Table of Contents

Chapter 1: Introduction5

1.1. Evolution of Machine Learning 5

1.2. Project Definition 7

1.3. Purpose of the Project7

1.4. Project Features 8

Chapter 2: Concepts9

2.1. Neural Network (NN) 9

2.2. Deep Learning 10

2.3. Convolutional Neural Network (CNN) 11

2.4. Rectified Linear Unit (ReLU)13

2.5. Softmax Function 13

Chapter 3: Literature Survey 15

3.1. Papers Referred 15

3.1.1. Title: EMNIST: an extension of MNIST to handwritten letters 15

3.1.2. Title: Region-based Convolutional Networks for Accurate Object Detection and segmentation 16

3.2. Existing System 16

3.3. Proposed System 17

3.4. Software Description 18

3.4.1. Python 18

3.4.2. TensorFlow 18

3.4.3. Keras 18

3.4.4. OpenCV 19

3.4.5. EMNIST byMerge Dataset21

Chapter 4: Requirement Analysis22

4.1. Functional Requirements 22

4.2. Non-Functional Requirements 22

4.3. Hardware Requirements 24

4.4. Software Requirements 24

Chapter 5: Design 25

5.1. Design Goals 25

5.2. System Architecture 26

5.3. Dataflow Diagrams 28

5.3.1. Image Processing and Character Segmentation28

5.3.2. Character’s Position Detection 29

5.3.3. Building CNN Model 29

5.3.4. Character Prediction 30

5.4. Sequence Diagram 31

5.5. Use Case Diagram 32

Chapter 6: Implementation33

6.1. Algorithms and Methodologies 33

6.1.1. Convolutional Neural Network (CNN) 33

6.1.2. Regional Convolutional Neural Network (R-CNN) 38

Chapter 7: Testing43

7.1. Unit Testing 43

7.2. Integration Testing 43

7.3. System Testing 43

7.4. Validation Testing 44

7.5. Black Box Testing 44

7.6. White Box Testing 44

7.7. Test Results 45

Chapter 8: Snapshots47

Chapter 9: Conclusion and Future Enhancements51

9.1. Conclusion 51

9.2. Future Enhancements 52

9.2.1. Dataset 52

9.2.2. Back Propagation 52

9.2.3. Natural Language Processing (NLP) 52

Chapter 10: References53

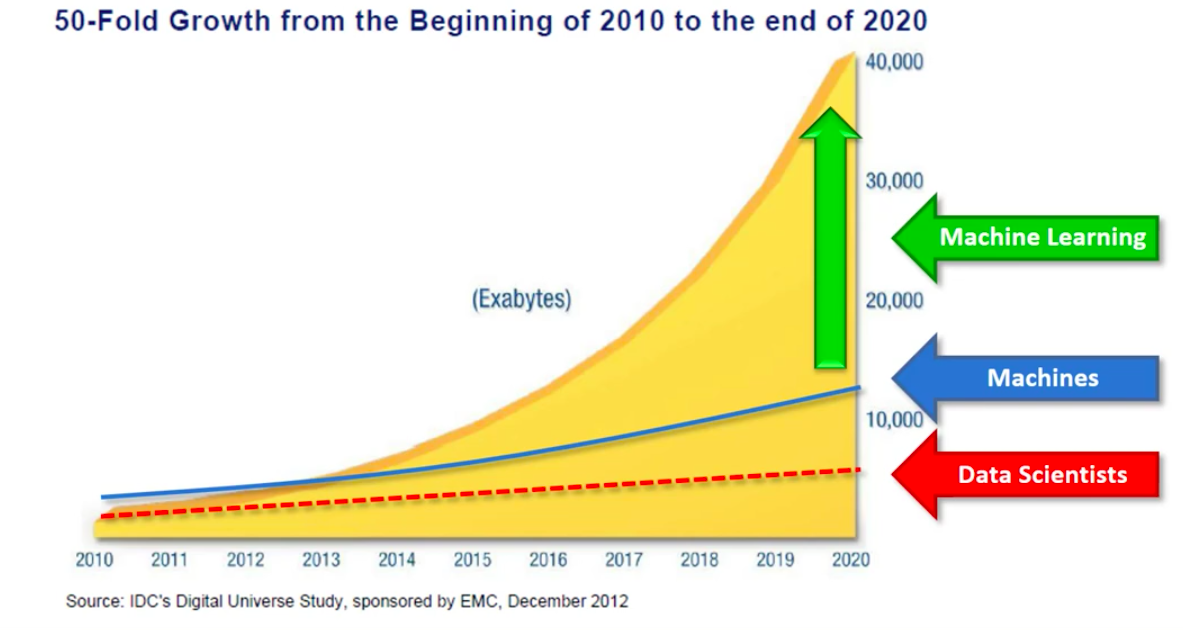
**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Figure Name** | **Page No.** |
| 1.1. | Data Growth and Importance of ML | 5 |
| 1.2. | Machine Learning | 6 |
| 2.1. | Neural Network (NN) | 10 |
| 2.2. | CNN | 11 |
| 2.3. | Comparison between NN and CNN | 12 |
| 2.4. | ReLU | 13 |
| 2.5. | Softmax function | 14 |
| 3.1. | EMNIST Datasets | 15 |
| 3.2. | R-CNN | 16 |
| 3.3. | EMNIST byMerge Dataset | 21 |
| 5.1. | Overall System Architecture | 26 |
| 5.2. | CNN Model | 27 |
| 5.3. | Image preprocessing and character segmentation | 28 |
| 5.4. | Character’s position detection | 29 |
| 5.5. | Building CNN Model | 30 |
| 5.6. | Character Prediction | 31 |
| 5.7. | Sequence Diagram | 31 |
| 5.8. | Use Case Diagram | 32 |
| 6.1. | R-CNN | 38 |
| 6.2. | Original Image and Adaptive Gaussian Thresholding | 40 |
| 7.1. | Correct Prediction of ‘B’ | 45 |
| 7.2. | Correct Prediction of ‘2’ | 45 |
| 7.3. | Incorrect Prediction of ‘7’ | 46 |
| 8.1. | Input Image | 47 |
| 8.2. | Color Inversion and Adaptive Thresholding | 47 |
| 8.3. | Masking Contours | 48 |
| 8.4. | Bounding Boxes | 48 |
| 8.5. | Segmented Characters | 49 |
| 8.6. | Predicting Line Numbers | 49 |
| 8.7. | Predicting Word Numbers | 50 |
| 8.8. | Output Text Document | 50 |

**CHAPTER 1**

**INTRODUCTION**

**1.1. Evolution of Machine Learning**

**Data is present everywhere and is growing exponentially. People, computers and other devices generate data at almost every instance of a second. Since the dawn of time, up until 2005, humans have created 130 Exabytes of data. It includes everything such as the books written, songs sung, words spoken and everything that humans have created. By 2010, the data rose up to 1,200 Exabytes. In next 5 years, it increased exponentially to 7,900 Exabytes and it is estimated to rise up to 40,000 Exabytes in 2020.

*Fig. 1.1. Data Growth and Importance of ML*

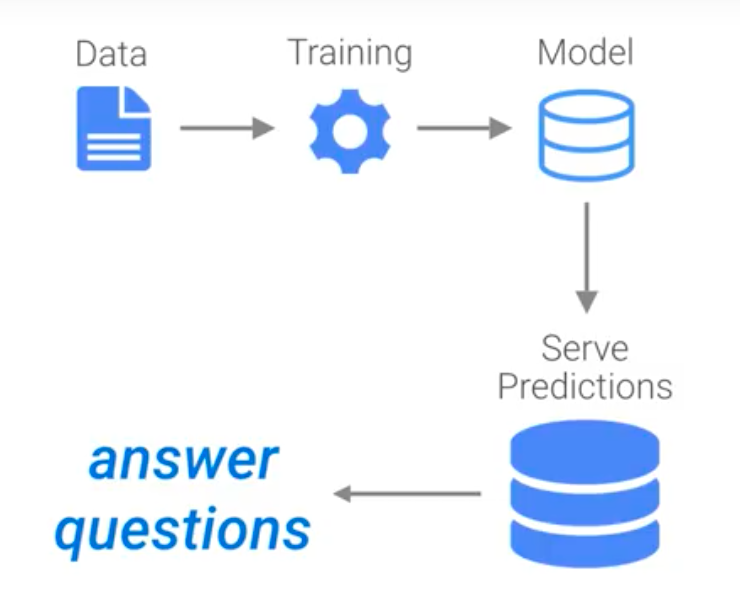
We, as data scientists can process about 7,000 Exabytes of data and make the machines process up to only 13,000 Exabytes of data. The remaining data is left void, and this is where machine learning comes into picture. Only the machine learning algorithms can make use of the vast data.

People often misinterpret Artificial Intelligence (AI) for Machine Learning (ML), while Artificial Intelligence is the entire field that makes the machine smart and Machine Learning is just a division of Artificial Intelligence.

Sixty years ago, machine learning was considered as a fiction, and currently it has become an integral part of our life. It is used almost everywhere from image recognition, fraud detection and recommendation systems, as well as text and speech conversion systems too.

In 1642, the thought of building a neural network imitating a human brain came into picture when a French man invented the first mechanical calculator. This theory came into reality in 1943 where the neural network was built using electrical circuits. In 1950, Alan Turing invented the Turing machine in order to test the intelligence of a computer if can process like a human brain. In 2006, Deep learning came into existence, which is a branch of machine learning concerned with algorithms inspired by the functioning of a human brain.

Machine learning can be defined as a machine, learning from experience, which is most often called as data. We make the machines think like humans and predict the result rather than programming. Machine learning can also be defined as ‘using data (training) to answer questions (prediction)’.

**

*Fig. 1.2. Machine Learning*

Training refers to using data in order to create and fine tune a predictive model, which is used to serve predictions on previously unseen data and answer the questions. Humans can make predictions using 1, 2 or 3 dimensions of data, whereas machines can make predictions based on 100s and 1000s of dimensions.

**1.2. Project Definition**

The objective of this project is to convert the handwritten document into digital document consisting of ASCII characters, which can be easily read or manipulated on any electronic device. Here, we use Regional Convolutional Neural Network (RCNN) for character recognition and prediction. RCNNs are usually used for object detection. Character segmentation is performed on the input handwritten document image, the output of which is passed to the Convolutional Neural Network (CNN). It predicts the probability of each character class with one character class having the highest probability. The highest probability character class is mapped to the ASCII character, which is written onto a text document. The output is post processed using spell check for improving the accuracy of the words.

**1.3. Purpose of the Project**

Even though we live in a digital world, where we have and abundance of technological writing tools, many people still prefer taking notes traditionally with pen and paper. Therefore handwritten text plays a major role. Majority of the education institutions still follow the handwritten examination pattern. Several book authors also prefer writing their script on papers. Editing these scripts is mandatory for corrections, which is hectic, and further they have to be typed for publication. It is difficult to store and access or search these physical documents. Therefore it is necessary to convert these handwritten documents into the digital format for efficient processing of text. Digital documents are easier to read, manipulate, store and share.

Our project helps in overcoming the drawbacks of handwritten documents by converting them into the digital format. We use machine learning and neural networks to predict the characters of various handwritings. It reduces the overhead of typing the whole handwritten document. We use object detection for detecting and segmenting the handwritten text into individual characters and further predicting the corresponding ASCII character. The output is also post processed for spelling corrections. The intended system can be used to process cheques and other handwritten forms, which has to be fed and stored into the computer.

**1.4. Project Features**

The process of converting the handwritten document into the digital format requires several processes. Firstly, the handwritten document is scanned and converted to a digital image, which is preprocessed to remove the noise and artifacts in the image. OpenCV is used to detect and segment only the handwritten text into individual handwritten characters. Further, the position of each character relative to the original document is stored. Once image is processed, each character image is passed one-by-one to the built and trained Convolutional Neural Network (CNN) model. The CNN model results a set of probabilities for each character class. The highest probability character class is chosen as the corresponding output for the given input character. The output character is stored in a text document at its respective positions. The CNN model is built using the EMNIST by-merge training and testing datasets. Each word in the output text document is verified using a spellcheck tool that corrects the incorrect words.

**CHAPTER 2**

**CONCEPTS**

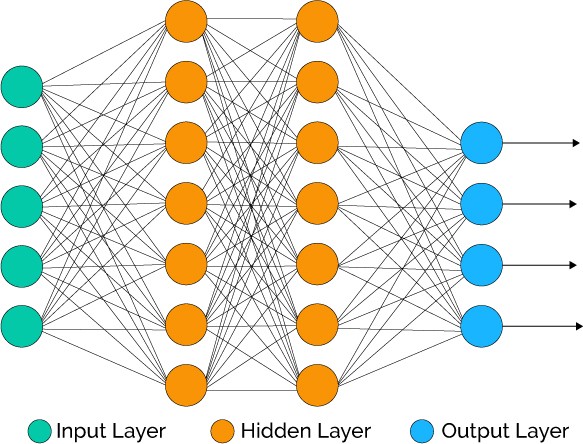
**2.1. Neural Network (NN)**

In the simulation of a biological neuron, information that flows into the neuron is processed by the neuron and the results flow out. This gives the neuron, the abilities to react based on previously learned patterns. Technology duplicates this by creating a structure that processes information similar to a biological neuron, except that this process uses mathematics instead. Just like a biological neuron, information flowing into the neuron is processed by the artificial neuron and the results flow out.

A neural network is inspired by a human brain. They use algorithms that are based on the way the brain works. A neural network can be used to build a predictive model by learning the patterns in the historical data. They are made up of small-interconnected processing elements. These elements are called as nodes or neurons. Each node is dedicated to process a small part of the task.

The most common type of neural network is a multilayered perceptron. In a multilayered perceptron, the nodes are organized into layers. The first layer is called the input layer. The outermost layer is termed as the output layer. Between these two layers, there are one or more layers called the hidden layers.

The input layer receives the values of the independent variables as input. The nodes of the hidden layer takes their input from the input layer, processes it and passes it on to the nodes of the next hidden layer or the output layer. A neuron holds a value between 0 and 1.

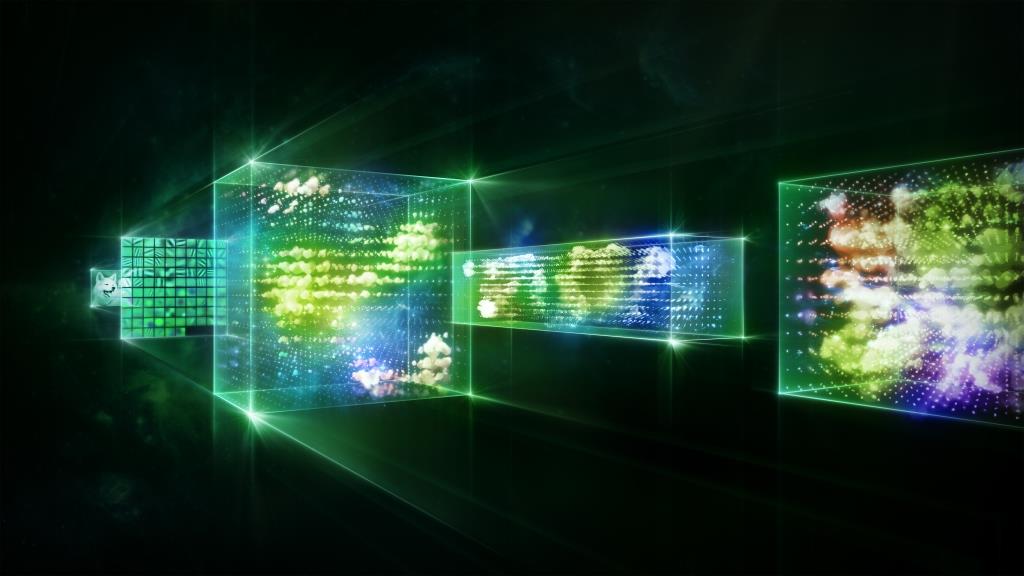
In a neural network, a node from one layer is connected to each node of its successive layer. A node therefore receives information from multiple nodes of the previous layer. These are multiplied by unique weights and added together with a small value called bias. The total is processed by a function called the activation function and leaves the node as output. Activations in one layer determine the activation on the other layer. This process proceeds till the information reaches the output layer and leads it as a prediction for the dependent variable. The network then compares the prediction with the actual value of the dependent variable. If these do not match, it adjusts all the weights in the neural network and repeats the process. These iterations get repeated till the neural network is able to produce accurate predictions for most of the observations. Once this is achieved, we are left with a neural network model that can be applied to a new set of data to provide predictions.

*Fig. 2.1. Neural Network*

**2.2. Deep Learning**

Deep learning is derived from machine learning that learns unsupervised data and the data that is not labeled, which is also known as deep neural network. It is inspired from the structure of human brain and the way it functions. Deep learning uses special types of neural networks to learn difficult patterns. Currently it is known for identifying objects in images. The success of this pattern recognition led to their usage in field of medical diagnosis and other business related problems. The advantage of using deep learning is they perform automatic feature extraction from raw data.

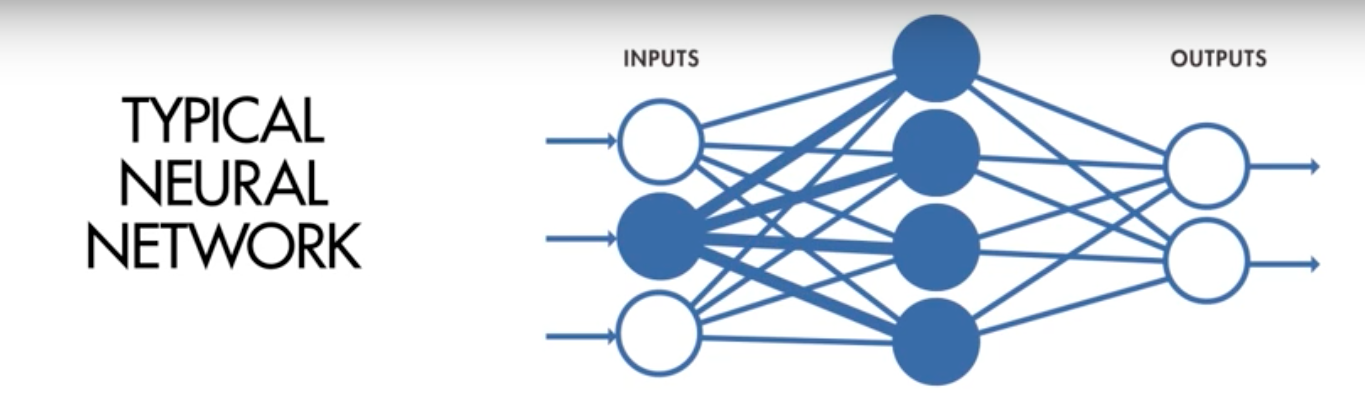
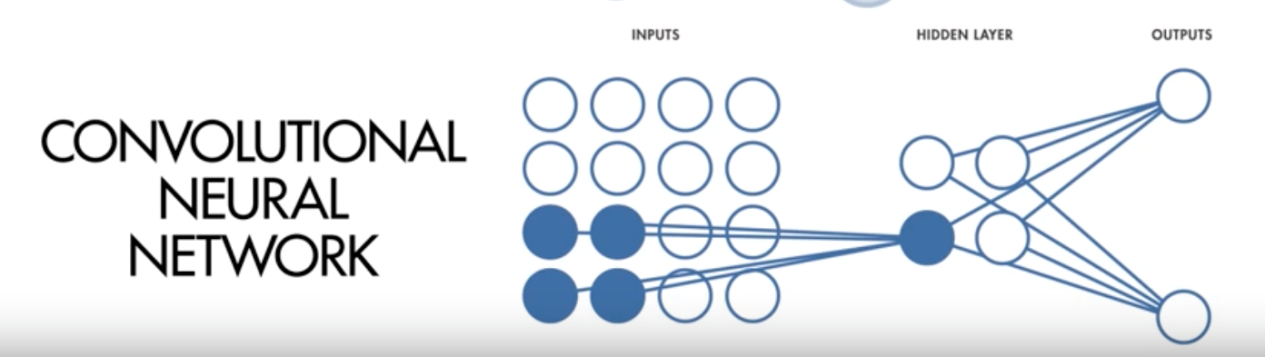
**2.3. Convolutional Neural Network (CNN)**

**Convolutional Neural Network is an example of deep neural network. A CNN is well suited for working with image data. The term “deep” usually refers to the number of hidden layers in the neural network. While traditional neural networks contain only 2 or 3 hidden layers, the recent networks contain 10s to 100s of hidden layers.

*Fig. 2.2. CNN*

A CNN can be trained to perform image tasks such as scene classification, object detection and segmentation, and image processing. A CNN is different from NN because of 3 major concepts:

1. Local Receptive fields
2. Weights and Biases
3. Activation and Pooling

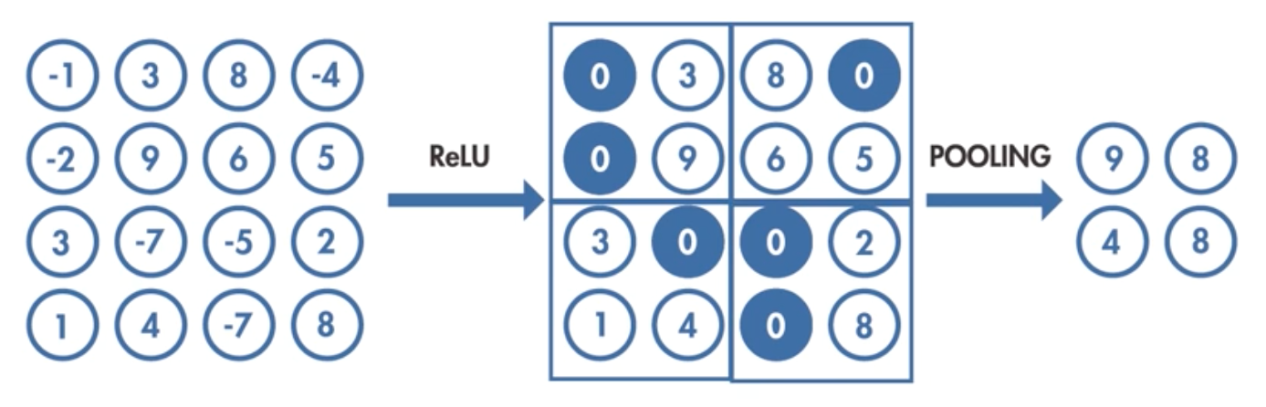
*Fig. 2.3. Comparison between NN and CNN*

In a typical neural network, the input layer is connected to each neuron in the hidden layer. However in CNN, only a small region of input layer neurons connect to neurons in the hidden layer. These regions are referred to as local receptive fields. The local receptive field is translated across an image to create a feature map form the input layer to the hidden layer neurons. We can use convolution o implement this process efficiently. That’s why it is called a convolutional neural network.

Like a typical neural network, the CNN has weights and biases. The model learns these values during the training process and it continuously updates them with each new training example. However, in the case of CNNs, the weights and bias are the same for all hidden neurons in a given layer. This means that all hidden neurons are detecting the same feature such as an edge or a blob in different regions of the image. This makes the network tolerant to translation of objects in an image. For example, a network trained to recognize cats will be able to do so wherever the cat is present in the image.

The third and the final concept is activation and pooling. The activation step applies the transformation to the output of each neuron by using activation functions. Rectified Linear Unit (ReLU) is an example of a commonly used activation function. The output of the activation step can be transformed by applying a pooling step. Pooling reduces the dimensionality of the feature map by condensing the output of small regions of neurons into a single output. This helps simplify the following layers and reduces the number of parameters that the model needs to learn.

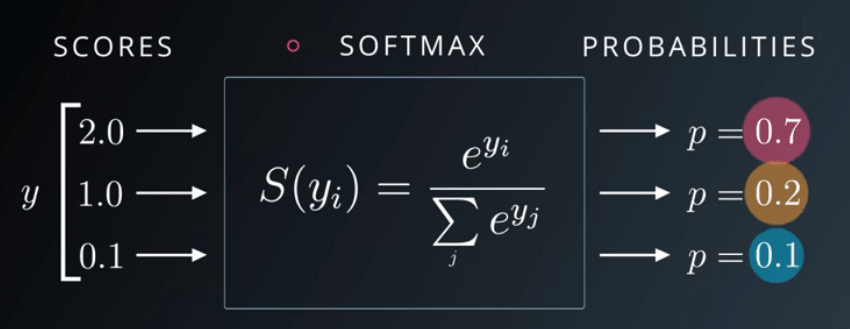
**2.4. Rectified Linear Unit (ReLU)**

**Rectified Linear Unit (ReLU) is an activation function that takes the output of a neuron and maps it to the highest positive value, or if the output is negative, the function maps it to zero. It is used to remove the unwanted portions of the image, by discarding the negative values. It is usually used before a pooling layer and improves the neural network by speeding up the training.

*Fig. 2.4. ReLU*

**2.5. Softmax Function**

A softmax function is a generalization of the logistic function. It is also referred to as the normalized exponential function. This function is usually used in the output layer as the activation function and results in a set of probabilities. However, softmax function is not a traditional activation function because an activation function produces a single output for a single input, whereas the softmax function produces multiple outputs for a single input array. The sum of the outputs is always 1.

*Fig. 2.5. Softmax function*

For example, the following results will be retrieved when softmax is applied for the inputs given in the above figure.

σ(x1) = ex1 / (ex1+ex2+ex3 ) = e2 / (e2 + e1+ e0.1) = 0.7

σ(x2) = ex2 / (ex1+ex2+ex3 ) = e1 / (e2 + e1+ e0.1) = 0.2

σ(x3) = ex3 / (ex1+ex2+ex3 ) = e0.1 / (e2 + e1+ e0.1) = 0.1

The outputs are normalized between [0, 1] and the sum of the outputs is equal to 1.

**CHAPTER 3**

**LITERATURE SURVEY**

**3.1. Papers Referred**

**3.1.1. Title:** EMNIST: an extension of MNIST to handwritten letters

**Author:** Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andr´e van Schaik

### This paper provided with the details of the Extended MNIST (EMNIST) Dataset, which we will be used to train the Convolutional Neural Network (CNN). The Dataset unifies the pre-existing MNIST dataset with the NIST Special Dataset 19, which was converted to 28x28 pixel image and the structure that directly matches the MNIST dataset. The EMNIST Dataset consist of 814,255-character image data written by about 3600 writers.

*Fig. 3.1. EMNIST Datasets*

### 3.1.2. Title: Region-based Convolutional Networks for Accurate Object Detection and segmentation

### Author: Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik.

### C:\Users\mohan\Pictures\Screenshots\Screenshot (4).pngThis paper inspired us in implementing the Recurrent Convolution Neural Network (R-CNN) consisting of two main ideas. Firstly, using selective search, the important features of the image are extracted. It defines a number of bounding-box object regions called as region of interest or RoI. These are input to Convolutional Neural Network (CNN) model from each region independently for performing classification.

### *Fig. 3.2. R-CNN*

**3.2. Existing System**

### The Existing system mainly composes of Optical character recognition (OCR) Tool. Which is used to detect character, which are usually printed. These tools use machine learning algorithms, which are trained on characters of different font rather than handwritten characters.

### LeNet is one of the implementation which performs handwritten character classification using Convolutional Neural Network (CNN) some of the implementation on MNIST dataset provided 99% accuracy.

### Advantages:

### Better performance in terms of speed.

### Disadvantages:

### Pre-processing and post-processing is not implemented.

### Document level conversion is not implemented because it could only process single handwriting character as its input.

**3.3. Proposed System**

### The Goal of Project is to improve the efficiency of conversion of a handwritten character and to implement Document or Paper level conversion to digital format using R-CNN (Regional Convolutional Neural Network). The Proposed system includes preprocessing the input handwritten Document image, which is passed as input for the character segmentation. The segmented characters are passed as input to the Convolutional Neural Network (CNN), which is designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability, distortions and simple geometric transformations.

### Advantages:

### Pre-processing and Post-Processing is implemented for better accuracy.

### Document level conversion is performed.

### Spell check is used for post processing the output for better results.

### Disadvantages:

### Requires a large amount of training time.

### Performance is reduced due to post processing.

**3.4. Software Description**

**3.4.1. Python**

### Python was created by Guido van Rossum in 1991. It is a scripting, interpreted high-level programming language for general-purpose programming. It is an object-oriented programming language. Python focuses mainly on code readability and maintainability. It has a support for large and extensive standard libraries. It is used for web and app development. The python packages are reusable in nature, i.e. when imported once, it can be used for other projects too. All python tools are available for free.

### Python can be used for processing images, detecting objects. It is used to implement machine learning. Several open source libraries are built using python which are user-friendly in developing machine learning models such as the neural networks.

### It is also an automaton language, used for interactions with GUIs. All the python modules and libraries are considered as objects. Python also has a garbage collector to handle with unused data. Currently there are two versions of python - python 2 and python 3. There are many third-party libraries available in python.

**3.4.2. TensorFlow**

### Tensorflow is an open source machine learning framework developed by Google Brain team for high performance numerical computation. It is a flexible architecture that allows easy deployment of computation across variety of platforms such as CPUs, GPUs and TPUs. TensorFlow’s main application lies in the Neural Network field. It is a low-level deep learning API that involves complex mathematical operations.

**3.4.3. Keras**

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. It is designed to enable fast experimentation with deep neural networks and focuses on being user-friendly, modular, and extensible. It offers a high-level, more intuitive set of abstractions that makes it easy to develop deep learning models regardless of the computational backend used. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools for working with image and text data.

### Features of Keras:

### User friendliness: Keras is an API designed for human beings, not machines by putting forth the user experience. Keras follows best practices for reducing cognitive load as it offers consistent & simple APIs, and it minimizes the number of user actions required for common use cases. It provides clear and actionable feedback upon user error.

### Modularity: A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that be combined to create new models.

### Easy extensibility: New modules are simple to add (as new classes and functions), and existing modules provide ample examples. The ability to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

### Work with Python: Separate models configuration files in a declarative format are not required. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

**3.4.4. OpenCV**

OpenCV which is Open Source Computer Vision Library and a machine learning library is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

### The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

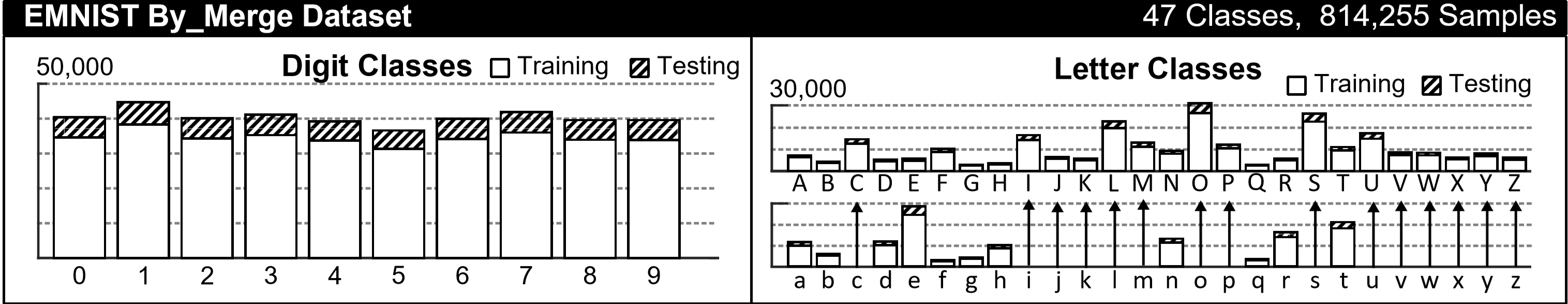
OpenCV's application areas include:

* 2D and 3D feature toolkits
* Ego motion estimation
* Facial recognition system
* Gesture recognition
* Human–computer interaction (HCI)
* Mobile robotics
* Motion understanding
* Object identification
* Segmentation and recognition
* Stereopsis stereo vision: depth perception from 2 cameras
* Structure from motion (SFM)
* Motion tracking
* Augmented reality

OpenCV includes a statistical machine learning library that contains:

* Boosting
* Decision tree learning
* Gradient boosting trees
* Expectation-maximization algorithm
* k-nearest neighbor algorithm
* Naive Bayes classifier
* Artificial neural networks
* Support vector machine (SVM)
* Deep neural networks (DNN)

**3.4.5. EMNIST byMerge Dataset**

*Fig. 3.3. EMNIST byMerge Dataset*

EMNIST extends MNIST by including images of handwritten letters (upper and lower case) as well as handwritten digits. Both EMNIST and MNIST are extracted from the same underlying dataset, referred to as NIST Special Database 19. Both use the same conversion process resulting in centered images of dimension 28×28. Although there are 62 potential classes for EMNIST (10 digits, 26 lower case letters, and 26 upper case letters) we shall use a reduced label set of 47 different labels. This is because of confusions, which arise when trying to discriminate upper-case and lower-case versions of the same letter, following the data conversion process. In the 47-label set, upper- and lower-case labels are merged for the following letters: C, I, J, K, L, M, O, P, S, U, V, W, X, Y and Z.

**CHAPTER 4**

**REQUIREMENT ANALYSIS**

**4.1. Functional Requirements**

In general, functional requirements are used for understanding the functionality of system and also its subsystems. They tell what a system should perform. It also depends on the constraints where the software is used. It might also depend on the users. They are the statements that describe how system should operate. We must mention what a system should do in detail. There are few key features that belong to functional requirements such as scope, performance, data conversion, audit tracking and few other fields.

* The handwritten document is scanned and converted to image, which is taken as input.
* The input is passed through the convolution model which has various layers such as conv2d, max pooling, dense, flatten, and dropouts.
* The CNN prediction model is trained using the datasets and tested against the validation datasets.
* The input document after the character segmentation is passed through each of these layers and output is predicted at the other end of the model.

**4.2. Non-Functional Requirements**

Non - functional requirements are used to depict all the conditions that can be used to determine the operations of a system. They cover all aspects that are left out by functional requirements such as scalability, availability, reliability, portability, etc. A few of them are listed below:

* **Data Integrity:**

The data integrity is referred to as data consistency. When data is transferred from one file to another and reaches the other end, it should make sure that the data is unaltered or unmodified and ensure that the integrity of the data is preserved.

* **Inter operability:**

It occurs when two products exchange and make use of their data. When two products share their information with each other and start working with each other or with other products, it is called as inter operability.

* **Recoverability:**

Recoverability means that the data is recovered after a fault. When a failure occurs and causes a system to fail or halt, recoverability is used to recover the data till the point of failure, restore and continue functioning.

* **Serviceability:**

It is a set of features that improve and speed up the maintenance of the system when required. It includes all actions required to repair a system when it is failed.

* **Security:**

Security ensures if the system has control over its access and also stores all its data in secured locations and in secured formats. It must also ensure that the communication channel is secure.

* **Performance:**

It is the quantitative metric that the system should meet. The system must produce accurate results within a short period of time. It is important to choose the right technological tools. It is used to improve the system features.

* **Maintainability:**

Once the system is deployed, it must be constantly be upgraded to newer versions with additional and modified functionalities to produce efficient results.

* **Usability:**

The system must satisfy the end users and it should be easy to use. The users should be provided with the documentation of the system, which details how to use the system.

**4.3. Hardware Requirements**

* **Processor :** Intel i5 and above (Preferred Intel i7)
* **Speed :** 2.5 GHz and above
* **RAM :** 8 GB ( minimum)
* **Hard Disk :** 40 GB
* **Graphic Card :** Nvidia 650 Ti and above (at least 2GB DDR5 VRAM)
* **Camera**
* **Display Monitor**

**4.4. Software Requirements**

* **Operating System :** Windows, Mac OS or Linux.
* **IDE :** Anaconda IDE.
* **Language :** Python.
* **Drivers :** Nvidia CuDNN
* **Editor :**  Spyder
* **APIs :** Keras, Tensorflow, OpenCV, Pandas, Numpy, glob.
* **Dataset :** EMNIST by Merge Dataset.

**CHAPTER 5**

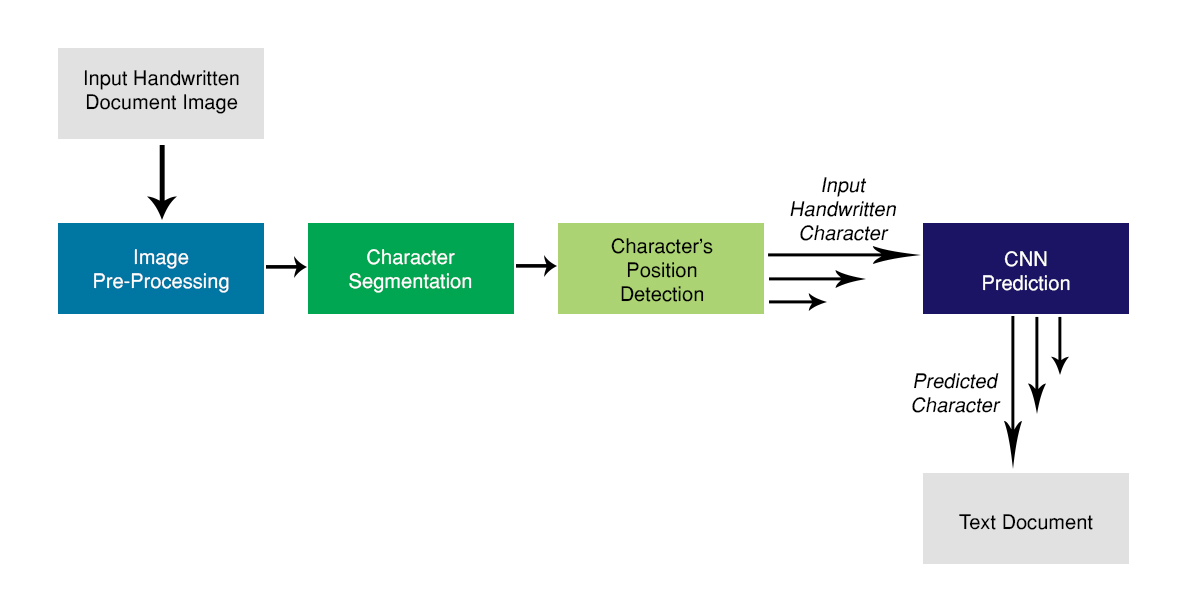
**DESIGN**

Design describes the overall system architecture. It includes system style, system design, hardware and software design, and information style. It establishes relationships between various modules in the system and contains data flow and interaction models.

**5.1. Design Goals**

The system has to meet various design goals such as:

* **Performance:** It is the speed or effectiveness of a software system. It must provide short response time, high throughput and low utilization of computer resources. The system should be able to perform character segmentation of the input image and predict characters using the neural network at a faster rate and low CPU utilization.
* **Efficiency:** It is the measure of productivity and effectiveness of a system. The system should pre-process the input image efficiently by discarding the noise and other artifacts in the image. During the prediction process, the neural network must produce a higher probability only to a single character.
* **Accuracy:** It is the degree of closeness of measurements of a quantity to a quantity’s true value. During the character segmentation, the resulting image should not be distorted. The neural network should be able to predict the correct handwritten character.
* **Applicability:** It represents the implementation of the system in various domains. The implementation of the system in various applications should be user-friendly. It can be used in processing cheques in banks, convert author’s handwritten books into digital format and several other applications.

****5.2. System Architecture**

*Fig. 5.1. Overall System Architecture*

System architecture is a conceptual model that holds the fundamental organization of a system. It represents the structure and behavior of the system. Fig. 5.2. shows the overall system architecture which takes a handwritten document image as its input and outputs a corresponding text document.

The input handwritten document image consists a group of handwritten sentences written on a plain background. The image has to be preprocessed to remove the noise and artifacts present in it. This processed image is then passed to the character segmentation module, which segments the processed image into separate handwritten characters. Each character’s position is then detected and stored along with the characters’ image. These segmented characters along with its position are passed to the CNN (Convolutional Neural Network) prediction model one-by-one. The model predicts the corresponding handwritten character and generates a text document containing the predicted characters in their respective positions.

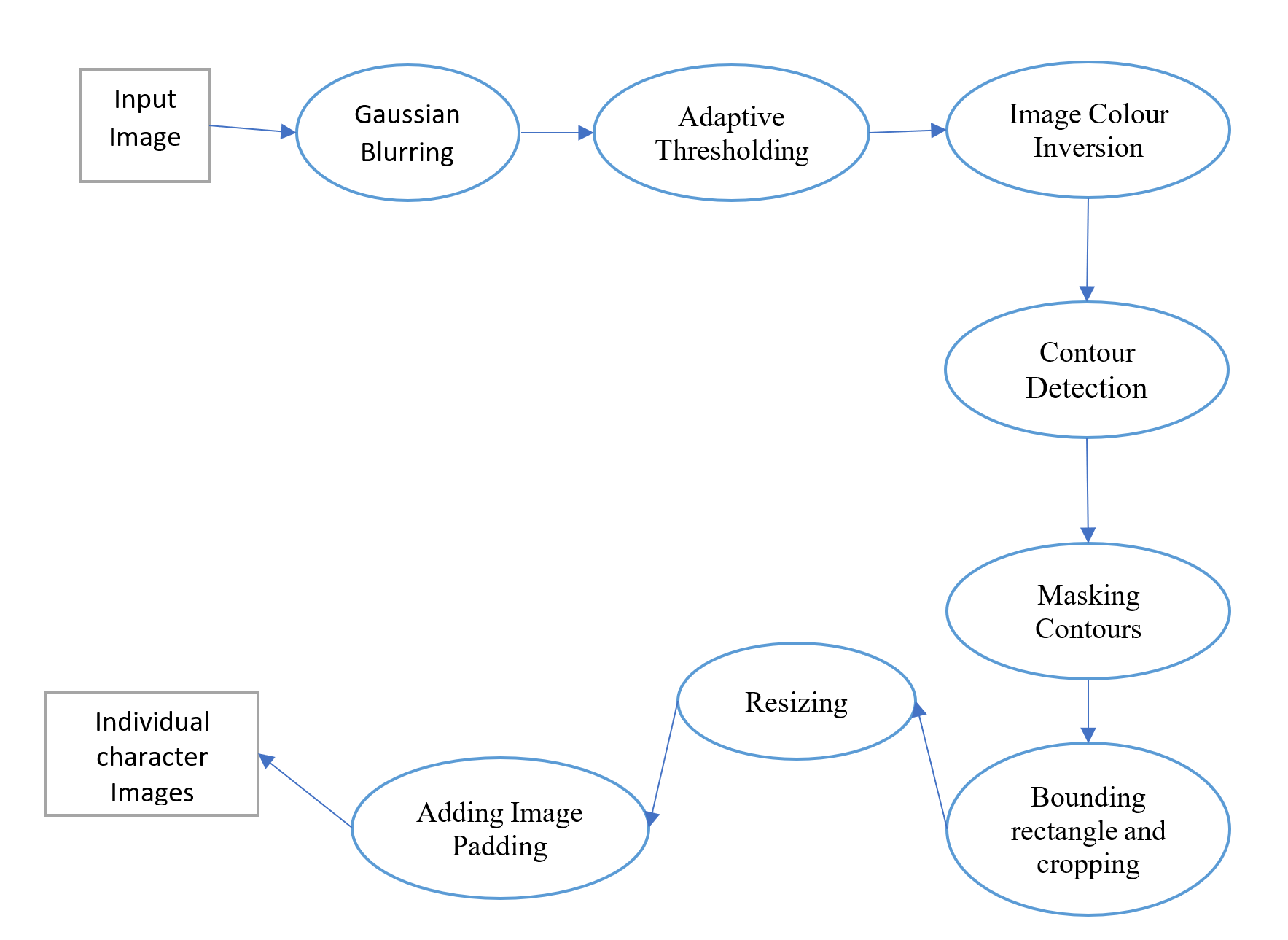
*Fig. 5.2. CNN Model*

The CNN (Convolutional Neural Network) model consists of 8 layers. It is a sequential model, which takes a handwritten character as its input in the MNIST format and outputs a series of probabilities of what the character might be with one character class having the highest probability.

1. The first layer is the input convolutional layer consisting of 32 neurons with a kernel window size of 3 \* 3. It is a 2-dimensional layer because the input is a grayscale image. It uses ReLU (Rectified Linear Unit) as its activation function.
2. The second layer is also a convolutional layer 64 neurons instead. It also uses ReLU as its activation function.
3. The third layer is referred to as the pooling layer, which reduces the dimensionality of the feature map by condensing the output of small regions of neurons into a single output with a spatial dimension of 2 \* 2.
4. The fourth layer performs dropout by a factor of 0.25. It is a form of regularization, which constraints network adaption to avoid it becoming too smart in learning the input data. Thus it helps in avoiding over-fitting.
5. The fifth layer flattens the structure of the feature map to create a single long feature vector, which will be used by the dense layer for classification.
6. The sixth layer is the dense layer (fully connected layer). Every node in this layer is connected to every node in the preceding layer. It contains 128 classes and uses ReLU as its activation function.
7. The seventh layer is similar to the fourth layer, but with a dropout factor of 0.50.
8. The eighth layer (output layer) is a fully connected layer. It contains 47 classes, which is mapped to every node in the preceding layer. It uses softmax function to generate probabilities for each character class.

**5.3. Dataflow Diagrams**

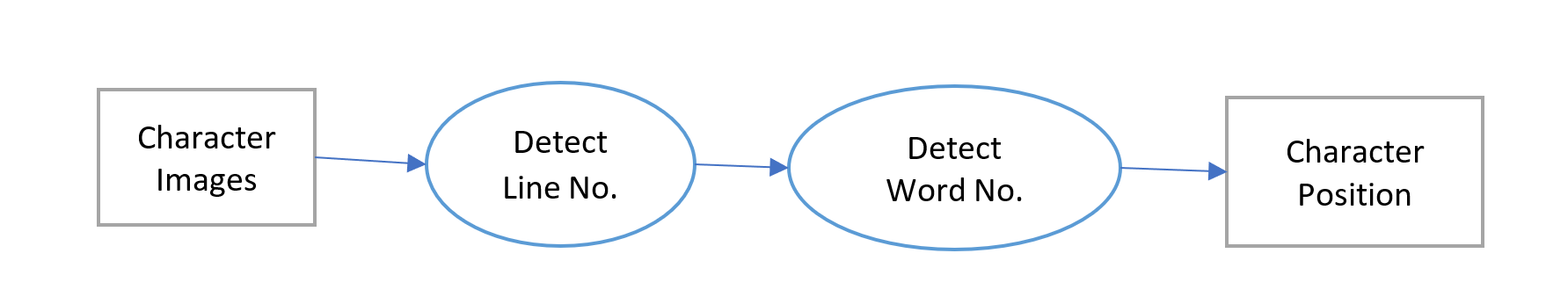
Dataflow diagram (DFD) indicates the flow of information through a process or system. It is built using standardized notations like rectangles, circles, arrows, text labels, and routes between the entities representing their relationships. Dataflow diagrams are used to model a system in an efficient manner.

**

**5.3.1. Image Processing and Character Segmentation**

*Fig. 5.3. Image preprocessing and character segmentation*

The above dataflow diagram represents the preprocessing and character segmentation of the input image. Gaussian blurring is performed on the input image followed by adaptive thresholding. The resulting image is color inverted and passed for detecting contours. These contours are masked and bounded by rectangles. The bounding rectangles are used for cropping each character from the image. These cropped images are further resized and padded with a black border. The result is a collection of 28 \* 28 images, each representing an individual handwritten character.

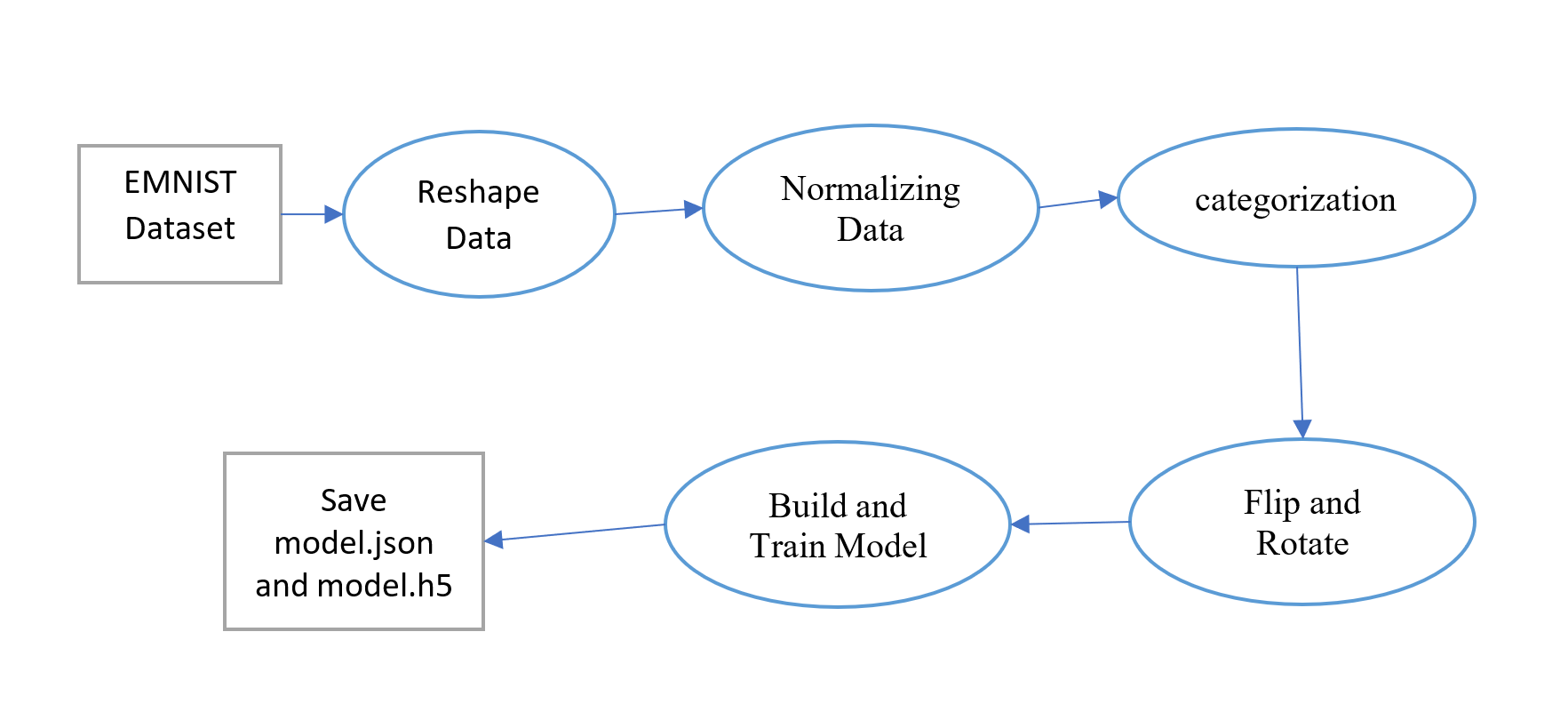
****5.3.2. Character’s Position Detection**

*Fig. 5.4. Character’s position detection*

The above dataflow diagram represents the process of retrieving the characters’ position in terms of line number and word number. The input is a set of images along with its pixels’ position in the handwritten document. The line numbers are retrieved based on the characters’ y-coordinate and the word numbers are retrieved based on the characters’ x-coordinate.

**5.3.3. Building CNN Model**

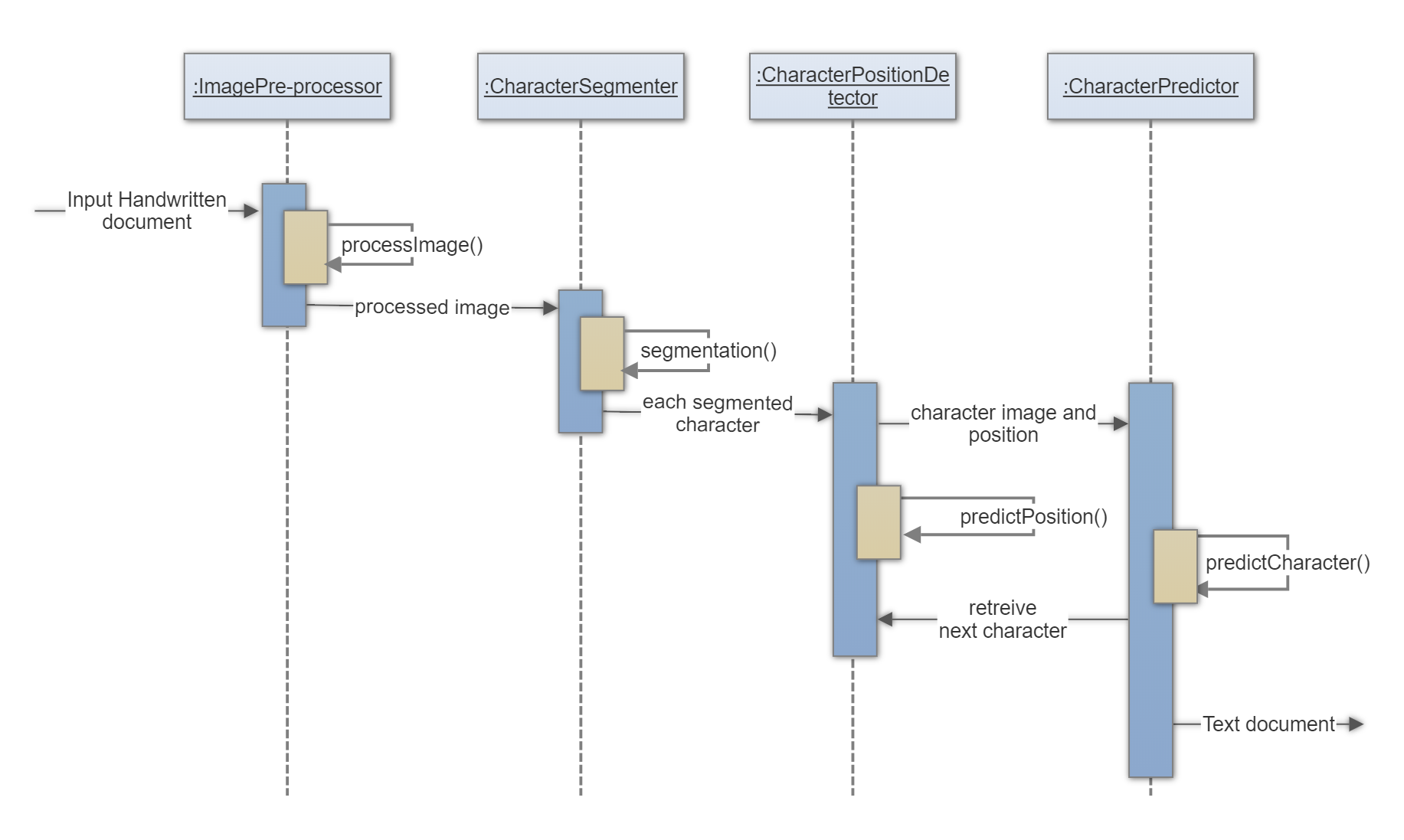
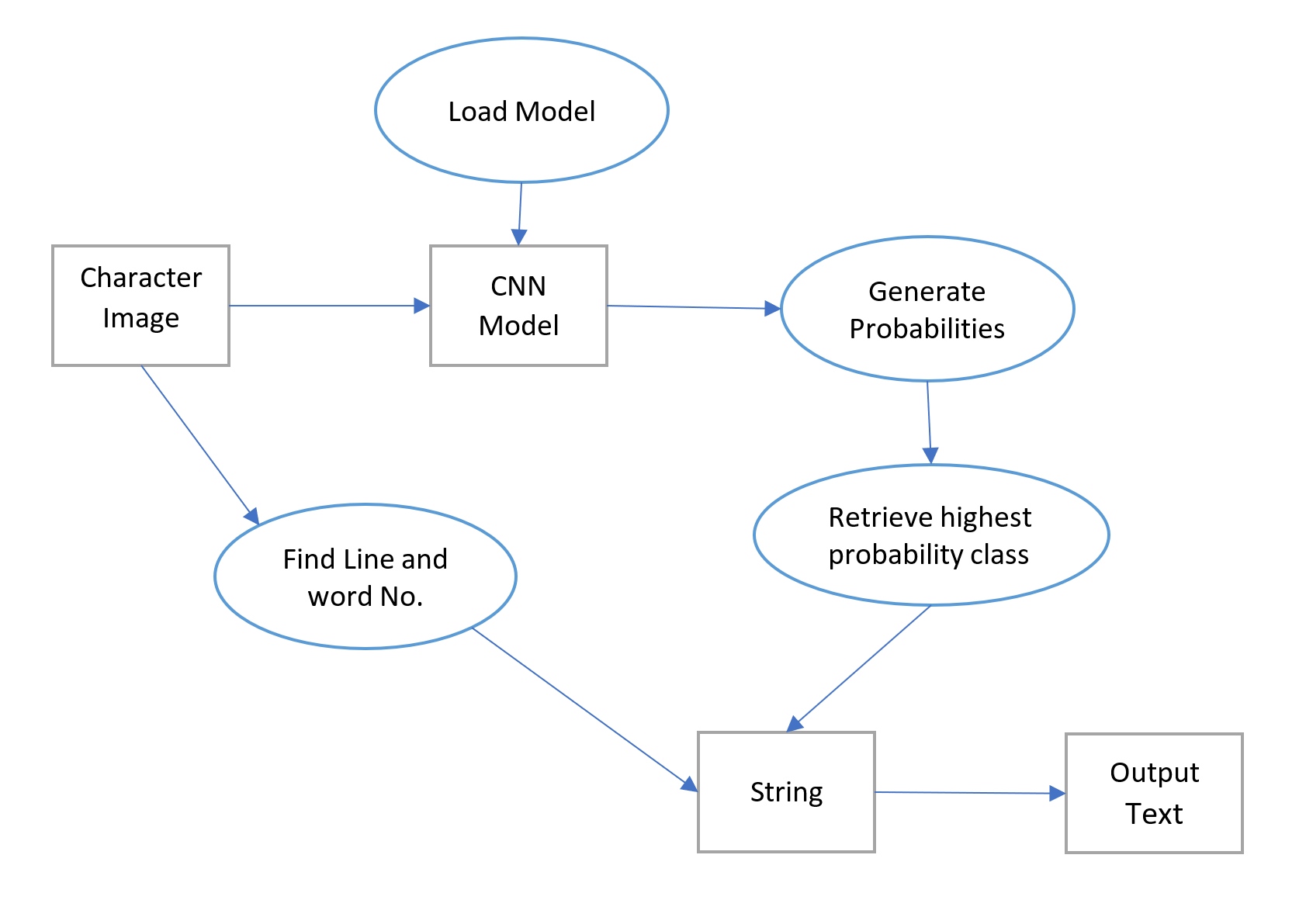
The dataflow diagram below represents how the EMNIST by-merge dataset is used to build the CNN model. After importing the dataset, the data is reshaped into a 2-dimensional 28 \* 28 matrix based on channel-first or channel-last format. The resulting matrix is normalized for better vector operations. The matrices are categorized based on the classes. The data in the dataset, by nature, are flipped and rotated. Hence they are anti-flipped and anti-rotated to get the true matrix used for building the model. The model is built and trained with the training dataset and tested against the testing dataset. Then the structure of the model is saved in the json file and the learnt weights are stored in the h5 file.

**

*Fig. 5.5. Building CNN Model*

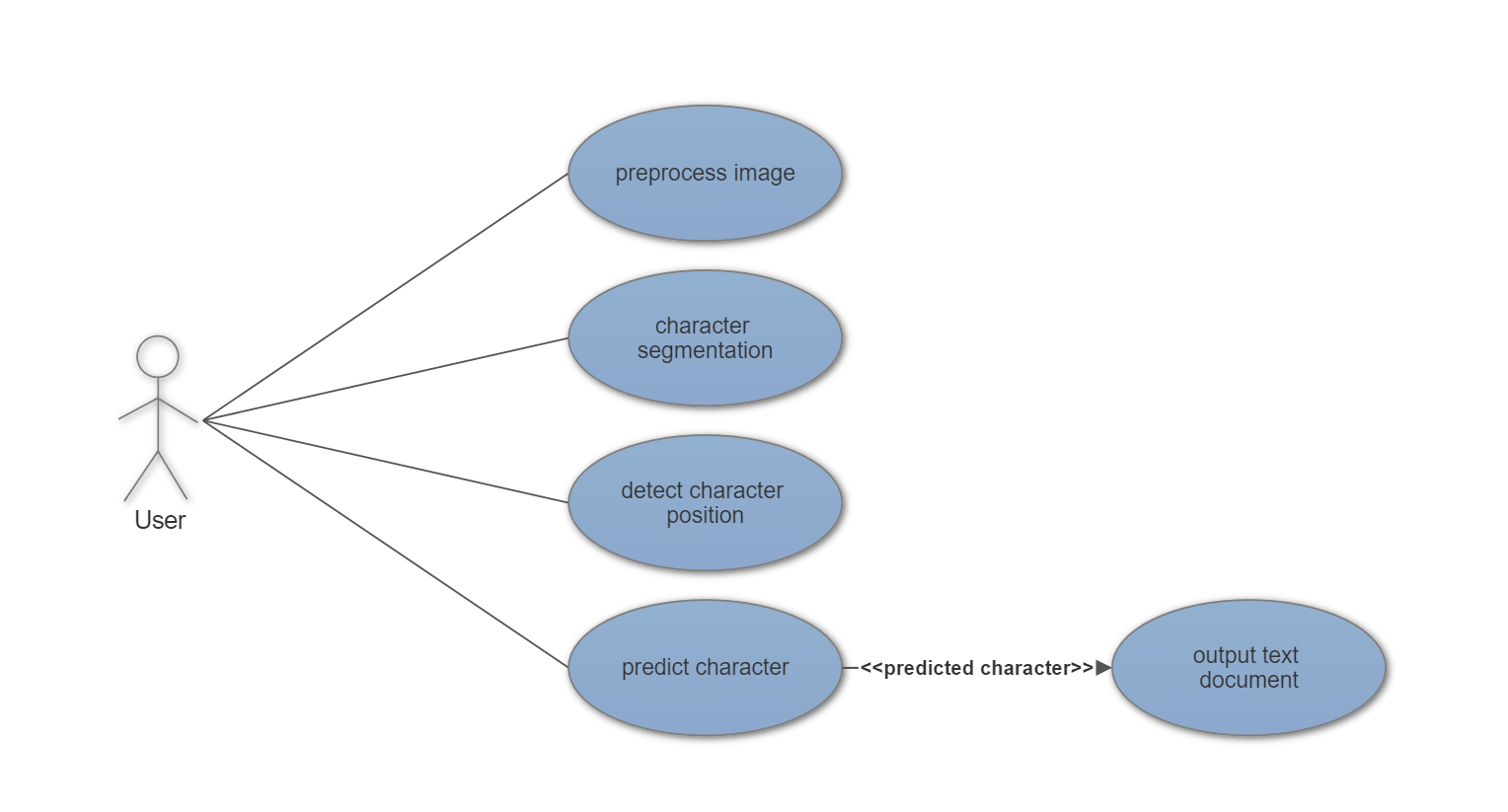
**5.3.4. Character Prediction**

The dataflow diagram below represents how the built CNN model is used to predict the output character for the input handwritten character. The CNN model is loaded from the json and the h5 file. The input character is passed to this model. The model generates a set of probabilities, among which, the character class with the highest probability is chosen as the output character. All these predicted characters are stored in a string with respect to their line and word numbers. This string is then written into a text document.

*Fig. 5.6. Character Prediction*

**5.4. Sequence Diagram**

*Fig. 5.7. Sequence Diagram*

**5.5. Use Case Diagram**

*Fig. 5.8. Use Case Diagram*

**CHAPTER 6**

**IMPLEMENTATION**

The implementation phase defines how the system should be built conforming to the requirements and design properties. It must ensure that the system is operational and used in real-world applications. It must also meet the quality standards. System implementation uses the structures created during the design stage along with the results of the analysis stage to build the subsystems, which meets the requirements obtained during the early life cycle phases. These subsystems are then integrated to form a complete system.

**6.1. Algorithms and Methodologies**

**6.1.1. Convolutional Neural Network (CNN)**

The Convolutional Neural Network is a deep artificial neural network mainly used to work with images. It is also referred to as a multilayer perceptron, which requires minimal processing. The first layer is called the input layer, while the outermost layer is called the output layer. The layers between these two layers are termed as the hidden layers.

The CNN model build for predicting the handwritten characters consists of 8 layers. It is built using Keras, an open source neural network library written in Python, which runs on top of Tensorflow. Tensorflow is an open source software library used in the production of deep learning models. Keras is a high-level neural networks API that runs Tensorflow at its backend, and is user-friendly when compared to Tensorflow. Tensorflow is a level API consisting low-level implementations of algorithms and involves complex mathematical operations.

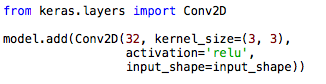
One of the major data structures of Keras is a model, which is used to build and organize layers. The model built in this project is a sequential model, i.e. a linear stack of layers. For complex architectures, Keras provides a functional API.

**

The function Sequential() is called to create a sequential model that would create a linear stack of layers.

The following APIs are used to build each layer. The method .add() adds the layers to the sequential model.

1. **First layer (Convolutional layer):**

**

* The Convolutional 2D layer is used for spatial convolution over the images. It creates a convolution kernel and since it is the first layer, it takes in the input as specified by the input\_shape parameter and produces a tensor of outputs.
* The first parameter represents the number of neurons (filters) used, i.e. 32 here.
* The second parameter represents the kernel size or the window size to be used, which represents the local receptive field. A kernel size of 3\*3 is used.
* The activation function used here is Rectified Linear Unit (ReLU). It replaces the negative values in the feature matrix to zeros and retains the positive values.

1. **Second layer (Convolutional layer):**

**

* The second layer is also a Convolutional 2D layer. It takes its input from the previous layer’s output.
* It uses 64 neurons.
* The kernel size used is 3\*3.
* It also uses Rectified Linear Unit as its activation function.

1. **Third layer (Pooling layer):**

**

* Pooling reduces the dimensionality of the feature map by condensing the output of small regions of neurons into a single output.
* MaxPooling2D is used to operate on spatial data.
* The parameter, pool\_size represents the factors by which to downscale (horizontal, vertical). A pool\_size of 2\*2 halves the input in both the spatial dimension.

1. **Fourth layer (Dropout):**

**

* Dropout is used for regularization. It avoids the network in becoming too smart to in learning the input data.
* It randomly sets a fraction rate of the input units to 0 at each run during the training process, hence prevents over-fitting.
* The parameter represents the input units to drop, which is 0.25 in this level.

1. **Fifth layer (Flatten):**

**

* This layer flattens the structure of the feature map to create a single long feature vector that will be used by the dense layer for classification.
* It is used to give a very small output prediction for example, a label instead of an array.

1. **Sixth layer (Dense layer):**

**

* It is a fully connected layer. Every neuron of this layer is connected to every neuron in the preceding layer.
* Dense performs the operation, output = activation(dot(input, kernel)) + bias
* The activation function used is Rectified Linear Unit (ReLU)
* Kernel represents the weight matrix used by this layer.
* Bias is a vector created by this layer and it is applicable only when the value is true. There is no bias specified in this layer for the CNN prediction model.
* The parameter 128 represents the dimensionality of the output space.

1. **Seventh layer (Dropout):**

**

* It is used for further regularization.
* In this layer, a dropout fraction of 0.50 is used to represent how much of the input units should be dropped.

1. **Eighth layer (Dense):**

**

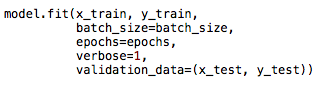
* The output layer is a fully connected layer.
* The number of character classes, i.e. 47, represents the dimensionality of the output space.
* The neurons of this layer are activated by the softmax function. Softmax function is the generalization of the logistic function. It is also called as normalized exponential function. Usually an activation function results in a single output for a single input, whereas the softmax function results in several outputs for a single input. The sum of the outputs of this function is always 1. In this CNN prediction model, the output is a set of probabilities for each character class, with one character class having the highest probability.

Once the structure of the model is built, it has to be compiled. Keras provides .compile() API for this purpose.

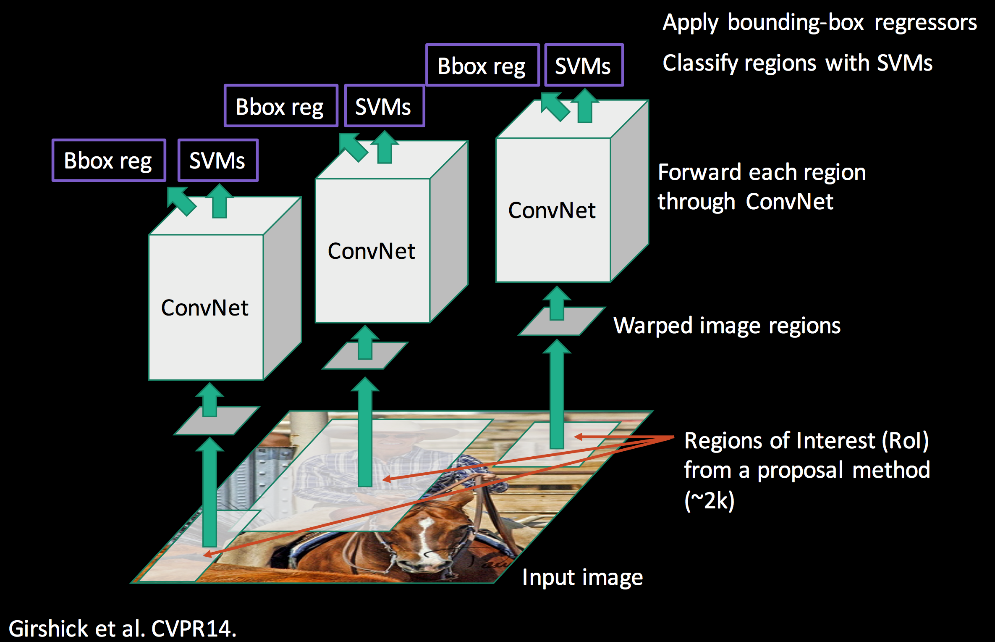
**

* This API configures the model for training.
* The first parameter represents the loss function, which is also referred to as the optimization score function. Since the problem is a multi-class classification problem, the appropriate loss function is categorical\_crossentropy.
* The optimizer controls the gradient clipping and aims to minimize the loss function. Adadelta is and optimization algorithm which uses the magnitude of recent gradients in order to obtain an adaptive step rate.
* The last parameter represents the metrics that is evaluated for performance during the training and testing process. The metrics used is accuracy.

Once the model is compiled, it is trained with the training dataset and tested against the testing dataset. Keras provides .fit() API for training the model.

**

* It takes in the training dataset as specified by the x\_train and y\_train.
* Batch\_size is the number of samples required per iteration during training, which is 1000.
* The training is performed for a given number of iterations as represented by the epoch value, i.e. 20.
* Finally, the model is validated against the validation or training dataset as specified by x\_test and y\_test.

**6.1.2. Regional Convolutional Neural Network (R-CNN)**

*Fig. 6.1. R-CNN*

The Objective of R-CNN (Regional Convolutional Neural Network) is to accept an image as input, and accurately identify where the main objects in the image. These proposed images also called as Region of Interest (RoI) bounded by Boxes. Each Image contained within these bounded boxes is input to the CNN (Convolutional Neural Network), which is a deep, [feed-forward](https://en.wikipedia.org/wiki/Feedforward_neural_network) [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network) and is used to analyze visual imagery here.

The Region of Interest (RoI) is generated using OpenCV API, which is a library that includes several hundreds of computer vision algorithms. Here we use OpenCV for detecting and extracting each character in the handwritten image. The following functions are performed on the handwritten image to generate the region-of-interest (RoI), which is later passed to the convolutional neural network for character prediction:

**1. Gaussian Blurring:**

Blurs the image using Gaussian filter

**

* The first parameter is the input image
* The second parameter represents the width and height kernel
* The third parameter is the standard deviation in X and Y direction represented by sigma X and sigma Y respectively.  If only sigma X is specified, then sigma Y value is equated to sigma X value.

**2. Adaptive Thresholding:**

Adaptive thresholding algorithm calculates the threshold for a small region of the image. So, we get different thresholds for different regions of the same image and if pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). Thus, it gives us better results for images with varying illumination.

**

* The first parameter is the input grayscale image.
* The second parameter is the maximum value of the pixel threshold.
* The third parameter is threshold value, which is the weighted sum of neighborhood values where the weights represent the Gaussian window.
* The fourth parameter represents the function transforms a grayscale image to a binary image.
* The fifth parameter is the block size which represents the size of neighborhood area
* The sixth parameter is a constant C, which is subtracted from the mean or weighted mean calculated.

* *

*Fig. 6.2. Original Image and Adaptive Gaussian Thresholding*

**3. Color Inverting Image:**

The Grayscale image generated from thresholding is color inverted to change the white color to black and vice-versa.

**

* The parameter for this function is the image data, which is inverted.

**4. Contour Detection:**

The Contour detection is used to find the outlines of the characters, which are represented as curve joining all the continuous points of the characters of the image. These contours detection acts as important tool for shape analysis and object detection and recognition.

**

* The first parameter is a 8-bit single channel image input.
* The second parameter represents the Contour retrieval mode. Here, we use the **CV\_RETR\_EXTERNAL** function, whichretrieves only the extreme outer contours.
* The third parameter is the Contour approximation method. Here, we use the **CV\_CHAIN\_APPROX\_TC89\_KCOS** which is one of the flavors of the Teh-Chin chain approximation algorithm

**5. Bounding Rectangle and Cropping**

The Bounding Rectangle calculates and returns the minimal up-right bounding rectangle for the specified point set which are the set of contour points detected in the previous stage.

Each bounding rectangle contains a single handwritten character which is represented by 4 values x, y, w, h. The values represent the x co-ordinate, y co-ordinate, width and height of the bounding rectangle respectively with respect to the original handwritten document.

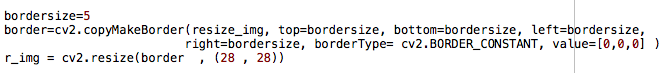
These values of x, y, w, h of each bounding rectangle is used to crop-out each handwritten character and save them as separate image.

**

* The parameter to bounding rectangle is 2D point set.
* The image is cropped based on x, y, w, h values.

**6. Padding & Resizing Image**

Each of the cropped images contains handwritten character till the edges of the image. So, we must add padding to the image so give better prediction of the character. After padding the image with black color, it is then resized to 28 \* 28 resolution so that the image can be used as an input to the CNN (Convolutional Neural Network) Model.

**

For Image Padding:

* The first parameter is the input image
* The next four parameters represent the border width in number of pixels in corresponding direction.
* The sixth parameter represents the border type. Here we use the **cv2.BORDER\_CONSTANT** function which adds a constant colored border
* The last parameter is the value passed to the function defined previously.

For resizing Image:

* The first parameter is the input image.
* The next two parameters represent the output image resolution.

**CHAPTER 7**

**TESTING**

System testing is a level of software testing where the complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. System testing can also be defined as the process of testing an integrated system to verify that it meets the specified requirements. It performs a critical role on assuring quality and reliability of the software. To ensure that the program works as predicted, it has to be checked with a set of test cases and the result is evaluated. For future references, errors will be corrected using the following stated testing steps and the results will be recorded. All the test cases have to be evaluated before the implementation phase.The security, correctness, quality of the developed computer software and completeness are identified during the system testing process.

**7.1. Unit testing**

Each component of the system is tested individually. The system is first tested using the test cases and the results are compared with the predictions. It focuses on every small unit of the software during verification. Hence it is also known as module testing.

**7.2. Integration testing**

It is performed in order to detect faults in the system components. The units obtained from the previous test are gathered together and tested, which helps to resolve the errors easily. All the modules are then integrated to form the entire software system.

**7.3. System testing**

It is performed to check if the entire system meets the users’ needs.

**7.4. Validation testing**

It is performed in different ways. It succeeds when the product works in a way that can be realistically expected by the client.

**7.5. Black Box testing**

This testing is performed for the following:

* To discover absent or mistaken capacities.
* To discover a blunders of interface.
* Execution faults are recognized.
* Blunder with respect to initialization and the end result.

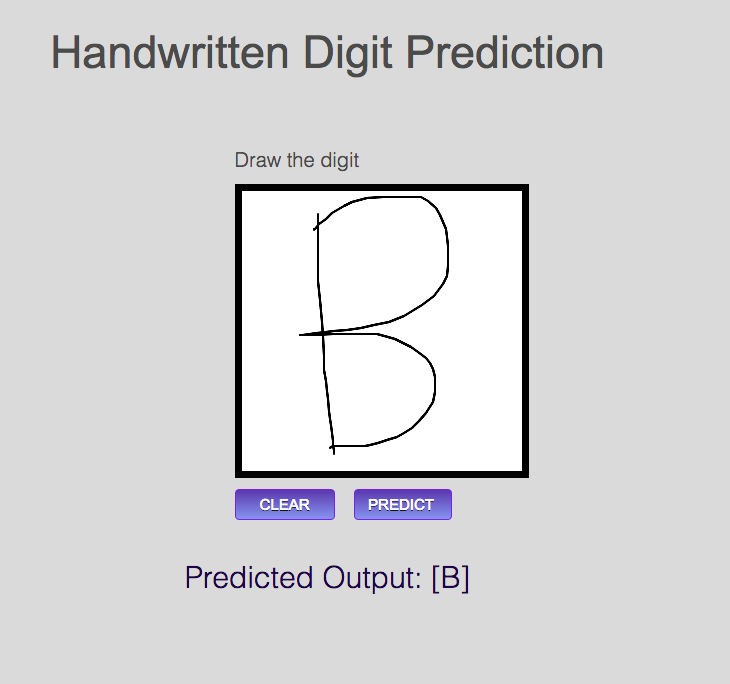
**7.6. White Box testing**

This testing is performed for the following:

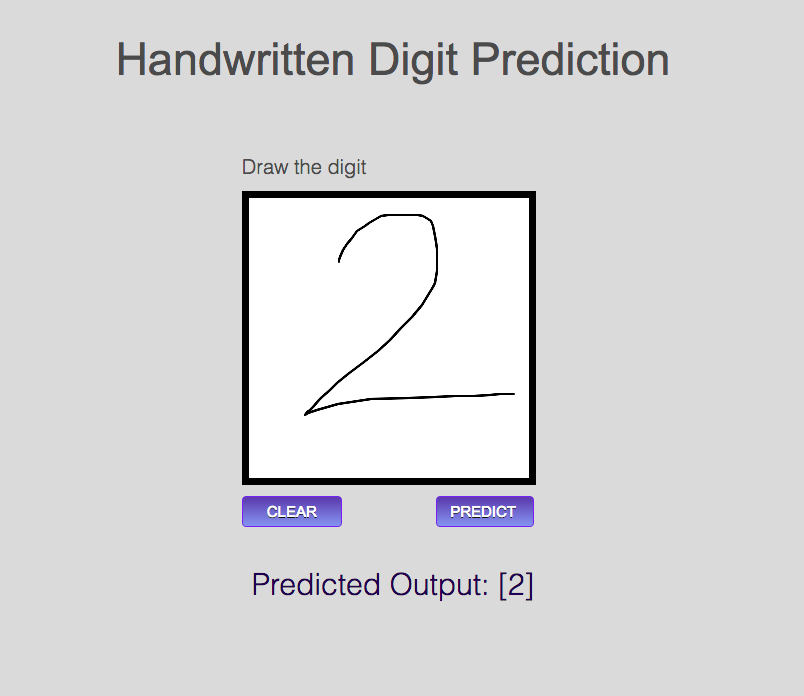
* To determine whether all individual modules inside the system have been utilized at least once.
* Implementing all reasonable choices.
* Execution of the loop.
* Check the internal information to guarantee validity.

**7.7. Test Results**

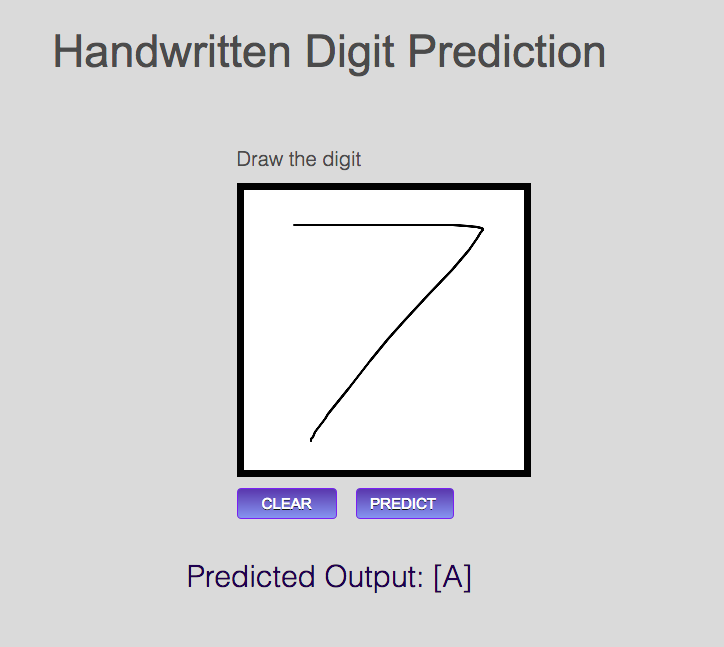
During training phase, the CNN model gave an accuracy of 89.97% for the testing dataset. For testing against the user’s handwritten character, a web application was built using Python Flask API, where the drawn on a web page using the cursor. The CNN model predicted 60% of characters correctly. The following are some of the test results obtained.

**

*Fig. 7.1. Correct prediction of ‘B’*

**

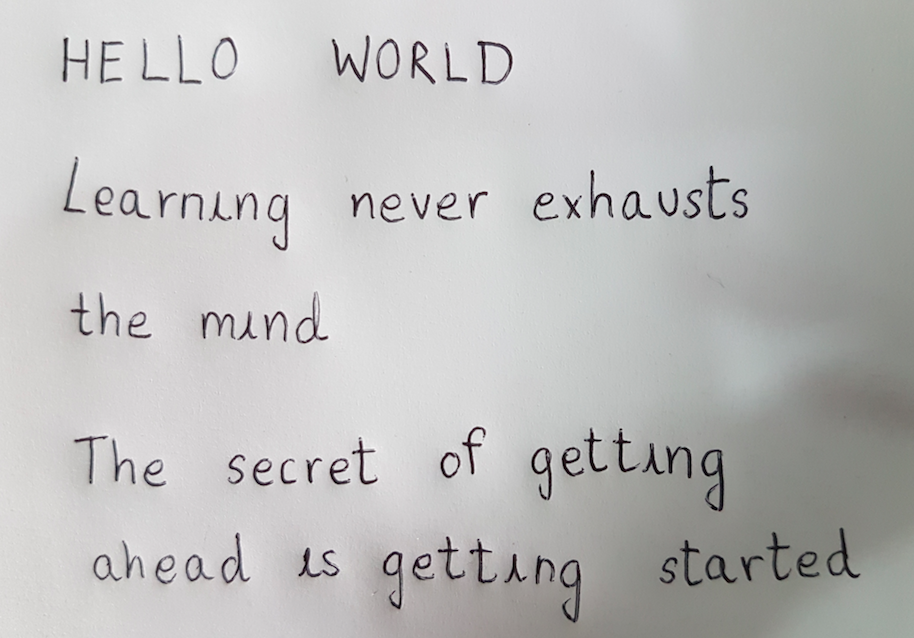
*Fig. 7.2. Correct prediction of ‘2’*

**

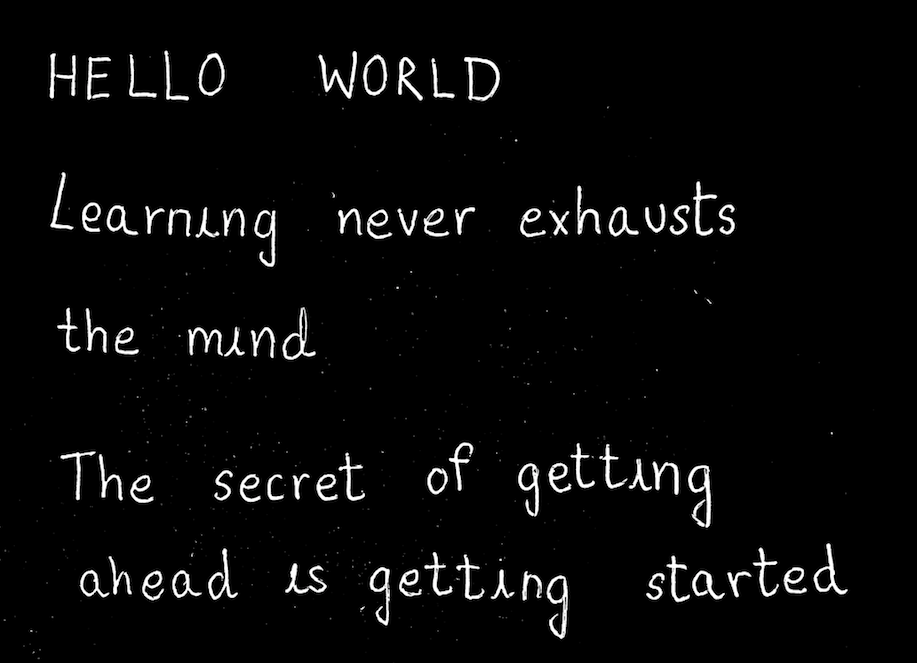
*Fig. 7.3. Incorrect prediction of ‘7’*

**CHAPTER 8**

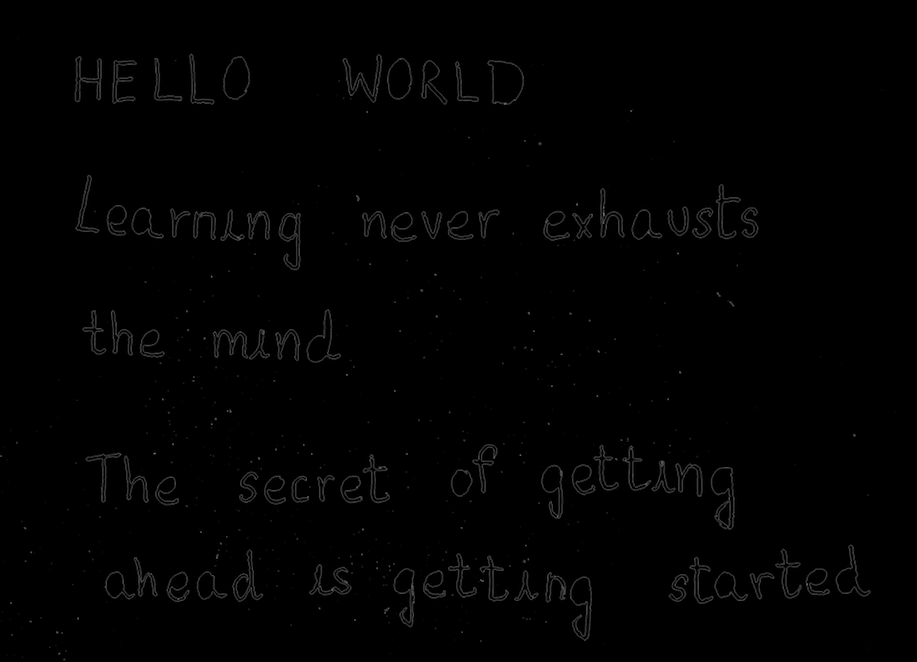
**SNAPSHOTS**

******

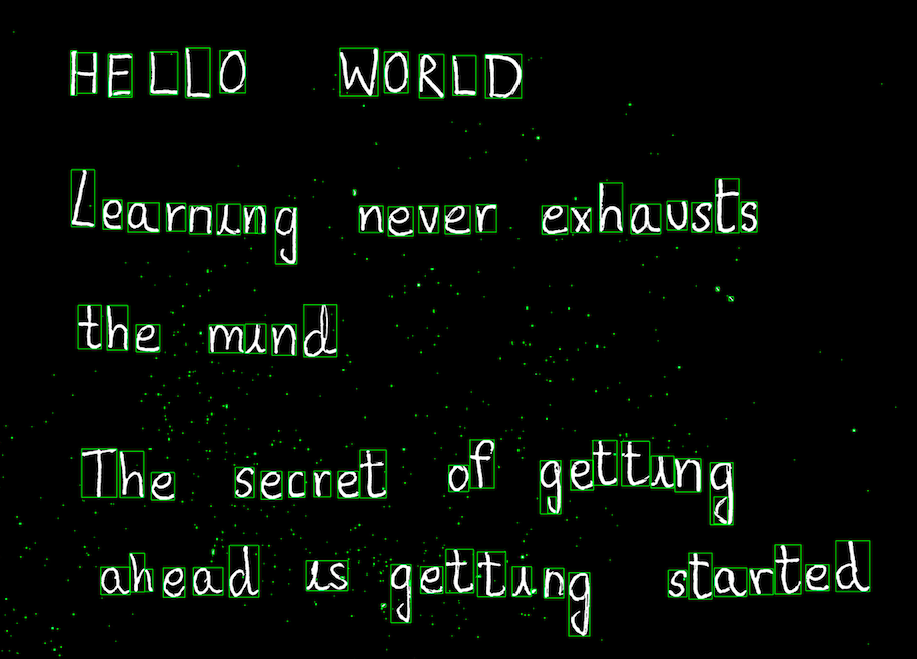
*Fig. 8.1. Input Image*

******

*Fig. 8.2. Color Inversion and Adaptive Thresholding*

******

*Fig. 8.3. Masking Contours*

******

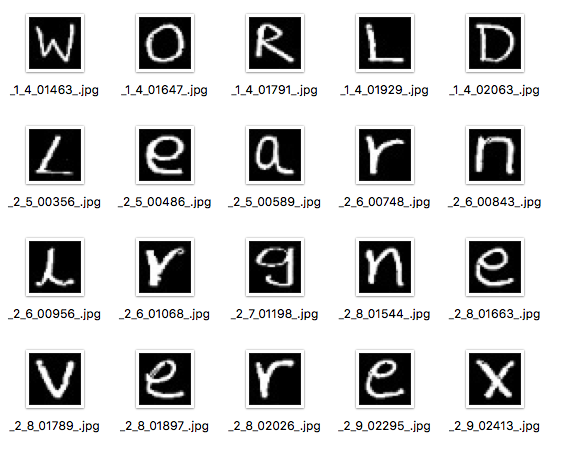
*Fig. 8.4. Bounding Boxes*



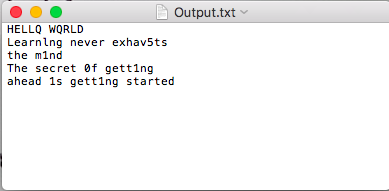
*Fig. 8.5. Segmented Characters*

******

*Fig. 8.6. Predicting Line Numbers*

******

*Fig. 8.7. Predicting Word Numbers*

******

*Fig. 8.8. Output Text Document*

**CHAPTER 9**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**9.1. Conclusion**

The system developed in this project converts the handwritten text document into the digital text document using Regional Convolutional Neural Network (R-CNN). The existing systems that use machine learning convert only individual handwritten characters into the corresponding ASCII characters. We have developed a system that performs the conversion at the document level and generates a text document as the output.

Several difficulties were faced while building the system. It wasn’t easy to predict the number of layers to be used in the CNN model, and also the number of nodes in each layer.

Deciding the parameters for training the model was a hectic task because the accuracy of model depended completely on them. The training batch size and the epoch values had to be chosen appropriately such that it falls within the gradient descent of the training curve. Under-training degrades the accuracy, while over-training leads to over-fitting of the input data and fails to recognize new patterns. Any modification to the training file requires re-training for a minimum of 6 hours.

The goal of image preprocessing was to remove noise and artifacts completely from the image before performing character segmentation. But preprocessing didn’t result in the complete removal of noise. This affected character segmentation and required some manual removal of unwanted images. The final output was sufficiently good.

**9.2. Future Enhancements**

The system can be further enhanced in the future that overcomes all the drawbacks of the proposed system by implementing the following modules.

**9.2.1. Dataset**

The EMNIST dataset was originally derived from NIST Special Dataset 19, which had images of size 128\*128. During the conversion of the NIST dataset into the 28\*28 EMNIST format resulted in data loss. This affected the overall output accuracy. Therefore, datasets with higher pixel density can be used to produce better results. Using datasets with varied handwriting styles including cursive writings, the system can detect majority of the handwritten texts.

**9.2.2. Back Propagation**

Back propagation in Artificial Neural Networks is used to calculate gradient that is needed in the recalculation of weights with respect to the previously estimated weights in each neuron in order to decrease over-fitting and increase efficiency. In the proposed system, back propagation was implemented only during the testing phase. It should be implemented after deployment too so that the system can also learn from the new user inputs.

**9.2.3. Natural Language Processing (NLP)**

Natural Language Processing is a branch of Machine Learning, which is used to analyze and synthesize the natural language and speech. The post processing of the generated output text can be performed to make the text more grammatically readable using NLP, which inserts special characters such as comma, period, etc.

**CHAPTER 10**

**REFERENCES**

* https://arxiv.org/pdf/1702.05373
* https://www.kaggle.com/crawford/emnist
* https://github.com/Coopss/EMNIST
* https://docs.opencv.org/3.4/d7/d4d/tutorial\_py\_thresholding.html
* https://docs.opencv.org/3.3.1/d3/d05/tutorial\_py\_table\_of\_contents\_contours.html
* https://keras.io/
* https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4
* https://www.udemy.com/machinelearning/
* https://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf
* http://web.stanford.edu/class/cs20si/syllabus.html