g., newspapers) contain recognizable visual structural information (e.g., gaps between stanzas and unjustified lines).[63] As a result, some have proposed using image-based document image classification to detect poems automatically[58] by exploiting such visual cues. Harley et al.[21] built a large dataset, RVL-CDIP, for image-based document classification. Specifically, the RVL-CDIP is used to evaluate state-of-the-art document image classification and retrieval using features learned by CNNs. The RVL-CDIP consists of 40,000 grayscale document images in 16 classes with 25,000 images per class. The dataset is split into the training set, testing set, and validation set for training and evaluation of CNNs.

2.3 *Image Classification Using CNN*

Deep learning using a CNN has shown great promise in image-based classification. One of the most famous CNNs was LeNet, proposed by LeCun et al.[20] in 1998. Since then, numerous CNN models and applications have been proposed. For example, Krizhevsky et al.[64] proposed a CNN known as AlexNet (inspired by LeNet) to classify high-resolution images in ImageNet, and it drew much attention for outperforming the previous state-of-the-art by a large percentage. He et al.[65] proposed ResNet, which used a connection between the output and input to maintain the identity map of the input resolution to reduce the training difficulties caused by vanishing gradient.[66] Hu et al.[67] further proposed a new block for ResNet that combined Inception,[12] fully connected layers, and ResNet block to improve ResNet further.

CNN-based approaches have been evaluated in the domain of general images, which include both generic images and document images. In particular, studies of document images using CNNs have focused on five areas. The first area is category classification. Pondenkandath et al.[23] explored four applications for document classifications including handwriting styles, layout, font, and authorship using a residual network.[65] Jain and Wigington[22] fused visual features extracted using the CNN-based deep learning network and noisy semantic information obtained using OCR to identify document categories. Khan et al.[26] proposed a CNN-based approach to detect mismatching ink-color in hyperspectral document images for identifying forged documents. The second area is layout analysis. Chen et al.[25] proposed a CNN for historical newspaper

segmentation to distinguish text content from the background and other content types, such as figures, decoration, and comments. Kosaraju et al.[27] adopted a CNN network with a dilated convolutional kernel to analyze document layouts. Renton et al.[28] proposed a CNN-based network to segment handwritten text lines that have various issues such as slanted lines, overlapped texts, and inconsistent handwritten characters. Xu et al.[29] applied a fully CNN to perform page segmentation and extraction of semantic structures of document layouts. The third area is document binarization such as Tensmeyer and Martinez,[32] which uses a fully CNN to binarize document images. Basu et al. investigated the performances of two deep learning-based approaches for degraded document image binarization: U-Net and Pix2Pix. The fourth area is text line extraction. Grüning et al.[33] combined a CNN-based U-shape network with a bottom-up clustering method to identify text lines in historical documents with complex layouts such as curved arbitrarily oriented text lines. Mechi et al.[34] applied a CNN-based U-shape network to segment text lines and tested their solution on a challenging cBAD dataset.[68] The fifth area is OCR. Uddin et al.[36] proposed an approach to recognize Urdu ligatures by separately recognizing primary and secondary ligatures using CNNs. Zahoor et al.[37] proposed recognizing Pashto ligatures by fine-tuning pretrained AlexNext, GoogleNet, and VGGNet.

3 Methodology

In our investigations, motivated by the challenges outlined in Sec. 1, we focus on two primary research questions and two subsequent questions of the second research question. The two primary research questions are: (1) What is the performance gap among shallow, deep, and very deep CNNs on printed historical documents and document images? and (2) What combination of preprocessing and learning model is the most helpful? The second research question includes two subquestions: (2.1) Can some combination of CNN and conventional document image processing techniques outperform a CNN? and (2.2) Can preprocessing help the CNN have a better performance in a case of a small training set? These investigations involve two levels of preprocessing techniques: light-level and aggressive-level, with a total of four different techniques (smoothing, deskewing, binarization, and consolidation) that are commonly used in document image processing and a range of shallow, deep, and very deep CNN models such as LeNet,[20] ResNet,[65] MobileNetV2,[69] and EfficientNet.[70]

3.1 *Preprocessing*

We consider three preprocessing levels: no preprocessing, light, and aggressive. Preprocessing is generally necessary to clean input images, e.g., filtering out noise, in document image analysis tasks. First, at the no preprocessing level [Fig. 1(a)], we feed the original images into the CNN model without any preprocessing. Second, for the light-level [Fig. 1(b)] category, we consider preprocessing techniques that remove noise but merely distort the objective information on the

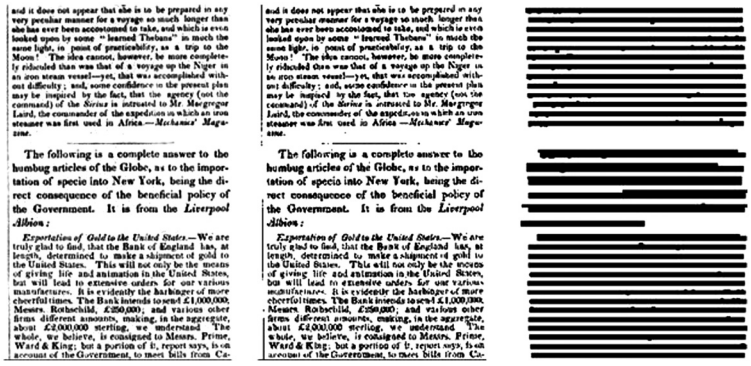


Fig. 1 Examples of three levels of preprocessing: (a) no preprocessing, (b) light level, and (c) aggressive level.

original image, such as smoothing (based on a guided filter[57]), deskewing (based on Ref. 56), and binarization (based on Otsu's method[48]). Third, at the aggressive level [Fig. 1(c)], we apply a multi-step preprocessing strategy, such as consolidation,[58] which not only removes gray level information and noise but also highlights visual structural information, such as textual line position, length, and height and masks specific textual character information (e.g., space between two neighboring letters)

3.1.1 *Light level of preprocessing*

At the light level of preprocessing, we remove noise to a certain level from background pixels, while minimizing information loss of object pixels. Smoothing, deskewing, and binarization are considered in this level of preprocessing strategies.

Smoothing reduces noise in an image using a filter. For document images, preserving edges, such as character strokes, are important. We use the guided filter,[57] which can reduce noise and suppress the gradient-reversal artifacts (i.e., false edges) while creating a good edge profile of the image. Also the guided filter is a fast non-approximate linear smoothing algorithm with a computational complexity of $O(n)$, where n is the number of pixels.

Deskewing first detects the orientation and the skewness angle of a document image. Then it corrects the skewness using the geometric transformation. We use a resolution-independent skewness detection algorithm[56] that derives orientation and skewness angles of a document image based on the text lines detected by connected components. Its computational complexity is $O(n+e)$, where n is the number of pixels and e is the number of connection directions for the connected component. In addition, we only consider the skewness of the entire document image. Hence, for one rotation centroid, the geometric transformation is a linear algorithm bound to the number of pixels, $O(n)$. Thus the computational complexity of deskewing here is $O(n+e)$.

Binarization is used to obtain object pixels from the background for further processing. For newspaper pages, histograms typically follow a bimodal distribution since, on the newspaper page, the textual pixels are darker while the background pixels are lighter. For our investigation, we use two binarization techniques based on two different underlying approaches. The first technique is Otsu's method,[48] which evaluates between-class variance for each intensity in the histogram to find the optimal threshold. A fast Otsu's binarization method[71] shows that the computational complexity is up to $O(L^2)$, where L is the number of gray-level intensities. The second technique is another state-of-the-art documentation image binarization method, namely Howe's method.[55] This method has been shown to outperform Otsu's method in DIBCO 2013.[50] Howe's method defines an energy function with tunable parameters. The optimal threshold is found when the energy function has the lowest value. Since the tuned energy function reported in the DIBCO-13 contest[50] is applied, we do not consider the computational cost of the function tuning. Hence, the computational complexity of the algorithm is $O(n)$, where n is the number of pixels.

3.1.2 *Aggressive level of preprocessing*

At the aggressive level of preprocessing, we aim to remove as much noise as possible, while preserving visual structures. Hence, we adopt the approach by Soh et al.[58] called consolidation. This preprocessing strategy segments and horizontally smears the text lines such that the overall structural characteristics of each text line are highlighted and made more pronounced. Although the specific textual information is sacrificed, the consolidation enhances the sizes and shapes of the visual structures effectively. This approach to noise removal is motivated by the intuition to enhance visual structures by filling out the holes and gaps within text lines, and, at the same time, to eliminate possible false or noisy pixels that are caused by folding, bleeding, and skewing.

The consolidation strategy, shown as Algorithm 1, has three stages. First, the consolidation binarizes the input image to roughly identify object pixels from the background using a binarization method (e.g., Otsu's method) (step 1).

Second, a projection-based text line segmentation is used to locate and segment text lines using a horizontal profile (steps 2 to 3). This stage takes the binarized image to locate potential text lines of which the values in the horizontal projection are larger than the overall average.

Algorithm 1 Adaptive projection-based text line segmentation (APB)

Input: Newspaper Page Snippet, s.

Output: Segmented Snippet, s_{rec}.

1. Adopt binarization method to binarize the input snippet, $s_b \leftarrow \text{Binarization}(s)$.
2. Compute horizontal projection histogram, $hist_{\text{ROT}}$, based on the binarized snippet, s_b.
3. Compute average textual line height and non-textual line height, $ht_t,\ ht_{nt} \leftarrow AVGHEIGHT(hist_{\text{ROT}})$.
4. For each textual line found, blk_r, in $hist_{\text{ROT}}$, where r is the first row of the line.
   4.1. If $tblk_r.height$ is bigger than ht_t:
      a. Recursive call: $APB(tblk_r)$.
   4.2. Otherwise, smear corresponding textual line in $s_{\text{rec}} \leftarrow SMEAR(s_b, r, tblk_r.height)$.
5. For each non-textual line found, $ntblk_r$, in $hist_{\text{ROT}}$, where r is the first row of the line.
   5.1 If $ntblk_r.height$ is bigger than ht_{nt}:
      a. Recursive call: APB ($ntblk_r$).

End of Algorithm

In addition, each text line found occupying a large structural area triggers a recursive process (step 4) to break down the large area further to attempt to find potentially misrecognized text lines within the area.

During the third stage, the consolidation horizontally smears each resultant text line from the second stage into a solid rectangle (step 4.2) and, correspondingly, the non-textual lines as well (step 5). By this process, individual symbolic characteristics of the textual content are removed completely as the smearing process fills out the holes and gaps among symbolic characters to produce larger, contiguous visual structures. We can compute the time complexity of APB as follows.

First, using Otsu's method as an example, the time complexity of binarization in step 1 in APB is $O(L^2)$, where L is the number of gray-level intensities. Second, for step 2, the computation step for the horizontal projection histogram traverses each pixel to count the number of textual pixels for each row. Thus the time complexity bounds to the number of pixels, which is $O(n)$. Third, step 3 traverses the horizontal histogram row by row to discover both textual and non-textual lines and, at the same time, to compute the average height. So the time complexity is $O(r)$, where r is the number of rows in the image. For step 4 we compute the time complexity for SMEAR first. The SMEAR operation evaluates a window of pixels for each column to find the beginning and the end of the textual lines, and the size of the window bounds to the height of each corresponding textual line. Note that, in the worst-case scenario, the height could be the number of rows r. Hence, SMEAR processes r^2c pixels, where c is the number of columns. Since $r \times c = n$, the time complexity for SMEAR is $O(rn)$. Therefore, the number of gray-level intensities L is a constant number. Without any recursive call, the time complexity of the algorithm is

$$O(L^2) + O(n) + O(r) + O(rn) \approx O(rn).$$

However, the time complexity of APB with the recursive call could become exponential. Hence, we limit the recursion depth of APB to seek a computationally cheaper solution. By comparing APB results with different recursion depth limits, we find that limiting the recursion depth to one could make APB more efficient while maintaining consolidation outcomes that are as good or better. Figure 2 shows one typical example of the comparison of APB results with limiting the recursion depth to one, two, and three levels. The comparison shows that APB with recursion limiting to depth one performs well on addressing the textual line missing problem

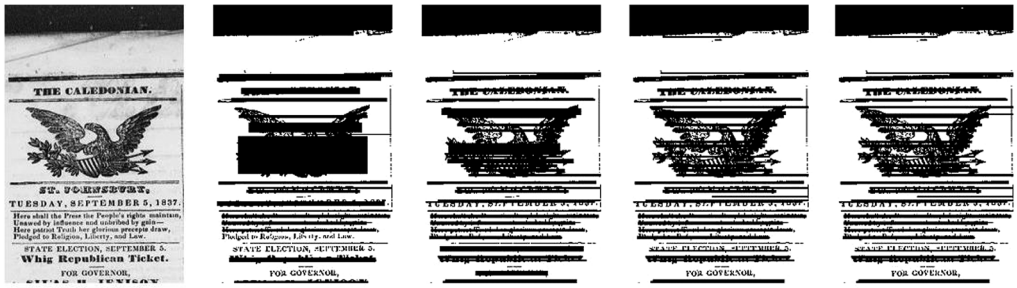


Fig. 2 Different recursion depth examples: (a) original snippet; (b) consolidation result when depth = 0; (c) consolidation result when depth = 1; (d) consolidation result when depth = 2; and (e) consolidation result when depth = 3.

[e.g., the missed textual lines in the bottom of Fig. 2(b) were covered in Fig. 2(c)]. And, even for some cases in the example, APB with recursion limiting to level 1 depth could outperform APB with recursion limiting to level 2 and level 3 depths [e.g., the covered "subtitle" in the bottom of Fig. 2(c) was missed in both Figs. 2(d) and 2(e)]. Finally, it follows that by limiting the recursion depth to one, the time complexity of APB is

$$O((rn)^2).$$

3.1.3 *Comparison between preprocessing strategies*

To provide further context for our investigations, here we provide a comparison between the preprocessing techniques: (1) no-preprocessing (no), (2) light preprocessing, binarization using Otsu[48] (light-Otsu), (3) aggressive preprocessing, consolidation based on Otsu (aggressive-Otsu), (4) light preprocessing, binarization using Howe[55] (light-Howe), and (5) aggressive preprocessing, consolidation based on Howe (aggressive-Howe).

Figures 3–6 show the results of applying the five preprocessing strategies to the four image snippets and another four challenging image snippets. We observe that preprocessing reduces the corresponding noise effects and enhances the object pixels visually, as shown in Figs. 3 and 4, with better contrast [e.g., Figs. 3(b)–3(d), and 4(c)], reduced range effect [e.g., Fig. 4(a)], removal of bleed-through pixels [e.g., Fig. 4(b)], and enhanced, more connected textlines [e.g., Fig. 4(b)]. However, we also observe that individual textual characteristics are not retained after consolidation.

In Figs. 5 and 6, we see that both Otsu-based and Howe-based approaches are effective in binarization and have their strengths and weaknesses. Howe-based preprocessing addressed the range effect more effectively than Otsu-based preprocessing [e.g., comparing Figs. 6(a) and 4(a)] and reduced the blobs more significantly than Otsu-based preprocessing [e.g., comparing Figs. 6(d) and 4(d)]. On the other hand, Otsu-based preprocessing introduced fewer artifacts to the images than Howe-based preprocessing [e.g., consider the vertical "line" artifacts on the left side of the image snippets found in rows (b) and (d) in Fig. 5] and produced thinner, and thus more precise, lines than Howe-based preprocessing [e.g., comparing Figs. 3(b) and 3(d)].

We also compare the different preprocessing strategies' performance in terms of the computational time that each strategy took to preprocess images. Specifically, as shown in Table 1, we report the total execution time that the preprocessing took to preprocess all images (16,928 snippets) in the dataset. And the execution runs on an eight-core processor, AMD Ryzen 7 5800X. The computational time shows that the computational cost of the preprocessing strategy is much lower than the training time (see more details in Sec. 4.1).

3.2 *CNN Model Architectures*

The CNN represents the state-of-the-art machine intelligence method for deep learning. In a CNN, briefly, there are several types of layers, with a layer being a network of neural nodes

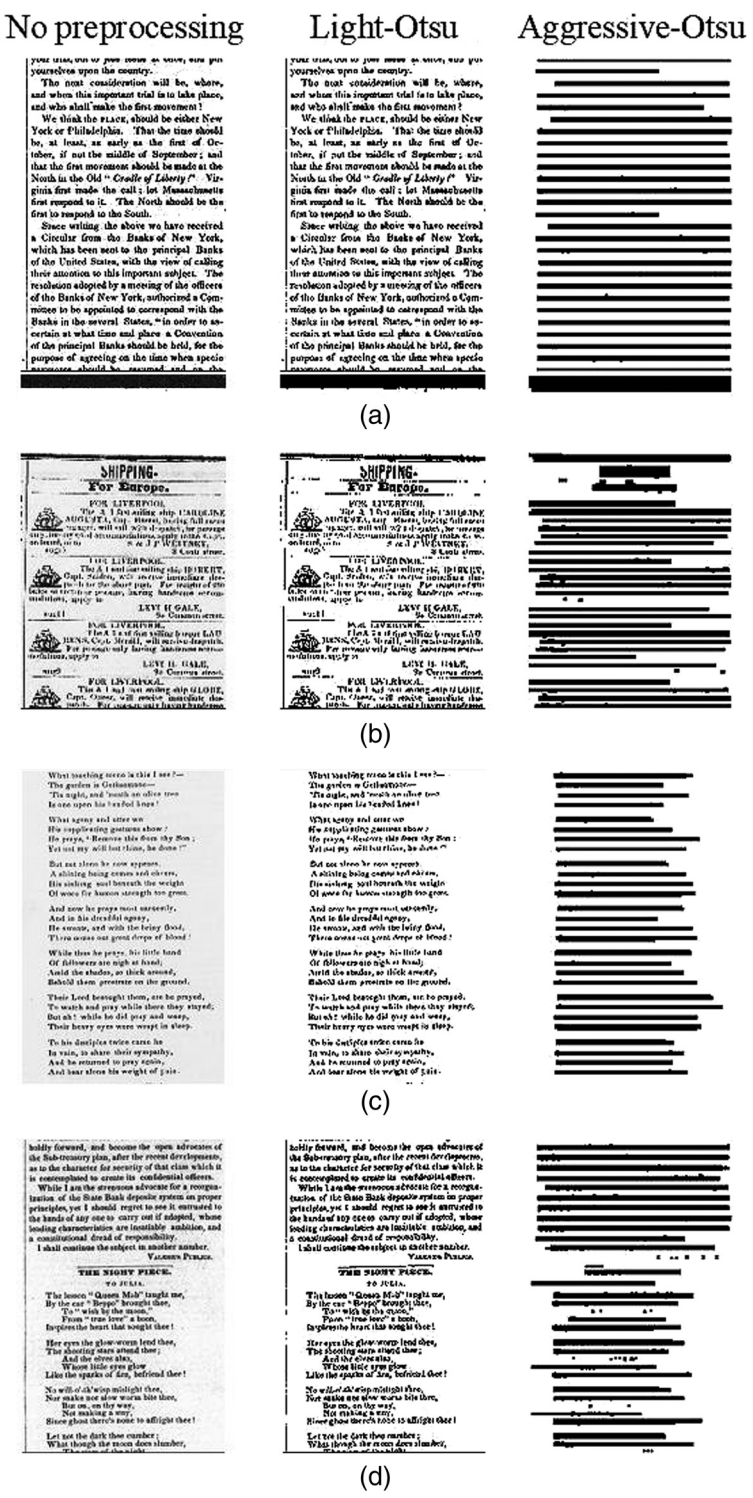


Fig. 3 (a)–(d) Results of the two preprocessing strategies: light-Otsu (middle column) and aggressive-Otsu (right column) compared with no-preprocessing (left column) applied to the image snippets.

such that each node receives signals from the nodes in the previous layer and then generates a signal for some nodes in the next layer. In particular, there are convolutional layers, pooling layers, fully connected dense layers, and output layers. A convolutional layer's purpose is taking a matrix of the image or a feature map from the previous layer to compute a convolution product to represent a feature at a certain level using a kernel. A pooling layer's purpose is to reduce the

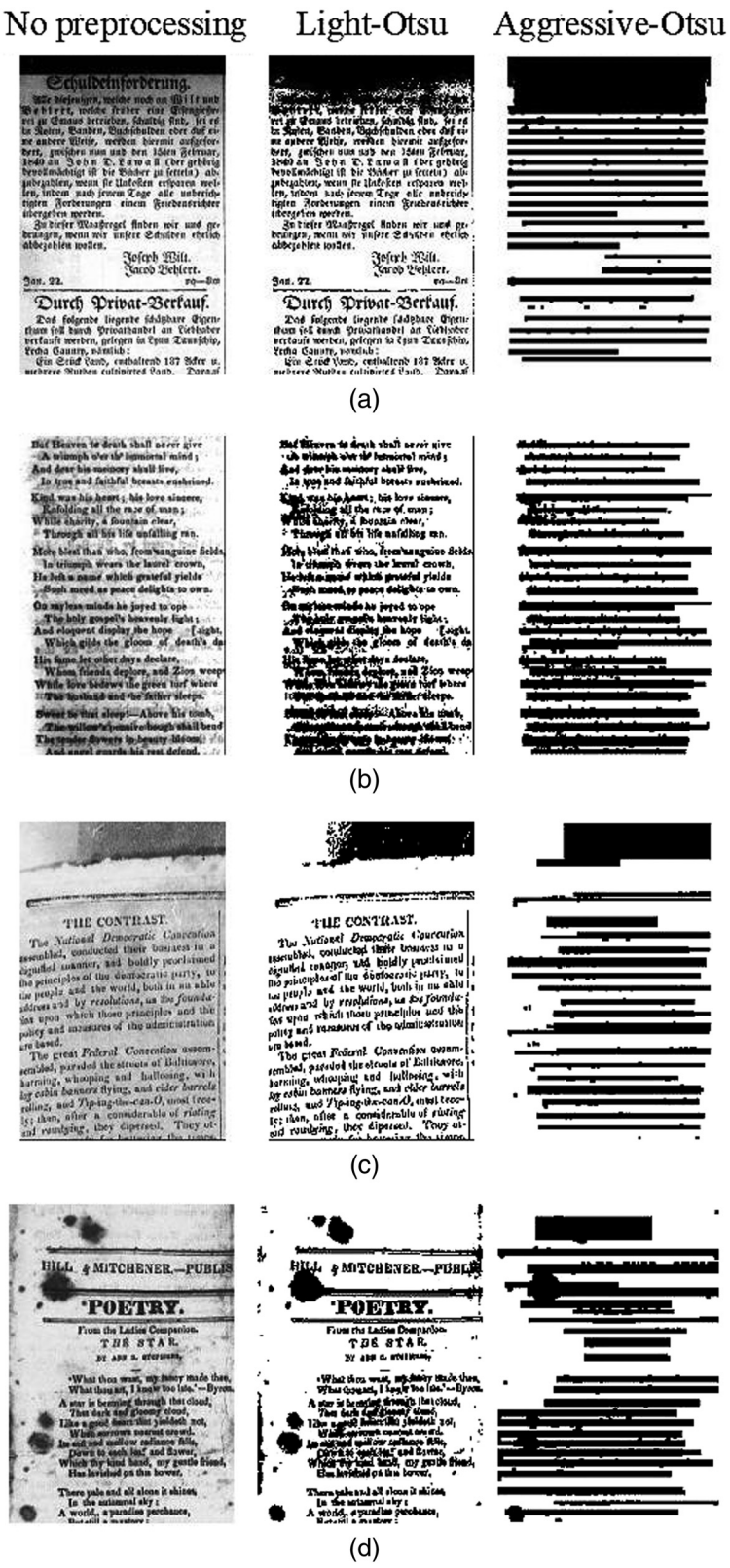


Fig. 4 (a)–(d) Results of the two preprocessing strategies: light-Otsu (middle column) and aggressive-Otsu (right column) compared with no-preprocessing (left column) applied to the challenging image snippets.

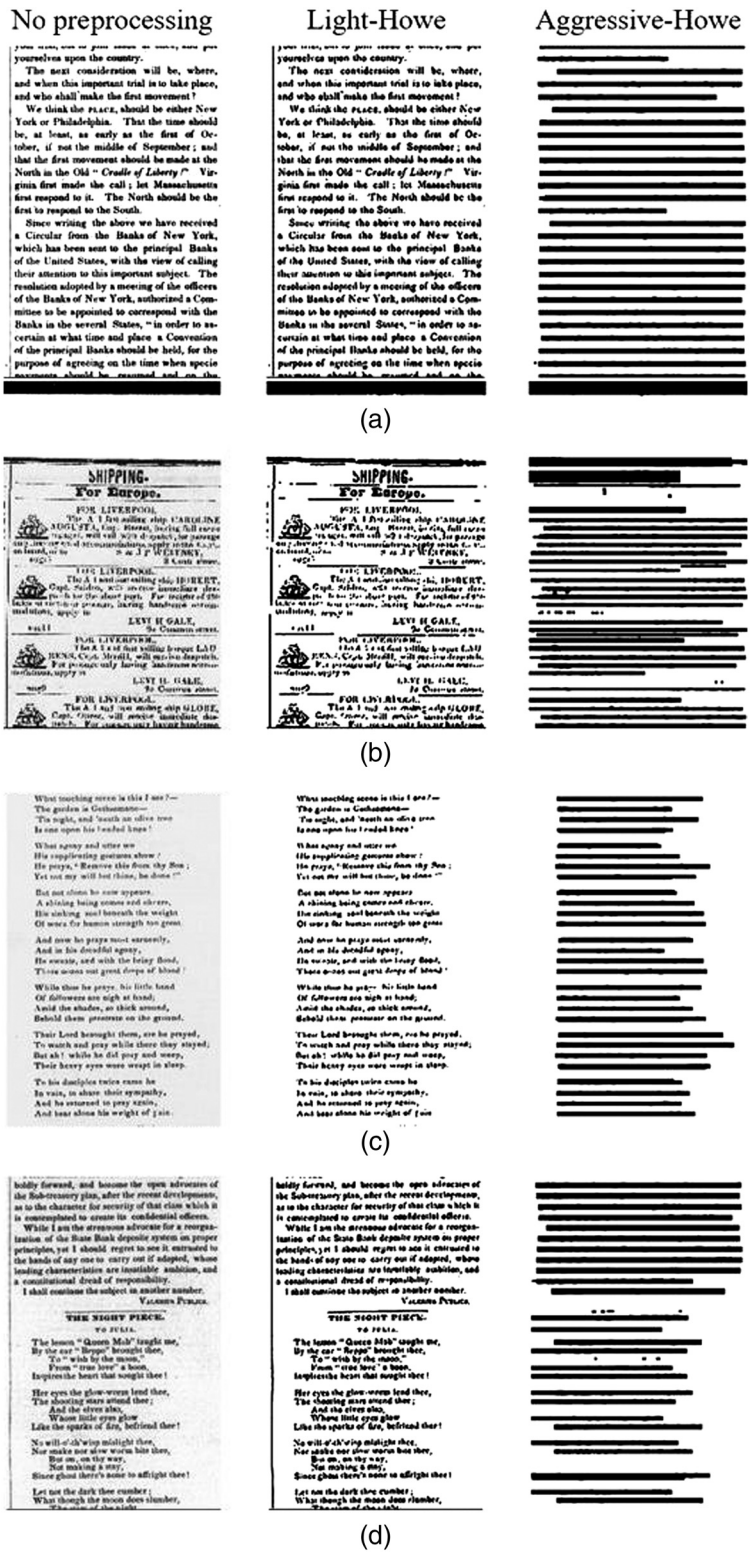


Fig. 5 (a)–(d) Results of the two preprocessing strategies: light-Howe (middle column) and aggressive-Howe (right column) compared with no-preprocessing (left column) applied to the image snippets.

spatial size of representations to reduce the computational load in the network. A fully connected dense layer's purpose is to allow the network to map high-dimensional results of the convolutional layers to a flat (one-dimension) vector layer to prepare for the final classification using softmax. Dropout between the fully connected dense layer and output layer is a regulation

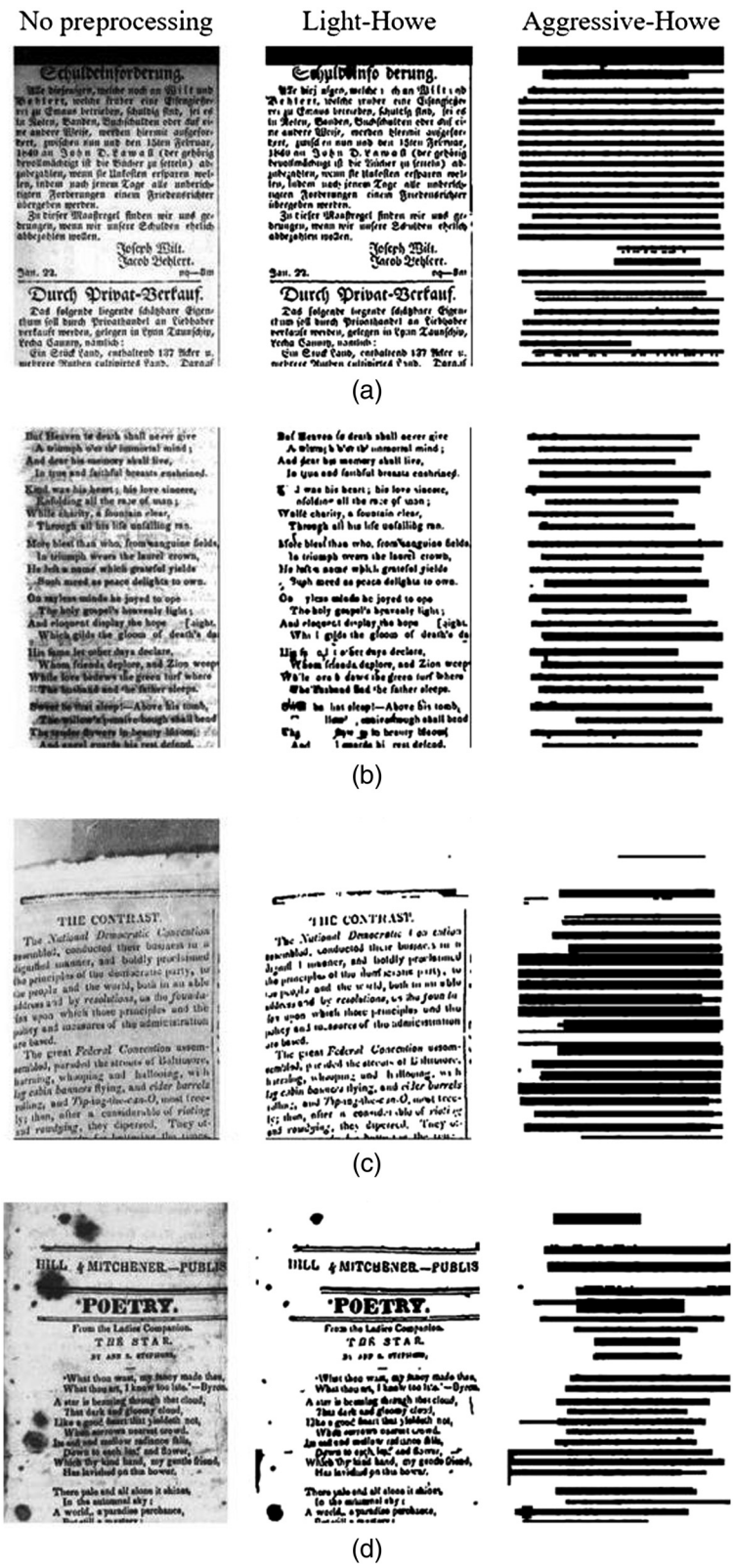


Fig. 6 (a)–(d) Results of the two preprocessing strategies: light-Howe (middle column) and aggressive-Howe (right column) compared with no-preprocessing (left column) applied to the challenging image snippets.

Table 1 Computational time different preprocessing strategy categories.

Preprocessing strategy	Total computational time (s)
Light-Otsu	6
Light-Howe	128
Aggressive-Otsu	20
Aggressive-Howe	142

technique that has been used to reduce overfitting issues.[72] An output layer, known as a one-hot vector, presents the classification result using a one-by-*N* vector, with each element in the vector representing a specific label.

When designing a CNN, one is concerned about the number of layers, the depth of the CNN. According to feature visualization,[47] the deeper the layer is, the more comprehensive the feature that it can capture is. Also trends at the ImageNet competition[19] have shown that a deeper CNN can have a better classification performance than a shallower one. However, as alluded to in Sec. 1, printed document images differ from the generic images used in the ImageNet competition in terms of monochrome color, structurally dense layout, and unique type of noise (bleed-through), such that document image classification could be sensitive to the depth of CNN differently, compared with the generic image classification. Based on the number of trainable layers, which contain trainable parameters, we divide CNN models into three categories, shown in Table 2: (1) a shallow CNN model has fewer than 10 trainable layers, (2) a deep CNN model has more than 10 but fewer than 100 trainable layers, and (3) a very deep model has more than 100 trainable layers. In this paper, we consider several architectures that fall under the three general CNN models: (1) shallow: LeNet[20] and its variants (LeNet-5, LeNet-7, and LeNet-9), (2) deep: a ResNet[65] variant (ResNet-18), and (3) very deep: ResNet-152, MobileNet,[69] and EfficientNet.[70]

LeNet was first presented by LeCun et al. to classify handwritten digits. It is a shallow CNN that performed very well with a 0.9% error rate on the MNIST dataset.[73] The originally proposed model (LeNet-5) has two pairs of convolutional-pooling layers following by a dense layer, as shown in Fig. 7. Inspired by the work of Zeiler and Fergus,[47] we see that, in LeNet, each convolutional-pooling layer is a functional block to identify the certain level of feature and that each added convolutional-pooling layer can potentially increase LeNet's classification capability. Hence, we also build deeper models based on the original LeNet-5, namely, LeNet-7 and LeNet-9, by adding convolutional-pooling layers. LeNet-7, shown in Fig. 8, has an additional pair of the convolutional-pooling layer, and LeNet-9, shown in Fig. 9, has two additional pairs of the convolutional-pooling layer. In addition, the LeNet design inspired the AlexNet, another shallow CNN that won the ImageNet challenge in 2012[19] with 15.3% of the top-5 error rate. Hence, similar to the AlexNet, LeNet would have poorer performance than the deep model, ResNet, for generic images.

Table 2 Number of layers containing trainable parameters.

Category	Model	# layers
Shallow	LeNet-5	4
Shallow	LeNet-7	5
Shallow	LeNet-9	6
Deep	ResNet-18	43
Very deep	ResNet-152	311
Very deep	MobileNetV2	105
Very deep	EfficientNet	131

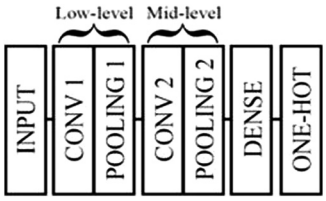


Fig. 7 LeNet-5 CNN architecture.

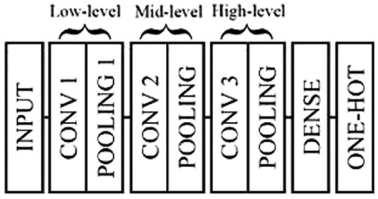


Fig. 8 LeNet-7 CNN architecture.

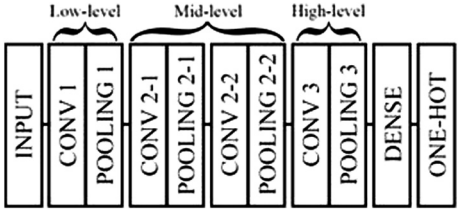


Fig. 9 LeNet-9 CNN architecture.

ResNet is a deep CNN model that won the ImageNet challenge in 2015[74] with 78.25% top 1/93.95% top 5 accuracy. Note that, because of the issue of vanishing gradient,[66] it is not possible to stack LeNet much deeper. As a result, to compare the deep CNN model, we apply ResNet in our investigations. As alluded to earlier, ResNet was proposed by He et al.[65] for the ImageNet Competition. It provided a solution for solving the vanishing gradient in a very deep CNN model. The design of ResNet included a base building block. Here we apply the original design. For ResNet-18, the building block is two 3×3 convolutional layers, and for ResNet-152, the building block is consecutive 1×1, 3×3, and 1×1 convolutional layers, known as the bottleneck block.

MobileNetV2[69] is a very deep CNN model that is designed to significantly reduce the architecture's demand for computing resources. It factorizes the standard convolutional layer into combinations of channel-wise convolution and point-wise convolution to trade-off between latency and accuracy. By factorizing, the size of the latency is smaller, allowing for an efficient convolutional computation, but the connection between channels is weakened, lowering the accuracy.

EfficientNet,[70] a very deep CNN model, leverages the tensor shapes of each functional block to find the best combination to scale up the convolutional networks based on MobileNetV2.[69] It formulates the baseline CNN model (i.e., MobileNetV2) with three factors: depth, width, and resolution. Using the formulation to maximize accuracy and minimize computing resources at the same time, EfficientNet finds the best factor combination to scale up the network. As a very deep CNN, EfficientNet achieves 84.4% top 1/97.1% top 5 accuracy.

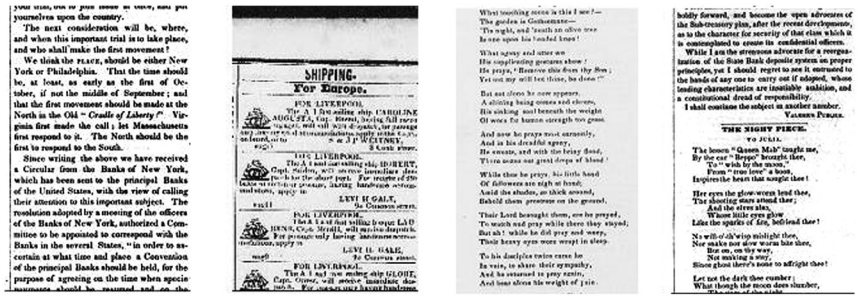
4 Investigations and Results

As alluded to in Sec. 1, we investigate printed historical documents due to several reasons. First, historical documents have been increasingly digitized and archived, which leads to increasing demand for enhanced searchability in digital libraries. Second, their unique layout structures are

different from generic images, especially in terms of compactness, and yet are rather well suited for image-based classification.[59–61] Third, digitized historical documents are noisy due to a range of duplication processes (microphotography and digitization) over time, to material degradation or other damage over time, and to qualities of their original paper form.

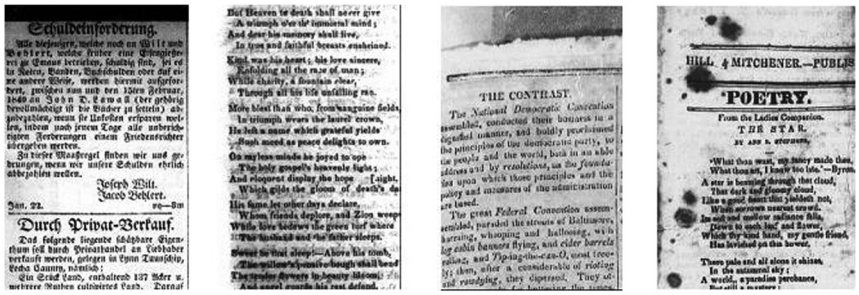
In this section, we present four sets of investigations in response to the two primary research questions posed in Sec. 3. To gain more generalizable insights into the investigations, we use two classification tasks: (1) a binary poem classification task in which a CNN is trained to determine whether a document image snippet is a poem or not using the Aida-17k[75] dataset and (2) 16-class document type classification[21] in which a CNN is trained to label document images into 16 different classes, [16 document image classes are: (1) letter, (2) memo, (3) email, (4) file-folder, (5) form, (6) handwritten, (7) invoice, (8) advertisement, (9) budget, (10) news article, (11) presentation, (12) scientific publication, (13) questionnaire, (14) resume, (15) scientific report, and (16) specification.] using the RVL-CDIP[21] dataset. Note that the second task is a more complex classification task than the first one. These two datasets represent a wide range of problems or issues that a document classification task could encounter.

Aida-17k consists of 16,928 image snippets extracted from hundreds of historical newspaper pages from the Chronicling America repository between the years 1836 and 1840. The dataset is balanced: half of the snippets have poems (true), and half do not (false) (see Fig. 10 for examples of the snippets). In other words, there are two classes with 8464 image snippets per class. Each snippet has the same width-to-height ratio of 2:3. However, the actual dimensions of the images can be different due to the various levels of resolution found in the newspaper pages. Considering both constraints above, the input image is sized to 128×192 pixels for batched training, and thus we scaled each image to those dimensions prior to feeding each into the CNNs. The challenging aspect of this task comes from the profound noise effects on the images in the dataset as it has various noise types (Fig. 11) and a wide range of severity in noise effects (Figs. 12 and 13). Finally, for our investigations involving Aida-17k, the 10-fold cross-validation approach is used; each 10% subset of the dataset is excluded from the training process but is used to obtain the



(a) (b) (c) (d)

Fig. 10 Examples of historical newspaper page snippets: (a), (b) snippets that do not contain poems and (c), (d) snippets that contain poems.



(a) (b) (c) (d)

Fig. 11 Examples of noise effects: (a) range effects, (b) bleed-through, (c) skewed orientation, and (d) blobs.

Fig. 12 Data examples that contain a poem with a wide range of noise: from very clean to very noisy.

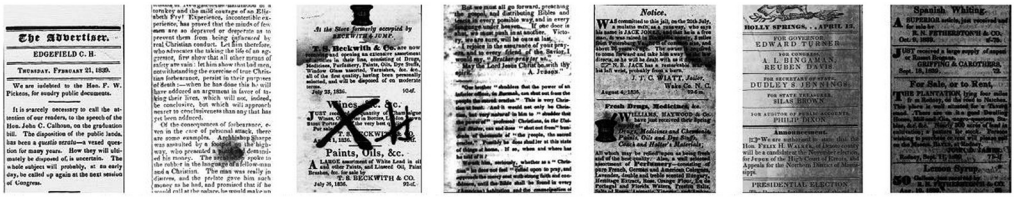


Fig. 13 Data examples that do not contain a poem with a wide range of noise: from very clean to very noisy.

testing accuracy. All of the results reported later in this section are computed from the average of the 10 rounds of training and testing. Also note that, for each training, we use the result from the best epoch of training that has the highest testing $F1$-score (i.e., the harmonic mean of precision and recall).

RVL-CDIP consists of 16 classes with 25,000 images per class. This dataset has different types of document images ranging from printed documents to handwritten manuscripts and from mostly text-based images to mostly graphic-based images. Among these images, there are 320,000 images in the training set, 40,000 images in the validation set, and 40,000 images in the testing set. The images are sized so that the heights of the images do not exceed 1000 pixels, whereas the widths of the images are not limited. The actual dimensions (width–height pair) of the images are different due to the width-to-height ratio varying. Hence, for batched training, we resized the images to 384×256 prior to feeding each into the CNNs.

Table 3 summarizes the four investigations regarding the two research questions using the above datasets in the two classification tasks.

4.1 *Investigating Performance Gap among Shallow, Deep, and Very Deep CNNs*

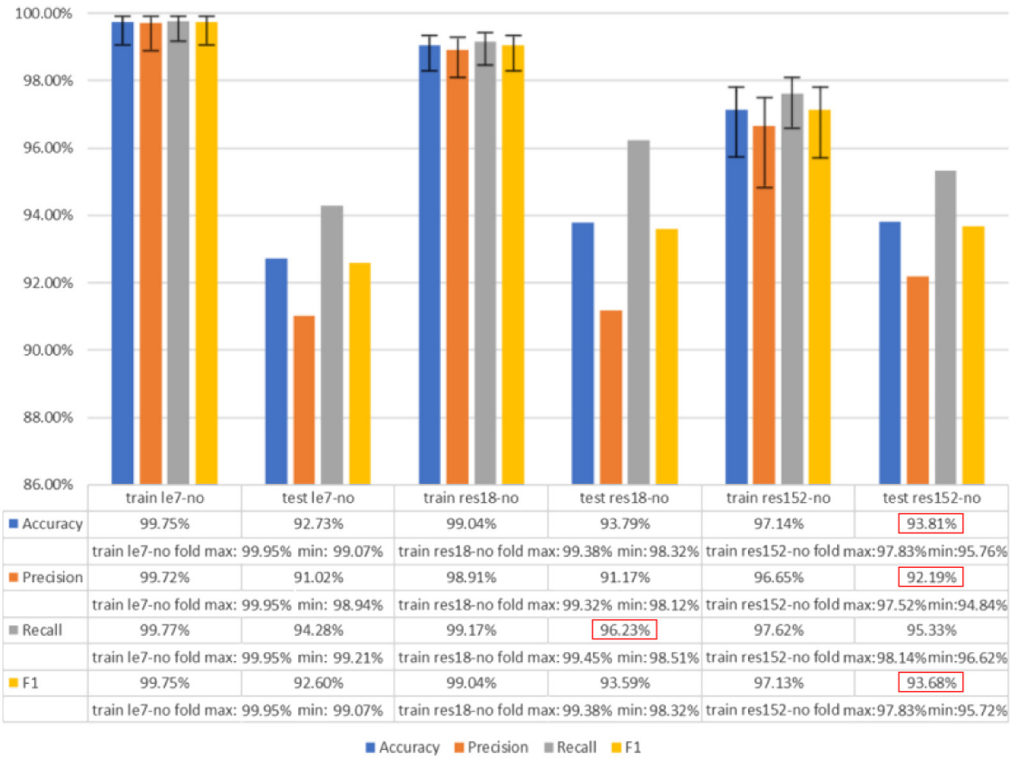
In this investigation, we compare the performance of CNN models with different depth configurations on the two classification tasks to establish a baseline effectiveness of such models. A gap is defined as the performance difference between two CNNs in terms of accuracy, precision, recall, and $F1$-score.

4.1.1 *Task 1: binary poem classification*

The CNN models used in this investigation are shallow: LeNet-7 (le7), deep: ResNet-18 (res18), and very deep: ResNet-152 (res152). In Fig. 14, we show the average, maximum, and minimum training folds performance. We noticed that Res-Net-152 had the lowest scores in training. To make sure ResNet-152 was properly trained, in our subsequent investigation, we found that, despite the lower training scores, ResNet-152 was fully trained since all fold tests of ResNet-152 reached its best testing performance at an average of 110 epochs while every training lasted 150 epochs. Thus training in Fig. 14 was valid. It also shows that, in terms of test accuracy, precision, and $F1$-score, ResNet-152 performed the best. However, ResNet-18 resulted in a better recall score, and accuracy, precision, $F1$-score of ResNet-18 is only lower than

Table 3 Summary of four sets of investigations (Sec. 4).

Section	Comparison	CNN model	Classification task	Preprocessing techniques	Datasets used	Research question
4.1	*Baseline investigation*: comparing performance between shallow, deep, and very deep CNN models in two classification tasks with different levels of difficulty	LeNet-7 LeNet-9 ResNet-18 ResNet-152 MobileNetV2 EfficientNet	– Binary poem classification – 16-Class document type classification	None	Aida-17k RVL-CDIP	1
4.2	*Different preprocessing levels investigation*: comparing performance of different levels of preprocessing techniques when coupled with CNN models of different depths	LeNet-7 ResNet-18 ResNet-152	– Binary poem classification	Light-level: – Binarization Aggressive-level: – Consolidation	Aida-17k	2
4.3	*Different task difficulty levels investigation*: comparing performance of CNN models of different depths coupled with light-level preprocessing in two classification tasks with different levels of difficulty	LeNet-5 LeNet-7 LeNet-9 ResNet-18 ResNet-152 MobileNetV2 EfficientNet	– Binary poem classification – 16-Class document type classification	Light-level: – Smoothing – Deskewing – Binarization	Aida-17k RVL-CDIP	2.1
4.4	*Smaller training set investigation*: comparing performance of CNN models of different depths coupled with preprocessing techniques with different percentages of training set in two classification tasks with different levels of difficulty	LeNet-7 LeNet-9 ResNet-18	– Binary poem classification – 16-Class document type classification	Light-level: – Smoothing – Deskewing – Binarization	Aida-17k RVL-CDIP	2.2



	train le7-no	test le7-no	train res18-no	test res18-no	train res152-no	test res152-no
Accuracy	99.75%	92.73%	99.04%	93.79%	97.14%	93.81%
	train le7-no fold max: 99.95% min: 99.07%		train res18-no fold max: 99.38% min: 98.32%		train res152-no fold max: 97.83% min: 95.76%	
Precision	99.72%	91.02%	98.91%	91.17%	96.65%	92.19%
	train le7-no fold max: 99.95% min: 98.94%		train res18-no fold max: 99.32% min: 98.12%		train res152-no fold max: 97.52% min: 94.84%	
Recall	99.77%	94.28%	99.17%	96.23%	97.62%	95.33%
	train le7-no fold max: 99.95% min: 99.21%		train res18-no fold max: 99.45% min: 98.51%		train res152-no fold max: 98.14% min: 96.62%	
F1	99.75%	92.60%	99.04%	93.59%	97.13%	93.68%
	train le7-no fold max: 99.95% min: 99.07%		train res18-no fold max: 99.38% min: 98.32%		train res152-no fold max: 97.83% min: 95.72%	

Fig. 14 Classification results of 10-fold cross validation using Aida-17k comparing LeNet-7 without preprocessing strategy (le7-no); ResNet-18 without preprocessing strategy (res18-no); and ResNet-152 without preprocessing strategy (res152-no).

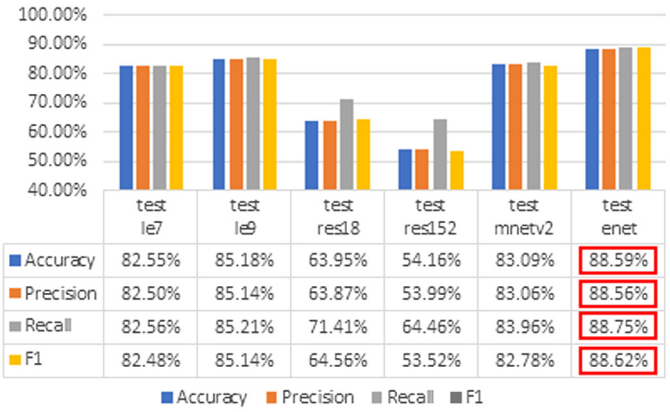
Table 4 Training time of LeNet-7, ResNet-18, and ResNet-152.

Model	Training time (h)
LeNet-7	35
ResNet-18	46
ResNet-152	94

ResNet-152 by <0.1%. Further, the shallow (LeNet-7) and deep (ResNet-18) models also resulted in very similar performances. Specifically, except for the recall score, all other score differences were lower than 1% between LeNet-7 and ResNet-18. On the other hand, when compared in terms of training time, shown in Table 4, the training of ResNet-152 took 94 h and was much longer than the training hours needed for ResNet-18 (46 h) and LeNet-7 (35 h). Thus the performance gap between shallower and deeper CNNs for the investigated task may not be large. Further, the marginal benefit to increasing the depth of the CNN model to improve the performance is very low considering the additional computational cost.

4.1.2 *Task 2: 16-class document type classification*

Due to the task's complexity, we use an expanded set of CNN models in this task, which are shallow [i.e., LeNet-7 (le7) and LeNet-9 (le9)], deep [i.e., ResNet-18 (res18)], and very deep [i.e., ResNet-152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet)]. Figure 15 shows that, overall, EfficientNet, a very deep CNN model, outperformed all other models, whereas LeNet-9, a shallow CNN model, finished as second best. However, neither ResNet-18 (deep) and ResNet-152 (very deep) performed well. In our further analysis, we found that, while the training of ResNet-18 and ResNet 152 were converging, they suffered from a significantly



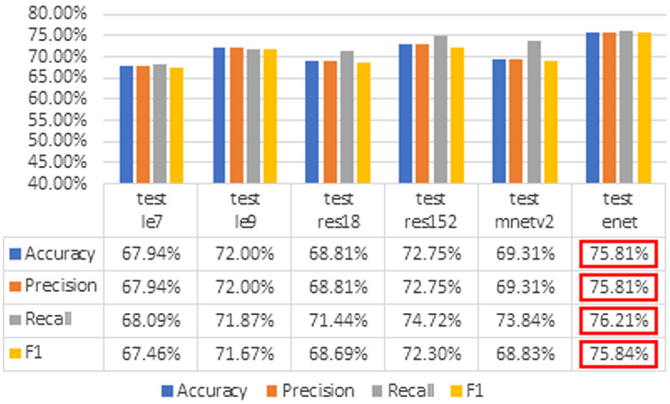
	test le7	test le9	test res18	test res152	test mnetv2	test enet
Accuracy	82.55%	85.18%	63.95%	54.16%	83.09%	88.59%
Precision	82.50%	85.14%	63.87%	53.99%	83.06%	88.56%
Recall	82.56%	85.21%	71.41%	64.46%	83.96%	88.75%
F1	82.48%	85.14%	64.56%	53.52%	82.78%	88.62%

Fig. 15 Test results of 16-class document type classification using RVL-CDIP comparing LeNet-7 (le7), LeNet-9 (le9), ResNet-18 (res18), ResNet-152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet).

slower gradient descent issue (i.e., with a cross-entropy loss of between 1 and 2.7 throughout the training epochs, compared with that of between 0 and 0.5 for the other models). Note that all models used the Adam optimization algorithm[76] for stochastic gradient descent in the training process. The algorithm is known to be able to adaptively estimate the learning rate and momentum for the training parameters to obtain optimized training. Thus the issue showed that deeper CNNs, such as ResNet-18 and ResNet-152, could be significantly challenging to be trained, even when an advanced hyper-parameter optimization algorithm is applied. Extending the training time may allow ResNet-18 and ResNet-152 to achieve high performance under slow gradient descent. However, the LeNet-9 counterpart had easily outperformed ResNet-18 and ResNet-152, suggesting that the additional computational cost may not be worth it.

4.1.3 *Task 2 variant*

To better understand the results from tasks 1 and 2 above, we derive a subset of low-quality 19,200 images, i.e., 1200 images for each of the 16 classes, namely RVL-CDIP-balanced, from the original RVL-CDIP dataset. Being low quality, these images have (1) an intensity range similar to those of the Aida-17k image, (2) low contrast, (3) high background noise, and (4) high global skewness. As shown in Fig. 16, similar to the full RVL-CDIP task (task 2), EfficientNet performed the best, outperforming ResNet-152 and LeNet-9 each by more than 3%. LeNet-9 and ResNet-152 performed very similarly: ResNet-152 outperformed LeNet-9 by <1% in accuracy, precision, and $F1$ scores. Note that, among these three CNNs, ResNet-152 is the deepest with



	test le7	test le9	test res18	test res152	test mnetv2	test enet
Accuracy	67.94%	72.00%	68.81%	72.75%	69.31%	75.81%
Precision	67.94%	72.00%	68.81%	72.75%	69.31%	75.81%
Recall	68.09%	71.87%	71.44%	74.72%	73.84%	76.21%
F1	67.46%	71.67%	68.69%	72.30%	68.83%	75.84%

Fig. 16 Test results of 16-class document type classification using RVL-CDIP-balanced comparing LeNet-7 (le7), LeNet-9 (le9), ResNet-18 (res18), ResNet-152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet).

311 layers, EfficientNet has only 131 layers, and LeNet-9 has only 6 layers. A 3% performance difference between EfficientNet and LeNet-9 is larger for this more challenging document classification task than for the less challenging tasks (first task: binary poem classification).

This investigation serves as a baseline. While confirming that the performance gap between shallower and deeper CNN models generally increases with the difficulty of classification task, we also find that the performance gap between shallow, deep, and very deep CNN models could be very small, such as <1% in accuracy, precision, recall, and $F1$-score in the simpler binary poem classification task. Interestingly, we also observe that a shallow CNN (LeNet-9) outperformed a very deep CNN (MobileNet) in terms of accuracy, precision, and $F1$ scores in the more challenging 16-class document type classification task. These findings demonstrate the viability of shallower CNN models matching the performance of deeper ones in classification tasks.

4.2 *Investigating Different Levels of Preprocessing*

In this investigation, we compare different combinations of CNN models coupled with preprocessing to study the effects of three preprocessing levels—no-preprocessing, light-preprocessing, and aggressive-preprocessing—on the performance of CNN models. The rationale behind this investigation is as follows. Intuitively, a deeper network tends to learn objects better since more detailed features could be encoded by the model,[47] but at the same time, the network is computationally more expensive to train. Therefore, we investigate shallower and deeper CNNs to explore the possibility of a coupling of conventional image processing and a CNN that could outperform a deeper CNN alone.

4.2.1 *Task 1: binary poem classification*

There are three CNN models coupled with the preprocessing strategies in this task: (1) shallow: LeNet-7, deep: ResNet-18, and very deep: ResNet-152. There are three CNN models in this task: (1) shallow: LeNet-7, deep: ResNet-18, and very deep: ResNet-152. Table 5 shows that, in terms

Table 5 Test results on Aida-17k with different preprocessing strategy categories.

Model	Process level	Accuracy (%)	Precision (%)	Recall (%)	$F1$ (%)
LeNet-7	No	**92.52**	**91.20**	93.69	**92.41**
	Light-Otsu	92.22	90.50	93.75	92.08
	Aggressive-Otsu	87.10	85.94	88.02	86.95
	Light-Howe	92.44	90.54	**94.17**	92.28
	Aggressive-Otsu	89.78	87.88	91.40	89.57
ResNet-18	No	92.86	**91.19**	94.39	92.72
	Light-Otsu	***93.09***	91.10	**94.91**	***92.93***
	Aggressive-Otsu	88.92	87.30	90.31	88.71
	Light-Howe	92.86	90.95	94.60	92.71
	Aggressive-Otsu	91.22	89.77	92.51	91.06
ResNet-152	No	92.49	90.02	94.75	92.28
	Light-Otsu	92.71	90.26	***95.04***	92.49
	Aggressive-Otsu	86.91	83.42	89.80	86.41
	Light-Howe	**92.78**	***92.65***	93.06	**92.77**
	Aggressive-Howe	90.97	90.71	91.29	90.93

of test accuracy and $F1$-score, ResNet-18 with the light-Otsu strategy outperformed all other approaches. Note also that both ResNet-18 with light-Otsu and ResNet-152 with light-Howe outperformed their counterparts without preprocessing. Thus we see that preprocessing can improve the performance of CNNs in the poem classification task. Moreover, aggressive preprocessing resulted in worse performance than the no- and light-counterpart (i.e., light-Otsu versus aggressive-Otsu, and light-Howe versus aggressive-Howe). This is likely because an aggressive preprocessing such as the aforementioned consolidation can overprocess an image causing information loss to the object pixels. Furthermore, we also see that ResNet-18 with light preprocessing outperformed ResNet-152 with no preprocessing. This is insightful. A deep CNN model, with the implications of being more efficient to train, can outperform a much deeper CNN model by just incorporating some light-level, computationally inexpensive image processing techniques.

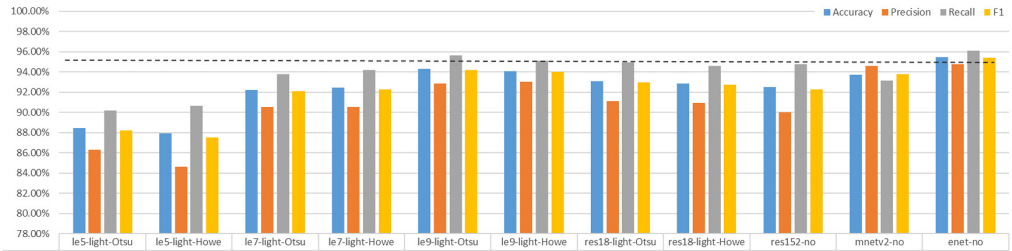
In summary, we find from this investigation that CNNs coupled with light-level preprocessing (i.e., binarization) outperformed their counterparts that are coupled with aggressive-level preprocessing (i.e., consolidation). Note that consolidation generated a more connected and enhanced visual layout of text lines than binarization. One might expect that, as a result, a CNN coupled with consolidation would outperform one with binarization. Our findings indicate that, unexpectedly, though the visual cues were enhanced after consolidation, there was sufficient information loss that degraded the CNN's performance. Thus one should be cautious when deciding on the appropriate preprocessing techniques for CNN and not be reliant on only the visual quality of the preprocessed images.

4.3 *Investigating CNNs with Different Levels of Task Difficulty*

In this investigation, we compare the performance of CNN models with different depth configurations coupled with light-level preprocessing on the two classification tasks with different levels of difficulty. Note that we only apply light-level preprocessing, light-Otsu, and light-Howe since only light-level preprocessing improved CNN's performance in the second investigation, as reported in Sec. 4.2.

4.3.1 *Task 1: binary poem classification*

In this investigation, the configuration of task 1 is similar to the configuration used in the second investigation (Sec. 4.2). However, there is one key difference. We use an expanded set of CNN models, which are shallow [i.e., LeNet-7 (le7) and LeNet-9 (le9)], deep [i.e., ResNet-18 (res18)], and very deep [i.e., ResNet-152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet)] to match the models used in task 2 (see details next) for comparison. We observe that LeNet-9 with light level preprocessing using Otsu's method (light-Otsu), outperformed almost all very deep CNN models without preprocessing in terms of test accuracy, precision, recall, and $F1$-score, as shown in Fig. 17. But there are two exceptions: EfficientNet and precision of MobileNetV2.



	le5-light-Otsu	le5-light-Howe	le7-light-Otsu	le7-light-Howe	le9-light-Otsu	le9-light-Howe	res18-light-Otsu	res18-light-Howe	res152-no	mnetv2-no	enet-no
Accuracy	88.45%	87.91%	92.22%	92.44%	94.29%	94.09%	93.09%	92.86%	92.49%	93.70%	95.45%
Precision	86.32%	84.61%	90.50%	90.54%	92.85%	93.02%	91.10%	90.95%	90.02%	94.58%	94.78%
Recall	90.21%	90.66%	93.75%	94.17%	95.63%	95.08%	94.91%	94.60%	94.75%	93.11%	96.09%
F1	88.18%	87.49%	92.08%	92.28%	94.20%	94.02%	92.93%	92.71%	92.28%	93.76%	95.42%

Fig. 17 Test results of the binary poem classification on the Aida-17k dataset using 10-fold cross validation comparing shallower CNNs, LeNet-5 (le5), LeNet-7 (le7), LeNet-9 (le9), and ResNet-18 (res18) coupled with light level of preprocessing and deeper CNNs ResNet152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet) without preprocessing.

Compared with EfficientNet, the LeNet-9 was not very far behind; however, there were −1.16% in accuracy, −1.93% in precision, −0.46% in recall, and −1.22% in $F1$-score. Furthermore, ResNet-18, the deep model coupled with preprocessing, also outperformed a very deep model, ResNet-152, in all four metrics used.

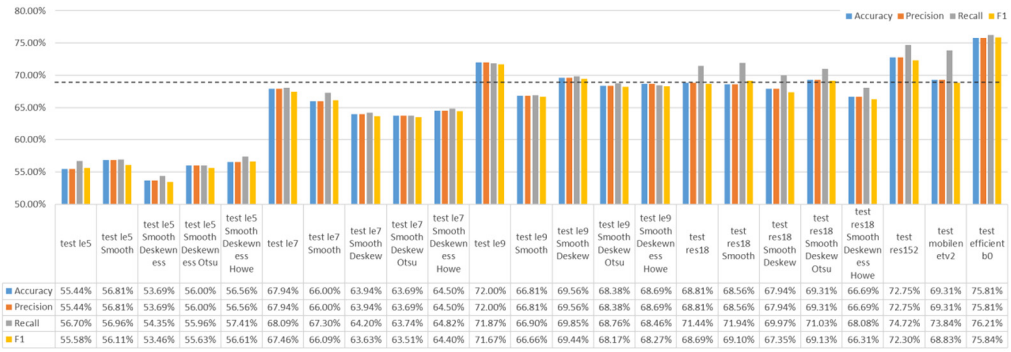
4.3.2 *Task 2: 16-class document type classification*

Here we use additional light-level preprocessing in addition to binarization to include smoothing and deskewing, as they are relevant in dealing with full document images in the RVL-CDIP dataset. Again, we use the same two different binarization methods, Otsu's and Howe's. Thus these techniques together yield four preprocessing variants: (1) smoothing only (smooth), (2) smoothing and deskewing (smooth + deskew), (3) smoothing, deskewing, and the Otsu's binarization (smooth + deskew + Otsu), and (4) smoothing, deskewing, and the Howe's binarization (smooth + deskew + Howe), which are compared with no-preprocessing (no). Figure 18 shows the testing results of LeNet-5, LeNet-7, LeNet-9, and ResNet-18 with the four light-level preprocessing variants and ResNet-152, MobileNetV2, and EfficientNet without preprocessing. Only LeNet-9 with smoothing and deskewing outperformed a deeper CNN model, MobileNetV2, by 0.25% in accuracy, 0.25% in precision, and 0.27% in $F1$-score.

In summary, we see that, for classification tasks of different levels of difficulty, such as the simpler binary classification task and the more challenging 16-class document type classification task, a shallower CNN's performance (i.e., LeNet-9) with respect to very deep CNNs' can be impacted by coupling it with preprocessing. When coupled with preprocessing, the shallower CNN outperformed, in terms of $F1$ score, those of very deep CNNs by as much as 1.92% in the binary classification task and by as much as 0.61% in the 16-class document type classification task. Note that the percentage of improvement for the more challenging task is smaller. The reason could stem from the increased difficulty of the 16-class classification task. A preprocessing strategy cleans up such that their desired visual features are more salient. However, in a 16-class classification task, it is challenging for such enhancement to also lead to increased separation among the classes; for example, a strategy that further differentiates two classes A and B might lead to classes B and C being closer visually. Thus the preprocessing's positive impact on the 16-class document type classification is less.

4.4 *Investigating Smaller Training Sets*

In this investigation, we compare the performance of CNN models almost exactly the same way as that in the third investigation except for using smaller training sets. Here we construct smaller training sets based on both the Aida-17k and the RVL-CDIP-balanced datasets to investigate whether, in the case of a smaller training set, preprocessing can help to train a CNN-based classifier with better performance. In the following, we designate a smaller training set



	test le5	test le5 Smooth	test le5 Smooth Deskewness	test le5 Smooth Deskewness Otsu	test le5 Smooth Deskewness Howe	test le7	test le7 Smooth	test le7 Smooth Deskew	test le7 Smooth Deskew Otsu	test le7 Smooth Deskewness Howe	test le9	test le9 Smooth	test le9 Smooth Deskew	test le9 Smooth Deskew Otsu	test le9 Smooth Deskewness Howe	test res18	test res18 Smooth	test res18 Smooth Deskew	test res18 Smooth Deskew Otsu	test res18 Smooth Deskewness Howe	test res152	test mobilenetv2	test efficientb0
Accuracy	55.44%	56.81%	53.69%	56.00%	56.56%	67.94%	66.00%	63.94%	63.69%	64.50%	72.00%	66.81%	69.56%	68.38%	68.69%	68.81%	68.56%	67.94%	69.31%	66.69%	72.75%	69.31%	75.81%
Precision	55.44%	56.81%	53.69%	56.00%	56.56%	67.94%	66.00%	63.94%	63.69%	64.50%	72.00%	66.81%	69.56%	68.38%	68.69%	68.81%	68.56%	67.94%	69.31%	66.69%	72.75%	69.31%	75.81%
Recall	56.70%	56.96%	54.35%	55.96%	57.41%	68.09%	67.30%	64.20%	63.74%	64.82%	71.87%	66.90%	69.85%	68.76%	68.46%	71.44%	71.94%	69.97%	71.03%	68.08%	74.72%	73.84%	76.21%
F1	55.58%	56.11%	53.46%	55.63%	56.61%	67.46%	66.09%	63.63%	63.51%	64.40%	71.67%	66.66%	69.44%	68.17%	68.27%	68.69%	69.10%	67.35%	69.13%	66.31%	72.30%	68.83%	75.84%

Fig. 18 Test results of the 16-class document type classification on the RVL-CDIP-balanced dataset comparing shallower CNNs, LeNet-5 (le5), LeNet-7 (le7), LeNet-9 (le9), and ResNet-18 (res18) coupled with light level of preprocessing and deeper CNNs ResNet152 (res152), MobileNetV2 (mnetv2), and EfficientNet (enet) without preprocessing.

"Aida-17k-90" if it consists of 90% of the original Aida-17k dataset and so forth. Further, for both datasets, we make sure that the numbers of images for every class are balanced in each smaller training set. In this task, we train each CNN six times. Each time, 10% of training samples are removed from the training set. Hence, the training sets used are (1) 100%, (2) 90%, (3) 80%, (4) 70%, (5) 60%, and (6) 50% of the full training set. To do so, we build different smaller training sets from the Aida-17k and the RVL-CDIP-balance.

4.4.1 *Task 1: binary poem classification*

In this investigation, the configuration of task 1 is similar to the configuration in the third investigation (Sec. 4.3). There is one difference that we compare the performances of shallow CNNs, LeNet-7 and LeNet-9 and a deep CNN, ResNet-18, coupled with using light-level preprocessing—having found them to be effective from previous investigations—using smaller training sets. Table 6 shows that there were only three (out of 15) cases (LeNet-7 at 70%, LeNet-9 at 90%, and ResNet-18 at 60%) with light-level preprocessing that outperformed their no-preprocessing counterparts, among all smaller training sets (90% to 50%). This indicates that light-level preprocessing does not help address the challenge of having smaller training sets in this task.

4.4.2 *Task 2: 16-class document type classification*

Here we also use a similar configuration as task 2 in the third investigation. We compare three CNNs coupled with light-level preprocessing: shallow, LeNet-7 and LeNet-9 and deep, ResNet-18. Table 7 shows that the majority (13 out of 15 cases) of light-level preprocessing combinations outperformed their no-preprocessing counterparts, except for LeNet-7 at 90% and LeNet-9 at 60%.

In summary, from the mixed performance results dealing with a smaller amount of training data, we observe that preprocessing can play an effective role in improving a CNN's performance. The performance of CNN was improved in the more challenging 16-class document type classification task more than the simpler binary poem classification task. It implies that the CNN model coupled with preprocessing may be able to generalize better than the model without preprocessing in some cases. Further, we find that preprocessing impacts ResNet a bit more than LeNet: preprocessing improves ResNet's performance 6 out of 10 times (60%) and LeNet's performance 10 out of 20 times (50%). This is likely due to a fundamental difference between the two architectures. ResNet has a "shortcut connection" structure[65] that LeNet does not have. It is known that the shortcut connection helps CNN retain information or details better from the beginning layers to the last layers.[65] As a result, ResNet could retain the detailed visual cues, for example, enhanced by preprocessing better than LeNet. On the other hand, as the layer gets deeper in LeNet, the information becomes more abstracted, diminishing the subtle visual cues and thus minimizing the impact of preprocessing.

5 Conclusion and Future Work

In this paper, to understand the impact of preprocessing on CNN's performance in terms of effectiveness and efficiency, we studied several state-of-the-art CNN models of different depths (Sec. 3.2), two levels of preprocessing techniques (Sec. 3.1), and two classification tasks with different levels of difficulty in four sets of investigations (Sec. 4). The first investigation provides a baseline for the performances of shallow, deep, and very deep CNN models on two classification tasks and demonstrates the potential of shallower CNNs to match the performance of deeper CNNs. This baseline contextualizes the subsequent three investigations.

Building on the baseline investigation, the second investigation compared light-level and aggressive-level preprocessing techniques using a binary poem classification task. We found that even though aggressive-level preprocessing could enhance the cues visually, it could degrade CNN's performance due to excessive information loss. Encouraged by the findings from the second investigation, the third investigation looked into how the improvement provided by preprocessing could bridge the performance gap between shallow CNNs and deep CNNs. We

Table 6 Test accuracies of smaller training results on Aida-17k dataset.

		Smaller training samples											
		100%		90%		80%		70%		60%		50%	
	Accuracy (%)	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
LeNet-7	No	99.56	91.15	99.69	**92.74**	99.86	**91.91**	99.73	90.79	96.86	**90.67**	99.97	**89.79**
	Light-Otsu	97.75	92.09	99.89	91.03	93.46	91.32	99.91	90.79	99.31	89.14	96.63	87.78
	Light-Howe	99.96	**92.86**	99.71	92.38	97.58	90.97	99.95	**91.26**	99.86	88.61	95.13	89.43
LeNet-9	No	99.29	***95.04***	99.70	94.21	99.92	***94.57***	99.59	***93.51***	99.27	***94.69***	99.94	***91.85***
	Light-Otsu	97.95	94.57	99.89	93.62	94.08	91.32	99.61	92.74	99.11	90.32	99.29	91.50
	Light-Howe	98.49	92.50	99.93	***94.75***	99.79	93.86	98.92	91.79	99.18	92.62	99.51	91.62
ResNet-18	No	96.69	91.15	97.76	**94.04**	98.16	**94.21**	98.13	**92.50**	97.32	89.08	98.73	**91.79**
	Light-Otsu	98.33	92.44	98.74	92.33	98.75	91.79	97.68	92.27	98.76	**91.15**	98.97	90.79
	Light-Howe	95.27	**93.57**	96.88	91.50	98.36	91.62	98.16	91.74	98.65	91.03	98.64	85.42

Table 7 Test accuracies of smaller training results on RVL-CDIP-balanced dataset.

		Smaller training samples											
		100%		90%		80%		70%		60%		50%	
	Accuracy (%)	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
LeNet-7	No	96.58	**67.94**	99.63	**65.31**	99.34	65.00	99.72	62.81	99.26	60.63	98.56	58.19
	Smooth	97.68	66.00	99.40	64.81	97.22	**66.75**	99.18	62.56	98.30	**62.69**	96.38	**63.06**
	Smooth + deskew	99.64	63.94	86.96	63.50	99.73	58.06	99.46	61.00	99.68	57.50	99.76	60.31
	Smooth + deskew + Otsu	97.04	63.69	99.08	62.00	99.30	65.31	97.62	**63.44**	99.74	60.69	98.29	59.75
	Smooth + deskew + Howe	99.46	64.50	99.53	58.75	97.80	61.75	99.63	56.38	99.47	56.44	99.68	57.13
LeNet-9	No	99.01	***72.00***	98.95	65.56	98.27	65.19	98.35	64.88	99.28	***67.44***	99.41	61.38
	Smooth	94.44	66.81	98.99	***70.63***	98.47	65.06	99.03	66.75	99.35	65.63	99.55	61.81
	Smooth + deskew	99.30	69.56	96.00	69.26	99.41	***69.38***	98.68	***66.88***	98.17	60.06	98.78	***64.63***
	Smooth + deskew + Otsu	99.28	68.38	98.63	66.69	96.80	68.63	98.97	61.31	99.27	57.81	99.59	62.00
	Smooth + deskew + Howe	99.09	68.69	99.08	65.25	98.08	63.94	99.35	64.75	97.80	63.19	99.43	60.38
ResNet-18	No	78.63	68.81	94.65	65.00	94.75	64.69	91.77	63.25	94.80	61.00	95.88	58.63
	Smooth	91.41	68.56	94.49	67.25	94.33	**66.50**	95.61	**63.88**	95.32	61.75	96.35	**59.38**
	Smooth + deskew	92.83	67.94	93.91	**68.06**	95.52	62.69	95.29	62.31	95.20	**62.00**	89.94	55.81
	Smooth + deskew + Otsu	94.71	**69.31**	94.58	64.94	96.03	62.63	95.31	61.69	95.61	58.44	96.04	57.69
	Smooth + deskew + Howe	95.06	66.69	95.40	66.56	94.75	64.25	95.74	62.75	95.55	58.00	95.29	53.75

found that shallow CNNs coupled with preprocessing could yield better performance than deep CNNs' in both the binary and 16-class classification tasks. However, the degree of improvement was smaller when the classification task was more challenging, as in the 16-class document type classification task in which it was more difficult to enhance the separation between the many classes. For the fourth investigation, we considered efficiency and the constraint of having a small number of training samples. We found that CNN models coupled with preprocessing could outperform those without preprocessing in cases in which there were smaller training samples. This was more so when the classification task was more challenging (e.g., the 16-class classification task). This implies that preprocessing could help CNN models learn more from a smaller set of training samples. This in turn hints that preprocessing could help CNN training more efficiently as it would require a smaller set of training samples.

Overall, based on our investigations, we derive three pieces of insights or suggestions for when and how to use preprocessing for classification tasks using CNNs.

- An aggressive preprocessing technique such as consolidation is not helpful, even though it could highlight visual cues better than a light preprocessing technique, since it could also remove potentially useful information in the document image. This means that even when, say, preprocessing technique A generates a better image visually than preprocessing technique B, it is not guaranteed that coupling A with a CNN would yield better classification accuracy than coupling B with a CNN.
- A preprocessing technique coupled with a shallow CNN could help improve performance effectively for a relatively less challenging classification task, even to the point of outperforming a much deeper CNN. This means that practitioners could feasibly consider using preprocessing instead of always opting for deeper CNN models, as deeper CNN models typically demand more computing power than shallow CNNs and different classification tasks have different levels of difficulty.
- A preprocessing technique coupled with a CNN (shallow or deep or very deep) could help improve performance effectively in the case of limited training data, especially when the classification tasks are relatively more challenging. This means that in cases in which the number of ground-truthed training samples is small, practitioners could look to using preprocessing to improve the performance of the CNN model.

Considering together the second and third insights given above, we see that preprocessing was more helpful to a shallow CNN in a classification task that was less challenging, whereas preprocessing was more helpful to CNNs in a classification task that was more challenging in cases in which the number of samples was small. One would expect that preprocessing's role or impact on the performance of a CNN to trend similarly in such classification tasks, but our findings show otherwise. This could mean that there is a sweet spot at which an optimal level of preprocessing could yield the most effectiveness and efficiency when coupled with a CNN. This motivates our next steps in further investigating coupling preprocessing with a CNN.

In terms of future work, first, the CNN models are developed essentially based on general images. However, there are special visual cues that only the document image has, such as aligned text lines and sematic information between characters. We will continue to extend our investigation to develop more suitable CNN models for document classification tasks. This would involve exploring more CNN architectures such as inception[12] and DenseNet;[77] more document image preprocessing techniques such as Zemouri and Chibani's[78] binarization for degraded document images, Koo and Cho's skewness estimation,[79] and image augmentation[80] to increase the amount of "groundtruth" training data; and more document image databases such as a medieval document image collection.[81] Second, our investigations revealed impact trends of coupling preprocessing the CNN's performance and demonstrated that the impact of coupling preprocessing could stem from different factors. This means that the selection of appropriate preprocessing techniques is a non-trivial problem. In particular, can we automate the selection of preprocessing techniques to couple with a CNN for a particular type of classification task? We plan to investigate the properties of the document images in our classification tasks and the effectiveness of preprocessing techniques in terms of visual cues exploited by CNN models to lay the groundwork for such an intelligent system to select preprocessing adaptively. Third, we plan to

investigate the width of the CNN architecture as another factor that influences the impact of preprocessing. Fourth, with respect to the application domain, historical newspaper classification using CNN is underdeveloped. For example, Chronicling America has a vast historical newspaper collection of which searchability eagerly needs an expansion. Hence, we will continue to investigate other classification tasks that involve color document images and for other journalistic elements (e.g., advertisements, obituaries, and job postings) using CNNs to extend the searchability of historical document collections.

Acknowledgments

This project was supported in part by the Institute of Museum and Library Services and has received previous support from the National Endowment for the Humanities. Charles Nugent helped build the initial convolutional neural network architecture that shaped its development. This work was completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative.

References

1. X. Cao et al., "Hyperspectral image classification with Markov random fields and a convolutional neural network," *IEEE Trans. Image Process.* **27**, 2354–2367 (2018).
2. Y. He et al., "Multiscale dual-level network for hyperspectral image classification," *J. Electron. Imaging* **29**, 033008 (2020).
3. H. Lee and H. Kwon, "Going deeper with contextual CNN for hyperspectral image classification," *IEEE Trans. Image Process.* **26**, 4843–4855 (2017).
4. L. Wang et al., "Knowledge guided disambiguation for large-scale scene classification with multi-resolution CNNs," *IEEE Trans. Image Process.* **26**, 2055–2068 (2017).
5. J. Yang et al., "Multi-channel and multi-scale mid-level image representation for scene classification," *J. Electron. Imaging* **26**, 023018 (2017).
6. T. Fang et al., "Crop leaf disease grade identification based on an improved convolutional neural network," *J. Electron. Imaging* **29**, 013004 (2020).
7. S. H. Lee, C. S. Chan, and P. Remagnino, "Multi-organ plant classification based on convolutional and recurrent neural networks," *IEEE Trans. Image Process.* **27**, 4287–4301 (2018).
8. S. Bianco et al., "Artistic photo filter removal using convolutional neural networks," *J. Electron. Imaging* **27**, 011004 (2017).
9. T. Chen, S. Lu, and J. Fan, "SS-HCNN: semi-supervised hierarchical convolutional neural network for image classification," *IEEE Trans. Image Process.* **28**, 2389–2398 (2019).
10. G. Ding et al., "DECODE: deep confidence network for robust image classification," *IEEE Trans. Image Process.* **28**, 3752–3765 (2019).
11. Z. Pan et al., "Topic network: topic model with deep learning for image classification," *J. Electron. Imaging* **27**, 033009 (2018).
12. C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 1–9 (2015).
13. T. Remez et al., "Class-aware fully convolutional Gaussian and Poisson denoising," *IEEE Trans. Image Process.* **27**, 5707–5722 (2018).
14. K. Zhang et al., "Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.* **26**, 3142–3155 (2017).
15. X. Fu et al., "Clearing the skies: a deep network architecture for single-image rain removal," *IEEE Trans. Image Process.* **26**, 2944–2956 (2017).
16. Y. Liu et al., "DesnowNet: context-aware deep network for snow removal," *IEEE Trans. Image Process.* **27**, 3064–3073 (2018).
17. K. Zhang, W. Zuo, and L. Zhang, "FFDNet: toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.* **27**, 4608–4622 (2018).
18. A. Krizhevsky, "Learning multiple layers of features from tiny images," PhD Thesis, University of Toronto (2009).

19. J. Deng et al., "ImageNet: a large-scale hierarchical image database," in *IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 248–255 (2009).
20. Y. Lecun et al., "Gradient-based learning applied to document recognition," *Proc. IEEE* **86**, 2278–2324 (1998).
21. A. W. Harley, A. Ufkes, and K. G. Derpanis, "Evaluation of deep convolutional nets for document image classification and retrieval," in *13th Int. Conf. Doc. Anal. and Recognit.*, pp. 991–995 (2015).
22. R. Jain and C. Wigington, "Multimodal document image classification," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 71–77 (2019).
23. V. Pondenkandath et al., "Exploiting state-of-the-art deep learning methods for document image analysis," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 30–35 (2017).
24. S. Ukil et al., "Improved word-level handwritten Indic script identification by integrating small convolutional neural networks," *Neural Comput. Appl.* **32**, 2829–2844 (2020).
25. K. Chen et al., "Convolutional neural networks for page segmentation of historical document images," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 965–970 (2017).
26. M. J. Khan et al., "Deep learning for automated forgery detection in hyperspectral document images," *J. Electron. Imaging* **27**, 053001 (2018).
27. S. C. Kosaraju et al., "DoT-net: document layout classification using texture-based CNN," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 1029–1034 (2019).
28. G. Renton et al., "Handwritten text line segmentation using fully convolutional network," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 5–9 (2017).
29. Y. Xu et al., "Page segmentation for historical handwritten documents using fully convolutional networks," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 541–546 (2017).
30. S. Tarride et al., "Combination of deep neural networks and logical rules for record segmentation in historical handwritten registers using few examples," *Int. J. Doc. Anal. Recognit.* **24**, 77–96 (2021).
31. A. Basu et al., "U-Net versus Pix2Pix: a comparative study on degraded document image binarization," *J. Electron. Imaging* **29**, 063019 (2020).
32. C. Tensmeyer and T. Martinez, "Document image binarization with fully convolutional neural networks," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 99–104 (2017).
33. T. Gruning et al., "A two-stage method for text line detection in historical documents," *Int. J. Doc. Anal. Recognit.* **22**, 285–302 (2019).
34. O. Mechi et al., "Text line segmentation in historical document images using an adaptive U-net architecture," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 369–374 (2019).
35. A. Dutta et al., "Segmentation of text lines using multi-scale CNN from warped printed and handwritten document images," *Int. J. Doc. Anal. Recognit.* (2021).
36. I. Uddin et al., "Recognition of printed Urdu ligatures using convolutional neural networks," *J. Electron. Imaging* **28**, 033004 (2019).
37. S. Zahoor et al., "Deep optical character recognition: a case of Pashto language," *J. Electron. Imaging* **29**, 023002 (2020).
38. D. Sinwar et al., "Offline script recognition from handwritten and printed multilingual documents: a survey," *Int. J. Doc. Anal. Recognit.* **24**, 97–121 (2021).
39. K. C. Santosh, *Document Image Analysis: Current Trends and Challenges in Graphics Recognition*, Springer, Singapore (2018).
40. J.-C. Burie et al., "Deep learning for graphics recognition: document understanding and beyond," *Int. J. Doc. Anal. Recognit.* **24**, 1–2 (2021).
41. A. Sulaiman, K. Omar, and M. F. Nasrudin, "Degraded historical document binarization: a review on issues, challenges, techniques, and future directions," *J. Imaging* **5**, 48 (2019).
42. C. Tensmeyer and T. Martinez, "Analysis of convolutional neural networks for document image classification," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, pp. 388–393 (2017).
43. G. Nagy, "Twenty years of document image analysis in PAMI," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 38–62 (2000).
44. M. van Erp, M. Wevers, and H. Huurdeman, "Constructing a recipe web from historical newspapers," *Lect. Notes Comput. Sci.* **11136**, 217–232 (2018).

45. T. Lansdall-Welfare et al., "Content analysis of 150 years of British periodicals," *Proc. Natl. Acad. Sci. U. S. A.* **114**, E457–E465 (2017).
46. V. P. d'Andecy et al., "Discourse descriptor for document incremental classification comparison with deep learning," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 467–472 (2019).
47. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lect. Notes Comput. Sci.* **8689**, 818–833 (2014).
48. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.* **9**, 62–66 (1979).
49. I. Pratikakis, B. Gatos, and K. Ntirogiannis, "ICDAR 2011 document image binarization contest (DIBCO 2011)," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 1506–1510 (2011).
50. I. Pratikakis, B. Gatos, and K. Ntirogiannis, "ICDAR 2013 document image binarization contest (DIBCO 2013)," in *12th Int. Conf. Doc. Anal. and Recognit.*, pp. 1471–1476 (2013).
51. I. Pratikakis et al., "ICDAR2017 competition on document image binarization (DIBCO 2017)," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, Vol. 1, pp. 1395–1403 (2017).
52. I. Pratikakis et al., "ICDAR 2019 competition on document image binarization (DIBCO 2019)," in *Int. Conf. Doc. Anal. and Recognit.*, pp. 1547–1556 (2019).
53. J. Liu, W. Li, and Y. Tian, "Automatic thresholding of gray-level pictures using two-dimension Otsu method," in *China Int. Conf. Circuits and Syst.*, Vol. 1, pp. 325–327 (1991).
54. O. Nina, B. Morse, and W. Barrett, "A recursive Otsu thresholding method for scanned document binarization," in *IEEE Workshop Appl. Comput. Vision*, pp. 307–314 (2011).
55. N. R. Howe, "Document binarization with automatic parameter tuning," *Int. J. Doc. Anal. Recognit.* **16**, 247–258 (2013).
56. J. van Beusekom, F. Shafait, and T. M. Breuel, "Combined orientation and skew detection using geometric text-line modeling," *Int. J. Doc. Anal. Recognit.* **13**, 79–92 (2010).
57. K. He, J. Sun, and X. Tang, "Guided image filtering," *Lect. Notes Comput. Sci.* **6311**, 1–14 (2010).
58. L.-K. Soh, E. Lorang, and Y. Liu, "Aida: intelligent image analysis to automatically detect poems in digital archives of historic newspapers," in *Proc. Thirtieth Innovative Appl. Artif. Intell. Conf.* (2018).
59. J. Hu, R. Kashi, and G. Wilfong, "Document classification using layout analysis," in *Proc. Tenth Int. Workshop Database and Expert Syst. Appl.*, Vol. **99**, pp. 556–560 (1999).
60. V. Loia and S. Senatore, "An alternative, layout-driven approach to the clustering of documents," *Int. J. Intell. Syst.* **23**(7), 795–821 (2008).
61. C. Shin, D. Doermann, and A. Rosenfeld, "Classification of document pages using structure-based features," *Int. J. Doc. Anal. Recognit.* **3**, 232–247 (2001).
62. K. C. Santosh, "G-DICE: graph mining-based document information content exploitation," *Int. J. Doc. Anal. Recognit.* **18**, 337–355 (2015).
63. E. Lorang et al., "Developing an image-based classifier for detecting poetic content in historic newspaper collections" (2015).
64. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM* **60**, 84–90 (2017).
65. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 770–778 (2016).
66. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Thirteenth Int. Conf. Artif. Intell. and Stat.*, pp. 249–256 (2010).
67. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 7132–7141 (2018).
68. M. Diem et al., "cBAD: ICDAR2017 competition on baseline detection," in *14th IAPR Int. Conf. Doc. Anal. and Recognit.*, Vol. 1, pp. 1355–1360 (2017).
69. M. Sandler et al., "MobileNetV2: inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 4510–4520 (2018).
70. M. Tan and Q. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in *Int. Conf. Mach. Learn.*, pp. 6105–6114, PMLR (2019).
71. N. Zhu et al., "A fast 2D otsu thresholding algorithm based on improved histogram," in *Chin. Conf. Pattern Recognit.*, pp. 1–5 (2009).

72. G. E. Hinton et al., "Improving neural networks by preventing co-adaptation of feature detectors," arXiv:1207.0580 (2012).
73. L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.* **29**(6), 141–142 (2012).
74. O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision* **115**, 211–252 (2015).
75. E. Lorang et al., "Aida NEH start-up grant data, 1836-1840 case study," (2017).
76. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *3rd Int. Conf. Learn. Represent.*, Y. Bengio and Y. LeCun, Eds., San Diego, CA (2015).
77. G. Huang et al., "Densely connected convolutional networks," in *IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 2261–2269 (2017).
78. E.-T. Zemouri and Y. Chibani, "Nonsubsampled contourlet transform and k-means clustering for degraded document image binarization," *J. Electron. Imaging* **28**, 043021 (2019).
79. H. Koo and N. I. Cho, "Robust skew estimation using straight lines in document images," *J. Electron. Imaging* **25**, 033014 (2016).
80. N. Journet et al., "DocCreator: a new software for creating synthetic ground-truthed document images," *J. Imaging* **3**, 62 (2017).
81. S. En et al., "New public dataset for spotting patterns in medieval document images," *J. Electron. Imaging* **26**, 011010 (2016).

Yi Liu received his BE degree in computer science and technology from Shanghai University of Engineering Science, Shanghai, China, in 2015. He is a doctoral student at the University of Nebraska–Lincoln. He is currently pursuing his PhD in computer science at the University of Nebraska, Lincoln, NE, USA, where he is a graduate research assistant. His research interests include the development of image-based historical newspaper analysis using computer vision and machine learning techniques.

Leen-Kiat Soh is a professor of Computer Science and Engineering at the University of Nebraska–Lincoln. His research areas are in multiagent systems and modeling, computer science education, computer-aided education, and intelligent data analytics including image processing, applied artificial intelligence, and multiagent simulation. He has also contributed to broadening participation in computing and computational thinking. He has published more than 200 journals and conferences. He is a member of ACM, AAAI, and IEEE.

Elizabeth Lorang is an associate professor in the University Libraries at the University of Nebraska–Lincoln, and she is a fellow of the Center for Digital Research in the Humanities and the Center for Great Plains Studies. She co-leads a research team exploring image analysis and machine learning in digital libraries of historic materials. She has received funding from the Council on Library and Information Resources, Institute of Museum and Library Services, and National Endowment for the Humanities.