Recursive Recurrent Nets with Attention Modeling for OCR in the Wild

Chen-Yu Lee[†]
UC San Diego
chl260@ucsd.edu

Simon Osindero
Flickr / Yahoo Inc.
osindero@yahoo-inc.com

Abstract

We present recursive recurrent neural networks with attention modeling (R^2AM) for lexicon-free optical character recognition in natural scene images. The primary advantages of the proposed method are: (1) use of recursive convolutional neural networks (CNNs), which allow for parametrically efficient and effective image feature extraction; (2) an implicitly learned character-level language model, embodied in a recurrent neural network which avoids the need to use N-grams; and (3) the use of a soft-attention mechanism, allowing the model to selectively exploit image features in a coordinated way, and allowing for end-to-end training within a standard backpropagation framework.

We validate our method with state-of-the-art performance on challenging benchmark datasets: Street View Text, IIIT5k, ICDAR and Synth90k.

1. Introduction

Photo Optical Character Recognition (photo OCR), which aims to read scene text in natural images, is an essential step for a wide variety of computer vision tasks, and has enjoyed significant success in several commercial applications. These include street-sign reading for automatic navigation systems, assistive technologies for the blind (such as product-label reading), real-time text recognition and translation on mobile phones, and search/indexing the vast corpus of image and video on the web.

The field of photo OCR has been primarily focused on constrained scenarios with hand-engineered image features. (Here, constrained means that there is a fixed lexicon or dictionary and words have known length during inference.). Specifically, examples of constrained text recognition methods include region-based binarization or grouping [5, 24, 33], pictorial structures with HOG features [47, 46], integer programming with SIFT descriptor [41], Conditional Random Fields (CRFs) with HOG features [32, 31, 39], Markov models with binary and connected component features [49]. Some early attempts [26, 53, 10] try to learn local mid-level representation on top of the hand-crafted features, and some methods in [48, 19, 16] incorporate deep convolutional neural networks (CNNs) [25, 13] for a better image feature extraction. These methods work very well when candidate ground-truth word strings are known at testing stage, but do not generalize to words that are not present in the list of a lexicon at all.

A recent advance in the state-of-the-art that moves beyond this constrained setting was presented by Jaderberg *et al*. in [17]. The authors report results in the unconstrained setting by constructing two sets of CNNs – one for modeling character sequences and one for N-gram language statistics – followed by a CRF graphical model to combine their activations. This method achieved great success and set a new standard in photo OCR field. However, despite these successes, the system in [17] does have some drawbacks. For instance, the use of two different CNNs incurs a relatively large memory and computation cost. Furthermore, the manually defined N-gram CNN model has a large number of output nodes ($10k$ output units for N = 4), which increases the training complexity – requiring an incremental training procedure and heuristic gradient rescaling based on N-gram frequencies.

Inspired by [17], we continue to focus our efforts on the unconstrained scene text recognition task, and we develop a recursive recurrent neural networks with attention modeling (R^2AM) system that directly performs image to sequence (word strings) learning, delivering improvements over their work. The three main contributions of the work presented in this paper are:

(1) Recursive CNNs with weight-sharing, for more effective image feature extraction than a "vanilla" CNN under the same parametric capacity.

(2) Recurrent neural networks (RNNs) atop of extracted image features from the aforementioned recursive CNNs, to perform implicit learning of character-level language model. RNNs can automatically learn the sequential dynamics of characters that are naturally present in word strings from the training data without the need of manually defining N-grams from a dictionary.

(3) A sequential attention-based modeling mechanism that

[†]Author to whom correspondence should be addressed.

This work is supported by Flickr Vision and Machine Learning Team.

performs "soft" deterministic image feature selection as the character sequence is being read, and that can be trained end-to-end within the standard backpropagation.

We pursue extensive experimental validation on challenging benchmark datasets: Street View Text, IIIT5k, ICDAR and Synth90k. We also provide a detailed ablation study by examining the effectiveness of each of the proposed components. Our proposed network architecture achieves the new state-of-the-art results and significantly outperforms the previous best reported results for unconstrained text recognition [17]; *i.e*. we observe an absolute accuracy improvement of 9% on Street View Text and 8.2% on ICDAR 2013.

2. Methodology

In this paper, we focus on the scene text recognition task, predicting all characters from a cropped image of single word. We refer to the cropped word region as an input image in the rest of the paper. The current section describes related literatures and the proposed architecture: Recursive Recurrent Nets with Attention Modeling (R^2AM). Figure 1 shows our overall system architecture.

2.1. Character sequence model review

Many text recognition methods focus on capturing individual characters of a word as the first step in the system pipeline, and then apply statistical language models or visual structure prediction to refine/prune-out misclassified characters as in [46, 48, 32, 4, 39, 26, 53]. However, there are significant challenges since each character is intimately positioned with respect to others within the same word, and therefore classic character recognition components need to deal with a large amount of inter-class and intra-class confusion – this is well illustrated by Figure 3 from [32]. Even in sophisticated word recognition systems incorporating higher-order language priors based on CRFs or Markov models, the overall system performance is still largely dominated by the capability of the first step of the system pipeline: the character recognition component.

Goodfellow *et al*. [9] first used a CNN with multiple position-sensitive character classifiers for street number recognition. Recently, Jaderberg *et al*. in [18, 17] proposed character sequence model that directly encodes the character at each position in the word using deep CNNs and so predicts the sequence of characters in an image region. This approach largely overcomes the aforementioned issues by directly modeling the natural spacing and overlapping patterns in scene characters that can not readily be leveraged by sliding window based character recognition methods. For details of this character sequence model please refer to [17]. We refer to this baseline method as Base CNN (and labeled in Figure 3 as Base CNN) in the rest of the paper. Our proposed system is built upon this Base CNN model; we

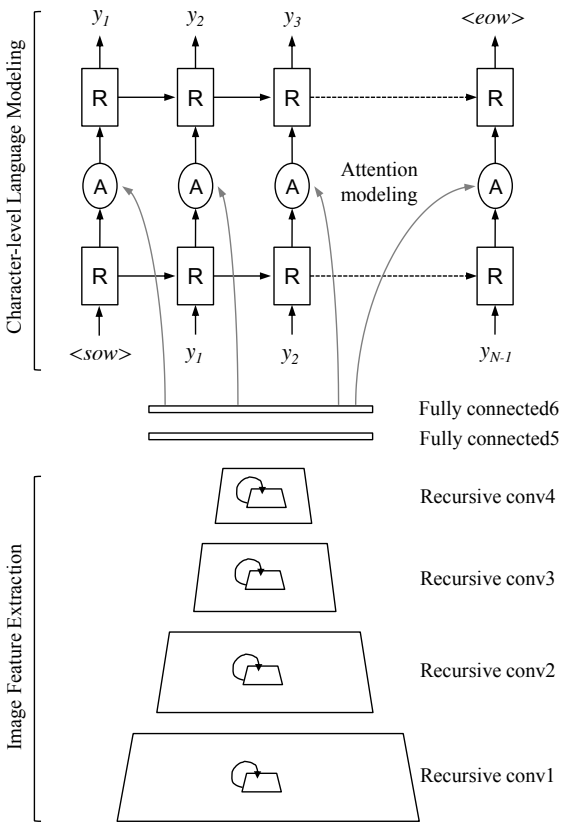

Figure 1: Recursive recurrent nets with attention modeling (R^2AM) approach: the model first passes input images through recursive convolutional layers to extract encoded image features, and then decodes them to output characters by recurrent neural networks with implicitly learned character-level language statistics. Attention-based mechanism performs soft feature selection for better image feature usage.

describe our extension of novel image encoding in Section 2.2, character-level language modeling in Section 2.3, and attention-based mechanism in Section 2.4.

2.2. Recursive CNNs for image feature extraction

2.2.1 Recursive convolutional layers

One key to the great success of the aforementioned character sequence model is the ability to capture contextual dependencies during character prediction by employing multiple convolutional layers that operate on the whole input image.

One possible way to improve upon this Base CNN model to enable even longer range contextual dependencies for character prediction would be to consider using a larger kernel size for each convolutional layer or a deeper network, increasing the corresponding receptive field size. However, this approach induces more parameters and a higher model complexity, thereby leading to potential training and gener-

alization issues.

Another way to expand longer data dependencies while controlling the model capacity is to make the Base CNN network recursive or recurrent as suggested in [35, 7, 29]. By using recursive or recurrent convolutional layers, the network architecture can be arbitrary deep without significantly increasing the total number of parameters by reusing the same convolutional weight matrix multiple times at each layer.

We now describe the recursive CNNs used in our approach: the instance of the recursive convolutional layer at time step t (where $t \geq 0$) is fed with an input image/feature response as:

$$h_{i,j,k}(t) = \begin{cases} \sigma((\mathbf{w}_k^{hh})^T \mathbf{x}_{i,j} + b_k) & \text{at } t = 0 \\ \sigma((\mathbf{w}_k^{hh})^T \mathbf{h}_{i,j}(t-1) + b_k) & \text{at } t > 0 \end{cases} \quad (1)$$

where $\mathbf{h}_{i,j}(t-1)$ and $\mathbf{x}_{i,j}$ denote the vectorized feed-forward and input patches centered at (i, j) of feature maps, respectively. $\mathbf{w}_k^{hh}$ is the vectorized feed-forward weight for output channel k. b_k is the bias for output channel k. σ is a deterministic non-linear transition function.

Recursive CNNs increase the depth of traditional CNNs under the same parametric capacity, and also produce much more compact feature response than CNNs. In a slightly different interpretation of this architecture, the recursive interactions can also be seen as implementing a form of "lateral connectivity" within a feature map, allowing the representation at a given layer to better capture higher order dependencies. For a longer discussion on recursive/recurrent convolutional layers, see [29] for details.

2.2.2 Untying in recursive convolutional layers

We have seen the definition and potential gain of recursive convolutional layers in the previous section. However, the formulation in Eqn. 1 restricts all weights $\mathbf{w}_k^{hh}$ to share the same internal values – they are "tied" together. One consequence of this tying is that the number of channels will be identical across all of the layers due to the fact that the shared weights $\mathbf{w}_k^{hh}$ always project the incoming feature maps to the same dimension (width×height×number of channels) of output feature maps. This strongly contrasts with the common practice in CNNs of varying the number of channels to control the amount of computation performed and the spectrum of different feature types. (For example the popular and successful VGGNet [40], in which the number of channels increases like {64, 128, 256, 512} as the spatial extent of the convolutional layers decreases according to {224, 112, 56, 28}.)

In this work we propose to use an "untied" variant of recursive convolutional layers that distinguishes between initial inter-layer feed-forward weight $\mathbf{w}_{\text{untied},k}^{hh}$ and the following intra-layer recursive weights $\mathbf{w}_{\text{tied},k}^{hh}$. This allows the network to have different numbers of channels at different layers, and also allows the recursive weights to specialize more freely.

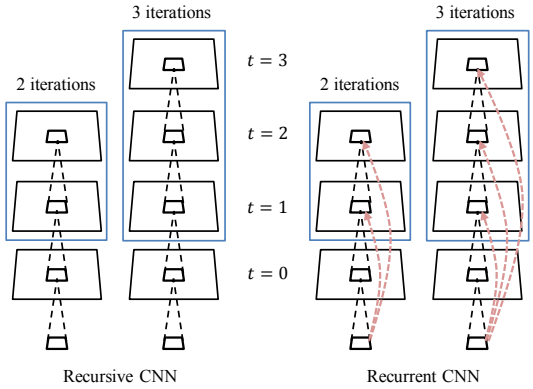

Figure 2: Illustration of the proposed untied recursive and recurrent convolutional layers. We untie the first feed-forward weight at time step $t = 0$ and the rest feed-forward weights at $t \geq 1$. The layers inside the blue box have tied (shared) weights.

By untying the feed-forward weights at time step $t = 0$, Eqn. 1 becomes:

$$h_{i,j,k}(t) = \begin{cases} \sigma((\mathbf{w}_{\text{untied},k}^{hh})^T \mathbf{x}_{i,j} + b_k) & \text{at } t = 0 \\ \sigma((\mathbf{w}_{\text{tied},k}^{hh})^T \mathbf{h}_{i,j}(t-1) + b_k) & \text{at } t > 0 \end{cases} \quad (2)$$

In doing so, the number of channels for any recursive convolutional layer can be adjusted by the untied weight $\mathbf{w}_{\text{untied},k}^{hh}$, controlling the overall computational cost. We can use the same logic here to untie recurrent convolutional layer [29]. Please see Figure 2 for an illustration.

In the experiment section we observed that both recursive and recurrent versions of Base CNN model significantly improve the performance on many recent standard benchmarks such as Synth90k, SVT, and ICDAR13. Please refer the details in Section 3.3. We further found that recursive version consistently outperforms recurrent version in all the tasks that we explored, which is in line with findings in other recent literature [42, 15] that recursive structures can learn compositional features and part interactions effectively so injecting input $\mathbf{x}_{i,j}$ multiple times at each time step is not necessary for obtaining high performance. It is also possible that recursive models are forced to more effectively use their tied weights relative to the recurrent model, since there is no option for information to "short-cut" as there is in the recurrent architecture. For this reason, we choose the recursive version of Base CNN model for our overall system pipeline as shown in the bottom part of Figure 1.

2.3. RNNs for character-level language modeling

The proposed untied recursive character sequence model in Section 2.2 can already serve as an end-to-end trainable

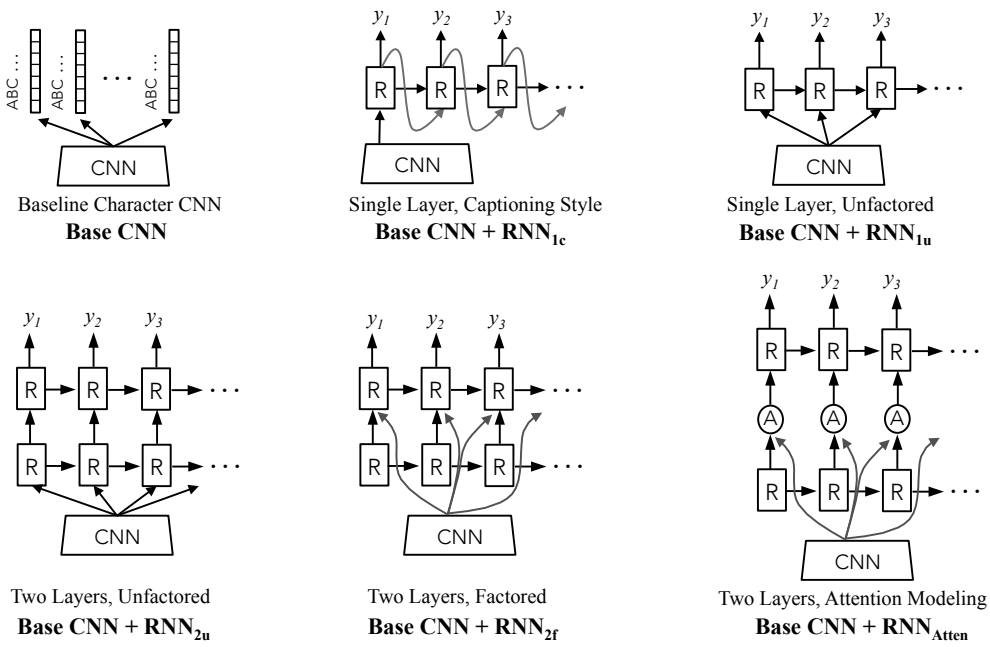

Figure 3: Five variations of the recurrent in time architecture that we experimentally evaluate for photo OCR task. We explore the image captioning style RNNs, the effect of depth in the RNN stack, the effect of factorization of the modalities (also explored in [23, 6]), and the effect of attention modeling.

photo OCR system that significantly outperforms sophisticated structured learning method in [17] by a large margin (7.2% on SVT and 6.7% on ICDAR13 absolute improvement as shown in the experiment section). Nonetheless, we observed that the Base CNN model (either plain CNNs, recursive CNNs, or recurrent CNNs) trains each character position independently by using multiple loss functions. We are then motivated to ask whether we can allow some sort of interaction between each character position and exploit the underlying character-level language statistics.

The most common approach is to add some kind of graphical models (*e.g.* CRFs) on top of the output prediction of each character position as in [31, 32, 34]. However, these methods need to compute unary and higher-order terms for all candidate characters, and can require expensive computation during inference stage. Another way to access such character-level language information is to directly model all possibilities using a CNN – as in the bag-of-N-grams component of [17]. Such a CNN model requires pre-defined N-grams from a dictionary and uses a huge number of output nodes in which each node represents an element in N-gram combinations (*e.g.* $10k$ output nodes for $N = 4$ in [17]). In order to jointly train a whole range of N-grams in this CNN model using backpropagation, the method in [17] rescales the gradients for each N-gram class by the inverse frequency of its appearance in the training word corpus because some of the N-gram classes barely occur in the training dataset, even with the recently released Synth90k [16] dataset that has more than 7 million training samples.

In contrast to the above methods, we propose to use recurrent neural networks (RNNs) [50, 38, 37] to model the character-level statistics of text. Recurrent neural networks and its variant Long Short-Term Memory (LSTM) [14] have recently experienced a renaissance and are extremely effective models when dealing with sequential data, such as handwriting recognition, machine translation, speech recognition, and image captioning. Recognizing characters in images can be essentially considered as a task of solving sequential dynamics and learning mappings from pixel intensities to natural character-level vectors. Specifically, our model takes a single image and generate a sequence of 1-of-K encoded characters.

$$\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\},\ \mathbf{y}_t \in \mathbb{R}^K \tag{3}$$

where K is the size of the possible characters and N is the length of the word.

We propose the use of an RNN that produces a word string by generating one character at every time step conditioned on image feature $\mathcal{I}$, the previous hidden state $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$ using the recurrence equations:

$$\begin{aligned} \mathbf{h}_t &= \sigma(W_{xh}\mathbf{x}_t + W_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \sigma(W_{hy}\mathbf{h}_t + \mathbf{b}_y) \end{aligned} \tag{4}$$

where σ is an element-wise non-linear transition function, $\mathbf{h}_t \in \mathbb{R}^M$ is the hidden state with M units, and the input $\mathbf{x}_t$ can be encoded image feature $\mathcal{I}$ or previously generated

character $\mathbf{y}_{t-1}$, depending on the RNN structure used. The encoded image feature $\boldsymbol{\mathcal{I}}$ is extracted from the last fully-connected layer of the a CNN model. This CNN models can be either plain CNN, recurrent CNN, or recursive CNN. We will show how different CNN models perform in the experiment section.

There are potentially many ways to feed an image feature $\boldsymbol{\mathcal{I}}$ to a RNN, and the RNN itself can also have many different structures. In this paper, we empirically explore a range of settings. Figure 3 demonstrates the base CNN model and five RNN variants that we explored. We detail four variants in this section and will explain the last variant (including attention modeling) in the next section:

Base CNN: Baseline character sequence CNN trained with multiple loss functions where each loss function focuses on one character position as described in Section 2.1.

Base CNN + RNN$_{1c}$: A single-layer RNN inspired from image-captioning work [45]. The extracted image feature $\boldsymbol{\mathcal{I}}$ is sent to RNN only at the first time step. The predicted character y_{t-1} of RNN at time $t-1$ is fed to the RNN at time t until we obtain an end-of-word (EOW) label. This variant serves as an good sanity check and helps us validate the capability of our RNN to perform character-level language modeling given an initial CNN representation.

Base CNN + RNN$_{1u}$: An unfactored single-layer RNN receiving image feature $\boldsymbol{\mathcal{I}}$ at every time step – therefore the character predictions are conditioned on both image feature and previous hidden state at all time.

Base CNN + RNN$_{2u}$: An unfactored two-layer RNN using two stacks of RNNs. This model has a deeper structure at each time step. This variant also has access to image feature at every time step.

Base CNN + RNN$_{2f}$: A factored two-layer RNN that uses two stacks of RNNs. This variant only has access to the image features at the second layer RNN. In this way we force the first stack of RNN to focus on character-level language modeling and force the second stack of RNN to focus on combining language statistics and image feature.

As noted previously, all the RNN variants that we explored implicitly perform character-level language modeling and benefit from not being constrained to pre-defined N-gram sequences. For instance, Karpathy *et al*. [22] demonstrate that RNN-based methods consistently outperforms N-gram models at character-level text prediction where N is as large as 20. In the experiment section we will show the error analysis with and without the proposed RNNs.

2.4. Attention modeling

Attention-based mechanisms can allow the model to focus on the most important segments of incoming features, as well as potentially adding a degree of interpretability [3, 51]. There are generally two categories of attention-based image understanding: hard-attention and soft-attention. Hard-attention models learn to choose a series of discrete glimpse locations, and can be challenging to train since the loss gradients are typically intractable. In this work we choose a soft-attention model, which can be trained end-to-end with standard backpropagation.

We now describe our attention modeling function illustrated in Figure 3 as **Base CNN + RNN$_{\text{Atten}}$**. At every output step t, the attention function (denoted as a letter A in the figure) computes an energy vector $\boldsymbol{\tau}_t$ conditioned on the image feature $\boldsymbol{\mathcal{I}}$ and the output of the first stack RNN $\mathbf{s}_t$:

$$\boldsymbol{\tau}_t = f_{\text{attention}}(\boldsymbol{\mathcal{I}}, \mathbf{s}_t) = \tanh(\varphi(\boldsymbol{\mathcal{I}}) + \psi(\mathbf{s}_t)) \tag{5}$$

where φ and ψ can be multilayer perceptrons or simple weight matrices that project both $\boldsymbol{\mathcal{I}}$ and $\mathbf{s}_t$ to the same space. Then the context vector $\mathbf{c}_t$ is computed as weighted image feature based on the energy coefficients α_t at time step t:

$$\begin{aligned} \alpha_{td} &= \frac{\exp(\tau_{td})}{\sum_{d=1}^{D}\exp(\tau_{td})} \\ \mathbf{c}_t &= \boldsymbol{\alpha}_t \circ \boldsymbol{\mathcal{I}} \end{aligned} \tag{6}$$

where $\circ$ is the Hadamard product. This mechanism generates a set of positive weights α_{td} which can be understood as the relative importance to give to location d in fusing the image feature $\boldsymbol{\mathcal{I}}$, and the computed context vector $\mathbf{c}_t$ is then sent to the second stack of RNN for final output prediction.

3. Experiments

3.1. Datasets

We evaluate the proposed Recursive Recurrent Nets with Attention Modeling (R^2AM) framework on five standard benchmark datasets: ICDAR 2003, ICDAR 2013, Street View Text, IIIT5k and Synth90k.

ICDAR 2003 [30] contains 251 full scene images and 860 cropped images of the words. Even though the focus of this paper is unconstrained text recognition, we nonetheless provide constrained text recognition results for the ease of comparison. The per-image 50 word lexicons defined by Wang *et al*. [46] are referred to as IC03-50 and the lexicon of all test words (563 words) is referred as IC03-Full.

ICDAR 2013 [21] contains 1015 cropped word images from natural scene images and is referred as IC13.

Street View Text [46] contains 647 cropped word images from Google Street View. The per-image 50 word lexicons defined by Wang *et al*. [46] are referred to as SVT-50 and the lexicon of all test words (4282 words) is referred as SVT-Full.

IIIT5k [31] contains 3000 cropped word images downloaded from Google image search engine. Each image has a lexicon of 50 word (IIIT5k-50) and a lexicon of 1k word (IIIT5k-1k).

Synth90k [16] contains synthetically generated word images. The dataset contains around 7 million training images,

900k validation images, and 900k test images.

We follow the setting in Jaderberg *et al*. [17] to prepare training and test sets that our method is trained purely on the Synth90k training set and all parameters are selected via validation set. We do not use the validation data to retrain our model. We also follow the evaluation protocol in Wang *et al*. [46] that performs recognition on the words containing only alphanumeric characters (0-9 and A-Z) and at least three characters.

3.2. Implementation details

The network architecture for our Base CNN model is shown in Table A1. It has 8 convolutional layer with 64, 64, 128, 128, 256, 256, 512 and 512 channels, and each convolutional layer uses kernel with a 3×3 spatial extent. Convolutions are performed with stride 1, zero padding, and ReLU activation function. 2×2 max pooling follows the second, fourth, and sixth convolutional layers. The two fully connected layers have 4096 units. The input is a resized 32×100 gray scale image.

We now provide details for the network structures of the proposed untied recursive CNNs in Table A1. Notice that each of the even number convolutional layer (conv2, conv4, conv6 or conv8) use its own shared weight matrix that has exactly the same input and output dimensionality, and so projects feature maps to the same space multiple times within one recursive convolutional layer under the same parametric capacity as Base CNN model.

For the character-level language modeling, we use RNNs with 1024 hidden units equipped with hyperbolic tangent activation function. Our overall system pipeline is shown in Figure 1.

We apply backpropagation through time (BPTT) algorithm to train the models with 256 batch size SGD and 0.5 dropout rate. Initial learning rate is 0.002 and decreased by a factor of 5 as validation errors stop decreasing for 2 epochs. All variants use the same scheme with 30 total epochs determined based on the validation set. We apply gradient clipping at the magnitude of 10, and find it with in place weight decay did not add extra performance gains. All initial weights are sampled from Gaussian distribution with 0.01 standard deviation. We implemented the system in the open source deep learning framework Caffe [20]. The average inference time per image is 2.2 ms on single Nvidia Titan X GPU for the overall system framework.

3.3. Ablation study

In this section we empirically investigate the contributions made by three key components in the proposed method, namely: recursive CNNs for image encoding, RNNs for character-level language modeling, and attention-based mechanism for better image feature usage.

In an effort to decouple the performance improvement that is due to architectural variations from that which might simply come from having more parameters, we first gradually increase the depth of the baseline CHAR model in [17] from 5 conv layers until we reach the performance plateau at 8 conv layers shown as Base CNN in Table A1. Having observed this plateau, we did not explore even deeper networks such as (16 or 19 conv layer) VGGNet [40]. The bar chart in Table A1 shows the corresponding performance for networks with different depths on Synth90k dataset.

Method	Synth90k	SVT	ICDAR13
CHAR [17]	87.3	68.0	79.5
Base CNN	91.9	75.1	85.7
Recurrent CNN (2 iter)	92.6	75.8	86.1
Recurrent CNN (3 iter)	93.5	76.9	87.4
Recursive CNN (2 iter)	93.3	77.1	87.3
Recursive CNN (3 iter)	**94.2**	**78.9**	**88.5**

Table 1: Unconstrained (lexicon-free) text recognition accuracies on recent benchmarks. See Figure 2 for diagrams of these architectures. The results indicate that "iteration" is important to both recurrent and recursive CNNs. Recursive CNNs outperform recurrent CNNs on all three datasets.

Method	Synth90k	SVT	ICDAR13
CHAR [17]	87.3	68.0	79.5
Base CNN	91.9	75.1	85.7
Base CNN + RNN_{1c}	93.4	76.2	86.4
Base CNN + RNN_{1u}	93.5	76.9	87.2
Base CNN + RNN_{2u}	93.7	77.9	87.6
Base CNN + RNN_{2f}	94.0	78.8	88.0
Base CNN + $\text{RNN}_{\text{Atten}}$	**94.3**	**79.1**	**88.9**

Table 2: Unconstrained (lexicon-free) text recognition accuracies on recent benchmarks. See Figure 3 for diagrams of these architectures. The results indicate that the simply uses image captioning style method can already boost performance, while factored RNNs with attention modeling overall achieves the best results.

Method	Synth90k	SVT	ICDAR13
CHAR [17]	87.3	68.0	79.5
JOINT [17]	91.0	71.7	81.8
R^2AM (ours)	**95.3**	**80.7**	**90.0**

Table 3: Unconstrained (lexicon-free) text recognition accuracies on recent benchmarks. Our combined model R^2AM (Recursive CNN + $\text{RNN}_{\text{Atten}}$) significantly outperforms previous state-of-the-art methods in [17].

3.3.1 Recursive and recurrent convolutional layers

Table 1 shows the effectiveness of the proposed untied recursive and recurrent CNNs over Base CNN model on unconstrained text recognition tasks. We observed that more iterations in both proposed methods led to higher accuracies on all datatsets evaluated, and the improvement essentially came from the same parametric capacity since the weights

Method	SVT-50	SVT	IIIT5k-50	IIIT5k-1k	IIIT5k	IC03-50	IC03-Full	IC03	IC13
Baseline ABBYY [46]	35.0	-	24.3	-	-	56.0	55.0	-	-
Wang *et al*. [46]	57.0	-	-	-	-	76.0	62.0	-	-
Mishra *et al*. [31]	73.2	-	-	-	-	81.8	67.8	-	-
Novikova *et al*. [34]	72.9	-	64.1	57.5	-	82.8	-	-	-
Wang *et al*. [48]	70.0	-	-	-	-	90.0	84.0	-	-
Bissacco *et al*. [4]	90.4	78.0	-	-	-	-	-	-	87.6
Goel *et al*. [8]	77.3	-	-	-	-	89.7	-	-	-
Alsharif and Pineau [2]	74.3	-	-	-	-	93.1	88.6	-	-
Almazán *et al*. [1]	89.2	-	91.2	82.1	-	-	-	-	-
Lee *et al*. [26]	80.0	-	-	-	-	88.0	76.0	-	-
Yao *et al*. [53]	75.9	-	80.2	69.3	-	88.5	80.3	-	-
Rodriguez-Serrano *et al*. [36]	70.0	-	76.1	57.4	-	-	-	-	-
Jaderberg *et al*. [19]	86.1	-	-	-	-	96.2	91.5	-	-
Su and Lu *et al*. [43]	83.0	-	-	-	-	92.0	82.0	-	-
Gordo [10]	90.7	-	93.3	86.6	-	-	-	-	-
*DICT Jaderberg *et al*. [18]	95.4	80.7	97.1	92.7	-	98.7	98.6	93.1	90.8
Jaderberg *et al*. [17]	93.2	71.7	95.5	89.6	-	97.8	**97.0**	**89.6**	81.8
R^2AM (ours)	**96.3**	**80.7**	**96.8**	**94.4**	**78.4**	**97.9**	**97.0**	88.7	**90.0**

Table 4: Scene text recognition accuracies (%). "50", "1k" and "Full" denote the lexicon size used for constrained text recognition defined in [46]. The last two rows list methods that are capable of performing unconstrained text recognition (lexicon-free). Our proposed R^2AM method significantly outperforms previous best unconstrained text recognition method [17] in most of the cases (bold numbers), especially on the recent released datasets such as SVT, IIIT5k, IC13. *DICT [18] is not lexicon-free due to incorporating ground-truth labels during training.

of convolutional layer are shared as shown in Figure 2 and Table A1. The lateral interactions between these shared weights allow for broader receptive fields and competition between representational units in the same "layer". In addition, we consistently found that recursive version outperforms recurrent version. It might because the recursive convolutional layer can prevent error signals directly backpropagate back via recurrent connections in the recurrent convolutional layer, which is especially true for convolutional operation since the input signal remains unchanged. Therefore, we choose the recursive CNNs for our final system architecture. The architectural variants we study differ from Residual Networks (ResN) [12] in that our recurrent model receives the same bottom-up input at each stage of recurrency, whereas in ResN's the identity connections skip layers.

3.3.2 Character-level language modeling

In Table 2, we report unconstrained text recognition results for each of the architectural variants of RNNs in Figure 3. We observed an immediate performance boost by using any kind of the proposed RNN variants atop the Base CNN network which has already hit its performance plateau. RNN_{1c} serves as a good sanity check module because the image features from the Base CNN are only fed to the RNN at the first time step, and then RNN_{1c} is able to predict the first and the following character correctly based on the previously predicted character and the hidden state information.

The comparison of RNN_{1u} and RNN_{1c} results indicates that feeding image feature from Base CNN to a RNN at every time step can further improve the performance, as RNN_{1u} has access not only to the previously predicted character and hidden state information, but also the raw image feature during inference. (This architecture presumably also allows the RNN to expend more capacity on sequence modeling, since it no longer needs to retain the image feature information in its hidden state.) In addition, based on the observation that RNN_{2f} outperforms RNN_{2u}, we note that factorization seems to be a more effective architecture than the unfactored models. This might because that the encoded image features can only be accessed by the second stack RNN, and this therefore allows/forces the first stack RNN to focus on modeling character-level statistics (and/or attentional processes).

The proposed $\text{RNN}_{\text{Atten}}$ uses an attention-based mechanism that learns a set of weight matrices to rescale image features and perform soft attention modeling before feeding the image feature to the top-level RNN. This architecture was observed to give the best performance across all five of the RNN variants explored.

Thus, our final network architecture contains the aforementioned recursive CNNs for image feature extraction and the $\text{RNN}_{\text{Atten}}$ for character-level language modeling with attention as shown in Figure 1.

At this point it is also worth mentioning that we have explored backward RNNs and bidirectional RNNs for character-level language modeling as well, however neither of these extensions delivered further improvements. This is in contrast to the observations in Graves *et al*. [11]. Perhaps this is because our work focuses on predicting characters in scene images that contain around 8 characters on average, while method in [11] focuses on longer sequences

Figure 4: Lexicon-free scene text recognition results by the proposed Recursive Recurrent Nets with Attention Modeling (R^2AM) framework on examples from SVT, ICDAR and IIIT5k datasets. R^2AM is able to recognize words with low contrast, significant transform and clutter background. It is also able to recover missing/occluded characters (*e.g.* PARK, BURBANK, SAN and STAR) by the implicitly learned language model.

of handwritten scripts. For this reason, we did not explore LSTM memory cells, which are often used to help improve the ability of RNN's to retain information over longer time-scales (usually at a cost in terms of model complexity and computational run-time).

3.3.3 Constrained and unconstrained text recognition

Recognizing text in the wild without a lexicon or dictionary is a challenging task. Bissacco first reported lexicon-free (but heavily dictionary weighted) results in [4], and Jaderberg *et al*. [17] recently presented a notable advancement in lexicon/dictionary-unconstrained scene text recognition. Table 3 compares the accuracy of our proposed method to the previous best results in [17] on fully unconstrained text recognition task. As can be seen, our method significantly outperforms the JOINT model in [17] by margin of 9% on SVT and 8.2% on ICDAR 2013.

Even though our proposed method aims at the unconstrained scenario, we also compare our results to the constrained setting in which the output is selected with the smallest edit distance between the predicted character sequence and words in the pre-defined lexicon. Table 4 shows these comparisons. Our method obtained the new best results for unconstrained recognition on several benchmarks, especially the recent released ICDAR 2013 (IC13) dataset. We also report unconstrained text recognition result on IIIT5k that has not been recorded in previous literature. We are also competitive with the very best results in the constrained setting as well. (Notice that the DICT model of [18] is trained on a specific dictionary that contains ground-truth words for the test set; it is not able to handle previously unseen word strings.)

Figure 4 demonstrates the lexicon-free scene text recognition results by the proposed R^2AM framework on examples from SVT, ICDAR, and IIIT5k datasets. The R^2AM method is able to recognize words with low contrast, significant transform and clutter background. It is also able to recover missing/occluded characters by the implicitly learned language model. We further demonstrate our language model by performing text prediction on non-text images as shown in Figure A1, seeing that our method can exploit the underlying character-level statistics and produce word-like strings even though the images have no alphanumeric characters.

4. Conclusion and future directions

We have presented a new lexicon-free photo OCR framework that incorporates recursive CNNs for image encoding, RNNs for language modeling, and attention-based mechanism for better image feature usage. Extensive analysis shows the effectiveness of each of the proposed component, generalizability to both constrained and unconstrained scenarios with the same methodology, and the practical ability of the proposed method to recognize real-world scene text with state-of-the-art results.

In the future we will explore recursive fully convolutional networks for better connecting extracted image features and corresponding location on input image, and to visualize attention coefficients on the input domain. Also, we will adopt gated units to allow incoming signal to alter the state of attention-based mechanism [27] and inject deep supervision [28].

A further future direction is to extend this work into text-detection [52, 44] in addition to reading, leading to an end-to-end full-image-to-text reading pipeline.

Acknowledgements

The authors thank Jack Culpepper and Cyprien Noel for valuable discussion, and also thank Flickr Vision and Machine Learning Team for supporting the project.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *TPAMI*, 2014. 7

[2] O. Alsharif and J. Pineau. End-to-end text recognition with hybrid hmm maxout models. In *ICLR*, 2014. 7

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 5

[4] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *ICCV*, 2013. 2, 7, 8

[5] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *CVPR*, 2004. 1

[6] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 4

[7] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. In *ICLR Workshop*, 2014. 3

[8] V. Goel, A. Mishra, K. Alahari, and C. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *ICDAR*, 2013. 7

[9] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *ICLR*, 2014. 2

[10] A. Gordo. Supervised mid-level features for word image representation. In *CVPR*. 2015. 1, 7

[11] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *TPAMI*, 2009. 7

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 7

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 2006. 1

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997. 4

[15] O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *NIPS*, 2014. 3

[16] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NIPS*, 2014. 1, 4, 5

[17] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. In *ICLR*, 2015. 1, 2, 4, 6, 7, 8, 10

[18] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 2015. 2, 7, 8

[19] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV*. 2014. 1, 7

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 6

[21] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. Gomez i Bigorda, S. Robles Mestre, J. Mas, D. Fernandez Mota, J. Almazan Almazan, and L.-P. de las Heras. Icdar 2013 robust reading competition. In *ICDAR*, 2013. 5

[22] A. Karpathy, J. Johnson, and F.-F. Li. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015. 5

[23] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, 2014. 4

[24] K. Kita and T. Wakahara. Binarization of color characters in scene images using k-means clustering and support vector machines. In *ICPR*, 2010. 1

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 1

[26] C.-Y. Lee, A. Bhardwaj, W. Di, V. Jagadeesh, and R. Piramuthu. Region-based discriminative feature pooling for scene text recognition. In *CVPR*, 2014. 1, 2, 7

[27] C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016. 8

[28] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015. 8

[29] M. Liang and X. Hu. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015. 3

[30] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *ICDAR*, 2003. 5

[31] A. Mishra, K. Alahari, and C. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012. 1, 4, 5, 7

[32] A. Mishra, K. Alahari, and C. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, 2012. 1, 2, 4

[33] L. Neumann and J. Matas. Real-time scene text localization and recognition. In *CVPR*, 2012. 1

[34] T. Novikova, O. Barinova, P. Kohli, and V. Lempitsky. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV*. 2012. 4, 7

[35] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. *ICML*, 2014. 3

[36] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin. Label embedding: A frugal baseline for text recognition. *IJCV*, 2014. 7

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985. 4

[38] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015. 4

[39] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang. Scene text recognition using part-based tree-structured character detection. In *CVPR*, 2013. 1, 2

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 6

[41] D. L. Smith, J. Field, and E. Learned-Miller. Enforcing similarity constraints with integer programming for better scene text recognition. In *CVPR*, 2011. 1

[42] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, 2012. 3

[43] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *ACCV*. 2015. 7

[44] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016. 8

[45] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 5

[46] K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *ICCV*, 2011. 1, 2, 5, 6, 7

[47] K. Wang and S. Belongie. Word spotting in the wild. In *ECCV*, 2010. 1

[48] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *ICPR*, 2012. 1, 2, 7

[49] J. J. Weinman, Z. Butler, D. Knoll, and J. Feild. Toward integrated scene text reading. *TPAMI*, 2014. 1

[50] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1988. 4

[51] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 5

[52] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, 2012. 8

[53] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *CVPR*, 2014. 1, 2, 7