Convolutional Recurrent Neural Network for labeling unsegmented sequences in blurry and noisy invoices

Ahmed MAZARI

June 5, 2017

Abstract

A summary of Convolutional Recurrent Neural Network for sequence prediction is given . The article proposes an architecture that predicts sequence labels without any pre-segmented inputs or post-process outputs. The pipeline stacks three neural networks models. CNN which takes the input images, Bidirectional Long Short Term Memory (BLSTM) and Connectionist Temporal Classification (CTC). It's an end-to-end trainable neural network. Then the model is applied to correct automatically OCR errors on a noisy and a blurry invoices.

Keyword : CNN, BLSTM, CTC,HMM

1 Introduction

Optical character recognition (OCR) is a challenging and ubiquitous problem. From one hand, characters of the same class can take different shapes and representations. From other hand, in real life, characters are present in blurry and noisy environment.
Due to this huge variability within the same class a rich data representation that covers the variability of data is needed to avoid misclassification. Such as : S and 5 , 6 and 9. A versatile approach to handle such problem is to get a balanced dataset characterized by a comparable amount of examples for each class. However, the more we extend our alphabet, the more the problem becomes difficult to solve.
To cope with these constraints a model should be able to :

- minimize the variance intra-class
- maximize the variance inter-class
- be insensible to data transformation and perturbation
- be robust to temporal and spatial noise

The second issue is how to correctly predict the label of a given character when they appear in noisy and blurry context ?
A given character remains the same either in a clean or noisy environment. But, when the character appears in a noisy context, its shape may take an analogous form to other class(es), it's due to shape perturbation and characters overlapping. Moreover, the variable font-size and background color of each character render the task more challenging. As a consequence, it gives rise to spatial distance learning between characters issue. How to learn this metric ? What are its underlying properties and criteria ? Taking into consideration this huge variability in terms of :

- Font size
- Background color
- Characters overlapping
- Shape
- Noise
- Blur
- Spatial distance between characters

A model should be able to learn complex regularities and hierarchical representation of data . It's all about learning hierarchical invariant in high dimensional space in order to discriminate the classes.

To solve that, we need to look for a model which is able to take into consideration all the aforementioned constraints and responds to the following questions :

- Is the model able to label unsegmented sequences (images) ?
- Does the model require to post process the outputs to map them with labels ?
- How can the model segments the characters of a sequence ?
- Is the model robust to spatial and temporal noise ?

Currently, probabilistic graphical models such as hidden Markov model (HMM), conditional random fields (CRF) are predominant frameworks for sequence labeling. Besides, the resurgence of deep neural networks makes learning models on bigdata much more tractable. In this context, recurrent neural network (RNN) are well suited and outperforms the aforementioned probabilistic models because RNNs are able to learn long term dependencies and tend to be robust to temporal and spatial noise.

Even if RNN, HMM, CRF have enjoyed lots of success in language modeling, they have several drawbacks (are discussed in the next section) when it comes to label unsegmented data.

2 Advantages and drawbacks of probabilistic graphical models and recurrent neural networks

Advantages :

- RNNs require no prior knowledge of the data beyond the choice of the input and output representation
- RNNs can be trained discriminatively
- The internal state of RNN provides a powerful, general mechanism for modeling time series
- RNNs tend to be robust to temporal and spatial noise

Drawbacks :

- They require pre-segmented training data
- They require post processing to transform their outputs into label sequences
- They require a significant amount of task specific knowledge : to design the state models for HMMs and choose the input features for CRFs
- They require explicit dependency assumptions to make inference tractable such as the assumptions that observations are independent for HMMs
- Standard HMMs, training is generative, even though sequence labeling is discriminative
- Not possible to apply directly RNN for sequence labeling, because the standard neural network objective functions are defined separately for each point in the training sequence
- RNN can be trained to make a series of independent label classifications, so they require data to be pre-segmented and the network outputs must be post-processed to give the final label sequence
- RNN requires a preprocessing step that converts an input image into a sequence of images features. For instance word image into sequential HOG features

Due to the aforementioned drawbacks, the most effective RNN is to combine it with HMM which gives an hybrid system. It can be justified by the fact that HMMs are used to model the long range sequential structure of the data and RNNs are used to provide localized classifications. However, this hybrid model suffers from the drawbacks of HMMs, thus RNN doesn't exploit its full potential for sequence modeling.
In the next section, we present a model that handles all the aforementioned drawbacks and solves the problem of labeling unsegmented sequence without pre-processing the inputs or post-processing the outputs.

3 Convolutional Recurrent Neural Network for labeling unsegmented sequence

Convolutional recurrent neural network (CRNN) [3] is an end-to-end trainable neural network that integrates features extraction, sequence labeling and transcription. It handles sequences of arbitrary length with no character segmentation or horizontal scale normalization. It's not confined to any predefined lexicon.

However, standard deep neural network can't be applied to sequence prediction due to sequence length variability where they often operate on inputs and outputs with fixed dimensions. Thus, they are incapable of producing a variable length label sequence.

4 CRNN architecture :

The CRNN stacks three models : CNN, RNN (BLSTM) and CTC as it's depicted in 1

- CNN extracts feature sequence from each input image.
- RNN is build on the top of CNN, it has to make prediction for each frame of the feature sequence outputted by the convolutional layer.
- CTC which is called transcription layer. It's built on the top of BLSTM. It translates the per-frame predictions by the recurrent layers into a label sequence.

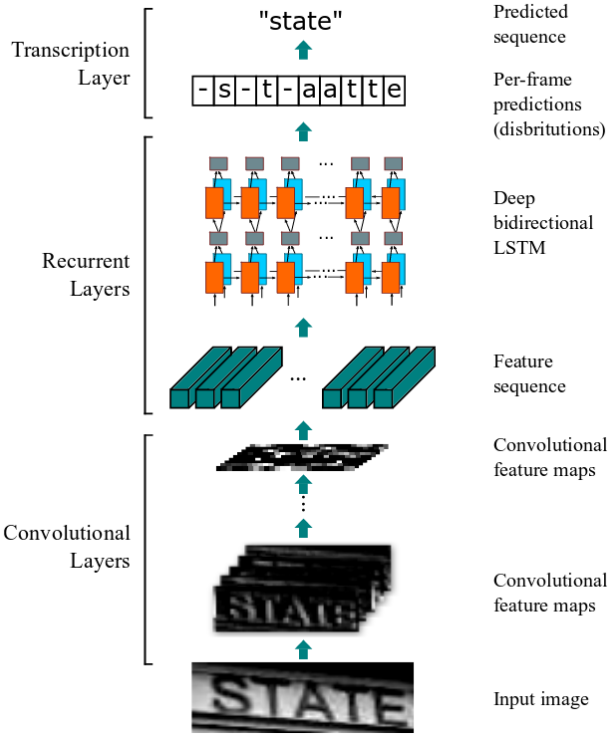

Figure 1: the network architecture

During the process of features extraction. The CNN alternates between convolutional and max pooling layer

which allows to build sequential features representation from an input image. The image should be scaled to the same height (32) before being fed to the network. A sequence of feature vectors is extracted from the feature maps produced by the component of convolutional layers which is the input of feature vectors. Each feature vector is extracted from the left to right on the feature maps by column. This means that the $i-th$ feature vector is the concatenation of the $i-th$ columns of all the map. The width of each column is fixed to a single pixel. Each column of the feature maps corresponds to a rectangle region of the original image. Such rectangle regions are in the same order to their corresponding columns on the feature map from left to right. Each feature vector in the feature sequence is associated with a receptive field which can be seen as the image descriptor for that region. hence, in CRNN we convey deep features into a sequential representation in order to be invariant to the length variation of sequence-like objects. Since convolution, max pooling, element wise activation function operate on local region, they are translation invariant.

The RNN is built at the top of DCNN to predict a label distribution for each frame in feature sequence. it has a strong capability of extracting contextual information within a sequence. For instance ambiguous characters are easier to distinguish when observing their context i and L : the height makes the difference rather than recognizing them separately (wide characters take more frames). RNN is powerful when it comes to capture and store past context. Each time it receives a frame x_t in the sequence, it updates its internal state h_t with a non linear function that takes both current input c_t and past state h_{t-1} as its inputs : $h_t = g(x_t, h_{t-1})$, then the prediction y_t is made based on h_t. In this way, past context $x_{t'}$ where $t' < t$ are captured and utilized for prediction.
However, RNN suffers from the vanishing gradient problem, hence they store a limit range of contexts. Long short term memory (LSTM) are designed to address this problem. It consists of a memory cell and three multiplicative gates, namely the input, output and forget gates. Conceptually, the memory cell stores the past contexts, and the input and output gates allow the cell to store contexts for a long period of time. Meanwhile, the memory in the cell can be cleared by the forget gate. In image-based sequences, LSTM is important since it capture long-range dependencies. But LSTM is directional, it only uses past contexts from both directions . However, in image based sequence, context from both directions are useful and complementary to each other. Therefore, we combine to LSTM one forward and another backward which creates a a bidirectional LSTM. Furthermore, multiple LSTM are stacked which results a deep bidirectional LSTM.

The transcription layer convert per-frame prediction of BLSTM into a label sequence. It takes sequence like $-h-l-ll-o-$ and transform it to *hello*. It is defined by the following conditional probability :

$$p(l|y) = \sum_{\Pi:\beta(\Pi)=l} p(\Pi|y) \tag{1}$$

Such that : $y = y_1, ..y_T$ is the per-frame prediction and l is the true label.
A sequence-to-sequence mapping function is defined on sequence Π. β maps Π onto l by firstly removing the repeated labels. Then, the conditional probability is defined as the sum of probabilities of all Π that are mapped by β onto l as described in *equation*(1).

The *equation*(1) would be computationally infeasible due to the exponentially large number of summation items. However, it can be computed using the backward forward algorithm in [5]. In lexicon-free transcription we take the argmax, which means taking the most probable label Π_t at each time stamp t and map the resulted sequence onto $l*$..

5 End-to-end network training

The objective function minimizes the negative log likelihood of the conditional probability of ground truth :

$$O = -\sum_{I_i, l_i \in X} logP(I_i|y_i) \tag{2}$$

such that :

- $X = (I_i, l_i)_i$ that presents respectively training image and ground truth label
- y_i the sequence produced by the recurrent and convolutional layer.

The cost value is directly computed by the training image and the ground truth label. It eliminates the need of manually labelling individually each character. The network learns to crop and segment each character.

The network is trained using stochastic gradient descent. Gradients are calculated by the back propagation algorithm.

The architecture is end to end trainable. However, CNN, BLSTM and CTC are backpropagated with different algorithms

- CNN layer is trained with the standard stochastic gradient descent.
- BLSTM layer is trained with back propagation through time (BPTT)
- Transcription layer (CTC) is trained with forward backward algorithm described in [6]

For optimization, per dimension learning rate is calculated by $ADADELTA$ [7] . Comparing $ADADELTA$, $ADAGRAD$ and conventional *momentum*, it turned out that $ADADELTA$ requires no manual setting of a learning rate. Thus, in this context $ADADELTA$ converges faster than the momentum method.

6 Implementation details

The stride is special designed for $height = 32$. Heigh is a discriminative feature in the context of character recognition.

All the images are converted into gray scale because sequence is more about shape and structure.
In the CTC [5] model the inputs are supposed to be words without blanks. In other term, CTC takes sequence of words and not sentences.
In practice the maximum length of sequence is 69 characters. Constraint required by $CUDA$ parallel computing.

The architecture of CNN layer is based on the VGG deep network [8] where a tweak is made to make it suitable for recognizing english text sequences mainly.
In order to support large feature sequence a tweak is made to yield feature maps with larger width. The tweak is associated with the 3rd and 4th max pooling layer where rectangular of $1 * 2$ are set rather than the conventional squared $2 * 2$.

On the top of rectangular pooling windows , yields rectangular receptive field 3.They are beneficial for recognizing some characters that have narrow shapes, such as I and i.
Since the CRNN is a deep architecture, batch normalization layer is put after the forth and fifth convolutional layer. It helps to train deep networks and accelerates the training. $ADADELTA$ sets the learning rate to 0.9.
The inner structure of CNN is depicted in 2.

Table 1. Network configuration summary. The first row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively

Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	W × 32 gray-scale image

Figure 2: Table description of CNN layers

6.1 Advantages of CRNN :

- No character level annotation is required. CRNN can learn directly from unsegmented sequence.
- It can learn informative representation with neither hand-craft features nor preprocessing steps, including binarization, segmentation and object localization that DNN requires
- It has the same property of RNN : being able to produce sequence label since it implement a variant of RNN which is called bidirectional long short term memory (BLSTM).
- It in unconstrained to the length of sequence-like objects, requiring only height normalization. Height normalization it's a discriminative parameter because it allows to discriminate between characters thanks to their heights
- It contains much less parameters than a standard DCNN, consuming less storage space. 8.3 million parameters for CRNN [3] and 304 million parameter for DCRNN [4]

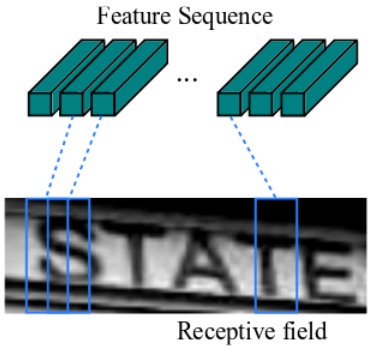

Figure 3: he receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field

7 Comparative evaluation

- CRNN outperforms most state-of-the-arts approaches, in average beats the best text reader ($ABBY$) proposed in [9]
- Unlike [9], CRNN is not limited to recognize a word in a known dictionary, and able to handle random strings such as telephone numbers, sentences and Chinese words.
- [8] has 490 million parameters and CRNN has 8.3 million parameters to learn taking only 33MB RAM, using 4-bytes single precision float for each parameter. Thus, it can be ported to mobile devices.
- CRNN is able to take input images of varying dimensions and produces predictions with different lengths.

8 Conclusion

CRNN shows an interesting results where it performs state of art algorithms. However, CRNN is slow in prediction due to the complexity of the architecture. So, the next tempting track toward a more powerful CRNN is to speed it up so that to make it more practical in real world application.

References

[1] Yarin Gal, Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. NIPS 2016

[2] Sergey Ioffe, Christian Szegedy.Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.2015

[3] Baoguang Shi, Xiang Bai and Cong Yao .An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. December 2016.

[4] Max Jaderberg, Karen Simonyan, Andrea Vedaldi and Andrew Zisserman. Deep Structured Output Learning for Unconstrained Text Recognition. In ICLR, 2015.

[5] A.Graves, S. Fernandez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In, ICML, 2006

[6] A. Graves, S. Fernandez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML , 2006

[7] M. D. Zeiler. ADADELTA: an adaptive learning rate method. CoRR , abs/1212.5701, 201

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 201