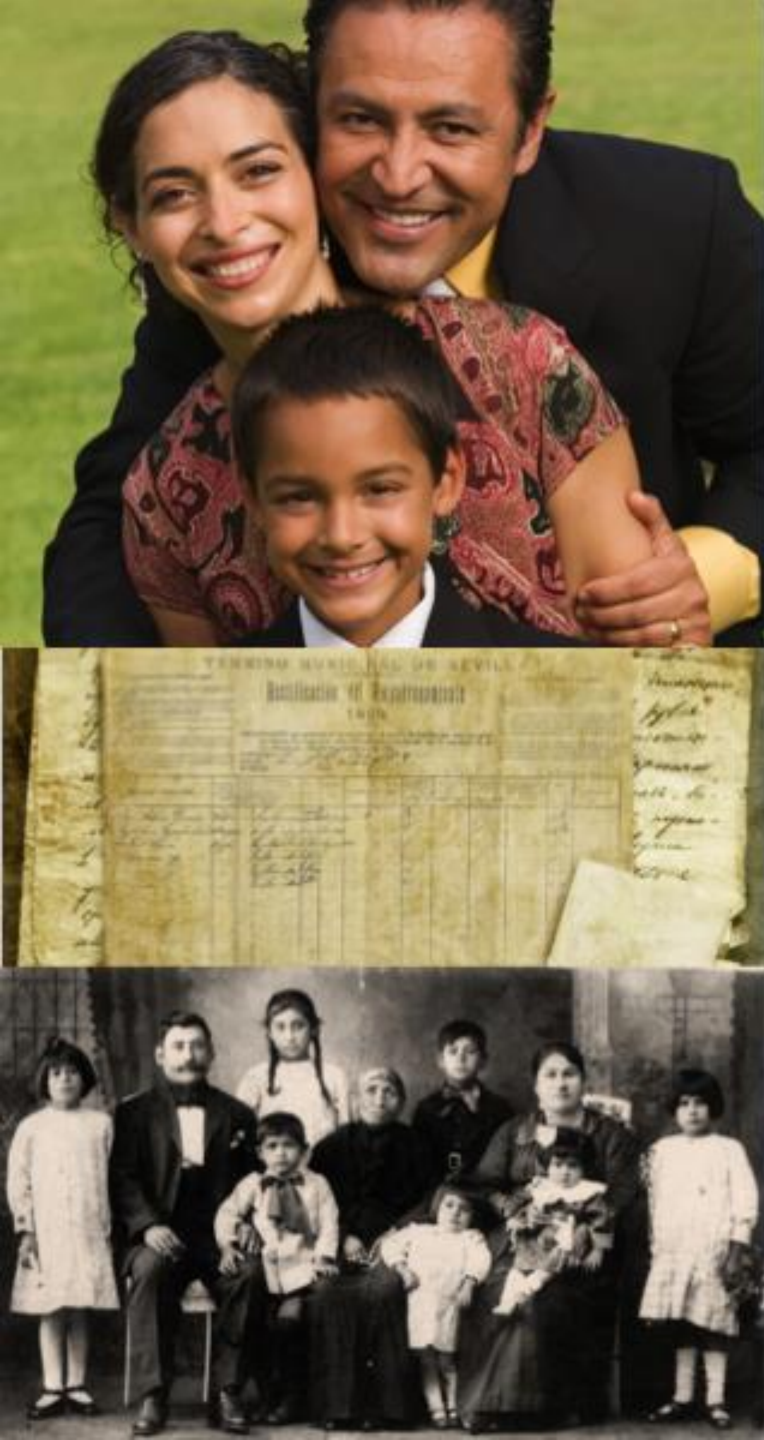


Using Neural Cells to Improve Image Textual Line Segmentation

Patrick Schone
(patrickjohn.schone@ldschurch.org)

7 February 2017

THE CHURCH OF JESUS CHRIST OF LATTER-DAY SAINTS

FAMILYSEARCH

Overview

- Motivation
- Neural Cells for Line Counting
- Hybrid Segmentation
- Evaluation Methods

Motivation

- To do OCR/HWR, usually need to break up into lines.
- Most use common image processing &/or statistics.
- Techniques work fine on small / homogeneous sets.
- Huge heterogeneous DBs are a challenge.
- Activity or Connectivity vs intentionality.
- Deep Neural Nets can be taught some about human-ness. Can they be useful for line segmentation?

Base System: “PreDNN”

We start off with a system we call “PreDNN” with the following properties:

- Multi-swath projection
- Bi-directional Dynamic programming with two-pass seam carving (first detects peaks, the troughs)
- Grayscale-based
- Use of connected components to help detect false lines or falsely merged lines.
- Statistical analyses to discard overgenerated peaks or troughs

We believe that this PreDNN rivals the state of the art.

DNNs to Count Lines

Want to create a neural network that can determine if we find well-segmented lines.

A B C D E F G

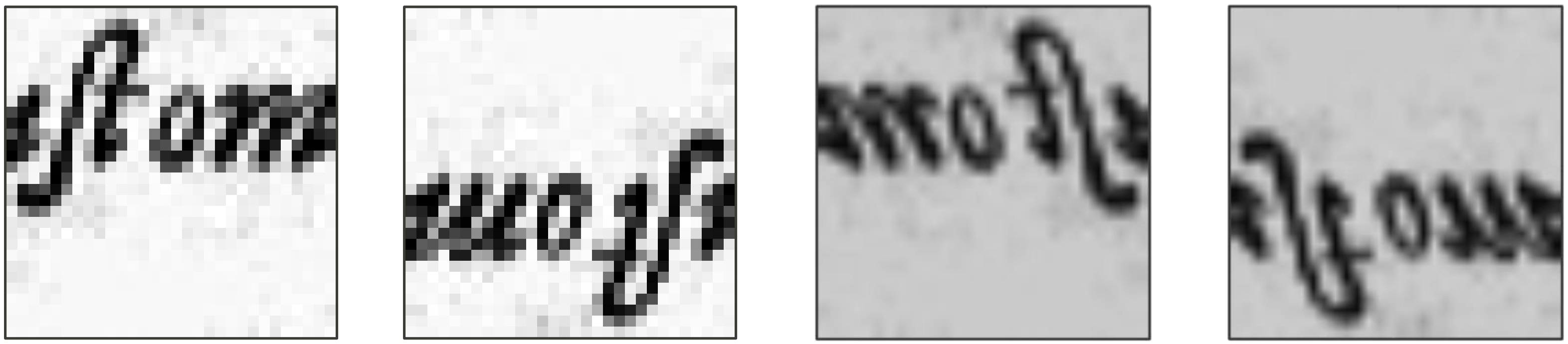
	Category
A	No-text Line
B	Single Text Line
C	Vertical Bar Only
D	Less Than One Text Line
E	Two Fragment Lines
F	More than one but less than two lines
G	Two-plus Line

Tagged Cells: Counts

	# in TRAIN	# in TEST
No-text Line	7330	625
Single Text Line	12651	848
Vertical Bar Only	703	89
Less Than One Text Line	16813	1098
Two Fragments	5027	407
More than one less than two	8573	801
Two-plus Line	14162	1409

Totals	70.5K	5.3K

Tagged Cells: Counts II



Note that the “lined-ness” of a cell is still the same even if the image has been rotated 180 degrees of flipped with respect to y-axis. We will refer to these as the “legal permutations.”

Using the legal permutations, we have 4 times more train/test:
280K training   21K testing.

The DNN

We train a **Convolutional Neural Network** for line count prediction.

We use Google's TensorFlow.
And use a variant of their MNIST-Digit-Recognition recipe.

Except, for speed, we reduce the parameters:

Kernel Size = 3
Layer #1 = 16
Layer #2 = 32
Layer #3 = 216

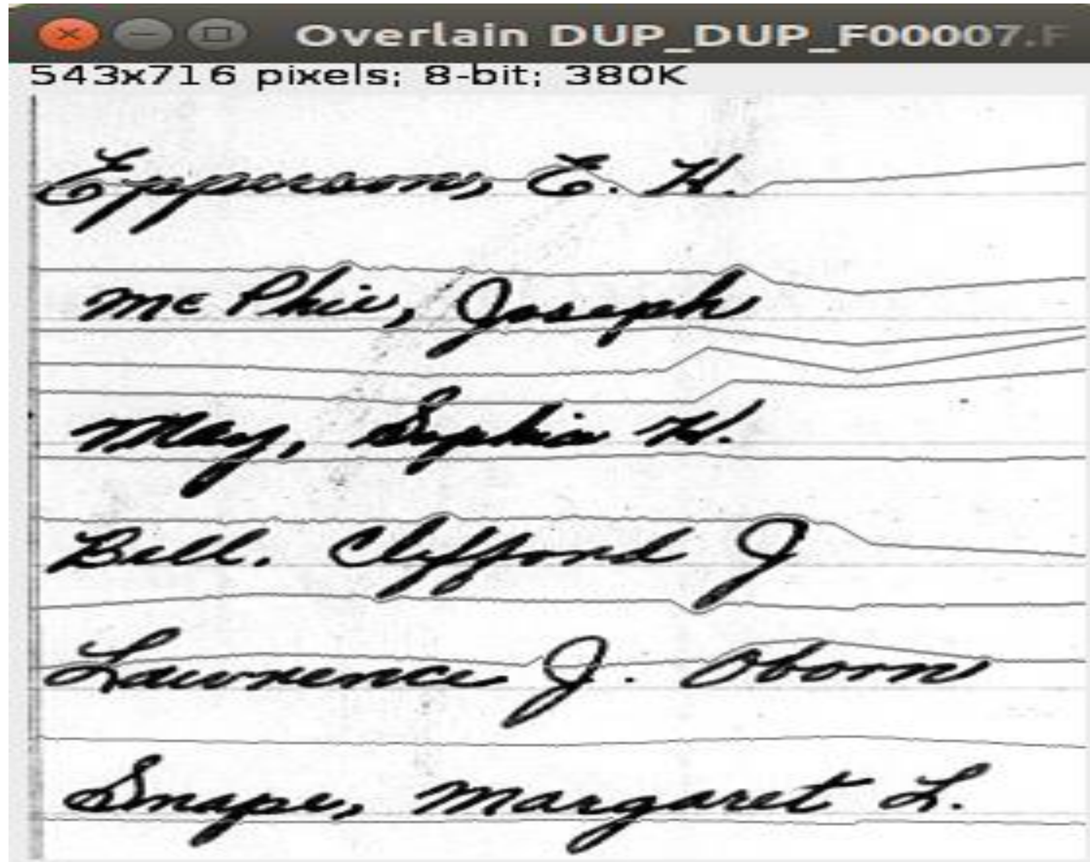
This yields a network with **91.0% accuracy**.

HOWEVER, we can get about 2% improvement by either considering the four permutations or by overlapping decision regions and voting.

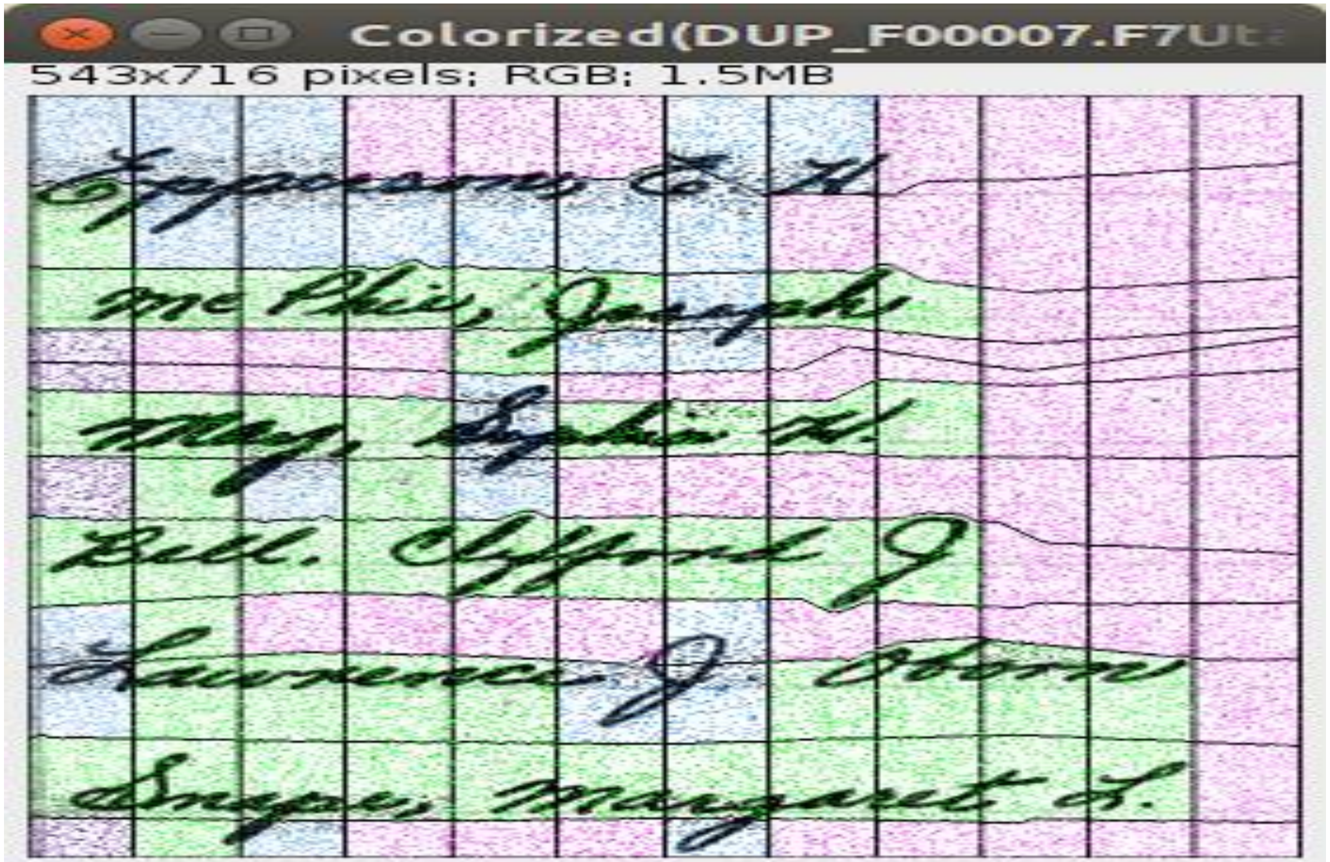
Line Counter= Line Fixer

Potential Usage:
How about if we just apply the system to PreDNN's outputs and then try to correct?

Example to the right is one where line segmenter does poorly. *Can we fix it*?



Line Counter = Line Fixer



Start off by colorizing the image.

For each swath, cut into overlapping regions.

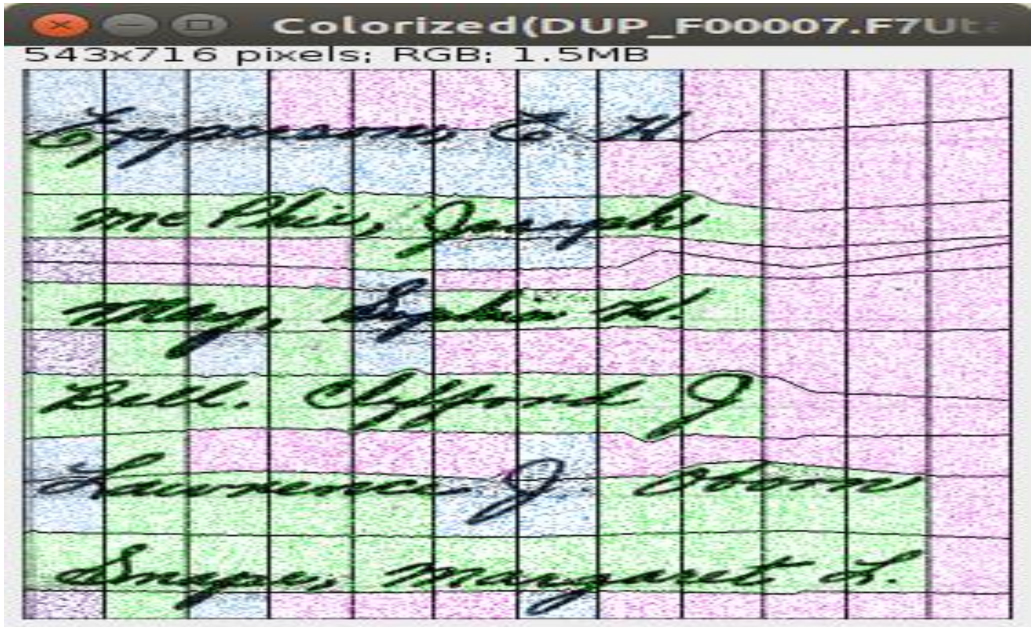
Predict color of each region.

Use voting to predict most likely color of each intersected area.

Line Counter = Line Fixer

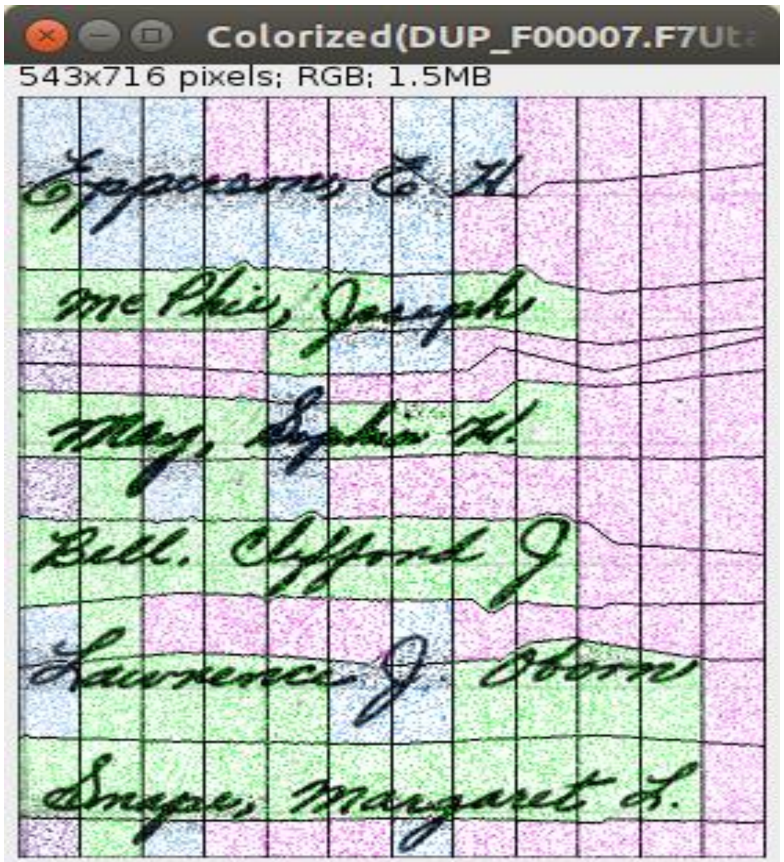
Compute "Greenness":

$$\frac{\#\,\text{GREEN Cells} + 0.1^*\,(\#\text{PINK} + \#\text{PURPLE})\ \text{Cells}}{\#\,\text{GREEN} + 0.1^*\,(\#\text{PINK} + \#\text{PURPLE}) + \#\{\text{BLUE, RED, YELLOW, ORANGE}\}}$$



Greenness = 69.9%

Line Counter = Line Fixer



We created formulas that comparing each row to the one above/below:

Symptoms of Potential False Split:

Blue/Blue: Likely split
Blue/Pink: Possible split
Green/Pink: Slight chance of split
Green/Green: Probably OK
Etc.

Symptoms of Potential False Merge:

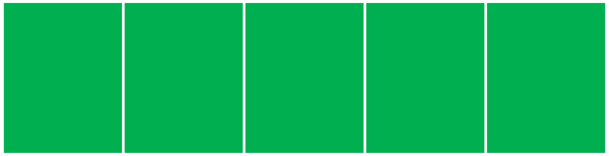
Red/Green: Likely Merge
Red/Red: Almost definite merge
Red/Orange: Probable merge

Line Counter = Line Fixer

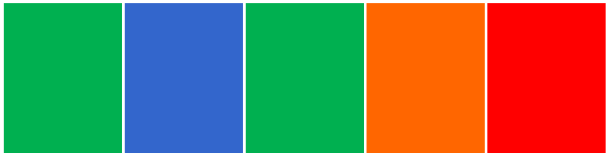
We handle potential false splits first, then false merges.
We sort from the most likely to the least, and throw out candidates with low scores.

For potential false splits (and fusions would be similar):

<= Start with row pair.
Offline, merge cells and evaluate.



<= If the results are better, replace the original with the new. ☺



<= If results are worse, though, skip that potential pairing. ☹

Line Counter = Line Fixer

Using this process, we are able to completely fix what was broken. The resultant image's line segmentation has 100% greenness!

Evaluation Methods

Option 1: Score against human-vetted lines.

Do-able, but costly to evaluate.

Option 2: Evaluate by Greenness

Very inexpensive, but “cheating” a bit

Option 3: Evaluate by Recognition

Ultimately, our end goal, so this is good eval.

Evaluation

<u>Option 2</u>: Evaluate by Greenness

	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%

Evaluation

Option 3: Evaluate by Recognition:

We built a test collection of 565 handwritten, prose-style US legal documents with 137K test words.

Also trained a handwriting recognition system using comparable but different training documents.
Then ran recognition using both PreDNN and PostDNN systems:

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

Line Segmentation cost 19% more, but recognition costs 11% less because there are fewer lines.
So for fairly comparable costs, we get 1.2% absolute gain.

Synopsis

- Can detect line count at the cellular level.
- Greenness: allows one to detect areas of potential problems.
- Neural improves segmentation of an already-good system.
- We expect it to be applicable w/ other systems.
- Final segmentation actually results in HWR improvements.

☺

ANY QUESTIONS?