Robust, Simple Page Segmentation using Hybrid Convolutional MDLSTM Networks

Thomas M. Breuel
NVIDIA Research
Santa Clara, CA, USA
tbreuel@nvidia.com

Abstract— **Analyzing and segmenting scanned documents is an important step in optical character recognition. The problem is difficult because of the complexity of 2D layouts, the small tolerance of segmentation errors in the output, and the relatively small amount of labeled training data available. Traditional approaches have relied on a combination of sophisticated geometric algorithms, domain knowledge, heuristics, and carefully tuned parameters. This paper describes the use of deep neural networks, in particular a combination of convolutional and multidimensional LSTM networks, for document image and demonstrates that relatively simple networks are capable of fast, reliable text line segmentation and document layout analysis even on complex and noisy inputs, without manual parameter tuning or heuristics. The method is easily adaptable to new datasets by retraining and an open source implementation is available.**

I. Introduction

Optical character recognition (OCR) of printed documents starts with physical layout analysis, which divides scanned pages into blocks containing text, tables, images; identifies column boundaries; and finds text lines. Text lines are then extracted and passed on to a text line recognizer. A large number of methods have been proposed for the problem of layout analysis, including XY-cuts[9], mathematical morphology[15, 10], geometric matching[3], and tab stop detection[19]. Machine learning has been incorporated into some of these algorithms, for example into HMM-based layout analysis[11], white space classification[1], and some forms of neural networks[14, 7]. Many layout analysis algorithms analyze page content as connected components, so they require binarized input. Furthermore, algorithms often look for geometric patterns such as parallelism, co-linearity, and rectangles, so they often require dewarped and deskewed inputs. Binarization, dewarping, and deskewing have all themselves been subject to extensive research.

Over the last few years, there have been big advances in the use of deep convolutional and recurrent networks for image segmentation and object recognition[17, 13]. Furthermore, the use of GPUs for deep learning has made it possible to run deep convolutional networks over large and high resolution images. Such methods have been applied to text detection and text recognition[12] in natural images. However, printed OCR problems are different from scene text problems in that they require large amounts of text to be recognized at very low error rates, involve the recognition of many text lines that span the entire image require accurate reading order determination, and operate on high resolution images.

While similar to semantic segmentation, layout analysis differs from common semantic segmentation problems in computer vision in that information necessary for segmenting tends to be sparse and exhibit long-range dependencies, for example, in the form of column boundaries, separators, and whitespace. Multidimensional LSTMs are good at capturing such dependencies[6]. Furthermore, a straightforward formulation of text line segmentation in terms of partitioning the page into text line regions and background/other regions does not work well because gaps between text line regions tend to be small. Those are the reasons behind the two major design decisions in the deep segmenter: using a hybrid convolutional-LSTM model and predicting text centerlines instead of text line regions.

High accuracy text line recognition for printed OCR has been described based on LSTM networks[4] and also works directly on warped grayscale images without prior binarization.

This paper describes a hybrid convolutional-recurrent network that addresses the document layout analysis problem by performing text line detection; by preprocessing the training data and the objective functions, this also addresses border noise removal, text frame detection, text/math segmentation, text/image segmentation, and column detection. The resulting layout analysis system is simple, yet performs as well as state of the art open source systems[19].

II. Layout Analysis with Deep Learning

A. Training Data

We formulate the layout analysis problem as an image to image transformation problem. As the input to the model, we use either scanned binary images or grayscale images. The output that the model is trained on is a binary image constructed from page segmentation ground truth. For all training reported in this paper, we use (subsets of) the University of Washington Database (UW3). UW3 ground truth indicates page segmentation at multiple levels (paragraphs, lines, words). Training data was augmented by random rotations by $\pm 2\deg$, random translations by $\pm 5\%$ and random anisotropic scaling by $\pm 5\%$.

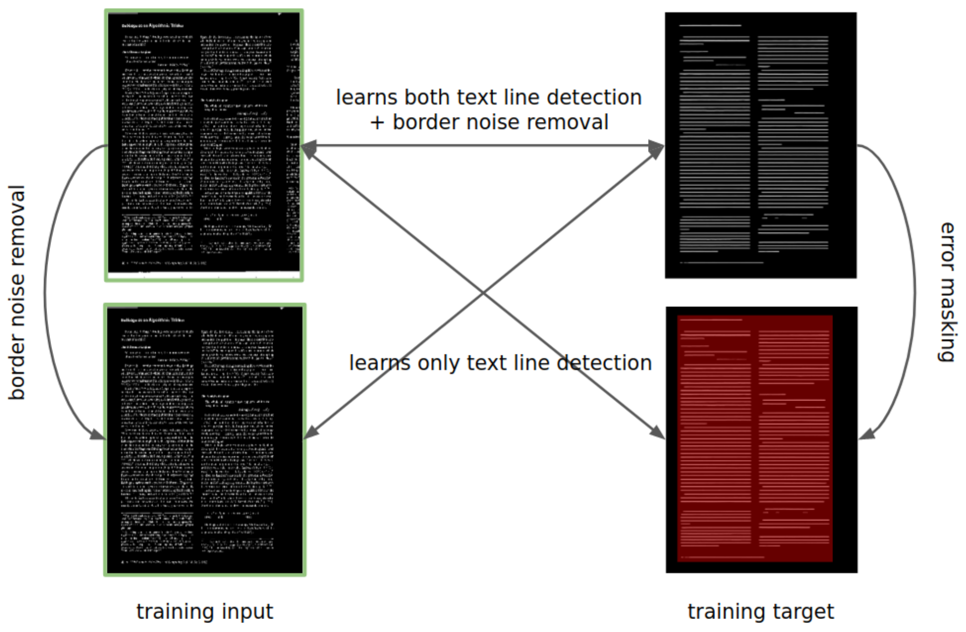

Fig. 1. Layout analysis is treated as an image-to-image transformation performed by a deep neural network trained on layout analysis data. The most basic form of training trains a deep neural network model to transform the original scanned page image into representation of the centerlines of the text lines (with the training data derived from the ground truth layout data). Such a deep model learns to perform both text line detection and border noise removal simultaneously. By preprocessing either the input image (removing border noise) or masking the errors during backpropagation, we can train a different kind of layout analysis model that performs text line detection without border noise removal.

By preprocessing the input images for layout analysis in different ways and constructing different targets for training, we can obtain deep neural networks that perform a variety of different layout analysis tasks. To illustrate this consider training a text line detector via image-to-image transformation. The ground truth for such training consists of a scanned document page together with a map of where the text lines are. The scanned document page may contain text lines from facing pages as well, but these additional text lines are not labeled in the ground truth as text lines, since they are considered border noise. Solving the text line detection problem successfully given training data that contains text lines within both the page and the border noise requires that a deep network not only learn the text line detection problem, but also the border noise detection problem. On the other hand, if we mask out the border noise (using the ground truth layout labels) prior to training, then the deep network can solve the text line detection problem without ever needing to make the distinction between text lines within border noise, and text lines within the main page.

Furthermore, by constructing different kinds of target images, we have more control over the kinds of layout analysis model we obtain. For example, by choosing as a target a binary representation of the center of each text line, we obtain a text line detector; if instead we choose as a target a binary mask covering each column, we obtain a text column detector. We will see examples of this later, but the focus of this paper is text line based segmentation.

For training a deep network that performs layout analysis by an image-to-image transformation, we need to construct an image target that encodes the position of text lines in the form of a binary image. Document layout analysis ground truth is usually given in the form of bounding boxes. An obvious representation of text line ground truth is therefore to take the bounding boxes for each text line from the layout analysis ground truth, render those bounding boxes as a binary image, and use that binary image as a target for the image-to-image training. However, that approach does not work very well because text line bounding boxes frequently touch or overlap; even if the deep model learned the image transformation perfectly, we could still not recover the position of the individual text lines from the output of the network.

A better approach is to learn a marker line for the location of the text line. Obvious markers are the center line of the bounding box or the baseline of the text line; however, both of those markers are not stable under small changes of the input image. The marker line we actually use is the same centerline that has been used successfully for 1D LSTM-based character recognition as described in [5]. That marker line has the advantage that it seems to be quite stable under noise and changes in image content. It can also be computed easily with simple image processing operations. In [5], the marker line is used for dewarping the text line image; here, instead, we render the location of the marker line back into the page image to in order to obtain a target binary image for layout analysis training. The marker line computation is carried out as follows:

- The original page image is binarized (this is just done for the construction of the ground truth) and inverted.

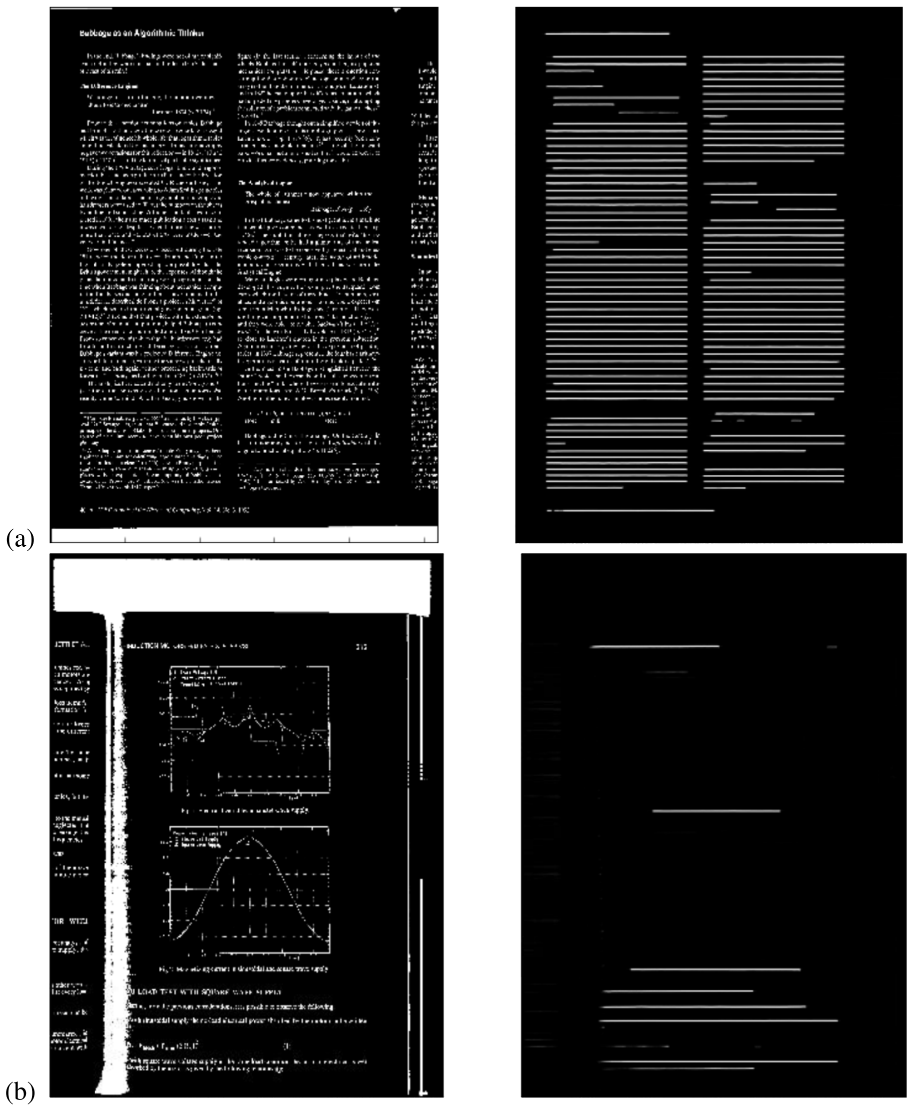

Fig. 2. Full end-to-end page layout analysis with a single deep network. The image shows representative outputs for complex inputs segmented using the text line segmenter. Figure (a) shows a two column document with part of a column from a facing page present in the image. The text line segmenter described in Section II outputs three columns in this case, but the text line segmenter successfully removes the text lines outside the page frame. Figure (b) shows a document containing figures, text-in-figures, text on facing pages, binarization noise, and mathematical formulas. After the text line segmenter successfully labels only the actual text line; this page shows two areas of uncertainty: some mathematical formulas are still assigned a nonzero probability of being text, and the page number is considered ambiguous for being part of the body text line.

- For each text line bounding box in the ground truth, we extract a text line image consisting of the connected components contained in the bounding box.
- The text line image is smoothed with an anisotropic Gaussian of with dimensions of approximately $(h/2, h)$ where h is the height of the bounding box of the text line.
- At each horizontal position, we find the maximum along the vertical; this is an approximate centerline.
- The centerline itself is smoothed with a 1D Gaussian with $\sigma \approx h/3$.
- We construct a binary image representing the centerline.
- The binary image is dilated by a 3 pixels.
- All the images of centerlines are reassembled into a full resolution page image.

Examples of marker line images can be seen on the right hand side of Figure 2 (those marker line images are actually the output from the deep layout analysis model, but since the model learns the task nearly perfectly, they look identical to the training targets).

For all training reported in this paper, text lines higher than 65 pixels (mostly titles) were removed from training and evaluation. This is because large text line recognition using the approach described in this paper is better handled via a multi-resolution approach instead of training a single network to handle both small and large text lines, both because we would need a deeper network to handle a larger range of scales in a single network, and because large text uses different character shapes from small text. Concretely, large text lines can be recognized in practice by training a

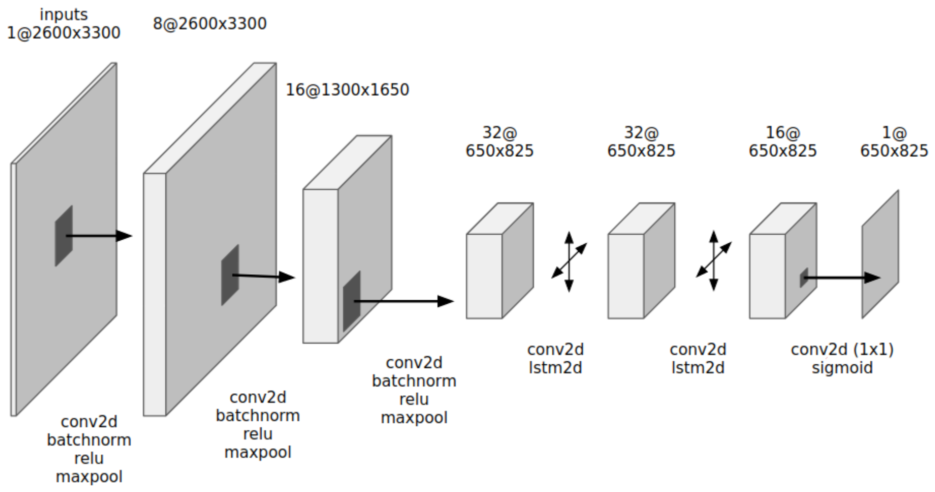

Fig. 3. The deep neural network model for which error rates are reported in the paper. Unlike many other models used for deep segmentation, we did not implement separate upscaling layers, and instead simply scaled up the per-pixel classification output back to the original image size.

separate page segmentation and text recognition pipeline on input images that have been scaled down by a linear factor of 4 but is otherwise identical. The regular, unscaled pipeline covers font sizes from about 5 pt to 24 pt, while the scaled-down pipeline coves font sizes from about 20 pt to 96 pt. For the purposes of this paper, however, we only evaluate the segmentation pipeline for regular-sized text lines.

As already mentioned above, some training was carried out with masked errors. In particular, in some cases, errors were only propagated for a region close to the target centerlines. Such error propagation masks were constructed by applying morphological dilation to the binary centerline images

B. Models

The models used for layout analysis are simple combinations of convolutional, max pooling, and separable multidimensional LSTM layers[8].

The structure of the best model among those tried is shown in Figure 3; this model was found after exploring different numbers of convolutional layers, different image depths, and different numbers of 2D LSTM layers.

Inputs to these models are full, high resolution page images. Outputs are predictions of the target centerline image. Note that the output is lower resolution than the input, so after output prediction, the output is rescaled back to the original page resolution (it would have been possible to use a U-net-like architecture[16], but that was not necessary). The size of the morphological dilation in the construction of the ground truth is chosen such that centerlines are still continuous after the subsampling implied by the max pooling operations.

Training these models is memory limited due to the high resolution of both the inputs and the outputs (a single high resolution page image is the equivalent of a batch size of 2000 for CIFAR-10 training). Most of the memory during training is taken up by the activations and deltas computed and stored during the forward and backwards passes; memory usage during inference is much smaller. The 2D LSTM layer is an essential component of these models, since it can capture and propagate structural information across the entire page. This appears to be important for aspects of page layout analysis like column detection and page frame detection.

The final output nonlinearity is a sigmoid and the networks are trained using an MSE criterion.

C. Segmentation

The output of the deep network model is a per-pixel probability estimate representing the probability that the pixel is part of a centerline. In order to transform this probability map into a page segmentation, we follow the following steps:

- Threshold the output probability map with an empirically determined threshold (0.4 for the best networks).
- Label the connected components in the binary centerline image.
- Assign the centerline labels to all overlapping and nearby connected components.

Text lines are arranged into reading order using the algorithm and implementation described in [2].

D. Evaluation

Several methods have been proposed for document layout analysis. Many such evaluation schemes assume layouts that consist of a nested hierarchy of layout elements; such methods are not well suited for the evaluation of a segmenter like the center line segmenter, both because it outputs non-hierarchical outputs and because it works for distorted and curled text lines. For the evaluation described here, we use a method based on bipartite graphs.

Both the ground truth segmentation and the output of a page segmenter consist of distinct integer labels for subsets page image pixels By overlaying the ground truth segmentation and the segmentation output, we obtain a bipartite graph

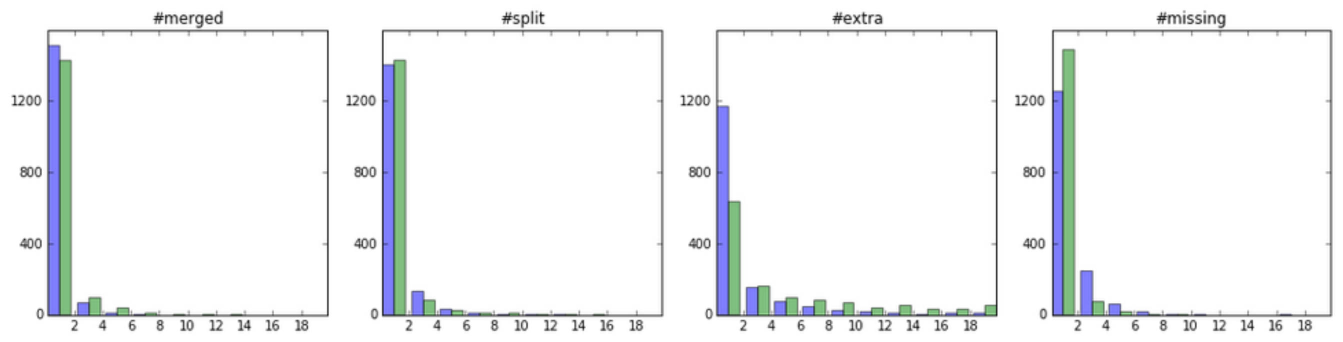

Fig. 4. Histograms showing the number of documents with the number of errors of each type for the deep segmenter (left bars, blue), and the tab-based segmenter (right bars, green). The deep segmenter used only thresholding and proximity-based assignment of connected components to centerlines, as described in the text; there was no further postprocessing. For a perfect segmenter, the leftmost histogram bin would be 1600 (zero errors on each page), with all other bars being zero.

between ground truth labels and segmentation labels. This bipartite graph contains several subgraphs:

- A unique correspondence between truth and output labels (a single edge connection) represents a correct segmentation.
- A ground truth label that corresponds to several output labels represents a split text line (oversegmentation). Split text lines in segmentations often occur when justified text lines contain space characters that have been stretched. They are usually not serious since they occur in paragraphs and do not affect the textual output.
- A segmentation label that does not correspond to a ground truth label represents an extra text line (false alarm). Extra text lines are often detected inside figures and mathematical formulas and can be removed either by postprocessing, or by changing the training method to train not just for text line detection, but also text/image detection and page frame removal.
- An output label that corresponds to several ground truth labels represents a merged or undersegmented text line. Merged text lines are the most serious segmentation errors since they cannot be corrected by later processing stages.
- An ground truth label that corresponds to no segmentation label represents a missing output. Missing outputs are also serious segmentation errors because they can also not be recovered from.

This procedure is essentially the procedure described in [18]. However, we report statistics on the number of documents containing each type of error in addition to the average overall number of errors across the whole database in Figure 6

Evaluations are performed in two different ways. For comparison with other OCR systems, we compute a full text line segmentation of each original binary page using each OCR system and the compute the bipartite graph from the labeled binary images. We can also apply the bipartite graph evaluation above directly on the binary centerline images, which gives very similar results, but is quicker to compute and works even for strongly distorted document images. Errors were estimated by five-fold cross-validation.

III. Results

Deep learning models trained using the above methods yield very low error rates in terms of least square error of the output image relative to ground truth. However, that error measure tells us little about the actual performance of the method compared to traditional layout analysis algorithms. To measure that, we need to actually measure layout analysis errors in terms of split and merged text lines.

As a baseline for the evaluation, we used the Tesseract tab-based segmenter, a widely used open source layout analysis engine that has good performance on printed documents. Whether or not the Tesseract segmenter is the best existing open source segmentation method, it provides an easily reproducible baseline for future experiments and comparisons. The experiments were also limited to the UW3 database of document images, which represents a good cross section of printed documents for performance evaluations and provides the necessary ground truth labels for training.

For the performance evaluation, we trained models using a 10:1 emphasis on propagating errors in the output that are within 20 pixels of the centerline during training. Input training images were binary UW3 images cropped to the actual page frame, resulting in a textline detector without boundary noise removal. This was chosen for the evaluation since Tesseract's tab-based segmenter does not perform boundary noise removal either. Representative output from these models is shown in Figure 2.

All training and testing was carried out on mid-range desktop machines using consumer GTX 1060 (6GB) and GTX 1080 (8GB) graphics cards. The prediction pass is the computationally most intensive part of the segmentation and takes about 0.1 to 0.5 seconds (depending on the complexity of the model) for a 2600 x 3300 pixel image (letter size at 300 dpi). Most of the segmentation overhead is in unoptimized Python code used for loading, transforming, and saving segmentations, but even with all of that overhead included, page segmentation takes less than two seconds per page.

The histograms in Figure 4 compare the performance of the segmenter described here against Tesseract's tab-

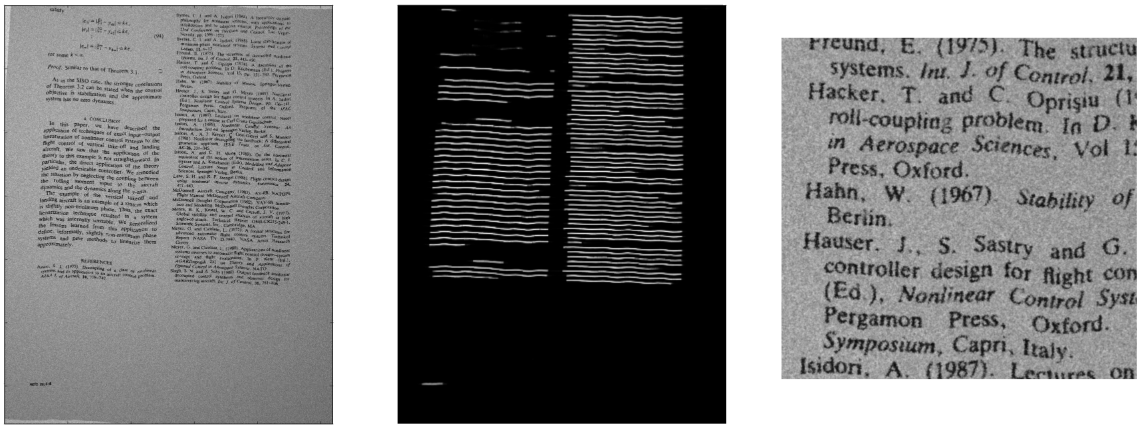

Fig. 5. Text line finding and text/image/math discrimination applied directly to warped, noisy grayscale images. From left to right: (a) grayscale page image with simulated projective transformation, non-linear page warping, an illumination model, and noise, (b) text lines found by the deep text line segmenter, (c) detail from the input page. In addition to text lines being detected correctly, text/math (and text/image, text/graphics discrimination; not shown) also continue to work when operating directly on grayscale images.

	merged	split	extra	missing
Tab Segmenter	0.79	0.91	12.76	**0.32**
Deep Segmenter	**0.30**	**0.65**	**2.83**	1.13

Fig. 6. Average number of occurrences of each error per page for the tab segmenter and the deep segmenter, evaluated on UW3. The only measure that the deep segmenter is worse on is on missing components; these appear to be largely due to text lines consisting primarily of mathematical formulas in running text, which are labeled as text but are otherwise similar to display formulas.

based segmenter[19] on the UW3 database. In addition to the histograms, average numbers of per page errors are reported in Figure 6. Output for the deep segmenter used only thresholding and proximity-based assignment of connected components to centerlines, as described above, with no further postprocessing.

The number of merged text lines is smaller using the deep segmenter than with tab based segmentation. In order to achieve such low error rates on merged text lines, the algorithm has to perform consistently well on segmenting text columns, often a difficult problem, since interword spacing and column spacing can be quite similar. Keeping the error rate for merged text lines low is one of the most important criteria for a good layout analysis method because merged text lines are very hard to recover from in postprocessing.

On splits, the deep segmenter generates fewer splits on average, although the distribution is slightly different from that of the tab segmenter. These splits are mostly due to large amounts of whitespace inside paragraphs in the presence of proportional spacing and large fonts. A second source of such split lines is the fact that UW3 ground truth contains a significant number of components that are spatially widely separated but grouped together on semantic grounds. Split text lines are usually easy to recover from using simple heuristic post-processing, although their number may simply be decreased further with additional training data.

The deep segmenter generates far fewer extra components per page; this appears to be due to its ability to distinguish text zones, math zones, graphics zones, and image zones with high accuracy. The tab-based segmenter appears to pick up components from those zones with some regularity.

The tab-based segmenter misses somewhat fewer text line per page on average. On examination, this appears to be due to the deep segmenter rejecting mathematical formulas in the body text as text line regions. This is not a problem for text recognition (since such regions do not represent recognizable text), and it could be addressed by different training data.

IV. Other Applications

The previous results on text line detection and segmentation have shown deep learning to be a feasible approach for text line segmentation in scanned binary document images, competitive with existing open source layout analysis methods.

Without detailed benchmarking, we wanted to explore two additional questions: whether the methods are applicable to distorted grayscale images like those that might be obtained from digital cameras, and whether the methods can also solve other, related tasks in document analysis, such as page frame detection, text area detection, etc.

Grayscale Images To test layout analysis directly on grayscale images, we generated simulated photographic images using a combination of Gaussian filtering, addition of Gaussian noise, geometric distortion simulating a ruled surface, and raytracing using a simple lighting model combining a point source and background light. Results were qualitatively similar to those obtained from binary images, with low pixel error rates for the segmentation output. An example is shown in Figure 5. This suggests that the method can be applied directly to photographic images, an area that we plan on exploring further.

Other Segmentation Tasks The approach to page layout analysis described above was based on locating text lines directly in the original document image, performing text/image discrimination and border noise removal in the same step. However, by choosing different combinations of image

Input	Target	Mask	Function
framed	centerlines	lines	Text line detector, weak text/image segmenter.
framed	centerlines	none	Text line detector and strong text/image segmenter.
unframed	centerlines	none	Border noise removal, text line detector, and strong text/image segmenter.
photographic	centerlines	none	Photographic text line detector and strong text/image segmenter.
framed	text regions	none	Text/image segmenter.
unframed	text regions	none	Border noise removal and text/image segmenter.
unframed	framed	none	Border noise removal.
unframed	unframed - framed	none	Border noise detection.

Fig. 7. Different combinations of input images, output images, and masks for training result in different functional blocks for document analysis. The table shows combinations that have shown to be feasible and that are useful for different applications. Note that text/image segmentation includes text/math segmentation.

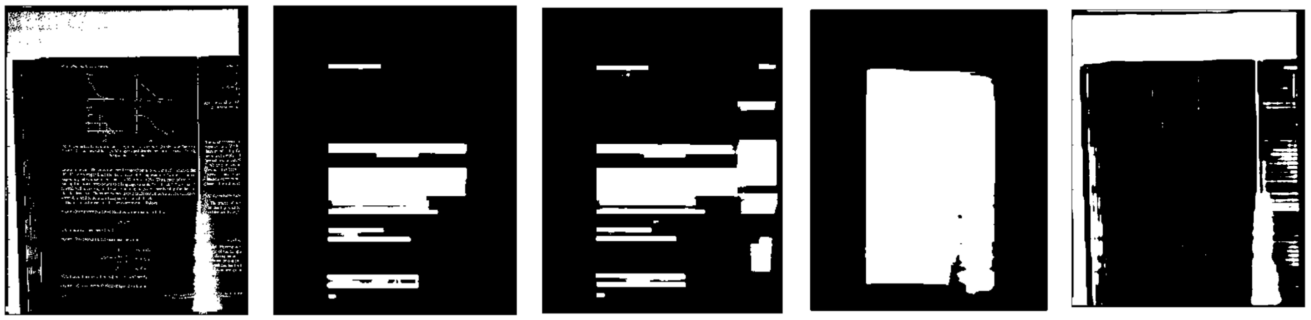

Fig. 8. Other layout analysis tasks solved by deep segmenters. From left to right: (a) raw input image, (b) a model trained for combined text area and page frame detection, (c) a model trained for text area detection only, (d) a model trained for page frame detection only, (e) a model trained for boundary noise detection. Note that the figure on the left contains a number of common problems in layout analysis: massive boundary noise, text on facing pages, line drawings containing text, and mathematical formulas. The text/non-text discrimination networks successfully detect the actual text regions, excluding text in figures, formulas, and on facing pages. The page frame and boundary noise detectors successfully detect their respective targets.

preprocessing, construction of training targets, and weight masking during training, we can easily perform a number of other common layout analysis tasks, such as page frame detection, text area detection, border noise removal, etc. Combinations of inputs, targets, and masks, and their corresponding functions are shown in Figure 7. Representative results of training with such image/target/mask combinations are shown in Figure 8.

V. Discussion

The paper has described a simple layout analysis system that reduces the problem of layout analysis to learning an image-to-image transformations using a combination of deep convolutional networks and MDLSTM networks, followed by a simple assignment of connected components to the thresholded output of the network. Except for the usual deep learning hyperparameters (number of layers, learning rate, depth), there are no document analysis related parameters or thresholds. Furthermore, the method requires no document cleanup and can deal with noise, boundary noise; it can even operate on grayscale inputs with variable illumination. An evaluation against a widely used open source system shows the system to have comparable performance even when trained on a fairly small (by deep learning standards) dataset.

Preliminary observations suggest that the same architecture also works for other layout analysis tasks, like image detection, table detection, line drawing detection, frame and box detection, etc. Performance is likely going to improve significantly with larger training sets. In particular, for global page properties, such as column structure, placement of images, and page frames, the effective number of training samples is only 1600 instances. In contrast, for the local detection and localization of text lines, the training set effective represents about 100000 “training samples”, in the form of individual text lines.

The availability of a fully trainable layout analysis engine with very simple training data requirements (only text lines need to be marked) opens up possibilities for rapid improvements in layout analysis performance. Synthetic training data can be generated quite well for OCR applications, using type setting systems, automated cut-and-paste, and image degradation models. A synthetic dataset often will give sufficiently low error rates for many documents, but markup of additional documents is easy, since the deep segmenter can be run on new datasets and usually only small amounts of corrections (breaking up lines, connecting lines, erasing lines) are necessary and are intuitive and easy even for non-experts to carry out. In addition, given that robust, low error rate text line recognizers are now available as well, self-supervised training and improvements become possible. For example, determining whether a detect text line is true or false is possible by running the text line through the OCR engine and testing the output against a language model. Language models can also help with reading order determination even in the absence of ground truth.

The ability of the deep segmentation algorithm to run well on noisy grayscale inputs with variable illumination also makes it attractive for camera-based document capture and facilitates the use of text lines for reliable rectification and dewarping of such documents.

The deep segmenter described in this paper will be released in open source form at `www.ocropus.org`, in hopes that it will enable people to develop layout analysis systems for other languages, scripts, and document types quickly.

References

[1] Henry S Baird, Susan E Jones, and Steven J Fortune. "Image segmentation by shape-directed covers". In: *Pattern Recognition, 1990. Proceedings., 10th International Conference on*. Vol. 1. IEEE. 1990, pp. 820–825.

[2] Thomas M Breuel. "The OCRopus open source OCR system". In: *Electronic Imaging 2008*. International Society for Optics and Photonics. 2008, 68150F–68150F.

[3] Thomas M Breuel. "Two geometric algorithms for layout analysis". In: *International workshop on document analysis systems*. Springer. 2002, pp. 188–199.

[4] Thomas M Breuel et al. "High-performance OCR for printed English and Fraktur using LSTM networks". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE. 2013, pp. 683–687.

[5] Syed Saqib Bukhari, Faisal Shafait, and Thomas M Breuel. "Towards generic text-line extraction". In: *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE. 2013, pp. 748–752.

[6] Wonmin Byeon et al. "Scene labeling with lstm recurrent neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3547–3555.

[7] Kamran Etemad, David Doermann, and Rama Chellappa. "Multiscale segmentation of unstructured document pages using soft decision integration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.1 (1997), pp. 92–96.

[8] Alex Graves and Jürgen Schmidhuber. "Offline handwriting recognition with multidimensional recurrent neural networks". In: *Advances in neural information processing systems*. 2009, pp. 545–552.

[9] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. "Recursive XY cut using bounding boxes of connected components". In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 2. IEEE. 1995, pp. 952–955.

[10] Robert M Haralick. "Document image understanding: Geometric and logical layout". In: *CVPR*. Vol. 94. 1994, pp. 385–390.

[11] Jianying Hu, Ramanujan Kashi, and Gordon Wilfong. "Document classification using layout analysis". In: *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*. IEEE. 1999, pp. 556–560.

[12] Max Jaderberg et al. "Reading text in the wild with convolutional neural networks". In: *International Journal of Computer Vision* 116.1 (2016), pp. 1–20.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[14] Simone Marinai, Marco Gori, and Giovanni Soda. "Artificial neural networks for document analysis and recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.1 (2005), pp. 23–35.

[15] Lawrence O'Gorman. "The document spectrum for page layout analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.11 (1993), pp. 1162–1173.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 234–241.

[17] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[18] Faisal Shafait, Daniel Keysers, and Thomas M Breuel. "Pixel-accurate representation and evaluation of page segmentation in document images". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. Vol. 1. IEEE, pp. 872–875.

[19] Raymond W Smith. "Hybrid page layout analysis via tab-stop detection". In: *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE. 2009, pp. 241–245.