Urdu Handwriting Recognition Using Machine Learning

Submitted by:

Shehryar Malik	2015-EE-167
M. Naeem Maqsood	2015-EE-168
Abdur Rehman Ali	2015-EE-188

Supervised by: Dr. Ubaid Ullah Fiaz

Department of Electrical Engineering

University of Engineering and Technology Lahore

Contents

List of Figures

List of Tables

Chapter 1

Problem Statement

One of the most important factors that drive human intelligence is the ability to read. For machines to ever acquire general artificial intelligence, they must master this art. Due to recent breakthroughs in the field of machine learning, especially in the form of deep learning, recogniton systems that can read documents are more realizable than ever before. Apart from bringing artificial intelligence one step closer to human intelligence, these recogniton systems can assist people in a wide range of affairs. For example, state institutions can use these systems to digitize old records, thus sparing them the need of large storage areas. Similarly, libraries can easily create electronic versions of any old books they might have, thus preserving them for eternity.

Pakistan is an emerging country in the technological world. It has recently been trying to introduce e-governance in its state institutions. One particular initiative taken was the computerization of old handwritten records, including land and medical records. Given that these number in millions, manually computerizing them is a time-consuming task. However, this process can be be accelerated through recognition systems.

This project aims to develop a recognition system for Urdu handwritten documents. Given a handwritten document, the goal is to create a system that can convert it into machine-encoded text, which can then be viewed and edited using a word processing software.

Several approaches have been proposed in this regard. Conventional recognition systems relied on Hidden Markov Models. These have now been replaced by artificial neural networks owing to major recent breakthroughs in the area. A type of artificial neural netwrork, the Recurrent Neural Network (RNN) and its variants, including the Long Short Term Memory (LSTM), the Bi-Dimensional LSTM and the Multi-Dimensional LSTM, are widely popular in recognition systems. This project aims to explore and build upon these existing techniques to develop a robust Urdu handwriting recognition system.

Chapter 2

Literature Review

Machine learning offers a powerful set of tools for analyzing large sets of data. In this setting, an algorithm is required to find a mathematical model for and patterns within a dataset, and is evaluated by the accuracy of its output. As a concrete example, consider the problem of differentiating between apples and oranges in which, given a certain set of properties, such as the texture, color, size and hardness of the fruit, the algorithm must find a way to distinguish between the two fruits.

Central to machine learning is the idea of a *loss function* which keeps a track of the number of correct or incorrect decisions the learning algorithm has made. The output of the loss function is fed back to the algorithm, which uses it to improve itself. This feedback mechanism eliminates the need to program an algorithm explicitly and is, perhaps, the reason behind the effectiveness of machine learning algorithms.

2.1 Types of Machine Learning Algorithms

Machine learning algorithms can be used to solve a wide variety of problems. Below, we present an overview of two settings in which these algorithms are widely employed.

2.1.1 Supervised Learning

Concretely, suppose we have N matrices each of order $M \times D$. We may stack these matrices on top of one another to form a multidimensional array, often known as a tensor, X such that $X \in \mathbf{R}^{N \times M \times D}$. Suppose, also, that we a have a vector y such that $y \in \mathbf{R}^N$. The problem of supervised learning, then, is to find a transformation (or a mapping) from X to y.

If the algorithm outputs a vector $\hat{y}$ where $\hat{y} \in \mathbf{R}^N$, we may then define our *loss function* as follows:

$$L(\varphi) = \|\hat{y} - y\|^2 \tag{2.1}$$

Here φ are the paramaters internal to the algorithm. Using the output of the loss function, the algorithm can find the value of φ that will minimize the loss by solving the following:

$$\frac{\partial L(\varphi)}{\partial \varphi} = 0 \tag{2.2}$$

Equation 2.2 may not have a closed-form solution. So instead an iterative algorithm, called as *stochastic gradient descent*, is used. At each iteration, φ is updated acording to the following rule:

$$\varphi = \varphi - \alpha \nabla_{\varphi} L(\varphi) \tag{2.3}$$

Here α is a hyperparameter known as the *learning rate*.

As an application of these types of algorithms consider the problem of differentiating between apples and oranges presented earlier in this text. Suppose that we have an N number of apples and oranges. For each fruit we have, we consider M of its properties and represent each of them by D numbers. We store all our data in the matrix X. The vector y will then contain labels corresponding to each of these examples. By using this dataset in a supervised learning setting, an alogirthm will be able to find a relationship between the properties of a fruit and its type.

2.1.2 Unsupervised Learning

Consider the following problem: in a cocktail party there are a number of different speakers, each producing a different sound. Suppose that we place an audiorecorder next to these speakers that records sound at different time instants and stores the data in a matrix X. The goal is to separate the sounds produced by each of these speakers at each time instant. Clearly, it is not possible to assign labels (the vector y) to the matrix X. This, then, is the problem of unsupervised learning.

One way of separating the different sounds is through the K-means clustering algorithm. In this setting, the data is partitioned such that similar points are grouped together. Each of these groups or *clusters* is represented by its mean value. One measure of similarity between two data points is the L2-norm of the Euclidean distance between them.

Other unsupervised learning algorithms include the *Expectation-Maximization* algorithm and the *Mixture of Gaussians* model.

Concretely, unsupervised learning algorithms try to draw inferences from datasets that only contain the input data and not the output labels.

2.2 Deep Learning

Deep learning is a subfield of machine learning inspired by the structure and function of the human brain. Algorithms in this class are often known as *artificial neural networks*, or simply, *neural nets*.

Figure 2.1 shows an artificial neural network. Suppose that we have some images of three types of animals: a cat, a dog and a bear. The input layer takes in an image as input, processes it and passes it on to a hidden layer. The hidden layer carries out some more mathematical operations and sends the result to the output layer. The output layer, through some mechanism (discussed later), outputs a *score* for each *class* (in this case three - the cat, the dog and the bear). The scores indicate the likelihood (that the artificial neural network thinks) of that particular image belonging to each class. If we know the correct class of the image, the *labels*, we can compute the error, or *loss*, in the artificial neural network. This error can then be *backpropagated* to the neural network and its parameters updated, in a method similar to the one discussed in Section 2.1.1.

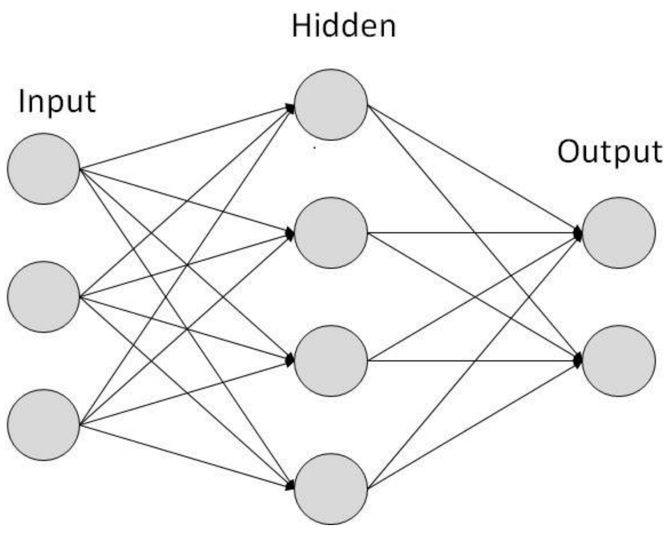


FIGURE 2.1: An Artificial Neural Network

There might be several hidden layers and of different types in one artificial neural network. In this section, we provide an overview of some of these layers. We also discuss some of the commonly used *loss functions*, alternatives to Equation 2.3 and an important concept known as regularization.

2.2.1 The Fully Connected (FC) Layer

Mathematically, the output y of a fully connected layer is given by:

$$y = Wx + b \tag{2.4}$$

Here, $x \in \mathbf{R}^{M\times D}$, $W \in \mathbf{R}^{C\times M}$ and $b \in \mathbf{R}^{1\times D}$ and are the inputs, weights and biases of the layer. Note that the same bias is added to each row of Wx.

The input and output layers in an artificial neural net are usually fully connected layers.

2.2.2 The Activation Layer

Fully connected layers are linear transformations on the input data. Activation layers are used to introduce non-linearities in an artificial neural network. The following table gives some of the most common activation layers used. x is the input to the layer.

Sigmoid	$\frac{1}{1+e^{-x}}$
tanh	$tanh(x)$
Rectified Linear Unit (ReLU)	$max(0, x)$
Leaky ReLU	$max(0.01x, x)$
Parametric ReLU	$max(\alpha x, x)$ where α is a hyperparameter

TABLE 2.1: Some Common Activation Functions

2.2.3 The Convolutional Layer

Suppose that our input consists of images. Each image can be represented by a matrix x, where $x \in \mathbf{R}^{H \times W \times C}$. Here, H and W are the height and width of the image and C is the number of color channels (usually 3, corresponding to red, green and blue). A convolutional layer slides a filter across all possible spatial locations of each input image and computes the dot product at each instant. Each filter $f \in \mathbf{R}^{F \times F \times C}$, where F is the height/width of the filter. Mathematically, the output, y of a convolutional layer is given by:

$$y[x,y] = f[x,y] * g[x,y] = \sum_{n1=-\infty}^{\infty} \sum_{n2=-\infty}^{\infty} f[n1,n2] g[x-n1, y-n2] \tag{2.5}$$

where *f(x,y)* is the filter and *g(x,y)* is the input image.

Note that y is a two-dimensional matrix. By using more than one filter in the convolutional layer, we can get several two-dimensional matrices, which can then be stacked on top of one another, giving a three-dimensional matrix.

The reader is referred to [2] for a more detailed explanation on covolutional layers.

2.2.4 The Batch Normalization Layer

The batch normalization layer is used to make each dimension in an input matrix, x, unit gaussian. This is done by using the following the equation:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbf{E}[x^{(k)}]}{\sqrt{\mathbf{Var}[x^{(k)}]}} \tag{2.6}$$

Here $x^{(k)}$ is the *kth* column of the input matrix X, where $X \in \mathbf{R}^{N \times M}$.

The output, y, of the batch normalization layer is given by:

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)} \tag{2.7}$$

where γ and β are learnable parameters of the artificial neural network.

2.2.5 The Pooling Layer

Pooling layers are used to reduce the dimensionality of the data in order to make representations smaller and more manageable. The most common type of pooling, the *max* *pooling*, slices the input image into equal sizes and selects the maximum pixel value in each slice. The pooling layer is usually used after a convolutional layer.

2.2.6 The Recurrent Neural Network

When analyzing text documents, it is important for a neural network to learn long-term dependencies in the text. For example, the gender of a particular object might only be specified once, but the forms that all pronouns and verbs in the resulting discussion of this object take are in conformity with that gender. While tradional neural networks can not learn these dependencies, recurrent neural networks can. Figure 2.2 shows a recurrent neural network Note, that the feedback loop ensures that the recurrent neural

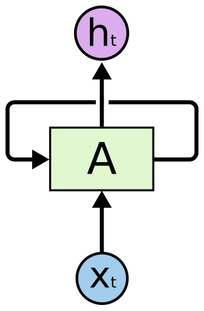


FIGURE 2.2: A RNN Cell
[8]

network retains some of the previous information. A more intuitive diagram can be drawn by *unrolling* the feedback loop, as shown in Figure 2.3. The output, h_t, of a RNN

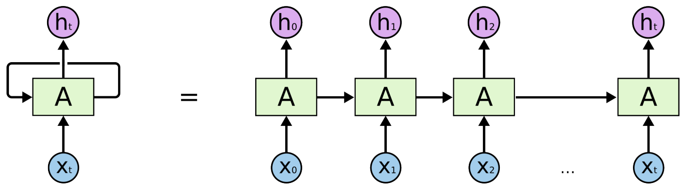


FIGURE 2.3: An Unrolled RNN
[8]

at timestep t is given by:

$$h_t = tanh\left(W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right) \tag{2.8}$$

where W is the weight matrix of the RNN cell.

However, while in theory RNN can handle long-term dependencies, in practice they are not able to do so. Two major reasons behind this are information morphing and vanishing and exploding gradients. The reader is referred to [4] for a more detailed (mathematical) discussion on this topic.

2.2.7 Long Short Term Memory (LSTM)

The Long Short Term Memory (LSTM) was first introduced in [21]. The paper argued that learning to store information, like a recurrent neural network does, is a time-consuming and inefficient process. To solve this problem, it introduced the LSTM cell as shown in Figure 2.4. Where in the RNN the neural network had to learn to store

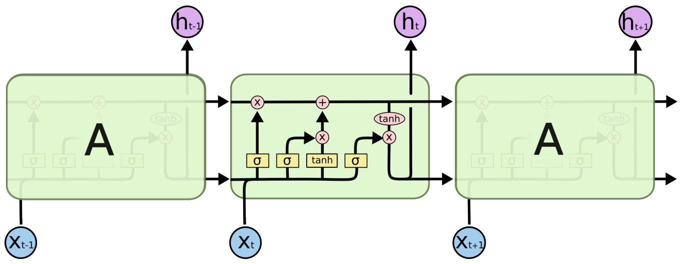


FIGURE 2.4: The LSTM Cell, Unrolled for Three Time Steps [8]

previous information, the LSTM makes this explicit. At each time instant, the LSTM cell computes a so-called *cell state*, in addition to the output, which retains essential information from previous states. At the next time instant, the cell decides which information should be retained from the previous cell state and what new information should be added to it. These decisions are made through a gating mechansim that consists of four gates: input gate, i, forget gate, f, ouput gate, o, and another gate represented by g. The reader is referred to [8] and [4] for a more detailed discussion. The relations for the cell state, c_t, and output, h_t, are given by the following:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{2.9}$$

$$c_t = f \odot c_{t-1} + i \odot g \tag{2.10}$$

$$h_t = o \odot tanh(c_t) \tag{2.11}$$

where W is the weight matrix of the LSTM cell and σ is the sigmoid function.

2.2.8 The Bi-Directional Recurrent Neural Network (BRNN)

A bi-directional RNN combines two RNNs, one moving forward in time and the other moving backwards in time. Figure 2.5 illustrates the concept. A bi-directional LSTM (BLSTM) replaces the RNN cell in BRNN with a LSTM cell.

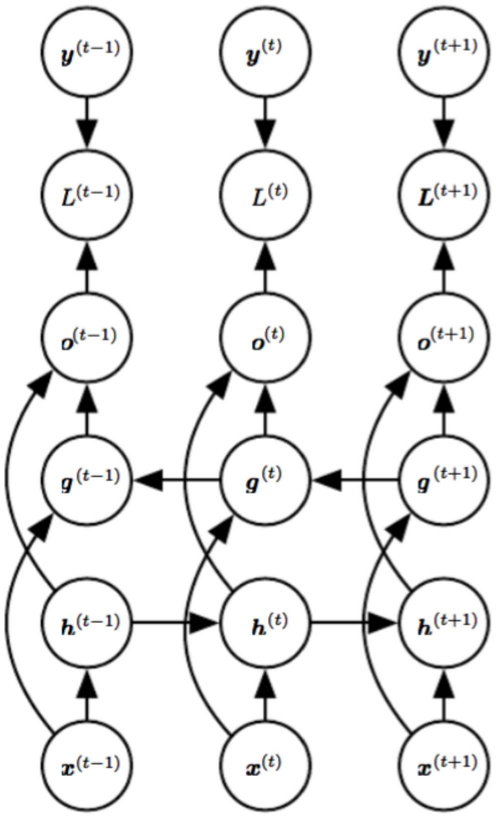


FIGURE 2.5: The Bi-Directional Recurrent Neural Network
[1]

2.2.9 The Multi-Dimensional Recurrent Neural Network (MDRNN)

The multi-dimensional RNN was first introduced in [10]. The core idea behind these networks is to replace the single recurrent connection in standard RNNs with as many connections as there are dimensions in the data. The multi-dimensional LSTM (MDLSTM) replaces the RNN cell in the network with a LSTM one.

2.2.10 Loss Functions

Suppose that we have N examples. We may define the loss, L, of an algorithm as follows:

$$L = \frac{1}{N} \sum_{i=0}^{N-1} L_i(f(x_i, W), y_i) \tag{2.12}$$

Here, $f(x_i, W)$ are the output scores of an artificial neural network with parameters W for an input x_i, while y_i is the desired output of the network. The function L_i computes the error for each x_i. Table 2.2 summarizes some of the commonly-used loss functions. Note that s_j is the score for the *jth* class in $f(x_i, W)$ and s_{yi} is the score of the correct class.

The connectionist temporal classification can be understood in the context of RNN and

L1	$\lvert s_j - s_{y_i} \rvert$
L2	$(s_j - s_{y_i})^2$
Multiclass SVM	$\sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$
Cross Entropy	$-log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$
Connectionist Temporal Classification (CTC)	$-log(\sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X))$

TABLE 2.2: Some Common Loss Functions

LSTM architectures. The term $p_t(a_t|X)$ is the probability distribution of the output at each time instant *t*. The CTC is used for problems where the input and the ouput might not *align* (for e.g. in speech-to-text problems). It computes the loss for all possible *alignments* of the output sequence. The summation in the loss function adds up the individual losses of all valid alignments (see [3] for a more detailed and intuitive discussion).

2.2.11 Regularization

Regularization is a technique used to encourage simpler models. The idea has emanated from the principle of Occam's Razor which states that among competing hypothesis, the simplest one is the best.

Regularization can be achieved by adding an additional term in Equation 2.12 as follows:

$$L = \frac{1}{N} \sum_{i=0}^{N-1} L_i(f(x_i, W), y_i) + \lambda R(W) \tag{2.13}$$

λ is a hyperparameter that controls the amount of regularization. *R(W)*, a function of the weight matrices in the artificial neural network, is a measure of the *complexity* of a model. The more complicated the model is, the higher is its loss. Table 2.3 lists some functions used for regularization.

L2 Regularization	$\sum_k \sum_l W_{k,l}^2$
L1 Regularization	$\sum_k \sum_l \lvert W_{k,l} \rvert$
Elastic Net	$\sum_k \sum_l \beta W_{k,l}^2 + \lvert W_{k,l} \rvert$

TABLE 2.3: Some Common Regularization Functions

2.2.12 Gradient Descent Algorithms

Equation 2.3 gives the stochastic gradient descent algorithm. There are several limitations to this algorithm. The algorithm, for example, does not work well in the presence of *local minma* and *saddle points*, which are very common in high dimension datasets. Many alternatives to this algorithm have, therefore, been proposed. [25] and a blog post by same author [5] review most of these algorithms.

2.3 Optical Character Recognition (OCR) Systems

Optical character recognition typically involves five steps: preprocessing, segmentation, feature extraction, classification and recognition and post-processing. In this section, we survey the OCR literature with particular emphasis on recognizing handwritten documents in Urdu.

2.3.1 Preprocessing

Preprocessing involves a number of steps [16] that help improve the accuracy of the stages later in the algorithm (see [15]). It does this by removing noise and unnecessary details from an image.

One important step in preprocessing is binarization, i.e. converting a pixel image to a binary image. In a binary image, all pixel values take on only two possible values: 0 or 1. This may be done through Otsu's method [24]. Otsu's method takes in account the fact that an image contains two classes of pixels and calculates the optimum threshold that separates these classes.

Another technique in preprocessing is smoothing. This tries to construct an approximating function to the image that captures all the important patterns in it (see [6] for more details).

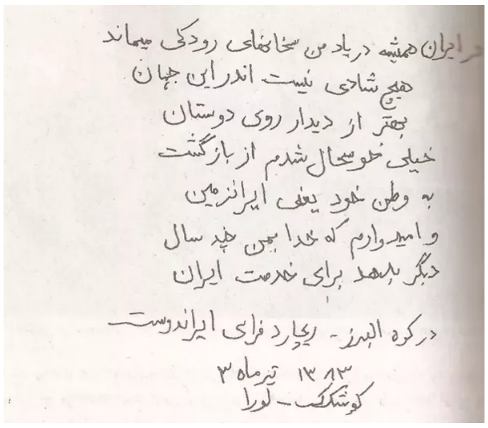
Thinning is another technique that may used in preprocessing. It involves finding the medial axis of an image (see [6] and [16]).

One particularly important stage in preprocessing is skew-correction. While scanning, documents might be titled a little, which introduces a *skew* in the corresponding image. [19] includes an algorithm for skew-correction. [14] presents a novel approach using the probabilistic hough transform for skew detection and correction that has been tested on Latin and Arabic scripts.

Other techniques involved in preprocessing include noise removal, background elimination, removal of black boundaries and extra white spaces, gray-scale normalization and size-normalization (see [16], [6], [15]).

2.3.2 Segmentation

Consider Figure 2.6. In order to aid our recognizer, we can divide or *segment* this figure into smaller pieces. Images can either be segmented into lines, words or ligatures [16]. Line segmentation can be accomplished by the method outlined in [19]. The algorithm presented therein sums all the pixel values in each row. Assuming that 0 corresponds to a white pixel, a row summing up to zero would indicate a white line. This information can be used to find the vertical coordinates at which each line starts and ends. The image can then be cropped at these coordinates. For handwritten documents word and ligature segmentation is generally harder. [16] reviews different segmenation techniques.



در ایران همیشه در یاد من سخنهای رودکی میماند
هیچ شادی نیست اندر این جهان
بهتر از دیدار روی دوستان
خیلی خوشحال شدم از بازگشت
به وطن خود یعنی ایرانزمین
و امیدوارم که خدا بمن چند سال
دیگر بدهد برای خدمت ایران

در کوه البرز - ریچارد فرای ایراندوست
۱۳۸۳ تیرماه ۳
کوشکک - لورا

FIGURE 2.6: A Sample Document Written In Urdu

Recent research has been focussing on using artificial neural networks for page segmentation. [7] uses an artificial neural network to segment historical document images and shows that a convolutional neural net with only one convolutional layer can outperform tradional state-of-the-art algorithms.

Preprocessing techniques may be applied on each segment separately to further reduce noise in the images.

2.3.3 Feature Extraction

Feature extraction is another technique that, like segmentation, aids the recognition process. Not all information contained in an image of a document is essential for recognition. Some information might just be noise. Feature extraction or engineering entails extracting only the information that is required for accurate recogniton. Feature extraction also reduces the amount of data that is fed into a recognizer, which helps speed up the algorithm.

[16] reviews different feature extraction approaches. These include the computation of curvature, slope, end-points axes ratio, the length variations of strokes and shape context. [9] uses and compares the discrete cosine transform and discrete wavelet transform methods for feature extraction. In doing so, it observes that the discrete cosine transform gives superior features compared to the discrete wavelet transform. [17] extracts zoning features for Urdu text-line recognition.

However, more recent research uses artificial neural networks for feature extraction. [12] uses a hybrid CNN-RNN architecture, wherein the convolutional neural network extracts features for the recurrent neural network. [11] proposes an autoencoder based on deep learning. The autoencoder essentially encodes the important information in a document image and leads to significant improvements in systems trying to recongnize handwritten digits in Arabic.

Due to recent breakthroughs in and increased effectiveness of deep learning, some systems do not include a separate feauture extraction stage. [15] does not compute explicit features and instead directly moves on to the recognition stage after preprocessing. The same architecture is used in [13] for recognition of printed Pashto documents. [27] presents an attention-based model for paragraph recognition without using any explicit feature extraction stage.

2.3.4 Classification and Recognition

The classification and recognition stage identifies all words in a particular image using the outputs of the previous stages. Tradionally, Hidden Markov Models (HMM) were used for this purpose. The reader is referred to [22] for a review of sytems using HMM. However, there is growing trend of using artificial neural nets in this stage. As such, in this section, we will give an overview of different artificial neural network architectures used for this purpose.

The problem of recognizing handwritten documents is a sequential learning one. The algorithm must start from one end of a line and move towards the other end. The algorithm should also learn to use the context of and the words that previously appeared in the document when trying to identify a particular word. The Recurrent Neural Network (RNN) and its variants, including the Long Short Term Memory (LSTM), the Bi-Dimensional LSTM (BLSTM) and the Multi-Dimensional LSTM (MDLSTM) are thus most suitable for this task.

Depending on the segmentation step, we will need to identify either separate characters or sentences or paragraphs.

For recognition of individual characters, the algorithm might not have access to the previous text of the document. As such, it may not be possible for it to learn to understand the context of a particular document and use it to its advantage. Therefore for character recognition simpler artificial neural networks will suffice. [23] proposes an architecture comprising of two convolutional layers, two pooling layers and a fully connected layer in addition to the input and output layers. A RELU activation layer is used after each of the convolutional and fully-connected layer. At the output, a softmax classifier is used. The neural net was trained through the stochastic gradient descent algorithm with momentum and L2 regularization. [26] uses a two layer neural network for recognizing Arabic characters.

[13] proposes two different architectures for recognizing printed Pashto sentences: a two-layer BLSTM and a three-layer MDLSTM model. In both models, the connectionist temporal classification is used as the loss function. The paper concludes that the MDLSTM model outperforms the BLSTM model for Pashto sentences. The same MDLSTM arhcitecture is used in [15]. [12] proposes a CNN-RNN hybrid model for recognizing printed Urdu documents. The documents are first segmented into lines. [19] also presents a 1-D BLSTM classifier for recognizing Urdu sentences.

[27] proposes an attention-based model for English documents that can recognize paragraphs as a whole. In this case, no explicit segmentation is required.

2.3.5 Postprocessing

Once the text in an image has been recognized and a word-processing document constructed, additional steps such as spell-checking and grammar corrections can be carried out to improve the accuracy of the recognition system.

2.3.6 Datasets

To train an artificial neural network, a dataset is required. Only a few datasets consisting of Urdu handwriting documents and their ground labels (the X and y matrices) exit. [16] lists some of these datasets.

[20] and [18] present datasets for Arabic handwritten documents. [28] introduces a dataset for Urdu text printed documents. The methods outlined therein can be used to construct a dataset for Urdu handwritten documents.

Chapter 3

Proposed Methodology

We propose an OCR pipeline comprising of 5 stages: preprocessing, segmentation, feature extraction, classification and recognition and postprocessing. Figure 3.1 shows a high-level block diagram of our proposed system.

3.1 Preprocessing

This stage will include the following:

- Binarization
- Skew correction
- Cropping of extra white spaces in the image

3.2 Segmentation

As recent advancements in the field of deep learning have made artificial neural nets quite powerful, we propose that a text-line segmentation will be able to provide sufficient accuracy. This can be carried out using a horizontal projection method based on the algorithm outlined in [19].

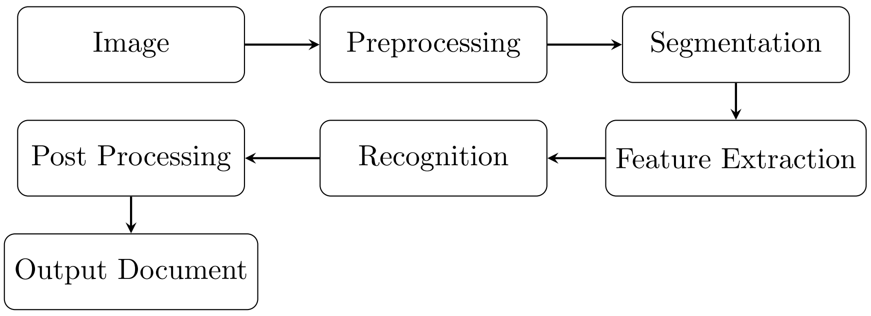


FIGURE 3.1: A High-Level Block Diagram of the Proposed System

3.3 Feature Extraction

While more recent research has avoided computing hand-crafted features, thanks to recent breakthroughs in deep learning, we propose that such features can considerably increase the accuracy of a recognition system. Hence, our optical character recognition system will make use of feature extraction.

3.4 Classification and Recogniton

The problem of recognizing a text line is a sequential learning one. The Recurrent Neural Network and its variants, including the Long Short Term Memory (LSTM), the Bi-Dimensional LSTM and Multi-Dimensional LSTM with a connectionist temporal classification loss function are therefore best suited for this problem. We propose an artificial neural network based on the architecture used in [15].

3.5 Postprocessing

This step will include spell-checking and grammar correction and will help increase the accuracy of our recognition system.

Chapter 4

Time Schedule and Deliverables

Month	**Activity**	
June	Plan a method to collect dataset and decide on its format	Literature review of recognition systems
July	Collect dataset	Experiment with different architectures for classification and recognition
August	Add labels to dataset	Work on feature extraction and computation
September	Complete preprocessing and segmentation	Test extracted features on different architectures
October	Continue on feature extraction methods	Test extracted features on different architectures
November	Continue on feature extraction methods	Test extracted features on different architectures
December	Finalize feature extraction methods	Test extracted features on different architectures
January	Improve architecture for classification and recognition	
February	Finalize architecture for classification and recognition	
March	Carry out postprocessing steps	

Table 4.1: Time Schedule

Month	**Deliverable**
September	Dataset Collection
October	Preprocessing and Segmentation
December	Feature Extraction
February	Classification and Recognition
March	Postprocessing

Table 4.2: Deliverables

References

[1] All of Recurrent Neural Networks. https://medium.com/@jianqiangma/all-about-recurrent-neural-networks-9e5ae2936f6e, Last accessed on June 07, 2018.

[2] Convolutional Neural Networks. http://cs231n.github.io/convolutional-networks/, Last accessed on June 07, 2018.

[3] Sequence Modeling With CTC. https://distill.pub/2017/ctc/, Last accessed on June 07, 2018.

[4] Written Memories: Understanding, Deriving and Extending the LSTM. https://r2rt.com/written-memories-understanding-deriving-and-extending-the-lstm.html, Last accessed on June 07, 2018.

[5] An overview of gradient descent optimization algorithms. http://ruder.io/optimizing-gradient-descent/, Last accessed on June 07, 2018.

[6] Yasser Alginahi. *Preprocessing Techniques in Character Recognition.* 2010.

[7] Kai Chen and Mathias Seuret. *Convolutional Neural Networks for Page Segmentation of Historical Document Images.* 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017.

[8] Colah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Last accessed on June 07, 2018.

[9] A. Lawgali et al. *Handwritten Arabic Character Recognition: Which Feature Extraction Method.* International Journal of Advanced Science and Technology, 34. pp. 1-8, 2011.

[10] Alex Graves et al. *Multi-Dimensional Recurrent Neural Networks.* ICANN 2007: Artificial Neural Networks ICANN pp 549-558, 2013.

[11] Mohamed Loey et al. *Deep Learning Autoencoder Approach for Handwritten Arabic Digits Recognition.* arXiv:1706.06720, 2017.

[12] Mohit Jain et al. *Unconstrained OCR for Urdu using Deep CNN-RNN Hybrid Networks.* The 4th Asian Conference on Pattern Recognition (ACPR), 2017.

[13] Riaz Ahmad et al. *KPTI: Katibs Pashto Text Imagebase and Deep Learning Benchmark.* 15th International Conference on Frontiers in Handwriting Recognition, 2016.

[14] Riaz Ahmad et al. *A Novel Skew Detection and Correction Approach for Scanned Documents.* DAS 2016, 12th Intl IAPR Workshop on Document Analysis SystemsAt: Santorini, Greece, 2016.

[15] Riaz Ahmad et al. *KHATT: a Deep Learning Benchmark on Arabic Script.* Document Analysis and Recognition (ICDAR), IAPR International Conference, 2017.

[16] S. Naz et al. *The optical character recognition of Urdu-like cursive scripts.* Pattern Recognition, Volume 47, Issue 3, Pages 1229-1248, 2014.

[17] S. Naz et al. *Zoning features and 2D-LSTM for Urdu text-line recognition.* Procedia Computer Science Volume 96, Pages 16-22, 2016.

[18] Saad Bin Ahmed et al. *A Comprehensive Isolated Arabic Character Database for Handwriting OCR Research.* Proceedings of the 10th International Workshop on Frontiers, 2006.

[19] Saad Bin Ahmed et al. *Handwritten Urdu Character Recognition using 1-Dimensional BLSTM Classifier.* Neural Computing and Applications, 2017.

[20] S.Al-Maadeed et al. *A dataset base for Arabic handwritten text recognition research.* Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition 6-8, 2002.

[21] Sepp Hochretier and Jürgen Schmidhuber. *Long Short Term Memory.* Neural Computation, 1997.

[22] L.M. Lorigo and V. Govindaraju. *Offline Arabic handwriting recognition: a survey.* IEEE Transactions on Pattern Recognition, Vol. 28, Issue 5, 2006.

[23] L.M. Lorigo and V. Govindaraju. *Arabic Handwritten Characters Recognition using Convolutional Neural Network.* WSEAS Transactions on Computer Research, 2017.

[24] N. Otsu. *A threshold selection method from gray-level histograms.* IEEE Transactions On Systems, Man, AND Cybernetics, Vol. SMC-9, No.1, 1979.

[25] Sebastian Ruder. *An overview of gradient descent optimization algorithms.* 2017.

[26] Ahmed Sahlol and Cheng Suen. *A Novel Method for the Recognition of Isolated Handwritten Arabic Characters.* arXiv:1402.6650, 2014.

[27] J. Louradou T. Bluche and R. Messina. *Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention.* arXiv:1604.03286, 2016.

[28] Qurat ul-Ain Akram et al. *A Comprehensive Image Dataset of Urdu Nastalique Document Images.* Proceedings of Conference on Language and Technology, 2016.