Unsupervised Feature Learning for Optical Character Recognition

Devendra K Sahu and C. V. Jawahar
Center for Visual Information Technology, IIIT Hyderabad, India.

***Abstract*—Most of the popular optical character recognition (OCR) architectures use a set of handcrafted features and a powerful classifier for isolated character classification. Success of these methods often depend on the suitability of these features for the language of interest. In recent years, whole word recognition based on Recurrent Neural Networks (RNN) has gained popularity. These methods use simple features such as raw pixel values or profiles. Success of these methods depend on the learning capabilities of these networks to encode the script and language information. In this work, we investigate the possibility of learning an appropriate set of features for designing OCR for a specific language. We learn the language specific features from the data with no supervision. This enables the seamless adaptation of the architecture across languages. In this work, we learn features using a stacked Restricted Boltzman Machines (RBM) and use it with the RNN based recognition solution. We validate our method on five different languages. In addition, these novel features also result in better convergence rate of the RNNs.**

Keywords*—*Word Prediction,* OCR, RNN, *Feature Learning.

I. Introduction

Learning appropriate representation for characterizing images have gained lots of attention in the recent years with deep and rich representations [1, 2]. In this work, we investigate the possibility of learning features for OCRs. We are especially interested in the recognition solutions based on Recurrent Neural Networks (RNN) which has emerged as the state of the art for recognizing printed [3], handwritten [4] and natural scene text [5]. This is especially relevant since the RNN based recognition schemes have been using mostly simple features such as profile representations. In this work, we learn a set of features and demonstrate more than 50% reduction in error rates for five different languages.

Conventional solutions to the OCR problem often demands a set of handcrafted features based on the shape of the characters and properties of the characters in the languages. Characterizing the shape or appearance with the help of statistical or structural features is very popular in literature. However, engineering the appropriate set of features have remained as more of an art than science. Handcrafting features for OCRs is a script-dependent process, which can be tedious. On the other hand, RNNs suffer from the burden of simplistic features. RNNs are a multilayer architecture where each layer "abstracts" out useful information from the previous layer. Thus, when presented with simplistic features, extracting complex concepts from them require the use of multiple layers. This leads to two problems with their training. First, having multiple layers increases the amount of data required to train RNNs, since naturally more data is required to get better estimates of huge number of parameters. Second, each iteration of the training is longer since information has to be propagated through a bigger network. In this work, we try to bridge these two different attempts by learning a set of features that are compatible with the word prediction architectures based on RNNs. To alleviate problems with both these approaches, we explore creating a representation that is not handcrafted but *learnt* from data. Such an approach simultaneously increases the portability of OCR architectures to multiple languages while reducing the time required for training the word recognition engine.

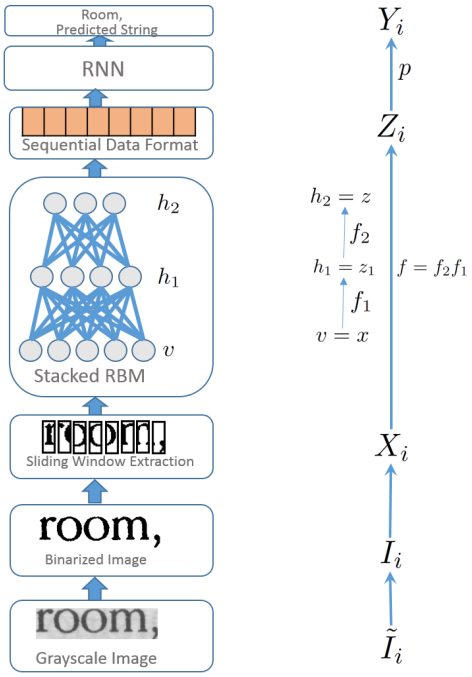
We use Restricted Boltzman Machines (RBMs) for learning features in an unsupervised manner. We start with a simple representation and learn the language specific features by finding the compact low-dimensional representation using RBMs. We learn a hierarchical representation by stacking RBMs with compressive architecture. The choice of RBM is supported by its attractive properties. First, it is an unsupervised feature learner and has an expressive representation. Second, it can be trained efficiently using contrastive divergence [6, 7]. It is also possibile to stack them so that they can learn hierarchical organization of explanatory factors in the data. Stacked RBM fits in our setting because we do not have well segmented labelled character windows for word prediction in OCR problem.

There have been many previous attempts in learning the feature representations for OCRs. Use of Principal Component Analysis (PCA) for creating a set of data specific basis vectors is a well known technique in OCRs. PCAs are linear and do not encode the nonlinear relationships in the data. RBMs, on the other hand, have the inherent capability to capture the nonlinear relationships. For the word prediction problem, an alternative to RNNs have been Hidden Markov Models (HMM). HMMs, which are popular for speech recognition had also seen the benefits of feature learning in offline handwriting recognition [8]. Chandra *et al.* [9] proposed a mixture model K-RBM, to learn feature representations on image patches and demonstrated its performance on object recognition.

In this work, we demonstrate our method on five different languages: English, Marathi, Telugu, Kannada and Malayalam. In all these cases, our feature learning method consistently improves the accuracy over profile features [10] which is a typical example of a hand crafted feature for the word recognition. Other advantages include that the learnt features will be of lower dimension and therefore are compact, efficient and takes smaller training time. We also demonstrate that, this results in faster convergence of RNN, which implies that the learnt features are more appropriate for the problem of interest.

II. Feature Learning for word prediction

We propose a framework for word prediction for printed text where instead of using features like profile [10], we give learned representations to the predictor. Our proposed pipeline is a cascade of two models namely a stacked RBM for learning representations followed by a RNN which is used as a word predictor (Figure 1a). The aim is to give compact, rich and expressive features to the word predictor



(a) Word prediction pipeline (b) Symbolic Pipeline

Fig. 1: Word Prediction Pipeline. (1a) Pictorial pipeline and (1b) Symbolic pipeline. Input image $\tilde{I}_i$ is converted to binarized image I_i, which is converted into a sequence by running a sliding window. Each vector in the sequence goes through a non-linear transformation $f = f_2 f_1$ using stacked RBM. These encoded windows collected in transformed sequence Z_i which is then given to BLSTM predictor p which outputs string Y_i.

which is learnt from the data in an unsupervised manner. Our representations are fairly adaptable to change in languages (shown in section III) which can have varying structural and statistical properties in their symbol set. Stacked RBMs also give compact and expressive representations because of being multi-layered distributed representations [11].

We start with a printed text image with variable width across samples. We first convert the image into sequential data by extracting a sequence of windows from the image by a sliding a window of fixed width and vectorize each window in that sequence by stacking all the pixels. Here each window might contain a partial, full or combination of multiple characters. The dimensionality of this data is reduced using stacked RBMs. Stacked RBM are unsupervised feature learners and need no supervision while learning. This is important because we do not have segmented characters. We also preferred an unsupervised setting as labels for partial character windows is difficult to obtain and are not unique. It also gives explicit non-linear projections which is desirable due to that fact that once learning is done we only need the learnt parameters to describe the model unlike kernel projection methods. Once parameters of RBM is learnt, we only need to do a matrix vector multiplication followed by a non-linearity for each layer to obtain our non-linear projection which can be done efficiently.

Let $\{\tilde{I}_i, Y_i\}_{i=1}^N$ be our dataset and image $\tilde{I}_i$ lies in $\mathbb{R}^{H \times W_i}$ where H is word image height common for all images and W_i is the width of i^{th} word image, $Y_i = \{y_1, y_2, ..., y_{M_i}\}$ be the corresponding label which is a sequence of M_i unicode characters. Let $\{I_i, Y_i\}_{i=1}^N$ be the set obtained by binarizing all images $\{\tilde{I}_i\}_{i=1}^N$. We represent I_i as a sequence $X_i = \{x_1, x_2, ..., x_{L_i}\}$ of L_i vectors lying in $\{0,1\}^{d_v}$ by running a sliding window of dimensions $w_h \times w_w$ with a step size of s and unrolling the windows obtained into a vector of length $d_v = w_h \times w_w$. Here, w_h is the height of the sliding window and is equal to the image height H, w_w is the width of the sliding window and d_v is the length of vectors in sequences after unrolling the window into a vector.

A. Feature Learning using Stacked RBM

Let $\{X_i\}_{i=1}^N$ be the unlabelled dataset used for training stacked RBM and X_i be a vector sequence of i^{th} printed word image obtained by sliding a window over corresponding I_i. Unsupervised feature learning in stacked RBM can be seen as learning a non-linear projection $f : x \rightarrow z \mid x \in \{0,1\}^{d_v}, z \in \mathbb{R}^{d_h}$. Therefore, we project our dataset $\{X_i = \{x_1, x_2, ..., x_{L_i}\}\}_{i=1}^N \xrightarrow{f} \{Z_i = \{z_1, z_2, ..., z_{L_i}\}\}_{i=1}^N$ using the learned non-linear projection f. Here, Z_i's are vector sequences where each vector in the sequence is a learnt compact and expressive representation. In the following paragraphs we focus on non-linear projection f using stacked RBM and RBM is trained on a sampled subset from collection of all vectors in all sequences X_i.

Now we briefly review Restricted Boltzman Machines (RBM) which is the core component used in our feature learning step. An RBM defines a distribution over $(\mathbf{v}, \mathbf{h}) \in \{0,1\}^{n_v \times n_h}$, $\mathbf{v}$ being visible variables and $\mathbf{h}$ being hidden variables. Here, $\mathbf{v} \in \{0,1\}^{n_v}$ corresponds to each vector $x_j \mid j \in \{1, 2, ..., L_i\}$ in each sequence X_i and $\mathbf{h} \in \{0,1\}^{n_h}$ corresponds to learned representation from corresponding RBM in stack. It can be observed in stacked RBM block in Figure 1a, any two consecutive layers constitute an RBM, like $(\mathbf{v}, \mathbf{h}_1)$ and $(\mathbf{h}_1, \mathbf{h}_2)$ are two RBMs stacked to form a two layer stacked RBM. Equation 1, 2, 3 and 4 describe the energy function, probability distribution, and the conditional distribution of random variables of one layer conditioned on the other and vice versa respectively.

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T W \mathbf{h} - b^T \mathbf{v} - c^T \mathbf{h} \tag{1}$$

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \tag{2}$$

$$P(h_j = 1|\mathbf{v}) = sigmoid(W^T \mathbf{v} + c)_j \tag{3}$$

$$P(v_i = 1|\mathbf{h}) = sigmoid(W \mathbf{h} + b)_i \tag{4}$$

Here $W \in \mathbb{R}^{n_v \times n_h}$ n_v, n_h are number of visible and number of hidden nodes respectively in the RBM. $W_{i,j}$ is symmetric weights between v_i and h_j. W, b, c are the parameters of the model and needs to be learned from the data by maximizing the log likelihood of the data with respect to parameters. The learning can be efficiently done by contrastive divergence (CD)[6]. Lets consider a two layer stacked RBM as shown in stacked RBM block of Figure 1a, Here for first RBM, $n_v = d_v$ and $n_h = n_{h_1}$ and for last RBM in stack $n_v = n_{h_1}$ and $n_h = d_h$.

Now we stack RBMs to build our projection f. We achieve this by cascading a series of RBMs and learning a series of latent intermediate representations, for example: $x \xrightarrow{f1} z_1 \xrightarrow{f2}$

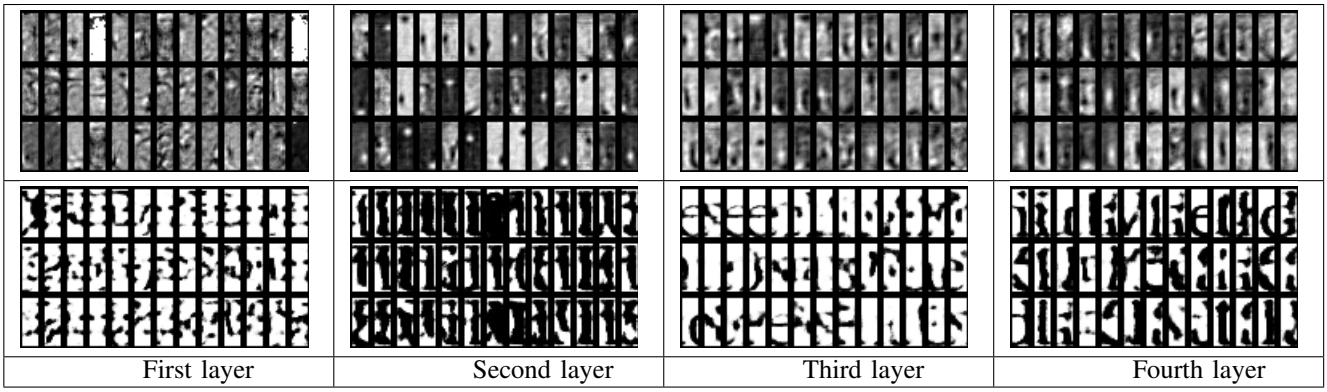


Fig. 2: Visualizing hidden layers in original image space using two methods of visualization, (First Row) Linear combination of previous layers (Second Row) Sampling.

$z_2 \xrightarrow{f3} z$ in 3 layer stacked RBM. Here, f can be seen as cascade of three projections $f = f_3 f_2 f_1$ and each f_1, f_2, f_3 can be learned by an RBM which gives f some characteristics like efficient, compact, non-linear and hierarchical representation [2]. We learn the intermediate latent representation z_1 using our dataset $\{X_i\}_{i=1}^N$ as input to RBM and the intermediate projection of dataset $\{X_i\}_{i=1}^N$ be $\{Z_{1i}\}_{i=1}^N$. We now get next intermediate representation z_2 by using $\{Z_{1i}\}_{i=1}^N$ as dataset for next RBM in cascade and keep repeating this till we get our final representation z. These representations can be used to represent the whole dataset $\{X_i\}_{i=1}^N$ as $\{Z_i\}_{i=1}^N$ which is input to Recurrent Neural Network(RNN) as discussed below.

B. Recurrent Neural Network(RNN)

We obtain a dataset $\{Z_i, Y_i\}_{i=1}^N$ from projecting $\{X_i, Y_i\}_{i=1}^N$ using stacked RBM as discussed earlier. Z_i and X_i are corresponding vector sequences of dimensionality d_h and d_v respectively, given sequence length remains same. We want to learn a mapping $p : Z_i \rightarrow Y_i$ from the data $\{Z_i, Y_i\}_{i=1}^N$. The predictor p can be seen as a mapping which tries to map each $Z_i \xrightarrow{p} Y_i$. Our choice for this predictor p is Bidirectional LSTM(BLSTM) which is a Recurrent Neural Network(RNN).

LSTM networks can remember long range context over several timesteps in the sequence. The output layer of our preferred network is Connectionist Temporal Classification (CTC) layer[12] which allows the network to handle a sequence of unsegmented features and perfectly match our requirement i.e. we learn features on set acquired from sliding windows which are not well segmented symbols. This enables us to use segmentation free model. LSTM networks are also easily trainable unlike general recurrent neural networks.

A BLSTM network contains two LSTM networks in which one network takes input from beginning to the end while other network takes the input from end to beginning. The activations from both networks are then used to predict the final output which is done using Connectionist Temporal Classification (CTC)[12]. CTC layer directly outputs the probability distribution over label sequences. The CTC objective function is defined as the negative log probability of the network correctly labelling the entire training set. Details for BLSTM and CTC algorithm can be found in [4, 12]. The measure of performance for word prediction is label error rate and sequence error rate. Label error rate is ratio of sum of insertions, deletions and substitutions relative to length of ground truth and sequence error rate is the ratio of number of words misclassified to total number of words.

C. Visualization

In Figure 2, we show qualitative interpretation of high level features represented by stacked RBM. Here, the goal is to visualize hidden nodes in arbitrary layer of a deep network in image space. We use two visualization methods as also discussed in [13], namely by sampling and linear combination of previous units. These visualizations also help to appreciate hierarchical representations from deep networks and help us understand what models have learned. It is not surprising that the visualization associated with hidden nodes in first hidden layer are the weights itself but we want to visualize higher layers, tools for which is described in [13]. We use two visualization techniques, namely sampling and linear combination of previous units as described below. Our observations also suggest that sampling methods produce better visualization than linear combination method for this application as shown in Figure 2.

1) Linear combination of previous units: Simplest possible way to visualize hidden units of stacked RBM is by linearly combining of features of hidden units of previous layers. In order to visualize n_l hidden units of l_{th} layer, we plot feature maps formed by $F = W_1 W_2 \cdots W_l \in \mathbb{R}^{d_v \times n_l}$ where each column of F can be formed into $w_h \times w_w$ feature maps to be visualized. First row in Figure 2 shows visualizaton of hidden units of different layers. This method is simple but looses non-linearly and is inferior to other alternatives to visualization as mentioned in [13].

2) Sampling: RBMs can be stacked to form a Deep Belief Network (DBN), as these models are associated with generative process, we can use visualization to get an insight as to what individual hidden units represent in image space. In order to visualize higher layers, consider a DBN with j layers. In particular, layers $j-1$ and j form an RBM from which we can sample using block Gibbs sampling, which successively samples from $P(\mathbf{h}_{j-1} \mid \mathbf{h}_j)$ and $P(\mathbf{h}_j \mid \mathbf{h}_{j-1})$. Here, the conditionals are same as Equations 3 and 4 with $\mathbf{v}$ replaced with $\mathbf{h}_{j-1}$. $\mathbf{h}_j$ is binary vector of units from layer j. In order to visualise the hidden node h_{ij}, we clamp this unit to 1 and perform gibbs sampling for layer j and $j-1$. Then we perform

TABLE I: Comparison of RBM and Profile features of OCR Corpus for RBM(160-90) & BLSTM(50-50-50). The numbers indicate number of hidden nodes in each layer.

Languages	LabelError(%) Profile	LabelError(%) RBM	SeqError(%) Profile	SeqError(%) RBM
English	0.84	**0.22**	2.84	**0.65**
Kannada	4.82	**2.82**	18.23	**10.81**
Malayalam	1.38	**0.66**	9.11	**3.78**
Marathi	3.79	**1.43**	13.40	**5.68**
Telugu	5.78	**2.27**	22.67	**10.02**

TABLE II: Measure of statistical significance of performance gain using paired t-test.

Languages	Mean Performance Gain(%)	Standard Deviation(%)	t statistics	p value
English	0.75	0.0603	25.05	$1.39e{-}04$
Kannada	2.62	0.4340	12.08	$1.20e{-}03$
Malayalam	0.80	0.1433	11.16	$1.50e{-}03$
Marathi	2.98	0.5289	11.27	$1.50e{-}03$
Telugu	3.28	0.4542	14.45	$7.17e{-}04$

ancestral top down sampling in DBN from layer $j-1$ to input layer. This produces a distribution $p_j(\mathbf{v} \mid h_{ij} = 1)$ which is used to characterize h_{ij}. Here, $p_j(\mathbf{v} \mid h_{ij} = 1)$ is the distribution over random variable $\mathbf{v}$ when i_{th} hidden unit in layer j is clamped to 1. This can be done by either by sampling from this distribution or by summarizing the information by computing the expectation $E[\mathbf{v} \mid h_{ij} = 1]$. Second row in Figure 2 shows visualization of hidden nodes for each layer using sampling method. We can observe that as we move from layer 1 toward layer 4 the visualization are more abstract and each node has more sense of the structure in the data. We can also observe in Figure 2 second row fourth layer, the visualization looks like parts of characters of english. We can see the higher layers has more sense of global structure of data.

D. Discussions

Given a dataset $\{X_i, Y_i\}_{i=1}^{N}$ where X_i is the input sequence and Y_i is the ground truth output string. We are trying to achieve the mapping $m = pf : X_i \xrightarrow{f} Z_i \xrightarrow{p} Y_i$ by learning intermediate compact representation Z_i in hope that it would give compact and expressive representation to sequence predictor p. The mapping $m = pf$ can be seen as getting the output word prediction by transforming to multiple intermediate representations in hope that the feature presented to the predictor p would be compact, expressive and hierarchical in representation. We can also interpret learning f as learning a non-linear operator $\{f : x \rightarrow z \mid x \in \{0,1\}^{d_v}, z \in \mathbb{R}^{d_h}\}$ through a combination of series of non-linear operators $\{f = f_3 f_2 f_1 : x \rightarrow z_1 \rightarrow z_2 \rightarrow z\}$, each f_1, f_2, f_3 can be learned by an RBM which gives f some characteristics like efficient, compact, non-linear and hierarchical representation [2].

III. Experiments & Discussion

A. Dataset and Implementation Details

We choose five languages for comparison, namely English Kannada, Malayalam, Marathi and Telugu with 295K, 171K, 65K, 135K and 137K samples respectively. Four Indian languages namely Kannada, Malayalam, Marathi and Telugu are taken from OCR corpus [14] and English is taken from nine books of our internal dataset.

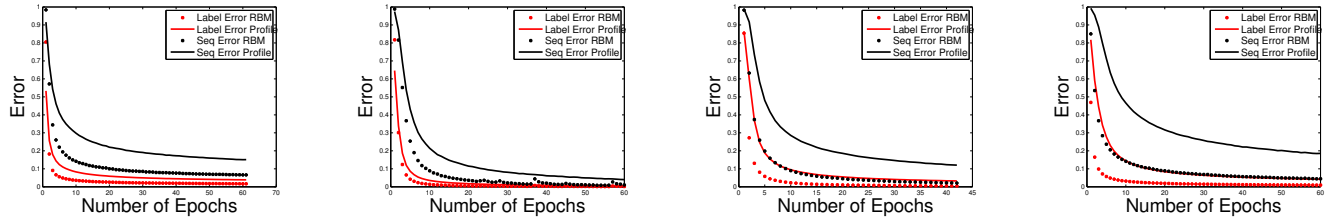
All images are binarized using *otsu* thresholding. We resize all word images to a height $H = w_h = 30$ pixels while maintaining the aspect ratio. We extract sequence of vectors for each word image by running a sliding window of width $w_w = 10$ pixels and step $s = 3$ pixels and unrolling it into a vector. BLSTM classifier has a constraint that number of windows in an image should be greater than ground truth sequence length and our chosen window width and step size does satisfy this for all languages. Then we perform unsupervised feature learning using a stacked RBM. We use around 200K windows for each language to train stacked RBM. All the vector sequences are encoded with stacked RBM and a BLSTM model is learned on these encoded vector sequences. Learning rate and momentum for RBM was set to 0.1 and 0.9 respectively. For BLSTM, learning rate and momentum was set to 0.0001 and 0.9 respectively. For fair comparison between profile features and RBM features, everything is kept fixed which includes word image height, windows size, step size for sliding window, BLSTM architecture. We tuned the hyper-parameters by measuring performance on validation set. In our experience momentum value of 0.9 or 0.95 works well for many different applications but training is very dependent on learning rate.

B. Results

The main purpose of presentation is comparison between hand engineered profile features[10] and features learned from data using stacked RBM. We show that features learned by using stacked RBM are compact, expressive and performs better than profile feature[10] for many languages. For all experiments we perform validation by three way data split, 60% for training, 20% for validation and 20% for testing. Figure 2 shows visualization of feature maps learned at first four layers. Each feature map is associated with a hidden unit. We visualize the higher layer as linear combination of feature maps of previous layers or by sampling particles in visible space given a specific hidden nodes being activated in a DBN. We observe that the learnt feature maps look like region detectors to partial alphabets in English language as we move from first layer towards fourth layer. Also all windows in data can be represented by a cascade of linear combination of feature maps followed by non-linearity.

We performed statistical testing of performance gain using paired t-test. We set our critical value(significance level) α = 0.05. For each language we create experiments by randomly sampling 1.0, 0.9, 0.8 and 0.6 of training population ratio and perform paired t-test on these generated populations. Table II shows mean performance gain, standard deviation, t-statistic and p-value for each language. p-value is less than α for all languages. Hence, stacked RBMs give statistically significant performance gain as compared to profile features.

Table I shows comparison of profile features[10] and stacked RBM features. We can clearly notice that RBM performs significantly better than profile features[10] in all five languages with respect to label error rate and sequence error rate. Table III shows the distribution of label errors into four bins. We can see that performance of stacked RBM features is much better as compared to profile features and it makes very few label error even in bin 1-2 with respect to profile features which shows that the learned representations as expressive. The data in bar graph is composed of four languages.



(a) Kannada (b) Malayalam (c) Marathi (d) Telugu

Fig. 3: Error convergence comparison of RBM (160-90) vs Profile feature for different languages. This figure demonstrates that stacked RBM features makes learning stage of BLSTM predictor converge faster in comparison to profile features. So stacked RBM features show better convergence rates.

TABLE III: Number of Sequences v/s Edit Distance. The table shows a comparison of number of sequences having edit distance in specific bin. We can see lower errors for RBM.

Edit Distance	1-2	3-4	5-6	7-10
Profile[10]	46299	4535	1264	1001
RBM	18402	1867	695	725

The convergence comparison is done in Figure 3 which highlights that with respect to number of epochs stacked RBM features show better convergence rates compared to profile features. Stacked RBM supports the BLSTM network in converging faster in comparison to profile features and can be observed in Figure 3.

C. Discussions

Stacked RBMs are feature learners which disentangle hidden factors of variations. For example, for natural scenes first layer can learn edge detectors, next layer can learn composition of features learnt from previous layers i.e. next layer can learn corner detectors. We can observe in second row of Figure 2 that features learnt at fourth layer look like partial characters of English. Also input image can be reconstructed by combination of those feature maps. Therefore, as we visualize higher layers we reach toward more abstract concepts like moving from pixels to small regions. Another advantage of using stacked RBM with bottleneck architecture is that it reduces the dimensionality of input and can speed up computation compare to raw features. We can also see that even if dimensionality of stacked RBM features have increased compared to profile features. This could be compensated to some extent by faster convergence rates and better performance but time taken per epoch is higher than profile features which is not surprising as profile features have much lower dimension. We also observe that with same architecture our proposed method performs better than RAW features for english but further experimentation is required for other languages.

IV. Conclusion

We proposed a framework for word prediction for printed text where learned feature representation are obtained by stacked RBM and a sequence predictor BLSTM. The framework is segmentation free and does not depend on hand engineered features. We investigate the expressiveness of the stacked RBM features which were learned in an unsupervised fashion. We demonstrate better results compared to hand engineered profile features which had worked well in similar settings in past. Stacked RBM features were expressive across many languages which has variation in structural and statistical properties of symbols. Our experiments show that stacked RBMs can learn feature representations from data and perform better than profile features in generic settings. It also have better empirical convergence rates. Unsupervised feature learning is applicable to script identification and script specific learning problems.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[2] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[3] T. Breuel, A. Ul-Hasan, M. Al-Azawi, and F. Shafait, "High-performance ocr for printed english and fraktur using lstm networks," in *ICDAR*, 2013.

[4] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2009.

[5] B. Su and S. Lu, "Accurate scene text recognition based on recurrent neural networks," in *ACCV*, 2014.

[6] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[7] T. Tieleman, "Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient," in *ICML*, 2008.

[8] Y. N. Hammerla, T. Plotz, S. Vajda, and G. A. Fink, "Towards feature learning for hmm-based offline handwriting recognition," in *International Workshop on Frontiers in Arabic Handwriting Recognition*, 2010.

[9] S. Chandra, S. Kumar, and C. V. Jawahar, "Learning multiple non-linear sub-spaces using k-rbms," in *CVPR*, 2013.

[10] P. Krishnan, N. Sankaran, A. K. Singh, and C. V. Jawahar, "Towards a robust ocr system for indic scripts," in *DAS, 2014*, 2014.

[11] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, 2009.

[12] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," University of Montreal, Tech. Rep. 1341, Jun. 2009.

[14] C. V. Jawahar and A. Kumar, "Content-level Annotation of Large Collection of Printed Document Images," in *ICDAR*, 2007.