Using Neural Cells to Improve Image Textual Line Segmentation

Patrick Schone; Family Search, Salt Lake City, Utah.

Abstract

Before one can begin applying automatic transcription processes to a document image, a line segmentation algorithm usually must be applied first in order to identify the individual text lines upon which recognition will be performed. Most line segmentation algorithms use standard image processing techniques and/or statistics to identify inking activity in the image. Unfortunately, these algorithms have no awareness of inking that is intentional (such as that written by an author) versus that which is merely background, darkness of copy, noise, etc. Moreover, the algorithms themselves cannot always tell when lines have been over- or under-segmented. Neural networks can be taught to learn such distinctions. We have trained a neural process which can detect image line phenomena and can be used to improve automatic line segmentation. To the best of our knowledge, no other researchers have demonstrated similar processes. We observe definite accuracy gains using our neurally-enhanced line segmenter over a previous high quality line segmenter, and we believe such improvements will apply when using other line segmentation algorithms.

1. Background on Line Segmentation

When transcription of document images is a user's goal, it is almost always a requirement to first transform the document into the sequence of its textual lines so that subsequent processing can then proceed on a line-by-line basis. The process of transforming a document into its constituent lines is referred to as *line segmentation*. The vast majority of line segmentation techniques leverage either statistics or common image processing techniques which include projection-based methods, the finding of connected components, smearing approaches, clustering, and Hough transforms (see, for examples [1], [2]). Other techniques also exist which try to use more sophisticated approaches such as peak and trough detections and carving seams between them, hidden Markov model-like analyses, and graphical approaches (see [1], [3], [4]). These methods provide good or even excellent results depending on the image collection, especially when collections are small or homogeneous.

Yet when the images in a collection number in the millions or billions and are quite diverse, as in a genealogical data set, typical line segmentation algorithms fail to account for many of the phenomena that will occur. This is largely because these systems are designed to identify contrastive areas of pixel darkness and/or pixel connectivity as opposed to trying to capture what we will refer to here as *deliberate marks*: markings on the page which were intentionally placed there by the document's author.

Let us consider some situations where line segmentation can be problematic for activity-based or connectivity-based algorithms:

a) In the presence of faint copy where inking is weak, systems that rely on connectivity get confused because single lines of text can appear as disconnected. This can result in falsely detecting multiple lines. Activity detectors, on the other hand, may treat these areas as having insufficient information to even predict the appearance of some of the lines.

b) Speckle noise can cause challenges for a connectivity-based algorithm since it may be treated as myriad regions of disconnected components and thus over-generate lines.

c) If textual lines have significant overlap (where, say, the descenders from one line intersect frequently with ascenders from below), connectivity-based algorithm may falsely merge these lines together. Activity-based algorithms, on the other hand, can treat such regions of overlap as if they were additional lines since there may be sufficient pixel darkness to warrant it.

d) When the image is copied from a book or on a black background resulting in the image having side regions with dark vertical streaks, the activity algorithms can end up over-generating lines if the streaks are of variable thicknesses.

e) If the slope of the text lines drifts from one side of the page to the other, both kinds of algorithms can either over-generate lines or can produce hypothesized lines where the left half and right half of the lines actually are drawn from different textual lines.

This list can be extended. Yet it suffices as an illustration that the base algorithms, absent a search for deliberate marks, are likely to have challenges with certain kinds of documents.

In this paper, we demonstrate that deep neural networks can be built which can identify deliberate marks and which can be used to improve a system's ability to properly find textual lines. We demonstrate through this effort that not only is the error rate of line segmentation reduced through our neural processes, but we actually train a handwriting recognition system on a very large corpus of US legal documents and show that the resultant line segmentation improves performance by 1.2% absolute (7.5% relative reduction in error).

2. DNNs to Count Lines

Prior to the work described in this paper, we had created a line segmentation algorithm (which we will refer to here as ***PreDNN***) which leverages many of the leading techniques mentioned in Section 1. The base components of our algorithm are described elsewhere (see [5]), but suffice it to say that it uses multi-swath projections to identify activity peaks, dynamic programming to carve seams between those activity peaks, and overlap of connected components to detect falsely merged lines. These processes were then extended using statistical analyses to find and remove the over-generation of detected peaks and to trim out falsely carved seams. The resultant algorithm seems to have quite high on a huge collections of generic *printed* documents and has accuracy (perhaps about 85%-90%) on a collection of tens of thousands of *handwritten* document spanning four centuries. We expect that the results of the PreDNN system rival those of state-of-the-art and we have observed that they are quite usable. That said, leaving out up to 15% of a document's content is undesirable.

2.1. Line Count Cells

Deep neural networks provide a potential means to overcome some of these gaps. We reasoned that if we could build a neural network which, given a snippet of an image, could predict if the snippet contains no text lines, fractions of a line, multiple lines, or exactly one line, then such a network could be used to either completely perform line segmentation or to improve upon an existing line segmentation algorithm.

To test this hypothesis, we assembled a mixed corpus consisting of printed Latin-script texts, Chinese print, and Latin-script handwritten documents. From these, we automatically generated a huge collection of image snippets of variable sizes whose edges were not necessarily straight. These snippets were then condensed down into small cells of size 30 pixels by 30 pixels. We tagged each of these 30x30 cells with one of seven different tags based on the number of lines observed in each cell. As shown in Figures 1a-g, these snippets fall into categories of:

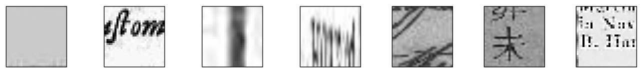


Fig1.a Fig1.b Fig1.c Fig1.d Fig1.e Fig1.f Fig1.g

(a) no-text-lines, (b) single-text-line, (c) vertical-bar-only, (d) less-than-one-text-line, (e) two fragment lines, (f) more-than-one-but-fewer-than-two, and (g) two-plus-lines. The colors that are selected here are deliberate in that they will be used throughout this paper in colorized images representing the various classifications.

Our collection of tagged cells currently consists of 70.5K different images of which 65,259 are used for training and 5,277 are used for testing. No specific effort was made to have a comparable number of each of the seven classes of cells, so the final distribution is reflective of what is observed in practice. Table 1 shows the actual distribution by category in both the training and testing sets:

Table 1: Distribution of Counts of Tagged Cells

CATEGORY	# in TRAIN	# in TEST
no-text-lines	7330	625
single-text-line	12651	848
vertical-bar-only	703	89
less-than-one-text-line	16813	1098
two fragment lines	5027	407
more-than-1/fewer-than-2	8573	801
two-plus-lines	14162	1409

An interesting thing about tagging cells in the way specified is that the tags remain the same even if the cells are flipped with respect to the y-axis, or if they are rotated 180 degrees. That means that if we consider these four permutations, we can likewise multiply the size of our collection by four resulting in a training set with 280K elements and a test set with 21K.

2.2. Training a DNN for Line Counts

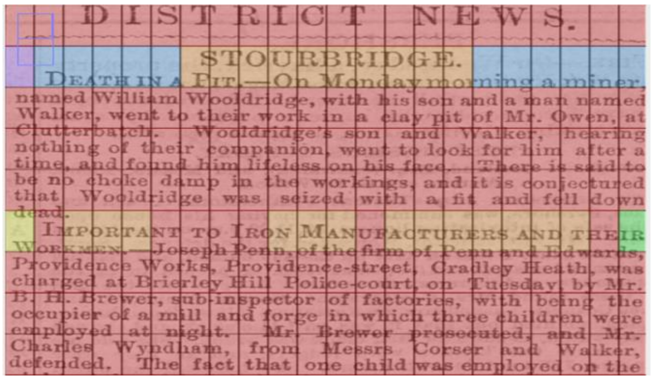
With training cells available, we can now train a deep neural network (DNN) to try to predict the line count of future image cells. To train our DNN, we make use of Google's open source Tensorflow [6] engine. Additionally, the Tensorflow developers created a "recipe" using convolutional neural networks (CNN) as applicable to the MNIST digit-recognition task which we have modified for our task. We use the same number of layers in our network as they have, but we use a kernel size of 3, and our first through third hidden layers have, respectively, 16, 32, and 216 nodes. Each of our layers are smaller than those of the recipe, but this is beneficial for recognition speed (which seems essential for our task).

Our particular DNN topology yields an average tagging accuracy of 91%. We can get up to 2% better prediction accuracy by doing one of the following: (a) on a per cell basis, apply the DNN predictor to each of the four legal permutations of that cell and vote, or (b) create cells from overlapping rectangular regions and keep the result if the prediction for both regions agree or otherwise use the prediction of the overlap region as a tie breaker. Since computational cost is a factor, method (a) is somewhat less desirable because one must perform four times more computation for a 2% gain. On the other hand, method (b) can be performed for only about 10% more cost than just doing cell-by-cell evaluation because the rectangles take the place of the cells and the tie breakers only need to be applied when there is disagreement. For this reason, we use method (b) throughout our computations unless stated.

3. Take 1: Line Counters As Line Segmenters

Once we have a DNN-based line count predictor, the next question is: how does one best apply these to the task of line segmentation? One option is to let the DNNs do the entire process. The thought here is that if the predictions are high quality, then one should be able to split up an image into a bunch of non-overlapping regions. Then, for a given region, if the DNN predicts that there is more than one line, it should be feasible to do a kind of binary search of splits through that cell until the system predicts that one of the splits has exactly one row. Then one can recurse over the remaining portion of that region until all the individual lines are detected. On the flip side, if the system predicts that there is less than one line, one should be able to merge with residuals from above or below the region in order to form a single cell.

We took a snippet of a newspaper image, split it into 9x22 non-overlapping regions, converted each region to a 30x30 cell, and made a DNN prediction on each cell to yield the image in Figure 2. The colors shown in the image are those indicated above in the descriptions of Figures 1a-g.


DISTRICT NEWS.

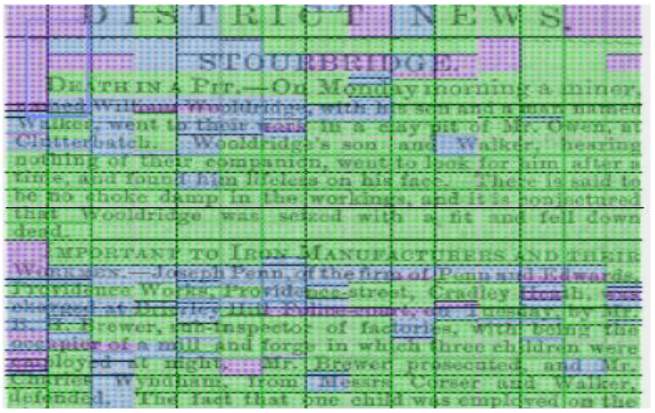
STOURBRIDGE.

DEATH IN A PIT.—On Monday morning a miner, named William Wooldridge, with his son and a man named Walker, went to their work in a clay pit of Mr. Owen, at Clutterbatch. Wooldridge's son and Walker, hearing nothing of their companion, went to look for him after a time, and found him lifeless on his face. There is said to be no choke damp in the workings, and it is conjectured that Wooldridge was seized with a fit and fell down dead.

IMPORTANT TO IRON MANUFACTURERS AND THEIR WORKMEN.—Joseph Penn, of the firm of Penn and Edwards, Providence Works, Providence-street, Cradley Heath, was charged at Brierley Hill Police-court, on Tuesday, by Mr. B. H. Brewer, sub-inspector of factories, with being the occupier of a mill and forge in which three children were employed at night. Mr. Brewer prosecuted, and Mr. Charles Wyndham, from Messrs Corser and Walker, defended. The fact that one child was employed on the

Figure 2: Image tagged completely with neural decisions

We next applied the recursive splitting method as described on any regions which were tagged as having more than one line. The result is shown in Figure 3. Encouragingly, we see that this process is able to segment out some textual lines (i.e., lines that are consistently green and pink). On the other hand, we also notice that many of the splits result in disconnected lines and we see that some of the narrow residual regions are tagged incorrectly. Moreover, if this image had drifting text lines as opposed to the nice horizontal ones we observe, it is unclear that splitting each cell using horizontal cuts would even be able to properly segment the image. To top it off, this process is computational expensive because every cell needs to be split repeatedly and both the top and bottom splits needs to be evaluated after each split.



DISTRICT NEWS.

STOURBRIDGE.

DEATH IN A PIT.—On Monday morning a miner, named William Wooldridge, with his son and a man named Walker, went to their work in a clay pit of Mr. Owen, at Clatterbatch. Wooldridge's son and Walker, hearing nothing of their companion, went to look for him after a time, and found him lifeless on his face. There is said to be no choke damp in the workings, and it is conjectured that Wooldridge was seized with a fit and fell down dead.

IMPORTANT TO IRON MANUFACTURERS AND THEIR WORKMEN.—Joseph Penn, of the firm of Penn and Edwards, Providence Works, Providence-street, Cradley Heath, was charged at Brierley Hill [illegible], on Tuesday, by Mr. B. H. Brewer, sub-inspector of factories, with being the occupier of a mill and forge in which three children were employed at night. Mr. Brewer prosecuted, and Mr. Charles Wyndham, from Messrs. Corser and Walker, defended. The fact that one child was employed on the

Figure 3: Post-splitting image

This technique may have some merit, but for the moment, it seems that its challenges outweigh those merits. We need to ensure that we have a process that is fast enough to use, but a process which is also less sensitive to errors in cell-level predictions or to non-horizontal text. So we try another approach.

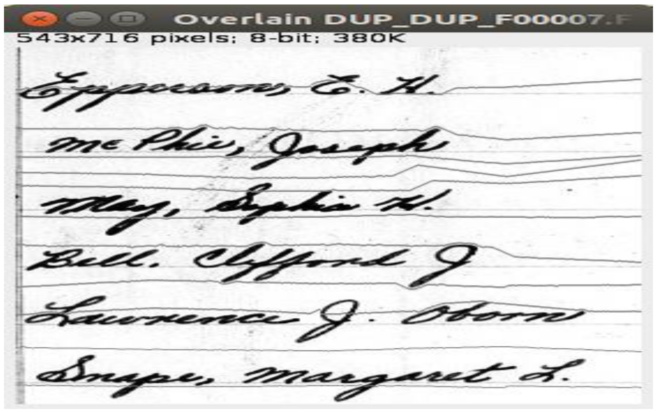
4. Take 2: Line Counters As Supplementer

Suppose, then, that rather than trying to use the neural cells to segment the whole image that we use them to try to identify and fix problem areas as a follow-on from a normal line segmentation process. This has a number of benefits. For one, the basic algorithms tend to be fast, so this should reduce the time requirements. Moreover, they are not as affected by the "horizontal-ness" of the text. Likewise, if the system says the original lines are appropriate, no subsequent work is required but we have much more confidence in the original decision.

4.1. Neural Cells for Error Detection / Greenness

With this in mind, we run PreDNN on raw images to detect potential lines. Although PreDNN does a good job with line segmentation, we show in Figure 4 an image with bad PreDNN segmentation so that we can later see the effects of the neural processing. This handwritten document does have drift on some lines and some ascender-descender regions which make the PreDNN system believe it should generate more lines than should exist. But whatever the reason, there are line segmentation errors which would affect recognition and we would like to remove those.

Figure 4: Example of bad line segmentation from PreDNN



Overlain DUP_DUP_F00007.F

543x716 pixels; 8-bit; 380K

Epperson, E. H.

McPhie, Joseph

May, Sophia H.

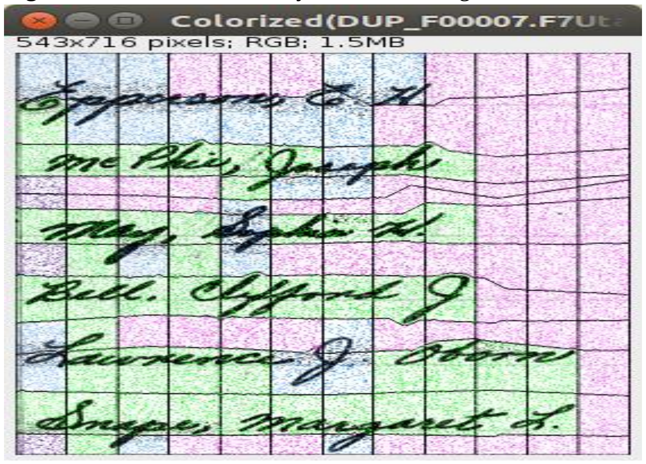
Bell, Clifford J

Lawrence J. Oborn

Snyder, Margaret L.

To take advantage of the neural predictions, we treat each region delineated by a line as a swath which we slice into overlapping cells whose tops and bottoms follow the contours of the line segments and the right and left edges are parallel and vertical. These odd-shaped regions are repackaged to become 30x30 cells and the neural net (with overlapping and voting) is used to predict the content of each cell. The outcome of this is shown in Figure 5.

Figure 5: PreDNN followed by Neural Coloring



Colorized(DUP_F00007.F7Ut

543x716 pixels; RGB; 1.5MB

Epperson, E. H.

McPhie, Joseph

May, Sophia H.

Bell, Clifford J

Lawrence J. Oborn

Snyder, Margaret L.

As can be observed, the neural network is able to identify, with reasonable accuracy, background regions (pink), single lines of text (green), and, in this case, lines that appear to be split (blue). If we consider weighting the cells with pink or purple as 0.1 and everything else as 1.0, then green+pink+purple here represents 69.9% of the cells. We will call this metric the "greenness." We would like the greenness to be much closer to 100%, so even before making any corrections, we still know that the initial line segmentation yielded a fairly poor result. (It should also be mentioned that the dark shading within a cell near the line boundaries is an effect which we automatically add based on a computation of the cell's pixel darkness and its proximity to the line boundary.)

4.2. Neural Cells for Error Correction

Although error detection is potentially useful, especially if humans can be used for fixing bad line segmentations, it would be beneficial, where possible, for the system to fix itself. Part of the challenge with this is in the recognition that we do not have perfect cell prediction, which means that many of the so-called errors are not line segmentation errors at all, but rather, are cell prediction errors. So until the cell prediction accuracies increase, we need to limit automatic correction to those areas which appear consistently to be bad. This means we will predominantly only be able to fix full or partial lines that appear to be broken.

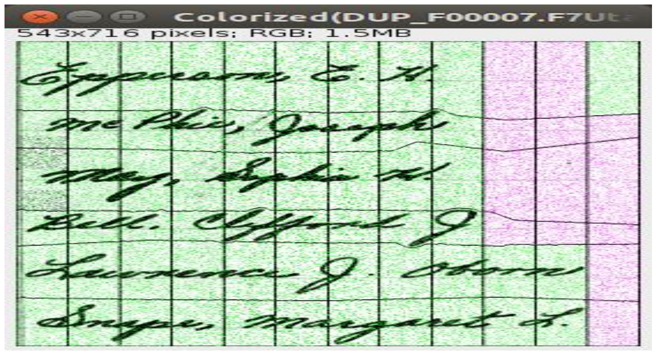
We saw in Figure 5 that there were a number of rows where there are quite a few blue cells in the row. This is likely indicative of over-segmentation. If there is over-segmentation, we would expect that two juxtaposed rows would have a high frequency of blue cells (or even pink ones in regions of the image where we might hope to see green). If there were under-segmentation, we would expect a single row to have more reds, oranges, or yellows.

Using these observation, we created two sets of formulas – one for potential fusion and one for potential splitting – where we count and weight the frequency of certain types of cell juxtapositions and decide whether we should do subsequent analysis. For example, if a cell is blue as is its vertical neighbor, these facts will contribute to a potential need for fusion. If the cell is blue and its vertical neighbor is pink, this also contributes, though less strongly, to the potential need for fusion. On the other hand, if the cell is green as is its vertical neighbor, this suggest that fusion is potentially undesirable. If the vertical neighbor is green but this cell is red, then perhaps line splitting is in order.

After each formula is applied to each row pair, the scores from those formulas are sorted and the highest-scoring outcomes are processed first. If the formula indicates that merging should potentially happen, the two rows are merged offline; or if splitting should happen, PreDNN is applied to that row offline. (By "offline" we mean that these images are handled separately from the original image.) These offline swaths are re-cellularized, and the cells are colorized. If the offline colored swaths are "greener" than the original swaths had been, the offline swaths are introduced into the main image and the original line segmentations are discarded.

This process provides images which almost consistently have equal or better line segmentation than original. Figure 6 shows the outcome of this process as applied to the data in Figure 5. These updates, which we will call *PostDNN*, are now used as the improved line segmentation. Note that greenness is now 100% giving high confidence that the image is properly segmented – or, at least, that it has improved. We have seen that the greenness measure actually provides a very reasonable estimate of how well the line segmentation has worked; and the updated line segmentation improves recognition capability.

Figure 6: PostDNN outcomes



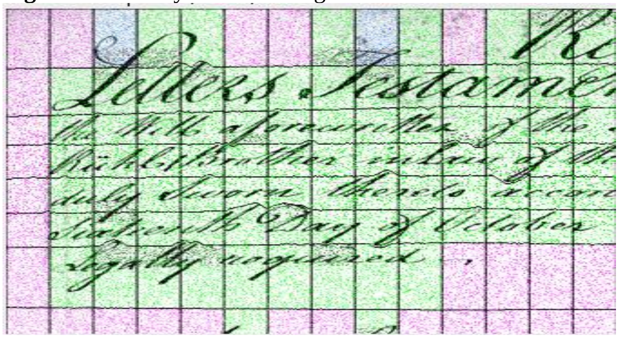
4.3. Still on the To-do List?

This ability to repair fully-merged and fully-fractured lines is a great benefit because these situations are the ones that likely represent the largest losses in overall recognition accuracy. When a line is split in two, the best case scenario for recognition is that no words will be recognized, and the worst is that a bunch of spurious words will be produced. When two lines are merged, a recognition system will obviously fail to recognize the words from either line, and has a pretty reasonable chance of reporting spurious elements. Thus, we would expect that the error fixes we have described in the PostDNN system should result in reasonable improvements in recognition capability.

That said, there are a number of situations where major or frequent errors are detected but for which we still do not quite know how to improve the lines. The fact that we are only getting 91-93% accuracy in cell tagging precludes us from fixing all of the potential issues since many of those issues are not line segmentation errors at all, but are just faulty cell-tagging. Yet there are still situations where there are numerous cells which are neither green nor pink, and yet it is still unclear how to move forward. We identify them here for completeness but place them in the arena of "still-to-do":

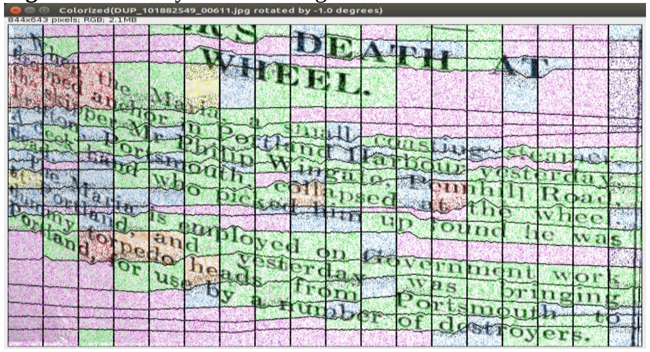
[a] Line Bumping: Perhaps the most common phenomenon in terms of errors is the occurrence of lines which are mostly correct but which, at some point, cut through some words. We have introduced functionality into the algorithm which allows the system to "bump" a line above or below its current location with the intent of avoiding character splitting. However, at this point, we have not found this approach particularly helpful because the neural prediction accuracy is not close enough to 100% -- meaning that many times the so-called corrections end up breaking something that was accurate but mislabeled. Figure 7, for example, shows an image where the lines have sliced through some of the words (the "L" in Letters and the "T" is Testamentary). The blue cells identify the presence of potential issues, and the dark shading, which was mentioned earlier, suggests that the blue areas probably belong to the rows below them. However, in this case, the green cells adjoining the blue cells are properly tagged since the strokes in those cells are fully bounded above and below by the corresponding line segments. This makes it difficult to know how to perturb the line correctly.

Figure 7: Neurally-Colored Image When Line Bisects Words



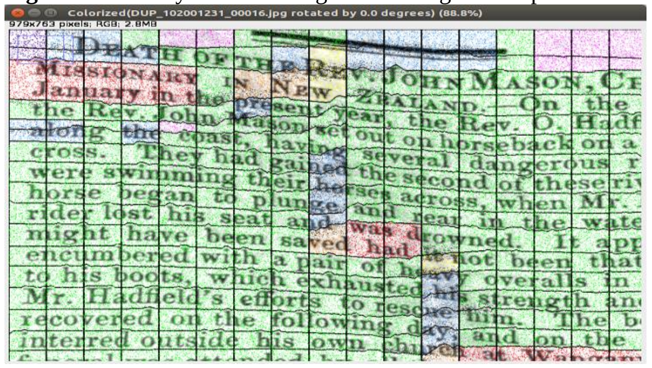
[b] Line straightening: A more concerning problem is when the image has warping in some way such that the text lines have large curvature. In such cases, line segmentation algorithms can fail. The neural prediction is an excellent "error detector" in that it can identify clearly that something is amiss in the line segmentation. In Figure 8, for example, we can see that the left side of the image has almost no green cells, whereas the right side of the image has a number of columns where most of the cells within the column are green or pink. Despite this beneficial error detection, we have yet to identify a clear mechanism which would allow us to convert an initial segmentation as in Figure 8 into a well-segmented image.

Figure 8: Neurally-Colored Image with Curved Lines


Colorized(DUP_101882549_00611.jpg rotated by -1.0 degrees)
844x643 pixels; RGB; 2.1MB

[c] Image "Unwarping": The last major issue which still needs to be resolved is that of images that are wrinkled like a flag blowing in the wind. An example of this is seen in Figure 9. As can be seen here, the left and right sides of the image have areas with mostly green cells, but in the center, there appears to be a disruptive region where the cells are almost all non-green. As with curved lines, the neural components are able to detect the presence of errors, but we have yet to decide on a process for fixing such situations.

Figure 9: Neurally-Colored Image when Image is Warped



Colorized(DUP_102001231_00016.jpg rotated by 0.0 degrees) (88.8%)
979x763 pixels; RGB; 2.8MB

DEATH OF THE REV. JOHN MASON,
MISSIONARY IN NEW ZEALAND.—On the
January in the present year, the Rev. O. Hadf
the Rev. John Mason set out on horseback on a
along the coast, having several dangerous r
cross. They had gained the second of these riv
were swimming their horses across, when Mr.
horse began to plunge and rear in the wate
rider lost his seat and was drowned. It app
might have been saved had he not been that
encumbered with a pair of heavy overalls in
to his boots, which exhausted his strength an
Mr. Hadfield's efforts to rescue him. The b
recovered on the following day, and on the
interred outside his own church at W

5. Evaluating Segmentation Improvements

There are three ways in which we can evaluate any improvements in line segmentation between the PreDNN and the PostDNN. We could tag a truth set of lines and determine how much better PostDNN is as compared to those, but this is an expensive endeavor. Another mechanisms is to compare the amount of "greenness" between the two data sets. A third option is to run recognition with PreDNN and PostDNN and see what happens to recognition. The second option is convenient and easy, so we will do that here. On the other hand, since it is not a direct measure and since the system we are creating here tries to directly improve greenness, it is somewhat "cheating" to only evaluate greenness. So we will also use the third mechanism and see if the PostDNN has any added benefit on what we really care about: recognition performance.

5.1. Evaluation by Comparing Greenness

To compare greenness, we selected 219 images of documents that were either English newprint or English handwriting. Since handwriting has more issues for line segmentation than print, there is a predominance of handwriting in our set. Other than that bias, the images were drawn mostly randomly. Table 2 shows various elements regarding "greenness" in both the PreDNN and in the PostDNN system. The first five columns show the number of documents that have certain levels of greenness and the last row shows the average percentage of greenness. Note that the average greenness increased by 4.6% absolute. This is substantially better!

Table 2: "Greenness" Comparisons: Before and After

	PreDNN	**PostDNN**
# of 95%+ green	69	92
# of 90-94% green	47	58
# of 80-89% green	52	46
# of 70-79% green	31	14
# Below 70% green	20	9
Average Greenness	86.88%	91.30%

That said, there is still clearly room for further improvement. Though 100% may not be attainable, it is likely that one could continue to move the amount of greenness by at least 5% absolute if the neural decisions were more correct and if one had proper mechanisms for handling the various phenomena that might arise. By the same token, it is not clear how much that additional greenness – or the amount we have recovered up to this point -- relates to better accuracy. So we consider that next.

5.2 Evaluation by Comparing Recognition

The ultimate proof of the effectiveness of this approach is to see how it actual plays out in terms of actual recognition rates. Doing this also allows one to get some sense of how greenness increases relate to increases in accuracy – if at all.

To evaluate this, we created a test collection of 565 US legal, fully-prose documents which are drawn from hundreds of US counties across a four-hundred-year time span. This set contains 137K words and is very representative of the kinds of documents that might be found in a genealogical collection. We also trained a handwriting recognition system using a separate, but similar, set of documents. We applied our recognition system to the test documents using line segmentation that comes both from the PreDNN and the PostDNN line segmenters. The results are shown in Table 3.

Table 3: Handwriting Recognition: Before and After

	PreDNN	**PostDNN**
Word Accuracy	83.9%	85.1%

As is clear from Table 3, the PostDNN system is better accuracy-wise than the PreDNN system. In order to attain this accuracy gain, we needed to expend an additional 19% more time in the line segmentation phase. Yet there end up being fewer false lines, which saves 11% in recognition time. Consequently, for a comparable computation costs, we get a nice 1.2% absolute (7.5% relative) reduction in recognition errors using this process.

6. A Word on Other Neural Techniques

Before concluding, it should be mentioned that this paper does not represent the first paper where neural processes have been used to try to perform line segmentation. Interestingly, though, despite the overwhelming success of deep neural networks, there are surprisingly few in the line segmentation arena. Yet one such paper is worth noting by Bluche, et al [7]. In it, the authors actually attempts to do handwriting recognition without the benefits of any line segmentation and instead uses a multi-directional LSTM for recognition instead of bi-directional LSTM. This work is noteworthy in that the author is able to get the system to perform recognition, but they appear to suffer a 60% relative drop in accuracy – which is certainly something that one could not afford in practice. Consequently, we did not use those techniques for our work, but which are definitely interested in any follow-on work.

7. Conclusions

In this paper, we have demonstrated that neural cells can be used to not only detect errors in high-quality line segmentation algorithms, but they can also be used for correcting such errors. We were able to show an approximate 5% improvement in line segmentation which yielded 1.2% accuracy gain in handwriting recognition. We think these approaches could also be applied to others' line segmentation approaches.

8. Acknowledgements

The author expresses appreciation to FamilySearch's partner companies for any images which have been used in this paper for which they were the originators. He also wishes to thank Ian James for Ian's early internal work with image zoning which bears resemblance to some of the work in this paper.

References

[1] L. Likforman-Sulem, A. Zahour, B. Taconet "Text Line Segmentation of Historical Documents: a Survey," *ICDAR*, 2006

[2] R. Garg, N Garg, "Problems and Review of Line Segmentation of Handwritten Text Documents," *Int'l Journal of Advanced Research in Computer Science and Software Engineering*, Vol 4: 4, Apr 2014

[3] D.J. Kennard, W. A. Barrett, "Separating Lines of Text in Free-Form Hand-written Historical Documents," *DIAL* 26, Lyon, France 2006.

[4] N. Arvanitopoulos, S. Süsstrunk. "Seam Carving for Text Line Extraction on Color and Grayscale Historical Manuscripts," ICFHR, 2014.

[5] P. Schone, A. Cannaday, S. Stewart, R. Day, J. Schone. "Automatic Transcription of Historical Newsprint by Leveraging the Kaldi Speech Recognition Toolkit," *DRR*, San Francisco, 2016.

[6] M. Abadi, et al. "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[7] T. Bluche, J. Louradour, R. Messina. Scan, Attend, and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. https://arxiv.org/pdf/1604.03286.pdf, 2016.