# High Performance Text Recognition using a Hybrid Convolutional-LSTM Implementation

Thomas Breuel
NVIDIA Research

# Question

*“Hey, do you know anything about OCR? We need a fast, GPU-based OCR engine.”*

# Goals

- Build a fast OCR engine using GPUs
- Make it extensible by using standard libraries (PyTorch)
- Explore more architectures (convolutional+LSTM hybrids)
- 100% Deep Learning, data-driven (this is new):
    - Eliminate explicit normalization
    - Eliminate hand-coded layout analysis (tomorrow)

# LSTM OCR

Graves, Alex, and Jürgen Schmidhuber. "**Offline handwriting recognition with multidimensional recurrent neural networks.**" Advances in neural information processing systems. 2009.

Breuel, Thomas M., et al. "**High-performance OCR for printed English and Fraktur using LSTM networks.**" Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, 2013.

Yousefi, Mohammad Reza, et al. "**A comparison of 1D and 2D LSTM architectures for the recognition of handwritten Arabic.**" DRR. 2015.

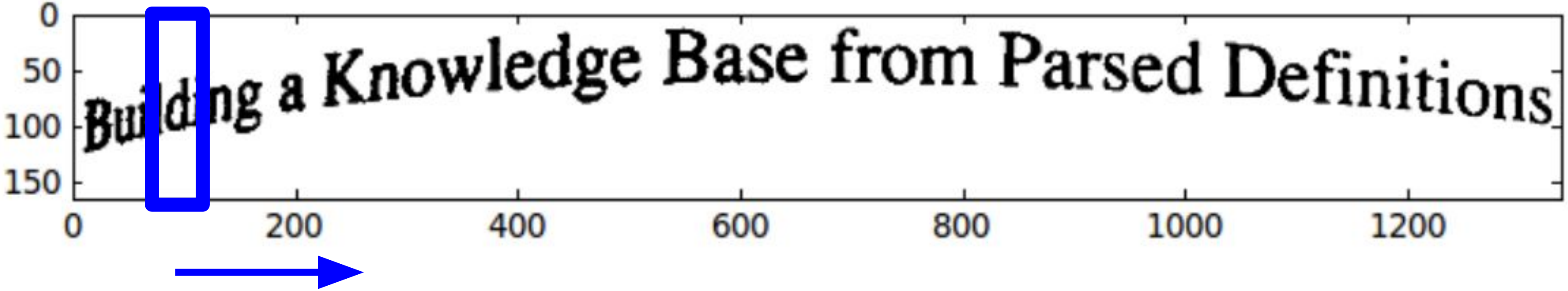
# Scene Text / Handwriting Recognition

Scene Text / HW is broadly similar, but…

- much larger range of distortions and text orientations
- generally less strict requirements for error rates
- even approximate binarization may be unavailable

HW uses MDLSTM to achieve invariance: does it yield competitive results for OCR?

# Sequential Recognition and Normalization



OCR Engine Types:

- segmentation + character based
- left-to-right scanning (HMM, LSTM), segmentation-free

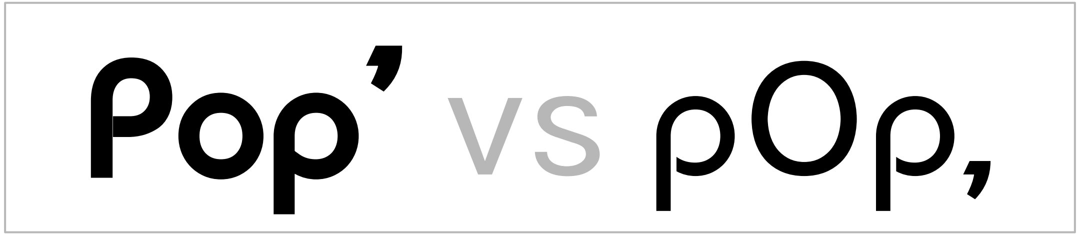
For left-to-right scanning, columns of the image become feature vectors and need to be as invariant as possible.

# Context Problem in OCR

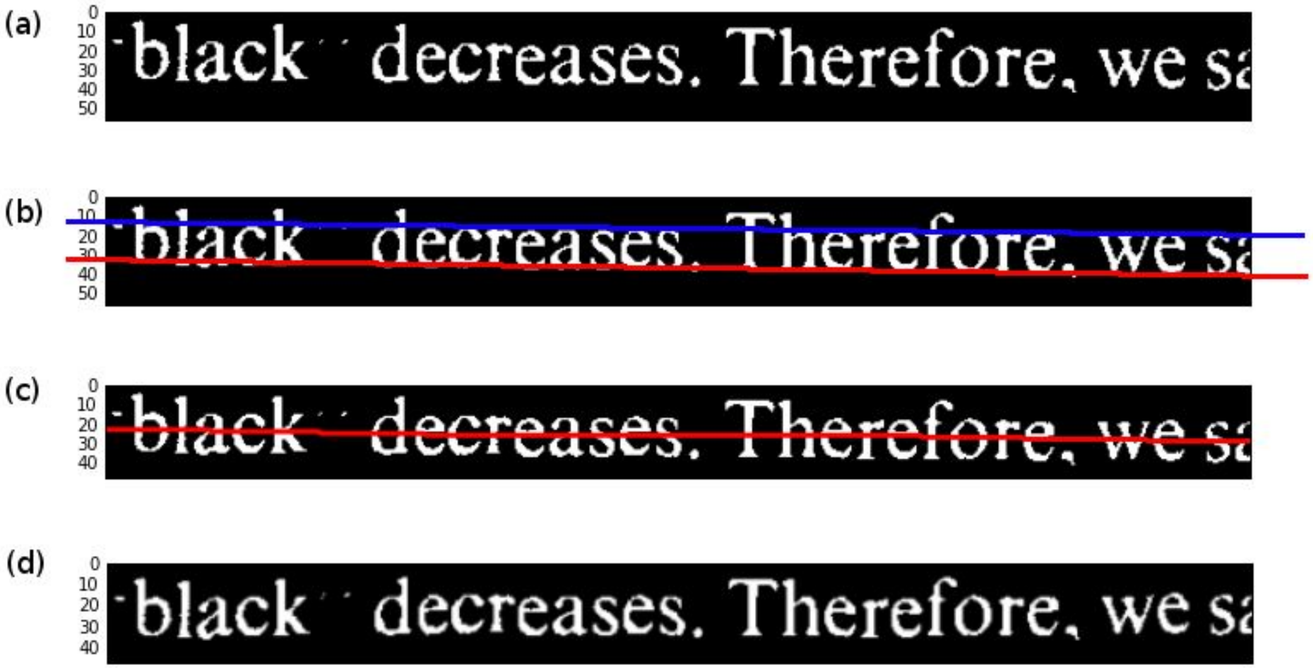
Character identity is context dependent (size + position relative to baseline)

Baseline can only determined using long-range dependencies.

Translation invariant feature extraction fails to capture long-range context.



# Explicit Normalization


(a)
0
10
20
30
40
50
black decreases. Therefore, we sa
(b)
0
10
20
30
40
50
black decreases. Therefore, we sa
(c)
0
10
20
30
40
black decreases. Therefore, we sa
(d)
0
10
20
30
40
black decreases. Therefore, we sa

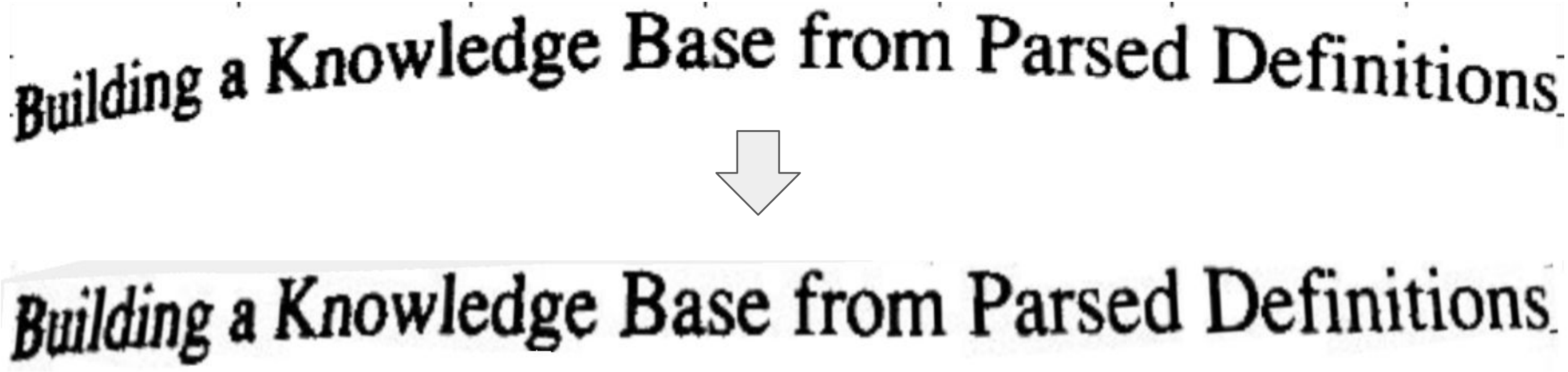
original

baseline + x-height estimation

smooth centerline estimation

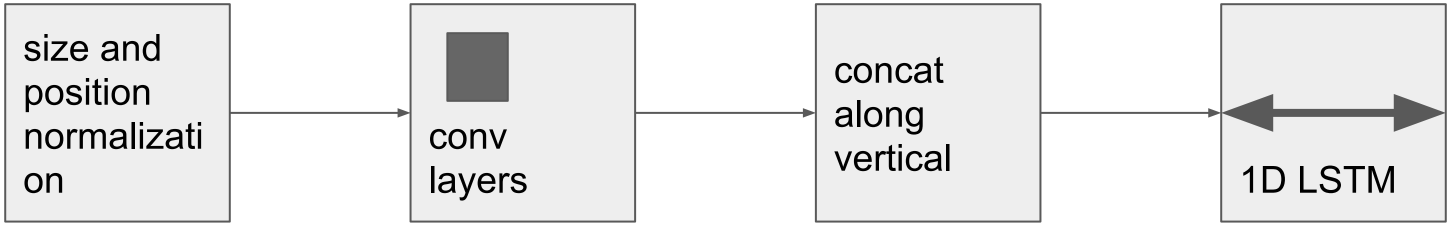
approximate normalization

# LSTM-based OCR With Explicit Normalization



Building a Knowledge Base from Parsed Definitions

Building a Knowledge Base from Parsed Definitions



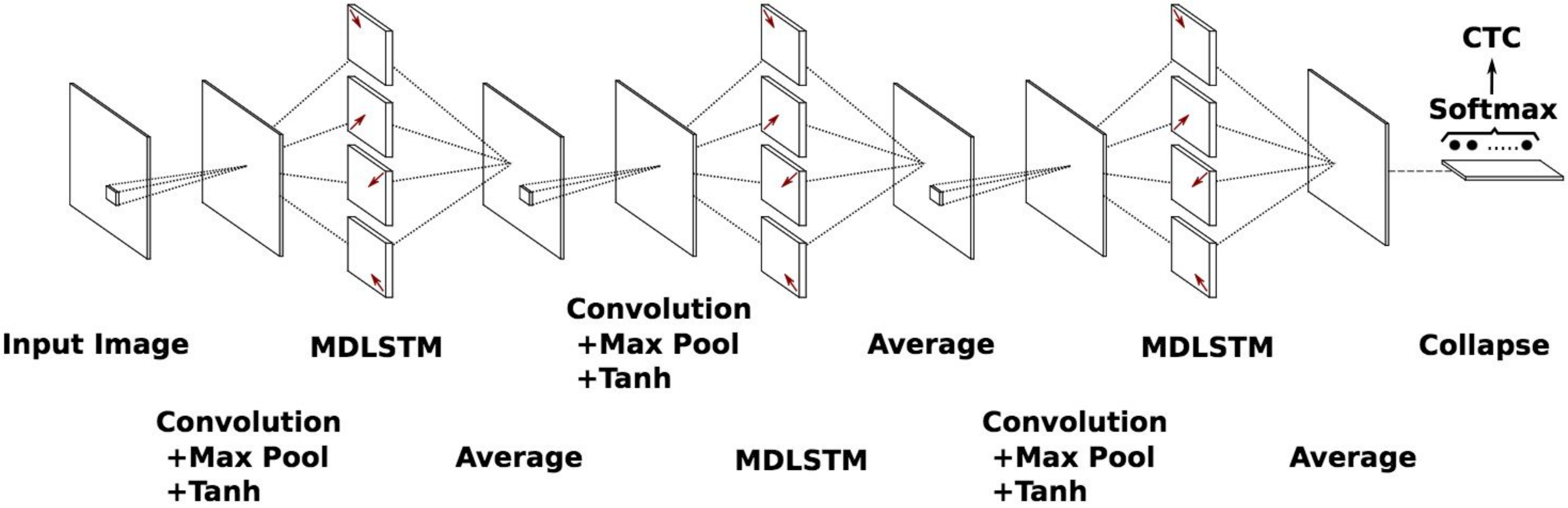
size and position normalization

conv layers

concat along vertical

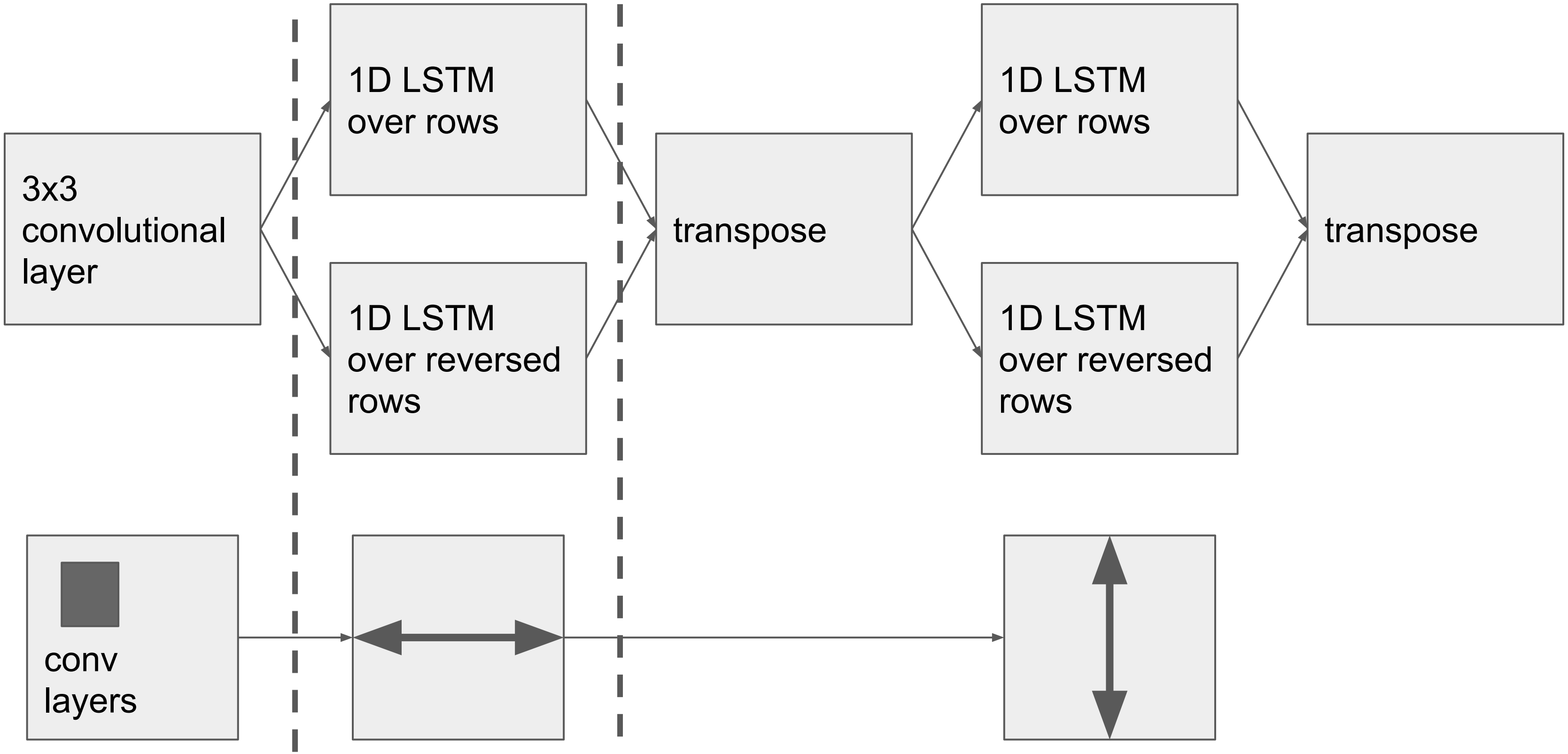
1D LSTM

# Invariance via Multidimensional LSTM (MDLSTM)


CTC
Softmax
Input Image
Convolution +Max Pool +Tanh
MDLSTM
Average
Convolution +Max Pool +Tanh
MDLSTM
Average
Convolution +Max Pool +Tanh
MDLSTM
Average
Collapse

Voigtlaender, Paul, Patrick Doetsch, and Hermann Ney. "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks." Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on. IEEE, 2016.

# Separable MDLSTM



3x3 convolutional layer

1D LSTM over rows

1D LSTM over reversed rows

transpose

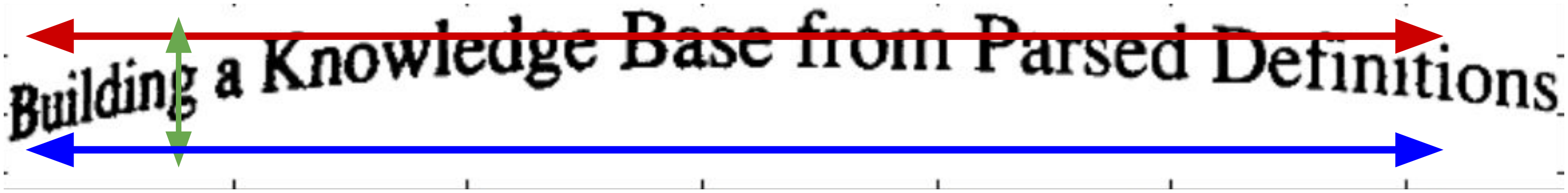
1D LSTM over rows

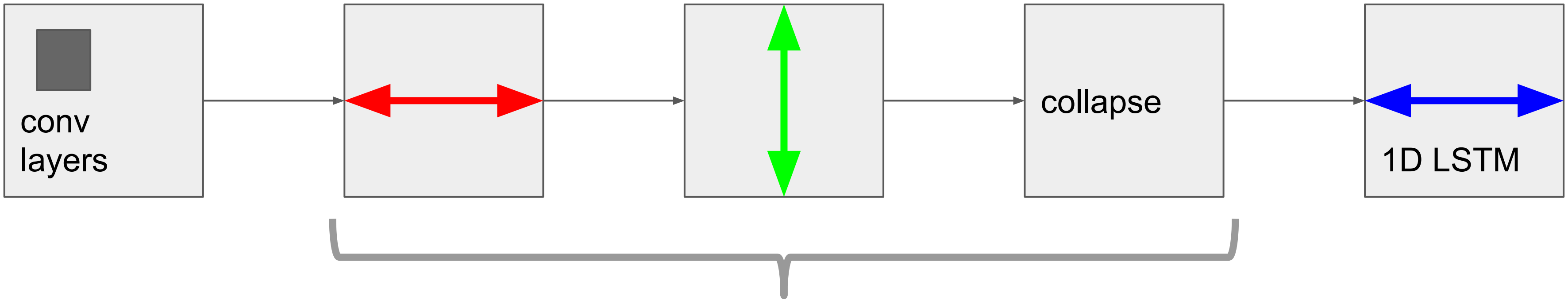
1D LSTM over reversed rows

transpose

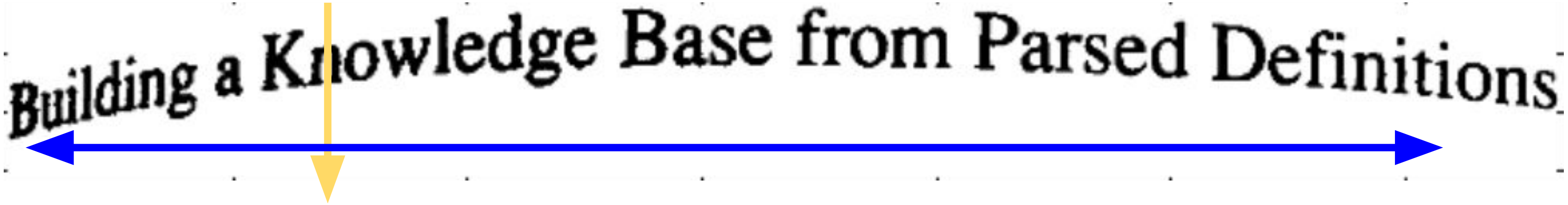
conv layers

# 2D LSTM + 1D LSTM

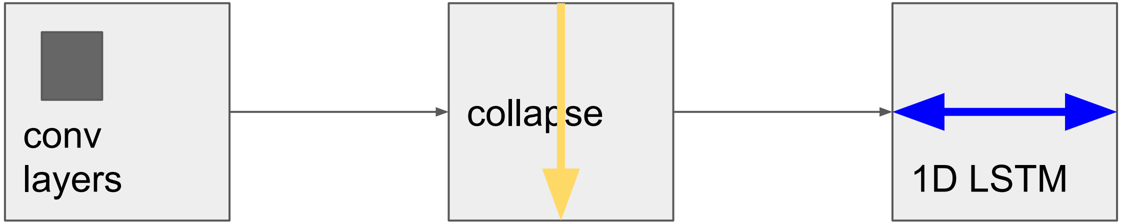




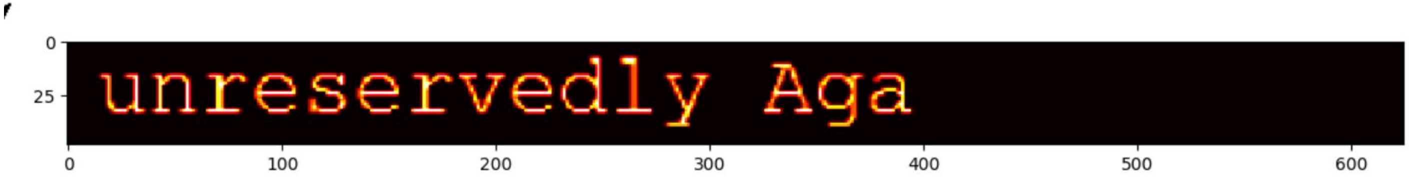
# Brute Force Vertical Invariance

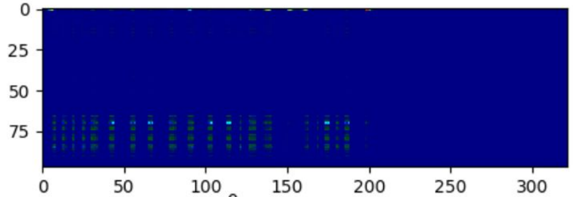


Building a Knowledge Base from Parsed Definitions

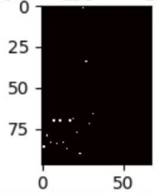


# CTC (forward-backward alignment)

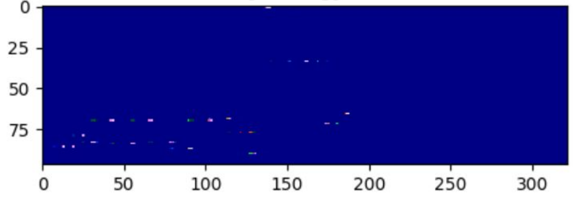




posterior probability at each pixel location



one-hot ground truth



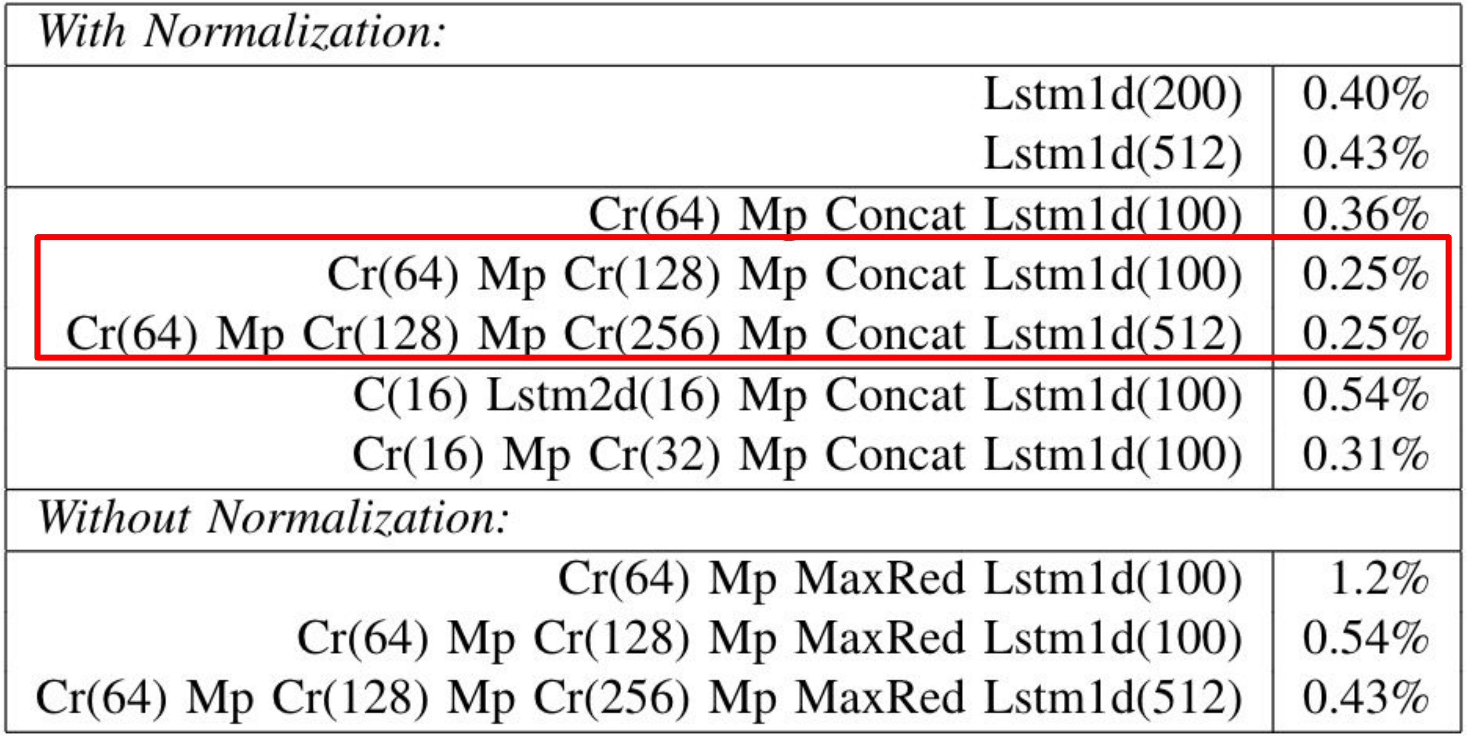
CTC-aligned ground truth

# Dataset

- Training and testing on University of Washington Database 3
- 1600 pages of scanned books, journal articles, etc.
- Physical degradation (repeated copying)
- About 100000 lines total (fairly small, but good for testing)
- 9:1 training/test split

`http://tmbdev.net/ocrdata-split`

# Experimental Results

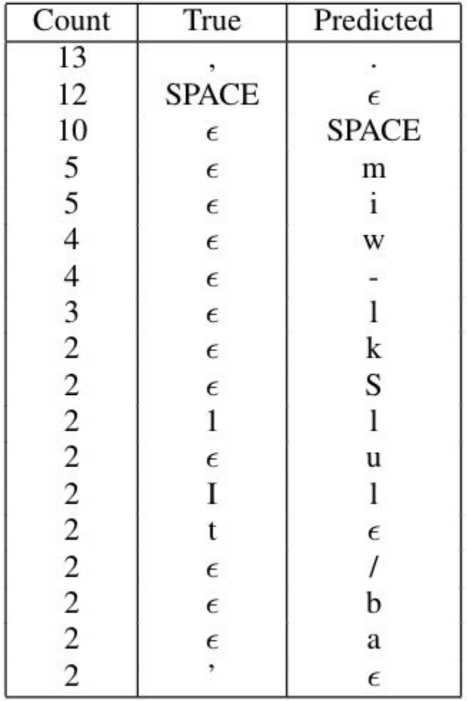


| | |
|---|---|
| *With Normalization:* | |
| Lstm1d(200) | 0.40% |
| Lstm1d(512) | 0.43% |
| Cr(64) Mp Concat Lstm1d(100) | 0.36% |
| Cr(64) Mp Cr(128) Mp Concat Lstm1d(100) | 0.25% |
| Cr(64) Mp Cr(128) Mp Cr(256) Mp Concat Lstm1d(512) | 0.25% |
| C(16) Lstm2d(16) Mp Concat Lstm1d(100) | 0.54% |
| Cr(16) Mp Cr(32) Mp Concat Lstm1d(100) | 0.31% |
| *Without Normalization:* | |
| Cr(64) Mp MaxRed Lstm1d(100) | 1.2% |
| Cr(64) Mp Cr(128) Mp MaxRed Lstm1d(100) | 0.54% |
| Cr(64) Mp Cr(128) Mp Cr(256) Mp MaxRed Lstm1d(512) | 0.43% |

lastest result

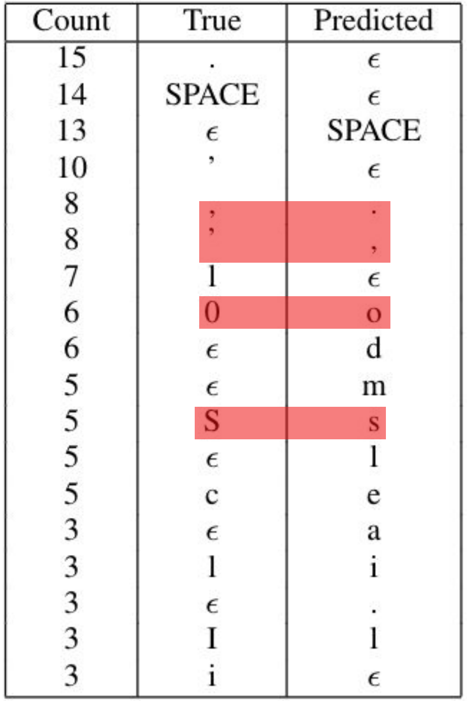
0.17%

# Top Confusions



| Count | True | Predicted |
|---|---|---|
| 13 | , | . |
| 12 | SPACE | ϵ |
| 10 | ϵ | SPACE |
| 5 | ϵ | m |
| 5 | ϵ | i |
| 4 | ϵ | w |
| 4 | ϵ | - |
| 3 | ϵ | l |
| 2 | ϵ | k |
| 2 | ϵ | S |
| 2 | 1 | l |
| 2 | ϵ | u |
| 2 | I | l |
| 2 | t | ϵ |
| 2 | ϵ | / |
| 2 | ϵ | b |
| 2 | ϵ | a |
| 2 | ' | ϵ |

Normalized



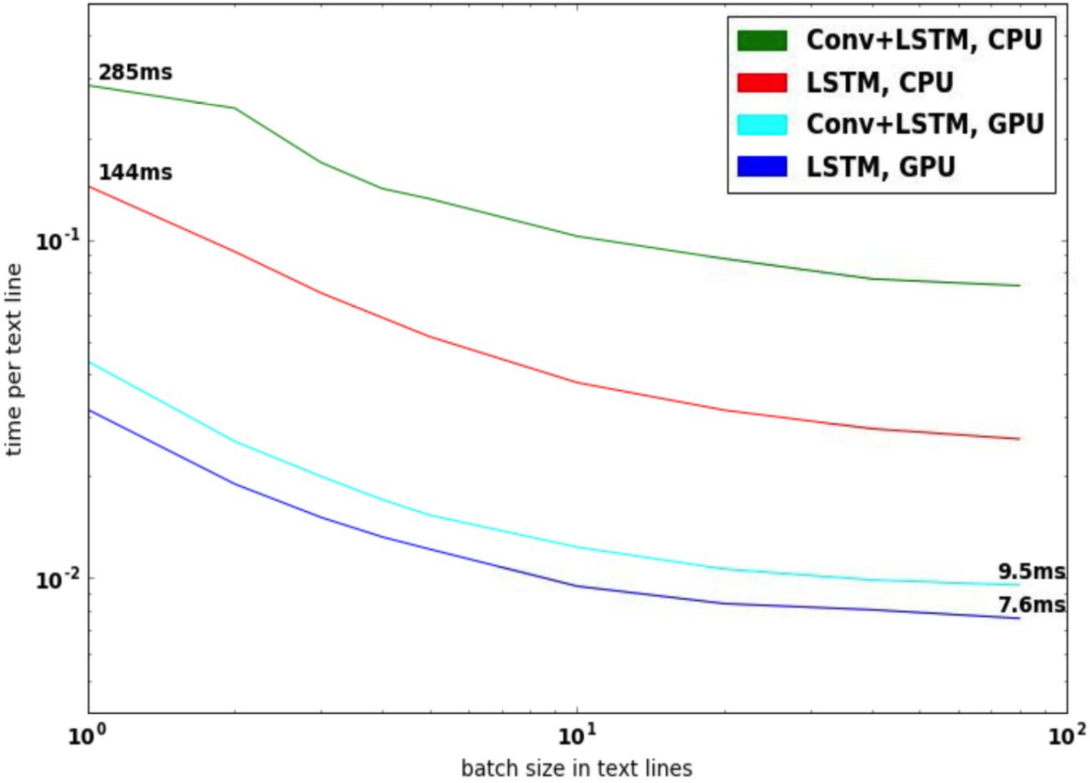
| Count | True | Predicted |
|---|---|---|
| 15 | . | ϵ |
| 14 | SPACE | ϵ |
| 13 | ϵ | SPACE |
| 10 | ' | ϵ |
| 8 | , | . |
| 8 | ' | , |
| 7 | l | ϵ |
| 6 | 0 | o |
| 6 | ϵ | d |
| 5 | ϵ | m |
| 5 | S | s |
| 5 | ϵ | l |
| 5 | c | e |
| 3 | ϵ | a |
| 3 | l | i |
| 3 | ϵ | . |
| 3 | I | l |
| 3 | i | ϵ |

Collapsed

Confusions due to poor context modeling.

Fig. 4. The most common errors made by the recognizers are confusions between period and comma, insertions and deletions of spaces, and insertions of spurious characters ($\epsilon$-to-something errors). The left results are for Cr(64) Mp Cr(128) Mp Concat Lstm1d(100); the right model is Cr(64) Mp Cr(128) Mp Cr(256) Mp MaxRed Lstm1d(512), and both evaluations are on the same randomly selected test set from UW3.

# Running Times per Text Line



note log scale

# Open Source Status

All code written in PyTorch

Official OK for open sourcing, just cleaning it up:

- dlinputs: open sharded record file inputs for distributed training
- dlmodels: compact network notation with size inference
- dltrainers: simple training wrappers
- ocropy2: top level OCR drivers

Code on Github under NVLabs and tmbdev

# Results

- explicit 1D normalization still outperforms MDLSTM
- convolutional + max pool + LSTM outperforms just LSTM
- can achieve 20x speedups on GPU for LSTM / conv+LSTM

# Next Steps

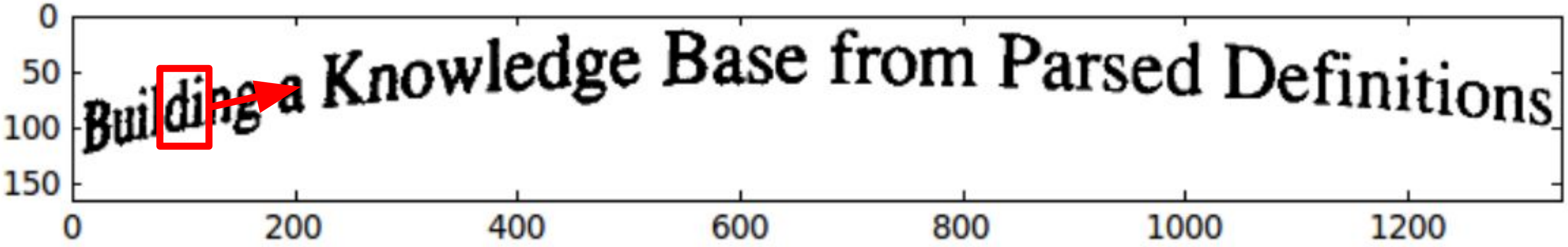
Achieve True End-to-End OCR

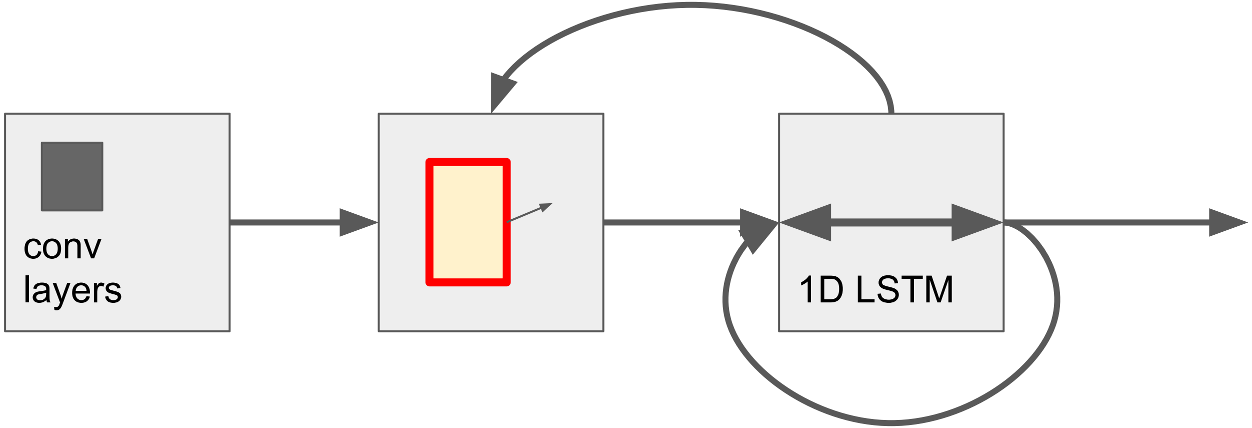
- eliminate explicit input normalization
- eliminate output decoding by thresholding
- fully self-training OCR

Possible Approaches

- seq2seq methods
- attention
- recognition lattice generation

# Attentional Mechanisms (seq2seq)


Building a Knowledge Base from Parsed Definitions


conv layers
1D LSTM