Semantic Text Extraction from CAPTCHA Using Neural Networking

by

Iftekhar Kabir Chowdhury
18101463
Md. Nahiyan Jarif
18101348
Sadman Showmik
18101124
Farah Farhin Oishi
18101033
Afif Bin Jinnat
18101047

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that,

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Iftekhar Kabir Chowdhury
18101463

Md. Nahiyan Jarif
18101348

Sadman Showmik
18101124

Farah Farhin Oishi
18101033

Afif Bin Jinnat
18101047

Approval

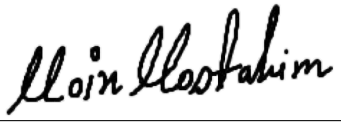
The thesis titled "Semantic Text Extraction from CAPTCHA Using Neural Networking" submitted by,

1. Iftekhar Kabir Chowdhury (18101463)
2. Md. Nahiyan Jarif (18101348)
3. Sadman Showmik (18101124)
4. Farah Farhin Oishi (18101033)
5. Afif Bin Jinnat (18101047)

of spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of BSc in Computer Science and Engineering on May, 2022.

Examining Committee:

Supervisor:
(Member)



Mr. Moin Mostakim
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Ms. Arnisha khondaker
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

The proposed here in this paper is a novel work.Furthermore, we the members formally and solemnly swear that this thesis was written using the results of our exhaustive research. This report appropriately acknowledges and cites all of the materials that were used. No other individual has ever submitted this research work, in whole or in part, to another university or institution for the purpose of receiving a degree or for any other reason.

Abstract

CAPTCHA stands for Completely Automated Public Turing Test to distinguish Computers and Humans Apart. CAPTCHA is used for a variety of reasons, including internet security. There are various CAPTCHA methods available nowadays, including text-based, sound-based, picture-based, puzzle-based, and so on. The most prevalent variety is text-based CAPTCHA, designed to be easily recognized by humans, frequently used to separate people from automated applications, and challenging to understand by machines or robots. However, as deep learning advances, it'll become much easier to create Convolutional Neural Network (CNN) models which will successfully decipher text-based CAPTCHAs. The CAPTCHA-breaking workflow consists of attempts, techniques, and enhancements to the computation-friendly Convolutional Neural Network (CNN) version that aims to reinforce accuracy. In comparison to the break of the whole CAPTCHA shutter at an equivalent time to separate CAPTCHA images for individual characters from 2 pixels on the corner of the sector with a replacement set of coaching data, then offered an efficient division of the network separation to interrupt the transmission of CAPTCHA text. Semantic textual content segmentation may be a natural step in developing coarse to first-class inference. The inspiration is often placed in classification, which creates a prediction for a whole input.

Keywords: CAPTCHA, Convolutional neural networks, Recurrent Neural Networks.

Dedication

This thesis is dedicated to our beloved parents and our department's respected faculty, who have supported and encouraged us throughout the thesis and inspired us to seek excellence in all areas.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis has been completed without significant interruption.

Secondly, our supervisor, Mr. Moin Mostakim sir, and co-supervisor Ms. Arnisha khondaker miss aided us with their kind assistance and advise anytime we required it in our work.

Finally, without our parent's support, we may not be able to achieve our goals. We are currently on the verge of graduating thanks to their kind assistance and prayers.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ANN Artificial Neural Networks

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart

CNN Convolutional Neural Networks

DCNN Deep Convolutional Neural Network

OCR Optical character recognition

RNN Recurrent Neural Networks

Chapter 1

Introduction

CAPTCHA is a test that can distinguish between human and machine programs on the Internet. Because it is extensively utilized and simple to deploy, text CAPTCHA is still quite popular. The majority of text-based CAPTCHAs are made up of uppercase and lowercase English letters (A-Z) and digits (0-9). Background noise, text distortion, rotation and warping, variable chain lengths, and letter merging are just a few of the technologies that have been devised to apply and improve text-based captcha. Because it is critical to define the image's content, semantic text within an image is very crucial. Compared to other semantic information, it is easily searchable and allows for applications such as picture keyword searches and text-based image classification[1].

There are two types of CAPTCHA recognition algorithms available: segmentation-based and non-segmentation-based. The two main processes in segmentation algorithms are segmentation[2] and character recognition[3]. The CAPTCHA image is separated into individual characters in the segmentation step, which the character recognition engine recognizes. The segmentation stage is regarded as the most critical because it impacts the whole system's accuracy and efficiency. However, most partitioning methods are inefficient and perform poorly. As a result, several researchers have begun to employ the partitioning CAPTCHA recognition technique to prevent the partitioning step's adverse effects.

No-segmentation methods currently dominate CAPTCHA recognition. The characters in the input CAPTCHA can be recognized and classified without separating the CAPTCHA into individual characters. Furthermore, the majority of segmentation models are highly accurate and efficient. However, neural network-based CAPTCHA recognition algorithms require enormous data sets to extract characteristics efficiently. Furthermore, non-partitioned models are more complicated and demand more storage space, especially as the number of characters in text CAPTCHAs grows. In figure:1.1 we have showed a simple extraction of captcha using models.

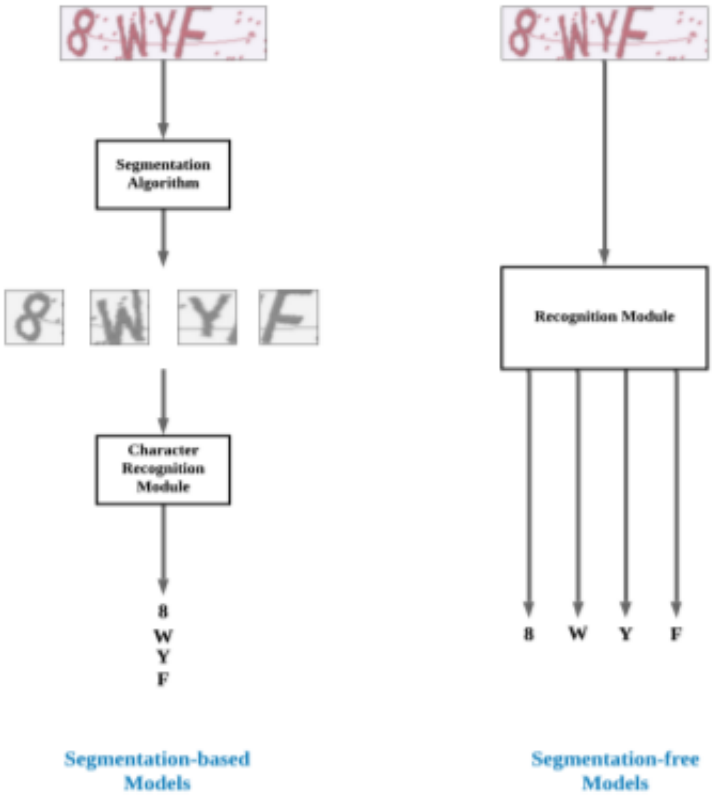


Figure 1.1: Extracting semantic texts using models

1.1 Research Problem

People communicate with each other through language and text. This process is called natural language. Computers understand this language of people through Natural Language Processing (NLP). It has made the interaction between people and computers very easy. Semantic analysis is under the field of NLP and machine learning, which helps understand the context of each text and understand the emotions. It helps to achieve vital information from the accuracy of the human level of computers. For the semantic analysis, we need semantic extraction. CAPTCHA is nearly universally used as a security measure on business websites. Many possible concerns with Captchas Imaging Recognition have yet to be extensively investigated. A small business may find it difficult to acquire a broad picture dictionary without access to and without automated acquisition assistance for new images designated; a challenge based on images does not generally answer the definition of a captcha. Understanding the various CAPTCHA approaches is essential. It's useful to categorize CAPTCHAs because there are so many distinct kinds. For simplicity, think about using a variety of text CAPTCHA codes. The most often used type of CAPTCHA is text-based. A string of recognized characters is presented to the user, but it is tough for the computer to decipher. Strings may be anything from simple text to extremely loud text. Text CAPTCHAs are available in various formats, including basic text CAPTCHAs that are easy to identify. This CAPTCHA is devoid of noise and distortion. The text in a distorted CAPTCHA is challenging for a computer program to detect since it is not straight but jumbled. It's unidentifiable due to

several noise patterns. CAPTCHA codes with complicated text are CAPTCHA codes that are very noisy, overlapping, and significantly deformed. Characters spin and distort dramatically. 3D CAPTCHA was given by Imsamai and Fimoltares. Both computers and humans have difficulty deciphering these CAPTCHA codes. CAPTCHA text security is frequently contested with solutions to combat machine recognition. The following are the critical characteristics of CAPTCHA text: a sufficiently big character set. The total number of CAPTCHA lines is only large enough to resist a forceful interruption when the character set is large enough. Characters with overlaying and overlapping distortions Using distorted, merged, and overlapping characters, the extraction algorithms can't simply separate the CAPTCHA picture into individual characters. Size, breadth, angle, location, and typeface differ amongst symbols. By analyzing the features of various characters, various changes can diminish recognition accuracy. Broken routes and empty signs Hollow characters have fewer characteristics than complete characters, and their fractured outlines can withstand the fill's onslaught. Complex backdrops have colors and forms similar to character colors and shapes. If the photographs fulfill these criteria, the noise will be difficult to eliminate. It might lower the accuracy of recognition. The functionalities described above successfully increase the security of text CAPTCHAs while also posing a considerable barrier to semantic CAPTCHA extraction research. Because blur makes the text stand out, picture filters blur the image horizontally and search for darker spots. The text outline, pattern detection, color picker, and other features are not included in edge detection filters. For these goals, we shall employ a color picker. It is a simple strategy that has shown excellent results and shown some promising outcomes. Multivalued image decomposition is the algorithm we'll utilize for our CAPTCHA example. In another way, we start by creating the image's color histogram. To do so, group all pixels by color and count each group separately.

1.2 Research Objective

There are many methods for recognizing where text lives in a picture and extracting it. In this research, we are aiming to do semantic extraction from captcha.

1. To understand what text Captcha is and why we need them.
2. To understand the security and privacy concerns that Captcha possesses.
3. To do semantic Text extraction from Captcha.
4. To deeply understand neural networks and their capabilities.
5. To find a better model of neural network modes that will help us to extract semantic text from Captcha.

1.3 Thesis Structure

In Chapter 1, an overview and introduction was discussed and the research problems along with research objectives. In chapter 2, research and related work on this

topic are mentioned. Then, in Chapter 3, the paper reviewed the design of CNN and RNN and ResNet50. It addressed the methodology, dataset, and pre-processing techniques and the layers of CRNN model in Chapter 4. Later, Chapter 5 fully describes the implementation of all the methods in this paper. Furthermore, in chapter 6 the results and comparison are shown of all the models. Additionally, in chapter 7, all challenges and difficulties experienced throughout the thesis while compiling the models is explained. Finally, along with plans for the future the conclusion is stated in chapter 8.

Chapter 2

Related works

This section tries to critically review the previous relevant research regarding text extraction from CAPTCHA. The most goal of text extraction from CAPTCHA is to enhance the text extraction method that's available now. The concept of text extraction from CAPTCHA represents extracting text from CAPTCHA from various texts that are widely used nowadays. Text-based CAPTCHA recognition systems are still commonly to break CAPTCHAs. The segmentation step is the most vital part of these text-based models' recognition processes. There are several algorithms proposed for segmenting text-based CAPTCHAs into distinct characters. For CAPTCHA segmentation, Zhang et al.[4] used the vertical projection method[5], [6], [7]. The vertical projection histogram was improved by combining characters' size features and locations with the vertical projection histogram when handling conglutination characters. They also talked about the way to segment different types of conglutination. Chellapilla and Simard [8] segmented several CAPTCHA schemes, including Yahoo and Google, using the connected component algorithm [9], [10] successfully rates starting from 4.89 percent to 66.2 percent. On the opposite hand, vertical projection and connected component algorithms necessitate several computationally expensive and time-consuming preprocessing operations. Another CAPTCHA segmentation method presented by Hussain et al. [11] is recognition-based segmentation. The first stage was to train an ANN to recognize manually cropped CAPTCHA characters. Cutting a CAPTCHA photo into parts and editing the characters with sliding windows became second nature to the professional ANN. This technique of segmentation includes applying the trained ANN to a large number of extracted sub-windows in order to compute their percentage of confidence, which can take a long time.

The character recognition module is also regarded as critical since it has an impact on the popularity and accuracy of segmentation-based CAPTCHA recognition systems. Sakkatos et al. [12] used the template matching [13], [14] approach to recognize characters by comparing different characters to template characters. Unless more complex methods are used, errors resulting from character similarities are deemed weak by template matching. As a method of recognizing CAPTCHA characters, Chen et al. [15] proposed "chosen learning confusion class" (SLCC). To detect CAPTCHA characters, SLCC uses a two-stage DCNN frame. The DCNN all-class technique is used to categorize the characters initially. The confusion class subsets employ a confusion relation matrix and a group partitioning algorithm.

Despite the CAPTCHA character recognition approach's high character recognition accuracy, particularly for confusing-class characters, allocating a replacement DCNN to each confusion class subset might considerably boost the overall system's storage capacity. DCNN has had a lot of success with CAPTCHA recognition in the last few years [16]. In previous research, many strategies were wont to improve DCNN's overall performance. Dropout should be modified to stop complicated co-adaptations on the training data, consistent with Hinton et al. [17], by eliminating half the feature detectors on each training sample randomly to limit overfitting. The drop connection was developed by Wan et al. [18] to line a randomly chosen portion of weights inside the network to zero. Each unit within the preceding layer receives input from a random subset of the units. Oversampling may be a viable strategy to scale back the impact of coaching data imbalances, consistent with Hensman and Masko [19]. It's used on unbalanced training sets to extend performance to the balanced group. Howard [20] looks into a couple of techniques for enhancing the present DCNN-based picture categorization process. More picture changes could also be added to the training data, and complementary models are often used on photographs of higher quality. Wang et al. [21] have developed a particular method for improving DCNN discriminability. The first concept is to create a tree structure that will learn fine-grained features over time to acknowledge a subset of classes by discovering the traits that these classes share.

To establish whether the top user may be a human or a robot, a Semantic Text CAPTCHA [22] is required. However, with recent advances in neural networking, creating Neural Network Models (CNN) capable of successfully identifying CAPTCHAs has become considerably more manageable. Research into the popularity of Semantic Text CAPTCHA images is critical because it aids in the identification of flaws and gaps within the CAPTCHA generated, which results in the avoidance of those flaws in freshly built CAPTCHA producing systems. Ground noise, text distortion, rotation and deformation, variable chain length, and character fusion are only a couple of the processes devised to make sure and reinforce CAPTCHA. A summary of the research on neural networking, image processing, and related topics will be provided to spotlight the areas where this thesis is presumably to contribute. The most commonly used phrases and theories in understanding convergence will also be displayed. The broad idea of neural networking will then be discussed, followed by how neural networking technology is often integrated to assure the security of systems and platforms.

Deep learning-based approaches are applied in nearly every facet of our lives, from surveillance systems to driverless vehicles, robotics, and even the recent global epidemic of COVID-19. They create Deep-CAPTCHA, a deep neural network architecture with specialized convolutional layers to address the CAPTCHA challenge. They [23] explain how to analyses, recognize, and break alphanumerical CAPTCHA pictures in detail. Input data pre-processing, output encoding, and the network structure itself are all part of the process. We have find some papers where captcha was extracted by Resnet50 but a single captcha was extracted not using a whole dataset. In maximum previous works we have seen that 4 letter captchas were extracted so we have tried to do 5 letter extraction.

Chapter 3

Background Analysis

Making neural network models (CNNs) that can successfully detect CAPTCHAs is getting simpler because of significant developments in neural networks. The research of Semantic Text CAPTCHA picture recognition is crucial because it aids in identifying flaws and gaps in CAPTCHAs generated so that these flaws can be avoided in the future CAPTCHA creation system. Background noise, word distortion, rotation and warping, variable chain lengths, and letter merging are only some of the techniques used to assure and enhance CAPTCHA.

With the rapid advancement of CAPTCHA technology in recent decades, the number of relevant literature sources is growing by the day. An overview of the literature on Neural Networks and Image Processing will be provided to highlight the areas where this thesis is most likely to contribute. The most commonly used phrases and theories in understanding convergence will also be displayed. Following that, the overall philosophy of Neural Networking will be discussed and how Neural Networking technology may be integrated to assure the security of Internet-of-things systems and platforms.

3.1 Neural Networking

The ability of neural networking algorithms to extract relevant features and functions from input pictures is excellent (Fig.2). They have a wide range of applications in image recovery and object recognition fields. Neural network approaches are fantastic for constructing reliable CAPTCHA recognition networks because of their excellent characteristics. Although some CAPTCHA identification algorithms utilize classic digital image processing methods, these methods still have flaws. Noise in input photos, for example, has a significant impact on the capacity to recover weak objects. As a result, sophisticated Neural Networking technologies are gradually replacing existing methodologies.

Text-based CAPTCHA security is compromised by interfering visual effects such as rotation, twisting, adhesion, and overlap. To protect text-based CAPTCHAs against machine recognition, a variety of strategies are utilized. The text-based CAPTCHA has the following important characteristics:

1. Only a large enough personality set can withstand the forceful breaking of the CAPTCHA strings.
2. Breaking strategies that employ characters with distortion, adhesion, or overlap cannot segment a CAPTCHA picture into single characters with ease.
3. Different differences in character attributes, like size, breadth, angle, location, font, and so on, may impair identification accuracy.
4. The categorization comes after that for lower case letters and also digits. (Figure:3.2)
5. CAPTCHA strings with variable lengths may make it harder to crack.
6. Hollow letters have fewer characteristics than solid characters, and distorted shapes can efficiently prevent filling attacks.
7. Classification for both uppercase and lowercase letters. (Figure:3.3)
8. If the complex backdrop of the image is similar in color and shape to the text, noise reduction, and recognition accuracy may be problematic.

The factors stated above successfully increase text CAPTCHA security while also creating a considerable obstacle for CAPTCHA cracking research. Figure (3.1,3.2,3.3) was taken from [24]

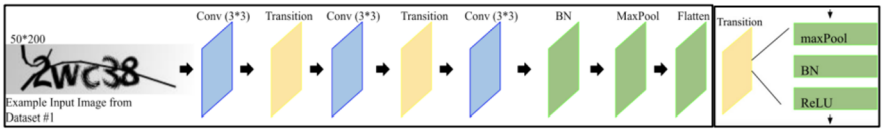


Figure 3.1: The Network Structure

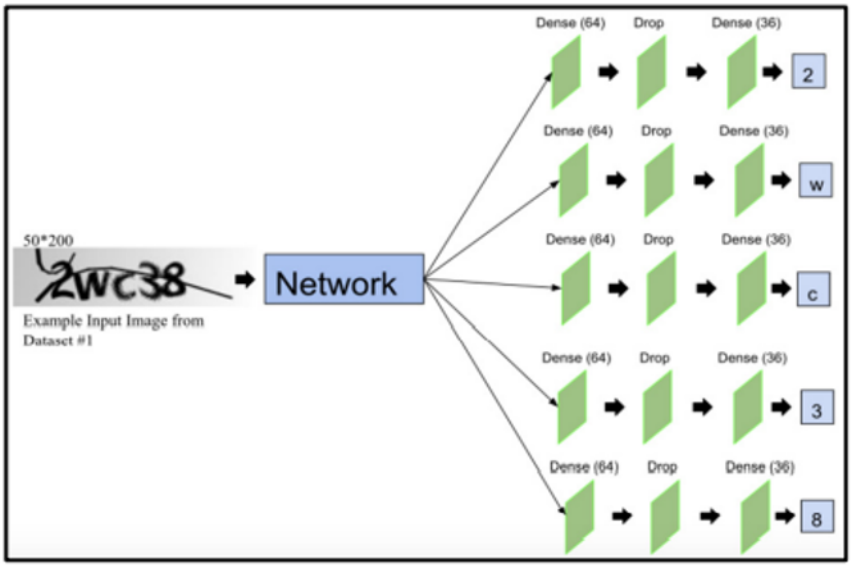


Figure 3.2: The categorization structure for lower case letters and digits alone

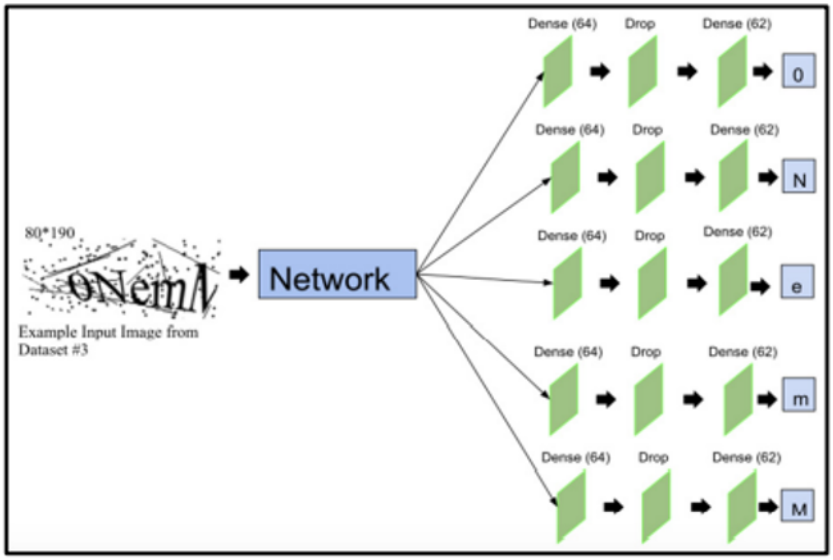


Figure 3.3: The structure of the classification for both uppercase and lowercase letters

3.1.1 Convolutional Neural Network (CNN)

CNN is just a Neural Learning method that can receive an image and prioritize or discriminate between various aspects/objects in the picture. Other classification techniques need far more pre-processing than ConvNets. ConvNets might learn these filters or characteristics with enough training, whereas simple approaches need to design filters by hand.

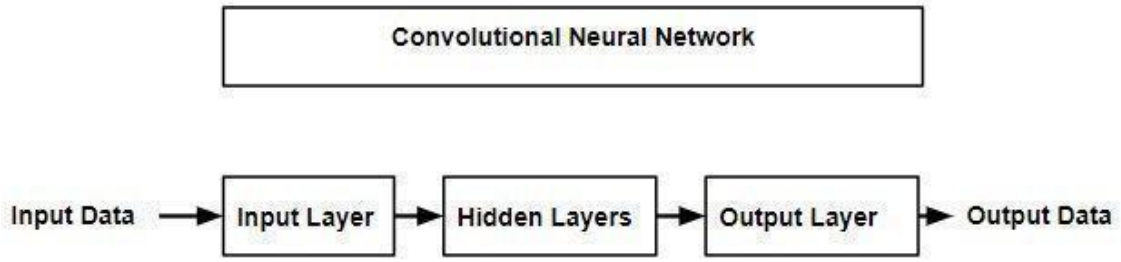


Figure 3.4: The basic block diagram of CNN

Convolutional neural networks comprise three layers: convolution, pooling, and fully coupled layers(Figure:3.4). The first two layers' extract features, while the third, a fully connected layer, transforms the data into a conclusion such as categorization. A convolution layer is a CNN component made up of a stack of arithmetic operations, such as convolution, which is a linear process.

3.1.2 Recurrent Neural Network (RNN)

A recurrent neural network, often known as an RNN, is a subtype of the convolutional neural network, which is an artificial neural network. The connections

between the nodes in this type of network can create either a directed or undirected graph along a temporal sequence. As a result, it can exhibit temporally dynamic behavior. RNNs are evolved from feedforward neural networks and can process input sequences of varying lengths by using their internal state (memory). RNNs function is by storing a layer's output and sending it back into the input to predict its outcome (Figure:3.5).

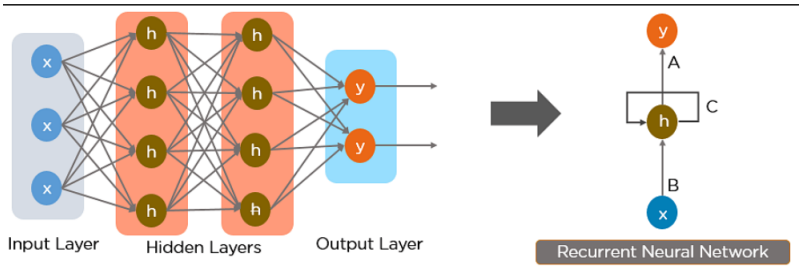


Figure 3.5: Simple Recurrent Neural Network

Information is looped back to the intermediate hidden layer in recurrent neural networks.

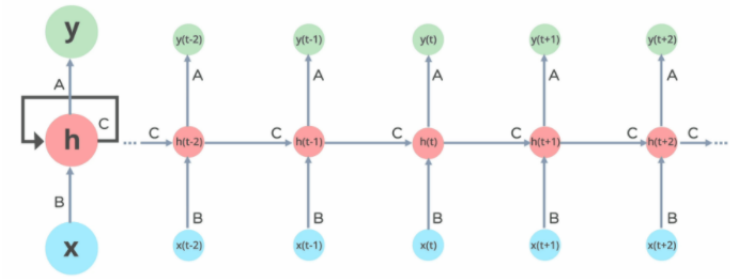


Figure 3.6: Working of Recurrent Neural Network

The working process of Recurrent Neural Network(Figure:3.6).

- The input layer 'x' receives and evaluates the neural network's input before passing it on to the intermediate layer.
- The intermediate layer 'h' has several hidden layers, each having its activation functions, weights, and biases. The neural network can be employed if the various parameters of succeeding hidden layers are unaffected by the preceding layer.
- The Recurrent Neural Network will standardize the different activation functions, weights, and biases, guaranteeing that each hidden layer has the same features. Instead of creating several hidden layers, it will generate one and loop over it as many times as needed.

3.1.3 CRNN Model:

- It is a hybrid model mixture of CNN and RNN. The CRNN-based optical character recognition (OCR) model suggested has the goal of recognizing and extracting alphanumeric captcha more quickly. For the model(Figure:3.7) [25] to accomplish this goal, it needs to be designed to receive data from images as an input and then methodically process that data before producing predictions. The model's general layout is depicted in the figure that can be found below.

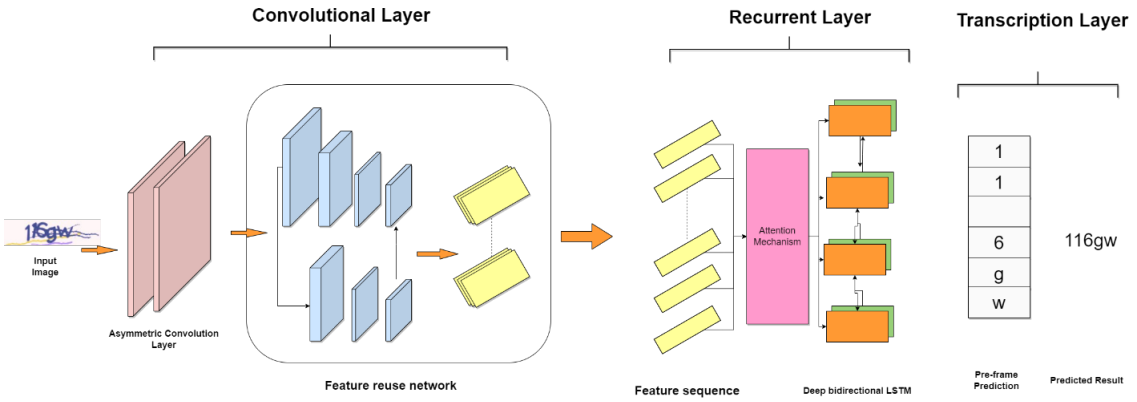


Figure 3.7: CRNN Model Architecture

- Feature extraction, sequence modeling, and transcription are all brought together by the design of this neural network. For this approach, character segmentation will be used as optional. Images are fed into convolutional neural networks, extracting various information from the images . The deep bidirectional recurrent neural network considers the characters and their relationships to one another when making label sequence predictions. The output of the RNN is transformed into a label sequence before being sent on to the transcription layer.

3.2 ResNet-50

After processing the input through many layers of neurons, a solitary output is created, along with the loss function. The loss function is then examined, and some internal adjustments are performed using variables. To accomplish so, you'll need a backpropagation algorithm. Backpropagation begins at the end node and ends at the beginning node. The function in this backtracking procedure is the derivative of that specific loss function multiplied by its previous function. When there are more levels in the process, traveling backward produces a lower and smaller gradient value. The derivative of that function multiplied by the primary function will be used for each iteration. The gradient value approaches 0 at some point. The "Vanishing Gradient Problem" is the result of this. This method makes it challenging to optimize the neural network layers. As a result, the accuracy plummets. A model dubbed "ResNet," which stands for the residual network(Figure:3.8), was devised to address this issue. It uses a feature known as the "skip function," which successfully connects the inputs to the convolution block's output while skipping intermediate layers for improved speed. The information will be coupled to both the convolution

and function-processing layers.

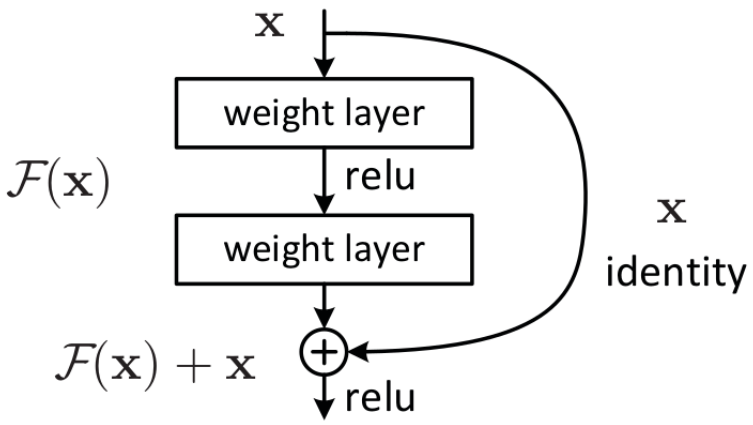


Figure 3.8: The Residual Network Block

A 50-layer convolutional neural community is represented via way of means of the ResNet50 version (CNN). The ResNet50 structure is based totally on the ResNet34 design, even though there's one predominant difference. Because of worries about the layers' schooling time, the construction block becomes reconfigured as a bottleneck. Instead of the preceding layers, a 3-layer stack becomes employed. As a result, the ResNet34 design's -layer bottleneck blocks have been changed with a 3-layer bottleneck block, yielding the ResNet50 structure. The 34-layer ResNet version is notably much less unique than this version. ResNet's 50 layers produce 3. eight billion FLOPS overall performance. ResNet y's output characteristic is the sum of the capabilities shaped via means of the convolution layers and the entry itself, as proven inside the diagram. The aim of the neural community is to offer the maximum correct output possible, as close to the entry as possible. This structure's cause is to make the F(x) fee 0, which allows you to make the version as correct as feasible. As a result, the output (y) will be the same as the enter (x) (x). As a result, the version is correct. Various neural community fashions for photograph categorization use the ResNet structure. There are numerous iterations of ResNet, which can be characterized by the number of layers every version contains because of the character of various demanding situations and accuracy requirements. ResNet-101, ResNet-34, ResNet-18, ResNet-50 and so forth are examples. To examine the entry and offer output, they are built with numerous layers and matrices of multiple dimensions. The ResNet-50 can concurrently teach approximately 23 million remarkable parameters. ResNet-50 is made from forty-eight convolutional layers with sixty-four specific kernels, one max pool layer with a long two strides, and one max pool layer with a length of four strides. ResNet-50 additionally consists of one max pool layer with a four stride length. Because each of those degrees could be duplicated and pasted in three instances, the final output could have nine layers. The subsequent layer consists of a wide variety of kernels and is reproduced three instances greater, growing the entire quantity of layers to 12. The following forty-nine steps are made from various kernels; each is used in numerous cases throughout the process. As a result, the layer above this one on this association is efficaciously

networked. It has capabilities of softmax characteristic and 1000 nodes, each of which assists us in producing a mean pool. This version can also additionally enter images from the ImageNet database to teach a pre-educated model of the community. There becomes a schooling session. The community can also additionally then be blanketed inside the version. As a result, the community now has more excellent complete know-how of function illustration for various pictures.

3.3 Layers

3.3.1 Convolutional Layer

The convolutional layer is responsible for applying various kernel filters to the input pictures and passing the filtered data to the subsequent layer, called the Python function known as Conv2d. Each kernel generates one unique image. The dot product is taken from the input image as the kernel moves over it and does its work. The image's feature map will be extracted as the primary focus of this convolutional layer's work. The species feature map served as the basis for creating the feature map. After that, the other layer receives this feature map as its input. A convolutional neural network has been used to extract sequence information from the input picture at the model's convolutional layer.

3.3.2 Max Pooling Layer

In many cases, a convolutional layer is the parent of a pooling layer. When feature maps created from a convolutional layer are extremely large in size, the cost of the calculation increases. The operation is slowed down as a direct consequence of this. The pooling layer is utilized to cut down on the size of the feature map, which in turn reduces the amount of computational effort required and speeds up the process. There are many different pooling algorithms available, and each one depends on the model.

3.3.3 Recurrent Layer

The result of the previous phase is utilized as input for the current phase in a recurrent neural network (RNN). The inputs and outputs of traditional neural networks are fundamentally dissimilar. The preceding words must be recalled in some situations, such as guessing the following word in a phrase. It is since traditional neural networks are incapable of remembering previous words. As a result, RNN was developed, and it was able to solve the problem using a Hidden Layer. The Hidden state of this type of neural network is the essential aspect since it remembers precise information about a sequence. The continual layer receives the image sequence feature fashioned by the convolution layer as an input. This feature aims to predict the label distribution for every enclosed picture sequence. During this model, the LSTM acts because of the RNN's continual unit. The forget gate, the input gate, and the output gate square measure the three gates that form up the Long immediate memory (LSTM). A Sigmoid network layer and an element-level multiplication

operation form up every gate. By selection, the Sigmoid network layer will select, add or eliminate info operations supported by the LSTM cell state.

3.3.4 Dropout Layer

The strategy of dropout is applied as a form of training technique. Because of this, some neurons are eliminated at random during the process. This indicates that any changes in weight are not communicated to the neuron on the return trip, and that any influence on the activity of neurons downstream is erased during the forward transit. Dropout is exclusively utilized throughout the training process for the models and is in no way used to evaluate their abilities. 2014, A Straightforward Approach to Preventing Excessive "Overfitting" in Neural Networks During the training process, the capabilities of the network are either increased or decreased depending on whether or not the outputs of a dropout layer are randomly subsampled. As a consequence of this, if you want to use dropout, you could need a more extensive network with more nodes. Dropout is another method that may be used to prevent the network from overfitting to the training data. After the convolutional and pooling layers comes the helpful dropout layer. The dropout layer is typically applied after the pooling layers, although keep in mind that this is only a recommendation. When the percentage of people that drop out of the survey falls below a certain threshold, the accuracy begins to progressively improve while the loss continues to decrease. When the rate of people leaving the study is higher than a predetermined threshold, the model loses its ability to fit correctly.

3.3.5 LSTM Layer

In situations involving sequence prediction, the LSTM network is a recurrent neural network that can learn order dependence. It is a quality that must be possessed to succeed in challenging problem domains such as machine translation, speech recognition, and others. The LSTMs that are used in deep learning are a tricky subject. The idea of long short-term memories (LSTMs) and how concepts such as bidirectional and sequence-to-sequence pertain to the area might be challenging to understand. The initial phase of the LSTM is to make use of the forget gate so that part of the information from the previous cell state can be filtered away. The input gate is used in the 2nd execution stage of the LSTM to incorporate some newly obtained info into the current cell state. In the third step of the LSTM process, the output gate is utilized to output the current implicit state. The predicted label distribution for the present time is the LSTM method's output.

3.3.6 Transcription Layer

The CRNN's very first layer, dubbed the transcription layer, converts the per-frame predictions generated by the recurrent layers into a label sequence. It is possible to train a CRNN using just a single loss function, despite its composed of two distinct network designs (namely, a CNN and an RNN). Transcription is the process of turning RNN's per-frame predictions into a label sequence. It can be done in several different ways. In the course of our transcription procedure, we will use a technique called Connectionist Temporal Classification (CTC) to decode the output of the

RNN and then transform it into a text label. The length of the label distribution may not match to the legitimate label length that corresponds to the image feature sequence after numerous operations through the convolution and recurrent layer. As a result, the training may not go as anticipated. As a result, the CTC approach must be used to de-integrate the label distribution from the final recognition result of the recurrent layer. To solve the problem of mismatched input and output labels, the CTC technique was created.

3.3.7 Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification, often known as CTC, is an output of neural networks that can be applied to the resolution of sequence problems, including those involving speech and handwriting recognition, in which the time varies. When you use CTC, the requirement for an aligned dataset is removed, making the training technique much easier to understand. CTC was written so that all that is necessary is the text displayed in the image, and it was done intentionally. It is not essential to consider the characters' width or location in a snap. The results of the CTC method do not require any additional processing after they have been generated. We can quickly acquire the network's outcome when we use decoding techniques. CTC addresses the three primary concerns of encoding text, calculating data loss, and decoding. To train the CRNN, we need first to calculate loss based on the image and the label it was given. The CRNN supplies us with a matrix that details the score of each character at each successive time step. For example, if an input picture's label is "n6p4t," the most prolonged label distribution that may be generated after CNN and RNN [26] procedures are 5. It means that a label can be associated with many possible character combinations. Each of these character combinations has the potential to be translated correctly or incorrectly into the label. The CTC was designed with a blank mechanism that inserts a separator between consecutively detected characters to alleviate this problem. The symbol is used to distinguish between the correct label sequence and the label sequence that is consistently recognized. As a result, the input label "n6p4t" can include values such as "m5p4t," "n6p44t," and so on. It is because blank character combinations are included in the system. Once several different varieties have been obtained, the data is translated into the ultimate recognition result.

3.3.8 Softmax Activation Function

Softmax activation is typically employed in the final layer of a neural network, rather than ReLU activation, sigmoid activation, then activation, or any other activation function. Softmax is a useful function for converting the output of your neural network's final layer into a probability distribution. Before returning the original vector of K real values, the softmax function transforms a vector of K real values to a vector of K real values with a total of one. The softmax approach converts positive, negative, zero, or greater-than-one input values to values between 0 and 1, allowing them to be understood as probabilities. The softmax will convert one of the inputs to a low probability if it is very little or negative, and a high probability if it is extremely big; it will always be between 0 and 1. Multi-class logistic regression or softargmax are other names for the softmax function. Because the softmax formula

is similar to the sigmoid function used in logistic regression, this is the case. The softmax function can only be used in a classifier when the classes cannot be merged. It's usually used to fit the output of neural networks within a range between zero and one. It is used to express the "probability" of certainty regarding the network's output.

3.3.9 ReLU Activation Function

The activation function is in charge of determining which neuron is stimulated first. The most widely utilized activation function is ReLU. A prominent non-linear activation function is the Rectified Linear Unit (ReLU). The most often utilized applications are multi-layer neural networks and deep neural networks. The highest integer between zero and the input value is the result of the ReLU algorithm. This is described in the ReLU equation. When the input is negative, the output equals zero, and when the input is positive, the output equals the input value. ReLU can be used to overcome the problem of disappearing gradients. Even when the number of layers rises, the problem of a vanishing gradient does not exist when utilizing the ReLU function. The addition of a brand-new feature almost always necessitates a significant product development. There are no such things as negative values since all negative integers are reduced to zero. Finally, because the ReLU function's derivative is equal to 1 for positive input, it may be used to speed up the training of deep neural networks, especially when compared to traditional activation functions.

3.3.10 Hidden Layers

A hidden layer is placed between the input and output of the algorithm in neural networks. This layer assigns weights to the inputs and routes them through an activation function as the output. To get down to brass tacks, the hidden layers apply a nonlinear transformation to the inputs of the network. The function of a neural network is what determines which layers are hidden inside the network, and the layers themselves can change depending on the weights that are linked with them. To put it another way, hidden layers are just many layers of mathematical functions, each of which is intended to give a unique output dependent on the final result that is sought. The activity of a neural network can be broken down into individual data modifications thanks to hidden layers, which make this possible. It is possible to tailor each function of a hidden layer such that it produces a specific impact.

3.3.11 Weight Layers

A neural network's input data can be modified by the weight parameter, which is located within the network's hidden layers. Nodes, also known as neurons, are the building blocks of neural networks and are the individual components that make up the network. Every node in the network has its own set of inputs, as well as its own weight and bias value. After a node in a neural network obtains an input, it performs the operation of multiplying that input by a weight value, and the product is either kept as is or sent on to the following layer of the neural network. A neural network's weights are frequently kept in the network's hidden layers. Consider weights as a

theoretical neural network to have a better grasp of their operation. The input layer of a neural network receives signals and passes them on to the succeeding layers. Following this, the input data is modified by a number of hidden layers that are contained within the neural network. The nodes of the hidden layers are each given a weighting that is ascribed to them. Before passing on the information to the subsequent layer, a single node, for instance, may first multiply the incoming data by a predetermined weight value. The output layer is also sometimes referred to as the final layer of the neural network. The output layer will routinely make adjustments to the inputs of the hidden layers in order to generate the necessary values within a predetermined range.

Chapter 4

Methodology

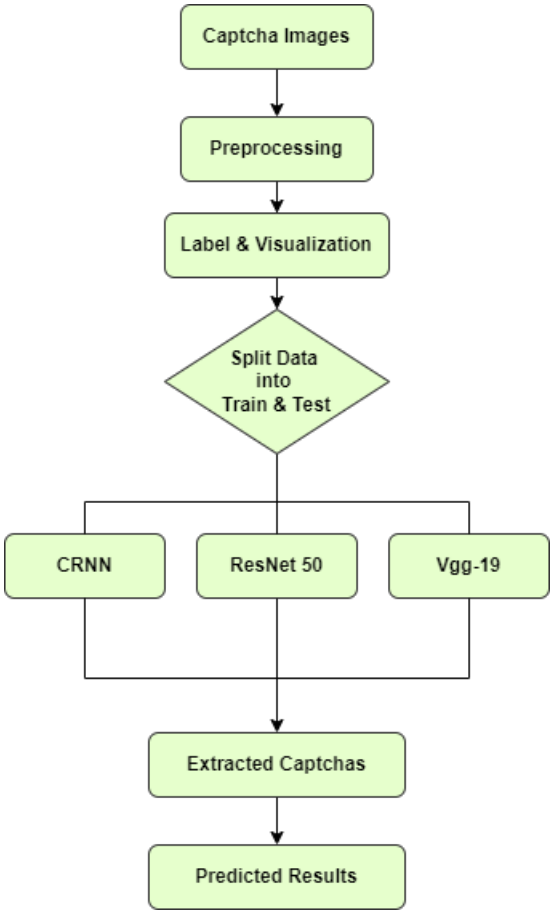


Figure 4.1: Work process of our model part 1

In (Figure:4.1) At first, we have taken a CAPTCHA image as input, and then some processing takes place for the next stages. After labeling the CAPTCHA, we see the complete visualization of the dataset. Then we split the data into train and test and we used the CRNN model to extracted the captchas. After extracting the captchas got a predicted result. We also used ResNet-50 and Vgg-19 model and run to extract the captchas. After that got the final prediction of the outcome

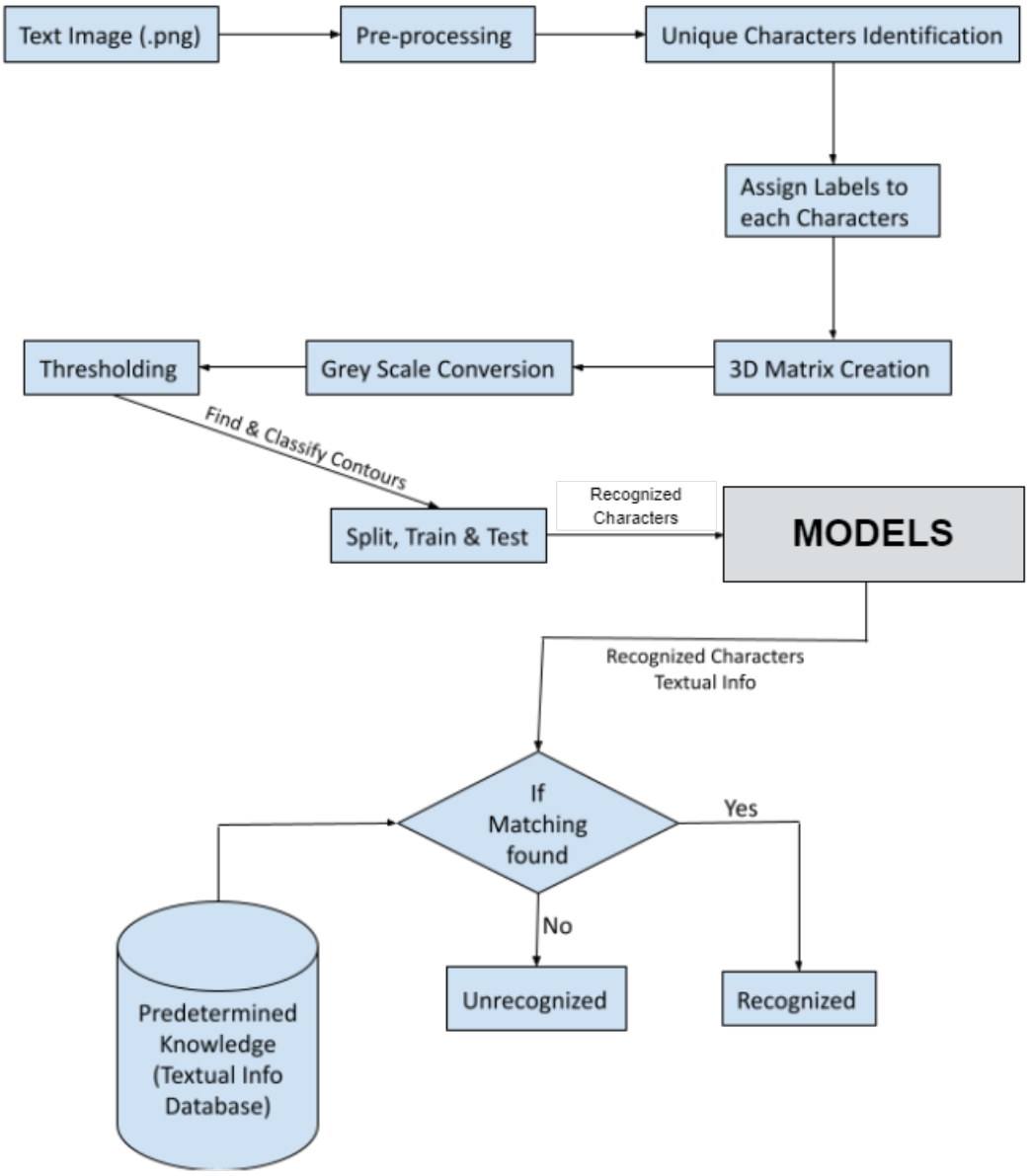


Figure 4.2: Work process of our models part 2

In (Figure:4.2) After the preprocessing (Image smoothing, image sharpening, shadow removal, and identifying the size and shape of characters), we have looked for unrecognized characters in that CAPTCHA. Then, after labeling and getting an overall visualization, we have created a 3D Matrix (size of image * text size * alphabet size) for all the CAPTCHA images. Next, we converted it into a grayscale image and did thresholding. After finding and classifying contours, we split the data into tests and train and run them. After that, we implemented the result in the selected models. Then we have searched for matching by comparing the output with the predetermined database, which contains all the textual information. If we find a match, it is stored as recognized (prediction if accurate), and if we find no matching, it will be stored as unrecognized (prediction is false). In this way, we have calculated the accuracy percentage and CTC Loss.

4.1 Dataset

The Kaggle Captcha Dataset [27] has 113,000 alphabetic and graphical images. We have selected 30,000 photographs from among them, and those picked images include alphanumeric images. We have split the data 80% and 20% .80% for training purposes 20% for testing purpose. Everything was uploaded in a zip file, and the files were given their specific name. It cuts down significantly on the amount of time needed to complete the pre-processing tasks.

4.2 Data pre-processing

Among the many pre-processing jobs, image size reduction, conversion to a different color space, and noise reduction filtering are just a few that have the potential to increase network performance significantly. Because the CAPTCHA image contains many blank spaces and a large number of interdependent pixels, the original size of the image data used in this investigation was 135 by 50 pixels. It is an unnecessarily large size because the CAPTCHA image contains many empty spaces. According to our research findings, if we reduce the image size to 67.25 pixels, we may be able to get practically the same outcomes without negatively impacting the system's performance. This size reduction helps speed up the training process since it minimizes the quantity of data while maintaining the same level of entropy as the original data.

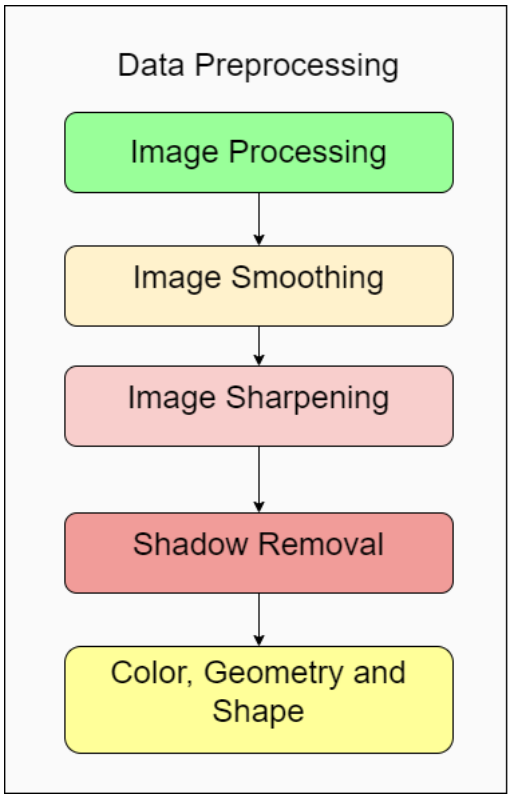


Figure 4.3: Steps of Data Preprocessing

Converting data from color space to gray space was yet another pre-processing method that we utilized to lessen the quantity of data needed to be processed while

keeping the same degree of detection precision. Because of this, the amount of redundant data might be cut even lower, which would also make the process of training and prediction much easier. Due to the fact that color is not taken into consideration by alphanumeric CAPTCHA systems, converting a three-channel RGB image to a grayscale image does not affect the results. The final pre-processing method that we shall investigate involves utilizing a noise reduction approach. After conducting an in-depth investigation into the various filtering methods, our team concluded that the most effective strategy would be to apply the default Median-Filter [25]. This method cuts down on visual noise by using the median value of the pixels close to the target pixel rather than the pixel itself. Below (Figure:4.3) the whole process is shown.

4.2.1 Image Processing

Applying specific techniques to an image to reinforce or extract valuable information is understood as image processing. The input used is a picture, and therefore the output is either an image or a characteristic or feature related to that image. Image processing techniques relate to any methods for improving the visual appearance of a given image or extracting specific components from thought for better human or automated system interpretation (Figure:4.4). Convolution and max pooling layer.

Figure 4.4: (a) gray scale image (b) blurred image (c) thresholder image and (d) eroded image

4.2.2 Image Smoothing

In noisy photos, image smoothing is commonly used. Picture noise is a negative shift in pixel values during image capture, transformation, and transmission. Noise-affected image pixels differ significantly from their surrounding pixels. Although noise cannot be completely eliminated, the smoothing technique can dramatically reduce it. The most typical method for noise reduction is to check a few nearest pixels of each pixel in the image, evaluate them, and apply a noise filter. When smoothing an image, the median filter is frequently used. With a designed mask, spatial filters separate strands of the input image(Figure:4.5). Gaussian smoothing is a powerful tool for eliminating Gaussian noise.

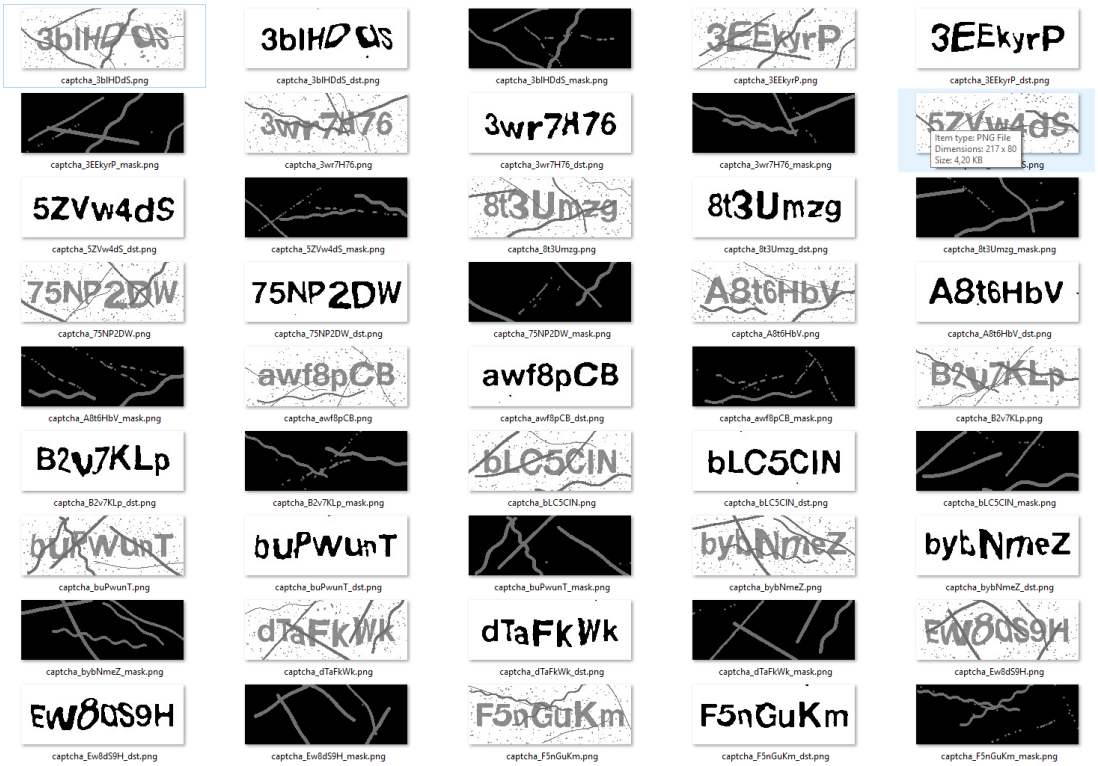


Figure 4.5: (a) Schematic diagram of convolution and max pooling layer.Cleaning up captcha image

4.2.3 Image Sharpening

Sharpness refers to the contrast between adjacent hues. It seems sharp if the color changes from black to white quickly. It also appears faded if the color changes gradually from black to gray and then from gray to white. Sharpening an image entails boosting the contrast along the boundaries where different colors collide. In a flash, image sharpening enables the high-frequency component. To improve the appearance of a picture, high-frequency elements such as the edges of objects should be sharp. Image sharpening adds the original image's filtered high-pass filter values to the original image, improving edges, and fine image detail and reducing high-frequency noise(Figure:4.6).

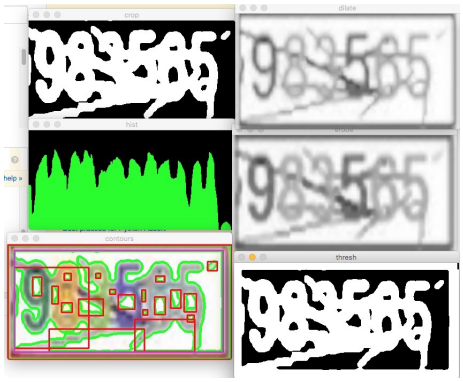


Figure 4.6: Preprocessing the grayscale image for removing tiny noise

4.2.4 Shadow Removal

The first stage in the shadow removal process(Figure:4.7) is to identify the shadow. Shadow detection techniques fall into two categories: model-based and performance-based. The most prevalent strategy is feature-based. The shaded zone in the image is defined by the standardized saturation value difference index, which suggests that image components in the shaded area have higher values than those in the non-shaded parts.

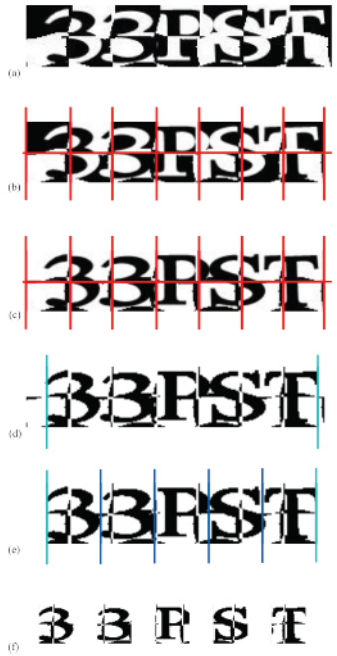


Figure 4.7: Initial CAPTCHA's and Shadow Removal process to Extracted Characters

4.2.5 Color, Geometry and Shape

Color, geometry, and form, which include main axis duration, minor axis length, circumference, size, and roundness, are the physical parameters of Captcha(figure:4.8). Color pictures using RGB to HSV conversions, morphological procedures, and other image processing algorithms, including derived histogram, segmentation, and subsequently binary image processing, are used to calculate the pixel density of each item.

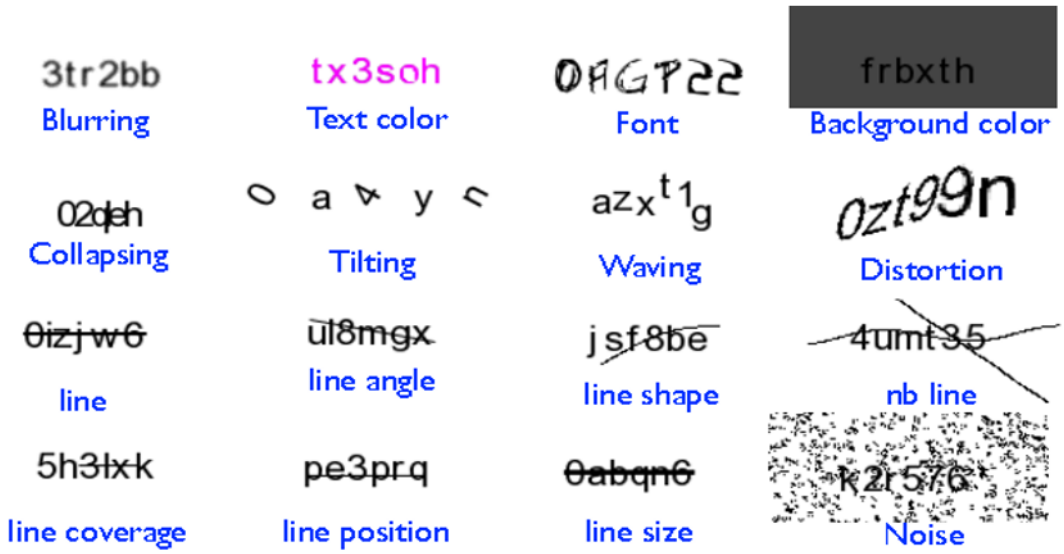


Figure 4.8: Color and Geometry of CAPTCHA's

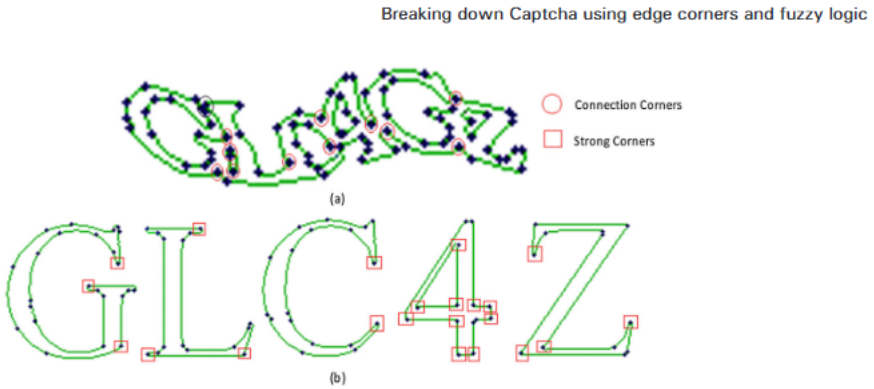
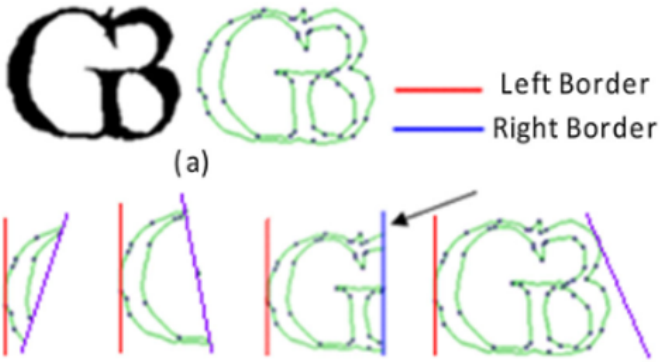


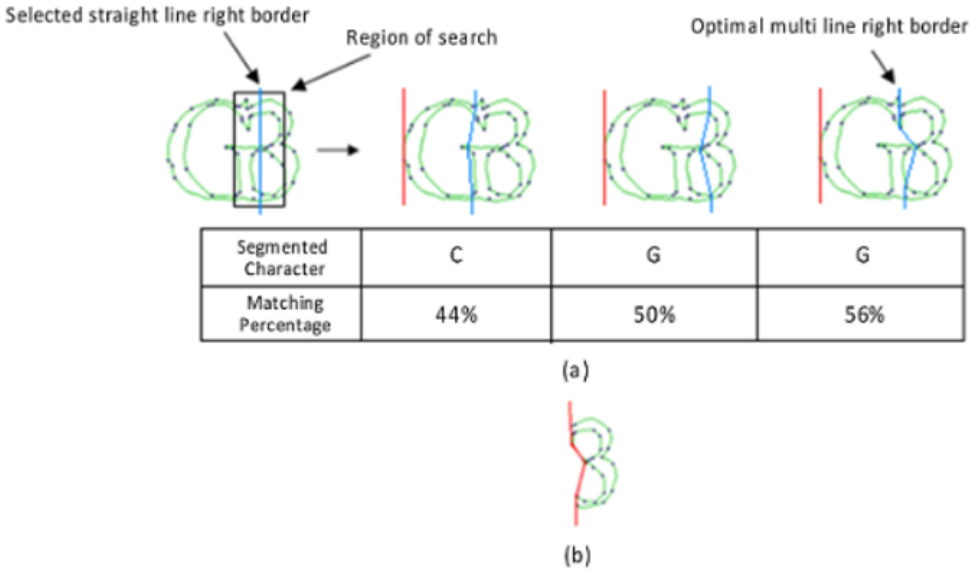
Figure 4.9: (a) Deformed characters' contours. (b) Strong corners on the original character's contours.



Segmented Character	1	C	C	G	G
Matching Percentage	18%	35%	42%	54%	41%

(b)

Figure 4.10: Find the optimum straight line right border for segmenting two linked deformed characters.



Segmented Character	C	G	G
Matching Percentage	44%	50%	56%

(a)

(b)

Figure 4.11: Optimal segmentation using multi-line right borders.

(a) Three different segmentation trials were carried out, each with a different proportion of multi-line right boundaries. (b) The remaining section will be identified using the same method.

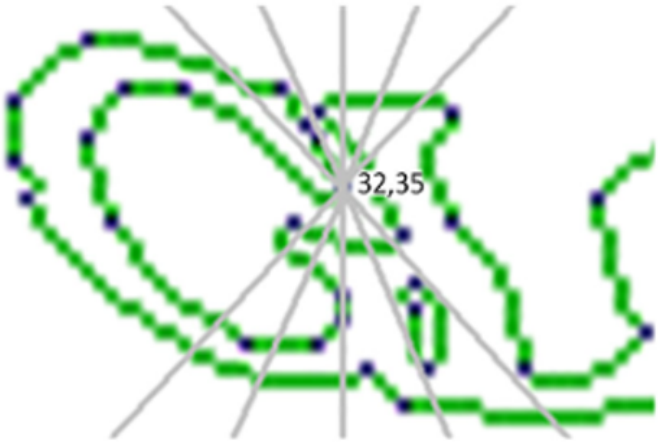


Figure 4.12: The MBS (M = 5) is drawn in grey at EC (32, 35).

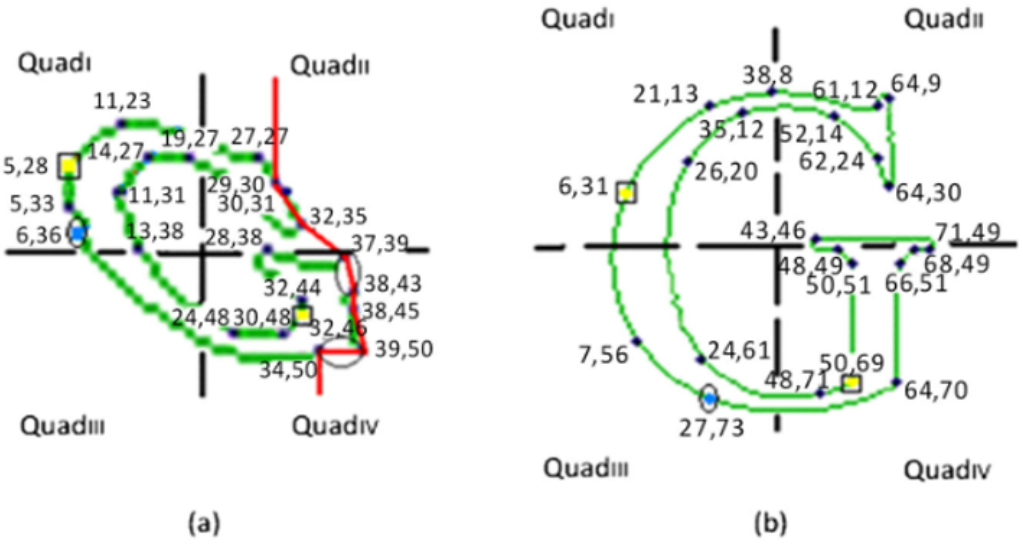


Figure 4.13: (a) Segmented part from the test image with quadrants. (b) Training edge image relative to the G character with quadrants.

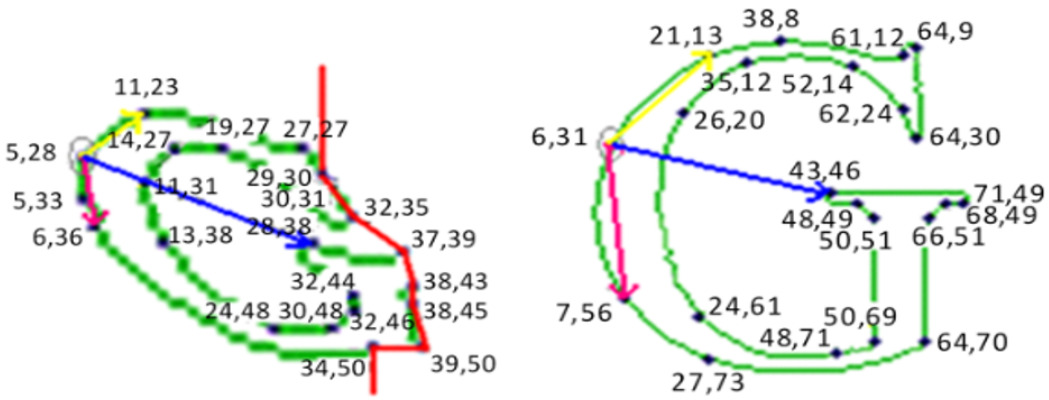


Figure 4.14: Vector angle difference of two matched ECs.

Figure 4.15: Deformation levels at different values of Sigma.

Figure 4.16: Two connected characters and Segmentation using connection corners.

The characters/digits in a distorted word picture are segmented and recognized simultaneously (Figure:4.9-Figure:4.16) in this technique. To begin, make a practice set. It includes representations of the characters/digits utilized by the targeted schemes to create the twisted word image. For the letters G, L, and C, as well as the number, the ECs of each training image are indicated as shown. A test picture containing the distorted word image is provided to the algorithm in order for it to detect the characters/digits correctly. The target character may be located by putting a left and right border on a section of the input word picture. The segmented piece is matched to the characters/digits training set using the recognition,

and a percentage match is provided. Figure (4.9-4.16) was taken from [28].

We have praised captchas with the following features throughout our work:

- Captcha with a lot of contrast in the foreground and background.
- The captcha has rotated characters.
- Captcha, which has added noisy lines.
- Captcha, which has scattered points around it.
- Captcha, which has wrapped characters.
- Captcha, which has connecting characters.

There are many text CAPTCHAs using this scheme, and most of them follow the connecting characters' principle. It contains merging textures horizontally after deporting them. This obtained image is more manageable for humans, but it will be harder for computer attacks.
We investigated several writers' papers to develop such a captcha system, and they stated the following characteristics:

- five characters are used in every challenge.
- Wrapping is used for captcha distorting.
- Symbols are placed according to their places.
- Characters are generally drawn to their immediate surroundings.
- Characters are attracted to their local surroundings in most cases.
- The backdrop is usually white, whereas the foreground is usually dark grey.
- Some captcha's entire text is cosine distorted.

Chapter 5

Implemented Models

Several different pre-trained CNN models were utilized in the research that we conducted. These models are the Convolutional RNN OCR Model based on CRNN, the Keras Model with VGG19 based on VGG19, and the ResNet-50 model. We can improve a model's performance by using models that have undergone pre-training. These CNN and RNN models each use a unique architecture to accomplish their respective tasks. In the following paragraphs, we will provide a comprehensive description of the models.

5.1 VGG19 OCR model

The VGG19 CNN is an all-inclusive model since it has pre-trained layers aware of visual shape, color, and structure. The VGG19 deep neural network was trained to perform a wide range of classification tasks on millions of images throughout its development. It is composed of the VGG-19, a 19-layer deep convolutional neural network. In addition to the more than 15 convolutional layers, five max-pooling layers, and three linked layers seen in the VGG model, this model includes one softmax layer. The architect of CNN has control over the number of convolutional layers and the size of fully coupled ones. For instance, the VGG19 design had input an RGB image with a predetermined length of (224 by 224) pixels, which led one to speculate that the matrix had the form (224 by 224 by 3). Before implementing the final method, it was necessary to pre-process each training set pixel's average RGB value. It was a prerequisite step. We took advantage of spatial padding so that the image would keep its original level of spatial resolution. A window of size two by 2 pixels was utilized for maximum pooling, and the stride was set to 2. In place of the tanh or sigmoid functions that were utilized in earlier models, the Rectified linear unit (ReLu) was introduced to express non-linearity. It was done in order to facilitate faster processing and improved classification of models. This model comprises three fully connected layers of 4096 pixels each, followed by a layer with 1000 channels for a 1000-way category. This model was developed in response to the (ILSVRC). The VGG19 is an excellent model, but it does have a few drawbacks, such as the fact that it calls for a significant amount of practice time each day. In (Figure:5.1) a VGG19 Model architecture have been given and figure 5.1 was taken from [29].

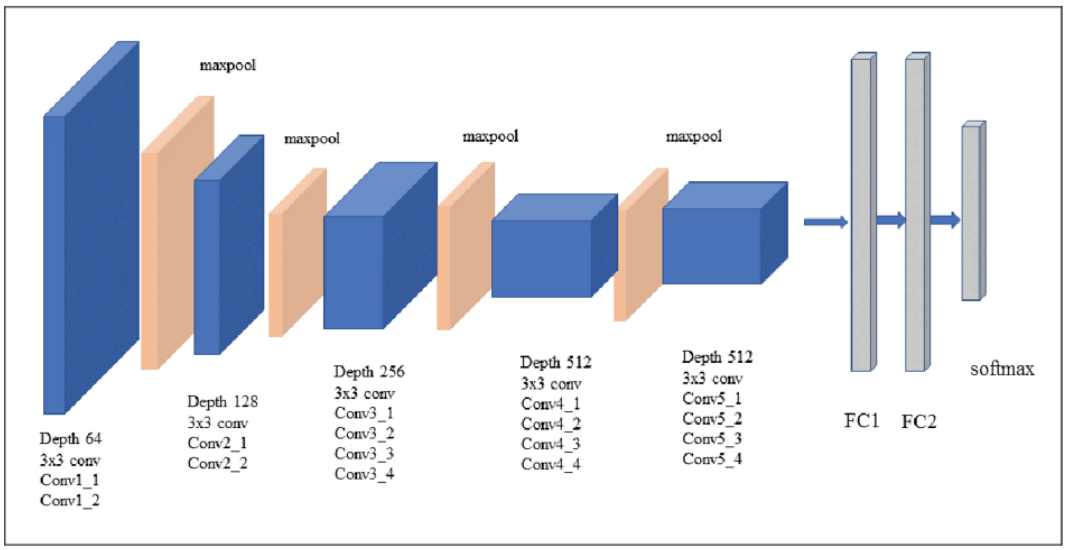


Figure 5.1: VGG19 Model architecture

5.1.1 Keras API

Keras is a Python-based deep learning API that operates on Tensor Flow. It was designed to allow for fast experimentation. When conducting research, moving swiftly from notion to conclusion is crucial.

Keras's fundamental data structures consist of 2 things layers and models. The most basic type of model there is the sequential model, which is a linear stack of layers. It would be best to utilize the Keras functional API for more advanced designs, which lets you create arbitrary layer graphs or develop models from scratch using subclasses.

5.2 Convolutional RNN OCR Model

The procedures of feature extraction, sequence modeling, and transcription are all included within a single neural network framework. It is not necessary to use a character segmentation model to accomplish the goals of this strategy. The system, which employs convolutional neural networks, takes an image as input and determines the image's attributes (text detected region). The sequence in which the labels appear will be determined by a sophisticated neural network based on the relationships between the characters. The RNN will create a per-frame output in the transcription layer to make producing a label sequence easier. It will be carried out to assist depending on personal taste. Transcription can be performed with or without the assistance of a lexicon. The method will be based on a dictionary, and the goal will be to predict the label sequence that is most likely to occur.

The convolutional layer of a CRNN model is generated from a normal CNN model's convolutional and max-pooling layers. CRNN stands for convolutional recurrent neural network (fully-connected layers are removed). From a picture input, this component may extract many feature representations. An image's height must be adjusted to the same standard before it may be shared on the internet. The recur-

rent layers are given a series of function vectors constituted of the function maps, which can be fed the output of the convolutional layer component. To be more specific, every function vector in a function series is built column via way of the column on function maps from left to proper. As a result, the i-th function vector displays the union of every map's i-th columns. In all of our options, the width of every column is ready to at least one pixel via way of means of default. Because they characteristic most effective at the regions at once across the input, convolution, max-pooling, and elementwise activation layers are translation-invariant. As a result, every function map column corresponds to a square phase of the unique image (additionally referred to as the receptive field) ordered inside the identical left-to-proper order because of the function map columns.

Deep convolutional features are popular for usage in a wide range of visual identification applications due to their durability, depth, and trainability. A reliable sequence-related object as scene text and other comparable elements has previously been learned using CNN. On the other hand, these algorithms often begin by building a holistic representation of the entire picture with CNN and then gathering in-depth local features to identify each component of a sequence-like item. CNN is not appropriate for use with sequence-like objects since the length of the things might vary significantly. The network requires that its input pictures be scaled to a constant size to fulfill its predetermined input dimension. As a result, sequence-like objects are incompatible with CNN. In-depth features are encoded as sequential representations in CRNN such that changes in the length of sequence-like items do not affect the network. It increases the network's stability.

The Functional API was used to create an OCR model. It also shows how to create a new layer and utilize it as an "Endpoint layer" for incorporating CTC loss and mixing CNN and RNN.

Specific characteristics of OCR tasks:

Text density: refers to the thickness of text on a printed or written page. Text is limited to a minimum when a photograph of a street with a single street sign is given.

Text structure: Text on a screen is usually ordered in perfect rows, but it might be thrown around in numerous rotations in the wild.

Typefaces: Printed fonts are more straightforward than noisy handwritten ones because they are more organized.

Character type: Text can be written in various languages, some of which are highly distinct from one another. Furthermore, text structure may differ from that of numbers, for example, home numbers.

Artifacts: Images taken outside are significantly noisier than those taken indoors.

Location: Text may be cropped or centered in some assignments, while text may be placed in others at random.

Conventional machine learning methods are quick to build but slow to run, and deep learning algorithms easily exceed accuracy and inference speed.

Conventional OCR approaches go through a set of pre-steps, including document cleaning and noise reduction. Following that, the document is binary to aid the contour detection of lines and columns.

5.3 ResNet-50

We decided to give it a go by utilizing Fast.ai's PyTorch-based framework. Here's a high-level overview of the two techniques we employed. We are using Class Activation Maps to solve for one character at a time. Using complete one-hot encoding, we solved the entire captcha in one shot. The attributes are present in the dataset. Each captcha is made up of 5 characters. A character may appear more than once in a single captcha. The filename of each image serves as its label. 'eig7e.png' for the preceding example. It allows label extraction while training. Firstly, each character approaches one at a time. This strategy trains the model separately for each character position and outputs a different model for each part.

Then, we apply these models to an image to solve for each captcha character. We are trying to figure out the first character of the captcha. It is an example of a classification task. The picture of the captcha is utilized as the input, and what is produced as the output is a single label that corresponds to the initial character. Now that we have the model, we will train it as a typical picture classification task. We will first construct our learner object using a particular architecture (ResNet-50 in this example) to achieve this goal. Then we will begin training using the "fit one cycle" function as is customary. Fit one cycle uses high cyclical learning rates so that models can be trained significantly more quickly and with more precision. Because of its superior performance in terms of speed and accuracy, the fit one cycle strategy is recommended over the fit method when training Deep Learning models using Fastai. It is because of the fit one cycle methodology. We initially trained for only 25 epochs because we wanted to determine which pair of characters presented the model with tremendous difficulty. We now train for seventy epochs using the initial character, reaching a very high level of accuracy in the process.

In the same fashion, we practice for every captcha spot (from 1 to 5). To train all of the places, we will repeat the code from before: As a direct consequence of this, we successfully deciphered the captcha. Now consider an alternative tactic in which, rather than cracking the captcha one character at a time, we interpret the whole thing at once using a one-hot encoding technique. The entirety of the captcha in a single attempt. The fact that a captcha is made up of seven different parts is not exploited by this method. The trained model might "hard-learn" the training images, but when it is tested on new data, it won't be able to make accurate predictions.

Consequently, it would help if you thought about utilizing a different strategy. We know that our data collection includes both letters and numbers ranging from A to Z and (0-9), a total of 36. Captcha can be modeled using a vector with a length

of 26 and a vector length of 10, with each member of the vector standing in for a different alphabetic or numeric letter. The integer located at each index of the vector represents the position within the captcha that is associated with the relevant character. Therefore, 1 will represent the first location, 2 will denote the second, etc. The value is considered zero if the character is not included in the captcha. Using this encoding, the captcha value "d08bl" has been found. There is a problem as every character has the potential to make multiple appearances within the same game. Using this approach would not be adequate to describe this. By utilizing comprehensive one-hot encoding, we will be able to resolve this issue. Encoding was done in one go-around only during the character-by-character classification process—the character located at the point I was given the representation of a vector with 36 lengths. A 36 by 5 matrix is produced due to encoding the complete captcha. Each column in the matrix represents a different one-hot encoded character found in a particular location. Flattening this encoding matrix results in producing a one-dimensional vector with a length equal to 36 times 5 or 180. Following the implementation of weight decay and various additional regularization procedures, this model learns remarkably well. After seventy iterations, the validation set reaches the level of precision that is required. In (Figure:5.2) a simple Resnet50 model architecture have been given and figure 5.2 was taken from [30].

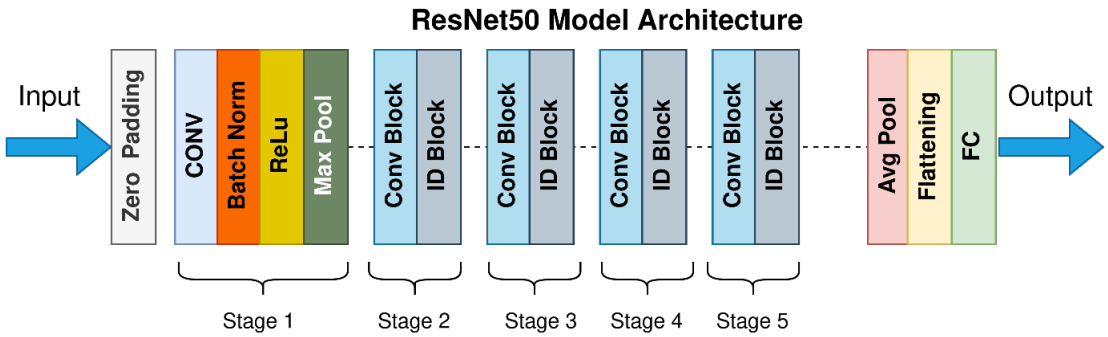


Figure 5.2: Resnet 50 model architecture

Chapter 6

Implementation and Results

The implementation of the models for Semantic Text Extraction is described in this section. Calculating the loss and accuracy, we ran 100 epochs for our proposed model Convolutional RNN OCR model and Resnet50 and 50 epochs. Pictures were resized into 200 * 50 pixels. The whole process was performed on Intel core i3 6100 3.4 GHz CPU, 12 Gigabytes of ram, AMD Radeon R7 360 2 Gigabyte GPU, and 220 gigabytes of Transcend SSD. While running the test, we use multiple datasets such as 1000 captchas,2500 captchas, 6000 captchas,10000 captchas, and 30000 captchas. We saw different types of loss and accuracy during the test on the models. While training our models, we saw different results on different types of captcha datasets. One is suitable for low quantity datasets, and another is for large quantity datasets.

6.1 Convolutional RNN OCR Model

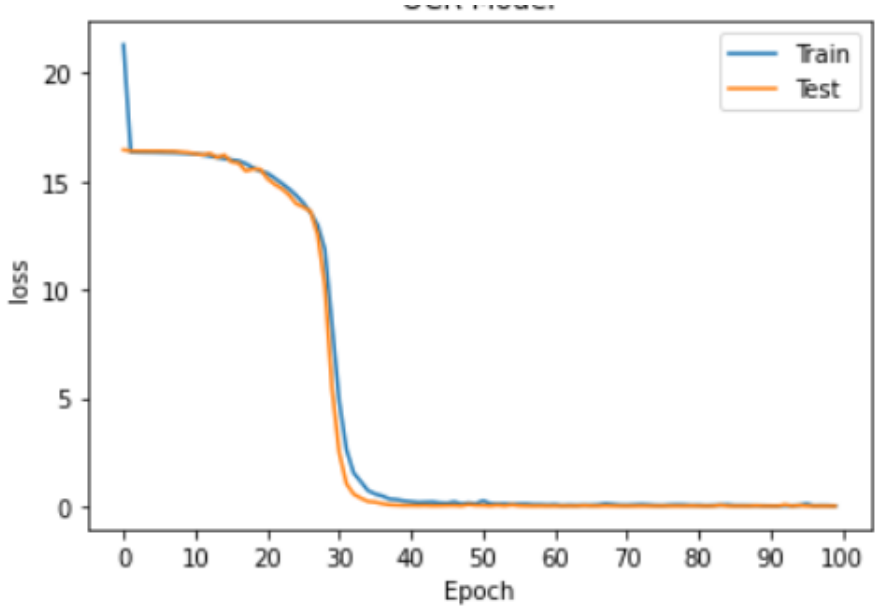


Figure 6.1: Convolutional RNN OCR model train and test lost

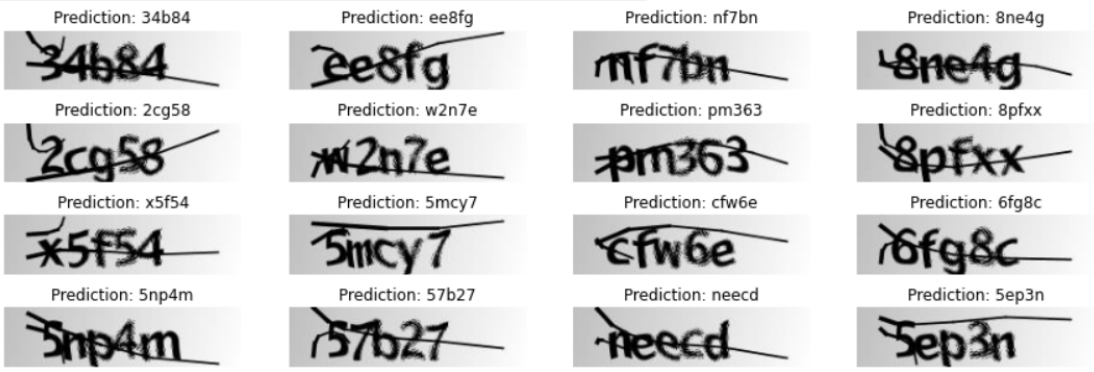


Figure 6.2: Captcha predication of Convolutional RNN OCR model

In (Figure:6.1) we can see train and test lost for Convolutional RNN OCR model and (Figure:6.2), we can see the captcha prediction for Convolutional RNN OCR model. We have run 0 to 100 epochs. On the graph, we can see that the loss gradually decreases while the epoch increases. Initially, on the 0th epoch, the loss function was high, and after completing the 100th epoch, it reached almost point zero. After 30 epochs, our model started to learn from our data, and soon it began to go on a straight line. Our training and testing lines almost touched together almost to the same point, which is a good sign. Which means it is predicting the accurate value.

6.2 VGG19 OCR Model

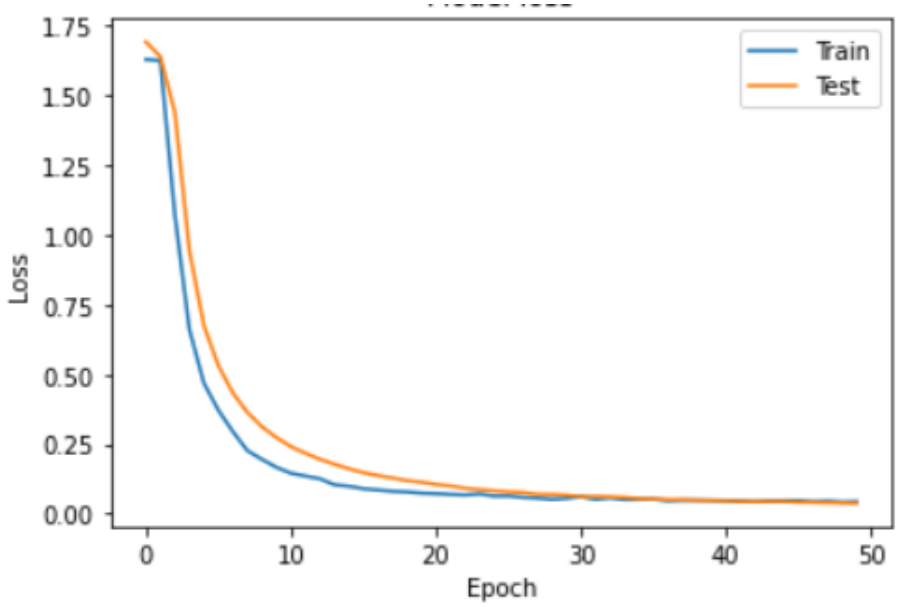


Figure 6.3: VGG19 OCR model train and test lost

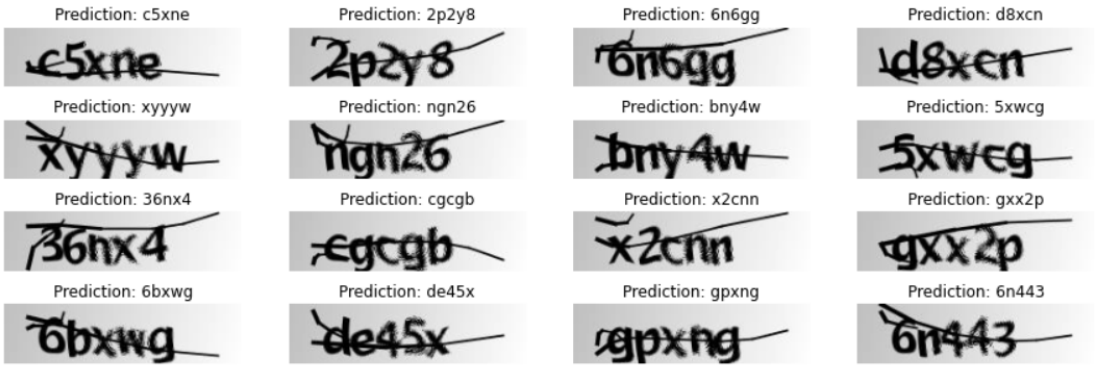


Figure 6.4: Captcha predication of VGG19 OCR model

In (Figure:6.3) we can see train and test lost for VGG19 OCR model and (Figure:6.4), we can see the captcha prediction for VGG19 OCR model. We have run 0 to 50 epochs because, for VGG19 when we increase the data the model breaks down on the middle point because we don't have the computational power to run it that is why we run the model for less data and less epoch. On the graph, we can see that the loss is gradually decreasing while the epoch is increasing. Initially, on the 0th epoch, the loss function was high, and after completing the 50th epoch, it reached almost point zero. After 20 epochs, our model started to learn from our data, and soon it began to go on a straight line. Our training and testing lines almost touched together almost the same point here, which is a good sign. Which means it is predicting the accurate value.

6.3 ResNet-50

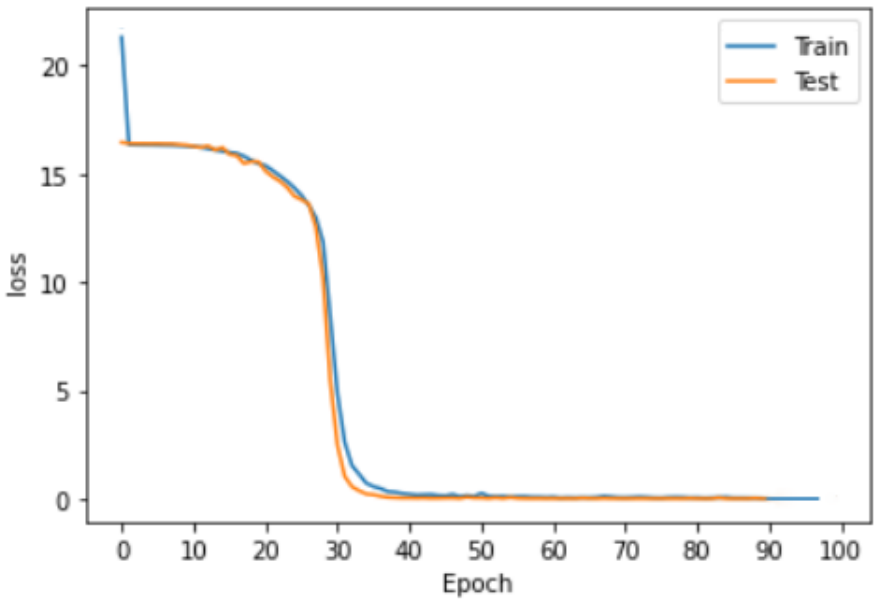


Figure 6.5: Resnet50 model train and test lost

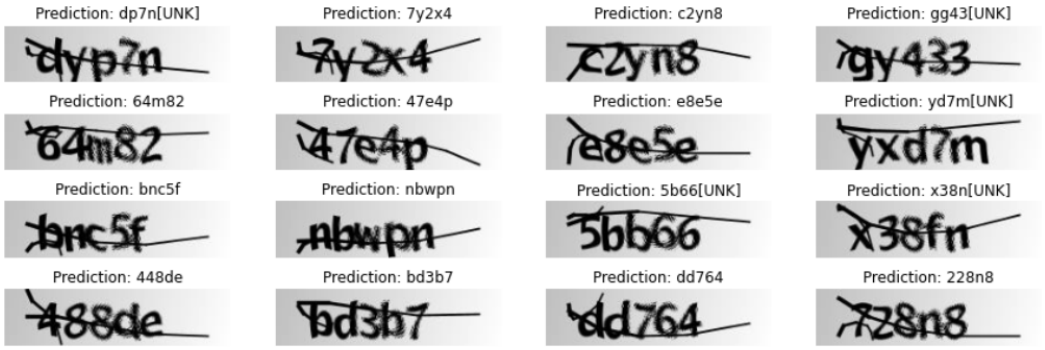


Figure 6.6: Captcha predication of Resnet50 model

In (Figure:6.5) we can see train and test lost for Resnet50 and (Figure:6.6) we can see the captcha prediction for Resnet50 model. We have run 0 to 100 epochs. On the chart, we can see that the loss gradually decreases while the epoch increases. Initially, on the 0th epoch, the loss function was high, and after completing the 100th epoch, it reached almost point zero. After 40 epochs, our model started to learn about our data, and soon it began to learn thoroughly and go on a straight line same as Convolutional RNN OCR model. Our training and testing lines almost touched together almost to the same point, which is a good sign. Which means it is predicting the accurate letters.

6.4 Comparison

6.4.1 Convolutional RNN based OCR model

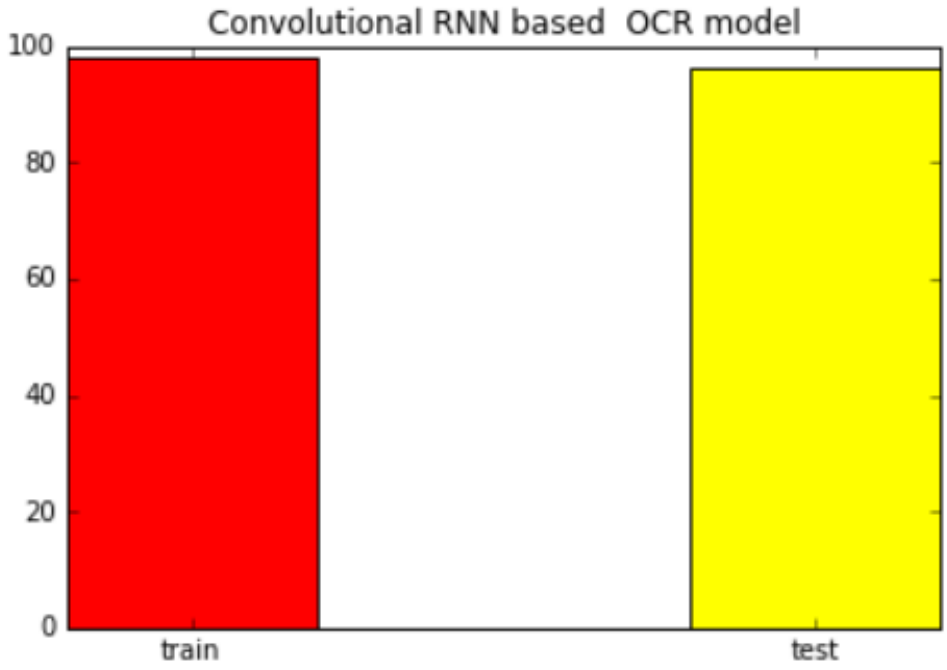


Figure 6.7: Train and test accuracy of Convolutional RNN based OCR model

Convolutional RNN based OCR model			
Train Loss	Train Accuracy	Test Loss	Test Accuracy
0.09956	98.03	0.004926	96.56

Table 6.1: Convolutional RNN based OCR model accuracy and loss

In (Figure:6.7) Train and test accuracy bar graph is shown for Convolutional RNN based OCR model.

6.4.2 VGG19 OCR model

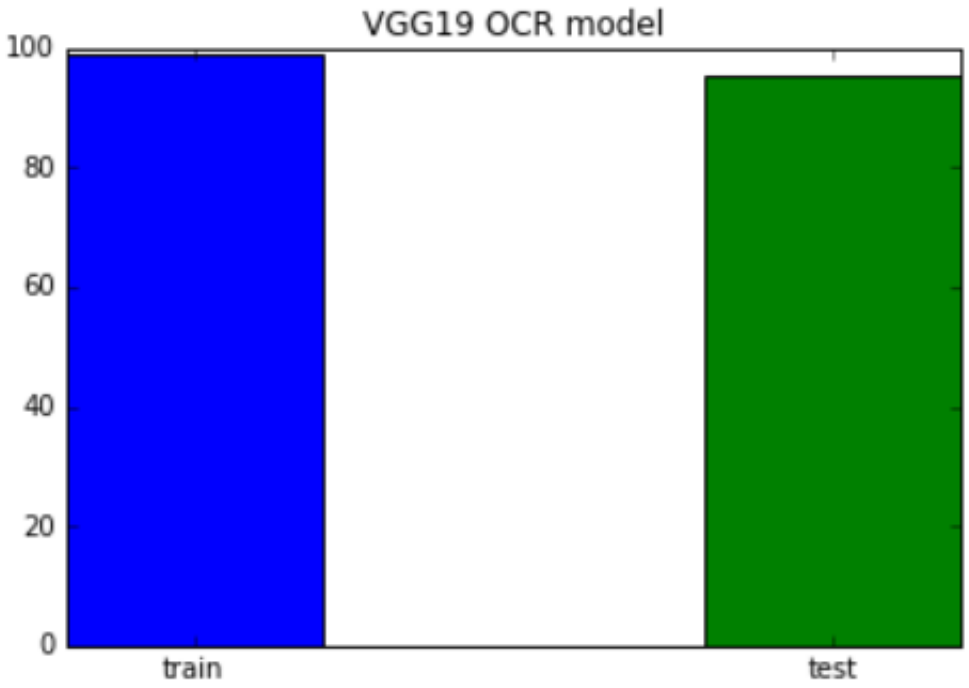


Figure 6.8: Train and test accuracy of VGG19 OCR model

In (Figure:6.8) Train and test accuracy bar graph is shown for VGG19 OCR model

VGG19 OCR model			
Train Loss	Train Accuracy	Test Loss	Test Accuracy
0.000876	99.90	0.017765	95.19

Table 6.2: VGG19 OCR model accuracy and loss

6.4.3 Resnet50

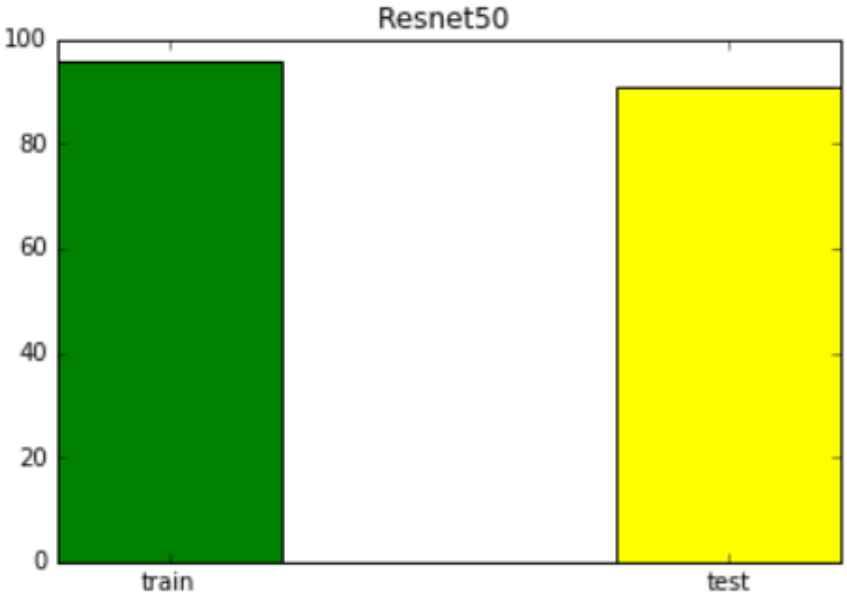


Figure 6.9: Train and test accuracy of Rsesnet50 model

In (Figure:6.9) Train and test accuracy bar graph is shown for Rsesnet50 model

Resnet50			
Train Loss	Train Accuracy	Test Loss	Test Accuracy
0.001976	96.90	0.018765	90.78

Table 6.3: Resnet50 accuracy and loss

6.4.4 Comparisons between the 3 models

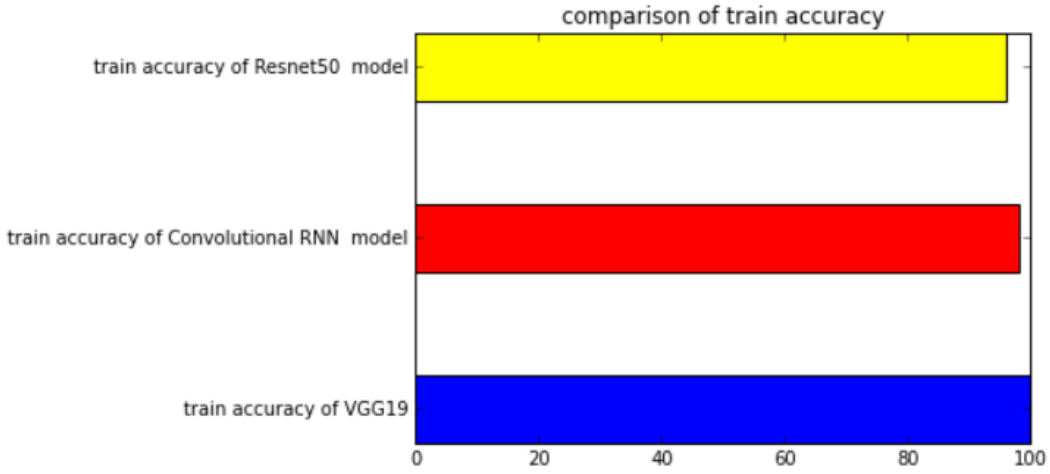


Figure 6.10: Comparison of train accuracy

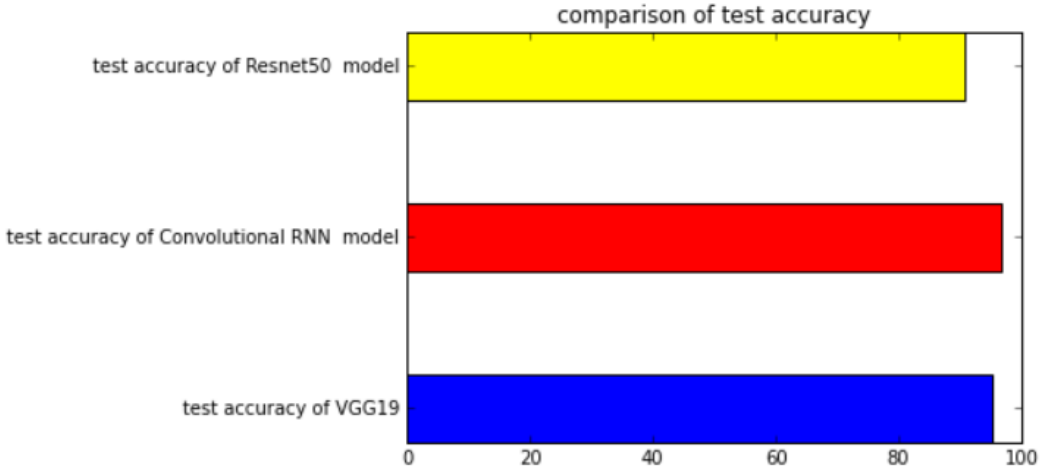


Figure 6.11: Comparison of test accuracy

Models	Train Accuracy	Test Accuracy
Convolutional RNN based OCR model	**98.03**	**96.56**
VGG19 OCR model	**99.90**	**95.19**
Resnet50	**95.90**	**90.78**

Table 6.4: Comparison of Train and test accuracy of all the models

The above tables and figures(Figure:6.10), (Figure:6.11) show that the Convolutional RNN-based OCR model gives more accurate testing captchas better than VGG19 and Resnet50, while the VGG19 based model gives more precise training captchas(Table:6.4). However, the VGG19-based model provides greater accuracy on smaller datasets, and the Convolutional RNN-based model delivers better accuracy on many datasets. For Resnet50, both train and test accuracy are less than CNN VGG19 and our Convolutional RNN-based OCR model. In the model testing, we have used 30000 datasets randomly based on alphanumeric captchas.Convolutional RNN-based model gives better testing accuracy than the 2 models because, it is a hybrid model and it performs better than the 2 models in terms of timing ,extracting captchas and bigger dataset.

Chapter 7

Challenges

7.1 Computational Power

The available computing ability to handle this data set with 30000 photos was extremely limited due to a shortage of electricity, our computing power, and a distance from the on-campus computer facility. The graphics processing power available was insufficient to perform specific algorithms like VGG19 and RNN. The VGG19 model crashed during execution so we have to run it only 50 epochs after that training phase was completed. As a result, the prediction run failed. Then, utilizing "Google Colab," we used Google's offered computational server to acquire the desired findings.

7.2 Excessive Training Time

It's a side consequence of the problem with processing power. The training and testing times for each model were quite long due to the lack of computer capacity. Each epoch might take an hour or more to finish in some situations. The team's research work was hampered due to this slow training time. By running additional CNN models for future comparisons and analysis, we expect to fix these two concerns.

Chapter 8

Conclusion and Future plan

8.1 Future Work

The above three models are significantly better results for five-character alphanumeric captchas. It also has some limitations. Above them, hybrid model which is CRNN model performs better. We have worked on a hybrid model to detect five character English language alphanumeric captchas. For that model, we choose more data, such as more than thirty thousand data from the Python captcha library and Kaggle to extract texts from the captchas. However, the model will pre-process the image, and then, it will extract the text and numbers. The valuable part is that it will work for alphanumeric captchas from four to five digits. Machine learning algorithms can defeat CAPTCHAs; even Google's captcha can be defeated. Although many simple CAPTCHAs pose no significant security risk, those with a complex structure that humans find difficult to read are ineffectual and wasteful. CAPTCHAs that need alphabetic or numeric input are straightforward to avoid. Alphanumeric CAPTCHAs are relatively straightforward to recognize because they are made up of alphabet and number combinations. The way to break images is to use image processing machine learning algorithms. CAPTCHA's weakness can be exploited in a limited manner using the CRNN model. We want to make it more robust and rugged enough so that will work on more datasets in less time .It will help to make captcha more hard that the amount of time necessary for a machine to break it is significant, forcing the device to fail when time is limited. It will be helpful not only for the captcha detection but also the help the result, and the security can be made strong. It will also be a great way to stop security breaking and develop a new type of captchas to protect the security.

8.2 Conclusion

Semantic text extraction can be a vital function that will help create a prediction of the entire input. With the help of CNN and RNN, it will be relatively easier to extract the text from the image-based text CAPTCHA. Extracting captcha will also help to strengthen the security also. We have focused on different types of alphanumeric-based CAPTCHA. We also have shown the workflow of how we will extract the text. However, none of the three models can handle more than five words. It also could not offer more accuracy when it faces many data. We are working on CRNN model to resolve this issue, which can handle up to seven-digit alphanumeric

captchas. In this research, we showed the previous study, and we also trained some models for different types of captchas and showed the result and accuracy, which is satisfactory for a low number of data. But with the help of CRNN model, it can handle a large number of data and can be used for checking the vulnerability to make the captcha stronger. In this manner, CRNN model will help develop more advanced alphanumeric captchas

Bibliography

[1] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern recognition*, vol. 37, no. 5, pp. 977–997, 2004.

[2] A. Abdussalam, S. Sun, M. Fu, H. Sun, and I. Khan, "License plate segmentation method using deep learning techniques," in *International Conference On Signal And Information Processing, Networking And Computers*, pp. 58–65, Springer, 2018.

[3] A. Abdussalam, S. Sun, M. Fu, Y. Ullah, and S. Ali, "Robust model for chinese license plate character recognition using deep learning techniques," in *International Conference in Communications, Signal Processing, and Systems*, pp. 121–127, Springer, 2018.

[4] L. Zhang, Y. Xie, X. Luan, and J. He, "Captcha automatic segmentation and recognition based on improved vertical projection," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 1167–1172, IEEE, 2017.

[5] C. J. Chen, Y. W. Wang, and W. P. Fang, "A study on captcha recognition," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 395–398, IEEE, 2014.

[6] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 3, pp. 377–391, 2008.

[7] Q. Wang, "License plate recognition via convolutional neural networks," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 926–929, IEEE, 2017.

[8] K. Chellapilla and P. Simard, "Using machine learning to break visual human interaction proofs (hips)," *Advances in neural information processing systems*, vol. 17, 2004.

[9] N. Saleem, H. Muazzam, H. Tahir, and U. Farooq, "Automatic license plate recognition using extracted features," in *2016 4th international symposium on computational and business intelligence (ISCBI)*, pp. 221–225, IEEE, 2016.

[10] A. Sasi, S. Sharma, and A. N. Cheeran, "Automatic car number plate recognition," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–6, IEEE, 2017.

[11] R. Hussain, H. Gao, R. A. Shaikh, and S. P. Soomro, "Recognition based segmentation of connected characters in text based captchas," in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, pp. 673–676, IEEE, 2016.

[12] P. Sakkatos, W. Theerayut, V. Nuttapol, and P. Surapong, "Analysis of text-based captcha images using template matching correlation technique," in *The 4th Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering (JICTEE)*, pp. 1–5, IEEE, 2014.

[13] C. Wu, L. C. On, C. H. Weng, T. S. Kuan, and K. Ng, "A macao license plate recognition system," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 7, pp. 4506–4510, IEEE, 2005.

[14] R. A. Baten, Z. Omair, and U. Sikder, "Bangla license plate reader for metropolitan cities of bangladesh using template matching," in *8th International Conference on Electrical and Computer Engineering*, pp. 776–779, IEEE, 2014.

[15] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, "Selective learning confusion class for text-based captcha recognition," *IEEE Access*, vol. 7, pp. 22246–22259, 2019.

[16] K. Chellapilla and P. Simard, "Using machine learning to break visual human interaction proofs (hips)," *Advances in neural information processing systems*, vol. 17, 2004.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[18] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, pp. 1058–1066, PMLR, 2013.

[19] D. Masko and P. Hensman, "The impact of imbalanced training data for convolutional neural networks," 2015.

[20] A. G. Howard, "Some improvements on deep convolutional neural network based image classification," *arXiv preprint arXiv:1312.5402*, 2013.

[21] Z. Wang, X. Wang, and G. Wang, "Learning fine-grained features via a cnn tree for large-scale classification," *Neurocomputing*, vol. 275, pp. 1231–1240, 2018.

[22] A. Thobhani, M. Gao, A. Hawbani, S. T. M. Ali, and A. Abdussalam, "Captcha recognition using deep learning with attached binary images," *Electronics*, vol. 9, no. 9, p. 1522, 2020.

[23] https://www.researchgate.net/publication/280330424_An_End-to-. Accessed: 2022-5-21.

[24] H. Yang, "Captcha recognition using convolutional neural networks with low structural complexity," in *Journal of Physics: Conference Series*, vol. 1693, p. 012040, IOP Publishing, 2020.

[25] J. Wang, J. Qin, X. Xiang, Y. Tan, and N. Pan, "Captcha recognition based on deep convolutional neural network," *Math. Biosci. Eng*, vol. 16, no. 5, pp. 5851–5861, 2019.

[26] S. Sarwari, "Solving CAPTCHAs using PyTorch(without using OCR)." https://medium.com/swlh/solving-captchas-using-resnet-50-without-using-ocr-3bdfbd0004a4, July 2020. Accessed: 2022-5-21.

[27] P. Samadnejad, "CAPTCHA dataset."

[28] R. A. Nachar, E. Inaty, P. J. Bonnin, and Y. Alayli, "Breaking down captcha using edge corners and fuzzy logic segmentation/recognition technique," *Security and Communication Networks*, vol. 8, no. 18, pp. 3995–4012, 2015.

[29] R. Hewage, "Extract features, visualize filters and feature maps in vgg16 and vgg19 cnn models." https://towardsdatascience.com/extract-features-visualize-filters-and-feature-maps-in-vgg16-and-vgg19-cnn-models-d2da6333edd0, May 2020.

[30] "File:ResNet50.Png." https://commons.wikimedia.org/wiki/File:ResNet50.png. Accessed: 2022-5-22.