Script Agnostic OCR for Complex Scripts

Rakesh Achanta
PhD Candidate
Department of Statistics
Stanford University
rakesha@stanford.edu

September 4, 2019

1 Goal

The aim of this project is to create a simplified Optical Character Recognition(OCR) framework using the latest advances in Deep Learning. OCR systems tend to be very specific to the script they are being designed for, thus limiting their adaptivity to other world languages. We intend to bridge that gap by building an OCR that is script agnostic and one that can be easily trained for a new language.

2 Methodology

A line of text is extracted from a printed image (this can be done in a script agnostic way). This slab of pixels is fed into a Recurrent Neural Network (RNN) one column at a time. The RNN is thus reading the text image slab from one end to the other. At training time, we have access to both the image slab and the Unicode string that it represents. However, we do not have access to the location of each character in the image. Hence a simple cross-entropy loss will not work.

Connectionist Temporal Classification (CTC)[1] is a loss function that is dynamically defined on the series of softmax firings of the RNN. i.e. the CTC loss layer takes in the output of the softmax layer at all 'time's and calculates how well the softmax firings match the target labelling. The key being that the CTC's loss is invariant to the location of the softmax firing. For a demo, see the github repository [2].

In Summary, we generate artificial training samples (in the absence of a big enough real one) of images and the unicode text. We define the RNN along with the CTC cost. And train it in an online fashion. Sample unicode text comes from an n-gram model and is rendered into text using various fonts of the language.

3 Challenges

- CTC is hard to implement symbolically*
- CTC is hard to train
- The underlying character-set could run into the hundreds making training harder
- Most complex scripts do not have a one-to-one mapping from a unicode character to a printed glyph (e.g:- the *virāma* in Indic languages, who function is to fuse two characters together)

- Artificially rendering complex scripts in python*
- Building a simple n-gram model (from a collected corpus of unicode text)†
- Securing fonts†
- etc.

* already implemented.
† implemented for Telugu.

4 Current Status

I have the CTC implementation in Theano, along with some basic RNNs (including BDLSTM). There is also the script to train the RNN. Currently it works on toy examples like Hindu-Arabic numerals (10 classes) and low resolution ASCII characters (84 characters) (with some effort). It also has features for various kinds of optimizers like Momentum SGD, Adagrad, RMSprop etc.

5 Your Job

Your job is to build an OCR for your language of choice. This involves:

1. Collect a modest sized Unicode corpus for the data.
2. Build a bi/trigram model that can generate 'vaild'ish text from the language.
3. Collect as many unicode fonts as you can.
4. Build a generator for sample text images from the language (you may use the code from the github repo [3])
5. Specify an RNN model using the github repo [2]
6. Train the RNN using the same repo [2]

If you are training for Telugu, you could just use the text generator provided in repo [3].

6 Main Challenge

RNN CTC models are notoriously hard to train. Especially given that we are training them in an online fashion, where the gradients are rather noisy. You will have to play around with various network configurations and training algorithms. You might also have to use another encoding scheme to represent ligatures (rather than plain unicode).

References

[1] A. Graves. Supervised Sequence Labelling with Recurrent Neural Networks. Textbook, Springer, 2012. *Chapter 7*
[2] R. Achanta. RNN with (BD)LSTM with CTC implemented in Theano. Includes a Toy training example. `https://github.com/rakeshvar/rnn_ctc`
[3] R. Achanta. Telugu OCR framework using RNN, CTC[2] in Theano.
`https://github.com/rakeshvar/chamanti_ocr`