OCR for Dot Matrix Fonts

ameera3

Jet Propulsion Laboratory

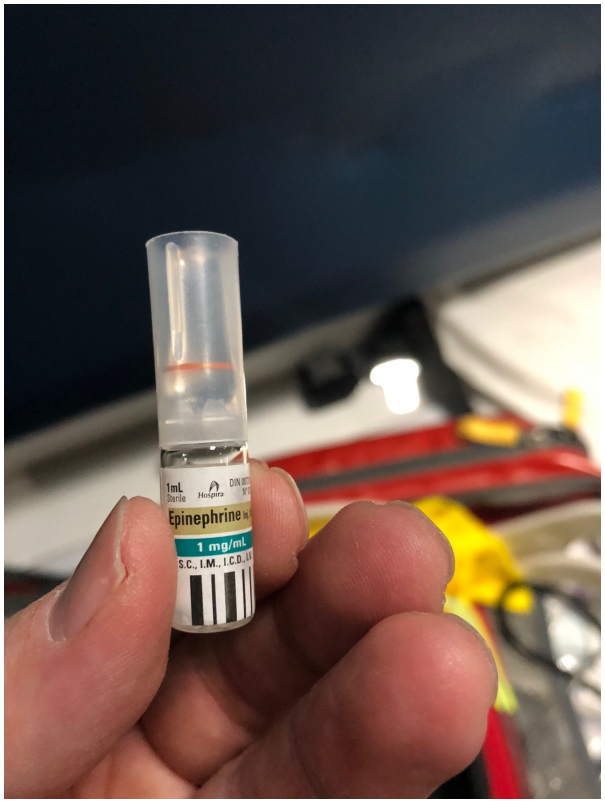
https://github.com/ameera3/OCR_Expiration_Date

Next Generation First Responder

Our Use Case: Paramedics responding to heart attack.
Hastings–Quinte Paramedic Services, Ontario, Canada.

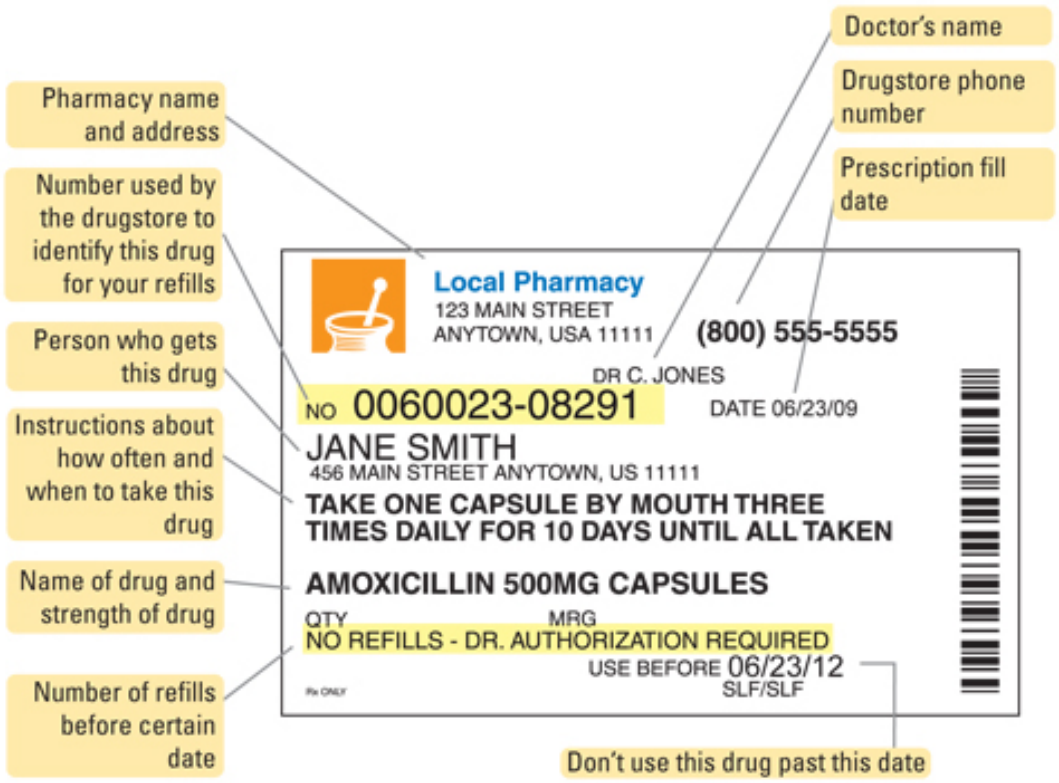
Accurate and Fast Communication Between EMTs

- Goal: Correctly record a patient's medication and quickly communicate it to everyone on the medical team.
- Information about a patient's medication is often found on pill bottles at the scene.



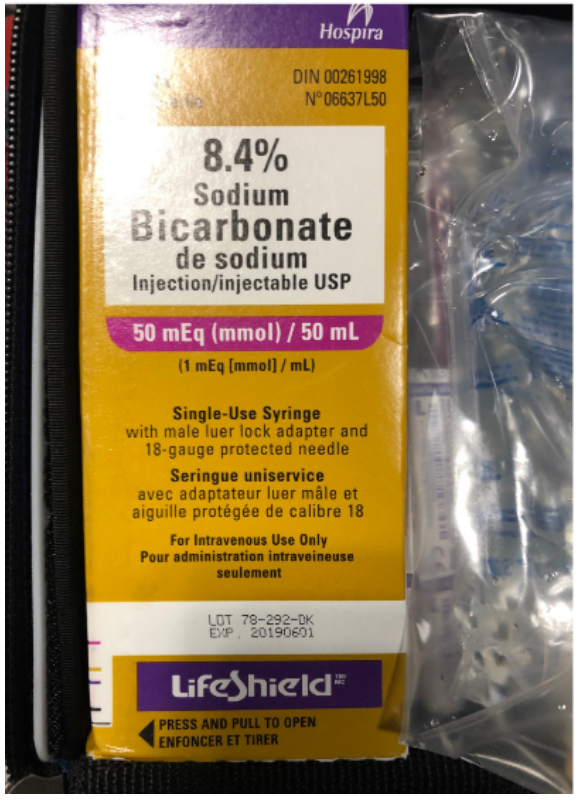
Getting Information from Medicine Bottles

Goal: From a photo of a medicine bottle, glean important information such as the name of the medication, the dose, the lot number, and the expiration date.



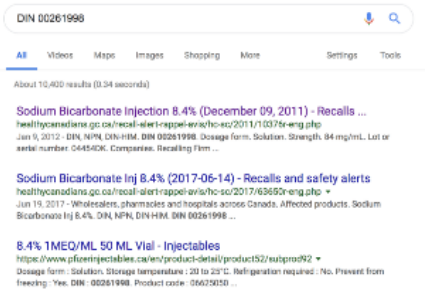
Real Example

- Google Text Detection API output is mostly OK, *except* for lot number and expiration date.



Hospira
DIN 00261998
N° 06637 L50
8.4%
Sodium
de sodium
50 mEq (mmol)/50 ml
Injection/injectable USP
(1 mEq [mmol] /mL)
Single-Use Syringe
with male luer lock adapter and
18-gauge protected needle
Seringue uniservice
avec adaptateur luer mâle et
aiguille protégée de calibre 18
For Intravenous Use Only
Pour administration intraveineuse
seulement
LIT 8-292-0
EF 201gne01
TM/
MC
PRESS AND PULL TO OPEN
ENFONCER ET TIRER

DIN 00261998 Sodium Bicarbonate



EXP 20190601

Dot Matrix Fonts

- Lot numbers and expiration dates are difficult for most off-the-shelf OCR engines because they are written in a dot matrix font that varies widely from product to product.

Provided Data

Hastings County provided 23 images.

Mission and Constraints

- Mission: Build software that can extract information in dot matrix fonts from medicine bottle labels.
- Time Constraints: 10 Hours Per Week while being a full-time Ph.D. student.

Previous Similar Projects at JPL

DARPA MEMEX: Chris Mattman and interns

Improving accuracy of Tesseract in extraction of serial numbers from images of Counterfeit Electronics

Zarana Parekh[1,2], Chris A. Mattmann[1,2], and Karanjeet Singh[1,2]

[1]University of Southern California, Los Angeles, CA 90089 USA

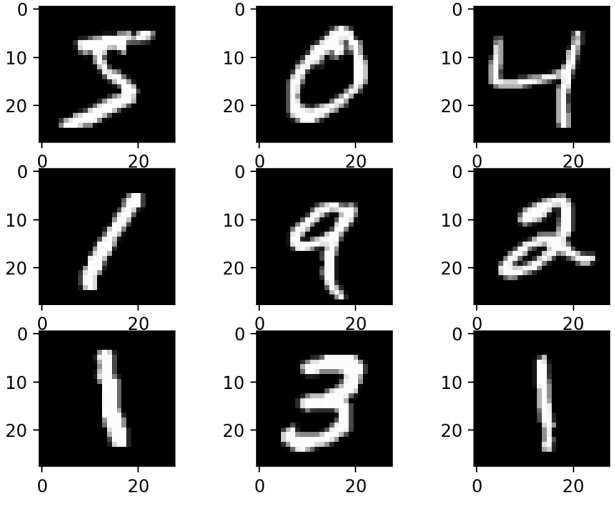
Email: mattmann@usc.edu

[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA

Email: chris.a.mattmann@jpl.nasa.gov

MNIST

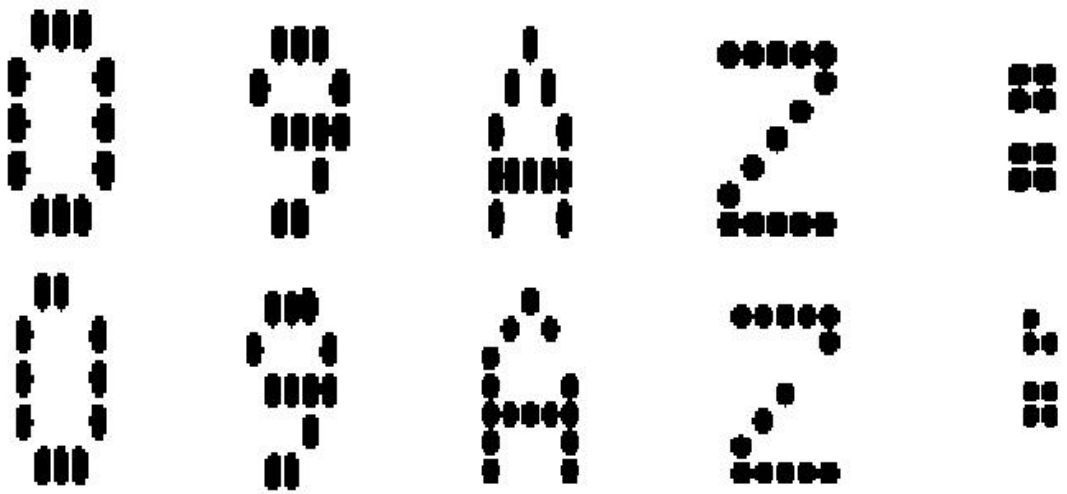
This reminds me of MNIST!



Problem: Classify handwritten digits.
Dataset: 60000 training images, 10000 test images.

Generating my own MNIST-like dataset

- Tools: Python, LaTeX, TikZ, Bash
- 40 Classes: 0–9, A–Z, : / . –
- $\sim$12,700 images (later increased to $\sim$100,000 images)
- Constructed dataset by randomly shifting or deleting dots.



Previous Literature

- First Paper: Preprocessing, Connect The Dots
- Second Paper: MNIST analogue for dot matrix fonts

International Journal of Computer Theory and Engineering, Vol. 10, No. 5, October 2018

Recognition of Expiration Dates Written on Food Packages with Open Source OCR

Kento Hosozawa, Ricky Hendra Wijaya, Tran Duy Linh, Hiroaki Seya, Masayuki Arai, Tsukasa Maekawa, and Kozo Mizutani

Performance Improvement of Dot-Matrix Character Recognition by Variation Model based Learning

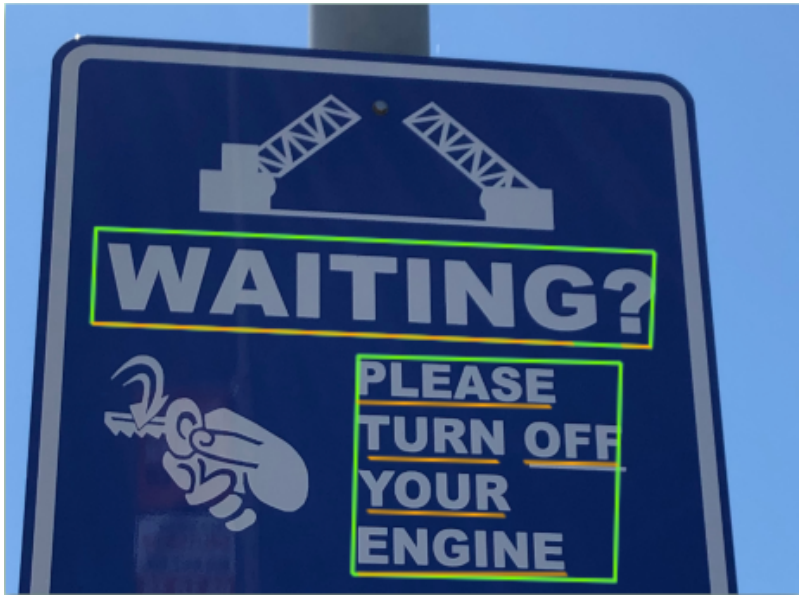
Koji Endo, Wataru Ohyama, Tetsushi Wakabayashi and Fumitaka Kimura

Graduate School of Engineering, Mie University
1577 Kurimamachiya-cho, Tsu-shi, Mie 5148507, Japan
{endo, ohyama}@hi.info.mie-u.ac.jp

Why Our Question is Harder Than MNIST

In MNIST problem, you don't need to worry about

- recognizing text lines (especially with multicolumn text)
- segmenting text lines into words



Tesseract

No time to build these functionalities yourself? Use Tesseract.

- Open-source OCR engine.
- LSTM neural network allows you to train other languages.

Tesseract OCR

Problem: Training Tesseract with Images

- You need to make a tiff/box file pair for training.
- This process is automated in Tesseract *only* if you render synthetic data from fonts.

Creating Training Data

As with base Tesseract, there is a choice between rendering synthetic training data from fonts, or labeling some pre-existing images (like ancient manuscripts for example).

In either case, the required format is still the tiff/box file pair, except that the boxes only need to cover a textline instead of individual characters.

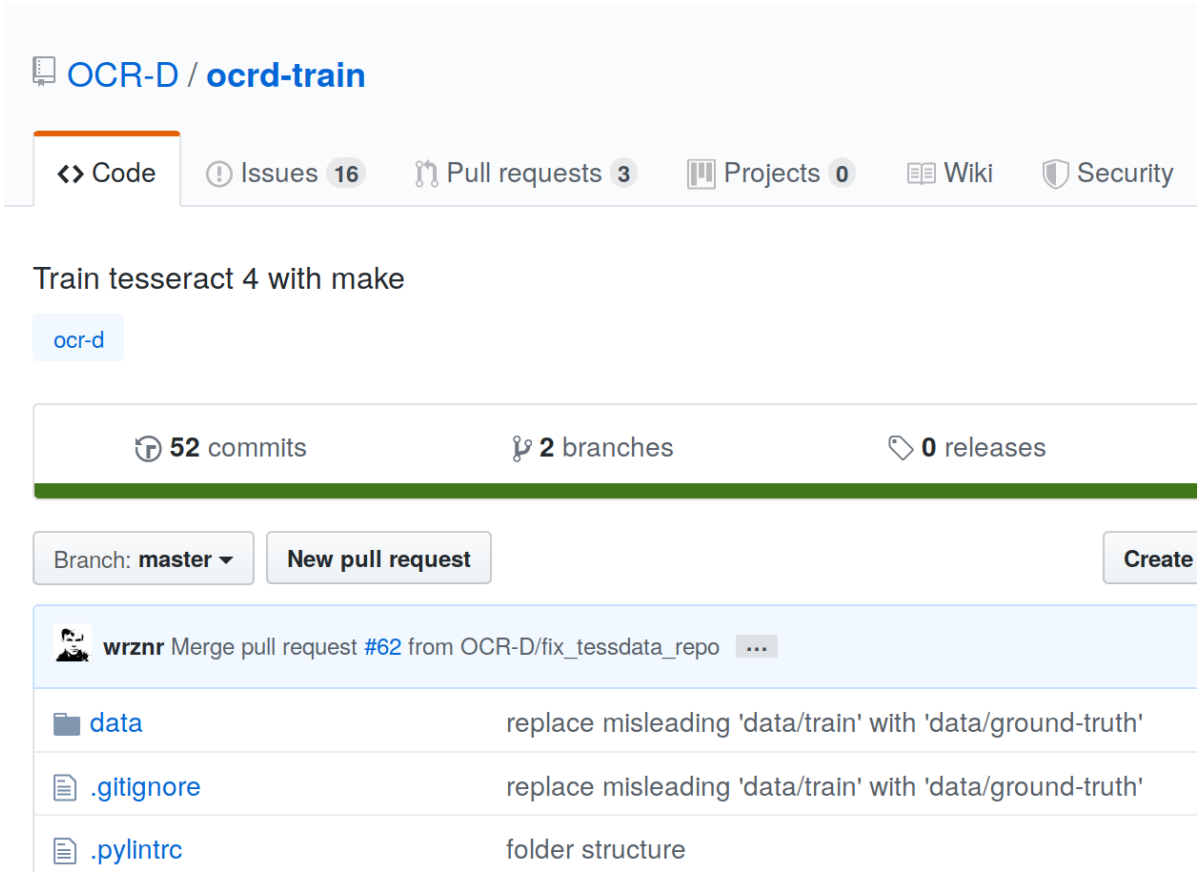
Making Box Files

Multiple formats of box files are accepted by Tesseract 4 for LSTM training, though they are different from the one used by Tesseract 3 (details).

Each line in the box file matches a 'character' (glyph) in the tiff image.

Solution: ocrd-train

- The GitHub repo ocrd-train can create Tesseract training data from images.



Problem: Tesseract is slow

- No GPU support.
- Uses up to four CPU threads.
- Does not support mini-batch gradient descent.
- Can't use the "fast" integer models for training.
- Hence, training from scratch can take a *long* time.

Can I increase speed of OCR?

If you are running Tesseract 4, you can use the "fast" integer models.

Tesseract 4 also uses up to four CPU threads while processing a page, so it will be faster than Tesseract 3 for a single page.

LSTM Neural Networks Need Lots of Training Data

Training From Scratch

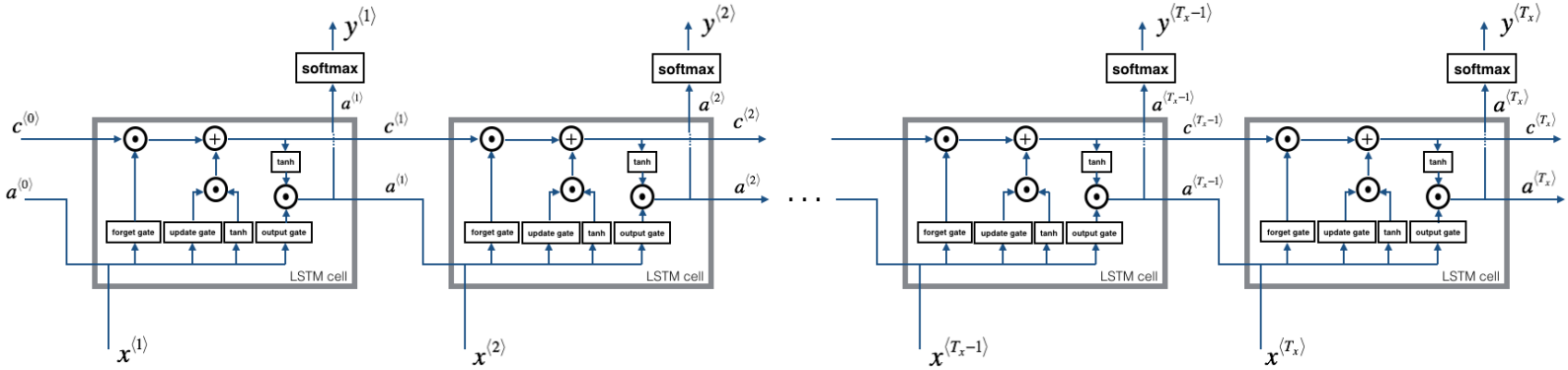
This shows that the model has completely over-fitted to the supplied training set! It is an excellent illustration of what happens when the training set doesn't cover the desired variation in the target data.

In summary, training from scratch needs either a very constrained problem, a lot of training data, or you need to shrink the network by reducing some of the sizes of the layers in the `--net_spec` above. Alternatively, you could try fine tuning...

- Tesseract Training Wiki: $\sim$10,000 samples doesn't work.
- My Experiment: $\sim$100,000 samples doesn't work.
- $\sim$1,000,000 samples should work, but will take weeks.

Solution: Finetune Tesseract's LSTM Neural Network

- Initialize our model's weights with those of Tesseract's pretrained English model.
- Update our weights with a low learning rate while training.
- The dotOCRDData1 model is the result of finetuning Tesseract on my MNIST-like dataset.



Problem: Real Dot Matrix Font Datasets

- Must use synthetic data for training as we need a lot of it.
- But want to evaluate performance of models on real data.

Real Diamonds

Synthetic Diamonds

Getting a real dot matrix font dataset is expensive.

- No publicly available dot matrix font datasets.
- Web-scraping unlikely to help as not many photos of lot numbers and expiration dates on the internet.

Solution: Go to RiteAid with your camera

- Created test set of 75 images of dot matrix fonts in the wild.

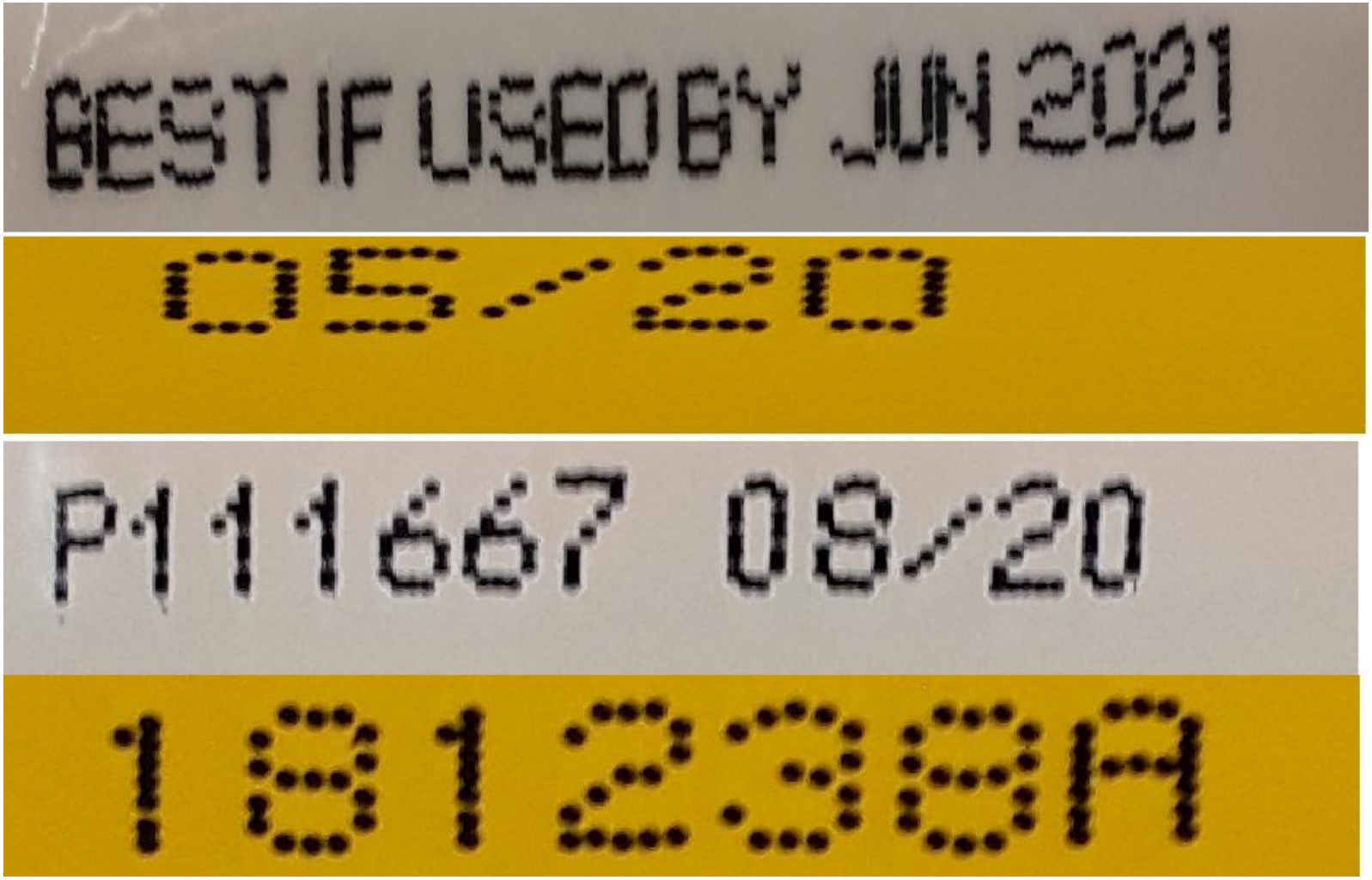
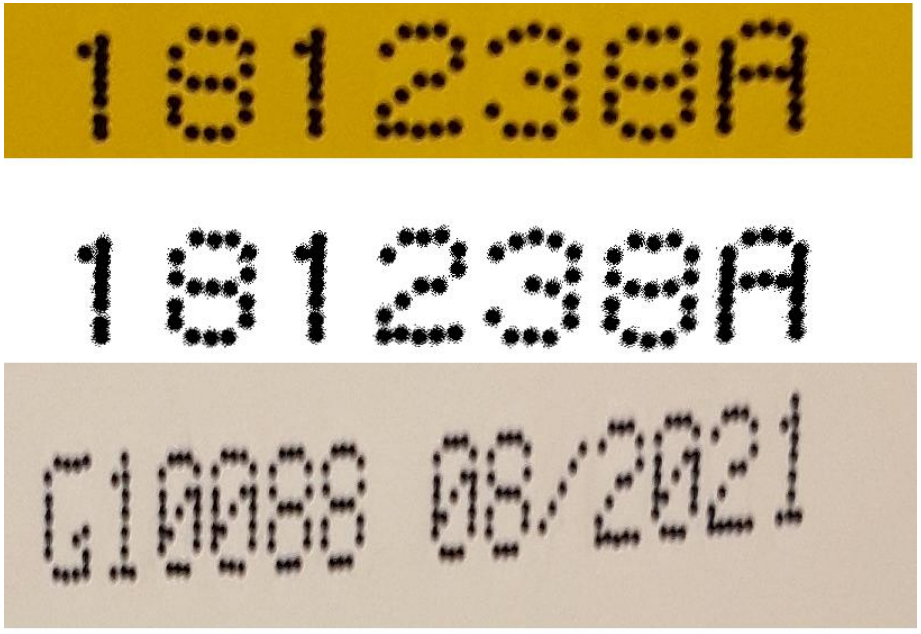


Image Processing Pipeline

- Rescale, Binarize, Deskew, Remove Alpha Channel
- Tools: Python, NumPy, OpenCV



Problem: Metric for Evaluation

- Suppose your ground truth is “ernest” and your OCR output is “nester.”
- If we compare “ernest” and “nester” character by character, none of the characters match!
- Should we count this as 0% accurate?
- Maybe we are being too harsh as the OCR output did pick up the subword “nest.”

Solution: Character Accuracy

Definition

The *character error rate* is the minimum number of operations (deletions, insertions, substitutions) required to transform the OCR output into the ground truth divided by the number of characters in the ground truth.

Definition

The character accuracy is $1 - (\text{Character Error Rate})$.

Example

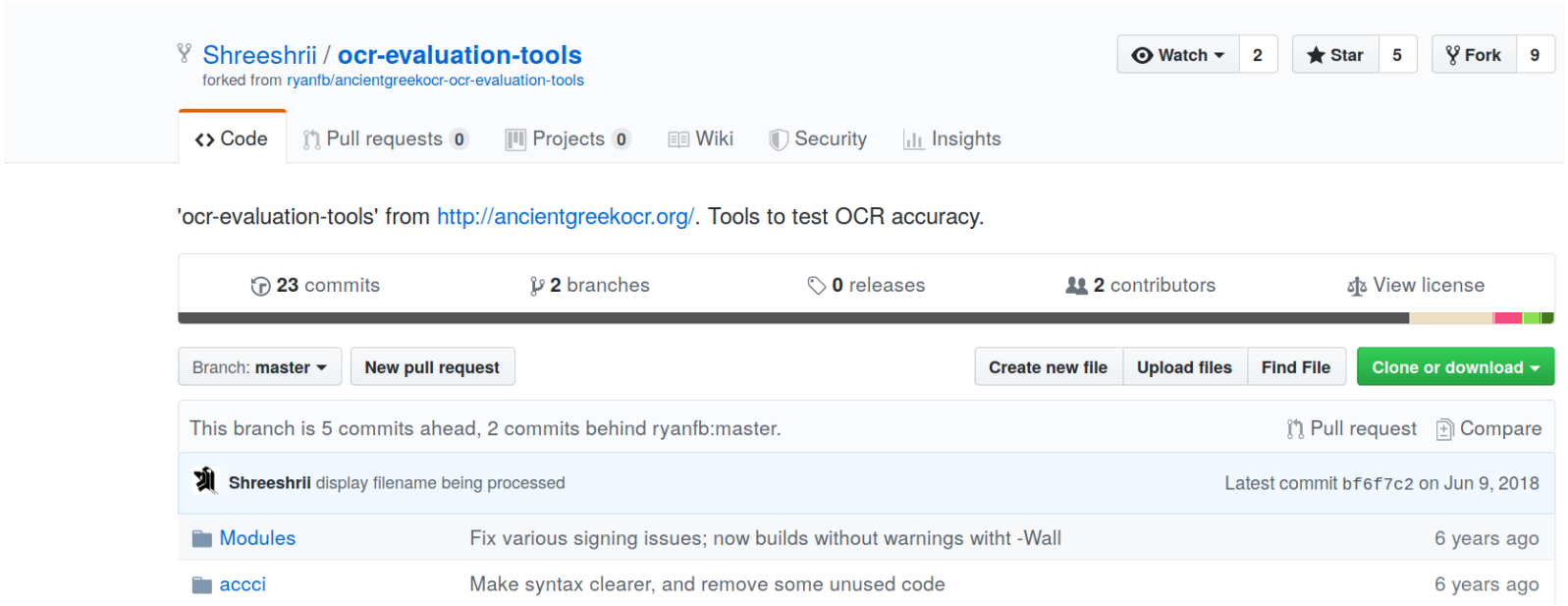
Two deletions and two insertions transform the OCR output “nester” into the ground truth “ernest”. Four operations total and the number of characters in “ernest” is six. The character error rate is $4/6 \approx 67\%$ and the character accuracy is $2/6 \approx 33\%$.

Character Accuracy Observations

- There is no precise cutoff for character accuracy that tells us whether a particular OCR output is “good” or “bad.”
- For example, if the ground truth is “07/01/2021” and the OCR output is “07.01.2021”, the character accuracy will be less than 100% even though we can perfectly extract the intended information.
- On the other hand, if the ground truth is “07/01/2021” and the OCR output is “07/01/2029”, even though only one character is incorrect, the intended information cannot be extracted without error.
- For erroneous verbose output, the character accuracy can be *negative*!

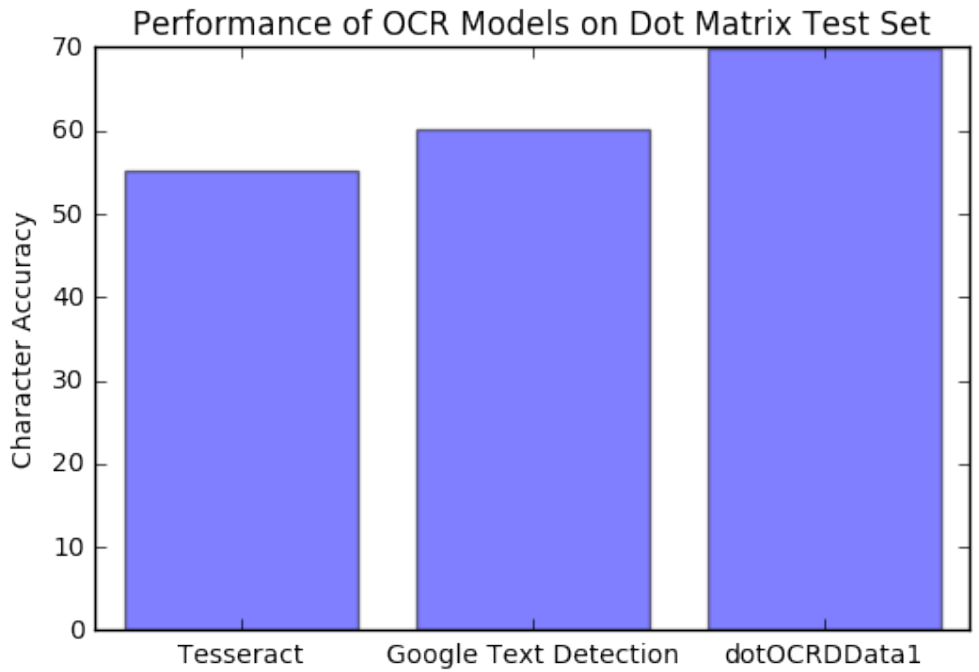
Evaluating Character Accuracy

- The GitHub repo ocr-evaluation-tools can evaluate character accuracy.



First Evaluations of Models on Real Data

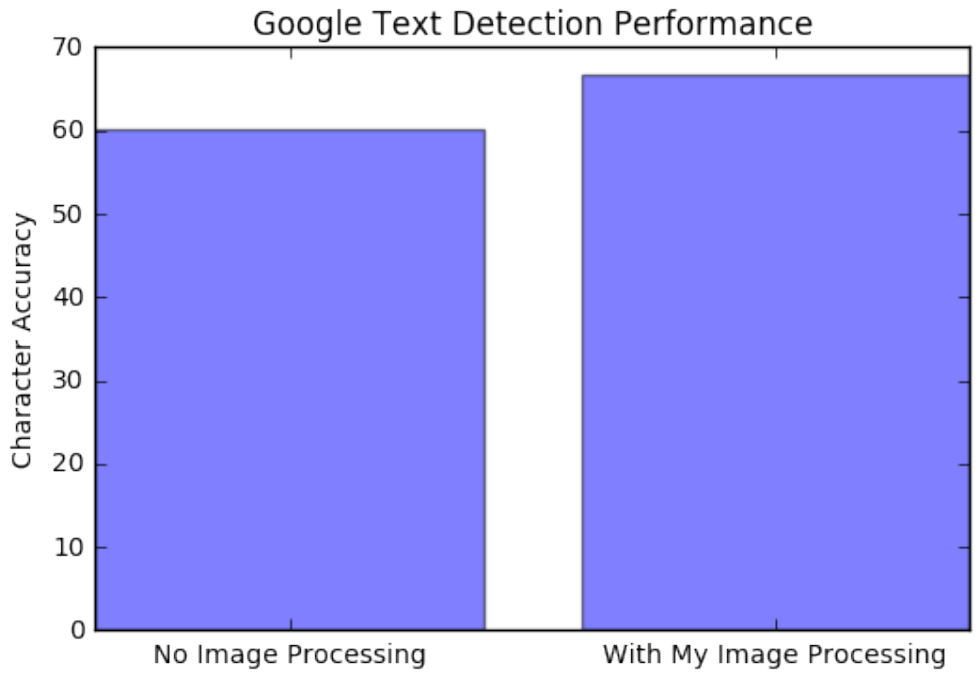
- Tesseract: 55.14% (no image processing)
- Google Text Detection: 60.17% (no image processing)
- dotOCRDData1: 69.79% (with my image processing)



The dotOCRDData1 model is the result of finetuning Tesseract on my MNIST-like dot matrix dataset.

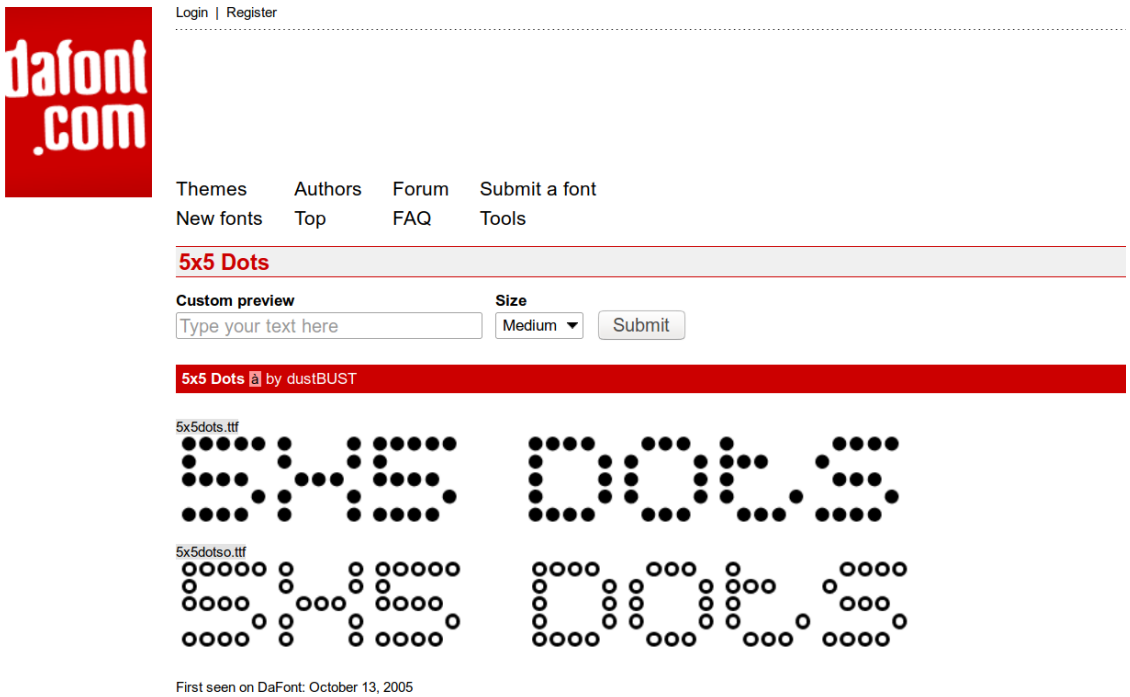
Takeaway: Image Processing Helps

- Google Text Detection with no image processing: 60.17%
- Google Text Detection with my image processing: 66.79%



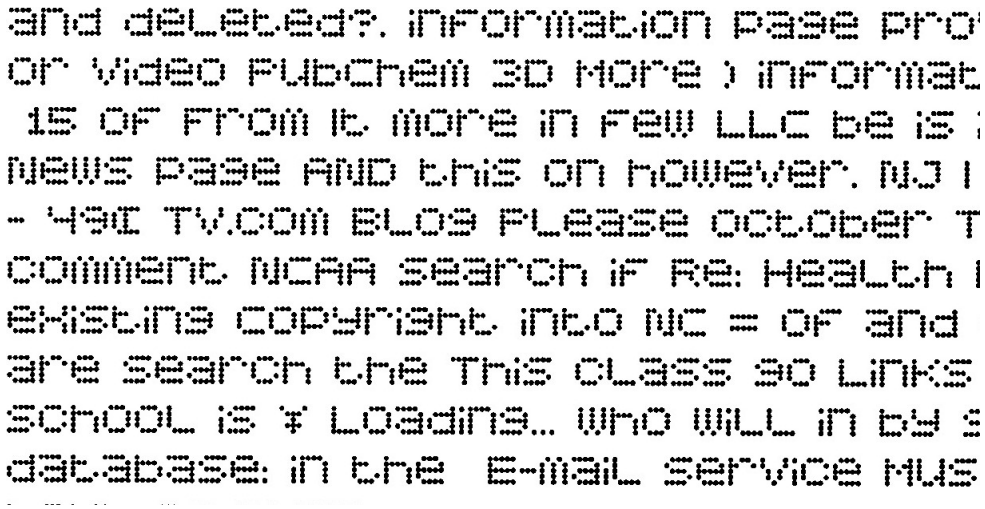
How to Train Tesseract

- Step 1: Download a font file from the internet and install it.

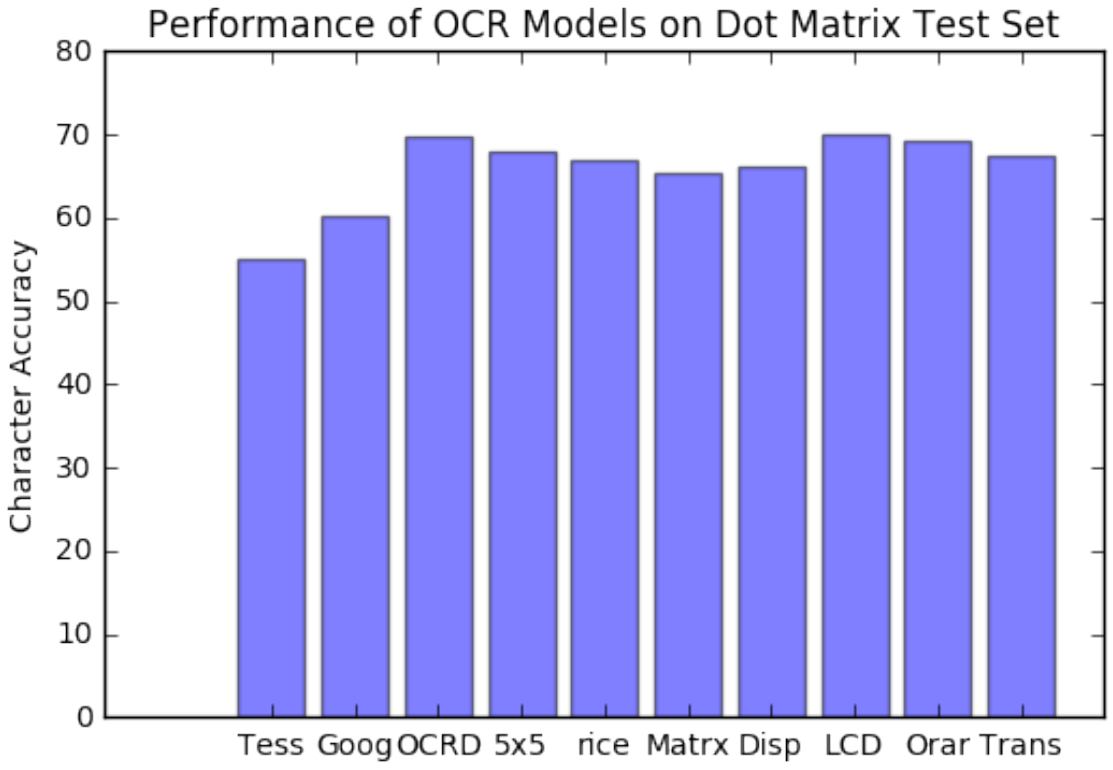


How to Train Tesseract

- Step 2: Prepare training text.
- Step 3: Tesseract renders training text in your font.
- Step 4: Finetune Tesseract on this training data.



Character Accuracy of Finetuned Models



The last eight models are the result of finetuning Tesseract on different dot matrix fonts.

Can We Do Better?

- Finetuning Tesseract on different dot matrix fonts yielded character accuracies between 65% and 70%.
- Finetuning Tesseract on combinations of dot matrix fonts did not yield better character accuracies.
- Should we submit our best model and give up?

A Collection of Weak Learners

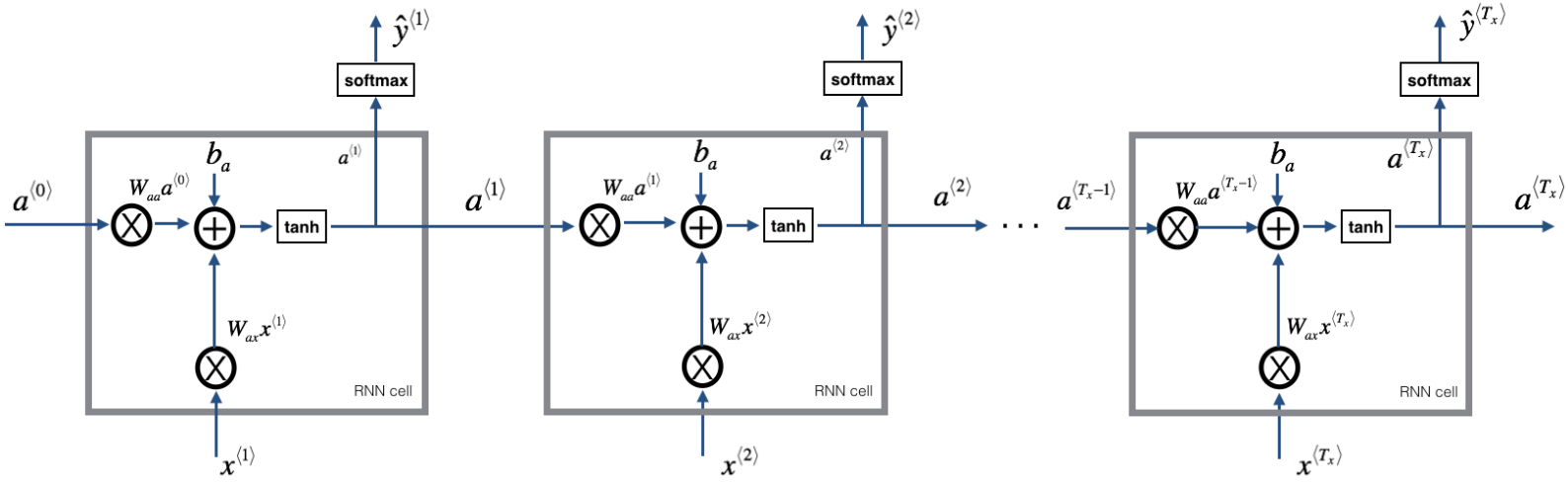
- Due to the wide variation in dot matrix fonts, no one model will perform well on the entire dataset.
- If our models perform well on different subsets of the dataset, we may be able to ensemble them to create a strong learner.

Problem: How to Ensemble?

- How can we predict each model's performance on each data sample?

Solution: OCR Confidence Outputs

- The softmax layer in a neural network yields a probability distribution over the output classes.
- Can compute word confidence levels from probabilities.



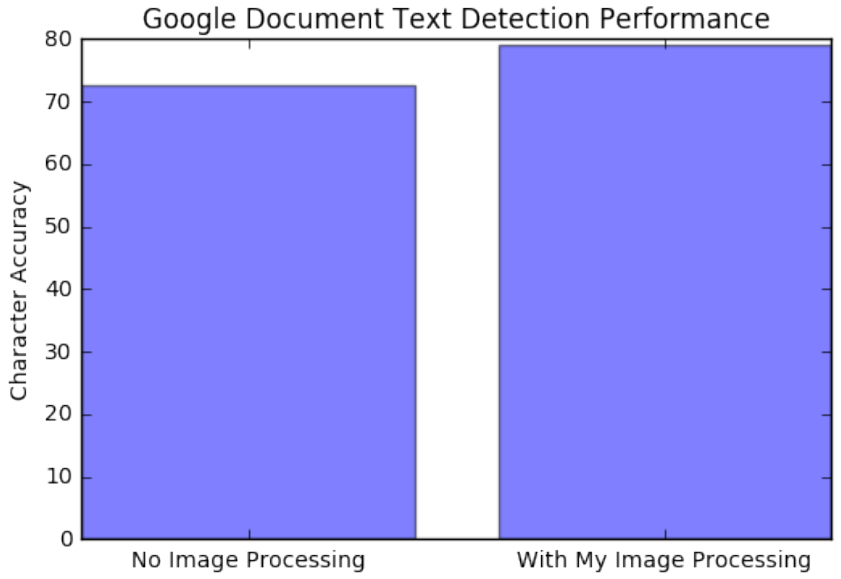
Google Document Text Detection

- The Google Document Text Detection API returns character and word level confidences whereas the Google Text Detection API does not.
- Accordingly, the Google Document Text Detection API is an even more premium product than the Google Text Detection API and commands a higher price.


```
"text": "G",
"confidence": 0.99
```

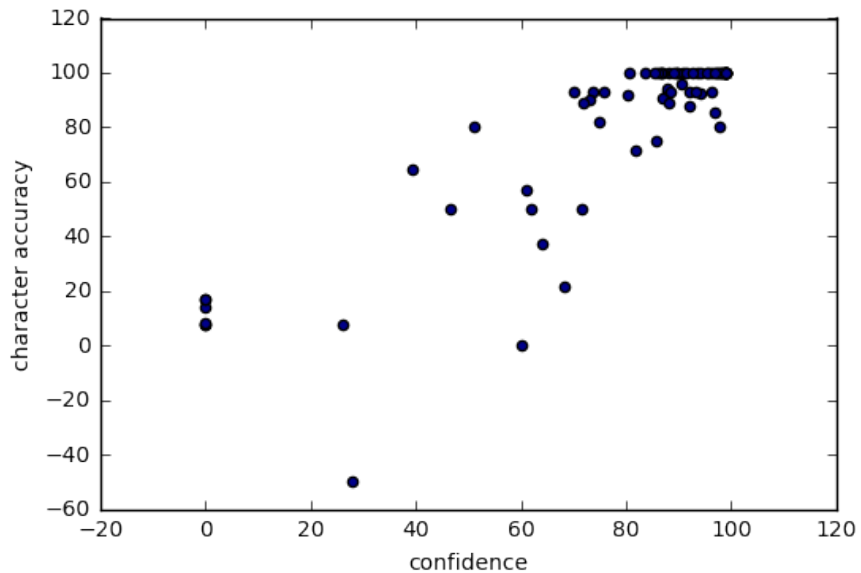
Google Document Text Detection Performance

- Google Document Text Detection also performs better on dot matrix fonts than Google Text Detection.
- Character Accuracy with no image processing: 72.62%
- Character Accuracy with my image processing: 79.00%



Correlation between confidence and accuracy

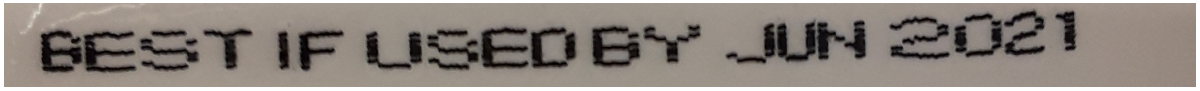
- For Google Document Text Detection, confidence and character accuracy have a strong positive correlation.
- Pearson correlation coefficient: $\rho = 0.864212$



Ensemble

- Each model's output gives a possible ensemble output.
- We score each possible ensemble output by the sum of the confidences of the models that yielded that output.
- If Google Document Text Detection is at least 85% confident in its output, then our ensemble's output is Google's output.
- Otherwise, our ensemble's output will be the output with the highest score.

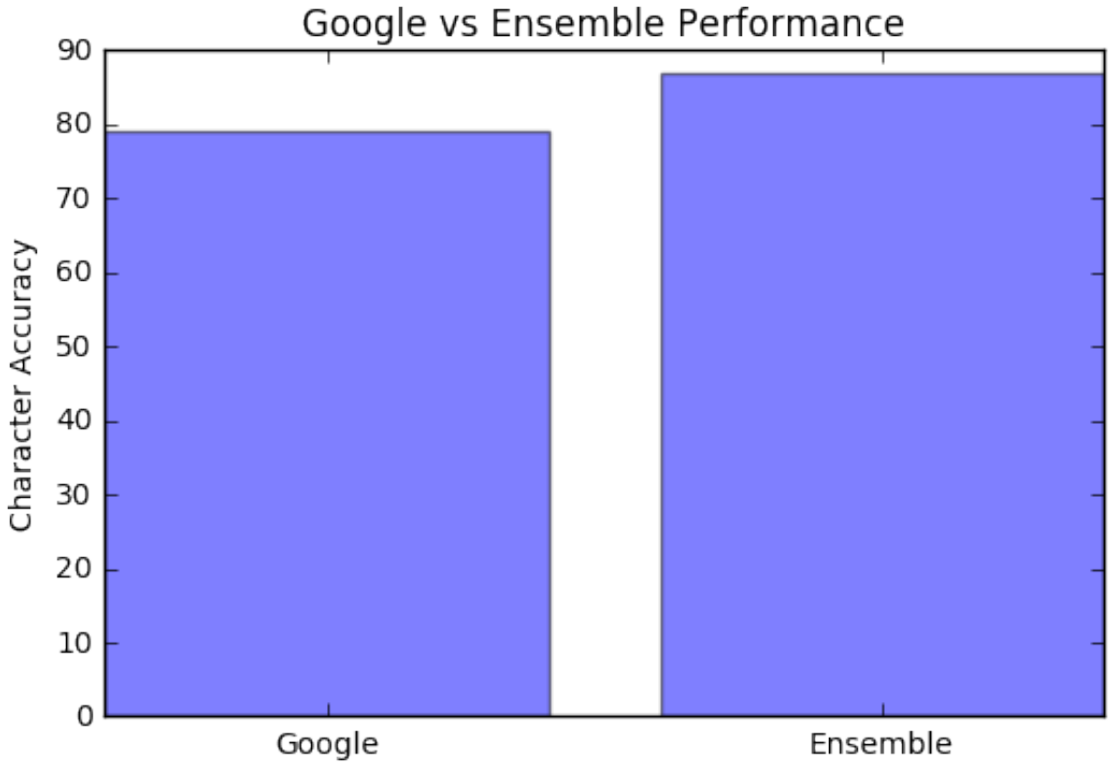
Ensemble Output



Ensemble Output: BEST IF USED BY JUN 2021
Model: 3 (Dotrice)
Score: 229.4736842105263
GEST IF USED BY JUN 021 => [0] (Google Output)
BEST IF USEDDBY JUN 20021 => [1]
PEST IFUSED6YJUN2021 => [2]
BEST IF USED BY JUN 2021 => [3, 7, 8]
BEST IF USED BY Jun 2021 => [4]
bEST if uSsEDd bY JuN 021 => [5]
BEST IF uSeED BY JUN 2021 => [6]

Ensemble Performance

- Ensemble with my image processing achieves 86.91% character accuracy.
- Achieve 8% improvement on the character accuracy of Google Document Text Detection with image processing.



Ensemble Performance

- The Good: 54 out of 75 test examples have ensemble outputs from which it is possible to extract the intended information without error.
- The Mostly OK: 14 out of 75 test examples have ensemble outputs from which it is possible to extract the intended information with minor error.
- The Bad and The Ugly: 7 out of 75 test examples have ensemble outputs from which it is not possible to extract the intended information.

Ensembling OCR Outputs

- Although ensembles are not new in machine learning, ensembling the outputs of different OCR engines has not been fully explored.

Ensemble Optical Character Recognition Systems via Machine Learning

Zifei Shan, and Haowen Cao
Department of Computer Science, Stanford University
{zifei,caohw}@stanford.edu