computergraphics tübingen

Reading scene text in deep convolutional sequences

Jan Haug, Marius Hobbhahn, Roman Schulte

April 2, 2018

Winter Term 2017
Practical Course on Computer Graphics

Reading scene text in deep convolutional sequences

Jan Haug, Marius Hobbhahn, Roman Schulte

Figure 1: You can place a teaser here.

Abstract

1 Introduction

Natural scene recognition is one of the most important steps in automation. Self-driving cars will have to be able to identify construction signs to follow their instructions, traffic agencies can identify license plates of traffic offenders and robots will need to read text and numbers in a natural environment. In 2015 He et. al [HHQ+15] proposed a model to read scene text in natural environments with a combination of a convolutional neural network (CNN) and recurrent neural networks (RNN) that not only allows to interpret words but also sequences of numbers or letters without any representation in a dictionary, i.e. an advertisement of a product. One of the main challenges of scene recognition is the high variance in image quality and style of displayed signs. An advertisement might look very different than the same word handwritten on paper even though it conveys the same meaning. The approach chosen by the authors has some advantages over previous approaches (i.e. which are not able to capture all representations of strings in a natural environment. It tries to use both the advantages for character recognition of CNNs and the context information for sequences of RNNs and combine them in one model:

1. CNNs have become very accurate at character recognition and building high level representations (citation needed). By moving a sliding window over a given picture, we are able to input single signs without worrying about context information at this stage. The simplicity of a sliding window approach implies that some frames will not represent anything meaningful but these problems will be corrected by the RNN. On the other hand this simplicity means no complex algorithm needs to be evolved and it can be applied on any given input image.

2. RNNs are very strong at accounting for context information (citation needed). Especially for representations that come in sequences, like words, this is valuable. Due to this the RNN can make decisions for a sign based on all previously identified signs, i.e. The letter 'a' might be more likely to occur after a 'b' than an 'y'. To use context information from previous and later signs we use a bidirectional approach, that independently applies the model forwards and backwards.

3. If property 1 and 2 are combined successfully it is possible to Process unknown words and arbitrary strings. Since the training is done in a compositional way it is independent of dictionaries or corpora of already known words and combinations of strings. Therefore it would be able to read the information on any given picture even if no meaning is attached to it.

The main contribution of this paper was to im-

prove the accuracy in scene text recognition on given benchmark test sets like the IIT5K. Our contribution is merely to apply the same methods on the same test sets and see if we are able to achieve the same results.

2 Related Work

[HHQ+15]

Figure 2: This is a figure.

An example is given in Figure 2.

3 Structure of the network

After applying a sliding window on a given image each frame is forwarded into a CNN. The sequence of resulting feature vectors is then sent through the RNN and results in a sequence of characters and numbers. An overview of the whole network is given in graphic

3.1 Data wrangling

We rescale the image to height 32 and then apply a sliding window of framesize 32x32 for every 8 pixels of length. When reaching the end of the image we adjust the current frame such that the last frame is 32x32 and includes the column of pixels on the right edge of the image.

3.2 Structure of the CNN

The CNN has 5 convolutional layers. In the first 3 we apply a 9x9 kernel on the inputs and use a max. group of 2. For the 4th layer a 8x8 kernel and a max group of 4 is used to reduce the dimensionality of the output vector to 128. Note that this output of a 128D feature vector is forwarded into the RNN. The 5th layer uses a 1x1 kernel and also uses a max group of 4 to further reduce dimensionality. This output is then forwarded through a fully connected layer with 36 outputs for the 26 possible lower case characters and 10 numbers. For activation we use the identity function. Note that the last convolutional and the fully connected layer are merely used for the training of the CNN but later not used for the classification in the RNN. A detailed description of the network can be seen in graphic

3.3 Structure of the RNN

The recurrent part of the network receives a sequence of feature vectors, each of which is provided by the CNN. We use bidirectional stacked long-short-term-memory (LSTM) cell blocks with 128 inputs respectively. Bidirectional means that the RNN essentially has 2 independent units. In one the sequence is put in forwards and the other backwards. The results are then concatenated afterwards and fed into a fully connected layer with 37 output classes. 26 for each character 10 for numbers and 1 for undetectable sign or no character.
At this point we still have redundancies or unclear information due to the sliding window. To solve this problem a connectionist temporal classification (CTC) is used. A detailed explanation would be outside of the scope of this paper and can be found here But for a working explanation consider it as a way to remove unlikely information given the labels. In sequential information the placement of non-character symbols or redundancies is always hard to deal with. The CTC approach therefore uses a hidden Markov model (HMM) to sample different possible outcomes as paths and choose the path with highest likelihood. Consider an example where the sequence after the LSTM cell block is ”iiii—mmmmaaagggee–”, where - represents the non-character symbol. The CTC would then reduce the redundancies and remove non-character symbols until the most likely outcome, ”image” is put out. The information learned by the CTC during the training are backpropagated through the network such that future classification can already access them.

4 Implementation details

4.1 Data wrangling

Sliding window

Batch sizes

4.2 Implementation details of the CNN

The convolutional layer were implemented using the conv2D() function provided by the tensorpack library and combined with a self-made maxout() function, since the tensorflow version had an error.

Instead of a softmax in the end, like in the original paper, we chose a fully connected layer with identity as activation function since it lead to a higher accuracy for the character recognition.
For the training we use a learning rate with exponential decay starting at 0.001 with decay rate of 0.99 and getting smaller every 5 epochs. As an optimizer we chose the AdamOptimizer() and used the prebuild InferenceRunner() from tensorpack to check for validation accuracy on a test set and prevent overfitting. Overall we train for 1500 epochs and use the prebuild SimpleTrainer() from tensorpack.

4.3 Implementation details of the RNN

The RNN is implemented using the prebuild tensorflow functions tf.nn.bidirectional_dynamic_rnn with tf.nn.BasicLSTMCell for both directions. We use 128 units as described in the original paper and a tanh as activation function. For the fully connected layer on top we use tf.contrib.layers.fully_connected with identity as activation.
The CTC exists prebuild as well. For the loss we use the tf.nn.ctc_loss and tf.nn.ctc_greedy_decoder for the model. The greedy decoder is a variant of the tf.nn.ctc_beam_search_decoder function that is faster but less accurate. For our training the beam search decoder was too slow and in comparison did not add any visible improvements during the first couple of steps in training which lead us to using the greedy decoder.
For the training we use the Adam optimizer with a learning rate of 0.0001 and a momentum term of 0.9 over 1500 epochs. Since we divided our data set in test and train split we also use the InferenceRunner() class provided by the tensorpack library to prevent overfitting and check for validation accuracy.

5 Experiments and Results

The test images solely came from the openly available IIT5K data set. In the original study two other set of images were additionally used but seemed to only contribute very small amounts of images. So only using the IIT5K seemed sufficiently justifiable in this case. As the name suggests we initially have 5000 images of which a small amount was sorted out since they did not have 32 pixels in width. After segmentation the CNN trains on labeled 32x32 pictures and the RNN on around 2500 pictures with scene text due to the test train split. The experiments were conducted on the just described data sets using the training details listed in the respective implementation details section.
The CNN converges against an accuracy of 0.97 on the training set with a validation accuracy of 0.62. The process can be viewed in graphic The validation accuracy seems rather low for a task like character recognition. This might be due to overfitting, very unclear images or just a suboptimal model. But note that the 5th and 6th layer of the CNN never get used at later stages anyways so only the accuracy of the 128D feature vectors matters. This can not be determined directly but only through the loss of the RNN at later stages.
The RNN converges against an accuracy of 1 on the training set and against 0.66 for the validation. Note here that this is an increase compared to the 0.62 of the CNN solely, meaning the context of the word significantly added to identification of the letters. Additionally most words that are not correctly identified are only off one letter. For example the word "wraps" was identified as "wras" and "chandigarh" became "chandgarh" as depicted in
In comparison to the originial, however, our results are rather unspectacular. Their accuracy was between 91,5 and 94 percent for the IIT5K data set and even identified very ambiguous pictures.

6 Conclusion

The results that we achieved where slightly less impressive than those of the original publication. There are a couple of improvisations that could be done in future research projects that we would like to outline. A first idea would be to train the network not in 2 separate steps but rather as one end-to-end process. This would likely result in better outcomes since the entire CNN would already get information from the RNN and CTC through backpropagation. A second idea is to use a bigger data set of images for training. The IIT5K seems like a good benchmark but due to the test train split overfitting might have occured which might be solved by significantly bigger datasets. A third possibility is to train the CNN and RNN on significantly lower sliding window frame sizes, i.e. 2 instead of 8
The paper we are replicating was published in summer 2015 which is already two and a half years back.

In a field with such rapid changes like scene recognition other projects have already further improved the methods that were revolutionary then. In the following we would like to have a closer look at some of the new ideas and improvements.

References

[HHQ+15] P. He, W. Huang, Y. Qiao, C. Change Loy, and X. Tang. Reading Scene Text in Deep Convolutional Sequences. *ArXiv e-prints*, June 2015.