TEXT DETECTION AND CHARACTER RECOGNITION IN IMAGES WITH NEURAL NETWORKS

A Thesis submitted towards the partial fulfilment of requirement for the award of degree of

MASTER OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION

Submitted By

Akhilesh Thakur

Roll No. 801661002

Under the supervision of

Dr. Vinay Kumar

Assistant Professor, ECED

Thapar University, Patiala

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY

(A Deemed to be University)

PATIALA – 147004, PUNJAB, INDIA

JULY-2018

DECLARATION

I, **Akhilesh Thakur**, hereby declare that the thesis entitled **"Text Detection And Character Recognition in Images With Neural Networks"** is an authentic record of my own work carried out towards the partial fulfilment for the award of degree of Master of Engineering in Electronics and Communication at Thapar Institute of Engineering & Technology (Deemed to be University), Patiala, under the supervision of **Dr. Vinay Kumar**, Assistant Professor, Electronics and Communication Engineering Department.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of any other degree.

Date: 17/08/18

Place: T.U., PATIALA

Akhilesh Thakur

Roll No.:801661002

This is to certify that the above statement made by the student is correct to the best of my knowledge and belief.

Date: 17/8/18

Dr. Vinay Kumar

Assistant Professor

ECED, TU, Patiala

ACKNOWLEDGEMENT

With deep sense of gratitude, I express my sincere thanks and utmost regards to my esteemed and worthy supervisor **Dr. Vinay Kumar**, Assistant Professor, Department of Electronics and Communication Engineering, Thapar University, Patiala for his valuable guidance in carrying out this work under his effective supervision, encouragement, enlightenment and cooperation. It has been a great honor to work under him.

I am also thankful to **Dr. Alpana Agarwal**, Associate Professor and Head, Electronics and Communication Engineering Department, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala, for providing us with adequate infrastructure in carrying the research work.

I am also thankful to **Dr. Amit Mishra**, Assistant Professor and P.G. Coordinator, Electronics and Communication Engineering Department and **Dr. Hem Dutt Joshi**, Assistant Professor, Electronics and Communication Engineering Department, for being a source of inspiration for me during my research work.

I would like to thank my parents who provide me with every necessity, I am thankful to my parents from the core of my heart for their encouragement and constant support that keeps me motivated to carry this work.

I would also like to thank the entire faculty and staff members of Electronics and Communication Engineering Department for their unyielding encouragement.

Most importantly, I express my gratitude to the almighty God for giving me inner peace and strength to undertake this research work.

Akhilesh Thakur

Roll No.: 801661002

ABSTRACT

This dissertation presents a Deep Learning approach for recognizing the handwritten Indian Devanagari characters from images. With the advancement in the technology and massive computational power of the GPUs, it has become possible to achieve human-level intelligence in machines to some extent. Devanagari script is written from left to right and a horizontal line at the top, which is known as the Shirorekha, joins all the characters in a word together. The Devanagari script is a constituent of Sanskrit script, which is used in other languages in India such as Nepali, Gujarati, and Marathi. Artificial Neural Networks are designed based on the biological neural networks in humans. Although researchers have already achieved a lot of success in English character recognition but nowadays Devanagari character recognition task has also gained the attention of many researchers. In this dissertation, Convolutional Neural Network (CNN) is used for feature extraction and offline Devanagari handwritten character recognition. Convolutional Neural Networks can be trained to learn the complex patterns from the images. Handwritten character recognition task is somewhat difficult due to the different writing styles of each person. The features of the input image samples are extracted with the help of convolutional and pooling layers. Softmax classifier is used to generate the class scores for final classification. The error i.e. the difference between the ground truth label and the predicted label is computed with a categorical cross-entropy loss function. The proposed approach for character recognition achieves an accuracy of 98.74% on handwritten Devanagari character dataset.

Next, we prepared a database of Devanagari words and their corresponding labels. We have segmented the Devanagari words of our database into characters. Then, we used our CNN model trained on the handwritten Devanagari characters to generate predictions on the segmented Devanagari characters of our database. The CNN output is then used as an input sequence for the Language Transliteration system and the target sequence are the corresponding labels of our Devanagari words. RNN based Encoder-Decoder model is used for the Language Transliteration task. The performance of the Language Transliteration system is evaluated with different RNNs viz. SimpleRNN, LSTM and GRU.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF ACRONYMS

ANN	Artificial Neural Network
BLSTM	Bidirectional Long Short Term Memory
BRNN	Bidirectional Recurrent Neural Network
CCA	Canonical Correction Analysis
CCD	Charge-Coupled Device
CE	Contrast Enhancement
CFBPN	Cascade-Forward Backpropagation Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
CTC	Connectionist Temporal Classification
DBLSTM	Deep Bidirectional LSTM
DCNN	Deep Convolutional Neural Network
EBPN	Elman Backpropagation Network
ELM	Extreme Learning Machine
ER	Extremal Region
FBP	Filtered Back Projection
FFBPN	Feed-Forward Backpropagation Network
FFNN	Feed-Forward Neural Network
GPU	Graphics Processing Unit
GRBF	Generalized Radial Basis Function
GRU	Gated Recurrent Unit
HCR	Handwritten Character Recognition
HMM	Hidden Markov Model

HOG	Histogram of Oriented Gradient
HWAI	Handwritten Address Interpretation
LF-MMI	Lattice-Free Maximum Mutual Information
LSTM	Long Short Term Memory
MAP	Mean Average Precision
MICR	Magnetic Ink Character Recognition
MLP	Multi-Layer Perceptron
MLRA	Multiple Linear Regression Analysis
MQDF	Modified Quadratic Discriminant Function
MSER	Maximally Stable Extremal Region
NLP	Natural Language Processing
NMT	Neural Machine Translation
OCR	Optical Character Recognition
PCA	Principle Component Analysis
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

CHAPTER-1
INTRODUCTION

1.1 Overview

Due to the exponentially expanding demand of the documents to be processed, the need of character recognition systems has gained huge importance. The area of handwriting recognition has gained attention of many researchers. Handwriting recognition is an active area of research, which found its applications in bank check processing, signature verification, vehicle license plate recognition etc. It is a challenging area of research as handwriting varies from person to person and even a same person can inscribe same character in a different shape at different time with different mood.

The characters have different size, orientations, thickness, and intensity that depends upon the conditions the digital cameras captures the characters all of which makes handwriting recognition a complex and a challenging task. Artificial Neural Networks (ANNs) are employed for the character recognition task. ANNs are developed based upon the idea of human biological nervous system where the neurons passes information from and to the brain. ANNs are trained to learn the features for the applications like pattern recognition and for classifying data. ANNs mainly have three types of layers viz. input layer, hidden layer, and output layer. There can be any number of hidden layers and more the number of hidden layers; deeper will be the neural network. The next layer in the neural network learns from its previous layer. Deeper neural networks with large amount of data can even take days of training for learning the features. Neural networks can process complicated information from a large dataset, which is even complex for the human's eye.

With the advancement in technology, neural networks are employed in various applications such as in automated driverless vehicles, recognizing words in spoken language, recognizing cursive handwriting, understanding human emotions through facial expressions etc. Such systems have been developed that can decode the human's mind; the basis for such systems are convolutional neural networks (CNNs). However, research in these fields are still on going for the improvement in these sectors.

Digitalization of the modern world has made all the data to be stored in the digital format. Manual processing and storing of such a huge amount of data is quite troublesome. With the help of neural networks, such tasks are easily done; saving time, cost and the labor.

1.2 Image

An image can be defined as a 2D-signal with a mathematical function f (x, y); where x and are the horizontal and vertical co-ordinates of an image. The function f (x, y) gives the pixel value of an image at a particular point. These pixel values ranges from 0-255 for an RGB image. A Binary image has pixel value of either zero or one. Image is a 2D array of numbers. Human beings processes the information with the help of their brain without knowing the way the brain works. These images needs to be present in a form such that computers can read and process them. So, a 2D array of numbers or pixel values represents an image. Images are captured with the help of cameras. The sunlight/flash of a camera falls on the object and the light reflects from the object. Sensors captures the incoming light signals. These sensors are known as the light detectors that are either a charge-coupled device (CCD) or a CMOS image sensor. The sensors generate a number of pixel values depending upon the incoming light signals. This is how the image is stored in a form of numerical array.

1.3 Image Features

Image features are the information present in the image in the form of numeric values. There are two types of image features viz. global feature and local features.

- **Global features**

  A multidimensional feature vector is used to represent an image. A single vector represents the various properties of the image such color, shape, brightness, edges. Each unique image will have its unique feature vector. These feature vectors represent images distinctively. For example, a global descriptor of shape will generate a different feature vector for the human and a bird. The features that are extracted from the original image have less dimensions as compared to the original ones. Global features are generally used for detecting and classifying the objects in an image. Global features requires lesser computations and have small memory requirements.

- **Local features**

  Local features do not depends upon the viewpoint and illumination conditions. Local features generate the feature vectors for the particular key points in an image. Local features are generally employed for the recognition of objects in images i.e. for recognition tasks. Local features are superior to the global features since they are not effected by viewpoint and illumination conditions. Local features requires larger memory and

computations since an image can have large number of local features. These large memory requirements are solved by using compact vectors to represent local image descriptors and further optimizing these compact vectors.

1.4 Character Recognition

Character recognition is a task of identifying the characters in an image. These character images are generated by scanning the documents and saving electronically in the form of images. Handwritten character recognition is an enhanced version of the optical character recognition. Artificial Neural Networks are employed for character recognition tasks. Feature extraction is an important step of the character recognition task. Basic steps for the character recognition task are scanning the input image samples, grayscale conversion, binarization, noise removal, smoothing, feature extraction, neural network training and classifying the characters. It is difficult to understand the functioning of the hidden layers of a deeper neural network for character recognition task. A great effort is required to understand the logic behind such systems. Characters recognition can be divided in the following categories:

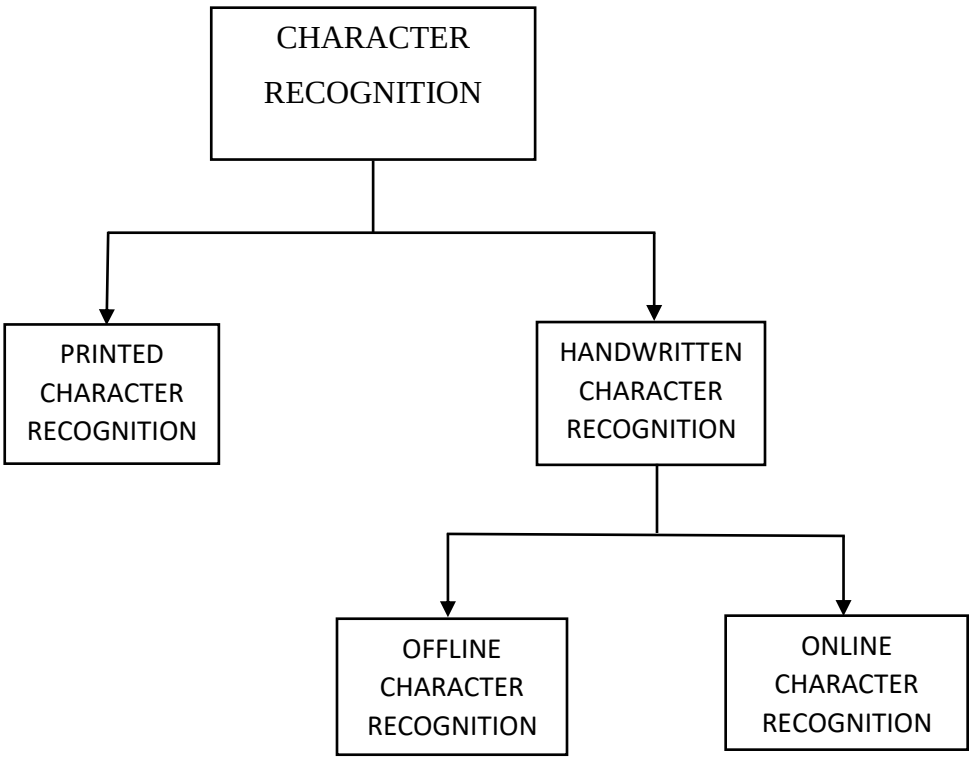

Fig. 1.1: Types of Character Recognition

- **Online Character Recognition**

Online character recognition is a real time process that captures and stores handwriting digitally with the aid of special input devices. A special pen moves on the electronic surface that stores the 2D co-ordinates in order, which represents the function of time. The online character recognition has higher accuracy than offline character recognition. Online character recognition system extracts real time features form the input, which enhances their recognition accuracy. Online character recognition process captures the input response quickly such as writing speed and direction.

Preprocessing of the online characters is quite easy and fast. The characters can be easily recognized and distinguished based on writing direction of the pen. However, online character recognition requires special devices to capture characters such as digital pen and a touch sensitive device as opposed to the use of a simple pen and paper in offline character recognition. Online character recognition system is not applicable on printed documents and handwritten characters on sheet of papers.

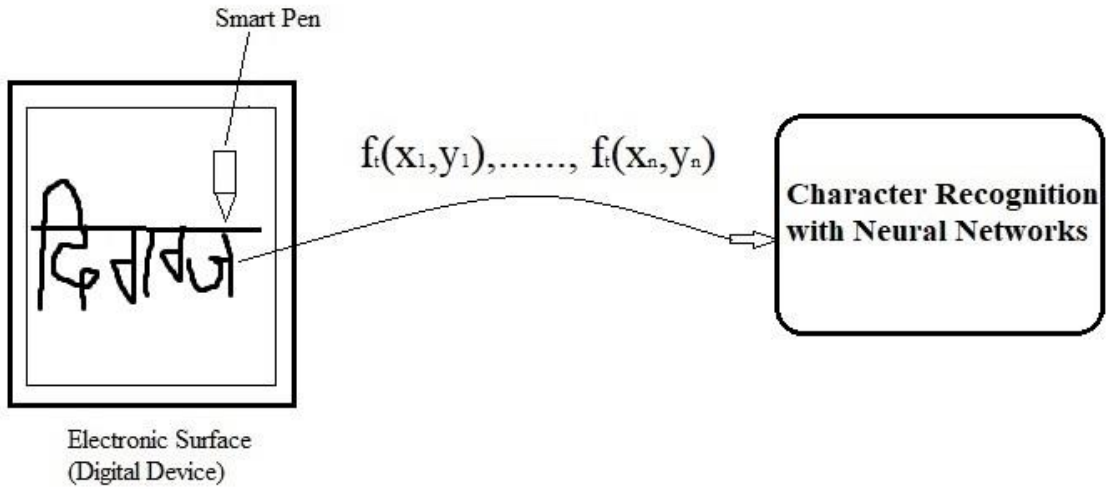

Fig. 1.2: Online Character Recognition System

- **Offline Character Recognition**

Offline character recognition is non-real time process. The character images are scanned from the documents, and stored in grayscale format for further processing. Offline character recognition are classified in two categories viz. Magnetic Ink Character Recognition (MICR), and Optical Character Recognition (OCR). MICR has its applications in banks for check verification, and signature verification. In MICR, the characters are distinguished with their magnetic field, which is unique for each character. OCR is used for recognizing the printed or scanned character images. OCR is further classified in two types:

i. Handwritten character recognition.
ii. Printed character recognition.

Handwritten character recognition is more complex as compared to the printed character recognition, since each person inscribes each character in a different style. A large character images dataset is required for training the neural network for higher accuracy. Pre-processing is an important step and requires larger computations for offline character recognition. Segmentation of offline words into characters is a difficult task since there is no pen-lift information recorded by a digital device in offline character recognition.

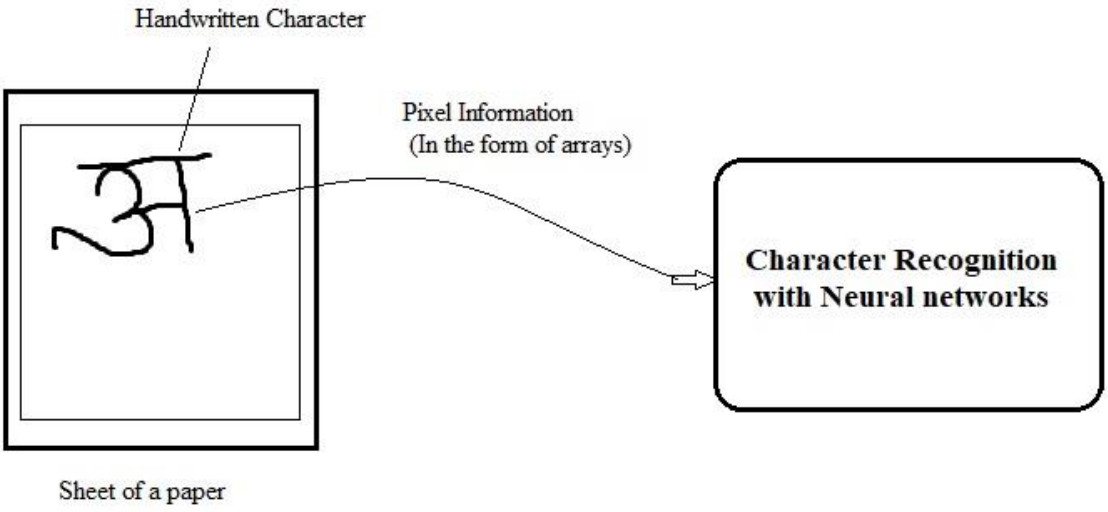

Fig. 1.3: Offline Character Recognition System

- **Features of Online and Offline Handwritten Character Recognition**

Table 1.1: Online and Offline HCR Features

Features	Offline Handwritten Character Recognition	Online Handwritten Character Recognition
Real Time Processing	No	Yes
Pre-processing	Requires higher computations, hence costly	Requires lesser computations, hence fast and economic
Segmentation	Complex	Easy with pen-lift information
Classification	Complex	Easy
Character Capturing Technique	On a sheet of paper	On a digital surface (Tablet)
Ambiguity	Slightly Higher	Less
Printed Character Recognition	Yes	No

1.5 Devanagari Script

Devanagari script is one of the ancient script that is mostly used in India and Nepal. Devanagari script is written from left to right with a horizontal line at the top, which joins all the characters together to form a word. Deva means ‘deity’ and nagari means ‘city’ so a Devanagari script is considered as the script of the city or script of the Gods in India. Devanagari script is used by over 132.42 crore population which is an extensive figure after English, and Chinese scripts. Devanagari script is derived from the ancient Sanskrit script, which was in use in early 1st to 4th century CE. The cave inscriptions, temple inscriptions and pillars with Sanskrit script embedded on them shows that its use was relevant not only in India but in other countries also such as Sri-Lanka, Mynmar and Indonesia. Buddhists used the Siddha Matrika Script that has a close relation with the Devanagari script. The Devanagari script is a foundation for over 120 other languages that are prevalent in India. For example Awadhi, Bhili, Dogri, Bhojpuri, Nepali, Garhwali, Rajasthani, Haryanvi, Maithili, Marathi, Sindhi etc. Awadhi script which is a derived from of Devanagari script is an official language of Fiji.

The use of Devanagari related scripts was found in medieval period such as in 8th century, the Pattadakal pillar in Karnataka has Siddha Matrika script, and Telugu Kannada Script both of which are closely related to the Devanagari script. Similarly, the inscriptions written in Sharada and Devanagari script on the Kangra Jwalamukhi temple of Himachal Pradesh shows the use of Devanagari script is an ancient one. Hindi written in the Devanagari script and English are the official languages of India.

In 7th Century, Tibetan king dispatched their ambassadors to India to learn the Sanskrit script. They returned and developed six new symbols for the local pronunciation in addition to the Sanskrit symbols. The Sanskrit script was recovered from the Maurya period having 1,413 pages of Sanskrit script. The horizontal bar at the top of Devanagari characters that joins all the characters together was taken as a source of inspiration form the Kutila Inscription of Bareilly having a composition date of Vikram Samvat 1049. Due to the numerous invasions in the later medieval period from Central Asia, Afghanistan, and Turkey the Devanagari script was influenced by their dialects and the modern Devanagari, which is spoken today, is an evolved version of the original script.

1.6 Features of Devanagari Script

- Devanagari script has thirty-six consonants, fourteen vowels, ten numerals, and there is no upper or small case letters in Devanagari script.
- In addition to the consonants and vowels, Devanagari script has conjunct characters.
- The vowels can also be represented in the form of Matras/Modifiers instead of a letter.
- Devanagari script has no articles like in English script.
- The verbs in Devanagari script appears at the end of a sentence.
- Devanagari script has characters with different unique sounds.
- Some Devanagari characters are also used in English script such as bungalow, thug, avatar, looting etc.
- Understanding the gender representation in Devanagari script is quite complex.
- A vertical bar known as the Purna Viram is used to end a sentence like full stop (.) is used in English script.

1.7 Applications of Handwriting Recognition

- The character recognition system is employed in handwritten address interpretation (HWAI) that sorts the physical mails automatically according to the country, state, city, zip code, name and other information. HWAI system is used in United States Postal Service, in United Kingdoms, and in Australia.
- Banks uses handwriting recognition for check verification. The handwriting recognition can check the account information such as account number, account holder name, amount written in the check etc. Handwriting recognition systems can save time and avoid error by banker in misinterpreting any information written in check.
- Handwriting recognition is used for signature verification to identify a person. The signature is matched to the previously stored signature of the person at the time of opening the account.
- Invoice imaging uses handwriting recognition system to find data information in an invoice. It allows individuals or various accounting firms to keep record of the finance related information.
- Legal industries such as courts and various law enforcement firms has large data in the form of paper documents that needs to be processed on daily basis. Such handwriting recognition systems are employed for processing automatically such large data, saving time and work force.
- Handwriting recognition is used in hospitals, hospitals has to deal with lot of paperwork. They have to keep track of the patient information such as name, age, treatment, condition, medicines supervised etc. There are daily thousands of patients seeking medical help in which case a handwriting recognition system is quite handful to process such a large amount of paperwork automatically.
- Various educational institutions uses handwriting recognition systems to automatically evaluate assignments and assign grades accordingly. This saves the time of manually checking assignments and assigning grades.
- Digital Libraries and repositories uses handwriting recognition that scans the documents and store them digitally. This provides access to various thesis and journals, presentations, online lectures, e-books etc., which facilitate students to learn various subjects free of cost at their own pace at the comfort of their home.

1.8 Limitations of the Artificial Neural Networks

The accuracy of the ANNs depends upon the exact information of the input and the training method that serves as an algorithm for the ANNs. However, there are lot of algorithms available for improving the accuracy of the ANNs; it still requires a large amount of training data for higher accuracy. Therefore, we can say there is still need of superior algorithms to boost the performance of the ANNs.

- The accuracy of the ANNs is not one hundred percent. Even a one percent error could create troublesome situations. For example, there are the automatic self-driving cars based on the ANNs, still these systems are vulnerable to hackers attack, weather conditions could affect their accuracy such as heavy rain can distort the functioning of the sensors that are implanted at the top of the car.
- ANNs can't give higher accuracy for statistical systems. Examples of such systems are gambling in casinos or a roulette game. ANNs cannot increase the chances of winning in such games.
- ANNs can even takes weeks to train on a large amount of data, which takes much of the CPU time and memory. Currently available GPUs further needs to be upgraded for faster implementations of the neural networks. Therefore, ANNs are hardware dependent, which limits their performance.
- The probing solution that the ANN generates does not explain the reason behind it, which sometimes can be misleading.
- The structure of the ANN for a particular set of problem can be realized only after experimentation. This causes the wastage of time and cost.
- The ANNs can only understand the numerical values, so any problem needs to be converted into the numerical arrays to further feed into the ANN.

The research in the field of artificial neural networks is picking momentum by eliminating the limitations and increasing the advantages. It is now becoming an integral part of our lives day by day. It can help people to avoid overload of work and can reduce stress factors. However, too much dependence on such technologies creates loss of human jobs and it kills our creativity. Our identity comes to risk at the hand of the hackers. The use of the ANN is infinite although the work in this field still needs to be done to work to its full potential.

1.9 Feature Detectors in Images

ANNs learns the features of the input images by adjusting its parameters i.e. weights and biases. These weights are the amplitude of the connection between neurons and bias values are used to indicate the strength of the input features. There are number of learning algorithms used for the feature learning. These learning algorithms differs from each other based on the fact how they update neural network parameters. Whenever, there is a strong input feature appears, the neurons fires giving output to the next neuron in the neural network. The network learns different features by adjusting its weights and biases. There are the three categories of the feature detectors :

- **Single-scale detectors**

  Single scale detectors uses only a single representation for the features. Detector's internal parameters represents a feature. Single-scale detectors remains unaffected with the changes in the input feature like rotation, translation, changes in illumination conditions, and addition of noise. Single scale detectors on the other hand are unfit for the scale variations. Single scale detectors can detect majority of the corners in an image by testing each pixel value in an image. Single scale detectors have also combined corner and edge detector technique, which is more favorable. A Fast corner detector, which is a type of the single-scale detectors, is used for real-time video processing. Examples of Single-scale detectors are Harris detector, SUSAN detector, Hessian detector, Fast detector etc.

- **Multi-scale detectors**

  Multi-scale detectors can deal with the scale variations. Such detectors are capable of detecting and extracting distinct features under scale variations. Multiscale detectors are not affected by the rotations and are invariant to the scale changes. Such detectors are useful for detecting blobs, corners, multi-junctions, edges, and ridges in an image. Multi-scale detectors can enhance the low-level features of an image. Examples of low level features are peaks, valleys, and ridges. Such detectors can extract points from the images with scale variations, and the points extracted have high accuracy and can adapt to the geometric transformations. Some examples of the multi-scale detectors are Laplacian of Gaussian detector, Difference of Gaussian detector, Harris-Laplace detector, Hessian-Laplace detector, Gabor-Wavelet detector etc.

- **Affine Invariant detectors**

These detectors are used to provide robustness against various affine transformations. The previous two detectors can handle affine transformations in a small way but they assume that scaling is different in each direction in spite of uniform scaling. Affine invariant detectors can handle such problems. Affine invariant detectors can detect higher reliable regions from an image. Such detectors have high performance for textured and structured scenes.

1.10 Characteristics of Feature Detection Techniques

The local feature of an image can be described as a pattern that differs from its next neighboring patterns. The feature detection techniques should have the following properties for enhancing the performance of the neural networks:

- **Robustness**

The feature detection technique should be able to avoid errors while producing the output and needs to cope up with input errors. A good feature detection technique should detect the features from the same region irrespective of scaling variations, rotation, color, shading, texture variations, and distortion in image by compression techniques.

- **Repeatability**

A feature detection technique should detect the same feature from a particular region in an image irrespective of the varying illumination conditions. It should not show any variations in the features extracted under different illumination conditions.

- **Accuracy**

A good feature detection technique should accurately describe the locations of the image pixel values. Such a task is of great importance for the image matching applications.

- **Efficiency**

A feature detection technique should quickly detect the features from an image, which is beneficial for the real time applications. It should produce the desired results without compromising with the accuracy in minimum amount of time with little efforts and computational cost.

- **Quantity**

  A feature detection technique should detect every feature of an image. Collectively these features gives the information content of the image.

However, all feature detection techniques should have the above-mentioned properties but their actual use may depend on particular application settings.

1.11 Different Paradigms for Training Artificial Neural Networks

There are the following two methods that can be used for training the artificial neural networks:

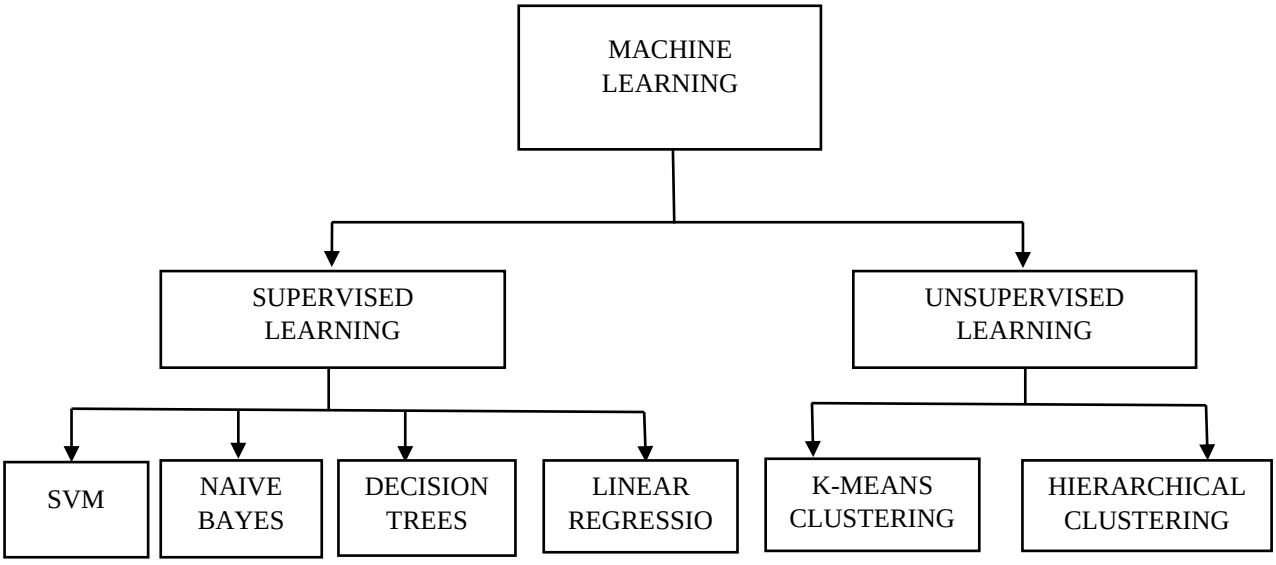

Fig. 1.4: ANN Training Methods

1.11.1 Supervised Learning

It is the method of training the neural network under supervision. Consider a multi-class classification problem where the neural networks classifies the input handwritten character images into different classes. If the network predicts that, the input is character अ then it should generate output equal to '1' otherwise 'zero', which is based on a particular threshold value. This output is then compared with the target value to determine the correctness of the neural network. Any variation from the target value is termed as an error. The neural network tends to remove such an error by adjusting its parameters i.e. network weights and biases. This method lets the neural network to learn or optimize its weights and bias values, so that the output value matches the target value. This allows the neural network to learn the different features of the input and output

mappings. The accuracy of the network is then evaluated on new dataset that is not the part of the training phase. An example of the supervised learning is back-propagation neural network.

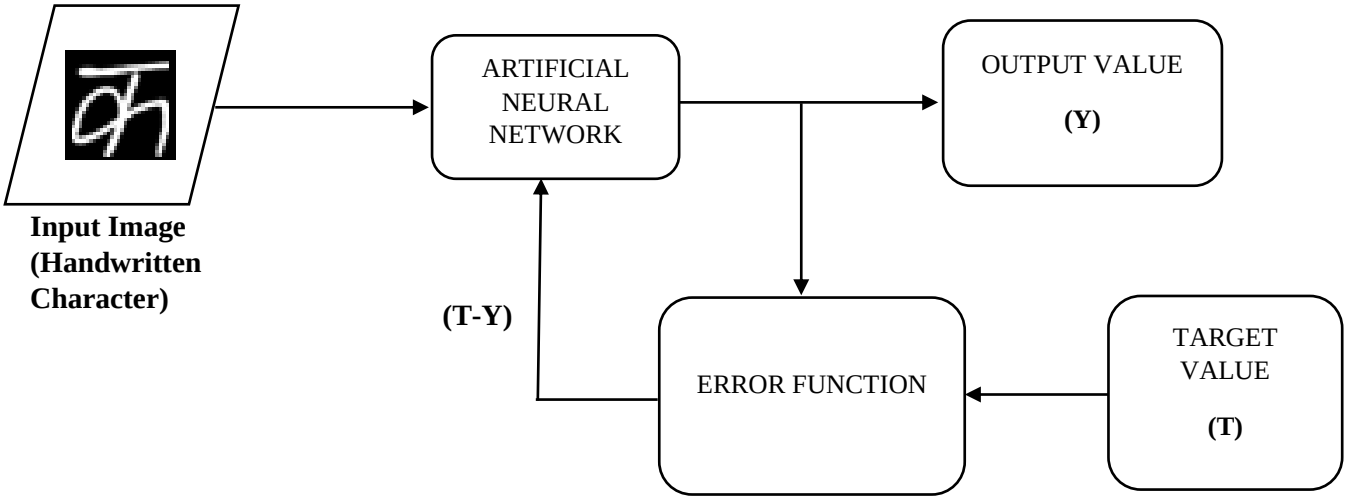

Fig. 1.5: Supervised Training with Backpropagation Network

1.11.2 Unsupervised Learning

It is a method of training the neural network without supervision. There is no target data given to the training dataset to learn how the actual output looks like. The network itself has to learn the various input patterns and decides the suitable weights and bias parameters for each input pattern. During the training phase of the artificial neural network, the similar input vectors combines to form clusters. These similar looking input vectors corresponds to the same output class. The clusters can be formed with the different methods given below:

- **K-means clustering**

  In k-means clustering, the input vectors in different clusters differs from each other while the vectors within the same cluster are similar. The idea is based on the fact that similar character images must have most of the features matching with one-another, so these images are grouped together into a cluster representing the features belonging to the image of the same class. The number of clusters depends upon the number of input classes.

- **Hierarchical Clustering**

Hierarchical clustering allows forming a sequence of clusters. It specifies the number of clusters that are required in the output. Clusters follows a particular order throughout. Hierarchical clustering can be achieved by either Divisive method or by Agglomerative method.

- **Principle Component Analysis (PCA)**

PCA is a method of unsupervised learning that reduces the dimensional representation of the input dataset. Principle components, which are the linearly combined variables, indicates the maximum variance and gives most of the information about the input data. These principle components differs from one-another in terms of the input data information. Obviously, the input dataset has many similar variables; PCA combines these similar variables into a single representation known as a Principle component. This helps to reduce the variables and thereby, reducing the data dimensionality.

In unsupervised learning, the artificial neural networks has to learn the input patterns, and features by itself without any supervision, so that the neural network can generalize over the input data to classify the input image to its correct class. This type of learning technique is complex as compared to the supervised learning. The ANNs that uses unsupervised learning are Hamming network, Max network, Kohonen network etc.

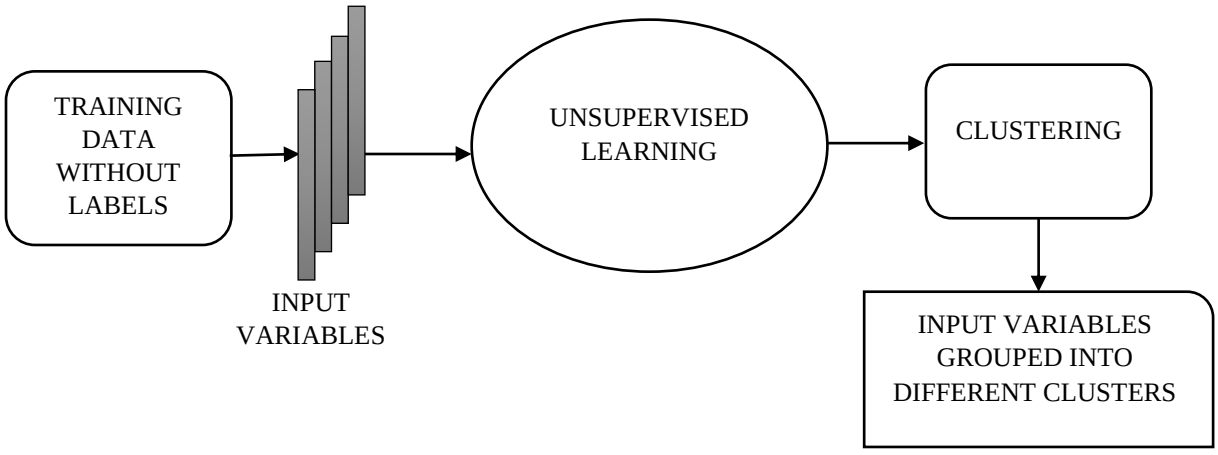

Fig. 1.6: Unsupervised Learning Algorithm

1.12 Basic Structure of Devanagari Handwritten Character Recognition System

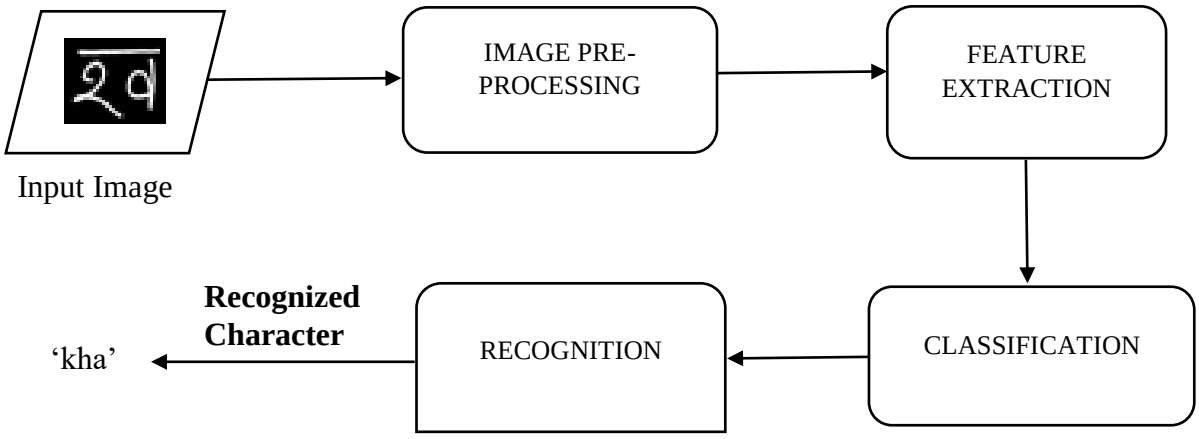

Fig. 1.7: Character Recognition System

1.12.1 Image Pre-Processing

Pre-processing is the first step that is required to intensify the image classification task. It enhances the image for further processing by removing unwanted noise, distortion and other imperfections, which makes recognition task difficult. The various pre-processing steps are noise removal, color to gray scale conversion, binarization, inversion, skeletonization, edge enhancements, and morphology, blur corrections etc. Some of the image pre-processing steps are discussed below:

- **Noise Removal**

  Imperfections in the image-capturing sensor can introduce noise in the image, transmitting an image from one source to another also introduces noise in an image; but not limited to only these causes. Noise removal task removes the unwanted distortions and other artifacts without compromising with the underlying information of the image. There are various filters available for noise removal task. Median filter removes random noise from an image; notch filter removes transient frequency spikes or spectral noise from an image etc.

- **Color to Grayscale Conversion**

  Grayscale images are computationally efficient as compared to the color or RGB images. Grayscale images have intensity values in the range 0-255; where 0 represents strong black color and 255 represents strong white color. The grayscale images are black and white images.

- **Binarization**

Binarization technique is applied to the grayscale image to convert the grayscale image into a binary one. The binary image has only two intensity levels either 0 or 1 that is why it is also called as bi-level image. Otsu's method, Kittler method, Fixed Thresholding method, Adaptive method, Sauvola method are some of the techniques which implements the binarization. The general method for binarization is fixed-thresholding. In fixed thresholding method, the grayscale image is assigned only two intensity levels either 0 or 1 based on some threshold value. The following function represents the fixed thresholding method:

$$g(x,y) = \begin{cases} 0, & g(x,y) \leq Threshold\ value \\ 1, & otherwise \end{cases}$$

- **Inversion**

The inversion is the process of converting the image pixels from white to black or black to white. Inversion achieves the negative effect of the original image. To get the inverted image, each pixel value of the RGB image is subtracted from the highest intensity value of 255. It can be described as below:

$$Img(x,y) = 255\text{-}img(x,y)$$

- **Skeletonization**

This process is used to obtain the region-based features of the image. It is also known as a thinning process that reduces the pixel values of an image into a single pixel value to get the underlying shape of an image content. Skeletonization method eliminates the irregularities from the image that makes the recognition process more accurate. Skeletonization process is applied for optical character recognition, script identification etc. Skeletonization process can be achieved by either iterative methods or non-iterative methods. Iterative method removes the irregular pixels from an image after identifying the desired pixels iteratively. Non-iterative methods are generally slower than iterative methods and are less desirable. Non-iterative method does not evaluate the each individual pixel rather it directly obtain the skeletonized image. Skeletonization reduces the pixel values thereby enhances the image processing speed and reduces the memory storage requirements.

- **Edge Enhancement**

This technique is often applied to increase the contrast of the image. Edge enhancement removes noise and sharpen the image. Edge enhancement helps in visualizing the actual shape of an image. However, before edge enhancement, edge detection is required. Sobel operators, Laplacian operators, Prewitt operators etc. can detect edges.

- **Smoothing**

Smoothing operation removes the unwanted pixels from the image while preserving the desired patterns in the image. Smoothing can be achieved with the Low pass filters, Kalman filter, Butterworth filter etc. Low pass filters helps in eliminating the high frequency noise in an image. A moving window operator is applied on an image that changes the intensity values of an image for smoothing. Mean filters smooth an image by taking a mean value of the neighboring pixels to replace each pixel value. The operation of a mean filter is shown below:

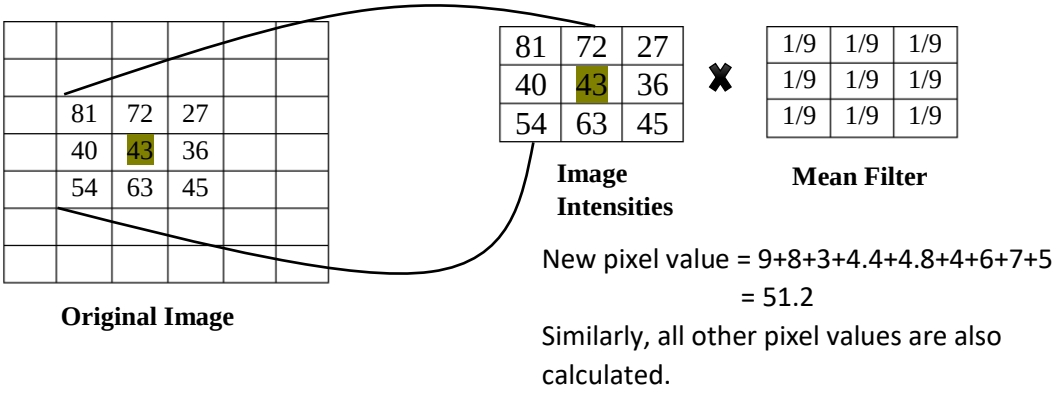

Fig. 1.8: Smoothing Operation with Mean Filter

- **Normalization**

Normalization operation changes the image pixel values to a desirable range. When it comes to the image processing, normalization is a general operation to convert the image into a standard size. This helps to use the data from different formats for further processing. Apart from the size normalization, the other normalization techniques are skew normalization that is required to correct tilt in an image or precisely to bring the image contents parallel to the horizontal, and slant normalization that is required to erect all the vertical strokes in an image.

1.13 Convolutional Neural Network

Convolutional Neural Network (CNN) is a branch of deep learning that works on the idea of biological neurons in humans. CNNs are a type of feed-forward artificial neural networks that is specially designed to work with the images. Neural networks are inspired from the visual cortex of the human brain. The biological neurons fires when they see a particular feature, this firing of the neuron causes another neuron to fire based on the strength of the firing of the previous neuron. This firing of neurons continues which transmits information to the human brain. The researchers have found that the biological neurons fires differently according to the visual patterns. Some neurons fires to differentiate between the colors of an object while some neurons fires based on the edge information in the visual pattern. CNNs have artificial neurons that can learn the input features from an image. Neurons receives an input feature from the image; performs a dot product between the input feature and the learnable weights, adds a bias value and produces an output followed by a non-linear function.

CNNs have multiple hidden layers that computes both lower and higher level features of an input image. The hidden layer neurons of the CNNs are connected to only small local region in the input image. This type of the arrangement makes the CNN unique from other neural networks such as fully connected neural network where each neuron in a layer is connected to all other neurons of the previous layer. The local connectivity of the CNNs reduces the amount of parameters and thereby reducing the amount of computations. The input features of an image are processed layer by layer and the final output layer holds the class scores for each input image. The CNNs uses the non-linear activation functions to learn the complex input features to enhance the performance of the neural network. The hidden layers of the CNNs are:

i. Convolutional Layers.
ii. Pooling Layers.
iii. Fully Connected Layers.
iv. Normalization Layers.

Convolutional layers and fully connected layers have trainable parameters whereas pooling layers and activation functions do not have any trainable parameters. CNNs can handle large amount of data and is much efficient for image processing and multi-class classification tasks as compared to the fully connected neural networks where wastage of network parameters occurs.

1.13.1 Convolutional Layers

Convolutional layer is the first layer in a CNN architecture. The function of the convolutional layer is to extract the features from an image. Convolutional layers extract features with the help of the learnable filters. These filters are the sliding windows, which slides through the entire image to generate the feature maps. However, the neurons in each layer is connected to a local region of the input image and the spatial extent of this local region is known as the receptive field. The elementwise multiplication occurs between the filter values and input image pixels and then adding a bias value to produce the output of the convolution operation. The convolution operation is represented as below:

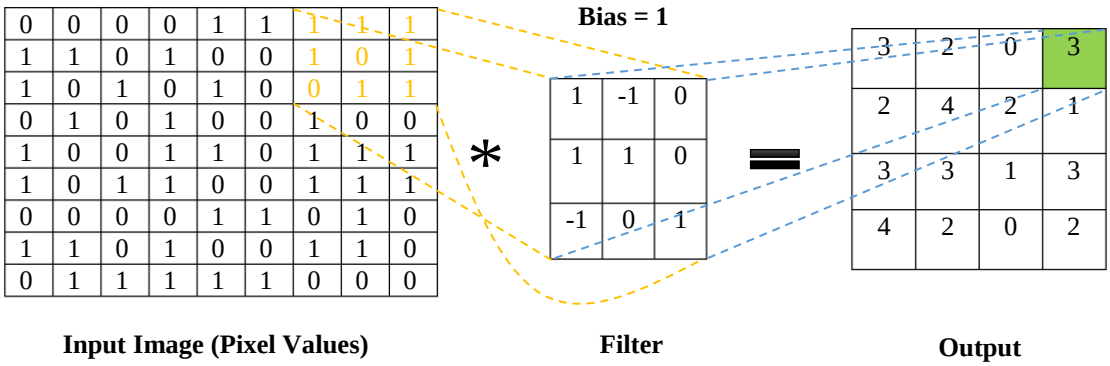

Fig. 1.9: Convolution Operation

The convolutional filters are the array of numbers, which detects the edges, contours, shape or other features in an input image. These filters gives a higher output whenever they come across a particular feature in the input. The parameters associated with the convolutional filters are:

- **Depth**

  The amount of filters determines the depth of the convolutional layer. The large number of filters in a layer results in higher depth of the particular convolutional layer. The large number of convolutional filters helps in learning the each individual feature deeply.

- **Stride**

  Stride determines the depth of the output volume. Stride is the amount by which we slide the filters on an input image. It can take any desired value depending upon the necessity.

- **Zero Padding**

In some cases, there is a need for zero padding the input image for performing the convolution operation with the desired filter size. We can perform the zero padding around the borders of the input image. Zero padding helps to maintain the spatial size of the input and the output volume. It could also avoid the loss of information at the borders of the input image.

1.13.2 Pooling Layers

Pooling layer is used in the neural network for subsampling operation. It reduces the number of parameters and computational time of the neural network. This layer helps in avoiding the problems of overfitting. Pooling layer reduces the size of the each input depth. Pooling layer is generally used after each convolution layer in the neural network. Pooling layer can reduce the input parameters up to 75% by using the filters with size 2×2 and a stride with value 2. Pooling layer has no trainable parameters. The pooling operation can be performed with the following methods:

- **Max-Pooling**

Max-Pooling operation outputs a maximum value within the sliding window space. Max-Pooling reduces the computational cost and makes the convolutional neural network invariant to the scale or orientation changes of the input image. The fact behind the max-pooling layer is that it gives importance to the relative position of the input patterns rather than the exact position. The following figure represents the max-pooling operation:

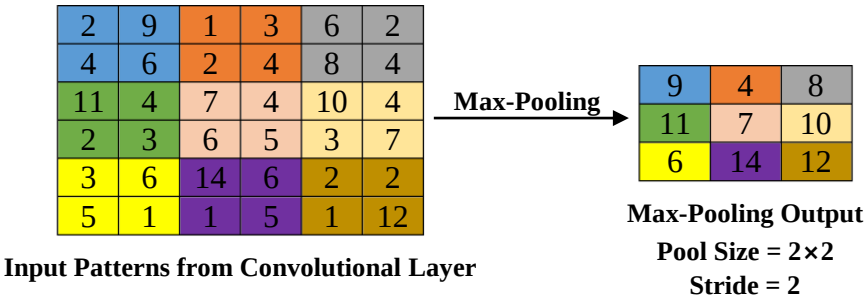

Fig. 1.10: Max-Pooling Operation

- **Average Pooling**

Average Pooling reduces the input size by taking the average value of the pixels within the sliding window. However, the average pooling is not widely used due to the better accuracy of the Max-Pooling. The following figure represents the average pooling operation:

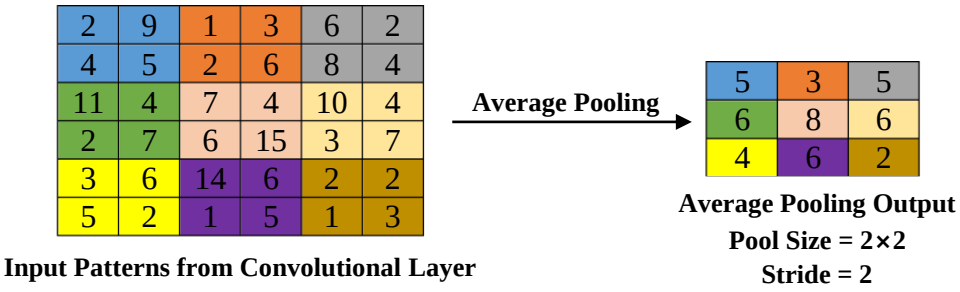

Fig. 1.11: Average Pooling Operation

1.13.3 Fully Connected Layers

The fully connected layer is generally used at the end of the CNN architecture. Each neuron in the fully connected layer has connections to all the neurons of the previous layer. The basic idea is that it receives all the high level features from the previous layer to classify the input image into an N-dimensional output vector, where N is the number of classes on which we are training our neural network. The fully connected layers uses a classifier such as softmax classifier, SVM classifiers to generate the class scores to which an input image belongs. The CNN can have one or more fully connected layers.

1.13.4 Normalization Layers

Normalization layers are generally less favorable in practice due to their low results. However, such layers normalize the output of the previous layer. This layer is useful for normalizing the unbounded activations and can detect high frequency features.

1.14 CNN Based Neural Networks

Some neural networks that uses CNN based approach are given as under:

- **ResNet**

ResNet is made up of many residual units that can be more than 100 layers deep. With the aid of ResNets it become easier to train the deeper networks. ResNets uses the concept of identity

mappings and shortcut connections. The accuracy of residual neural networks increases with increasing the number of layers. The ResNet can handle the accuracy degradation problems very well. ResNet can provide faster convergence for the neural networks with fewer layers.

- **VGGNet**

This network again focuses on the increased depth of the neural network. VGGNet uses number of convolutional layers with depth 64, 128, 256 and 512. It uses one max-pooling layer, three fully connected layers and a softmax layer that generate the class probabilities. However, the network involves large computational cost and memory requirements. This problem was later eliminated by discarding some fully connected layers.

- **GoogLeNet**

GoogLeNet introduces an inception architecture that can largely reduce the network parameters. Their architecture uses convolutional layers, pooling layers and a softmax layer; the network has no fully connected layers. The advantage of discarding the use of fully connected layers is the reduction for network parameters. GoogLeNet has a top-5 error of 6.67% in ILSVRC 2014 image classification challenge. The network trains on 1.2 million images, validated on 50,000 and evaluated on 1 lakh image samples. Inception-v4 is the most recent version of the GoogLeNet.

- **AlexNet**

AlexNet is based on the CNN architecture. AlexNet has five convolutional layers. Three max-pooling layers, two normalization layers and three fully connected layers. The data augmentation technique is used for increasing the size of the dataset. AlexNet trains on two GTX 580 GPUs having 3 GB of memory. The top-5 error rate is 16% for the AlexNet.

- **LeNet**

LeNet is an oldest CNN based seven-layer neural network. It uses three convolutional layers, 2 pooling layers and 2 fully connected layers. The first convolutional layer has six feature maps, the second convolutional layer has 16 feature maps and the third convolutional layer has 120 feature maps with 48,120 trainable parameters. The network trains on MNIST digits dataset that contains 60,000 training samples and 10,000 test samples of images.

CHAPTER-2

LITERATURE REVIEW

D. Wang *et al.* [1] has proposed convolutional neural network and multimodal fusion for image and text classification. A mini batch gradient descent and the backpropagation algorithms are introduced for tuning the neurons of the hidden layers. They also proposed the use of convolutional neural networks for text classification. They proposed the use of the one hot CNNs for longer texts and the CNNs with word embedding are more useful for shorter texts. They used the word embedding as input to the CNN for the text classification task. A skip-gram architecture is adopted to predict the sequence of words given a single word. They proposed the use of pre-trained input word embedding for good generalization and classification tasks. The multimodal fusion has two parts, which are multimodal CNNs early fusion and multimodal CNNs late fusion. Early fusion learns the same features of the different input samples whereas; the late fusion maintains the independence of the different input samples at the classification level. The largescale dataset enhances the effectiveness of the neural network.

A. Graves *et al.* [2] presented a connectionist system for unconstrained handwriting recognition. They suggested using dictionary to map individual characters to complete words. They proposed a recurrent neural network based approach with a Connectionist Temporal Classification (CTC) layer for mapping the input sequences to the output labels. The BLSTM network performs the unconstrained handwriting recognition task. They discussed the online data preparation with the eBeam interfaces. Various normalization steps for online character recognition such as skew correction, slant correction, delayed stroke correction in real-time handwriting recognition and scale normalization are proposed. They discussed the offline data preparation, which holds the gray scale images. Normalization methods applied to the offline data are skew correction, slant correction and scale normalization. Sliding window is proposed for features extraction from the input image. They discuss the advantages of BLSTM network that can store both the past and the future information. The neural network is evaluated on two different datasets viz. online dataset IAM-OnDB and offline dataset IAM-DB. The main goal is to transcribe the text lines in the output with the aid of dictionary words. They compared the performance of HMM and RNN neural networks. The online handwriting recognition task has 5,364 text lines for training and 3,859 text lines for evaluation purpose. Besides this there are two validation sets one having 1,438 text lines and other is having 1,518 text lines. The offline recognition system has 6,161 text lines for training, 920 text lines for validation and 2,781 text lines for test set. The proposed approach achieved 79.7% accuracy rate on online data and 74.1% accuracy on offline data. Input sequence segmentation is not required for RNN-CTC based network. They concluded that the recognition

accuracy of the proposed approach for both the online and offline recognition task is higher as compared to the HMM based system.

R. Tavoli and M. Keyvanpour [3] proposed a multi-layer perceptron neural network and a swarm optimization method for handwritten word spotting. They discussed the two methods for the handwritten word spotting viz. template based and learning based methods. They proposed a learning based approach for handwritten word spotting. The proposed MLP neural network is trained with the particle swarm optimization method. Based on the output of the MLP neural network the handwritten word is spotted or rejected. Learning based systems are more suitable as compared to the template-based methods for the handwritten words from different writers. Normalization steps includes binarization, filtering, scaling, skeletonization and feature extraction. The swarm optimization method updates the neural network parameters and tangent-sigmoid is used as activation function. The performance of the network is evaluated with the Mean Average Position and recall measure. The proposed approach is implemented on MATLAB software. The network is evaluated on the two features, which are LGH, and Gabor features. The improvement in the MAP measures of the proposed approach is 15% for the English dataset, 10% for the Arabic dataset and 5% or the Farsi dataset. They concluded that the proposed approach has achieved better results as compared to some of the recent experiments on handwritten word spotting.

T. He *et al.* [4] presented a Convolutional Neural Network (CNN) for scene text detection with Text-Attentional mechanism. They developed a deep multi-task learning mechanism for CNN training. They proposed a rich supervised information for CNN training to increase the robustness of the neural network. The supervised learning contributes to the low-level feature learning from the text patterns for high-level binary classification. They proposed a Contrast Enhancement Maximally Stable Extremal Regions (CE-MSERs) that increases the contrast between the text patterns and complex background. They focus on the exact text localization in an image. The Extremal Region (ER) detectors can detect low-level text patterns in an image. However, the ER detectors requires filtering operation for removing the non-text components. The text components are filtered with the text attentional CNN. The proposed CE-MSERs can detect highly distorted components in images with complex and low contrast background. They used the hinge loss for classification and least square loss for regression. The proposed text-attentional CNN has three convolutional layers, one max-pooling layer and two fully connected layers. A subnetwork is also designed in the same text-attentional CNN for region regression training. This subnetwork has two

deconvolutional layers. Stochastic gradient descent updates the network parameters with additional multi-level prior feature information. A cluster based approach enhances the contrast of the large regions in an image. For low region contrast enhancement, they proposed a two-step method; first calculate the remaining non-contrast region and then use a color space smoothing. The proposed CE-MSERs detectors outperforms the MSERs detectors. CE-MSERs improves the final detection rate by approximately 6%. The Text-attentional CNN is trained on CharTrain and CharSynthetic datasets. The former dataset has 17,733 text characters and 85,581 non-text characters and the latter one has 80,141 character image samples. The text-attentional CNN has an accuracy rate of 91% on ICDAR 2011 dataset, which is higher than the conventional CNN with 85% accuracy rate. The proposed approach has an 87% accuracy rate on ICDAR 2005 dataset, 76% accuracy rate on MSRA-TD500 dataset and 93% accuracy rate on ICDAR 2013 dataset.

J. Dolinsky and H. Takagi [5] discussed the naturalness of the handwritten characters. They mathematically analyze the font and its naturalness relation. The base for the mathematical analysis are the Canonical Correction Analysis (CCA), multiple linear regression analysis, feedforward neural networks and recurrent neural networks. They proposed a shape simulation technique. They defined a target system and the basic system. Basic system describes the behavior of the system. There is a small deviation between the behavior of the target system and the basic system. This deviation is the naturalness that has a relation with the basic system. The main objective is to understand this relationship. They suggested modeling the difference between the behavior of the target system and the basic system. Japanese based hiragana characters are used for experimentation. They choose the handwritten characters as the target system and the font characters as the basic system for learning the naturalness in the handwritten characters. The strokes of the characters are extracted with the vector graphics editor known as Inkscape. The difference vector shows the writing trajectory of each individual character that is position independent. The naturalness of the handwritten characters are represented in the form of two-dimensional vector. CCA explains the relationship between the two sets of variables. The naturalness is further modeled by the MLRA. Due to the low accuracy of the MLRA, FFNN is proposed for modeling the naturalness. Further analysis is done by FFNN with sliding window approach. RNN completes the task of temporal pattern learning and spatial pattern learning. They concluded that the naturalness of the handwritten characters could be understood with the help of the proposed approach.

B. Shi *et al.* [6] proposed an end-to-end trainable neural network for sequence recognition in images. The proposed neural network can perform feature extraction, sequence modeling and transcription. A Convolutional Recurrent Neural network (CRNN) is used for sequence recognition. CRNN can directly learn the sequence labels, it doesn't require the pre-processing steps, CRNN can produce sequence of labels, it doesn't depend on the length of the sequences. CRNN has high accuracy rate for word recognition and is memory efficient. The proposed network has convolutional layers, bidirectional LSTM layer and a transcription layer. The network trains on deep features. The convolutional layers perform the feature extraction task and max-pooling layer is used for down sampling. Sequence labelling is achieved with the RNN. They insert a Map to Sequence layer between the convolutional layers and the RNN layers. Transcription coverts the RNN predictions in the output label sequences. The network is trained end-to-end on image samples and sequences. They suggested SGD for neural network training. Adadelta is used to optimize the neural network. The performance of the proposed neural network is evaluated on scene text recognition and music score recognition problems. The network is trained on Synth dataset. They suggested that the use of batch normalization layers for neural network training. The proposed CRNN approach achieves higher recognition accuracy on IIIT5K and SVT datasets as compared to the other previous works. The accuracy rate on IC03 and IC13 datasets are very close to the state of the art results. Musical score recognition task involves three datasets viz. Clean, Synthesized and Real-world image datasets. CRNN achieves higher recognition accuracies on all three datasets as compared to the Capella Scan and PhotoScore methods. They concluded that the CRNN outperforms DCNN and RNN based methods for scene-text recognition task. CRNN outperforms the previous work on musical score recognition task.

F. Yan *et al.* [7] presented an efficient fast deep neural network that further leads to the design of SERF based systems. SERF is used for optimal parallel configuration of the system. SERF can adapt to any machine learning systems. They presented a technique to build scalable and responsive systems that can enhance the processing time of deep neural networks. They suggested parallel optimization for reducing the computational time. The first approach they suggested is to divide the computations among multiple servers, second approach is intra-node parallelism where multiple threads are used at each server and the last approach is to use service-parallelism where the multicore server handles multiple requests. Although, the service-parallelism approach is not beneficial for handling individual requests. The careful design of optimal parallelism scheme is necessary. The proposed approach can find optimal parallel configuration very fast. The queuing

based analytical model predicts the performance of the SERF. The accuracy of the proposed approach is evaluated on ImageNet and CIFAR datasets. Inter-node parallelism divides the neural network computations among several machines. Workload profiling reduces the complexity of the optimal parallelism. The speed of inter-mode parallelism is almost doubled with the parallelism degree of two. The factors which affects the computational speed of the deep neural networks are size of the network, cache size and memory bandwidth. They describe the deep neural network requests as the homogenous requests. The parallelism methods depends upon the input loads. They concluded that proposed approach provides automatic access to schedule service requests optimally. SERF update its scheduling parameters with the changes in workload and it can generalize to other workloads as well.

K. He *et al.* [8] presented a deep learning approach for training the deep neural networks. They presented a 152-layers neural network that is eight times deeper than the VGG neural network. The accuracy of the 152-layer neural network is evaluated on ImageNet 2012 classification dataset. The short-cut connections are used to perform identity mappings. Identity mappings does not increase the network parameters and does not affect the network complexity. With the help of proposed approach the network can be easily optimized and have higher accuracy for deeper neural networks. The ResNet evaluated on the ImageNet dataset is deepest and it is less complex than the other previous works on the same dataset. They experimented that the learned residual functions have low response. This helps to achieve good preconditioning. Shortcut connections can compare the performance of residual and non-residual networks simultaneously. Instead of using pooling layers, the downsampling operation is performed by the convolutional layers with stride value of two. A fully connected layer with 1000 neurons and a softmax classifier is used to generate class probabilities. The short connections makes the network a residual one. Batch normalization is used after every convolutional layer. 34-layer ResNet have higher accuracy by 2.8% than 18-layer ResNet. The ensemble model of ResNets has top-5 error of 3.57% on the test set. Whereas the single model ResNet has 4.49% validation error. The proposed ResNets has good accuracy rate with deeper models and is still less complex than VGG network. This approach also holds first position in ILSVRC and COCO 2015 competitions.

K. Simonyan and A. Zisserman [9] presented a deep convolutional neural network for image recognition. The accuracy of the convolutional neural network is evaluated with the increased number of layers or depth of the neural network. The input RGB image of size 224 x 224 pixels is

passed through the convolutional layers with filter size of 3 x 3. The dataset contains images of 1000 different classes; 1.3 million image samples are set aside for training the neural network, 50,000 image samples for validation and test set consists of 1,00,000 image samples. It is observed that increasing the depth of the network also increases the classification accuracy of the neural network model. The paper proposed that a deep neural network with small filters could give higher accuracy as compared to the shallow neural network with large filters. It is experimented that increasing the depth of the neural network to 16-19 weight layers attains a significant rise in accuracy.

Xu Chen [10] has discussed the different architecture of the convolutional neural networks to classify the handwritten characters of the Chinese character dataset. The gradient feature extraction is also experimented on the similar convolutional neural network architectures. The author has proposed CNNs as an efficient method to classify handwritten characters and discuss the improvements in feature extraction with convolutional neural networks. Different architectures of the CNNs are analyzed by varying the depth/number of layers and number of filters. The CNN is trained on the Chinese handwritten character dataset, which involves 90 thousands samples for training, 12 thousand samples for validation and 24 thousand samples for test set. A sobel filter is used to extract the gradient features from the input image samples. It is observed after experimentation that a CNN with larger number of filters, small filter size and increasing the depth of the CNN results in a high accuracy. It is presented that replacing the fully connected layer with the convolutional layer can increase the performance of the convolutional neural network. The gradient feature vectors can provide only small improvement in accuracy for the deeper neural networks.

Yuhao Zhang [11] discussed the performance of deep convolutional neural network for Chinese handwritten character recognition. The performance of the convolutional neural network is evaluated on the two datasets; the first one has 200-class dataset and the second one has 3755-class dataset. The convolutional and the fully connected layers are followed by the ReLU activation function. The 200-class classification task uses the fixed filter size of 3 and the 3755-class classification problem uses the various filter size of 11,7,5 and 3. The experimental results shows that with the same filter size the accuracy of the neural network is increased by increasing the depth of the convolutional neural network. The neural network with six weight layers and lager filter numbers can give higher accuracy for the classification task. CNN achieves higher accuracy

by replacing the fully connected layer with the convolutional layer. It is observed for a 7-weight layer neural network increasing the number of filters outperforms the addition of the fully connected layer to the neural network.

U. Pal *et al.* [12] presented an offline Devanagari handwritten character recognition system. A quadratic classifier is proposed for handwritten recognition of the Devanagari characters. The feature recognition task is performed by using the directional information, which is gathered from the gradient. They have described the complexity involved in the Devanagari character recognition task. The complexity is mainly due to the different ways each individual writes a particular character. The similar shape characters are more difficult to recognize. The dataset includes 36,172 image samples of the Devanagari handwritten characters. For feature extraction purpose, 392 dimensional feature vector is used. They have discussed the individual accuracy of some Devanagari characters. They obtained 94.24% accuracy in recognizing the handwritten Devanagari characters.

Seema A. Dongare *et al.* [13] presented the handwritten Devanagari character recognition system using neural networks. For feature extraction, a grid-based method is proposed. The segmentation-based approach is also adopted for character recognition task. They have discussed some complexities in recognition of Devanagari handwritten characters like individual characters in a particular word are connected by a horizontal line known as shirorekha, which makes the segmentation process difficult. Some characters are similar in shapes, which makes recognition task difficult. They have discussed preprocessing, segmentation and feature extraction process to carry out character recognition task. Segmentation task involves segmenting the documents into lines, lines into words and words are segmented into characters. Error back propagation technique is used for training the neural network. They have suggested the use of larger dataset to achieve higher accuracy in recognizing the handwritten characters.

Gaurav Kumar and Pradeep Kumar Bhatia [14] presented a review paper of feature extraction in image processing. They have discussed various feature extraction techniques and proposed different feature extraction techniques suitable for a particular problem. For the purpose of character recognition task, feature extraction technique is used. Feature extraction extracts the important information from the image such as particular shape, number of pixels, edges, contours etc. Feature vectors represent each character, which is unique for each character. They have described feature extraction and feature selection methods to obtain subset of features. Feature

selection is an important step for the handwritten character recognition task. They have described the two categories of the features namely local features and global features. For handwriting recognition by computers Macro and Micro features are used. They have discussed the three major feature groups, which needs to be extracted, they are Statistical features, Global Transformation and Series Expansion features and Geometrical and Topological features. Diagonal based feature extraction, Fourier descriptor, Principal Component Analysis, Independent Component Analysis, Gabor filters, Fractal Theory Technique, Shadow Features of Character, Extraction of Distance and Angle Features, Extraction of occupancy and End point features, Chain Code Histogram of Character Contour, Transition feature and Zernike Moments are some techniques that are discussed for the feature extraction problem.

R. Plamondon and S.N. Srihari [15] gave a survey paper on Online and Off-line Handwriting Recognition. They have described the nature of handwriting and the survival of the handwriting through pen and paper. Handwriting recognition, handwriting interpretation and handwriting identification is discussed. Handwritten documents is converted into the digital format by scanning the handwritten documents or by writing with a surface pen on an electronic surface. The first approach is known as the offline handwriting recognition and the second one is the online handwriting recognition. The data requirement for the online handwriting recognition is few hundred bytes and data is processed by taking hundred samples per second. The data requirement for the offline handwriting recognition is few hundred kilobytes and data processing rate is three hundred dots per inch. Top-down and Bottom-up models are discussed for handwriting recognition. Further, two general categories of the bottom-up models i.e. oscillatory and discrete models are also discussed. The have discussed Holistic and analytic methods of handwriting recognition. Preprocessing techniques are discussed such as thresholding, noise removal, line segmentation, and word and character segmentation. Handwritten address interpretation, bank check recognition, signature verification and writer identification techniques are discussed. They have discussed the major constraints in the offline and online handwriting recognition are the segmentation of words and characters.

Xiaoqing Liu and Jagath Samarabandu [16] proposed a Multiscale Edge-Based Text Extraction algorithm from Complex Images. The algorithm has the capability to automatically detect and extract the textual information from images. The algorithm is well applicable on printed document images as well as on scene text images. The algorithm can deal with different font sizes, style, colors, orientations and alignment of the text. Automatic detection and extraction of text from

images can be used in page segmentation, document retrieving, address block location etc. They have discussed the applications of the scene text extraction. They have described the edges as an important feature of the text. Edge strength, density and orientation variance are the important features in the text detection. They have discussed the three stages for the text detection and extraction in images. These three stages are candidate text region detection, text region localization and character extraction. Multi-scale edge detector technique is used to measure the strength of the edges. Feature map generation technique is discussed. They have used 75 test images of four different types with different font size, orientations, alignments and under different lighting conditions. They concluded that their proposed method is an effective and efficient technique to automatically detect and extract text from complex background images. The proposed method is computationally efficient which makes it useful for real time applications. The proposed algorithm uses variance of edge orientations to distinguish text regions. The proposed method can be used further for character recognition.

Hailong Liu and Xiaoqing Ding [17] proposed a gradient feature and quadratic classifier for handwritten character recognition. MQDF classifier is used for improved character recognition. Minimum Classification Error training is used to adjust the parameters of the classifier. Similar characters are distinguished by using the compound mahalanobis function to improve the classification rate of the MQDF classifier. Preprocessing and feature extraction techniques are discussed. 3x3 sobel operators extract gradient features, binary images are converted into a pseudo grayscale image, and images are normalized by gravity-center size normalization. The algorithm is applied to both the handwritten digits and handwritten Chinese characters. MNIST digits dataset is used for recognition and ETL9B and HCL2000 datasets are used for character recognition. Nearest mean classifier increases the classification speed. It is found after experimentation that gradient features are better than the directional elemental features. They propose the minimum classification error and compound mahalanobis function for improved handwritten character recognition. The proposed method has good result outcomes for the MNIST handwritten digits recognition and for handwritten Chinese character recognition.

Vikas J. Dongre and Vijay H. Mankar [18] have presented a Multiple Feature and Neural Network Classifier to recognize offline handwritten Devanagari character and numeral. They proposed structural and geometrical features to represent Devanagari characters and numerals. Multilayer Perceptron Neural Network and fivefold cross validation are proposed. Devanagari

characters and numerals are isolated for recognition. A self-generated dataset is used for both training and testing the neural network. The dataset contains 5137 numeral images and 20,305 character images. Preprocessing techniques are discussed such as normalization, extraction of geometrical and structural features, image binarization, noise removal, image thinning etc. Otsu's method is considered for image thinning. Dataset is randomized for better training of the neural network. Feature extraction technique is discussed. Extraction of eight structural features and nine global geometric features are discussed. Various parameters for training the neural network are discussed such as performance function, learning algorithm, learning rate, activation function etc. 80% of the dataset is recommended for training the neural network and 20% of the dataset for test set. Character classification is performed by multilayer perceptron neural network. Numeral recognition achieves highest 93.17% accuracy rate. The hurdles involved in Devanagari handwritten numerals recognition are discussed. They proposed the use of SVM classifiers for improved accuracy and PCA for reduction in the dimensions of the input features in images.

Sushama Shelke and Shaila Apte [19] presented a technique for optimizing the recognition accuracy at pre-classification stage, feature extraction stage and recognition stage. Different neural networks are applied for Devanagari handwritten character recognition and their performances are compared and analyzed. Different features of the Devanagari script are discussed. One of the important structural feature is the vertical line in the characters. They have discussed about the mid-line, end-line characters and no-line characters. Other structural features are the enclosed region in the character and position or number of end-points in the characters. Optimized Fuzzy based pre-classifier is proposed to reduce the error in recognition of the Devanagari characters. Recognition task is performed on thirty-nine unconstrained Devanagari characters. Image processing involves noise removal, binarization, image segmentation and cropping. Devanagari characters are assigned to different classes according to their structural features. They have discussed about the six classes of the Devanagari characters according to the vertical line and enclosed region. The six classes of the Devanagari characters are additionally classified into other 32-classes. Two stage structural classification results in 48-classes altogether. They have compared the recognition rate of the Mid-line characters, End-line characters and No-line characters. The optimized feature extraction technique is discussed. They have proposed Feed-forward back-propagation network (FFBPN), Cascade-forward back-propagation network (CFBPN) and Elman back-propagation network (EBPN). The dataset is split as 60% for training the neural network, 20% for validation and rest is used as a test-set. There are 69 features extracted from the

Devanagari character images with average projection profiles. The recognition rate of the proposed neural networks i.e. FFBPN, CFBPN and EBPN before and after optimization are compared. The result is that the optimized feature extraction technique gives higher accuracy as compared to the neural network before optimization for all three neural networks. The EBPN neural network obtains the highest recognition rate.

Partha S. Mukherjee *et al.* [20] have presented a Hybrid Model for End-End Online Handwriting recognition. They used Devanagari and Bangla scripts for online handwriting recognition. Devices used to store online handwritten characters are touch screen, pen tablets etc. They have discussed the features of Devanagari and Bangla script. They have proposed a hybrid model that can recognize unsegmented online handwriting. They have proposed Convolutional Neural Network (CNN), Recurrent Neural Network and Connectionist Temporal Classification (CTC) for online handwriting recognition. CNN architecture is discussed and backpropagation algorithm is proposed. Multi-channel image is used as an input to the CNN network. CNN architecture involves three pairs of convolution layer and maxpooling layer, one flatten and fully connected hidden layer and an output layer. RNN is proposed to process sequential information. RNN can store previous sequences in a memory in the form of its hidden units. RNN is a good option for temporal online handwriting recognition. However, this proposed simple RNN has a vanishing gradient limitation. LSTM and more enhanced version that is bidirectional long short-term memory (BLSTM) and bidirectional recurrent neural network (BRNN) can remove vanishing gradient limitations. BLSTM neural network is proposed for online handwriting data recognition without segmenting the data into characters. CTC layer is discussed to avoid the task of pre-segmentation. Devanagari dataset includes 41831 training samples and 9584 test samples. Bangla dataset includes 61728 training samples and 15277 test samples. The character level recognition accuracy rate for Devanagari script is 59.57% ($Feat_1$) and 82.39% ($Feat_2$). Similarly, Bangla script has accuracy rate of 68.21% ($Feat_1$) and 84.47% ($Feat_2$).

Shivajee Pandey *et al.* [21] proposed a gradient descent approach to learn the number of filters required at each stage of the PCANet. PCANet has three basic stages namely cascaded PCA, binary hashing and block-wise histogram. PCANet is used for dimensionality reduction. The proposed method is applied to the Devanagari and MNIST datasets. They have discussed the first stage of the PCANet in which each detailed feature is extracted from an image. The second stage of the PCANet is almost same as the first stage. The third stage involves hashing and histogram where

each pixel value ranges between $0\text{-}2^{L_2}$. After the completion of the last stage of the PCANet, feature vectors are obtained, which the SVM classifier for classification further uses these feature vectors. They also proposed the use of LIBLINEAR classifier for a multi-class classification task. PCANet achieves an accuracy rate of 91.10% for the MNIST dataset (Sample 1) whereas the proposed approach achieves 91.8% accuracy, for Sample 2 (S2) accuracy rate of PCANet is 98.20% and accuracy rate of proposed approach is 98.50%. Sample 3 (S3) includes Devanagari dataset which has training size and testing size of 605 and achieves an accuracy of 95.21% with PCANet and the proposed approach gives 95.87% accuracy. Sample 4 (S4) has training dataset size of 2420 and testing dataset size of 2417 and achieves an accuracy of 96.48% with PCANet whereas proposed approach gives 96.61% accuracy. The proposed approach achieves an accuracy of 84.09% on whole Devanagari dataset whereas PCANet gives an accuracy of 83.91%. The proposed approach of learning filters by gradient descend; required at each stage of PCANet achieves good results in image classification.

Durjoy Sen Maitra *et al.* [22] proposed a convolutional neural network for handwritten character recognition of multiple scripts. CNN is an efficient approach for extraction of feature vectors. They used English (MNIST) numerals, Devanagari numerals, Bangla basic characters, Bangla numerals, Telugu numerals and Oriya numerals for recognition task. They used 50-class Bangla basic characters to train CNN and extract features for five distinct 10-class numerals for recognition task. They have proposed normalization of the input image to size 32x32. CNN for character recognition task can take longer time for tuning its parameters and architecture. CNN architecture based on LetNet-5 architecture is used and trained for 10,000 iterations. The first convolutional layer has six feature maps. The proposed architecture has two sets of convolutional layers, a subsampling layer and a multilayer perceptron with one hidden layer. They have discussed feature maps that are obtained at each layer of the neural network. They have proposed the use of SVM classifier. Inductive Transfer Learning approach is used for handwritten character recognition task. The proposed approach achieves an accuracy of 95.6% for Bangla basic character recognition, 98.375% for Bangla numeral recognition, 98.54% for Devanagari numeral recognition, 97.2% for Oriya numeral recognition, 96.5% accuracy for Telugu numeral recognition and 99.10% accuracy for English numeral recognition. They have concluded that high recognition accuracies can be obtained by training the neural network on larger dataset.

Sushama Shelke and Shaila Apte [23] presented a Fuzzy based classification approach for handwritten Devanagari character recognition. They proposed a multistage classification scheme where fuzzy inference system is used as a first stage and structural parameters is used in second stage for classification. The character recognition accuracy depends upon acquisition device, pen width, pen ink-color, physical and mental condition of the writer etc. Devanagari characters are written under a horizontal line known as shiro-rekha, which connects all the characters in a word. About 60% of the Devanagari characters have vertical line and the remaining 40% characters have no vertical line in them. They proposed a fuzzy logic to perform the pre-classification of the Devanagari handwritten characters. Finally, a feed-forward neural network is used for character recognition task. The proposed system is carried in steps as input characters, character preprocessing, fuzzy based classification, feature extraction and then neural network is applied for character recognition task. The preprocessing includes averaging filters that are applied before binarization. They proposed the use of fuzzy rules for finding the vertical line and the position of the same in the characters. They have discussed fuzzy-inference method and defuzzification method. Defuzzification method is proposed for classifying the characters. They have proposed 8-adjacency to determine the connected components in the characters and enclosed regions within the characters. End-points in all eight directions are determined by eight structuring elements. Feature extraction technique is discussed where pixel density in non-overlapping zones and normalized pixel density are calculated. The character recognition task is discussed which involves a multilayer feed-forward backpropagation neural network. Each character has 500 samples, 60% of the dataset is used for training the neural network, 20% is used for validation and the remaining 20 % is used for testing the recognition accuracy of the proposed approach. The fuzzy-based classification approach achieves 96.95% recognition accuracy. Crisp classification achieves 95% recognition accuracy.

Peera Jarungthai *et al.* [24] proposed a generalized radial basis function Extreme Learning Machine (ELM) with center selection for handwritten character recognition. ELM is single hidden layer neural network for classification tasks. ELM has faster learning rates and it is more efficient than gradient-based neural network algorithms. They proposed a modified and improved version of the ELM known as MELM-GRBF for handwritten character recognition. ELM has more hidden neurons and structure is larger as compared to the gradient-based learning algorithms. MELM-GRBF has higher performance as compared to the sigmoidal, hard-limit, triangular basis and radial basis function. They proposed Histogram of Oriented Gradient (HOG) method for feature

extraction. They proposed a selective centers strategy to improve the performance of the MELM-GRBF. Selective centers strategy uses optimization method to find the optimum centers for GRBF-ELM. Selective centers strategy simplifies the optimization method by removing mutation and crossover processes. They used Thai handwritten characters, Bangla numerals and Devanagari numerals for character recognition task. Thai handwritten characters has 66 classes, Bangla numerals has 10 classes and Devanagari numerals has 10 classes. ELM achieved recognition accuracy of 96.14%, MELM-GRBF has 96.69% recognition accuracy and SCRBF-ELM has 97.89% recognition accuracy on Thai handwritten characters. Recognition accuracies on Bangla numerals are 95.02%, 95.96% and 96.36% for ELM, MELM-GRBF and SCRBF-ELM respectively. Recognition accuracies on Devanagari numerals are 77.67%, 78.35% and 79.30% respectively. The proposed approach has higher recognition accuracies on all three datasets as compared to the traditional ELM and MELM-GRBF methods without centers selection.

Adwait Dixit *et al.* [25] proposed a wavelet based feature extraction and classification scheme for handwritten Devanagari character recognition. They proposed wavelet transform method to extract wavelet statistical features from the image. They used a self-created Devanagari dataset consisting of 2000 handwritten characters dataset for training and validating the artificial neural network (ANN). The proposed method recognize 20 Devanagari characters. Images with pixel size 300×300 are fed as input to the ANN. Binarization is done using Otsu's method. They discussed other preprocessing techniques such as contrast enhancement, filtration etc. Training set includes 70% of the dataset and the rest is used as test set for evaluating the performance of the ANN. The hidden layer size varies from 1-50 to determine the optimum size. They proposed wavelet based feature extraction technique for faster computations and higher accuracy. The proposed approach achieves maximum accuracy of 70%. They proposed multistage feature extraction for further enhancing the recognition accuracy. They further proposed the use of SVM classifiers for classification and handwritten character recognition tasks. The proposed technique is applicable in document analysis.

C. Szegedy *et al.* [26] proposed a deep convolutional neural network in ILSVRC 2014 competition. They used the Hebbian principle and multiscale processing. Network in network method is applied to increase the depth of the CNN and enhance its learning power. The proposed network is 22-layers deep (without counting pooling layers) with 1×1 convolutional layers/projection layers that solves the purpose of dimensionality reduction and reduces the computational limitations. The depth of the network is 27-layers including pooling layers. R-CNN

is proposed for object detection in images. The network performance increases with the increase in network layers as well as increasing the number of units at each layer. The inception architecture uses filters with sizes 1×1, 3×3 and 5×5. CNN image classification and detection success has been driven by the alternative pooling operation in each stage. The alternative pooling path whose output combines with the output of the convolutional layer results in the computational over load. They proposed the use of embeddings to overcome this issue. They consider the use of inception modules at higher layers due to some limitations in current approach. The inception module architecture uses dimensionality reduction technique before the convolution operation to reduce the number of network parameters. With the prudent architecture design, the inception module architecture are more than ten times faster than the similar architecture without inception module. They proposed the use of rectified linear activations. They used the average pooling layer and an additional linear layer before the softmax layer. The use of average pooling layers while discarding the fully connected layers at the end of the network results in increasing the accuracy rate by 0.6%. They used the dropout layer to throw some of the network parameters by 40%. The overall network layers are 100 however; actual number is decided by the machine-learning infrastructure. They trained the network on 1000-classes with DistBelief machine learning system. They used the data provided by the ILSVRC 2014 competition. They achieved a top-5 error rate of 6.67% on validation and test dataset. They perform the object detection challenge in images on a 200-classes dataset. They used six GoogLeNets ensemble for classifying the object region in the image. The mean average precision rate of the detection challenge was 43.9%.

P. Li *et al.* [27] proposed a CNN based confidence estimation method to reduce the errors in character recognition task. They proposed a system that generates the confidence level for each image based on some threshold level. The misrecognized character images are represented as embedded character images in the output. They have discussed about the CNN network and the softmax classifier. Gradient descent algorithm is proposed to update the network parameters with the backpropagation technique. The output of the highest layer is used as a confidence parameter. They have discussed the rejection region theorem that tells the misrecognized characters and helps in minimizing the error. The output of recognition rejection module is one for the misrecognized characters and it outputs zero for the correctly recognized characters. They proposed a TH-OCR system to convert the scanned documents into the digital format. TH-OCR system takes the scanned documents followed by the confidence estimator, which takes the segmented character candidates as the input for further classification into correct or misrecognized characters. They

used the Maxout neural network with the overlap pooling approach. Maxout network has a gradient value of one while optimizing the network parameters during backpropagation. The proposed network is 10 layers deep (not counting the pooling layers). They used the max-pooling approach for subsampling. All three convolutional layers have 100 filters each and the last fully connected layer has 3755 neurons. They achieved 98% accuracy rate in single character recognition. The network was trained on Chinese character dataset using single NVidia TESLA K20 GPU. The training set included about 2.7 million image samples and the error rate is below 2% for single character recognition. The test set has 35,401 Chinese character image samples. Errors in character segmentation can be detected with the 93% accuracy rate. Discarding 19% of the character candidates in the segmentation stage would increase the accuracy rate of the neural network. They suggested to avoid the residual errors during digitalizing the scanned documents. The confidence estimator increased the accuracy rate from 99.0% to 99.90%.

Hadar I. Avi-Itzhak *et al.* [28] proposed a feed-forward neural network based on centroid dithering training process for high accuracy recognition on multi-size and multi-font characters. They divided the study in two parts; the first part performs the recognition task on single size and single font characters. The second part deals with the different font and size of the characters. The feed-forward neural network trained on a single size and single font characters is two layers deep with 20 neurons in the first fully connected layer and 94 neurons in the second fully connected layer. The neural network trained on multi-size and multi-font characters has 100 neurons in the first fully connected layer and 94 neurons in the second fully connected layer. The non-linearity is introduced with the sigmoid function. Although, the network is computationally inefficient but it gives the simple basic idea of the character recognition task. The network is trained with the backpropagation algorithm to minimize the squared-error loss. The dataset has the segmented character images and the first neural network is evaluated on 1.07 million test image dataset. The second neural network trains on a multi size and multi font characters with size varying from 8-32 and the fonts are Arial, AvantGarde, Courier, Typewriter, Times new Roman etc. The second neural network recognition performance is evaluated on 347,712 image samples with font size 8, 12, 16 and 32 each having 28,976 image samples. Before recognition task, the images are preprocessed and normalized. They proposed thresholding method to remove the background noise. The threshold value is kept at 20% of the highest pixel intensity. They discussed the image centroid and radial moment calculation. Conversion of two-dimensional images into vectors are discussed. The centroid dithering process operates on both the datasets, which dither the input

image samples. This process forms the multiple version of the same image sample. The centroid dithering makes the neural network invariant to the scale orientations. They proposed the centroid dithering process to be used during training the neural network only. They discussed the mean squared error and the loss function of the neural network. The first neural network took 430,000 iterations and second neural network took 8,650,000 iterations for training. The accuracy rate for the single single-size and single-font character recognition is 99.99991%, and for the multi-size and multi-font character recognition is 99.99971%.

W. Hu *et al.* [29] have proposed a CTC and Lattice-free MMI (LFMMI) objective function for offline English handwriting recognition. Deep neural networks trained with sequence discriminated method have lower word error rate as compared to the neural networks trained with cross-entropy approach. Connectionist Temporal Classification (CTC) based training methods can handle millions of training data. Principle Component Analysis (PCA) and Convolutional Neural Network (CNN) extracts features from the input image samples. They have compared the performance of the CTC and LFMMI training methods. They discussed the posterior probabilities of both CTC and LFMMI training methods where the posterior probability of CTC is locally normalized and posterior probability of LFMMI is globally normalized. A modified runtime text detector is proposed for mismatch reduction between the offline character training and online evaluation of the PCA-DBLSTM based neural network. De-skewing, de-slanting, and size normalization techniques are adopted for text normalization in images. PCA-DBLSTM based neural network uses filter with 30 pixels for feature extraction, stride is kept at value three pixels. They proposed a horizontal cosine window for smoothing operation. PCA converts 1800 dimensional feature vector to 50 dimensional feature vector. The CTC layer comprises of softmax layer. The PCA-DBLSTM based neural network is trained on total 96 classes. These 96 classes includes 52 case sensitive English characters, 10 numerals and other are the punctuation marks. They have further discussed the language model and decoder of the PCA-DBLSTM based neural network. Another network is CNN-DBLSTM based neural network, where CNN does the feature extraction task in place of the PCA. The character model of the CNN-DBLSTM differs from the earlier model. The character model has CNN layer, two BLSTM layers and a CTC layer at the output. They proposed CNN based VGG network for feature extraction due to its high performance. They evaluate both the networks on two test sets viz. E2E and IAM test set. CNN-DBLSTM based neural network has lower word error rate as compared to the PCA-DBLSTM based neural network. Therefore, they concluded that CNN-DBLSTM outperforms PCA-

DBLSTM for automatic feature learning from the input images. The combined CTC and LFMMI objective function outperforms both CTC and LFMMI individual performances.

K. H. Jin *et al.* [30] proposed a CNN based approach for reducing undesired inverse problems. They proposed an X-ray CT reconstruction. CNN approach effectively reduce the inverse image problems. They discussed the direct inversion and iterative inversion methods. Filtered back projection convolutional neural networks with iterative methods is used for inverse problems. The proposed CNN network consists of convolutional layers, max-pooling layer, batch normalization, skip connection and concatenation. The network consists of multichannel filters. The network uses a residual learning approach where a skip connection learns the difference between the input and the output. Skip connection reduces the vanishing gradient problems. Zero-padding is used after each convolutional layer to retain the image size. Instead of using fully connected layer at the end of the network, they proposed the convolutional layer. Fixed-size discretization kernel reduces the FBP computations. Three datasets are proposed for FBPConvNet performance evaluation and training viz. ellipsoid, biomedical and experimental dataset with data augmentation techniques. They concluded that the proposed CNN network with FBP reconstructs a 512×512 image in less than a second. FBPConvNet trains on NVIDIA GPU.

CHAPTER-3

LANGUAGE TRANSLITERATION WITH RNNs

3.1 Recurrent Neural Networks (RNNs)

RNNs provides a major relief to deal with the large input and output sequences. RNNs has internal states that are used as memory cells to store the previous information. RNNs can predict the next word in a sequence given the previous input word and in a similar way; RNNs can predict the next character in a word using the information of the previous characters. The basic structures of the RNNs can only see a few steps back at the input. They do not have long-term dependencies. RNNs can provide the information at each time step whereas the regular neural networks are unable to do so. RNNs have loops as shown in the following figure:

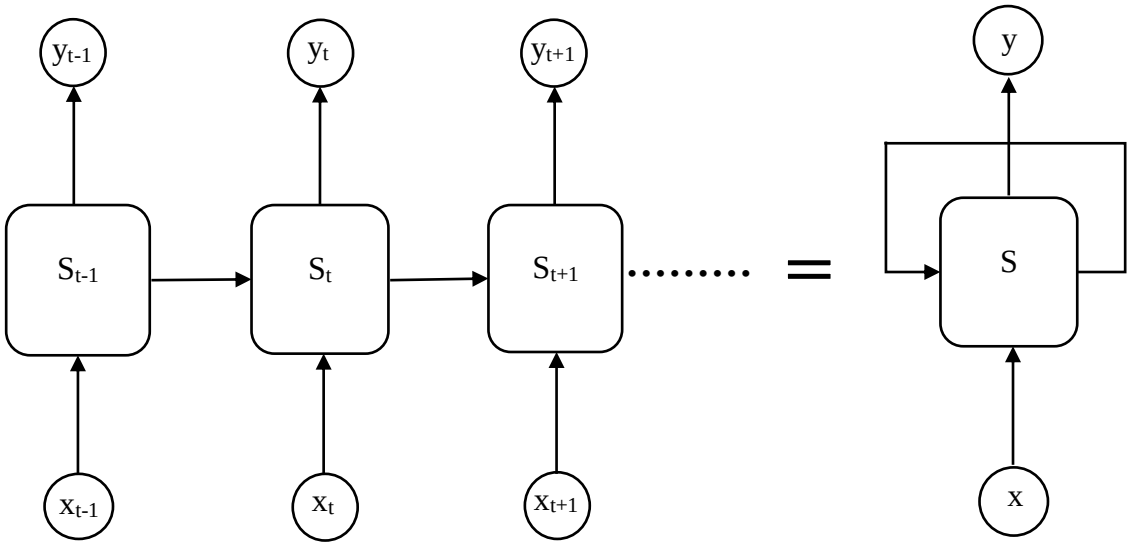

Fig. 3.1: Recurrent Neural Networks

- In the above figure, x_{t-1} is the previous step input, x_t is the input at the current step and x_{t+1} is the next step input.
- S_{t-1}, S_t, and S_{t+1} are the internal states of the network that behaves like a memory of the network. Based on the information provided by the previous hidden states, the network produces the output at each step.
- Where y_{t-1}, y_t, and y_{t+1} are the output at the previous step, current step and at the next step respectively.
- RNNs uses the same parameters at each step just with the different inputs. So, it makes the RNNs computationally efficient.
- Depending upon the task, we may or may not require the output at each time step.

3.2 Limitations of the Regular RNNs

Regular RNNs are unable to handle the sequences with larger length i.e. if we want to use the previous information where the gap between the current step and the previous step is large the RNNs can't handle such sequences practically. Since in RNNs the current output depends upon the previous state of the network, it becomes difficult for the regular RNN to connect the information with the previous steps with a larger gap. Such long-term dependency can cause the vanishing gradient problems.

This Long-Term Dependency of the regular RNN can be solved with the special RNN architectures viz. Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU).

3.3 RNN Applications

RNNs are widely used by the researchers in many Natural Language Processing tasks. Some applications of the RNNs are given as under:

- **Language Modeling**

  In language modeling task, the next word is predicted based on the information from the previous words. The network predicts the next word with the highest probability.

- **Machine Translation**

  Machine translation is the task of translating the input language into another language. The input language is the source language and the output language is known as the target language. For example, translating the English sentences into the Hindi sentences where English is the source language and Hindi is the target language.

- **Speech Recognition**

  RNNs are used in speech recognition where the user can input the data by just speaking to the system and the RNN are capable of producing the desired results.

- **Sentiment Analysis**

  In this task, the RNN analyzes whether the text shows a positive or negative sentiment.

- **Image Captioning**

In image captioning task, the RNNs are capable of generating labels to the images. It can give information about the events in the image. The image is of fixed size for such cases while the output labels have varying length.

- **Chatbots**

With the ever-evolving technologies, Chatbots have become quite popular for answering the queries. The user can ask a question to its personal chatbot and chatbot are capable of answering the user's query.

3.4 Long Short-Term Memory (LSTM) Networks

LSTMs are widely used for processing the input sequences having larger length. LSTMs are used for removing the vanishing gradient problems of regular RNNs. LSTM is a special architecture of the RNNs that consists of the memory cells. LSTMs have the capability of learning from the 1000 steps back in time without any loss of information. The repeating memory cells of the basic LSTM structure has a cell state to which we can add or remove the information. The gate controls this information, which is the sigmoid layer. The value between zero and one decides the amount of information that can be added to the current cell state. Another sigmoid layer is known as the forget gate layer that removes the information from the current cell state that we do not want to keep. A third sigmoid layer decides the amount of information to output in the current step. However, there are the different versions of the LSTM networks with slight variations. LSTM memory cell is as shown in the following figure:

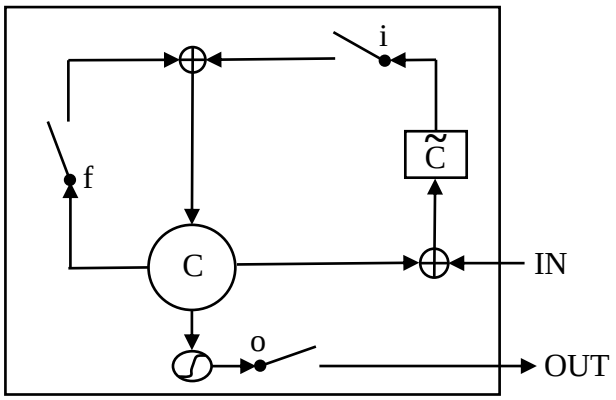

Fig. 3.2: LSTM Memory Unit

3.4.1 LSTM Advantages

- LSTM based networks solve the vanishing gradient problems of the regular recurrent neural networks.
- LSTM networks can remember the previous information even for over one thousand steps back in time.
- LSTM networks can handle the noisy input sequences very effectively.
- LSTM networks require very little tuning of network parameters.
- LSTM networks constantly perform the backpropagation through time within memory cells to reduce the error in each step.

3.4.2 LSTM Applications

LSTM networks perform all the tasks that a regular RNN can perform. Although some applications of the LSTM network are given as below:

- LSTM based networks are used for controlling the robots.
- It is used for time series prediction where the future output is predicted based upon the information from the previous outputs.
- LSTM based networks are used for learning the grammar of various languages.
- It is used for human action recognition and sign language translation.
- It is used for handwriting recognition tasks.
- LSTM based networks are also used for the semantic parsing where the natural language is translated into a machine-readable form.
- LSTM use is also found in rhythm learning and music composition.

3.5 Gated Recurrent Unit (GRU) Networks

GRU is another special kind of RNNs that is recently proposed by Cho et al. in 2014 mainly for the applications of machine translation. GRU outperforms the LSTM networks in the tasks of computational time and parameters update. GRU has a sophisticated activation function that uses the gating units in its network. Unlike LSTM, GRU does not have any separate memory cells to control the information flow inside each gating unit. In contrast to the regular RNNs, GRU can remember the previous information for longer time steps as well as it can add the new information

on the top of current information. No important features are removed by the GRU while updating the current state. This feature allows the GRU based networks to perform the error backpropagation easily with multiple short-cut paths. GRU does not have any control on the degree of information shared by each step. However, GRU has a control on the amount of information flow from the previous time step while computing the new candidate activation. GRU based RNN network outperforms the LSTM based RNN network on the JSB Chorales, MuseData, and Piano-Midi music datasets. However, the choice between the LSTM and GRU based networks depends upon the dataset and the problem domain. GRU consists of the recurrent units that have the update and the reset gates as shown below:

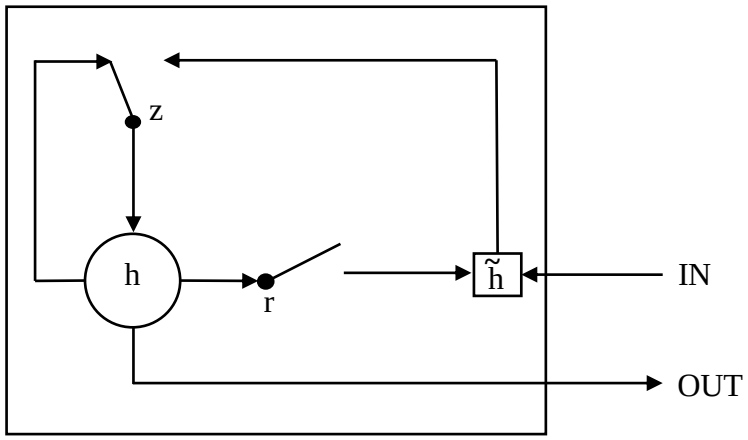

Fig. 3.3: Gated Recurrent Unit

Where 'r' is the reset gate, 'z' is the update gate, and 'h' is the activation.

- The following equation represents the activation h_t^j:

$$h_t^j = (1- z_t^j)\, h_{t-1}^j + z_t^j \, \tilde{h}_t^j \, .$$

- The following equation represents the update gate:

$$z_t^j = \sigma \, (W_z \, x_t + U_z \, h_{t-1})^j \, .$$

- The following equation computes the reset gate r_t^j:

$$r_t^j = \sigma \, (W_r \, x_t + U_r \, h_{t-1})^j \, .$$

3.6 Natural Language Processing

Natural Language Processing has gained the attention of many researchers that deal with the automatic text generation and understanding of the human languages. NLP makes the way for the tasks such as Language Transliteration, text generation etc. Natural Language Processing mainly involves the study of the following three areas:

- **Syntax:** It involves the arrangement of words and punctuation that forms a complete sentence.
- **Semantics:** It involves the study of the relation between the words and their structures. Its study helps to generate the mapping between the words.
- **Pragmatics:** Pragmatics deals with the study of context-dependent meaning.

The major complexity of the Deep Neural networks for the NLP tasks is that it deals with the input and output sequences having a fixed length. This problem of fixed-dimensionality is solved with the RNNs that can process the input and output sequences having varying length. LSTM and GRU based RNNs are the best networks for the NLP tasks. LSTMs and GRUs are able to learn the sequences with long-term dependencies. NLP tasks require a large amount of data to train the RNNs with higher accuracy. Human languages often involve many ambiguities and still, humans are capable of making meaning out of the complex and ambiguous sentences. However, humans are not good at understanding all the rules that a natural language follows. The goal is to make the machine understand all the rules that a particular language follows so that machines can extract the proper meaning out of complex and ambiguous sentences. With the help of RNNs and better algorithms, the natural language processing task has gained huge success in developing the robust end-to-end systems.

3.7 Language Transliteration

Language Transliteration is the process that takes one language as the input and converts/transliterate it into another language. The sequence-to-sequence learning has paved the way for many researchers for the Language Transliteration task. Language Transliteration takes a single word of the sentence and transliterate it into another language. This is known as the word level transliteration. The Language Transliteration models based on the probability and statistics has higher accurate results as compared to the models that are just based on the grammar rules.

The Language Transliteration model based on the statistics requires the abundant amount of training data to produce high accuracy results for the Language Translation/Transliteration task. The basic steps for the Language Transliteration task are given below:

i. The complete sentence is divided into smaller parts or several words, which makes transliteration easier.
ii. Next, all the possible transliteration for a particular word is considered. This helps to create the real-life scenario to see how human beings translate each sentence.
iii. The third step is to give scores for each possible output sentence.
iv. The sentence with the highest probability score is considered the correct transliterated output.

There are some limitations with the statistics based translation/transliteration approach. It is difficult to maintain and requires the huge amount of training data to build a robust language translation/transliteration system. For example, if we want to translate Norwegian to Devanagari then, first we have to translate Norwegian to English as an intermediate step and then perform the Devanagari translation. Otherwise, a lot of computations and a large training data is required to generate state of the art results. RNNs plays a major role as a self-learning transliteration system.

3.7.1 RNN Based Encoder-Decoder

RNN based encoder and decoder is recently used for the language translation problems and sequence-to-sequence prediction. Such models can handle the variable length input and output sequences. Each character is assigned a specific integer value, which is the process of mapping the characters to integer values. The RNNs are able to perform the encodings by itself. Therefore, a complete sentence is just the sequence of numbers and RNNs can differentiate between the sentences based upon such a sequence of numbers. Encoder-Decoder model consists of two RNNs, where the first RNN network performs the encoding part and the second RNN network performs the decoding part. In the decoding task, the sequences of numbers are converted back into the original target sentences. Each sentence is represented by its own set of integer values. This approach doesn't need to know the human language rules. The most successful application of this approach is the Google's Neural Machine Translation System. The Google's NMT System consists of the LSTM network with eight encoder and decoder layers and results in a 60% reduction in language translation errors on several language pairs.

CHAPTER-4

METHODOLOGY

4.1 CNN Based Character Recognition Model

This dissertation is based on the convolutional neural network approach used for Devanagari handwritten character recognition in images. The three basic steps involved in the character recognition are pre-processing, feature extraction and recognition. The following figure shows the character recognition model:

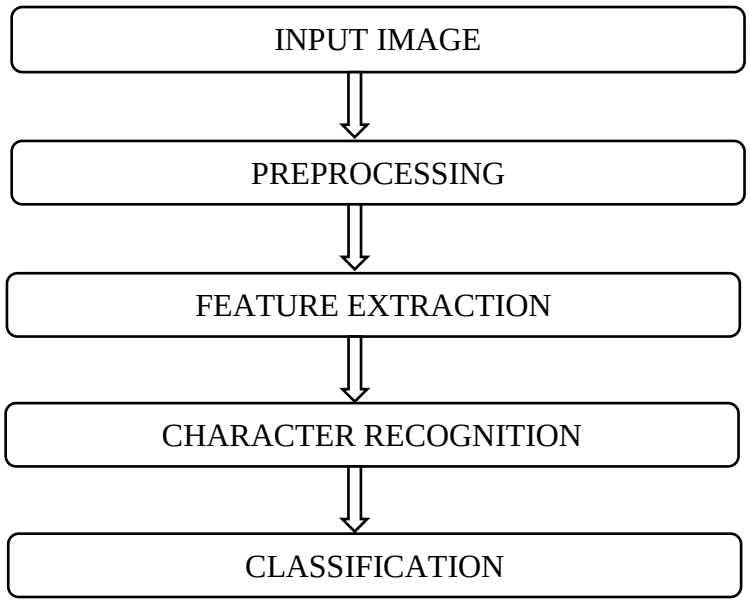

Fig. 4.1: Character Recognition Model

4.2 Devanagari Handwritten Character Dataset

The proposed Devanagari handwritten character dataset is used for recognition and classification task. The dataset includes 102,138 PNG images of size 32×32 with white characters and black background. The dataset includes images of 36 consonants, 10 numerals 12 vowels, and one non-character class. Recognition of the handwritten Devanagari characters is somewhat difficult because each person inscribes the same character in a slightly different way, which depends upon the writer's mood, writer's style of writing a certain character, the speed of writing etc. Since there are 36 consonants and 12 vowels in Devanagari language and each consonant can be combined with every vowel making 12 different forms with the same consonant. The following table shows the consonants in combination with the vowels:

Table 4.1: Consonant and Vowels Combination

म + आ = मा	म + इ = मि	म + ई = मी	म + उ = मु	म + ऊ = मू	म + ओ = मो
म + औ = मौ	म + ए = मे	म + ऐ = मै	म + अं = मं	म + अः = मः	म + अ = म

4.3 Image Preprocessing

The various preprocessing techniques carried on Devanagari handwritten character dataset is given as under:

- **Color to Grayscale Conversion**

  Grayscale images are useful for identifying the important edges and other features such as light intensity, contours of an image. Color images are also useful in some cases where objects need to be separated from one another based on the color e.g. finding orange from the bowl of apples where segmenting colors is useful. Grayscale images are less complex than colored images and require the lesser number of computations as compared to the color images. In our case, we proposed grayscale images.

- **Image Shuffling**

  Images are shuffled which randomizes the data for better learning of the neural network. When the data is shuffled; the neurons are provided different inputs at each epoch and the neural network can use better weights (amplitude of connection between each neuron) by learning from previous epoch result to give better accuracy. Shuffling makes output result more stable.

- **Dataset Splitting**

  The Devanagari handwritten character dataset is split into training and test set. The training set includes 85% of the data set and the test set includes 15% of the dataset. The neural network learns the features from the training dataset and evaluates the neural network model on the test dataset. The training variables such as Batch size is set to 128 i.e. training occurs in batches of 128 samples randomly taken from the training dataset, Number of classes are set to 59 since there are 36 consonants, 10 numerals, 12 vowels and 1 non-character class and the network is trained with different epoch values. In general, epoch means the number of times the neural network model will go through the entire training dataset to learn different features from the images. The higher batch size results in less time for training the neural network but this could

degrade the accuracy, so, an optimal value needs to be selected after experimentation, which achieves higher accuracy.

- **Data Normalization**

Data normalization is a method to reduce the redundancy in the data. The independent variables and other features of the image samples gets standardize by this process. In this paper, the image samples (pixel values) are converted to 32-bit floating-point numbers. The image samples (pixel values) are normalized in the range 0 to 1 by dividing the image pixel intensities by 255. This is a sort of data binarization method. Data normalization reduces the computational cost and time.

4.4 Convolutional Neural Network (CNN)

In this dissertation, a Convolutional Neural Network model is proposed for handwritten character recognition. A sequential model is defined that consists of the number of layers such as Convolutional layer, Max-Pooling layer, Dropout layer, Flatten layer etc. It is necessary to tell the convolutional neural network model about the input shape it is required to follow. The information about the input shape is given only in the first convolutional layer and the following layers can presume the input shape. The convolutional neural network is better than the regular neural network. In regular neural network, the neurons in a layer is connected to all the neurons in the previous layer e.g. consider an image of size 32×32 having 3 color channels, so, the number of neurons required in the first layer would be 32×32×3=3072, if the image size is larger; the number of parameters will increase quickly. This type of connection in a regular neural network could lead to overfitting problems. In a convolutional neural network, the neurons in a layer are connected to only a small local region in the previous layer and thereby reducing the parameters of the neural network.

The first layer of the network is known as the input layer, which takes the pixel values of the image as the input. The last layer of the network i.e. the output layer is a fully connected layer, each neuron in this layer is connected to all other neurons in the previous layer. This layer computes the class probabilities. The output layer consists of 59 neurons since there are total 59 classes in the dataset. The layers between input and the output layer are known as the hidden layers and the network can have either a single or multiple hidden layers. These hidden layers extract the different features of the input image. CNN architecture is as shown in the figure on next page:

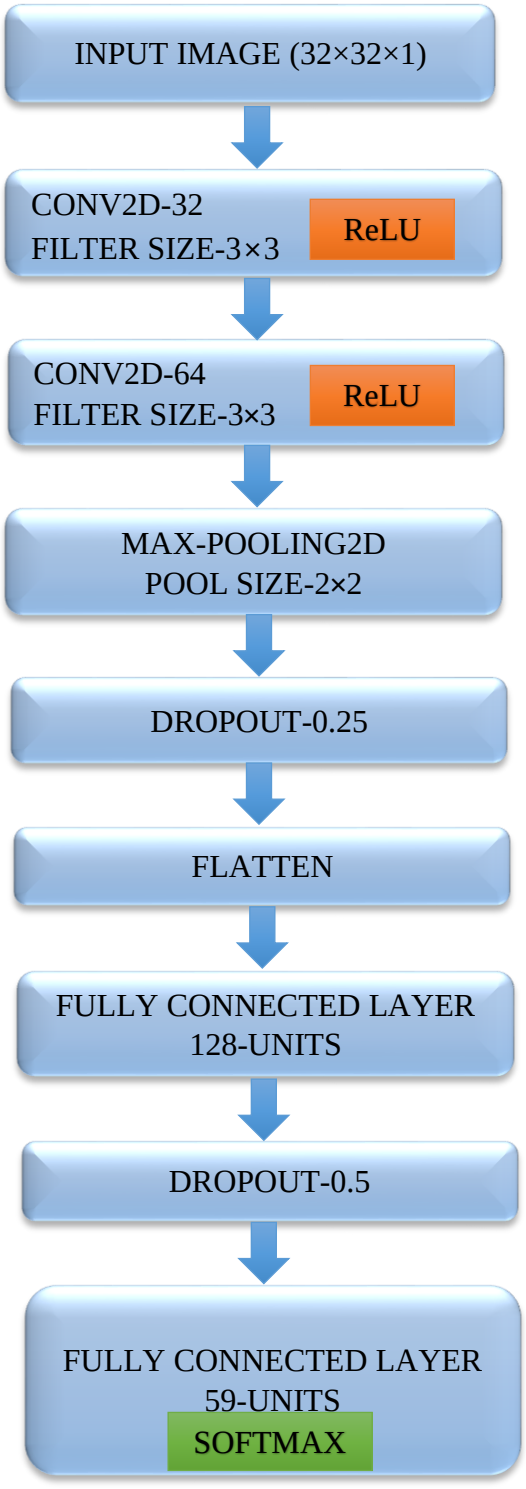

Fig. 4.2: Convolutional Neural Network Model

4.4.1 First Convolutional Layer

The first convolutional layer consists of 32 filters with the filter size of 3×3 (i.e. 3 pixels width and 3 pixels height). These filters can learn the features from the inputs images. The filter size determines the width and height of the convolution operator and defines the size of the local region to which the neurons are connected in the input images. The purpose of the filters is to extract or learn every individual features from the input images and the various filters applicable for this purpose are sharpening filters, blurring filters, edge detection filters, mean filters, median filters and many more. The input image has size 32x32x1 and the kernel/filter size is 3x3 so each neuron in the first convolutional layer will have 3x3x1= 9 connections/weights (with one bias parameter) in the local region of the input image. The convolutional layer convolves the input with the filters that move horizontally and vertically and computes the dot product between the weights and the input and then it adds a bias parameter to generate the output. The amount of filters in the convolutional layer determines the depth of the convolutional layer. The output volume of the first 2D-convolutional layer has size 30x30x32. Therefore, there are 32 neurons in each depth column of the first convolutional layer each having nine connections (with additional one bias parameter) to the local region in the input. Moreover, the convolutional neural networks make use of the parameter sharing which reduces the number of parameters in the convolutional layer.

Parameter sharing allows all the neurons of the same filter to share same weights and biases which reduces the overall amount of parameters used in the neural network. Without parameter sharing, the number of neurons in the first convolution layer will be 30x30x32 = 28,800 neurons each having nine connections (plus one bias parameter) to the input region. Together this results in 28,800x10 = 288,000 parameters in the first convolutional layer that could lead to overfitting. The parameter sharing reduces this much amount of parameters to 32x3x3x1 = 288 (plus 32 biases) = 320 parameters/weights. Parameter sharing causes all neurons corresponding to one filter in each depth column to share the same weights and biases.

4.4.2 Rectified Linear Unit (ReLU)

ReLU is a non-linear/activation layer that comes instantly after a convolutional layer. The ReLU activation layer is used to introduce non-linearity to the neural network. The convolution layer performs the linear operation by computing the elementwise multiplication and addition therefore, a non-linear layer is required to add non-linearity to the network. ReLU activation layer has higher

efficiency in terms of the computations and has mathematical form; f(x) = max(0, x) i.e. it thresholds the values to zero, where x is the input to the neuron. It gives the output x if x is positive and threshold the negative values to zero. The ReLU layer solves the vanishing gradient problem that causes the neural network to learn input features very slowly and thereby, vanishing gradient can degrade the accuracy of the neural network. Activation layer decides the firing of the neuron whenever it comes across a strongly responding new feature of the input image.

4.4.3 Second Convolutional Layer

The output of the first 2D convolutional layer serves as an input to the second convolutional layer. The second convolutional layer has 64 filters with size 3×3. Since the output of the first convolutional layer is having dimensions (30 x 30 x 32); each neuron in the second convolutional layer is having 3x3x32 = 288 weights/connections (with one additional bias parameter) connected locally to the input region. The connections are local in space with size 3x3 and full along the input depth i.e. 32. The output volume of the second 2D-convolution layer has dimensions 28x28x64. With the parameter sharing method of the convolutional layer, the total number of the parameters are 64x3x3x1=576 (plus 64 bias parameters) = 640 unique parameters for the second convolutional layer. Thereby, all 28x28 = 784 neurons in each depth column are using the same parameters, which reduces the computational complexity of the neural network. The output of the second convolution layer is passed through the ReLU activation layer to add non-linearity to the output of the second convolutional layer.

4.4.4 Max-Pooling Layer

The pooling layer is added to the neural network after the convolutional layer. The pooling layer performs the down-sampling operation that reduces the number of parameters in the neural network. Pooling layers allow the neural network to detect the same feature even with changes in their orientation or absolute position. It allows the network to learn the relative position of the features. Max-Pooling is used with the pool/filter size 2x2. The max pooling operates on the 2D input space with pool size 2x2 and gives the maximum value within the 2x2 input space as the output i.e. the maximum value within the 2x2 input space is the output of the max pooling operation. The output of the second convolutional layer serves as the input to the max-pooling layer. The input to the max pooling layer is having dimensions 28x28x64 (i.e. length x width x

depth) and with a pool size of 2x2, the output volume of the max pooling layer is 14x14x64. As it can be seen, the output volume is reduced to half while preserving the depth. The amount of parameters gets reduced which lessens the computational cost and shields the neural network from overfitting. The pooling operation does not affect the information in the image.

4.4.5 Dropout

Dropout is another layer employed in the neural network to prevent the problems of overfitting. Overfitting is the problem that arises when the neural network has a large number of parameters for the corresponding observations. The neural network starts learning the noise and errors rather than true features. The overfitting in the neural network to perform better on the training samples but produces less accuracy on the test samples. The dropout rate is set to 0.25 after max-pooling layer; it is the rate by which some of the input connections are dropped. Some input neurons are dropped randomly during training the neural network and only during training the network. By dropping some of the input neurons randomly during training the other neurons are required to take the place of the dropped neurons and make predictions. It allows the neural network to learn the true relationship among the input features and prevents the neural network from learning the specific weights and biases. The outcome is that the neural network can now better generalize and classify the training data. Although the dropout causes the neural network to train slower with two to three times slower than the neural network without dropout layer, the neural network with dropout layer can give higher accuracy by tackling the problem of overfitting.

4.4.6 Flatten Layer

A flatten layer is used after the first dropout layer whose purpose is to convert the input arrays to the one-dimensional feature vector. The fully connected layer in the neural network requires the one-dimensional feature vector as the input. Flatten layer accomplish this task. The output of the max- pooling layer after applying the dropout is fed to the flatten layer which converts the output of the max pooling layer to a 1D feature vector and that serves as an input to the first dense/fully connected layer of the neural network.

4.4.7 First Fully Connected Layer

In this paper, the first dense/fully connected layer consists of 128 neurons where each neuron in this layer is connected to every other neuron in the previous layer. The first fully connected layer

extracts the high-level features of the input image. The output of the first dense layer is followed by the ReLU activation function to decide the neuron's activation and to add non-linearity to enhance the output results. A dropout rate of 0.5 is also introduced after the first dense layer, which implies that 50% of the input neurons will be randomly dropped during training to avoid the neural network from learning specific weights and biases.

4.4.8 Second Fully Connected Layer (Output Layer)

The second fully connected layer or output layer has 59 neurons corresponding to the 59 classes of the input dataset (i.e. 36 consonants, 10 numerals and 13 vowels). The second dense layer takes the input from the first dense layer that are activation maps of the high-level features and produces a 59-dimensional feature vector. Each value of this 59-dimensional feature vector represents the probability that the input image belongs to a particular class. The highest probability value represents the true class of the input image. The second fully connected layer uses the softmax activation function. Softmax function generates the output scores/probabilities for each class between zero to one and the total sum of all the output scores equals to one. The output of the softmax classifier are normalized probabilities. Softmax classifier uses a loss function known as categorical cross entropy loss. This loss function is used for the multiclass classification problem. Softmax tries to minimize the loss function and maximize the score function of the correct class.

4.4.9 Adam Optimizer

Adam optimizer is used to update network weights and bias values. Adam is an appropriate optimizer for the neural network with larger parameters, it requires less amount of memory and very effective in terms of the computations. The optimizer minimizes the loss function by updating the network weights and bias values. The optimizer plays a very important role in learning and calculating the best parameters for the neural network, which could produce the desired output results. Adam optimizer is suitable for avoiding the problems of slow learning rate and calculates adaptive learning rate for each parameter of the neural network. The Adam optimizer follows the underlying equation:

$$\theta_t \longleftarrow \theta_{t-1} - \alpha.\hat{m}_t \Big/ (\sqrt{\hat{v}_t} + \varepsilon)$$

Where the initial parameter vector, step size, bias-corrected first moment estimate, bias-corrected second raw moment estimate and epsilon are represented by θ_{t-1}, α, $\hat{m}_t$, $\sqrt{\hat{v}_t}$ and ε respectively.

4.5 Proposed Algorithm for Devanagari Handwritten Character Recognition

Step 1: Import image samples.

Step 2: Read image and check whether the input image is RGB or Grayscale; if the input image is RGB then convert it into Grayscale.

Step 3: Store image samples into an empty dictionary.

Step 4: Randomly shuffle the image samples.

Step 5: Divide Devanagari handwritten character dataset into training and test set. The training set contains 88,500 image samples and the test set contains 13,638 image samples. Save the training set and test set as a numpy array.

Step 6: Convert training and test sets into 32-bit floating-point integers.

Step 7: Binarize the training and test sets by dividing the image samples (pixel values) by 255. Now, the intensities of each image sample will be in range zero to one.

Step 8: One-hot encode the training and the test labels.

Step 9: Design CNN model for handwritten character recognition and classification in images. The convolutional and the pooling layers perform the feature extraction task and the last fully connected layers perform the recognition and the classification task. The CNN architecture is as shown in Fig. 3.2.

Step 10: Train the CNN model on the training dataset. Set the batch size to 128 i.e. training occurs in batches of 128 image samples. Train the CNN model for 40 epochs.

Step 11: Calculate loss or error with categorical cross-entropy. This error is the difference between the predicted label and the true label. The following equation represents this loss:

$$J = -\frac{1}{N}\left(\sum_{i=1}^{N} y_i \cdot \log\left(\hat{y}_i\right)\right)$$

Where N is the number of classes, y_i is the ground-truth class probabilities, $\hat{y}_i$ is the predicted probabilities.

Step 12: Optimize the network parameters with the Adam optimizer. Set the learning rate value at 0.001, β_1 value at 0.9, β_2 value at 0.999, epsilon (ε) value at 10^{-8} and decay rate is set to zero.

Step 13: Evaluate the network performance or accuracy on the test dataset.

Step 14: Calculate the output scores or class probabilities with the softmax classifier. Softmax classifier generates normalized probabilities. The following equation represents the softmax function:

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$

Where j = number of columns in the input K-dimensional vector and i = 1,……..,K.

Step 15: Save the Convolutional Neural Network model after it is trained on the training dataset.

4.6 RNN Encoder-Decoder Based Language Transliteration Model

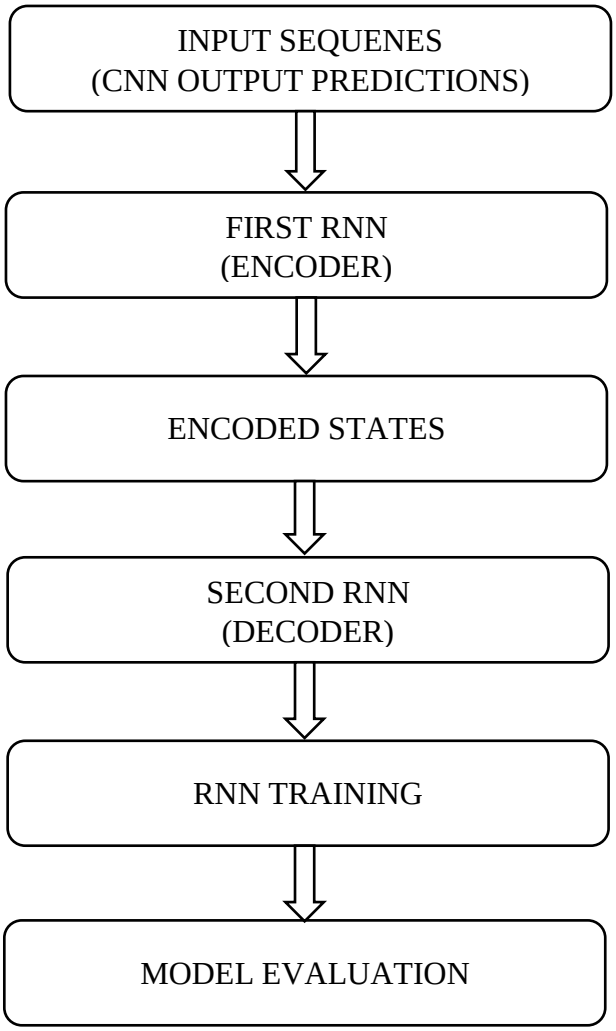

Fig. 4.3: Language Transliteration Model

4.6.1 Proposed Algorithm for Language Transliteration

The following steps are used for Language Transliteration task using RNN encoder-decoder based neural network:

Step 1: Import the input sequences. The input sequences are the CNN output predictions from the character recognition and classification task. The CNN output is stored in a pickle file.

Step 2: Find the maximum length of the input sequences.

Step 3: Append zeros to the input sequences up to the maximum length (as found in Step 2) to map the variable length input sequences to the fixed length vectors.

Step 4: Import the target sequences.

Step 5: Create two empty arrays, one to store target words and another to store the characters. Append each target text to an empty array.

Step 6: Split the words into characters.

Step 7: Check whether a particular character is already stored in an array. If not then append each character to another empty array.

Step 8: Sort the characters stored in an array to arrange them in an ordered way.

Step 9: Create a dictionary of tokenized characters i.e. assign each individual character a unique integer value.

Step 10: Find the number of target characters and the maximum length of the target text.

Step 11: One-hot encode the target data with the shape as (number of input sequences, maximum length of the target text, number of target characters).

Step 12: Define the input sequences to process it through the encoder model.

Step 13: Design the encoder model using LSTM layer and set the number of LSTM memory cells as desired.

Step 14: Discard the encoder model output while retaining only the internal states of the encoder.

Step 15: Design the decoder model using the LSTM layer and set the internal states of the encoder as the initial state of the decoder.

Step 16: Set up the decoder LSTM layer to return the full output sequences as well as the internal states of the decoder.

Step 17: Set up the decoder dense layer using the softmax activation function.

Step 18: Define the model to turn the encoder input data and the decoder input data into the decoder target data.

Step 19: Train the RNN encoder-decoder model. Set the batch size to 32 and train the model for 400 epochs.

Step 20: Optimize the network parameters using Adam optimizer. Set the α value at 0.001, β_1 value at 0.9, β_2 value at 0.999, epsilon value at 10^{-8} and decay rate is set to zero.

Step 21: Calculate the network loss using categorical cross entropy.

Step 22: Save the RNN encoder-decoder model after it is trained on the dataset.

4.7 Algorithm for Devanagari Character Segmentation in Images

Devanagari word image samples are segmented into the characters. The following steps are used for the Devanagari character segmentation task:

Step 1: Import image samples (Devanagari word images).

Step 2: Check whether the input image sample is an RGB or a Grayscale image. If it is an RGB image, convert it into the Grayscale image.

Step 3: Convert the grayscale image into the binary image.

Step 4: Invert the binary image.

Step 5: Calculate the row-wise sum of all the pixels. Remove the Shirorekha from the Devanagari words using a threshold value.

Step 6: Label the connected components in image samples.

Step 7: Measure the properties of the image regions and apply the bounding boxes on each character of the Devanagari word.

Step 8: Extract each character as defined by the bounding boxes.

Step 9: Save the extracted Devanagari character image samples.

CHAPTER-5

RESULTS AND DISCUSSION

5.1 Experimental Results of the Devanagari Handwritten Character Recognition

In the proposed approach, the performance of the recognition system is evaluated on the Devanagari handwritten character and numeral dataset that contains 88,500 training images, and 13,638 test images each of size 32×32. The dataset consists of 36 consonants, 10 numerals, 12 vowels and 1 non-character class. We set the batch size at 128 image samples, and CNN is trained for 50 epochs. This batch size and the epoch value is selected after experimentation that achieves the highest accuracy. The recognition accuracy of the convolutional neural network model presented in this paper is 98.74%. The recognition accuracy of the CNN model remains approximately constant after 50 epochs.

The overall time taken for image reading, computations, training, and for evaluating the performance of the convolutional neural network model is approximately 6.72 hours. We implement the CNN architecture using Python code on a personal laptop with Intel(R) Core(TM) i5 2.30 GHz processor, 4.0 GB RAM, and 64-bit Operating System.

Table 5.1: Numerals Dataset

Numerals	Training Dataset (each)	Test Dataset (each)	Total (each)
०, १, २, ३, ४, ५ , ६, ७, ८, ९	1500	200	1700

Table 5.2: Vowels dataset

Vowels	Training Dataset (each)	Test Dataset (each)	Total (each)
अ, आ, इ, ई, उ, ऊ, ए, ऐ, ओ, औ, अं, अः	1500	268	1768

Table 5.3: Consonants Dataset

Consonants	Training Dataset (each)	Test Dataset (each)	Total (each)
क, ख, ग, घ, ङ, च, छ, ज, झ, ञ, ट, ठ, ड, ढ, ण, त, थ, द, ध, न, प, फ, ब, भ, म, य, र, ल, व, श, ष, स, ह, क्ष, त्र, ज्ञ	1500	200	1700

The accuracy and loss graphs for the Devanagari Handwritten Character Recognition and Classification are given on the next page.

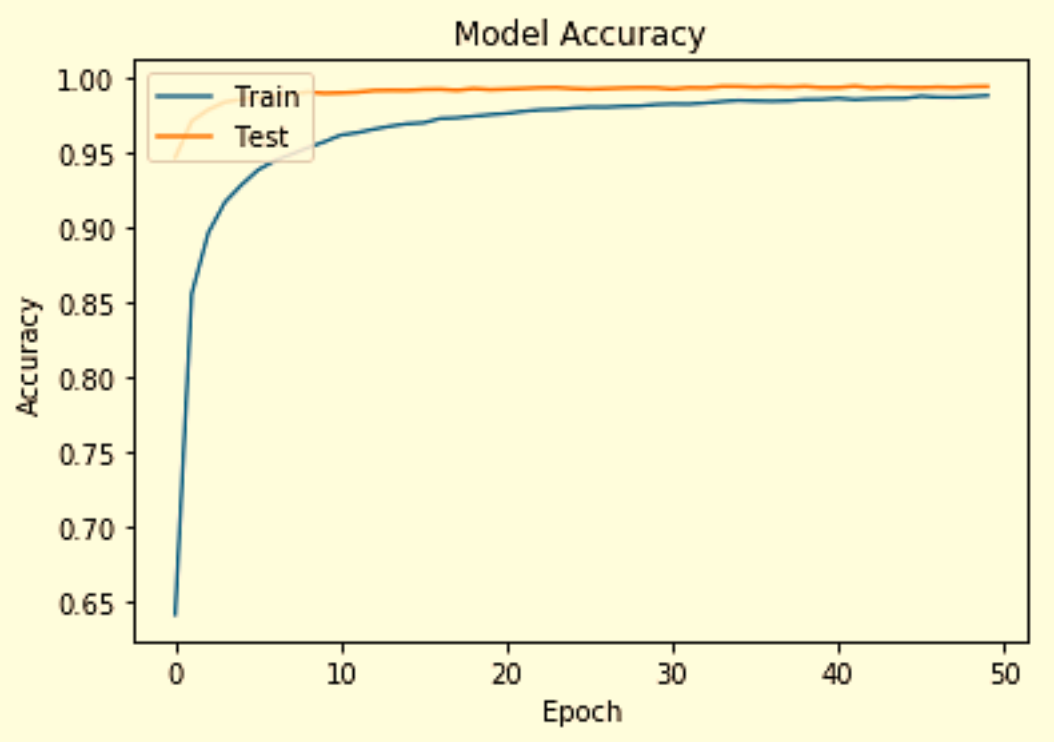

Fig. 5.1: Offline Character and Numeral Recognition Accuracy Curve With CNN- Model

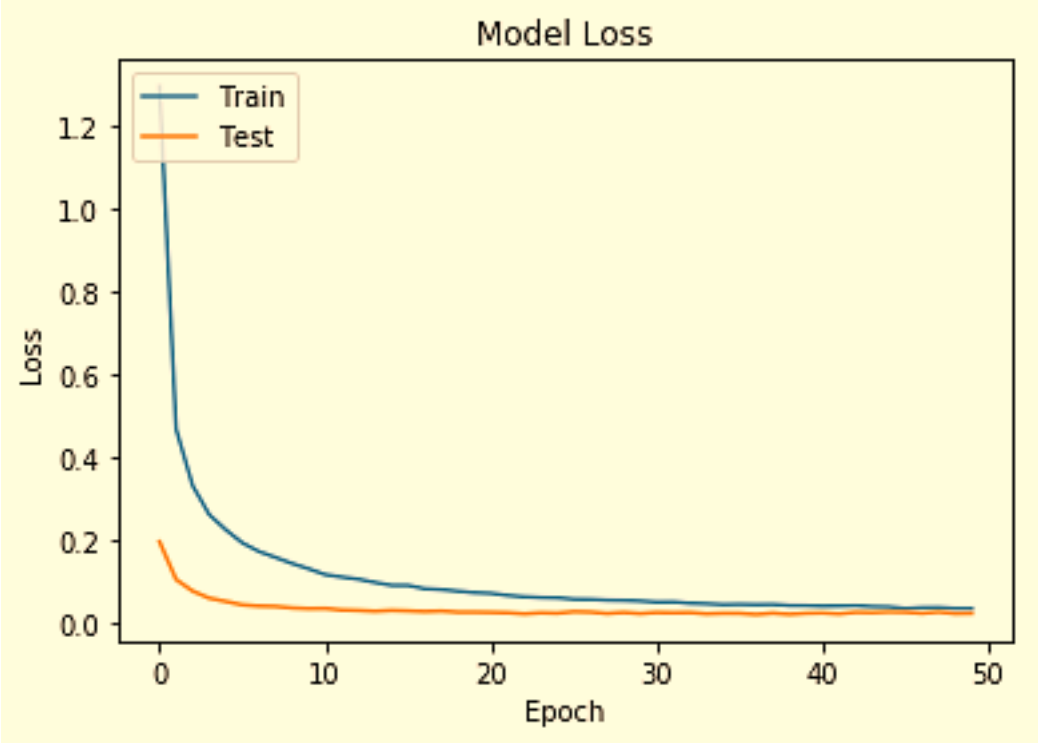

Fig. 5.2: Offline Character and Numeral Recognition Loss Curve With CNN- Model

5.2 Experimental Results of the Character Segmentation

We have segmented the Devanagari words into the characters and some of the results are as shown in the following figures:

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.3: Results on First Devanagari Word

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.4: Results on Second Devanagari Word

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.5: Results on Third Devanagari Word

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.6: Results on Fourth Devanagari Word

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.7: Results on Fifth Devanagari Word

(a) Original Input Image

(b) Binarized Image

(c) Inverted Image

(d) Image with Bounding Box

(e) Segmented Characters (Image Dimensions = 32×32)

Fig. 5.8: Results on Sixth Devanagari Word

5.3 Experimental Results of the Language Transliteration

In the proposed approach for the Language Transliteration task, we prepared a database consisting of 3000 Devanagari words and their corresponding transliterated labels. We segment the Devanagari words of our dataset into the characters. We then used our trained CNN model to generate predictions on our Devanagari segmented characters. For the Language Transliteration task, we provide these CNN output predictions on the Devanagari characters as the input sequence to the RNN Encoder-Decoder based Language Transliteration model. The target sequence for the RNN Encoder-Decoder based Language Transliteration model is the corresponding labels that we have created for our 3000 Devanagari words dataset. The following tables demonstrate the experimental results for the Language Transliteration task with the different RNNs:

Table 5.4: Experimental Results with the SimpleRNN, LSTM and GRU

No. of Hidden Units/Memory Cells	Epochs	Training Accuracy (%)			Test Accuracy (%)		
		SimpleRNN	LSTM	GRU	SimpleRNN	LSTM	GRU
128	400	19.61	27.20	27.24	15.53	17.43	17.72
256	400	21.55	27.24	27.14	15.87	17.70	18.76
512	400	22.55	27.64	27.17	16.39	18.25	19.00

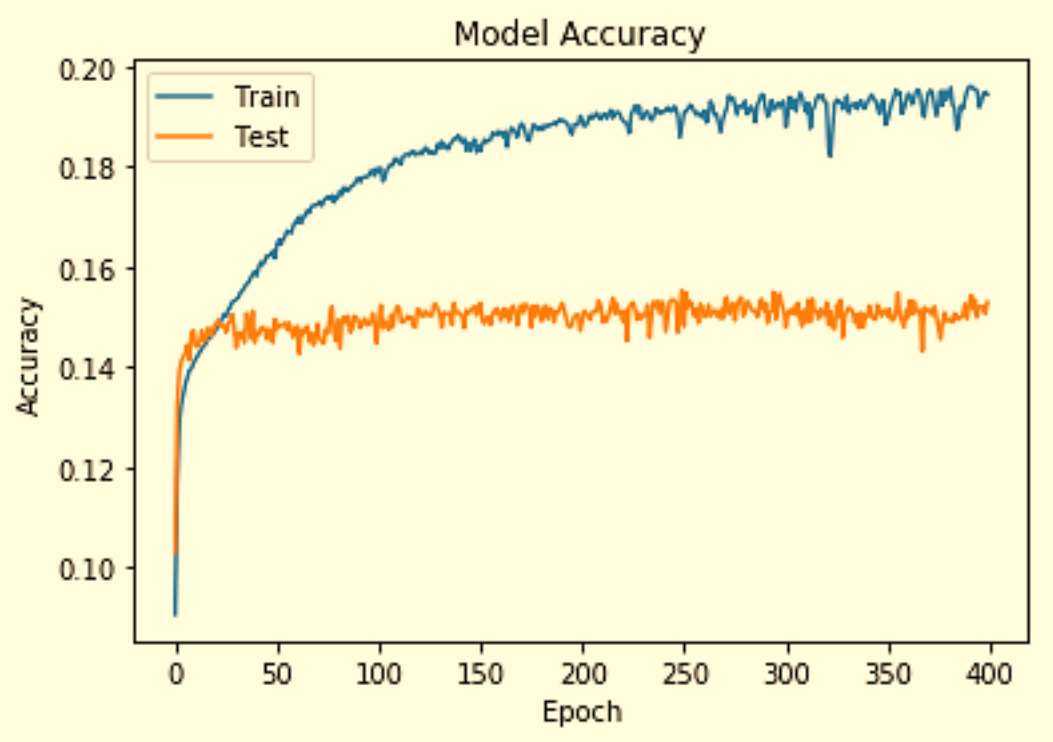

Fig. 5.9: Training and Testing Accuracy Curve with SimpleRNN (128 Hidden Units)

Fig. 5.10: Training and Testing Loss Curve with SimpleRNN (128 Hidden Units)

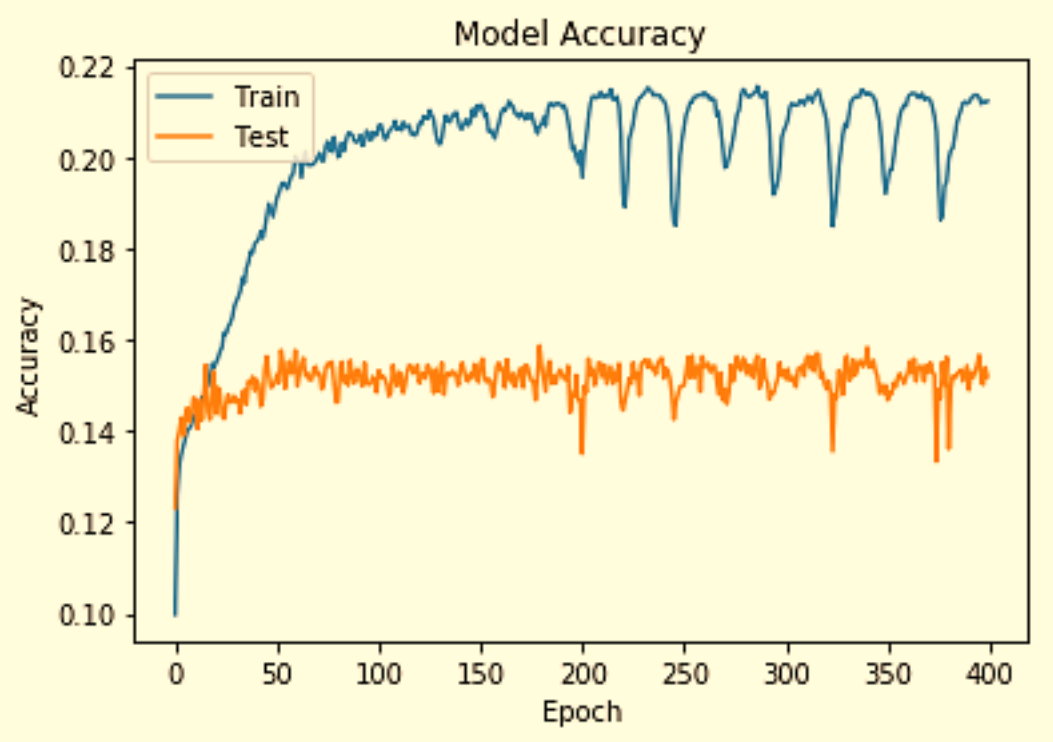

Fig. 5.11: Training and Testing Accuracy Curve with SimpleRNN (256 Hidden Units)

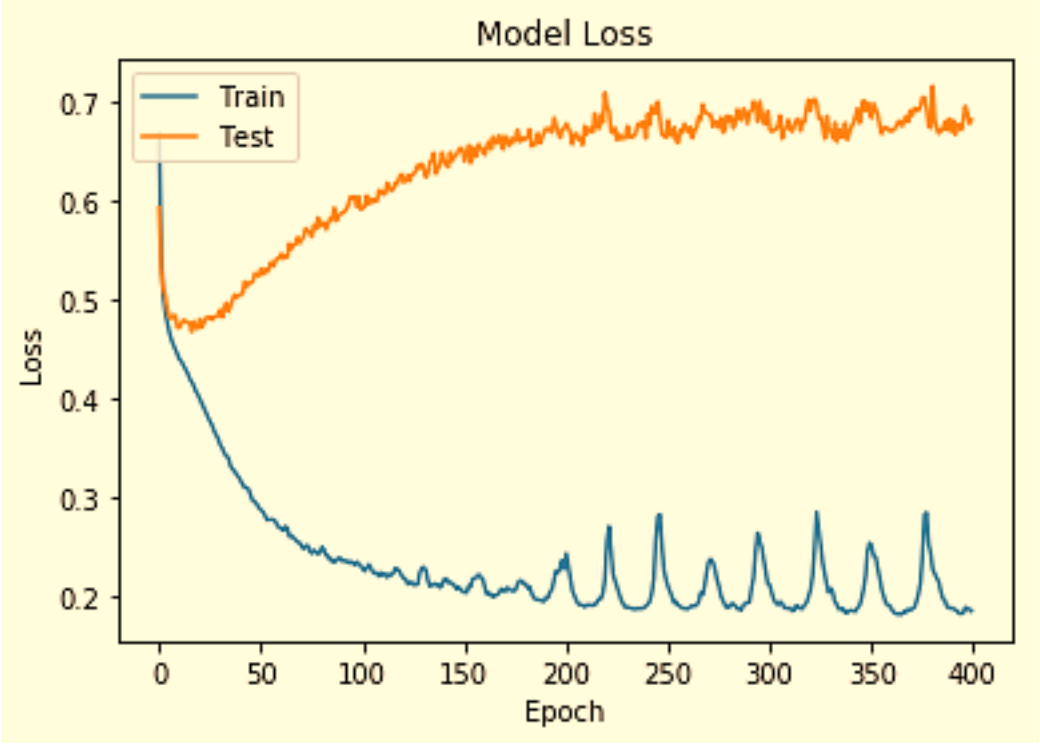

Fig. 5.12: Training and Testing Loss Curve with SimpleRNN (256 Hidden Units)

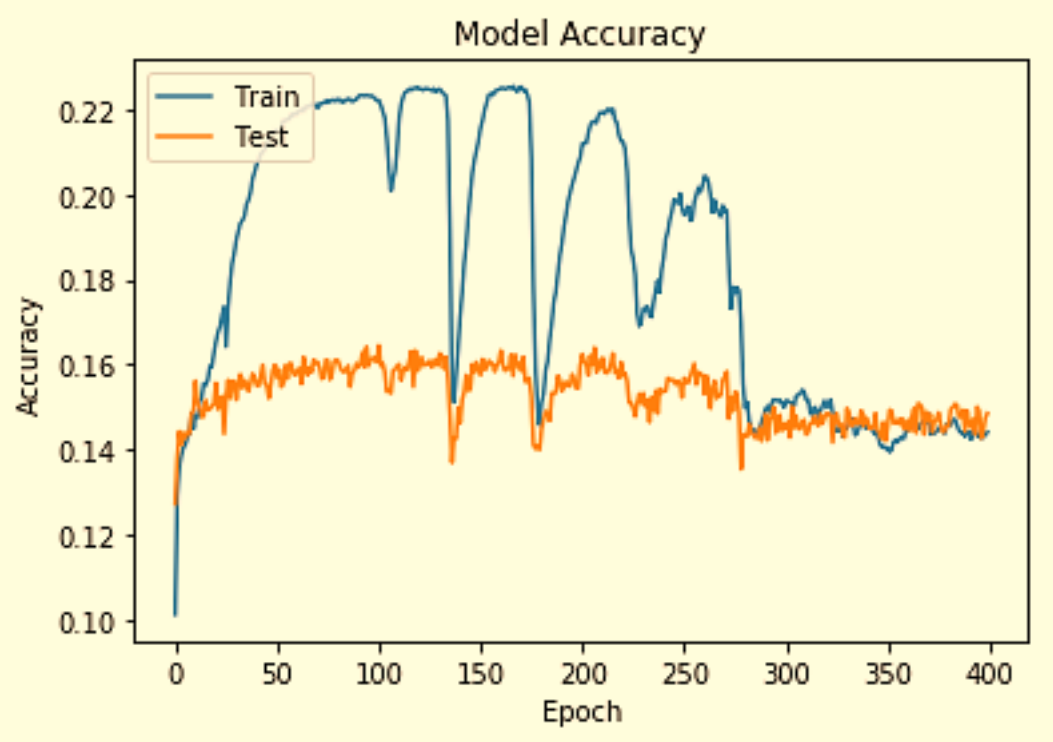

Fig. 5.13: Training and Testing Accuracy Curve with SimpleRNN (512 Hidden Units)

Fig. 5.14: Training and Testing Loss Curve with SimpleRNN (512 Hidden Units)

We have also evaluated the performance of the Language Transliteration model with SimpleRNN (512-Hidden Units) using L1 and L2 regularization techniques to avoid exploding gradients problem and to avoid the saturation of the results. In the Adam optimizer settings, we have used clipnorm value equals to 1 and clipvalue is set to 0.5. The training accuracy and testing accuracy achieved by executing the model for 400 epochs are 19.11% and 16.92% respectively. The accuracy and loss graphs are given as under:

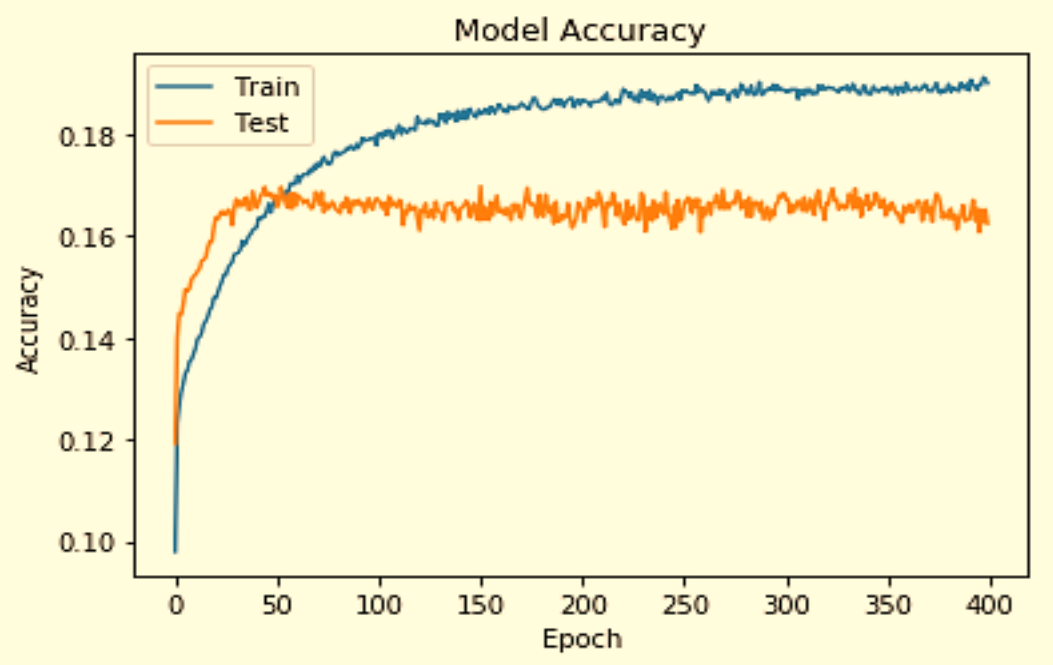

Fig. 5.15: Training and Testing Accuracy Curve with SimpleRNN (512 Hidden Units) and using L1 and L2 Regularization Methods

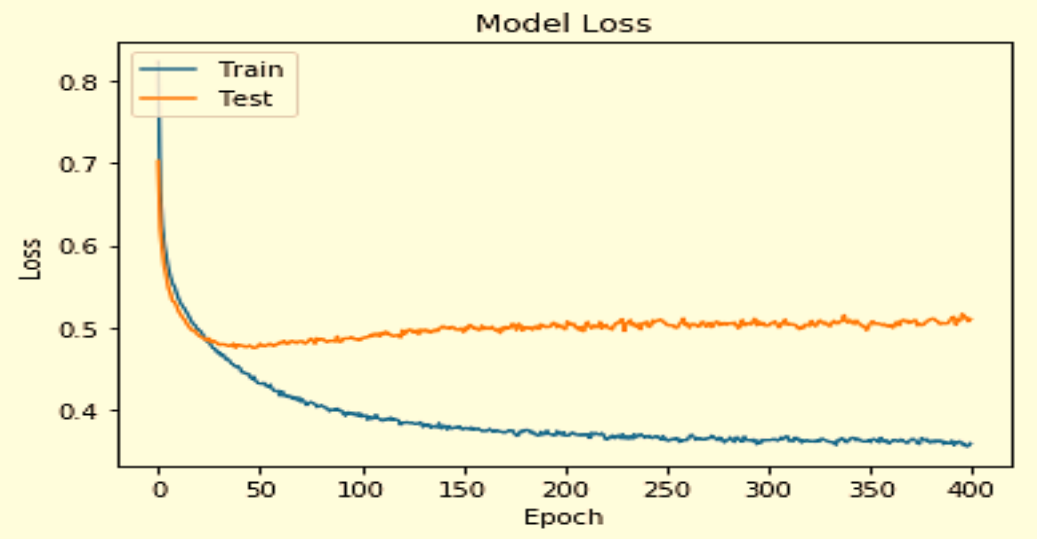

Fig. 5.16: Training and Testing Loss Curve with SimpleRNN (512 Hidden Units) and using L1 and L2 Regularization Methods

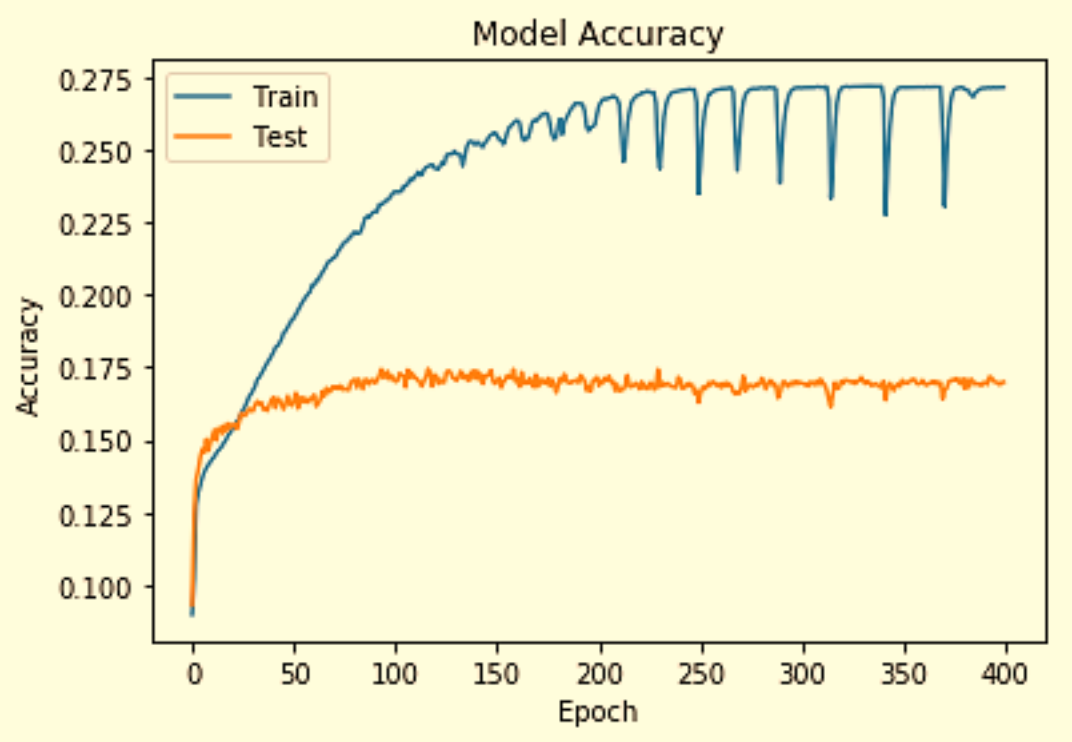

Fig. 5.17: Training and Testing Accuracy Curve with LSTM (128 Memory Cells)

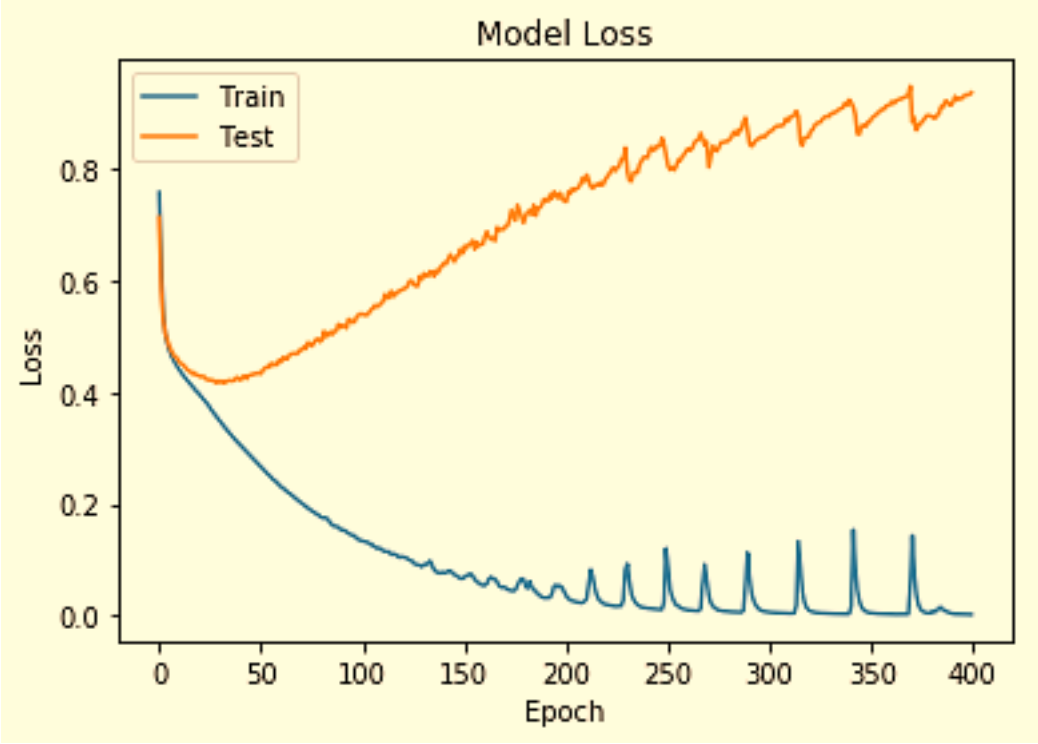

Fig. 5.18: Training and Testing Loss Curve with LSTM (128 Memory Cells)

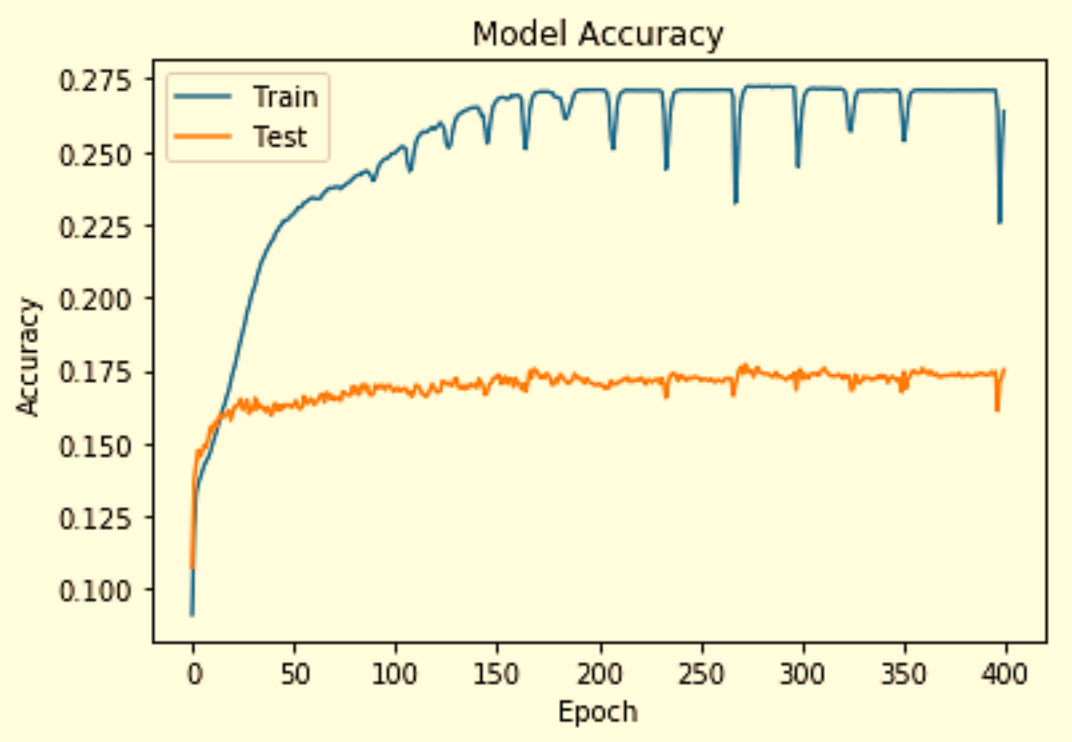

Fig. 5.19: Training and Testing Accuracy Curve with LSTM (256 Memory Cells)

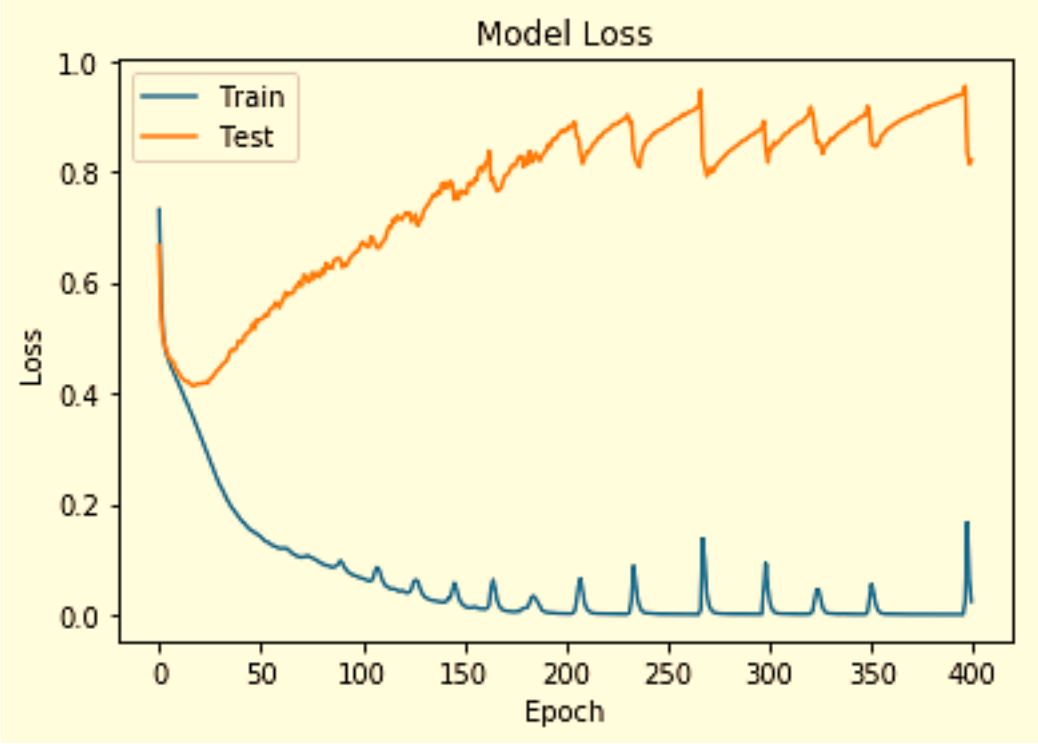

Fig. 5.20: Training and Testing Loss Curve with LSTM (256 Memory Cells)

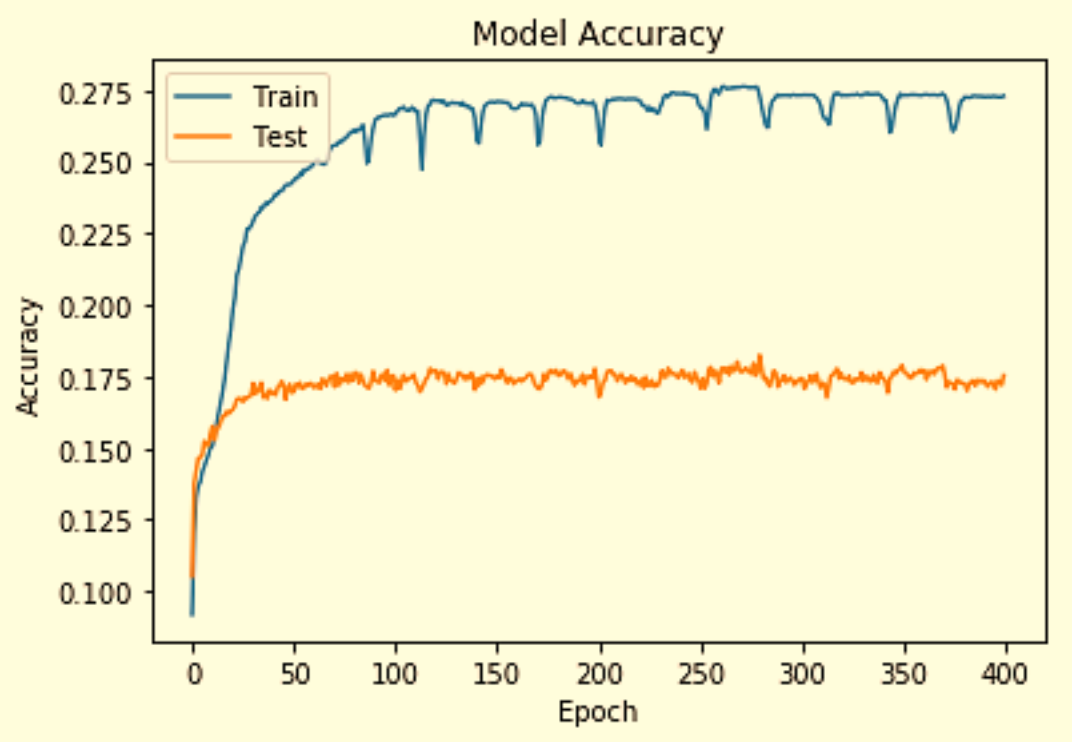

Fig. 5.21: Training and Testing Accuracy Curve with LSTM (512 Memory Cells)

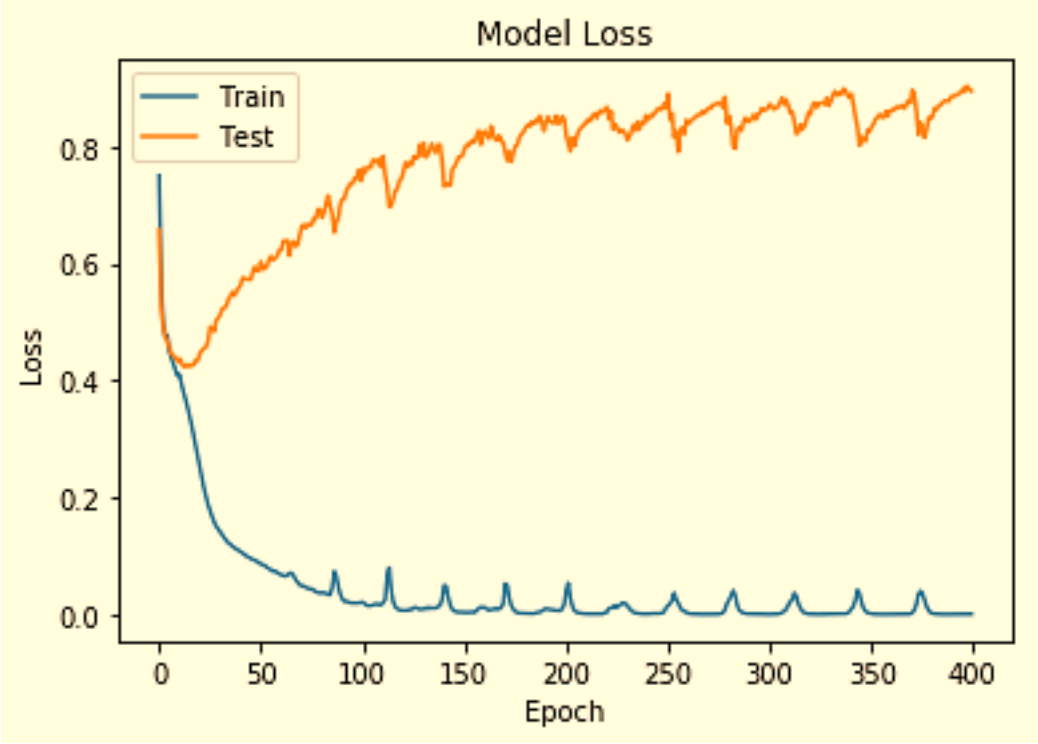

Fig. 5.22: Training and Testing Loss Curve with LSTM (512 Memory Cells)

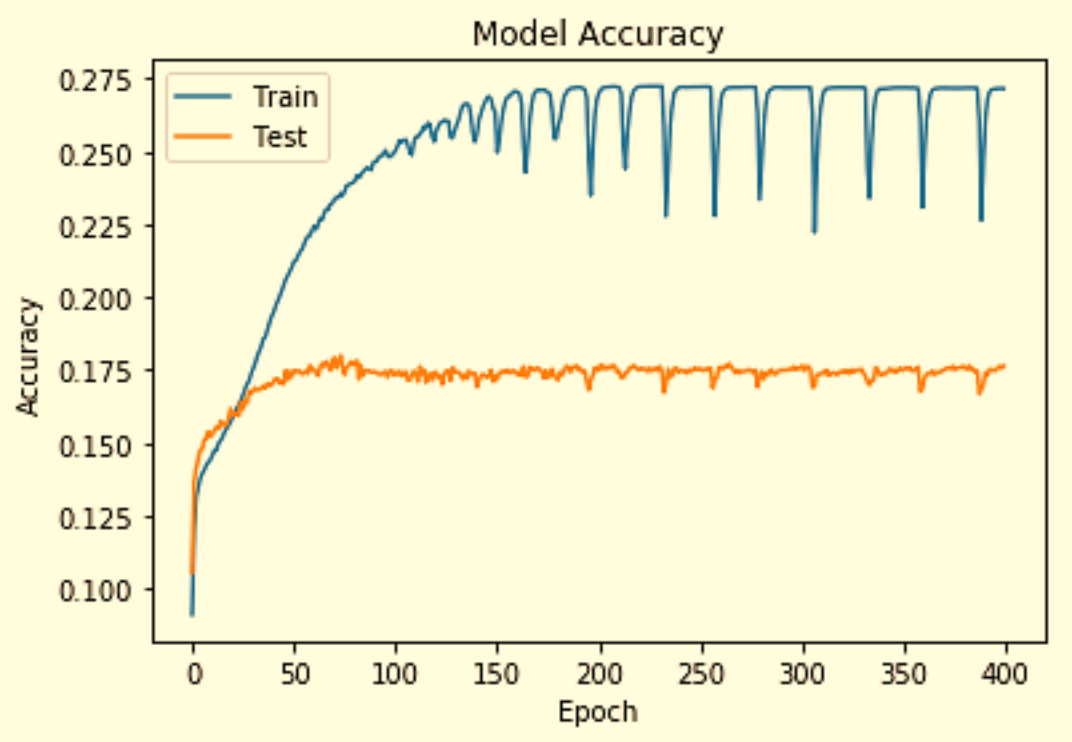

Fig. 5.23: Training and Testing Accuracy Curve with GRU (128 Hidden Units)

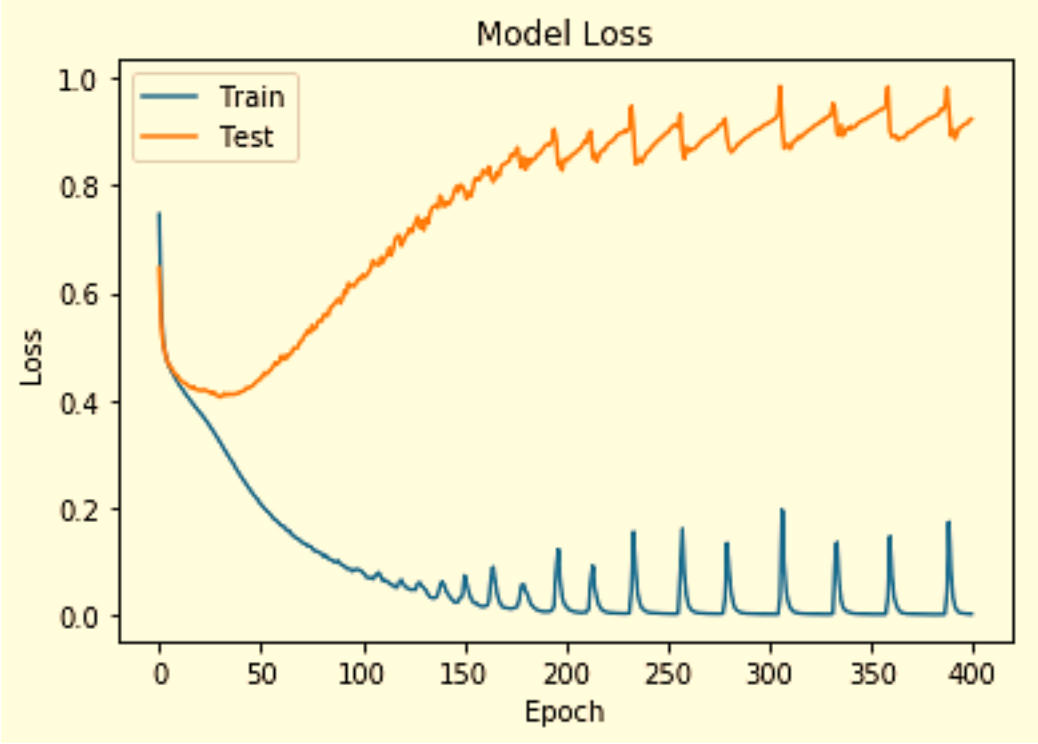

Fig. 5.24: Training and Testing Loss Curve with GRU (128 Hidden Units)

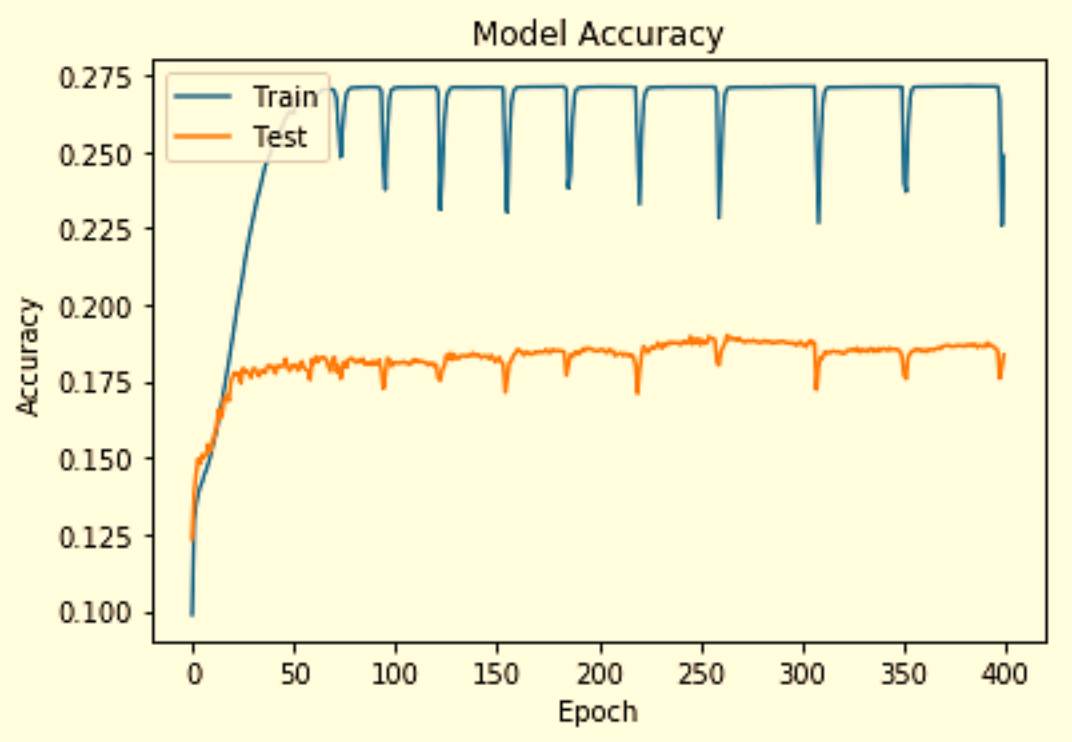

Fig. 5.25: Training and Testing Accuracy Curve with GRU (256 Hidden Units)

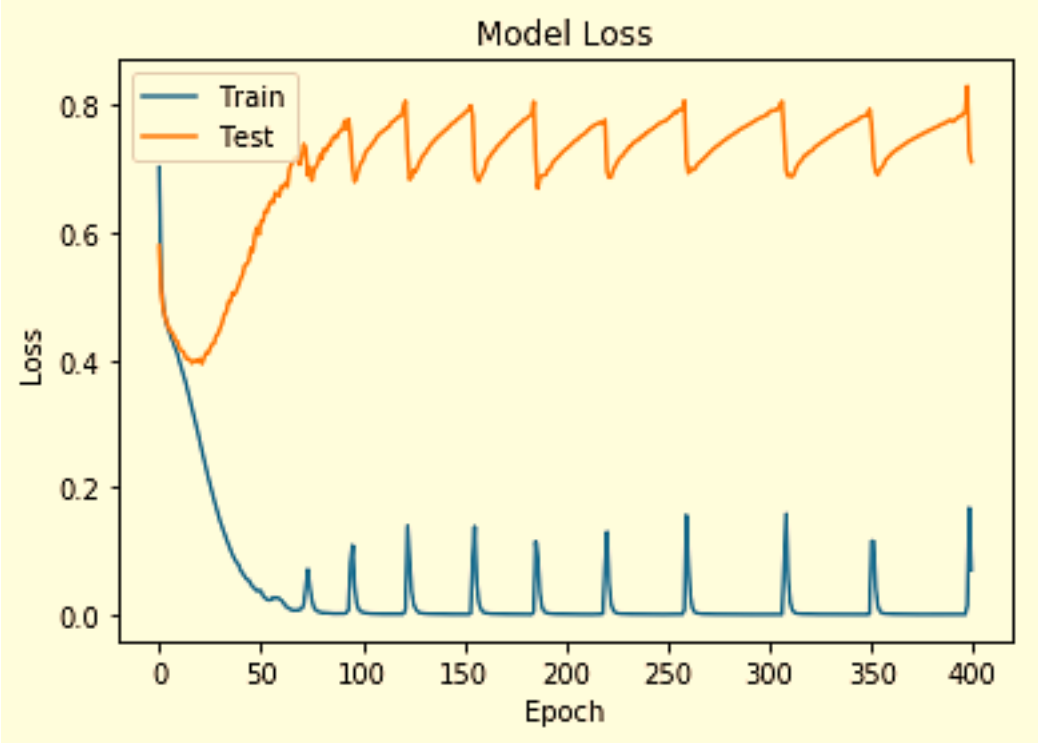

Fig. 5.26: Training and Testing Loss Curve with GRU (256 Hidden Units)

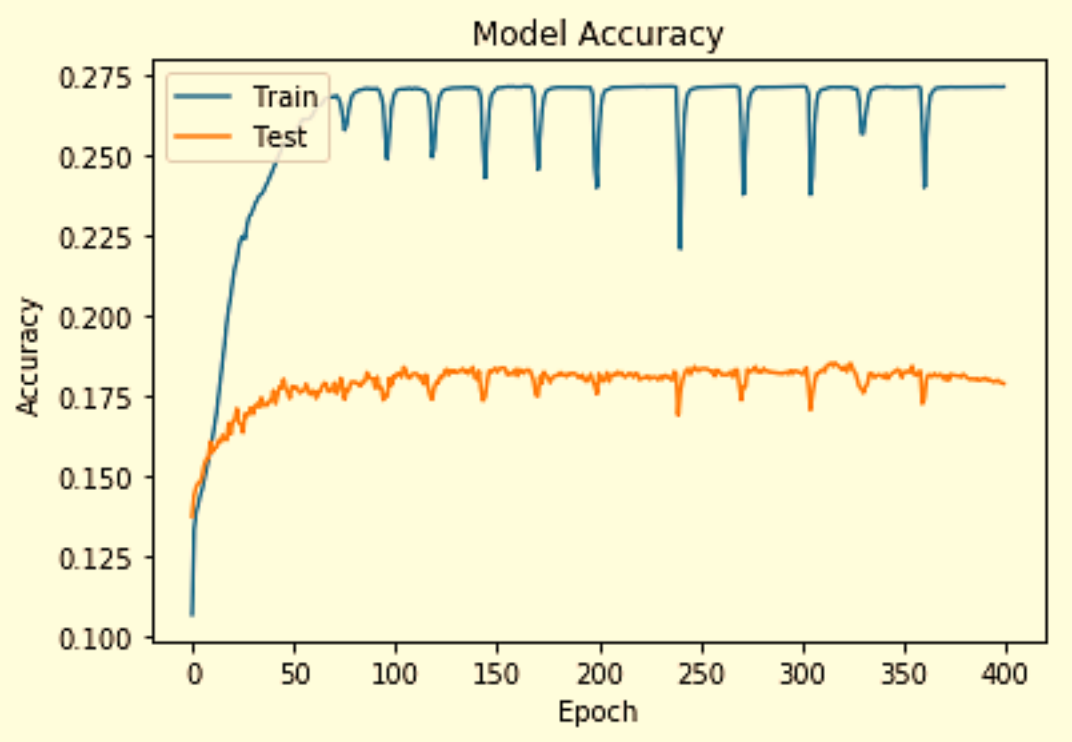

Fig. 5.27: Training and Testing Accuracy Curve with GRU (512 Hidden Units)

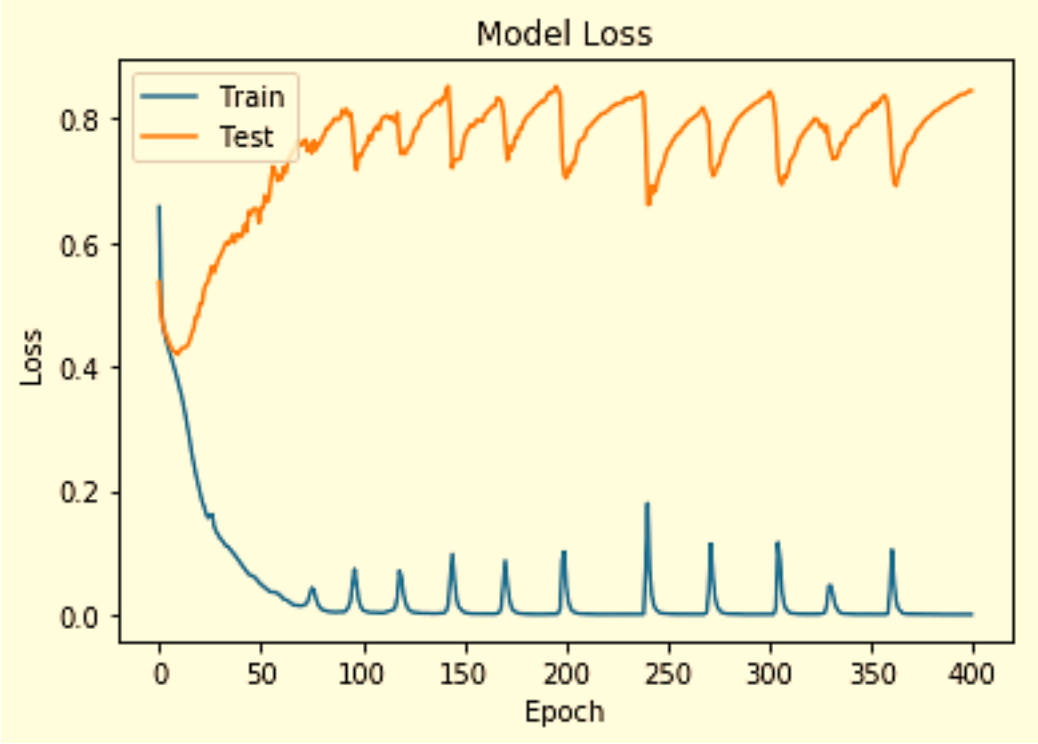

Fig. 5.28: Training and Testing Loss Curve with GRU (512 Hidden Units)

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In this dissertation, we present a CNN based handwritten character recognition system, which can automatically extract and learn features from the images. The feature extraction and character recognition tasks are performed on 59-classes of Devanagari dataset. In the proposed approach, we use the Softmax classifier and error is computed with the categorical cross-entropy. We used the Adam optimizer, which enhances the overall recognition performance of the neural network. The CNN based character recognition system achieves the state-of-the-art accuracy of 98.74% on the Devanagari Handwritten Characters and Numerals dataset.

Next, we prepare a database consisting of Devanagari words and their corresponding labels. We then apply our segmentation algorithm to segment the Devanagari characters of our database. We used our CNN model trained on the Devanagari Handwritten Character dataset to generate predictions on our segmented characters. The CNN output predictions are further used for the Language Transliteration task.

The RNN based Encoder-Decoder Model performs the Language Transliteration task. The CNN output predictions are used as an input sequence to the RNN Encoder-Decoder model and target sequence is the corresponding labels of our Devanagari database. We evaluate the performance of the Language Transliteration system with the different RNNs such as SimpleRNN, LSTM, and GRU. Each network is evaluated with the different number of hidden units. The experimental results are as shown in Table 5.4.

The accuracy obtained for the language transliteration task is low. The reason behind this low accuracy is the small dataset size. For the character recognition task, we have a dataset of 88,500 training image samples, which includes unique multiple sets of each Devanagari character image sample so the performance is better and the character recognition system has an efficiency of 98.74%. In the language transliteration task, we have a dataset consisting of only 3000 image samples of Devanagari words and their transliterated labels. Moreover, our 3000 Devanagari words database do not have the unique multiple sets of each image sample, which is another reason for a low accuracy of the language transliteration system. Our segmentation algorithm is unable to segment the connected characters that further causes the low accuracy rate of the language transliteration system. Some examples of such connected characters are given below:

The segmentation algorithm segments the matra ृ (ri) from the Devanagari words in a similar way whether it appears below the character like कृ or it appears at the top of a character like र्ण. While in both the cases, the transliteration, and pronunciation is different. In the case, कृ the matra ृ (ri) is pronounced and transliterated after the character क whereas in the case र्ण it is pronounced and transliterated before the character ण. This lowers the accuracy of the language transliteration system. The other problem with the character segmentation algorithm is that it segments the characters ग, ण, and श as if they are two separate characters whereas they are a single character. This further lowers the accuracy rate of the language transliteration system. Segmentation output of these three characters are given below:

6.2 Future Scope

The character recognition performance of the proposed CNN model can be further enhanced by increasing the dataset size, by increasing the depth of the convolutional neural network, and by increasing the number of kernels. We could use other models except CNN for the character recognition system. The proposed approach can also be implemented for the printed character recognition task. Although the accuracy of the language transliteration system is low, it can be further improved by increasing the dataset size and eliminating the problems as mentioned in the conclusion section.

REFERENCES

[1] D. Wang, K. Mao and G. W. Ng, "Convolutional Neural Networks and Multimodal Fusion for Text Aided Image Classification," *20th International Conference on Information Fusion* Xi'an, pp. 1-7, 2017.

[2] A. Graves et al., "A Novel Connectionist System for Unconstrained Handwriting Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855-868, 2009.

[3] R. Tavoli and M. Keyvanpour, "A Method for Handwritten Word Spotting Based on Particle Swarm Optimisation and Multi-Layer Perceptron," *IET Software*, vol. 12, no. 2, pp. 152-159, 2018.

[4] T. He, W. Huang, Y. Qiao and J. Yao, "Text-Attentional Convolutional Neural Network for Scene Text Detection," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2529-2541, 2016.

[5] J. Dolinsky and H. Takagi, "Analysis and Modeling of Naturalness in Handwritten Characters," *IEEE Transactions on Neural Networks*, vol. 20, no. 10, pp. 1540-1553, 2009.

[6] B. Shi, X. Bai and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2298-2304, 2017.

[7] F. Yan, Y. He, O. Ruwase and E. Smirni, "Efficient Deep Neural Network Serving: Fast and Furious," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 112-126, 2018.

[8] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv*:1409.1556, 2014.

[10] Xu Chen, "Convolution Neural Networks for Chinese Handwriting Recognition," Stanford University.

[11] Yuhao Zhang, “Deep Convolutional Network for Handwritten Chinese Character Recognition,” Stanford University.

[12] U. Pal, N. Sharma, T. Wakabayashi and F. Kimura, “Off-Line Handwritten Character Recognition of Devanagari Script,” *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007),* vol. 1, pp. 496-500, 2007.

[13] Seema A. Dongare, Dhananjay B. Kshirsagar, Snehal V. Waghchaure, “Handwritten Devanagari Character Recognition using Neural Network,” *IOSR Journal of Computer Engineering,* vol. 16, pp. 74-79, 2014.

[14] Gaurav Kumar, Pradeep Kumar Bhatia, “A Detailed Review of Feature Extraction in Image Processing Systems,” *Fourth International Conference on Advanced Computing & Communication Technologies,* 2014.

[15] R. Plamondon and S. N. Srihari, “Online and Off-line Handwriting Recognition: A Comprehensive Survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 1, pp. 63-84, Jan 2000.

[16] Xiaoqing Liu and Jagath Samarabandu, “Multiscale Edge-Based Text Extraction from Complex Images,” *IEEE International Conference on Multimedia and Expo, Toronto, Ont.,* pp. 1721-1724, 2006.

[17] Hailong Liu and Xiaoqing Ding, “Handwritten Character Recognition Using Gradient Feature and Quadratic Classifier with Multiple Discrimination Schemes,” *Eighth International Conference on Document Analysis and Recognition*, vol. 1, pp. 19-23, 2005.

[18] Vikas J. Dongre and Vijay H. Mankar, “Devanagari Offline Handwritten and Numeral Character Recognition using Multiple Features and Neural Network Classifier,” *2nd International Conference on Computing for Sustainable Global Developement (INDIACom),* pp. 425-431, 2015.

[19] Sushama Shelke and Shaila Apte, “Performance Optimization and Comparative Analysis of Neural Networks for Handwritten Devanagari Character Recognition,” *International Conference on Signal and Information Processing*, pp. 1-5, 2016.

[20] Partha S. Mukherjee, Ujjwal Bhattacharya, Swapan K. Parui and Bappaditya Chakraborty, "A Hybrid Model for End to End Online Handwriting Recognition," *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pp. 658-663, 2017.

[21] Shivajee Pandey, Divya Srivastava and Suneeta Agarwal, "An Efficient Approach for Dynamic PCA filter selection in PCANet for Image Classification," *4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, pp. 139-144, 2017.

[22] Durjoy Sen Maitra, Ujjwal Bhattacharya and Swapan K. Parui, "CNN Based Common Approach to Handwritten Character Recognition of Multiple Scripts," *13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1021-1025, 2015.

[23] Sushama Shelke and Shaila Apte, "A Fuzzy based Classification Scheme for Unconstrained Handwritten Devanagari Character Recognition," *International Conference on Communication, Information and Computing Technology (ICCICT)*, pp. 1-6, 2015.

[24] Peera Jarungthai, Sirapat Chiewchanwattana and Khamron Sunat, "Handwritten Character Recognition Using Generalized Radial Basis Function Extreme Learning Machine with Centers Selection," *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pp. 1-5, 2014.

[25] Adwait Dixit, Ashwini Navghane and Yogesh Dandawate, "Handwritten Devanagari Character Recognition using Wavelet Based Feature Extraction and Classification Scheme," *Annual IEEE India Conference (INDICON)*, pp. 1-4, 2014.

[26] C. Szegedy et al., Going Deeper with Convolutions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, 2015.

[27] P. Li, L. Peng and J. Wen, "Rejecting Character Recognition Errors Using CNN Based Confidence Estimation," *Chinese Journal of Electronics*, vol. 25, no. 3, pp. 520-526, 2016.

[28] Hadar I. Avi-Itzhak, T. A. Diep and H. Garland, "High Accuracy Optical Character Recognition Using Neural Networks with Centroid Dithering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 218-224, 1995.

[29] W. Hu et al., “Sequence Discriminative Training for Offline Handwriting Recognition by an Interpolated CTC and Lattice-Free MMI Objective Function,” *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pp. 61-66, 2017.

[30] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, “Deep Convolutional Neural Network for Inverse Problems in Imaging,” *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509-4522, 2017.

[31] H. Zhang et al., “A Character Level Sequence-to-Sequence Method for Subtitle Learning,” *IEEE 14th International Conference on Industrial Informatics (INDIN)*, pp. 780-783, 2016.

[32] B. Athiwaratkun and J. W. Stokes, “Malware Classification with LSTM and GRU Language Models and a Character-Level CNN,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2482-2486, 2017.

[33] Xiao-Xue Wang et al., “Neural Machine Translation Research Based on the Semantic Vector of the Tri-Lingual Parallel Corpus, *International Conference on Machine Learning and Cybernetics (ICMLC)*, pp. 69-74, 2016.

[34] H. Zhang et al., “Understanding Subtitles by Character-Level Sequence-to-Sequence Learning,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 616-624, 2017.

[35] J. Su et al., “A Hierarchy-to-Sequence Attentional Neural Machine Translation Model,” *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 26, no. 3, pp. 623-632, 2018.

[36] B. Zhang et al., “A Context-Aware Recurrent Encoder for Neural Machine Translation,” *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 25, no. 12, pp. 2424-2432, 2017.

[37] K. Czuszyński, J. Rumiński and A. Kwaśniewska, “Gesture Recognition With the Linear Optical Sensor and Recurrent Neural Networks”, *IEEE Sensors Journal*, vol. 18, no. 13, pp. 5429-5438, 2018.

[38] J. Zhang and C. Zong, “Deep Neural Networks in Machine Translation: An Overview,” *IEEE Intelligent Systems*, vol. 30, no. 5, pp. 16-25, 2015.

[39] K. Macherey, O. Bender and H. Ney, “Applications of Statistical Machine Translation Approaches to Spoken Language Understanding,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 4, pp. 803-818, 2009.

[40] R. Zhao et al., “Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1539-1548, 2018.