LITERATURE REVIEW

**OCR, Feature extraction and related work**

OCR consists of many phases such as Scanning of image, Pre-processing, Segmentation, Feature Extraction, Classifications and Recognition, Post Processing. The task of pre-processing relates to the removal of noise and variation in the image [1]. In scanning step, the image is acquired. The quality of image depends highly on the scanner being sed. In practical applications, the scanned images are not perfect there may be some noise due to some unnecessary details in the image which can cause a disruption in the detection of the characters in the image.[19] Pre-processing involves removal of noise (applying filters like Gaussian filter, Gabor filter etc.) and proper conversion of image like a coloured image can be converted into grayscale or binary image for further processing of image.

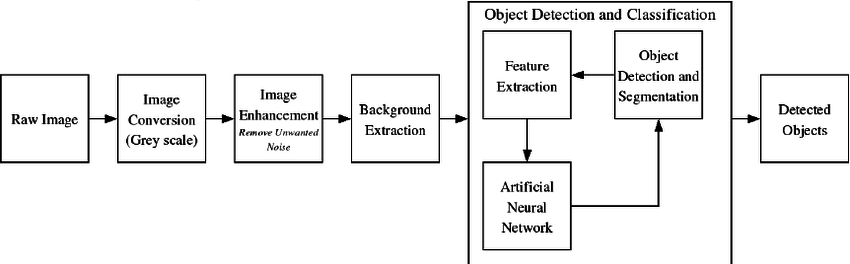
Feature extraction involves recognizing the feature required.[20] One of the most important phases in successfully achieving character recognition is the task of feature extraction. Feature extraction stage identifies and extracts various attributes from characters that help distinctly and uniquely distinguish different characters. A number of different feature extraction methods have been proposed in literature in accordance with different character representations. For example, different sets of features have been defined to best represent character shapes, boundaries, their skeletons and strokes etc. Trier et al. [2] comprehensively describe different types of features and methods for character recognition task. Among these methods, there are statistical feature extractors and structural feature extractors. Statistical features consider the statistical distribution of pixel values. Major statistical features used for handwritten character recognition task include zoning, projections, profiles, and crossings etc. Structural features consider the geometry and topology of character samples such as number of loops, end points, junction points, aspect ratio, type of strokes and their directions etc. Some feature extraction methods are based on different transformations such as those based on Fourier transform, wavelet transform, central moments, and Zernike moments etc. In [3], the authors describe a zoning-based feature extractor to recognize handwritten numerals of Indian Kannada script. Authors in [4] recognize handwritten numerals using Fourier descriptors and neural network. In [5], the authors recognize Chinese handwritten characters using gradient and wavelet-based features. In [6], the authors extract moment-based features in order to recognize handwritten Arabic letters. They use genetic algorithm for feature selection and use SVM to evaluate the classification error for the chosen feature subset.

Fig 1: Feature Extraction Steps in Image Processing

Instead of focusing on feature vector based on a single representation of a character, it is a trend now of combining different types of features extracted from different representations of the same character. The advantage of combining, and harnessing, such different kinds of features is that it can offer wider range of identification clues to help improve the accuracy of recognition. For example, Hettes et al. [7] combine different statistical and structural features for recognition of handwritten characters. They construct a 124-variable feature vector comprising following seven families of features: 1) intersection of the character with horizontal and vertical straight lines, 2) invariant moments, 3) holes and concave arcs, 4) extremes, 5) end points and junction points 6) profiles, and 7) projections. Aurora et al. [8] combine different feature extraction techniques such as intersection-based features, shadow features, chain code and curve fitting features for Indian Devanagari language script. Kimura et al. [9] propose a genetic algorithm-based strategy for finding a suitable combination of features from a large pool of features with the objective criteria to minimize the classification error.

**Pattern classification, SVM and related work**

The second most important component in successfully achieving handwritten character recognition is the pattern classification stage. This stage will assign an unknown character sample to one of possible classes by utilizing the information of feature extraction stage. Different types of classifiers can be built based on the nature and type of data samples and the extracted features. [26]

Classifiers used for character recognition problem include k-nearest neighbour classifier, hidden Markov model (HMM), support vector machine (SVM), and artificial neural network (ANN) etc. Jain et al. [10] give a review of statistical pattern recognition techniques. In [11], Pal and Singh train neural network to recognize uppercase handwritten characters based on Fourier descriptors of character boundaries as features. In [12], recognition of handwritten alphabets using neural network and zoning based diagonal features is addressed. In [13], Shubhangi and Hiremath recognize English handwritten characters and digits by extracting structural micro features for SVM classifier. Nasien et al. [14] also use SVM classifier to recognize handwritten alphabets by employing Freeman Chain codes as the features. In [15], Train et al. recognize accented handwritten French characters based on a combination of structural and moment features for SVM classifier. In [16], Liu and Nakagawa give a review of learning methods for nearest neighbour classifiers. [17] and [18] build HMM to recognize, respectively, offline handwritten Chinese characters and online English characters.

**Bengali Character Recognition and Related work**

Analysis of the structural features of the letterforms is often successfully used in handwritten character recognition. In 2006, Chowdhury et al. [20] developed a method for recognizing Bengali handwritten numerals where the characters are modelled as water reservoirs. In 2011, Mandal [21] developed a Bengali handwritten character recognition scheme based on the analysis of gradient features. Recently, Das et al. developed a method for recognizing handwritten Bengali numerals using mathematical morphology [22].

Input: Image file of a Bengali handwritten character and Output will be the correct classification of the character. The table below shows the Bengali characters in printed form along with the phonetics in English. Please note that the 50 basic characters can be further combined to create several complex alphabets with mixed sound but this is out of scope of the current project.[25]

Vowels:

1 অ aw 6 ঊ uuu 11 ঔ ou

2 আ aaa 7 ঋ rhi

3 ই e 8 এ ey

4 ঈ eee 9 ঐ oi

5 উ u 10 ও o

Consonants:

12 ক ka 17 চ cha 22 ট taw

13 খ kha 18 ছ chha 23 ঠ thaw

14 গ ga 19 জ ja 24 ড daw

15 ঘ gha 20 ঝ jha 25 ঢ dhaw

16 ঙ nga 21 ঞ nya 26 ণ naw(1)

27 ত ta 32 প paw 37 য jaw

28 থ tha 33 ফ phaw 38 র raw

29 দ da 34 ব baw 39 ল law

30 ধ dha 35 ভ bhaw 40 শ saw(1)

31 ন naw(2) 36 ম maw 41 ষ saw(2)

42 স saw(3) 47 ৎ khando-taw

43 হ haw 48 ◌্ onussar

44 ড় dra 49 : bisargo

45 ঢ় dhra 50 ◌ঁ chandrabindu

46 য় ya

Ray and Chatterjee [23] did the first significant work in Bengali HCR. After that, many more researchers tried several other methods for improving the performance of Bangla Handwritten Character Recognition (HCR) as evident in [24]. Hasnat et al. [32], proposed an HCR capable of classifying both printed and handwritten characters by applying Discrete Cosine Transform (DCT) over the input image and Hidden Markov Model (HMM) for character classification. Wen et al. [33] proposed a Bangla numerals recognition method using Principal Component Analysis (PCA) and Support Vector Machines. Liu and Suen [34], proposed a method of identifying both Farsi and Bangla Numerals. In Hassan and Khan [35], K-NN algorithm was used where features were extracted using local binary patterns. Das et al. [36], proposed a feature set representation for Bangla handwritten alphabets recognition which was a combination of 8 distance features, 24 shadow features, 84 quad trees based longest run features and 16 centroid features. Their accuracy was 85.40% on a 50-character class dataset. The above-mentioned methods however used many handcrafted features extracted for small dataset which turned out to be unsuitable for deploying solutions.[27]

**CNN and Related Work**

The CNN structure was ﬁrst time proposed by Fukushima in 1980 [37]. However, it has not been widely used because the training algorithm was not easy to use. In 1990s, LeCun et al. applied a gradient-based learning algorithm to CNN and obtained successful results [38]. After that, researchers further improved CNN and reported good results in pattern recognition. Recently, Cirean et al. applied multi-column CNNs to recognize digits, alpha-numerals, trafﬁc signs, and the other object class [39]. They reported excellent results and surpassed conventional best records on many benchmark databases, including MNIST [38] handwritten digits database and CIFAR-10 [40].

Why CNN?

CNNs can be thought of automatic feature extractors from the image. While if I use an algorithm with pixel vector, I lose a lot of spatial interaction between pixels, a CNN effectively uses adjacent pixel information to effectively down sample the image first by convolution and then uses a prediction layer at the end.[41]

Ruggedness to shifts and distortion in the image: Detection using CNN is rugged to distortions such as change in shape due to camera lens, different lighting conditions, different poses, presence of partial occlusions, horizontal and vertical shifts, etc. However, CNNs are shift invariant since the same weight configuration is used across space. In theory, we also can achieve shift invalidates using fully connected layers.[28] But the outcome of training in this case is multiple units with identical weight patterns at different locations of the input. To learn these weight configurations, a large number of training instances would be required to cover the space of possible variations.[46]

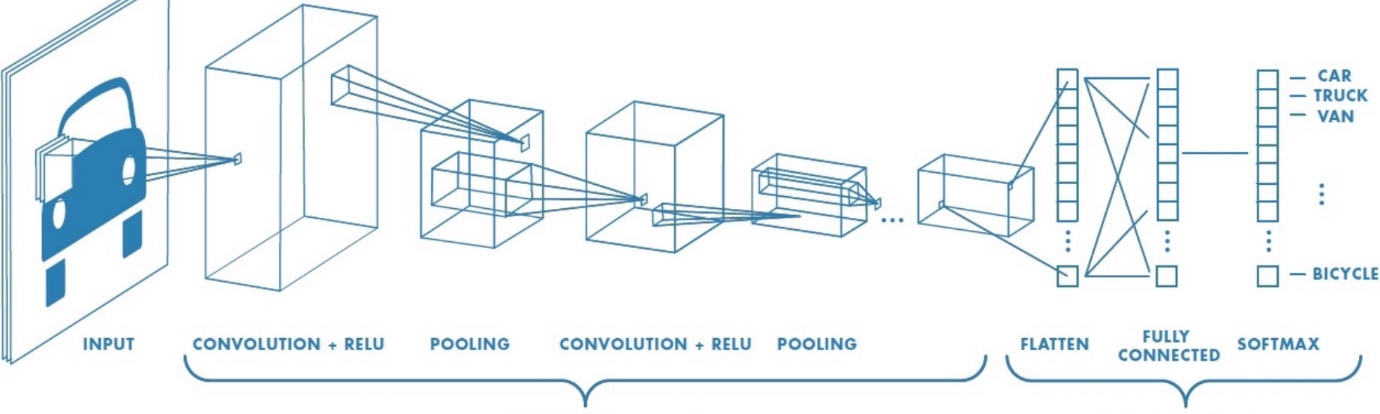
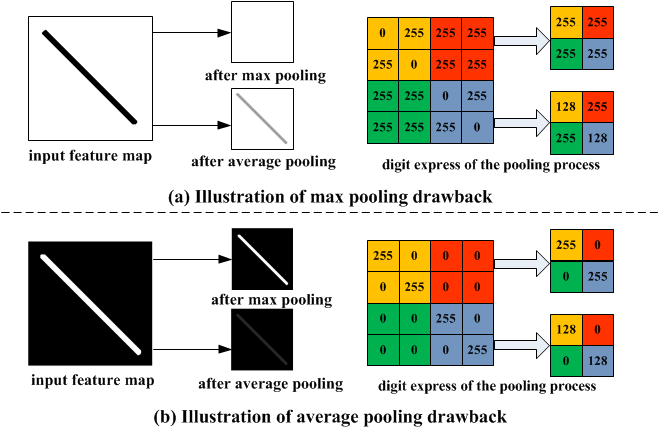
Fewer Memory requirements: In this same hypothetical case where we use a fully connected layer to extract the features, the input image of size 32x32 and a hidden layer having 1000 features will require an order of 106 coefficients, a huge memory requirement. In the convolutional layer, the same coefficients are used across different locations in the space, so the memory requirement is drastically reduced.[47]

Fig 02: Basic CNN Layer Working Diagram

In the simple neural networks, each neuron was fully connected to each of the neurons in the subsequent layer. More concretely, each neuron in the hidden layer computed a function that depended on the values of every node in the input layer. In visual recognition, however, it is often advantageous to exploit local substructure within the image. [29] For example, pixels that are close together in the image (e.g., adjacent pixels) tend to be strongly correlated while pixels that are far apart in the image tend to be weakly correlated or uncorrelated. Not surprisingly then, many standard feature representations used in computer vision problems are based upon local features within the image [30]. Ianthe CNN architecture, we capture this local substructure within the image by constraining each neuron to depend only on a spatially local subset of the variables in the previous layer. For example, if the input to the CNN is a 32-by-32 image patch, a neuron in the first hidden layer might only depend on an 8-by-8 sub window within the overall 32-by-32 window. These of nodes in the input layer that affect the activation of a neuron is referred to as the neuron’s receptive field. Intuitively, this is the part of the image that the neuron sees.

“Thus, in a CNN, individual neurons generally have a local receptive field rather than a global receptive field. In terms of network architecture, this translates to a sparser set of edges since adjacent layers are not always fully connected.[57] The second feature that distinguishes CNNs from simple neural networks is the fact that the edge weights in the network are shared across different neurons in the hidden layers. Recall that each neuron in the network first computes a weighted linear combination of its inputs.[31] We can view this process as evaluating a linear filter over the input values. In this context, sharing the weights across multiple neurons in a hidden layer translates to evaluating haemofilter over multiple sub windows of the input image. In this regard, we can view the CNN as effectively learning a set of filters= {Fiji= 1, n}, each of which is applied to all of the sub windows within the input image. Using the same set of filters over the entire image forces the network to learn a general encoding or representation of the underlying data. Constraining the weights to be equal across different neurons also has a regularizing effect on the CNN; in turn, this allows the network to generalize bettering many visual recognition settings. Another benefit of weight sharing is the fact that it substantially reduces the number of free parameters in the CNN, making it markedly easier and more efficient to train.[49] As a final note, evaluating a filter over each window in the input image amounts to performing a convolution of the image with the filter (deconvolved image with the filter).

Thus, in the convolutional step of the CNN, we take the input image and convolve it with each filter into obtain the convolutional response map. The final distinguishing component in a CNN is the presence of subsampling or pooling layers. The goal here is twofold: reduce the dimensionality of the convolutional responses and confer a small degree of translational invariance into the model.



The standard approaches through spatial pooling [50]. In spatial pooling, the convolutional response map is first divided into a set of m×n blocks (generally disjoint). We then evaluate a pooling function over the responses in each block. This process yields a smaller response map with dimension (one response for each block). In the case of max pooling, the response for each block taken to be the maximum value over the block responses, and in the case of average pooling, the response is taken to be the average value of the block responses. [51] An example of average pooling. In this case, the convolutional response map is a4-by-4 grid and we average pool over four 2-by-2 blocks arranged in a 2-by-2 grid. The pooled response is taken to be the average of the values in the block. After applying this average pooling procedure, we arrive at a final 2-by-2 pooled response map. Compared to the original 4-by-4 convolutional response map, this represents significant reduction in dimensionality of the response map. In a typical CNN, we have multiple layers, alternating between convolution and pooling. [52] For example, we can stack another convolution-pooling layer on top of the outputs of the first convolution-pooling layer. In this case, we simply treat the outputs of the first setoff convolution-pooling layers as the input to the second set of layers. In this way, we can construct a multi-layered or deep architecture. Intuitively, the low-level convolutional filters, such as those in the first convolutional layer, can be thought of as providing a low-level encoding of the input data. In the case of image data, these low-level filters may consist of simple edge filters.[56] As we move to higher layers in the neural network, the model begins to learn more and more complicated structures. By using multiple layers and large numbers of filters, the CNN architecture can thus provide vast amounts of representational power.

To train a CNN, we can use the standard technique of error backpropagation used to train neural networks [53]. Convolutional neural networks have enjoyed a series of successes in many problems related to text classification such as handwriting recognition visual object recognition and character recognition Coupled with the rapid advancements in distributed and GPU (graphics processing units) computation, it is now possible to train much larger and more powerful CNNs that achieve state-of-the-art performance on standard benchmarks [54]. Thus, by leveraging the representational capacity contained within these networks in conjunction with the robustness of features derived from unsupervised algorithms, were able to construct simple, but powerful and robust, systems for both text detection and recognition.

Apart from there also present several Bangla Handwritten Character Recognition and had achieved pretty good success. Halima Begum et al, “Recognition of Handwritten Bangla Characters using Gabor Filter and Artificial Neural Network” [42] works with own dataset that was collected from 95 volunteers and their proposed model achieved without feature extraction and with feature extraction around 68.9% and 79:4% of recognition rate respectively. “Recognition of Handwritten Bangla Basic Character and Digit Using Convex Hull Basic Feature” [43] accuracy for Bangla character 76.86% and Bangla numeral 99.45%. “Bangla Handwritten Character Recognition using Convolutional Neural Network” [44] achieved 85.36% test accuracy using their own dataset. In “Handwritten Bangla Basic and Compound character recognition using MLP and SVM classifier” [45] handwritten Bangla character recognition with MLP and SVM has been proposed and they achieved around 79.73% and 80.9% of recognition rate, respectively. Using these robust and highly-accurate components renders it possible to obtain full end-to-end results using only the simplest of post-processing techniques.[55]