Deep Learning Applied To Arabic And Latin Scripts: A Review
Muhammad Kashif Siddhu, Shahrul Nizam Yaakob

Abstract: Over the last few years deep learning has out classed traditional machine learning in several domains like machine translation, computer vision, speech recognition, natural language processing etc. The advent of neural networks based architectures and deep reinforcement learning has revolutionized the field of machine learning. Document analysis community has also taken advantage of this new era. Recently deep learning methods employed in document analysis have enjoyed tremendous success. These methods are robust to deformations, scaling and rotation. Therefore, they are best suited for text recognition. Particularly, convolutional neural networks and recurrent neural networks coupled with embedded attributes have been exploited extensively in word spotting as well as text recognition. In this paper, we summarize and compare most significant deep learning techniques used so far in Arabic, Urdu, Pashto and Latin scripts. A brief introduction to the state-of-the-art frameworks and libraries for building deep learning based systems is introduced at the end of the article. The article concludes by identifying unexplored possibilities which may serve as guidelines for future work.

Index Terms: Convolutional Neural Networks, Deep reinforcement Learning, Embedded attributes, Handwriting recognition, Recurrent neural networks, Word spotting, Siamese Networks.

1 INTRODUCTION
Document analyses is a vast field but in this work we focus on the task of text recognition and word spotting. Text recognition involves conversion of the text images (printed or handwritten) into machine editable text. It is a successful solution for printed text but for historical documents with degraded text and difficult document design, the text recognition systems have poor performance. Same is true for multi-writer handwritten documents. Poor writing style, diversity in writing styles as well as cursive nature of text make it difficult for handwriting recognition systems to achieve 100% accuracy. For these scenarios, an alternative approach called word spotting or keyword spotting in handwritten documents is introduced [1]. In keyword spotting the user provides a query to be searched in the document. The keyword spotting system matches the query with the document image and retrieves a list of matching words using a matching algorithm. If the system accepts the query in the form of image, then the approach is called Query by Example (QBE) word spotting and if the query is in the form of string then it is termed as Query by String (QBS) word spotting.

In traditional machine learning, features are first extracted from the input, then these features are passed through a classifier to “train” it on that input. The classifier is trained on several labelled or known examples before it is able to classify or recognize the unseen examples. The classifier is a shallow network which means having only one layer. Feature engineering plays a vital role in the performance of these systems. It requires deep expertise of the domain and tremendous human effort. Some of the examples of such feature extraction techniques which achieved state-of-the-art results in handwriting recognition include Scale Invariant Feature Transform (SIFT) [2], Histogram of Oriented Gradient (HOG) [3], Local Binary Pattern (LBP) [4] features etc. An advantage of these approaches is that they don’t need high computational power and can be run on CPUs. Deep learning is a sub field of machine learning. Deep learning algorithms involve multiple neural network layers. This is the reason they need a large scale annotated data for training as well as high performance parallel computing systems. Although LeCun et al. [5] introduced deep neural networks in 1998, due to the scarcity of large scale publically available data, limited computation power, vanishing gradient problem and above all inferior performance compared with other machine learning techniques, it couldn’t win the attention of the researchers.

Deep learning gained its popularity in 2006 by the ground breaking work of Hinton et al. [6] in which he introduced Deep Belief Networks (DBN). From there on, deep learning has overcome the field of computer vision, speech processing, natural language processing etc. This is made possible due to the following reasons:

1) Automatic feature extraction
2) Parallel processing using Graphics Processing Units (GPU)
3) Large scale publically available data
4) Availability of Pre-trained models
5) Deep learning frameworks like Tensorflow, Keras, PyTorch, MXNET etc. for rapid development of deep learning applications
6) Handling of problems like vanishing gradient using ReLu activation function, overfitting using dropout and efficient training using batch normalization

This work reviews the use of deep learning techniques in printed as well as handwritten documents in Arabic or Latin scripts. Although most of the research is carried out using Long Short Term Memory Networks (LSTM) and Convolutional Neural Networks (CNN), the latest works also use state-of-the-art techniques like the of attention mechanism, embedded attributes and reinforcement learning. In addition to that there have been efforts to use unsupervised learning technologies as there is a lack of availability of labelled handwritten datasets.

The contributions of the paper are as follows:

1) To the best of our knowledge this is the first survey on the use of deep learning techniques in documents analysis.
2) Summarize deep learning models and techniques used in the document analysis research and highlight significance and shortcomings.
3) Highlight the unexplored areas of deep learning.
4) Introduce latest deep learning frameworks

The paper is organized as follows: Section 2, 3, 4 and 5 present the works done in Arabic, Urdu, Pashto and Latin scripts respectively. Section 6 introduces the most popular frameworks for deep learning and section 7 is the conclusion and future directions.

2 ARABIC

Arabic handwritten text recognition is considered a difficult task compared to Latin script. This is due to its cursive nature, presence of diacritics, diagonal strokes, change in shape of characters according to their location in the word, inter-word and intra word spaces and difficulty in segmentation. Deep learning based recognition systems have shown encouraging performance even in the presence of challenges mentioned above. A word in Arabic script may consist of more than one disjoint parts called ligatures. The following section explores the use of deep learning techniques in Arabic text recognition and word spotting.

2.1 LSTM with Connectionist Temporal Classification (CTC)

The pioneers in presenting a deep learning based system for the automatic transcription of handwritten text are Graves and Schmidhuber [10]. Earlier, this task used to be done in two stages 1) feature extraction, which required domain knowledge as well as human expertise and 2) sequence learning using sequential models like HMM. Due to poor handwriting, some characters may not be recognizable even by the humans. In these cases, the context of the character helps to recognize it correctly. Hidden Markov Models (HMMs) showed great success in handwriting recognition due to its context modeling ability. In the deep learning paradigm, the LSTMs have the same feature. This is the reason the authors propose to use Multidimensional Long Short Term Memory (MDLSTM) with RNN in combination with CTC as output layer for this task. As the RNNs suffer from the drawback of vanishing gradient, the RNNs cannot deal with long-term dependencies and hence cannot store long-range context. To solve this problem, the LSTM cells are used in RNNs. MDLSTM can access long-range context in multiple dimensions. One rationale for using CTC based classifier is that they don’t need segmentation for the classification of sequences like text lines. The segmentation process for Arabic text; printed or handwritten is a highly error prone task. The CTC layer has the ability to transcribe the data without prior segmentation. The CTC layer predicts the transcription for the test images in the recognition phase. The out transcriptions are matched with the ground truth of the line image using Levenshtein edit distance. The authors test their system on the dataset used in ICDAR 2009 Arabic handwriting recognition competition [11]. The popular IFN/ENIT [12] dataset of handwritten Tunisian Town names was used in this competition. They achieve an accuracy of 91.4%, which was 4.2% better than the winner of this competition. Rashid et al. [13] experiment on the famous Arabic Printed Text Image dataset APTI [14]. This dataset typically has word images in low resolution. Furthermore, presence of noise and small fonts make it more challenging. However, the database also contains clean images in 10 different fonts in 6 different sizes. The gray scale images are first normalized to mean 0 and standard deviation 1. For the experiments with multi-font sizes, height normalization is performed, as images have different pixel heights. The image is divided into small pixel blocks, which are fed to the MDLSTM network. The CTC output layer has 120 output units. The authors trained 15 different models trained on different fonts and different sizes. For recognition, they devise a model selection procedure to select a suitable classifier for the test image. They define different thresholds for image height. If the test image falls in a specific threshold, it is rescaled to a particular height and is fed to the suitable model for classification. Morillot et al. [15] presented their method in the OpenHart 2013 Competition. The dataset consists of Arabic documents written by different writers. The documents are first passed through different preprocessing steps like text line image extractions, noise removal, slant and skew correction. Then the features are extracted by a sliding window. They extract a feature set of size 37, which includes statistical, geometrical and directional features from each frame. They also develop a bigram language model from the transcriptions of the database based on a dictionary having 22000 most frequent words. The recognition system consists of a Bidirectional Long Short-Term Memory (BLSTM) network. The BLSTMs can record context in both forward and backward directions. The CTC output layer consists of 160 units corresponding to different Arabic and Latin characters, numerals and punctuation marks. The system is trained only on 11% of the dataset. Therefore, they could achieve a recognition rate of 52%. The authors also employ an HMM recognizer for the same TASK and find that BLSTM shows better recognition rate than HMM. Chherawala et al. [16] compare state-of-the-art handcrafted features for handwriting recognition with the automatically generated features learned by an MDLSTM. IFN/ENIT [12] database is used for the experiments. The handcrafted features include distribution features, Concavity features and the popular Local Gradient Histogram (LGH) features. The features extracted from the input image are fed frame wise to an LSTM Network. The CTC output layer then produces the output transcription of the input image. On the other hand, to automatically generate the features, raw pixels from the input image are fed to an MDLSTM network. It is noticed that the distribution features that were actually designed for the Latin script perform better than other handcrafted features and also the automatically generated features. The authors also note that the use of MDLSTM instead of LSTM increases the recognition rate by 0.3%. The best recognition rates achieved are 84.2% while the automatically generated features achieve 72.9%. The authors note that the poor performance of automatically generated features can be improved using an improved network Architecture. Yousefi et al. [17] show that addition of a normalization step to the input image before feeding it to an LSTM Network for automatic feature learning can make the recognition accuracy comparable to that of an MDLSTM network. They compare their work with Chherawala et al. [16] and show significant improvement. The authors normalize the input image using centre-normalizer method, which is implemented in OCRopus [18]. The text image is centralized and then scaled along the vertical axis. Now the variations are only limited to the horizontal axis. This issue of handling the variations along vertical axis made Graves and Schmidhuber [10] to suggest the use of MDLSTM for offline handwriting.
feature extraction. The authors using the same dataset and same evaluation procedure as was used in [16], show that they are able to achieve 87.6% recognition rate for automatically generated features from LSTM rather than MDLSTM. As the handwritten datasets are usually small, larger networks like MDLSTM may overfit. In [19], the effect of adding the dropout layer before, after and inside the MDLSTM is presented. The experiments are performed on IFN/ENIT [12] database. This is also a sequence-learning problem as the unsegmented handwritten Arabic words are fed to the training network along with their transcriptions for training. The output layer has 121 nodes. The researchers reported that the least label error rate is obtained by applying dropout layer before MDLSTM, which is 11.62% while the worst label error rate is obtained without dropout, which is 16.97%. A similar work [20], uses MDLSTM with CTC for recognition of text in segmented lines of the KHATT dataset [21]. They augment the data using bleed through, high frequency, blurriness, edge enhancement and edge more algorithms there by generating 5 more samples for each text line image. One of the latest works [22] also addresses segmentation free text recognition. Their main contribution is on post processing in which they detect an output of vocabulary (OOV) word in the output and then recover it using a dynamic lexicon. The input to the system are text line images from the KHATT database. The system consists of a sequence of five pairs of convolutional and MDLSTM layers and finally a combination of CTC and weighted finite state transducer (WFST) is used to transform the output into a sequence of words, morphemes or part of Arabic words (PAWs). They recognize the words from the posterior probabilities using three different recognition methods: The first method uses a word statistical language model, the second a PAW statistical language model and the third a morpheme statistical language model. In all the three methods, a word is declared OOV if its confidence score is lower than a threshold. Then an OOV recovery method is employed. This method first prepares a lexicon from a large online corpus of words. Then the OOV word is matched with the words in the lexicon to determine the words similar to it. The word with the least Levenshtein distance is selected for the query OOV word. These words are added to the initial lexicon which was prepared using the transcription of the KHATT database.

2.2 LSTM with Attention mechanism and Reinforcement learning
The authors of [23] claim that they are the first to introduce reinforcement learning in the framework of handwriting recognition. They use CNN for feature extraction and then a sequence to sequence strategy to transcribe handwritten text lines. The system incorporates a Policy Network (PN) to implement reinforcement learning which is trained to adaptively choose an optimal context length from a number of context lengths (called action set). An attention module then attends to this context and encodes it into local context. The encoders and decoders are implemented using LSTM.

2.3 Convolutional Neural Networks
In [24], CNN is used as a feature extractor and SVM with Radial Basis Function (RBF) Kernel is used for classification. The authors emphasize on the use of drop out as a regularization technique on the fully connected layer during training to avoid overfitting. Dropout is applied by randomly setting some of the units to zero during the forward pass. A different set of units is randomly selected for every new training sample. The authors test their system on HACDB [25] and IFN/ENIT datasets. As HACDB has lesser number of samples available for training, the authors increase the size of this dataset 10 times by using elastic deformation technique suggested by Simard et al. [26]. For 66 character classes of this database, the system achieves 5.83% error rate which was 0.76% lesser than the system without dropouts.

2.4 Convolutional Neural Networks and Embedded Attributes
The researchers have extensively exploited the use of embedded attributes with CNNs. Poznanski and Wolf [27] represent the transcription of word images using an N-Gram attribute representation instead of labels. The Pyramidal Histogram of Characters (PHOC) [28] embedding is used to learn the attributes rather than the class label of the word image. In PHOC, the input images are transcribed using 5 levels. A binary vector of the size of the number of symbols in the alphabet is used to encode a word. At the first level, this vector is used to represent the complete word. At the second level, two such vectors are used to represent two halves of the word. Similarly at the third level, three such binary vectors are used to represent three halves of a word and so on. Two more binary vectors are used to represent common bigrams and tri-grams in the alphabet. Then all these vectors are concatenated to obtain the attribute representation of the word. A VGG Convolutional Neural Network (CNN) [29] is trained to learn the attributes in each training image. The network representations of the word image obtained after the last convolutional layer and the attribute representation are then projected to a common linear subspace using the regularized Canonical Correlation Analysis (CCA) [30]. They customize the CNN by replacing the fully connected layer with 19 parallel fully connected layers each representing attributes for the above mentioned n-gram layers. One of the limitations of the CNNs is that they need fixed size input e.g. 224X224. So the images have to be cropped or resized. These operations may distort or erase important content in the image. The authors of [31] conclude that the convolutional layers in a CNN can handle arbitrary size images, it’s actually the fully connected layer that needs a fixed sized input. So the authors propose to use spatial pyramid pooling layer [32] before the fully connected layer which can receive arbitrary size input and can still produce fixed output. They term this model as PHOCNet. The authors use PHOC [28] to encode the transcription of the word labels. In the testing phase, the attributes predicted by the fully connected layer are used to recognize the word.

2.5 Convolutional Siamese Network
Other than the conventional classification networks which take an input image and predict its label, there is a class of network which is called Siamese network which takes two input images and calculates a distance between the feature representations of two images. If the distance between query image and the test image is larger than a threshold, the query is not spotted and the result is represented by 0 and if the distance is smaller than the query word is spotted and is represented as 1. This network has two branches. Both branches are convolutional neural networks which have exactly the same architecture as well as the same weights. The last layer is a combined layer which is a distance metric layer. This layer is responsible to compute the loss function distance between the descriptors of
the two images obtained from the last convolutional layer of the two branches of the network. This layer doesn’t use cross entropy loss function as it doesn’t give any notion of distance, rather it just tells whether the predicted label is correct or not. The Siamese networks implement a contrastive loss function where a margin is used. The training is carried out using pairs of images. There are three types of images: Anchor, positive and negative. A pair must include an anchor image and a positive or negative image. During training the goal is to push the negative samples away from the margin and bring the positive samples near the margin. In [33] the authors apply Siamese network to spot words in a historical Arabic word image database called VML [34].

2.6 Unsupervised Learning
In this domain, the system is trained on unlabeled data. The unsupervised algorithm learns useful properties of the structure of the dataset [7]. The earlier works use Deep Belief Networks (DBN) and Convolutional Deep Belief Networks (CDBN). Like CNNs, DBNs and CDBNs are also used for automatic feature extraction. The advantage is that the DBNs and CDBNs are generative models and can be used to extract features even from unlabeled data. In [35] the authors use DBNs and and CDBNs in their experiments for Arabic character and word recognition. For character recognition, they use HACDB [25] database while for the word recognition they use IFN/ENIT database DBNs are a good choice if all the images are aligned by means of size, rotation and translation. CDBNs can learn low level features like lines and edges in the lower layers, arcs and corners in the middle layers and distinctive parts like circles and loops in the higher layers. Since in the unconstrained handwriting from multiple writers, the letters and words can have a lot of variation in size, slant and skew, CDBNs are a better choice than DBNs. Continuing their work [36], they test two regularization techniques namely dropout and dropconnect on the DBN. Dropconnect is another regularization technique used in Neural Networks based deep learning models in which we only set the individual weights of a node to zero rather than the node itself. Therefore, the node remains partially active. DBN with dropout reduces the error rate by 0.91% while DBN with dropconnect reduces the error rate by 1.37%. Unfortunately unsupervised learning techniques like DBNs and CDBNs couldn’t gain much popularity in the document analysis community because of their inferior performance compared to supervised approaches. Recently there is a growing interest in Generative Adversarial Networks (GANs) [36] which have the ability to generate images which look like real images. These are especially useful for low resource languages where only small datasets are available. The GANs have shown good performance on other computer vision tasks like eye gaze and hand pose synthetic data generation [37]. A GAN consists of two networks: A generator and a discriminator. In the domain of document analysis, the generator takes an image unknown to it (also called noise) as input and tries to generate similar image (fake image). The discriminator takes the real image and the fake image produced by the generator and learns to distinguish between the real image and the fake image. It computes the probability that the generated image is real or fake. The training process continues till the generator is able to generate images which the discriminator cannot identify as fake. In [38], GANs were used to generate Arabic words from Open Hart dataset.

An overview of the research on Arabic script is presented in TABLE 1.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Database</th>
<th>Architecture</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graves and Schmidhuber [10]</td>
<td>IFN/ENIT</td>
<td>MDLSTM+CTC</td>
<td>91.4%</td>
</tr>
<tr>
<td>Rashid et al. [13]</td>
<td>APTI</td>
<td>MDLSTM+CTC</td>
<td>99%</td>
</tr>
<tr>
<td>Morillot et al. [15]</td>
<td>OpenHart 2013 Dataset</td>
<td>BLSTM+CTC</td>
<td>52%</td>
</tr>
<tr>
<td>Chherawala et al. [16]</td>
<td>IFN/ENIT</td>
<td>MDLSTM+CTC</td>
<td>84.2%</td>
</tr>
<tr>
<td>Yousefi et al. [17]</td>
<td>IFN/ENIT</td>
<td>LSTM+CTC</td>
<td>87.6%</td>
</tr>
<tr>
<td>Maalej and Kherallah [19]</td>
<td>IFN/ENIT</td>
<td>MDLSTM+CTC</td>
<td>88.38%</td>
</tr>
<tr>
<td>Ahmad et al. [20]</td>
<td>KHATT</td>
<td>MDLSTM+CTC</td>
<td>80.92%</td>
</tr>
<tr>
<td>Jemni et al. [22]</td>
<td>KHATT</td>
<td>MDLSTM+CTC</td>
<td>20.83%</td>
</tr>
<tr>
<td>Gui et al. [23]</td>
<td>KHATT</td>
<td>LSTM+CNN+ATTENTION +REINFORCEMENT</td>
<td>93.07%</td>
</tr>
<tr>
<td>Elleuch [24]</td>
<td>IFN/ENIT (56 Classes)</td>
<td>CNN</td>
<td>94.95%</td>
</tr>
<tr>
<td>Poznanski and Wolf [27]</td>
<td>IFN/ENIT</td>
<td>VGG+PHOC Embedding</td>
<td>96.76%</td>
</tr>
<tr>
<td>Sudholt and Fink [31]</td>
<td>IFN/ENIT</td>
<td>CNN+spatial pyramid pooling layer</td>
<td>96.11%</td>
</tr>
<tr>
<td>Barakat et al. [34]</td>
<td>VML</td>
<td>Convolutional Siamese Network</td>
<td>62%</td>
</tr>
<tr>
<td>Elleuch et al. [36]</td>
<td>IFN/ENIT</td>
<td>DBN + CDBN</td>
<td>83.7%</td>
</tr>
<tr>
<td>Sudholt and Fink [74]</td>
<td>IFN/ENIT</td>
<td>CNN+temporal pyramid pooling</td>
<td>96.6%</td>
</tr>
<tr>
<td>Alonso et al. [38]</td>
<td>OpenHart 2013 Dataset</td>
<td>GAN</td>
<td>-</td>
</tr>
</tbody>
</table>

3 URDU
Urdu script has most of the characters from Arabic alphabet. It is also written cursively from right to left direction. It has 39 characters. Like Arabic, a word in Urdu may consist of more than one disjoint parts called ligatures.

3.1 LSTM and MDLSTM with CTC
Urdu printed text recognition has been extensively carried out using different variations of RNN BLSTM networks on different databases. Ul-hassan et al. [39] use text lines from Urdu Printed Text Image (UPTI) database. The database contains synthetically generated data. It has 10,063 lines. They use 80% of the database for training and validation while 20% for testing. They perform two types of experiments. In the first experiment they label a character without considering its shape variations which occur due to its locations in a ligature. That is to say, a character like Jeem (ق) which can appear in four different shapes depending on its position in the ligature or part of word is labelled by one class rather than 4 classes. While in the second experiment, the same character is
labelled by 4 different classes corresponding to its shape variations due to its location in the word. This results in 99 classes in the first experiment while 191 classes in the second experiment. They employ a BLSTM network with CTC as the output layer for the recognition task. Unsegmented text lines are first resized to a fixed height. A 30 x 1 window is traversed on the line image which extracts the raw pixels from the image. This generates a one dimensional vector which is fed to the classifier along with the ground truth of the text line for training. In [40], the authors introduce their Urdu handwritten dataset named Urdu Nastaleeq Handwritten Database (UNHD). They perform character recognition using 1-Dimensional LSTM RNNs and achieve an error rate of 7.93 % at character level recognition. A similar work [41], uses printed Urdu text dataset named Urdu-Jang. They employ a BLSTM based classifier with CTC and tested their system on the text line images achieving a character level recognition accuracy of 88.94%. Naz et al. [42] use statistical features instead of raw pixels for the recognition of printed text lines using a BLSTM RNN based classifier. They also perform their experiments on UPTI database. In the output layer, they use CTC loss function. They use 13 different features to build the feature vector. They actually train 8 different classifiers on 8 different combinations of features. A sliding window of size 4 X 48 is used to traverse the line image from right to left and top to bottom and then calculate the proposed features on them. They perform their experiments on 43 characters, which also include the blank space. They transcribe the different shape variations of a character due to its location in the word with a single label. They also did not use digits and punctuation marks in the database. This is the reason that the number of used labels by them is lesser than the experiments of Ul-Hassan [39]. Advancing their work, the authors of Naz et al. [43] use BLSTM with CTC and another set of 12 dimensional statistical feature vector. In the preprocessing phase, they normalize the text line images to a height of 48 pixels. Then the text line image is scanned using a sliding window. Features are extracted from each frame. The feature vectors of all of these frames are concatenated and fed to the classifier along with the ground truth of the text line image. The authors experimented on sliding window of sizes 2, 4, 6 and 8 and found the best result using the smallest window, which points to the fact that smaller window can better capture the details of the character image and thus can better handle the issues causing recognition errors. Another variation of the above-mentioned work is presented in Naz et al. [44]. Here again same dataset with the same number of labels has been used. This work also uses unsegmented text line images. Here MDLSTM with CTC is used which scans the image in all directions i.e. left, right, top and bottom. Therefore, it can access the context in all the four directions. The authors claim that the use of MDLSTM can eliminate the recognition error caused due to the diagonal nature of Urdu Nastaleeq script. Zoning features [45] are known to provide high speed and low complexity. Naz et al. [46] test the zoning features on the UPTI database. The text line images are converted to gray scale. They perform 4 experiments using different zoning sizes on their corresponding text image heights i.e. zones of sizes 3x3, 5x5, 7x7 and 9x9 were applied by resizing the text lines to heights of 30,50, 70 and 90 respectively. Features are extracted by superimposing an NxN zone on the line image and calculating average grey level in each zone. The line image is scanned by the zone in left to right and top to bottom. The feature vector is prepared in the same order to be fed to a BLSTM classifier having CTC as output layer. The experiments reveal that the recognition rate increases by increasing the size of the zones and the best recognition rate on character accuracy is achieved using the zone of size 9x9. Segmentation based approaches are known to show better results in text recognition. As segmentation of Urdu text at the character level is a highly error prone task, the text line image is usually segmented at the ligature level. Ahmad et al. [47] segment the text lines of UPTI database using their own proposed segmentation algorithm [48]. Here the ligatures are considered as classes. The model uses raw pixel values as features. The segmented ligatures and their labels were fed to their proposed gated BLSTM (GBLSTM). They modify the gate unit of LSTM by using Rectified Linear units (ReLU) as activation function rather than the sigmoid function. This gate is used to filter signals from forward as well as backward context. Then the element-wise product of the activated forward and backward context is carried out, which results in obtaining common information between the two contexts. This information is then passed to softmax function for final prediction. In a similar work Nizwa et al. [49] experimented on the segmented ligatures obtained from the text line images of an Urdu book. They use CNN to classify the ligatures.

3.2 Combination of MDLSTM and CNN

In the field of computer vision in general, as well as in document analysis, there is a lot of interest in using automatically generated features using Convolutional neural networks. Naz et al. [50] use a convolutional-recursive approach, which extracts features from the text line images using the convolution neural network, but use MDLSTM for the classification. This is because Huang and LeCun [51] concluded that CNNs can extract optimal features but are not good at classification, as the classification layer of CNN is a simple Multilayer Perceptron (MLP), which is not suitable for sequence learning. Instead of randomly initializing the filters of CNN, the authors first trained the CNN on MNIST database. The pixel values of the text line gray scale images are first normalized to (0-1). The text line images are then skeletonized. The skeletonized image is then convolved with the six kernels to get convolved images. These 7 images form the feature vector of text line image. This feature vector along with its ground truth is fed to the MDLSTM classifier for training. The CTC is used as an output layer. To find the label error rate they employ an edit distance formula to compare the ground truth with the predicted transcription. They achieve an accuracy of 98.12% on the test dataset which the authors claim is the best on UTPI dataset.

4 PASHTO

Pashto is one of the national languages of Afghanistan. There are around 38 million native speakers of Pashto in Pakistan also. Pashto script is derived from Arabic script and has 47 letters in its alphabet. Unfortunately there are few works available on Pashto document analysis using deep learning. Contributions in this regard are detailed below.

4.1 MDLSTM and CTC

Ahmad et al. [52] present an extended version of Pashto printed text image database which was originally developed by Mehreen et al. [53]. The database is extended synthetically and now it contains 48,000 images. It contains 1000 Pashto
words in different font sizes and rotations. They test their system on three different systems: an HMM based system, a SIFT descriptor based system and an MDLSTM based system. The LSTM system shows robustness to scale and rotation. In addition to that, it also showed best recognition result (98.9%) and lesser execution time. HMM based system shows the worst performance (89.9%) while SIFT based system had immense memory requirement and very slow recognition time (three minutes per image). Ahmad et al. [54] develop a Pashto text image database from Pashto newspapers, novels, poetry and religious books etc. The pages are segmented into text lines and are transcribed using UTF8 encoding. There are 17,015 text line images in this dataset. The authors test their system on BLSTM and an MDLSTM sequence learning classifiers, both having CTC output layer with 97 units. The MDLSTM system outperformed BLSTM system. An overview of the research on Urdu and Pashto script is presented in Table 2.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Database</th>
<th>Architecture</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ul-Hassan[39]</td>
<td>UPTI</td>
<td>BLSTM + CTC</td>
<td>94.95%</td>
</tr>
<tr>
<td>Ahmed et al.[40]</td>
<td>UNHD</td>
<td>LSTM + CTC</td>
<td>82.07%</td>
</tr>
<tr>
<td>Naz et al.[42]</td>
<td>UPTI</td>
<td>BLSTM + CTC</td>
<td>96.4%</td>
</tr>
<tr>
<td>Naz et al.[43]</td>
<td>UPTI</td>
<td>MDLSTM + CTC</td>
<td>98%</td>
</tr>
<tr>
<td>Naz et al.[44]</td>
<td>UPTI</td>
<td>MDLSTM+CTC+ Zoning Features</td>
<td>93.98%</td>
</tr>
<tr>
<td>Ahmad et al. [47]</td>
<td>UPTI</td>
<td>BLSTM</td>
<td>96.71%</td>
</tr>
<tr>
<td>Nizwa et al.[49]</td>
<td>UPTI</td>
<td>CNN</td>
<td>95%</td>
</tr>
<tr>
<td>Ahmad et al. [52]</td>
<td>Wahab et al.[53]</td>
<td>MDLSTM+ SIFT features</td>
<td>98.9%</td>
</tr>
<tr>
<td>Ahmad et al. [54]</td>
<td>Kpti</td>
<td>MDLSTM+ CTC</td>
<td>90.78%</td>
</tr>
</tbody>
</table>

5 LATIN SCRIPTS
As is evident from the previous sections, there seems less contribution in Arabic scripts regarding the use of deep learning. We now explore here the latest literature on Latin script where the reader may also identify areas which have not been explored for Arabic scripts but have produced excellent performance in Latin scripts.

5.1 BLSTM and CTC
Earlier works in Latin scripts used handcrafted features along with the combination of LSTM and CTC for the recognition of unsegmented text line images. Chherawala et al. [55] propose different context dependent character labelings for the Latin scripts at the ground truth level. The character labels include information about the context. This eventually increases the number of units in the CTC output layer. They compare their proposed hybrid and selected context labelling with n-gram context labelling. They also test different features and identify that the MB features proposed by [56] produce the best results. The experiment performed using Hybrid context labelling in BLSTM outperformed other models achieving 92.8% accuracy on RIMES database [57].

5.2 CNNs and EMBEDDED ATTRIBUTES
Although CNNs have shown cutting edge performance in all applications of computer vision as well as handwriting recognition, but it requires a large amount of annotated data for training. Usually this limitation prevents the handwriting recognition systems to perform well as little amount of data is usually available. This problem specifically effects handwriting recognition system for low resource languages like Urdu, Pashto and even Arabic. The researchers of [58] have shown that if the CNN is trained on synthetic data created using different handwriting-like fonts and applying some data augmentation techniques like affine transformations and elastic deformations, the CNN then requires very little amount of annotated data from the handwritten data set to show competitive performance. Here the PHOCNet proposed by [31] is first trained on the synthetic dataset named HW-SYNTH [59]. Then they train the system on a small fraction of the handwritten datasets and achieve competitive results compared to the experiments where the system was trained on complete training set. In continuation of their work, Krishnan and Jawahar [60] use ResNet34 [9] in their experiments. They introduce region of interest (ROI) pooling. This makes the network to be trained on variable sized word images. To train the network, they use curriculum learning technique [61], where the network is first trained on shorter length words and then on longer words. They also note that given huge number of classes, training the network first on smaller number of classes and then gradually increasing them, helps converge the network fast. The network is first trained on the above mentioned dataset and then it is fine-tuned on George Washington (GW) [62] and IAM datasets [63]. They claim that they are the first to achieve an mAP of 90% on IAM dataset for QBE task. On GW dataset they achieve an mAP of 96%. Inspired by PHOC [28], the authors of [64], suggest another representation of word image labels which they term as Pyramid of Bidirectional Character Sequences (PBSC). This representation can capture the original sequence of characters in a word rather than only representing the presence of a character, as well as it records this sequence in left to right and also right to left direction. They use a CNN similar to the VGG-16 [65] to learn their proposed representation. As in PHOC, the PBSC representation is learned for each word. During testing the output probabilities of the last fully connected layer are translated to the PBSC representation which eventually is used to recognize the word image. They claim that applying their proposed resizing technique and elastic transformation for data augmentation, better results are achieved. The researchers of [66] argue that the PHOC and Discrete Cosine Transform of words (DCToW) [67] representations for strings are not suitable as a small difference in strings produces a big change in their representations. Thus a representation is required which brings similar strings to nearby points in the representation space. They propose Levenshtein Space Deep Embedding (LSDE) in which the word images are also represented in the same space. To get such a representation for strings, a Siamese network consisting of a single convolutional network is trained on pairs of word strings. This network outputs the representation vectors of the word string pair and calculates their Euclidian Edit distance. The goal of the loss function here
is to make the edit distance equal between the vector representations of strings and the Levenshtein distance of strings. To get the representation of the word images, a CNN based on Jaderberg et al. [68] is employed. After the string and image models have been trained, the joint training of image and string pairs is carried out using the respective networks jointly. Now the loss function is implemented by adding the loss functions (mentioned above) between image-image, string-string and image-string pairs. In [69], Retsinas et al. prove that a CNN like PHOCNet can be used to extract features by using the activations of its different layers. They experimented on the feature maps obtained from different layers. They also test the use of manifold learning method called t-sne [70] and show that it improves the results. They also test different distance measures like Euclidian distance, Bray Crutis similarity etc. and found that the normalized versions of these measures produce better results. In the seminal work by Sudholt et al. [71], the authors theoretically prove the validity and significance of loss functions which are suitable for training binary attribute representations like PHOC and the real-valued representations like Discrete Cosine Transform of Words (DCToW) [67]. They prove that the sigmoid activation functions can be coupled with the binary cross entropy loss function. They also prove that the cosine loss is the best loss function. In addition to that, furthering their work [31], they propose another spatial pooling layer which they termed as temporal pyramid pooling (TPP) layer and so the network TPP-PHOCNet. In sequential models, the image is split into frames and the frames are then sent for processing sequentially. In case of TPP, the image is represented such that in the first layer of the pyramid there is no split, in the second layer the image is split into two along the horizontal axis and so on. They experiment on a number of databases including IFN/ENIT, IAM and GW. They achieve the best results with TPP-PHOCNet with mAPs of 97.96% and 97.92% for QBE word spotting on GW and IAM datasets respectively. On ifn/enit, they achieve 96.58% and 94.52% for QBE and QBS respectively. After the success of PHOCNet [31], Eugen Rusakov et al. [72] test the state-of-the-art convolutional neural Networks like ResNet [9] and DenseNet [73] by plugging the PHOC embedding layer at the end of the networks. They term the resulting architectures as PHOCResNet and PHOCDenseNet respectively. As these networks were originally designed for natural images, they made some modifications like dropping the pooling layer, using a Multilayer Perceptron network (MLP) instead of a single fully connected layer, deleting batch normalization layers etc. from the original architectures of these networks to make them more suitable for the word images. Their experiments conclude that it is not necessary that a CNN may perform better just by increasing its depth or that it is new and have achieved best results on some other image dataset. They compare the performance of TPP-PHOCNet, PHOCNet, PHOCResNet and PHOCDenseNet for QBE and QBS word spotting on GW and IAM dataset. TPP-PHOCNet achieve best results on GW database in QBS word spotting, while PHOCResNet achieve highest results in QBE and QBS case on IAM dataset. The deepest highly parametrized PHOCDenseNet was not able to outperform other simpler networks on any of the datasets. Building on [71] and [72], Eugen Rusakov et al. [74] propose STPP-PHOCNet which uses a concatenation of Temporal Pyramid Pooling and Spatial Pyramid Pooling. For training they use the binary cross entropy loss (BCE). They carried out word spotting using cosine similarity and their proposed Probabilistic retrieval model (PRM) which is based on Direct Attribute Prediction (DAP) [75]. For cosine similarity, a nearest neighbor search is carried out in the string embedding space while PRM gives a similarity between the probability of PHOC representations of the test set and the predicted PHOC probability of the query. RPM provides robust distance measure in high dimensional representation which Cosine and Euclidean distance are unable to provide. In [76], the authors present a segmentation-free QBS word-spotting system which takes in the whole handwritten page as input. First a ResNet [9] is used to down sample it by a factor of 8. This down sampled feature map is then passed through a Region Proposal Network (RPN) [77] that extracts as well as scores the region proposals. On these two outputs, non-max suppression (NMS) is applied which removes redundant proposals. Dilated Text Proposals are also extracted from the page directly and are added to the region proposals. Then bilinear interpolation is applied which gives descriptors for each proposal region. Following that, another CNN is applied and the extracted feature maps are fed to three separate FC layers: one to give coordinates of the box, another to give embedding and then another one to give the wordness score. Lastly another NMS is applied to give the final region proposals. A word string embedding is computed for each region. For the embedding they use PHOC and DCTow. The query is matched with the proposed regions using the cosine distance. This networked, termed as Ctrl-FNet outperformed other segmentation free word spotting systems on GW and IAM datasets.

5.3 Combination of RNNs and CNNs with Attribute Embeddings

Advancing their work [60], Krishnan et al. [78] use the activations from the penultimate layer of the HWnet2 to generate an embedding representation. During training, they represent each label in PHOC [28]. They use two streams of feature extraction. In one of the streams, the feature representation of the real word image is obtained from the convolutional layers of the ResNet34 [9] network. While in the other stream, feature representation of the corresponding synthetic image is obtained from the convolutional layer of the AlexNet. This feature vector is then concatenated with the PHOC representation of its textual label. To project these features to a common subspace, a Siamese style fully connected embedding layer is proposed which is used to merge both these representations using common subspace regression (CSR) [28]. For word spotting, they use above mentioned network, while for word recognition, they use a custom Convolutional Recurrent Neural Network (CRNN) [79]. They use a Spatial Transformer layer (STN) [80] layer at the start of the network to correct the geometric transformations. Features are learned using ResNet18 which are fed to a bidirectional long short term (BLSTM) network. This network is trained on IIIT-HWS [60] synthetic dataset. In the query by example word spotting task, they achieve a mAP of 98.14% on GW database. In the word recognition task, their CRNNsynth network proposed in this work outclassed recently proposed systems reducing the WER and CER on IAM Dataset to 0.51% and 0.266% respectively. A combined embedding of word images and their corresponding texts into a common embedding sub-space is presented in [81]. The embedding for word image is obtained using a CNN architecture and the
embedding for the corresponding text is obtained using an RNN architecture having Gated Recurrent Units (GRU) [82]. Then an L2 distance loss between these two embeddings is calculated. Two fully connected layers are implemented after the last convolutional layer in the CNN branch of the network to preserve the structural information of the word images. The last one being a sigmoid layer has units corresponding to the PHOC representation of the text of word. A mean cross-entropy loss is calculated between the predicted probabilities and the binary values in PHOC or a bag of characters (BOC). The loss function is calculated by combining these two losses. In QBE word spotting, the framework outputs nearest neighbors of the query image by calculating the Euclidean distance between the embedding of the query image and other images. In QBS, the query string is projected to the embedding space and the nearest neighbors are retrieved. To further refine the matches, the authors propose another two layered dense network, which gives the final list of matches. Toledo et al. [83] use a combination of the well-known PHOCNet and a BLSTM network to recognize word images of GW and the Esposalles (BCN) dataset [84]. Here rather than achieving PHOC attribute embedding of complete words, the researchers use a sliding window to extract same sized patches from the word image and achieve their embedding. Thus a sequence of PHOC attribute embedding is achieved for each word. This sequence is then fed to a BLSTM having a CTC loss at the end to recognize the words. The work of Wu et al. [85] is unique in that, they use a position embedding [86], [87] concept to solve the problem of segmentation-free handwriting recognition. The system takes the word image as input to a 101-layer ResNet to extract the features. Now as many one hot encoded vectors are generated as there are characters in the image to represent the position of the characters. Same number of copies of the feature maps are also generated and each copy is concatenated to a one-hot position vector representing the index of characters in the word. Now these feature maps are fed to a BLSTM network. The output of each hidden state is input to a fully connected layer and lastly the softmax layer predicts the character. After getting the complete string at the output, a lexicon is used to find the word closest to the output string in the lexicon using a distance metric. They achieve a character error rate of 2.78% on Esposalles marriage record database [84]. Most of the work on handwriting recognition is on the segmented words. One of the few works that apply deep learning at the line segmentation level is [88]. The authors experimented on a BLSTM only model and a combination of CNN and BLSTM to recognize the lines from a historical handwritten Spanish database named RODRIGO [89]. In both of the systems mentioned above, CTC loss function is used for an end to end training. The researchers also use a decoder named Weighted Finite State Transducers (WFST) [90] to map CTC labels to words using a language model. The experiments depict that the system based on the combination of Convolutional Neural network and Bidirectional LSTM performed best with a word error rate (WER) of 14.8%.

5.4 Convolutional Siamese Networks
A new concept in word spotting termed as semantic word spotting was proposed in [67]. In semantic word spotting, the system retrieves the word images which are semantically similar to the query e.g. dog is semantically similar to cat and tiger. They also perform the usual verbatim word spotting in which the goal is to retrieve the same word. Their system uses a triplet CNN proposed in [91] which is inspired by Siamese Network. This triplet CNN consists of three replicas of the same CNN which share weights. The input is provided as a set of three images, out of them two belong to same class while one belongs to another class. The CNN learns representations of these images and back propagates the distance between the two similar images and lesser of the similarities between the odd image with the other two images. The authors also propose a novel word representation which they termed as Discrete Cosine Transform of Words (DCToW). Each character is one hot encoded using a vector of length 36 corresponding to the number of letters and numeric symbols. These vectors are concatenated to form a matrix. Discrete Cosine Transform is then applied to each row to get the DCToW representation. For semantic word spotting they derive a word representation from LSTMChar-Large model [92] which they term as semantic representation. From the same framework, they also extract n-gram representation to test for verbatim word spotting. The authors also extract the PHOC representations of the transcription of the word images. For each of the representations mentioned above they perform separate experiments in which they learn an embedding between the descriptors of the image obtained from the Triplet-CNN and their word representations. This is done using a fully connected network. This fully connected networks outputs the image representation equal in size to the word representation. This representation is obtained for each of the three images in the triplet. Now a special loss function is applied which is optimized such that to bring closer the positive images to their word representation and push away the negative image from the current word representation.

5.5 RNNs with Attention Mechanism
Attention based RNN networks have rarely been used for handwriting recognition. One such effort is [93]. This framework consists of an encoder, attention mechanism and a decoder. The encoder has two parts: A VGG-19-BN [29] and a bidirectional Gated Recurrent Unit network (BGRU) [82]. The input image is first fed to the VGG-19-BN network which generates the feature representation of the word image. This representation is reshaped into a matrix. Each column of this matrix is then fed to its respective bidirectional GRU unit. Using the outputs of the GRU units, an attention is calculated which helps the GRU units of the decoder to concentrate only on that part of the input sequence which can be used to predict the current character at the current time step of the decoder. The decoder consists of unilayered GRUs. The authors perform their experiments on the words of IAM database. In a similar work [94], the authors use LeNet-5 [5], which is a convolutional network for feature extraction followed by a sequence2sequence architecture having an attention mechanism. Here rather than using full word image as input to LeNET, the image is fed as sequence of overlapping patches to the convolutional Network which generates the feature representation of each patch. LSTM cells are used in encoder as well as in decoder. They experiment with several architectures involving LSTM and BLSTM layers and the best results are obtained using BLSTM. They achieve the best WER of 12.7 % on IAM database and 6.6% on RIMES database. An overview of the research on Arabic script is presented in Table 3.
6 CONCLUSION

This article presents a review of state-of-the-art in document analysis research which is being carried out using deep learning algorithms and techniques. The article first introduces the significance and advantages of deep learning over traditional machine learning. Thereafter it briefly introduces the prominent works carried out in Arabic, Urdu, Pashto and Latin scripts. We present an overview of these works along with their results and deep learning frameworks used by them in different tables. We highlight our findings and identify some unexplored areas which suggest future directions:

1) Although extensive research has been carried out on text recognition task in Arabic script, the use of deep learning has been scarce for the word spotting task.

2) In case of Urdu and Pashto, to the best of our knowledge we couldn't discover any work on word spotting using deep learning techniques.

3) Attribute embeddings and attention mechanisms are yet to be explored for Urdu and Pashto scripts.

4) Owing to the success of HWNet v2 which exhibited competitive performance by using simple architecture but being trained on massive synthetic data, points to the importance of availability of large scale data.

5) Deep reinforcement learning can open new opportunities for document analysis after its integration with the popular libraries like Tensorflow and PyTorch as well as due to the tools open sourced recently by DeepMind.

6) The attention mechanisms coupled with recurrent neural networks have shown better results than the RNN only networks. But these are computationally expensive and hard to train.

7) The latest advancement in attention mechanisms is end-to-end Attention Convolutional Networks [86] which have been successfully applied for machine translation and scene text recognition tasks. The document analysis research community is yet to explore this area.

As a closing note, deep learning has served to address the challenges of document analysis tasks. With the explosive development of this technology, it will keep on providing new solutions and opportunities to the researchers.

REFERENCES


[40] S. B. Ahmed, S. Naz, S. Swati and M. I. Razzak, "Handwritten Urdu Character Recognition using


