SEQUENCE TO SEQUENCE LEARNING FOR OPTICAL CHARACTER RECOGNITION

Devendra Kumar Sahu & Mohak Sukhwani
International Institute of Information Technology
Hyderabad, Telangana 500032, India
dksahuji@gmail.com
mohak.sukhwani@research.iiit.ac.in

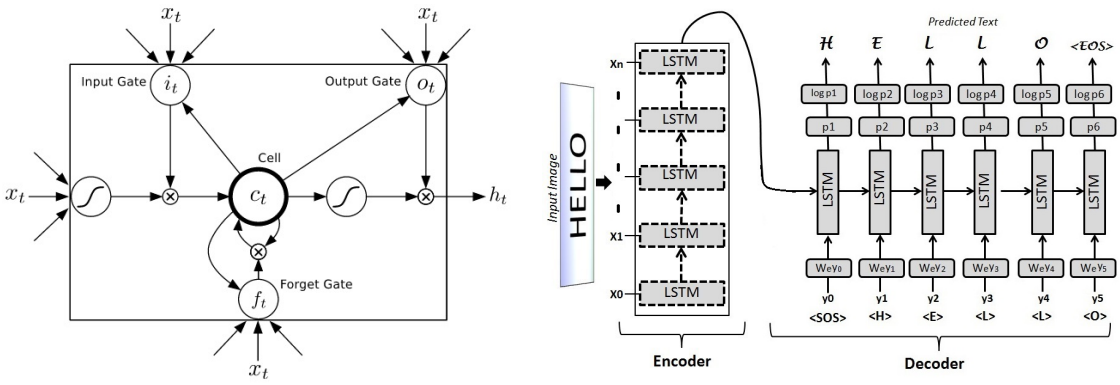
ABSTRACT

We propose an end-to-end recurrent encoder-decoder based sequence learning approach for printed text Optical Character Recognition (OCR). In contrast to present day existing state-of-art OCR solution [Graves et al. (2006)] which uses CTC output layer, our approach makes minimalistic assumptions on the structure and length of the sequence. We use a two step encoder-decoder approach – (a) A recurrent encoder reads a variable length printed text word image and encodes it to a fixed dimensional embedding. (b) This fixed dimensional embedding is subsequently comprehended by decoder structure which converts it into a variable length text output. Our architecture gives competitive performance relative to Connectionist Temporal Classification (CTC) [Graves et al. (2006)] output layer while being executed in more natural settings. The learnt deep word image embedding from encoder can be used for printed text based retrieval systems. The expressive *fixed* dimensional embedding for any variable length input expedites the task of retrieval and makes it more *efficient* which is not possible with other recurrent neural network architectures. We empirically investigate the expressiveness and the learnability of long short term memory (LSTMs) in the sequence to sequence learning regime by training our network for prediction tasks in segmentation free printed text OCRs. The utility of the proposed architecture for printed text is demonstrated by quantitative and qualitative evaluation of two tasks – word prediction and retrieval.

1 INTRODUCTION

Deep Neural Nets (DNNs) have become present day de-facto standard for any modern machine learning task. The flexibility and power of such structures have made them outperform other methods in solving some really complex problems of speech [Hinton et al. (2012)] and object [Krizhevsky et al. (2012)] recognition. We exploit the power of such structures in an OCR based application for word prediction and retrieval with a single model. Optical character recognition (OCR) is the task of converting images of typed, handwritten or printed text into machine-encoded text. It is a method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line and used in machine processes such as machine translation, text-to-speech and text mining.

From character recognition to word prediction, OCRs in recent years have gained much awaited traction in mainstream applications. With its usage spanning across handwriting recognition, print text identification, language identification etc. OCRs have humongous untapped potential. In our present work we show an end-to-end, deep neural net, based architecture for word prediction and retrieval. We conceptualize the problem as that of a sequence to sequence learning and use RNN based architecture to first encode input to



(a) LSTM block

(b) Encoder Decoder Framerwork for OCR

Figure 1: Figure 1a is the LSTM block comprised of input, output and forget gates. Figure 1b showcases the proposed recurrent encoder decoder framework for Optical Character Recognition. The Encoder LSTM section reads the input image and converts it to a fixed-dimensional vector representation. Decoder LSTM in turn generates the text output corresponding to fixed-dimensional vector representation.

a fixed dimension feature and later decode it to variable length output. Recurrent Neural Networks (RNN) architecture has an innate ability to learn data with sequential or temporal structure. This makes them suitable for our application. Encoder LSTM network reads the input sequence one step at a time and converts it to an expressive fixed-dimensional vector representation. Decoder LSTM network in turn converts this fixed-dimensional vector (Figure 1b) to the text output.

Encoder-Decoder framework has been applied to many applications recently. [Sutskever et al. (2011)] used recurrent encoder-decoder for character-level language modelling task where they predict the next character given the past predictions. It has also been used for language translation [Sutskever et al. (2014)] where a complete sentence is given as input in one language and the decoder predicts a complete sentence in another language. Vinyals et al. [Vinyals et al. (2014)] presented a model based on a deep recurrent architecture to generate natural sentences describing an image. They used a convolutional neural network as encoder and a recurrent decoder to describe images in natural language. Zaremba et al. [Zaremba & Sutskever (2014)] used sequence to sequence learning for evaluating short computer programs, a domain that have been seen as too complex in past. Vinyals et al. [Vinyals & Le (2015)] proposed neural conversational networks based of sequence to sequence learning framework which converses by predicting the next sentence given the previous sentence(s) in a conversation. In the same spirit as Vinyals et al. (2014); Sutskever et al. (2014), we formulate the OCR problem as a sequence to sequence mapping problem to convert an input (text) image to its corresponding text.

In this paper, we investigate the expressiveness and learnability of LSTMs in sequence to sequence learning regime for printed text OCR. We demonstrate that sequence to sequence learning is suitable for word prediction task in a segmentation free setting. We even show the expressiveness of the learnt deep word image embeddings (from Encoder network of prediction) on image retrieval task. In (majority of) cases where standard LSTM models do not convert a variable length input to a fixed dimensional output, we are required to use Dynamic Time Warping (DTW) for retrieval which tends to be computationally expensive and slow. Converting variable length samples to fixed dimensional representation gives us access to fast and efficient methods for retrieval in fixed dimensional regime - approximate nearest neighbour.

2 SEQUENCE LEARNING

A recurrent neural network (RNN) is a neural network with cyclic connections between its units. These cycles create a concept of 'internal memory' in network and thus differentiate RNNs from other feed forward networks. The internal memory of RNN can be used to process arbitrary sequences of inputs – given a variable length input sequence X we can generate corresponding variable length output sequence Y. This is done by sequentially reading each time-step x_t of input sequence X and updating its internal hidden representations h_t. More sophisticated recurrent activation functions like LSTM [Hochreiter & Schmidhuber (1997)] and GRU [Cho et al. (2014); Chung et al. (2014)] have become more common in recent days. They perform better when compared to other vanilla RNN implementations.

Long Short-Term Memory [Hochreiter & Schmidhuber (1997)] is a RNN architecture that elegantly addresses the vanishing gradients problem using 'memory units'. These linear units have a pair of auxiliary 'gating units' that control the flow of information to and from the unit. Equations 1-5 describe LSTM blocks.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + w_{ci} \odot c_{t-1} + b_i) \tag{1}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + w_{cf} \odot c_{t-1} + b_f) \tag{2}$$
$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + w_{co} \odot c_{t-1} + b_o) \tag{4}$$
$$h_t = o_t tanh(c_t) \tag{5}$$

Here, $W_{xi}, W_{hi}, w_{ci}, b_i$ are input gate parameters. $W_{xo}, W_{ho}, w_{co}, b_o$ are output gate parameters. $W_{xf}, W_{hf}, w_{cf}, b_f$ are forget gate parameters. W_{xc}, W_{hc}, b_c are parameters associated with input which directly modify the memory cells. The symbol $\odot$ denotes element-wise multiplication. The gating units are implemented by multiplication, so it is natural to restrict their domain to $[0, 1]^N$, which corresponds to the sigmoid non-linearity. The other units do not have this restriction, so the tanh non-linearity is more appropriate. We use collection of such units (Figure 1a) to describe an encoder-decoder framework for the OCR task. We formulate the task of OCR prediction as a mapping problem between structured input (image) and structured output (text).

Let, $\{I_i, Y_i\}_{i=1}^N$ define our dataset with I_i being image and Y_i be the corresponding text label. Image I_i lies in $\mathbb{R}^{H \times T_i}$, where H is word image height (common for all images) and T_i is the width of i^{th} word image. We represent both image and label as a sequence – Image $I_i = \{x_1, x_2, ..., x_{T_i}\}$ is sequence of T_i vectors lying in $\{0, 1\}^H$ (x_i is a i_{th} pixel column) and $Y_i = \{y_1, y_2, ..., y_{M_i}\}$ is corresponding label which is a sequence of M_i unicode characters. We learn a mapping from image to text in two steps – (a) $f_{encoder} : I_i \rightarrow z_i$, maps an image I_i to a latent fixed dimensional representation z_i (b) $f_{decoder} : z_i \rightarrow Y_i$ maps it to the output text sequence Y_i. Unlike CTC layer based sequence prediction [Graves et al. (2006)] we don't have any constraint on length of sequences I_i and Y_i. Equations 6-7 formally describe the idea. The choice of $f_{encoder}$ and $f_{decoder}$ depends on type of input and output respectively. Both input and output correspond to a sequence in our case, hence we use recurrent encoder and recurrent decoder formulation.

$$z_i = f_{encoder}(I_i) \tag{6}$$
$$P(Y_i|I_i) = f_{decoder}(z_i) \tag{7}$$

2.1 ENCODER: LSTM BASED WORD IMAGE READER

To describe the formulation we use vanilla RNNs with L hidden layer and no output layer. The encoder reads $I_i = \{x_1, x_2, ..., x_{T_i}\}$ one step at a time from x_1 to x_{T_i}. Hidden state h_t^n is updated using equations 8-9 using current input x_t and previous hidden state $\{h_{t-1}^n\}_{n=1}^L$ where L is the number of hidden layers in RNN.

$$h_t^1 = relu(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_h^1) \tag{8}$$
$$h_t^n = relu(W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_h^n) \tag{9}$$

where $W_{ih^1}, W_{h^1h^1}, b_h^1, W_{ih^n}, W_{h^{n-1}h^n}, W_{h^nh^n}, b_h^n$ are parameters to be learned.

To obtain our fixed dimensional latent representation z_i we use the final hidden states $\{h_{T_i}^n\}_{n=1}^L$.

$$z_i = \{h_{T_i}^n\}_{n=1}^L \tag{10}$$

It should be noted that we have used LSTM networks instead of vanilla RNNs and no output layer for encoder is needed. The hidden states of last step T_i are used as initial state of decoder network.

2.2 DECODER: LSTM BASED WORD PREDICTOR

Similar to encoder, we describe the idea using vanilla RNNs with L hidden layers and softmax output layer. The goal of word predictor is to estimate the conditional probability $p(Y_i|I_i)$ as shown in equation 11-12, where I_i is image input sequence and Y_i is output sequence.

$$p(Y_i|I_i) = p(Y_i = \{y_1, \ldots, y_{T'}\}|I_i = \{x_1, \ldots, x_{T_i}\}) \tag{11}$$

$$= \prod_{t=1}^{T'} p(y_t|I_i, y_1, \ldots, y_{t-1}) \tag{12}$$

The updates for single step for RNN is described in equations 13-16. The hidden state h_t^n is updated using equations 13 - 14 using current input x_t and previous hidden state $\{h_{t-1}^n\}_{n=1}^L$, where L is number of hidden layers in RNN. The hidden activations, h_{t-1}^L are used to predict the output at step t using equations 15-16. x_t is the embedding of the most probable state in previous step $t-1$ shown in equation 18.

$$h_t^1 = relu(W_{ih^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_h^1) \tag{13}$$

$$h_t^n = relu(W_{h^{n-1}h^n}h_t^{n-1} + W_{h^nh^n}h_{t-1}^n + b_h^n) \tag{14}$$

$$o_t = W_{h^Lo}h_t^L + b_o \tag{15}$$

$$p_t = sofmax(o_t) \tag{16}$$

where $W_{ih^1}, W_{h^1h^1}, b_h^1, W_{ih^n}, W_{h^{n-1}h^n}, W_{h^nh^n}, b_h^n, W_{h^Lo}, b_o$ are parameters to be learned.

Decoding begins at $t = 0$ with $\langle SOS \rangle$ marker (Start Of Sequence). The state $t = -1$ is initialized with the final state z_i of encoder as shown in equation 17. The 1^{st} character is predicted using the embedding for $\tilde{y}_0 = \langle SOS \rangle$ as input $x_0 = W_e\tilde{y}_0$ and output $p_1(y_1|I_i, y_0)$ where W_e is the embedding matrix for characters. As shown in equation 18, every t_{th} character is predicted using the embedding for $\tilde{y}_t = \arg\max p_t(y_t|I_i, y_{<t})$ as input and output $p_t(y_t|I_i, y_{<t})$. This is iterated till $t = T'$ where $y_{T'} = \langle EOS \rangle$ (End Of Sequence). T' is not known in priori, $\langle EOS \rangle$ marker instantiates the value of T'.

$$h_{-1} = f_{encoder}(I) \tag{17}$$

$$x_t = W_e\tilde{Y}_t \tag{18}$$

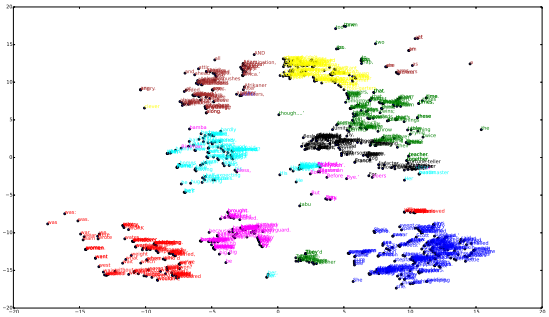
$$p_{t+1} = f_{decoder}(x_t) \tag{19}$$

W_e in equation 18 is the embedding matrix for characters. It should be (again) noted that we use LSTM networks (equations 1 - 5) instead of vanilla RNNs (equations 13, 14).

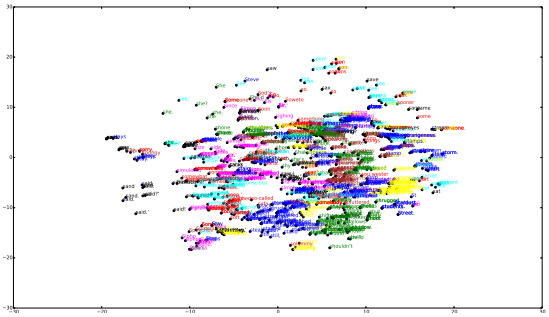
2.3 TRAINING

The model described in section 2.1 and 2.2 is trained to predict characters of the input word (image) sequence. The input at time t of decoder is an embedding of the output of time $t-1$. The loss L for a sample (I, Y) is described by equation 20.

$$L(I, Y) = -\log p(Y|I; \theta) \tag{20}$$



(a) TSNE plots for feature representation of unique words starting with top eight most frequent alphabets(S, T, W, H, B, C, F, A including lowercase. (Color based on first alphabet of words)



(b) TSNE plots for feature representation of unique words starting with S and top eight *second* alphabets (t, h, o, e, i, u, a, p) in word with respect to population in S. (Color based on second alphabet of words starting with S.)

Figure 2: TSNE plots characterizing the quality of feature representations computed by encoder. Each word has a unique high dimensional feature representation which is then embedded in a two dimensional space (using TSNE) and is visualized using scatter plots. Similar words are grouped together (shown with various colours) and dissimilar words tend to get far away. As shown in the figure, (a) words starting with same alphabets belong to same clusters and rest are in other clusters (b) words beginning with 'S' and having same second alphabet belong to same clusters and rest are in other clusters. (Readers are requested to magnify the graphs to look into the intricate details of clusters)

Here, $\log p(Y|I;\theta) = \sum_{t=0}^{M} \log p(y_t|I, y_0, \ldots, y_{t-1};\theta)$ is the log probability of correct symbol at each step. We search for the parameters θ^* which minimize the expected loss over true distribution $P(I,Y)$ given in equation 21. This distribution is unknown and can be approximated with empirical distribution $\tilde{P}(I,Y)$ given in equation 22-23.

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{P(I,Y)} L(I,Y;\theta) \tag{21}$$

$$\approx \arg\min_{\theta} \mathbb{E}_{\tilde{P}(I,Y)} L(I,Y;\theta) \tag{22}$$

$$= \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(I_i, Y_i;\theta) \tag{23}$$

3 Implementation Details

Keeping the aspect ratio of input images intact we resize them to height of 30 pixels. The resized binary images are then used as an input to the two layer LSTM encoder-decoder architecture. We use embedding size of 25 for all our experiments. The dimensionality of output layer in decoder is equal to number of unique symbols in the dataset. RMS prop [Tieleman & Hinton (2012)] with step size of 0.0001 and momentum of 0.99 is used to optimize the loss. All relevant parameters are verified and set using a validation set. We use Python's numpy library to implement LSTM based architecture. The network is built using computational graphs.

Features	Dim	mAP-100	mAP-5000
BOW	400	0.5503	0.33
BOW	2000	0.6321	-
AUGMENTED PROFILES[3]	247	0.7371	0.6189
LSTM-ENCODER	400	0.7402 $(h1-h2)$ 0.8521 $(c1-c2)$	0.8521
OCR - TESSERACT	-	0.6594	0.7095
OCR - ABBYY	-	0.8583	0.872

Table 1: mAP computed for various methods: mAP-n stands for mean average precision computed over top n retrievals. hi is hidden representation of layer-i at last timestep of input sequence. ci is memory for layer-i at last timestep of input sequence. A-B is the concatenation of representation A and B. For example, $h1-h2$ represents concatenation of both $h1$ and $h2$.

Model	Label error(%)
ABBYY[2]	1.84
TESSERACT[1]	35.80
TESSERACT[2]	16.95
RNN ENCODER-DECODER	35.57
LSTM-CTC [Graves et al. (2006)]	0.84
LSTM ENCODER-DECODER	0.84

Feature	mAP-100
h1-h2	0.7239
c1-c2	0.8548
h1-h2-c1-c2 L1	0.8078
h1-h2-c1-c2 L2	0.7834
h1-h2-c1-c2	0.8545

Table 2: **Left:** Label Error Rate comparison of RNN-CTC and Recurrent encoder-decoder.**Right:** Effect of different concatenation and normalization on features from LSTM-Encoder. L1 and L2 represent normalization scheme.

4 EXPERIMENTS

We demonstrate the utility of the proposed recurrent encoder-decoder framework by two related but independent tasks. Independent baselines are set for both prediction and retrieval experiments.

Prediction: We use 295K annotated English word images from seven books for our experiments. We perform three way data split for all our experiments – 60% training, 20% validation and remaining 20% for testing. Results are reported by computing 'label error rate'. Label error rate is defined as ratio of sum of insertions, deletions and substitutions relative to length of ground truth over dataset. We compare the results of our pipeline with state-of-art LSTM-CTC [Graves et al. (2006)], an open-source OCR TESSERACT [tes] and a commercial OCR ABBYY [abb].

Retrieval: We use 108K annotated word images from book titled 'Adventures of Sherlock Holmes' for retrieval experiments. In all 43K word images are used for querying the retrieval system. We compare retrieval results with SIFT [Lowe (2004)] based bag of words representation, augmented profiles [Kumar et al. (2007)] and commercial [OCR ABBYY].

[1]Original images are used as input.

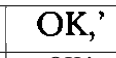
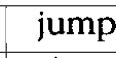
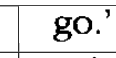
[2]Images are padded along boundary pixels for better results.

[3]Augmented Profiles [Kumar et al. (2007)]

Table 3: Qualitative results for retrieval: Comparison of retrieval scheme using simple Bag of Words (BoW) and proposed Deep Word Image Embeddings (DWIE). We use text labels (to save space and more clarity in presentation) of both query and retrieved images to illustrate difference in performance of two approaches. The proposed approach is far more robust to text variations in images and captures much more intricate details about the images.

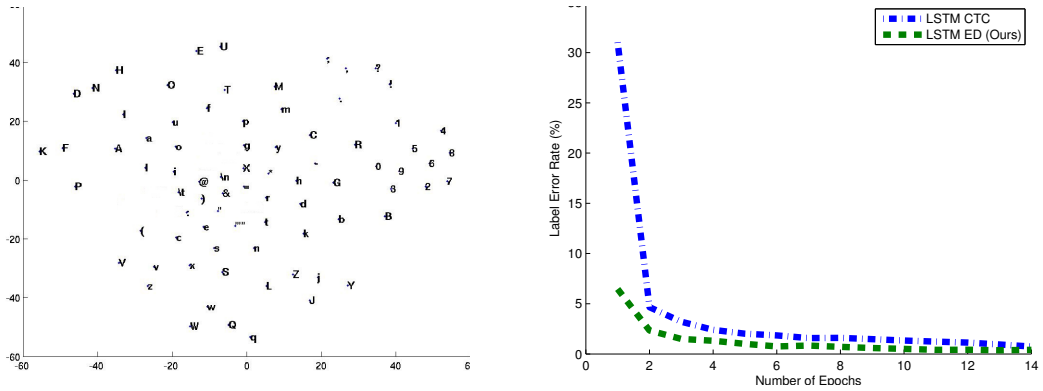
	DWIE	**BoW**	**DWIE**	**BoW**	**DWIE**	**BoW**	**DWIE**	**BoW**
Query (→) / Retrival (↓)	A.	A.	following	following	returned	returned	For	For
R1	A.	”A	following	long	returned	turned	For	For
R2	A.	A	following	long	returned	read	For	For
R3	A.	A	following	long	returned	refused	For	For
R4	A.	At	following	following	returned	retorted	For	For
R5	A.	Ah	following	long	returned	lifted	For	For
R6	A	A	following	flowing	returned	ceased	For	For
R7	A	A	following	long	returned	rolled	For	For
R8	A	A	folding	long	returned	and	For	For
R9	A	A	follow	long	returned	carried	For	for
R10	A	A	followed,	following	returned	red	For	for
R11	A	A	foolishly	long	returned	Head	For	for
R12	A	A	fellow,	along	returned	raised	For	for
R13	A	A	foolscap	long	returned	caused	For	for
R14	A	A	fellow,	following	returned	turned	For	for
R15	A	A	foliage.	long	returned	turned	For	for
R16	A	A	fellow,	long	returned.	road	For	for
R17	A	A	fellow,	long	retired	and	For	for
R18	A	A	fellow,	long	retorted	returned	Fer-	For
R19	A	As	falling	long	return.”	God	Fer-	for
R20	A	Af-	follow,”	closing	return	acted	Fer-	For
# relevant matches in corpus	5	5	7	7	15	15	17	17

Table 4: Qualitative results for prediction: We illustrate some of the success and failure cases of our word prediction output. The figure highlights the cases where both commercial and open-source OCRs fail

Query Image(→)				
True Label	mom	OK,’	jump	go.’
Tessaract [2]	IDOITI	ox,’	iump	80.
Abbyy [2]	UJOUJ	ok;	duinl	g-’
LSTM-ED	mom	OK,’	jump	go.’

4.1 Results and Discussion

Table 2 exhibits prediction baseline. We observe that LSTM Encoder-Decoder outperforms vanilla RNN Encoder-Decoder by significant margin. It even scores better when compared to LSTM with CTC output layer and ABBYY. When compared to CTC layer based LSTM networks, our network requires more memory



(a) Learnt embedding of characters using tSNE.

(b) Convergence of LSTM-CTC [4] and LSTM-ED.

Figure 3: Plot showing character embedding and convergence of LSTM-CTC [4] and LSTM-ED.

space. The strength of our network is fixed length representation for variable length input which enables us to perform better and faster retrieval.

Table 1 depicts retrieval baseline. Features from LSTM encoder (referred as deep word image embedding (DWIE)) are used for comparisons with other state-of-art results. We observe that DWIE features significantly outperform SIFT [Lowe (2004)] based bag of words (BOW) and augmented profiles [Kumar et al. (2007)]. When compared to ABBYY, DWIE features perform a notch better for top 5000 retrieval but perform similar for top 100 retrieval. The memory states at last position of each sequence are used as DWIE features. Various normalization (L1 and L2) and augmentations with hidden states were tried out as shown in Table 2.

Table 3 demonstrates the qualitative performance of retrieval system using both deep word image embedding (DWIE) and bag of words (BOW) models. The table illustrates top 20 retrievals using both the methods. We observe the proposed embeddings to be better than naive (BOW) in such settings. In majority of the cases we find all relevant(exact) matches at top in case of deep embeddings, which is not the case with (BOW) model. DWIE seems highly sensitive to small images components like '.' (for query 'A.') which is not the case with BOW model. Simple BOW fails to recover any relevant samples for query 'A.' in top 20 retrievals.

Figure 2 shows T-SNE [van der Maaten & Hinton (2008)] plots of word image encodings. We show two levels of visualization along with groupings in context of word image representation. It's clear from the figure 2a that representation is dominated by first character of the word in word image. Sequence of correct encodings play a major role in full word prediction – a wrong letter prediction in early stages would result in overall invalid word prediction.

Figure 3a is a plot of learnt embeddings which shows relative similarities of characters. The similarities are both due to structure and language of characters – (i) all the numbers (0-9) are clustered together (ii) punctuations are clustered at top right of graph (iii) upper case and lower case characters tend to cluster together, viz. (m,M), (v,V), (a,A) etc. As embeddings are learnt jointly while minimizing cost for correct predictions, they tend to show relative similarity among nearby characters based jointly on structure in image space and language in output space. Figure 3b illustrates training label error rate for various learning models – LSTM with CTC output layer and LSTM encoder-decoder.

[4] Standard LSTM-CTC implementation [Graves et al. (2006)].

5 CONCLUSION

We demonstrate the applicability of sequence to sequence learning for word prediction in printed text OCR. Fixed length representation for variable length input using a recurrent encoder-decoder architecture sets us apart from present day state of the art algorithms. We believe with enough memory space availability, sequence to sequence regime could be a better and efficient alternative for CTC based networks. The network could well be extended for other deep recurrent architectures with variable length inputs, e.g. attention based model to describe the image contents etc.

REFERENCES

ABBYY OCR ver. 9. `http://www.abbyy.com/`.

Tesseract Open-Source OCR. `http://code.google.com/p/tesseract-ocr/`.

Cho, Kyunghyun, van Merrienboer, Bart, Gülçehre, Çaglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, 2014.

Chung, Junyoung, Gülçehre, Çaglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 2014.

Graves, Alex, Fernández, Santiago, Gomez, Faustino, and Schmidhuber, Jürgen. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

Hinton, Geoffrey E., Deng, Li, Yu, Dong, Dahl, George E., Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N., and Kingsbury, Brian. Deep neural networks for acoustic modeling in speech recognition:the shared views of four research groups. *Signal Processing Magazine*, 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 1997.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Kumar, Anand, Jawahar, C. V., and Manmatha, R. Efficient search in document image collections. In *ACCV*, 2007.

Lowe, David G. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey. Generating text with recurrent neural networks. In *ICML*, 2011.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, 2014.

Tieleman, T. and Hinton, G. Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning. *Technical report*, 2012.

van der Maaten, L.J.P and Hinton, G.E. Visualizing high-dimensional data using t-sne. *JMLR*, 2008.

Vinyals, Oriol and Le, Quoc V. A neural conversational model. *CoRR*, 2015.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *CoRR*, 2014.

Zaremba, Wojciech and Sutskever, Ilya. Learning to execute. *CoRR*, 2014.