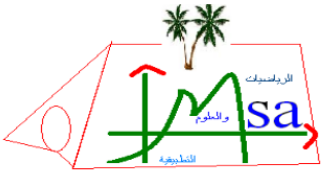
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University of Ghardaia
Faculty of Science and Technology



Laboratory of Mathematics and Applied Sciences
Department of Mathematics and Computer

Master Thesis

Presented to obtain the Master diploma in Computer Science
Specialty: Intelligent Systems for Knowledge Extraction

Theme

A Deep Learning Model for Text-based CAPTCHA Breaking

Presented by:

Khadidja METTAS

Maissa Fatna RAI

Jury members

Mr	Chaker Abdelaziz KERRACHE	MCB	Univ.Ghardaia	President
Mr	Slimane BELLAOUAR	MCB	Univ.Ghardaia	Examiner
Mr	Youcef MAHDJOUB	MAA	Univ.Ghardaia	Examiner
Mr	Slimane OULADNAOUI	MCB	Univ.Ghardaia	Supervisor

University Year 2019/2020

Abstract

Machine learning has developed widely in recent years, where intelligent systems and computer programs become parallel to the capabilities of the human brain. This is due to the development of machine learning algorithms (especially deep learning), the improvement of computer performance, and the availability of huge volumes of data. However, there are negative uses for this development, including the ability to break and pass the CAPTCHA test.
CAPTCHA for (Completely Automated Public Turing test to tell Computers and Humans Apart), is one of the verification systems used by a large number of websites and web services to protect them from being hacked by computer programs (bots), the most typically used form of CAPTCHAs is text-based CAPTCHA.
Our aim is to study the mechanisms for breaking this type of CAPTCHA and the challenges arising from the developments of machine learning algorithms facing its security. We conduct an experimental study using a model based on an End-to-End CRNN-CTC network used for handwriting and text recognition to break a text-based CAPTCHA generated by python. The obtained results show that our model succeeds to break text-based CAPTCHA with a good result compared to state-of-the-art methods.

key words:
Text-based CAPTCHA, CAPTCHA breaking, deep learning, End-to-End CRNN-CTC.

ملخص

عرف التعلم الآلي في السنوات الأخيرة تطورا كبيرا بحيث أصبحت الأنظمة الذكية وبرامج الحاسوب توازي قدرات العقل البشري و هذا راجع إلى التطور المستمر في خوارزميات التعلم الآلي (وبشكل خاص التعلم العميق)، قدرة الحواسيب و كمية البيانات الكبيرة المتوفرة. من جهة أخرى نجد لهذا التطور إستعمالات سلبية من بينها قدرة هذه البرامج على كسر وإجتياز إختبار كلمة التحقق (كابتشا).
كابتشا، إختبار تورنغ العام و الاوتوماتيكي بشكل كامل للتمييز بين الحاسب و الإنسان، أحد أنظمة التحقق الهامة المستخدمة من قبل عدد كبير من المواقع و خدمات الويب قصد حمايتها من الإختراق من قبل البرامج الضارة و من أهم انواعها الاكثر استعمالا هي الكابتشا النصية.
هدفنا هو دراسة آليات كسر هذا النوع من الكابتشا و التحديات الناجمة عن تطورات خوارزميات التعلم الآلي التي تواجه أمنها. بالإضافة الى دراسة تطبيقية بإستعمال شبكة End-to-End CRNN-CTC المستخدمة في التعرف على الكتابة اليدوية و النص لكسر كابتشا نصية تم انشائها بواسطة بايثون. تظهر النتائج المتحصل عليها أن نموذجنا نجح في كسر الكابتشا النصية بنتيجة جيدة مقارنة مع الأنظمة الاخرى.

الكلمات المفتاحية:
الكابتشا النصية، كسر الكابتشا، التعلم العميق، End-to-End CRNN-CTC.

Résumé

L'apprentissage automatique s'est largement développé ces dernières années, où les systèmes intelligents et les programmes informatiques deviennent similaires aux capacités du cerveau humain. Cela est dû au développement d'algorithmes d'apprentissage automatique (en particulier d'apprentissage en profondeur), ainsi qu'à la capacité des ordinateurs et aux mégadonnées disponibles. Cependant, il y a des utilisations négatives pour ce développement, y compris la capacité de casser et de réussir le test CAPTCHA.
CAPTCHA pour (Completely Automated Public Turing test to tell Computers and Humans Apart), est l'un des systèmes de vérification utilisés par un grand nombre de sites Web et de services Web pour les protéger contre le piratage par des programmes informatiques (bots), le CAPTCHA textuel est géneralement le type le plus utilisé.
Notre objectif est d'étudier les mécanismes de rupture ce type de CAPTCHA et les défis liés aux développements d'algorithmes d'apprentissage automatique face à sa sécurité. Nous menons une étude expérimentale utilisant un modèle basé sur un réseau End-to-End CRNN-CTC utilisé pour la reconnaissance de l'écriture manuscrite et du texte pour rupture un CAPTCHA textuel créé avec Python. Les résultats obtenus montrent que notre modèle est capable de rupture le CAPTCHA textuel avec un bon résultat par rapport à les méthodes d'état de l'art.

Mot clés:
CAPTCHA textuel, rupture de CAPTCHA, apprentissage profond, End-to-End CRNN-CTC.

Dedicated

I dedicate this modest work to

My dear parents for all their sacrifices, unwavering love, support and endless patience, i ask God to provide you with happiness and a long life.

My dear sisters, Fariha and Marwa, for their constant encouragement and moral support, which always gave me hope and believed in my abilities.

My dear brothers and all my family for their encouragement throughout my academic career.

A special dedication to my dear partner Maissa fatna for her patience and hard work despite difficult circumstances.

To all my colleagues and friends especially MjAch group for the unforgettable moments of pure feelings and happiness. May allah bless you with success.

To all my teachers since my first years of studies. To all those who feel dear to me and whom i have failed to mention.

sincerely
Khadidja

Dedicated

I dedicated this work

To my late father, who unfortunately didn't stay in this world long enough to see this thesis completed. May Allah have mercy on him and make his abode in his spacious gardens.

Also to all beloved people who are no longer of this world, but they continue to mean so much to me. May Allah grant them Jannah Firdaws.

To my mother for all her prayers and benedictions and help.
Also my maternal grandmother, for her endless love and encouragement.

To my sister Bouchera who was there for me throughout and gave me lots of support, also my sister Sabrina, and my brothers Abdelmalek and Abdelkader.

To my dear friend and partner Khadidja for her hard work and lots of support. I wish all the best for her.

To all of my colleagues and my friends especially MjAch group, for the unforgettable memories. I hope that they fulfill all that they wish.

To all teachers who believed in my abilities and encouraged me in my education career. I am extremely grateful to them.

To all who encouragement throughout my academic career.

with all love

Maissa Fatna

Acknowledgement

First thanks to God Almighty, for giving us the strength and courage to complete this thesis.
Then our parents for all of their efforts.

We convey our sincere thanks to our respected professor M.OULADNACUI Slimane for support and guidance by providing advice and correction.

We are also pleased to thank everyone who advised, guided, or contributed with us in preparing this research.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Context and Motivations

"I'm not a robot" We can confirm that you encountered this sentence one day while using internet services and maybe more than once. Since the advent of the websites and their services, we must prove that we are not a machine before advantage from these services, by passing what is known as the CAPTCHA test. **CAPTCHA** (Completely Automated Public Turing test to tell Computers and Humans Apart) is a test designed to distinguish between human users and computer programs (bots) on websites in order to protect their services from harmful use by these bots. This test has many variations, such as : text-based CAPTCHA, image-based CAPTCHA, video-based CAPTCHA. And to make the CAPTCHA test that a human can pass it but the machine cannot. They relied in their design on tasks that humans excel at performing against a machine such as text understanding, character recognition, object detection, speech recognition, video classification.

From the first use of the CAPTCHA test in 1997 by the Alta Vista website to the present day, attempts are made to pass and break it, whether from computer programs (bots) or from scientists, in order to know its weaknesses and find new strengths every time in line with the rapid development of intelligent systems. The vast amount of data available (big data) and improving the capabilities of computers made deep learning one of the most advanced fields, which achieved good results in breaking the CAPTCHA test. This makes the CAPTCHA test that's the machine can't break it still a challenge.

The text-based CAPTCHA is the most used compared to other types, due to its ease of use and low cost of design, this is what made it vulnerable to attack and break, and on the other

hand, a wide field of research to make it more solid and unbreakable or prove its inefficiency in light of the developments in machine learning algorithms and deep learning.

1.2 Goals

Our goal in this work is to strength CAPTCHA techniques. By proving the weakness of the text-based CAPTCHA in front of deep learning methods. That is what made us try to break a text-based CAPTCHA using a model based on an End-to-End CRNN-CTC network used for handwriting and text recognition with dataset of images generated py CAPTCHA library in python. Each CAPTCHA image contains between four to seven characters from the upper case 26 English alphabets.

1.3 Organization

This thesis includes three chapters:

- **Chapter 2:** This chapter introduces some preliminaries. It is divided into three sections: The first introduces the basic concepts of machine learning, the second concerns neural networks, and the third involves deep learning, especially Convolutional Neural Networks and Recurrent Neural Networks.
- **Chapter 3:** This chapter overviews the CAPTCHA test and its types. It presents the text-based CAPTCHA generation process also the state of the art in text-based CAPTCHA breaking.
- **Chapter 4:** This chapter contains our experiment to break a text-based CAPTCHA using End-to-End CRNN-CTC network used for handwriting and text recognition. With dataset generated by python in which every image contains between four to seven characters.

CHAPTER 2

ARTIFICIAL INTELLIGENCE TO DEEP LEARNING

2.1 Introduction

Nowadays, Deep Learning (DL) constitutes a hot topic and is at the top of intelligence systems. With the development of the machines and abundance of data, it was developed systems that simulate the ability of the human brain. Deep learning is one of the techniques of Artificial Intelligence (AI) derived from machine learning. So it is not possible to talk about deep learning without a reminder of the principles of both machine learning and artificial intelligence.

In this chapter, we first present a brief history of the long development of artificial intelligence from its basic techniques to deep learning. We also review machine learning and its main techniques and branches. Since neural networks are at the core of deep learning, we devoted the next section to it, where we explain how they work. And present their different types. The last section is devoted to the most important DL architectures.

2.2 History

Machine learning is the most advanced field in AI. In the following points, we try to mention the most important stages in the history of the development of intelligent systems of (artificial intelligence, machine learning, and deep learning) :

- 1943: The first mathematical model of a neural network: Walter Pitts (logician) and Warren McCulloch (neuroscientist) invented the first mathematical model of the neural network through a set of mathematics and algorithms that aim to simulate human thinking processes (McCulloch and Pitts, 1943).
- 1950: The prediction of machine learning : Alan Turing (mathematician) proposed what he called a Turing test to determine if a computer can think (Machinery, 1950).
- 1952: First machine learning programs : Arthur Samuel invented the first program that can play the checker game and capable to learn by correcting its mistakes (Samuel, 1959).
- 1957: Setting the foundation for deep neural networks: Frank Rosenblatt (psychologist) proposed an electronic system that was known as the basis of deep neural networks (Rosenblatt, 1957).
- 1960: Control theory: Henry J. Kelley published "Gradient Theory of Optimal Flight Paths" which was later used to develop the basics of backpropagation model (Kelley, 1960).
- 1965-71: The first working deep learning networks: Alexey Ivakhnenko and V.G. Lapa developed what is known as (GMDH) Group Method of Data Handling, which is a set of algorithms applied to neural networks. In 1971, they designed the first deep neural network (Ivakhnenko and Ivakhnenko, 1995).
- 1979-80: An Artificial Neural Network (ANN) learns how to recognize visual patterns. Kunihiko Fukushima creates what is known as "Neocognitron," which is an artificial neural network that learned to recognize patterns. It is considered as the first convolutions neural network (Fukushima, 1989).
- 1982: The creation of the Hopfield Networks: John Hopfield invented a system that bears his name, which is the first recurrent neural network (Hopfield, 1982).
- 1985: A program learns to pronounce English words: Terry Sejnowski (neuroscientist) proposed a program able to learn how to pronounce English words (NETtalk) (Rosenblatt, 1957).
- 1986: Improvements in shape recognition and word prediction : David Rumelhart, Geoffrey Hinton, and Ronald J. Williams presented details of backpropagation process and

its importance in neural networks for many tasks such as shape recognition, and word prediction (Rumelhart et al., 1986).

- 1989: Q-learning: Christopher Watkins introduced the concept of Q-learning which is one of the most important reinforcement learning algorithms (Watkins, 1989).
- 1995: Support Vector Machines (SVM) : Cortes and Vapnik in 1993 designed SVM and presented them later in 1995 (Cortes and Vapnik, 1995).
- 1997: Long Short-Term Memory (LSTM): A recurrent neural network framework was proposed by Schmidhuber and Hochreiter (Hochreiter and Schmidhuber, 1997).
- 2000s: Modern deep learning such as Launch of ImageNet 2009 (Deng et al., 2009), AlexNet 2012: A convolutional neural network designed by Alex Krizhevsky (Krizhevsky et al., 2012).
- 2014: GAN (Generative Adversarial Networks): Ian Goodfellow and his colleagues describe the first working implementation of a generative model based on adversarial networks (Denton et al., 2015).
- 2016: Google's AlphaGo: AlphaGo developed by DeepMind [1] and it is the first computer program that defeats a professional human player in Go game.

[1] https://deepmind.com/

2.3 Machine Learning

Can Machines Think ? This question was asked by mathematician scientist Alan Turing in his paper "Computing Machinery and Intelligence" in 1950. To answer this question, Turing proposed the first definition of Artificial Intelligence (AI) which is known as the Turing test (*Imitation Game*). This test includes a machine and a human person that an examiner asks them questions with each of them is placed in a separate room. If the examiner is unable to distinguish between the human's answer and the machine's answer, then we say that the machine has passed the test. The Turing test is a strong demonstration that the machine possesses artificial intelligence, and his proposal has been widely influenced in computer sciences, cognitive sciences, and philosophical communities for more than 50 years (Turing, 2009; Machinery, 1950).

"Artificial intelligence is a set of algorithms and intelligence to try to mimic human intelligence. Machine learning is one of them, and deep learning is one of those machine learning techniques." Frank Chen (Chen, 2016). This is illustrated in the Figure 2.1.

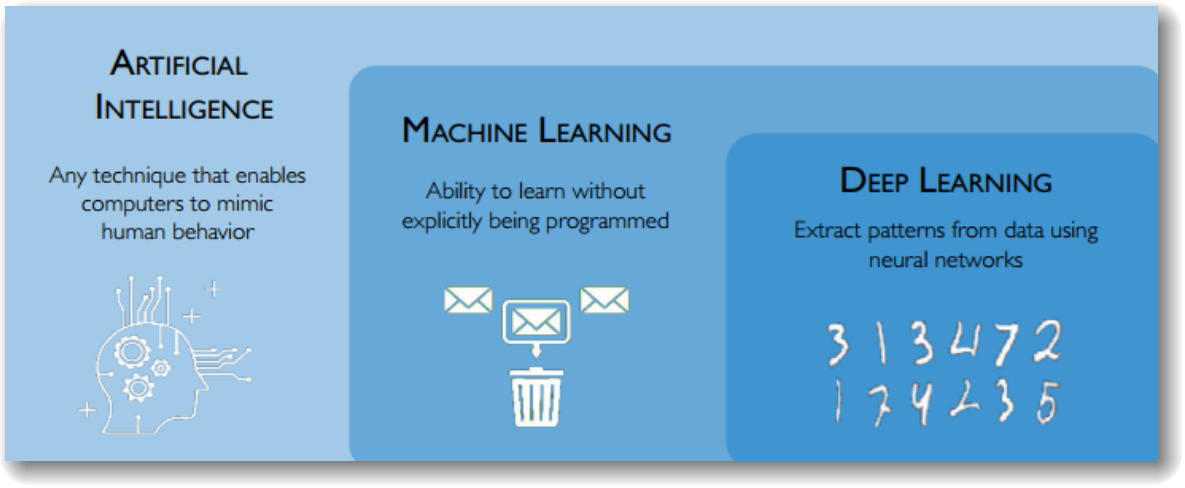


Figure 2.1: AI / ML / DL [2]

2.3.1 Definition

"Machine Learning (ML) is a field of study that gives computers the ability to learn without being explicitly programmed" Arthur Samuel (Samuel, 1959).

ML is a vast field that studies the technology of constructing a model (algorithms) that allow computers to possess the learning feature without programming the rules for each problem,

[2]Introduction to Deep Learning MIT 6.S191 Alexander Amini January 28, 2019

where these model rely on a set of data to extract the rules that make them able to learn and predict in the future.

2.3.2 Machine Learning Types

Depending on the type of problem and according to input and output data, learning can be divided into different types as shown in Figure 2.2 :

- **Supervised Learning:** Supervised learning is a learning technique that uses labeled data, so the environment has a set of corresponding inputs and outputs (x, y) and it learns from this data in order to predict for a new input X its output Y (Alom et al., 2019). Supervised learning includes two tasks :
  1. **Regression:** If the prediction is continuous values (quantitative), the output takes continuous values, for example predicting house prices. The most famous algorithm is Linear regression.
  2. **Classification:** If the prediction is discrete values or classes (qualitative), the output takes class labels, for example, the classification of email as spam or not spam. The most famous algorithm is SVM (Support Vector Machine) (Cortes and Vapnik, 1995).
- **Unsupervised Learning :** Unsupervised learning technique uses unlabeled data. In this case, the algorithm learns the internal representation or important features to discover relationships or structure within the input data (Alom et al., 2019) we present here only two parts of this class:
  1. **Clustering:** This task is based on dividing data into groups so that the data are similar within each cluster and dissimilar from the other. The most famous clustering algorithms is K-means.
  2. **Dimensionality Reduction:** The aim of this task is to reduce the complexity/size of the data by reducing their dimensions while trying to maintain important one's dimensions. PCA (Principal Component Analysis) is the most famous algorithm.
- **Reinforcement Learning:** Reinforcement learning is the training of machine learning models to make a sequence of decisions, it is guided by the environment in the form of rewards or penalties given according to the error made during learning. The most famous algorithm is Q-Learning (Watkins, 1989).

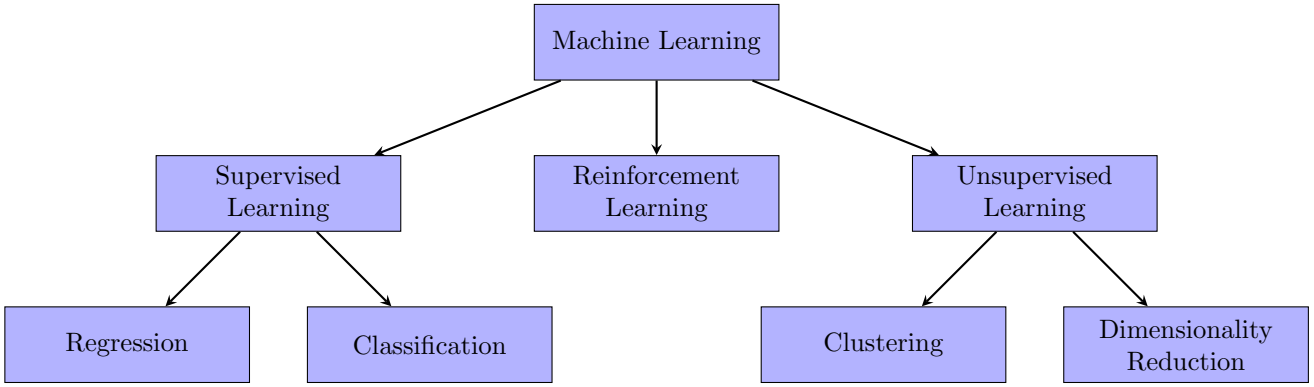


Figure 2.2: Machine learning tasks

2.3.3 Machine Learning Application

Machine learning used in many domains, among which we mention:

- Image recognition (face detection, character recognition).
- NLP Natural Language Processing (word prediction, top modeling).
- Speech Recognition.
- Medical diagnosis (anomaly identification in medical images).
- Information extraction.
- Big data visualization.
- Real-time decision and Robot navigation.

2.4 Neural Network

"A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns" (Gurney, 1997).

2.4.1 Biological Neuron

The human brain is the most complicated organ in the human body due to its tremendous ability to perceive the outside world and to learn. The foundational unit of the human brain is the neuron (Silva et al., 2017). This unit is divided into three main parts as shown in Figure 2.3a.

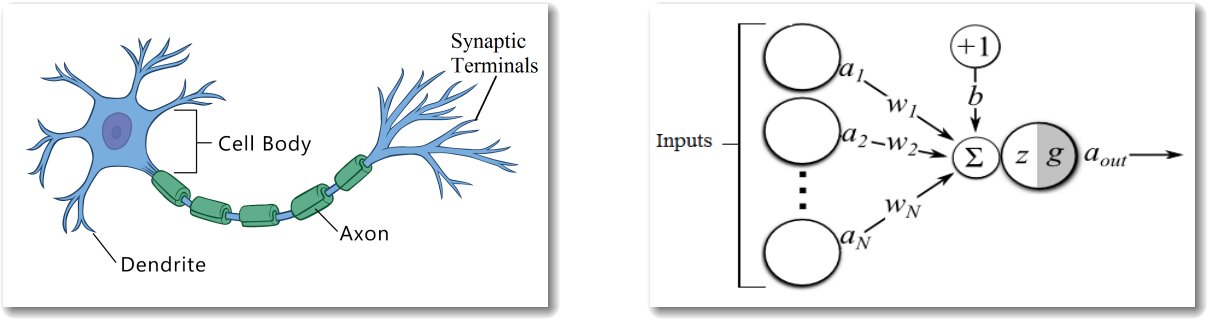
1. Dendrites: consist of a group of thin extensions and are intended to receive stimuli (information) from many other neurons.
2. Cell body (also known as "soma"): is the part responsible for producing the activation by processing all the information that comes from the dendrites, and it contains a group of organelles (nucleus, lysosome, centriole,..).
3. Axon: its mission is to direct stimulation to other neurons via synaptic terminals.

Synapses are bonds that enable the transfer of stimuli from a particular neuron to the dendrites of other neurons. In order to simulate the human learning mechanism and make the machine able to learn. Artificial neural networks are designed.

2.4.2 Artificial Neuron

Artificial neural networks are popular machine learning techniques that simulate the mechanism of biological learning, where each neuron from a network can be implemented as shown in Figure 2.3b (Da Silva et al., 2017), its parameters are:

1. Input connections (inputs): is a vector $(a_1, a_2, ..., a_n)$ with weights $(w_1, w_2, ..., w_n)$, each input is multiplied by its weight.
2. Pre-activation function z: is a summation function that sums weights after multiplies each of input by their own associated weight, with the addition of the bias b (used to adjust the output along with the weighted sum of the inputs to the neuron); $z = \sum_1^n a_i w_i + b$.
3. Activation function g: transforms the pre-activation; $g(z)$.
4. Output: output the final activation; $a_{out} = g(z)$.



(a) Biological Neuron (b) Artificial Neuron

Figure 2.3: Biological neuron vs Artificial neuron

Activation Functions

Are nonlinear mathematical functions that calculate the level of neuron activation. The function is chosen according to the problem and outputs, we list the most popular activation function:

- **Sigmoid**: Normalizes the output of each neuron, its output value is in the range [0,1].

$$g(z) = \frac{1}{1 + e^{-z}} \tag{2.1}$$

- **Rectified Linear Unit (ReLU)**: It's gives 0 if it receives a negative input, and returns the same positive value x otherwise.

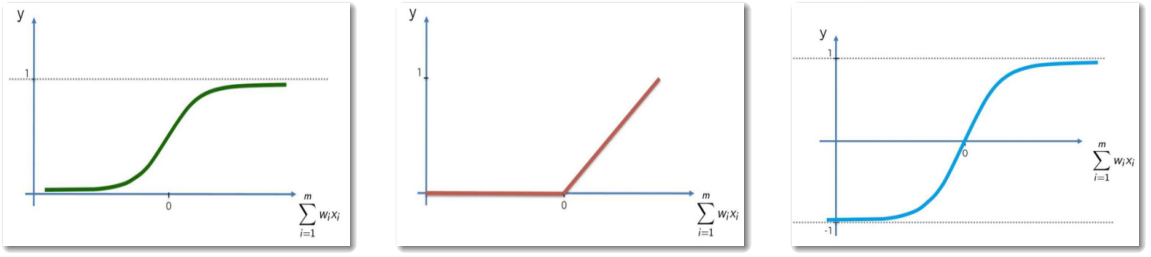
$$g(z) = \max(z, 0) \tag{2.2}$$

- **Hyperbolic Tangent (TanH)**: Normalizes the output of each neuron, its output value is in the range [-1,1].

$$g(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{2.3}$$

- **Softmax**: Used when outputs are multi class (multi-classification) and its output value is in the range [0,1]

$$g(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{n} e^{z_j}}, i = 1, 2, 3, .., n \tag{2.4}$$



(a) Sigmoid (b) ReLU (c) TanH

Figure 2.4: Activation functions

2.4.3 Architectures of ANN

The basic architecture of an artificial neural network consists of three basic parts (named layers) (NUNES and DA SILVA, 2018):

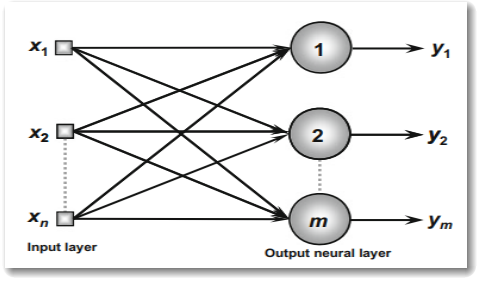
- Input layer: it is the first layer responsible for receiving information (data) from the external environment.

- Hidden (intermediate or invisible) layers: these layers perform most of the basic work in a network. The layers are made up of neurons responsible for excreting features.
- Output layer: after processing with neurons in the previous layers, this layer produces and delivers the final network outputs.

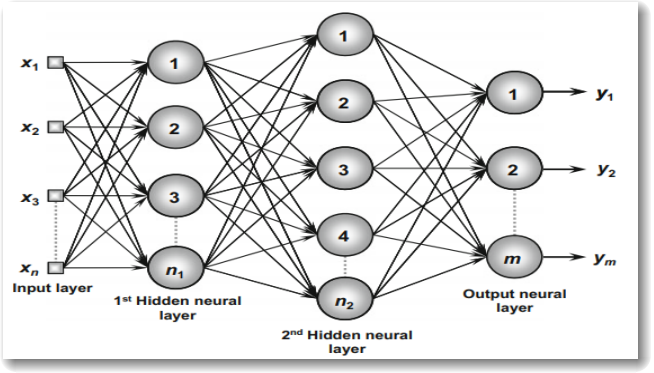
The main architectures of the ANN are divided into two parts (Silva et al., 2017):

Feedforward Architectures: This architecture is called feedforward because information flows always in one direction which is from the input layer to the output layer (no feedback connections). It has two types:

- **Single-Layer Architecture:** This artificial neural network consists of the input layer and the output layer and is usually in linear filtering problem. This network is illustrated in Figure 2.5a.
- **Multiple-Layer Architecture:** This artificial neural network consists of the input layer and a number of hidden layers (one or more) and the output layer and is usually used in pattern classification and optimization. Figure 2.5b illustrates this network.



(a) Single-Layer architecture



(b) Multiple-Layer architecture

Figure 2.5: Feedforward architectures

Among the main network types belonging to feedforward architecture, we find Multilayer Perceptron Networks (MLP). Each perceptron in the first layer (the input layer), sends outputs to all the perceptrons in the second layer (the hidden layer), and all perceptrons in the second layer send outputs to the final layer (the output layer).

Feedback (Recurrent) Architectures: In this architecture as shown in Figure 2.6, the outputs of the neurons are used as feedback inputs for other neurons, this feature makes the network able to dynamically process the information, it also features the ability to maintain relationships and to store information. So it is used in time series prediction, system identification, and optimization.

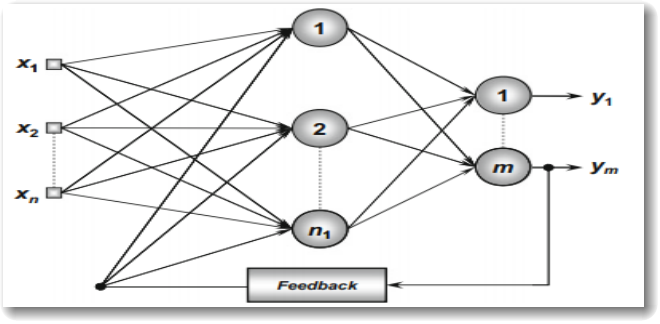


Figure 2.6: Feedback architecture

2.4.4 Neural Network Learning

One of the most important features of an artificial neural network is its ability to learn. Neural network learning divided into two types (Brabazon et al., 2015):

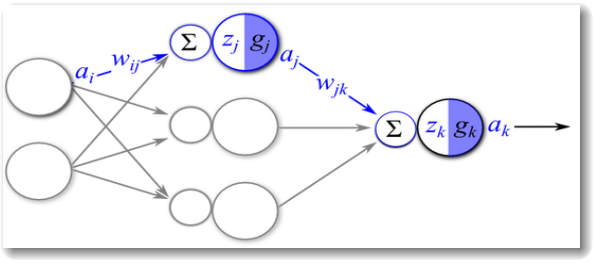
1. Supervised Learning: This type of learning is based on inputs with their correct outputs. The neural network is trained by improving the values of the weights in order to reach the appropriate weights, to be able to produce the output with the required accuracy corresponding to the input, and that makes it able to produce the correct outputs for any new input. This type of learning is based on a training algorithm called The Backpropagation Algorithm.

2. Unsupervised Learning: This type of learning depends only on the inputs without their correct outputs. The network works to find relationships that link these inputs and try to classify them into similar categories by extracting distinct patterns for each category. This enables the network to give output for the new input, depending on its patterns.

Backpropagation Algorithm

The backpropagation (BP) algorithm is one of the most popular learning algorithms in neural networks. It's short for the backward propagation of errors since the error is computed at the output and distributed backward throughout the network's layers. This is done through these successive steps:

1. Forward-propagate: The first step of the backpropagation algorithm is to propagate the inputs forward through the network layers towards the outputs, as shown in Figure 2.7, a_k is the output of the network, and to obtain it the pre-activation z_l and activation a_l applied for all layers l (index i with the input layer, index j with the hidden layer, and index k with the output layer) by the equation 2.5.



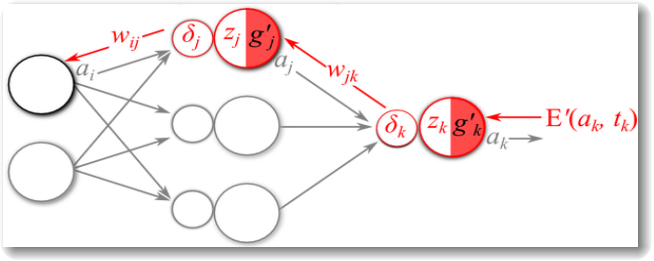
$$a_k = g_k(b_k + \sum_j g_j\,(b_j + \sum_i a_i w_{ij})\,w_{jk}) \quad (2.5)$$

Figure 2.7: Forward-propagate

2. Back-propagate: The second step of the algorithm is to calculate the error E between the network output a_k and the real output t_k. This error calculates through a cost function (this function can be as simple as MSE (mean squared error) or more complex like cross-entropy).

$$E = \frac{1}{2}\sum_{k\in K}(a_k - t_k)^2 \quad (2.6)$$

The error signal δ' is calculated (δ_k for the output layer and δ_j for the hidden layer) by the following equations (2.7, 2.8) to backpropagate it toward the input as shown in Figure 2.8.



$$\delta_k = g'_k(zk)E'(a_k, t_k) \quad (2.7)$$

$$\delta_j = g'_j(z_j)\sum_k \delta_k w_{jk} \quad (2.8)$$

Figure 2.8: Back-propagate

3. Calculate parameter gradient: The third step of the algorithm is to calculate the gradients of the error function for weights in each layer using the forward signals a_{l-1} (a_i and a_j) and the backward error signals δ_l (δ_j and δ_k). And also for the biases.

$$\frac{\partial E}{\partial w_{ij}} = a_i\delta_j \quad (2.9) \qquad \frac{\partial E}{\partial w_{jk}} = a_j\delta_k \quad (2.11)$$

$$\frac{\partial E}{\partial b_j} = b_j\delta_j \quad (2.10) \qquad \frac{\partial E}{\partial b_k} = b_k\delta_k \quad (2.12)$$

4. Update parameters: The last step is to update all weights and biases using the gradients calculated in the third step. With the learning rate parameter η. By the following equations and as shown in Figure 2.11.

$$w_{ij} = w_{ij} - \eta(\frac{\partial E}{\partial w_{ij}}) \quad (2.13) \qquad w_{jk} = w_{jk} - \eta(\frac{\partial E}{\partial w_{jk}}) \quad (2.15)$$

$$b_j = b_j - \eta(\frac{\partial E}{\partial b_j}) \quad (2.14) \qquad b_k = b_k - \eta(\frac{\partial E}{\partial b_k}) \quad (2.16)$$

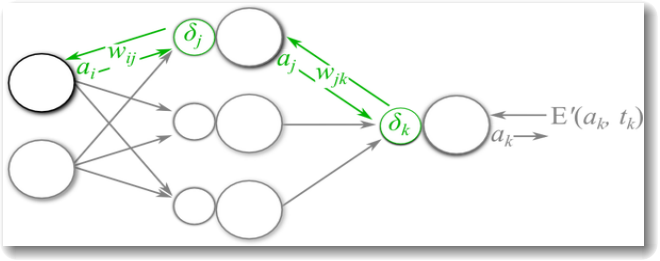


Figure 2.11: Update parameters

These four steps are repeated until the network error reaches an acceptable low value. At this point, we can say that the artificial neural network has been trained (Stansbury, 2014).

Practical Issues in Neural Network Training

To improve the performance of neural networks, we face many challenges, especially during the training process (Aggarwal, 2018). The most important of them are:

Overfitting: The model is ideally predicted during training does not guarantee that it gives ideal results during testing, especially for complex models and a small set of data. There is often a gap between the training data and test data performance and this is the problem of Overfitting or the generalization problem as shown in Figure 2.12. Among the signs of this problem:

- When completely different data sets are used in the model training, this results in different predictions each time for the same test data.
- The gap of prediction error between training data and test data.

Among the tricks that we can be used to avoid falling into the problem of Overfitting we mention:

- *Penalty-based regularization*: the idea is to implement a set of restrictions or establish a penalty. It is the most used technique in neural networks in order to avoid overfitting.
- *Dropout*: a technique designed specifically for neural networks to reduce the overfitting problem based on the selective dropping of nodes to create different neural networks.
- *Early stopping*: consists to stop the training at a specific point to avoid the gap between the error during training and the test.

Underfitting: The counterpart of overfitting, this happens when the model does not learn enough from the training data and becomes unable to predict correctly as shown in Figure 2.12.

In order to avoid this problem, one can :

- *Add neuron layers or input parameters*: adding input parameters into the model or adding neuron layers helps to improve the model and generate better predictions.
- *Reduce regularization parameter* : by reducing bias, this helps to avoid underfitting.
- *Increase the training time.*

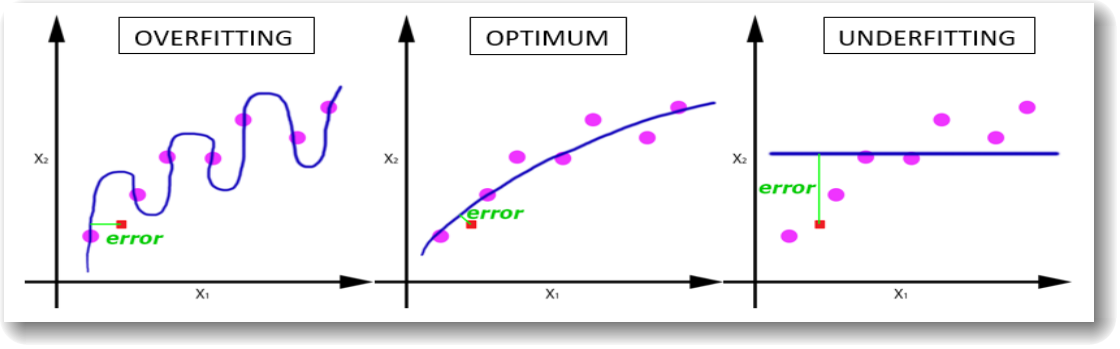


Figure 2.12: Overfitting vs Underfitting

The Bias-Variance Trade-Off: Is a way of analyzing a learning algorithm's expected generalization error. It captures the trade-off between the power of a model and its performance on limited data (Aggarwal, 2018). As shown in Figure 2.13 the basics of this concept are :

- *Bias*: It is the error caused by the predictions of the model and the correct value that we tried to predict (underfitting when the model has high bias).
- *Variance*: is the error due to the amount of overfitting done during model generation (overfitting when the model has high variance).

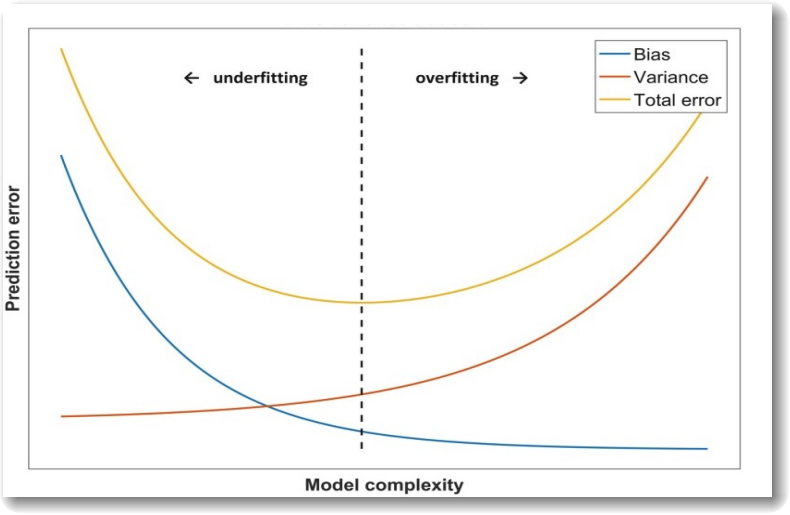


Figure 2.13: Bias-Variance trade-Off

2.4.5 Advantages and Disadvantages

Advantages

Neural networks have many advantages among them (Lancashire et al., 2009):

- Solve non-linear problems: unlike traditional algorithms, neural networks can implicitly detect complex nonlinear relationships between variables.
- Fault-tolerant: neural networks can deal with noisy or fuzzy information, as well as incomplete data.
- Continuous Improvement: the network becomes more experienced and efficient in making decisions when it trained more

Disadvantages

- Black Box: the most common disadvantage of the neural network is that it is a black box so that it cannot be known how and why the neural network produced these outputs.
- Development Time: the development of neural networks is not a simple matter, especially when trying to solve new problems not previously solved by neural networks.
- Amount of Data: the large amount of data required for the network learning process.

2.5 Deep Learning

Due to the success of artificial neural networks in solving difficult problems in artificial intelligence and their advantage via learning, they relied on them upon the emergence of deep learning.

Deep learning is one of the most important branches of machine learning and its most important features are:

- Depth: so that the term 'deep' refers to the number of hidden layers that make up the deep neural network and this is what distinguishes it from normal neural networks.
- Special ability to extract features: by processing a large amount of data without the need for external intervention, unlike traditional machine learning algorithms as shown in Figure 2.14.

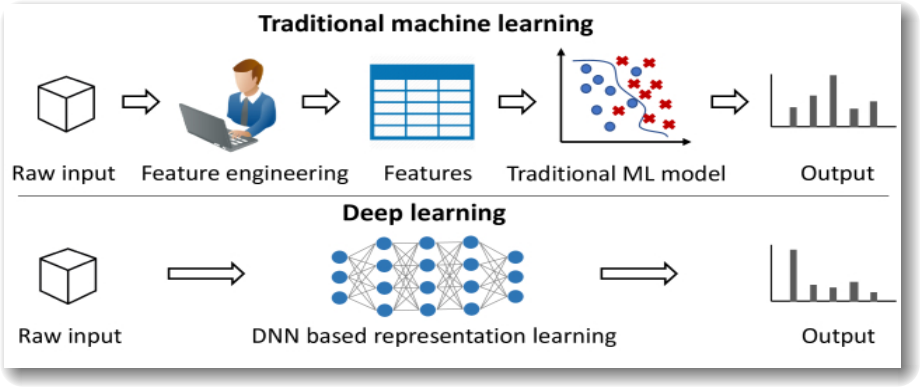


Figure 2.14: Machine learning vs Deep learning

DL has witnessed great prosperity in recent years due to a combination of factors:

- Big data: it is one of the most important things that deep learning needs. Where digitization led to the presence of a large amount of data, which has become easy to access and use.
- GPU and cloud computing: Deep Learning (DL) relies on processing a large set of data, which requires computers with high capabilities. Therefore, the increase in computing power encouraged the use of deep learning.

2.5.1 Convolutional Neural Network

Convolutional Neural Networks (CNNs) is a class of deep feedforward artificial neural networks inspired by the mechanism of visual perception of living creatures, that the neurons in the visual cortex search for specific characteristics (Gu et al., 2018). CNNs are applied to analyze visual images, such as image classification, object detection, and character recognition. The most important thing that distinguishes it from neural networks is the convolutional layer, which extracts features.

CNN Architecture

As shown in Figure 2.15, the structure of convolutional neural network divided into layers, each with its own role.

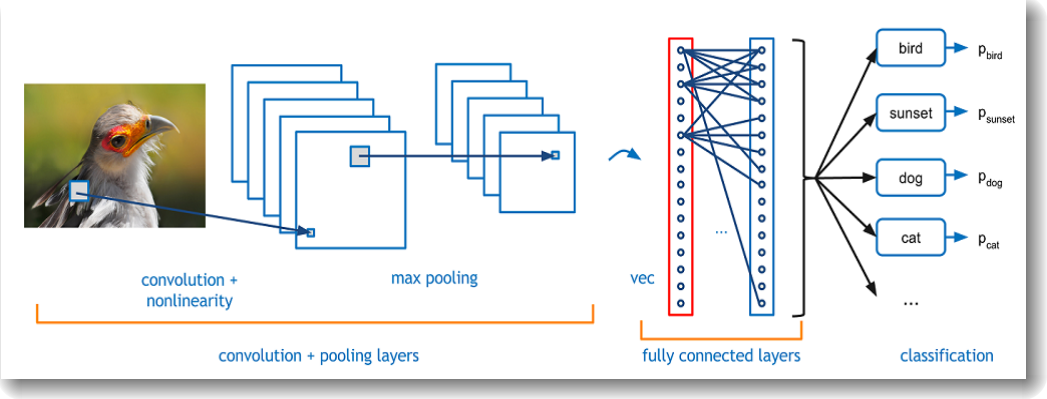


Figure 2.15: CNN Architecture

- **Convolutional Layer:** It is the first and basic layer for CNN aims to extract the features from the input image. So the convolution is a mathematical operation applied to input data (input image pixels) using the convolution filter (kernel) to produce a features map as illustrated in Figure 2.16.

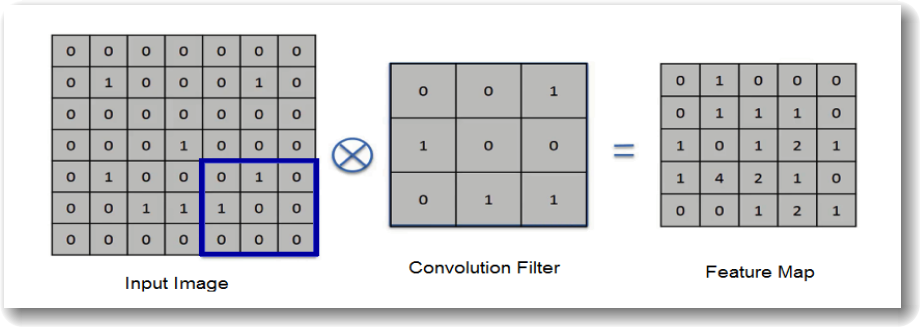


Figure 2.16: Convolution operation

Kernels: Also called filters (like the classic filters in image processing). Usually, kernels are small in spatial dimensions (3×3) and applied to all pixels entered with certain strides. These Kernels work for edge detection, sharpen, box blur, or Gaussian blur (Albawi et al., 2017).

In order to optimize the outputs of the convolution layer. Three hyperparameters are usually adjusted (O'Shea and Nash, 2015; Albawi et al., 2017).

- Depth: can be reduced by reducing the number of filters used in the convolution operation.
- Stride: is the step taken to multiply with a filter. In order to reduce the amount of overlapping, the stride should be incremented.
- Padding: it is a useful process to give more control over the output dimensions, by zero-padding (padding the zeros in the border of the input) or valid padding (drop part from the input and keep only the valid part).

- **Activation Layer:** This layer use the activation function ReLU. To enter the non-linearity in the CNN and get rid of all negative values as shown in Figure 2.17.

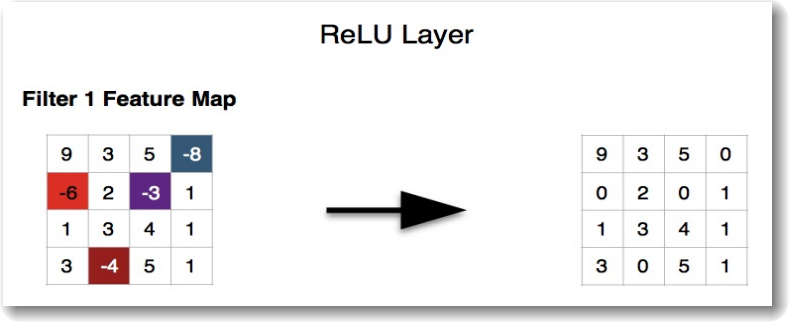


Figure 2.17: Activation operation

- **Pooling Layer:** The goal of this layer is to reduce dimensions by minimizing the number of parameters while preserving important ones. It is usually placed between two convolutional layers (Albawi et al., 2017). Different types of pooling exist as shown in Figure 2.18:
  - Max Pooling: take the maximum value in the pooling window and it is the most common type.
  - Average Pooling: take the average of values in the pooling window.

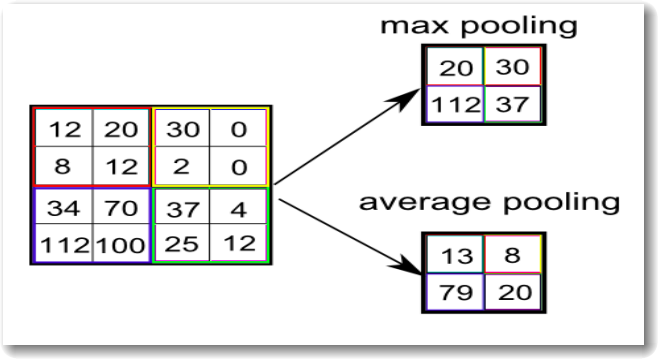


Figure 2.18: Pooling types

- **Flattening Layer:** In this layer, the input size is transformed from shape (width, height, depth) to a one-dimensional array. This is to take advantage of all layer information and be ready to link to the artificial neural network shown in Figure 2.19. It is considered the last step in extracting features.

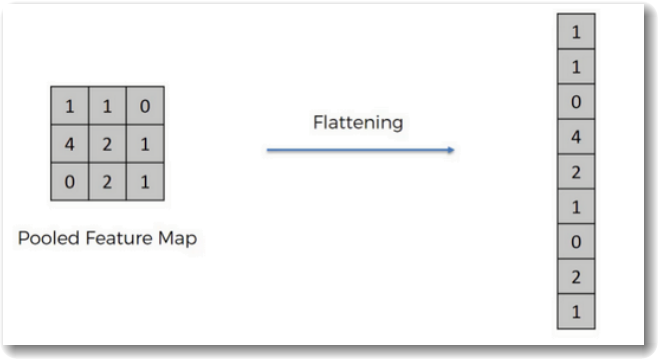


Figure 2.19: Flattening

- **Fully Connected Layer:** This layer as shown in Figure 2.20, after the flattering the one-dimensional array is fed it into an artificial neural network. This ANN is a fully connected layer and it is the last layer in the network. Fully connected layer uses the non-linear Softmax function to classify its outputs.

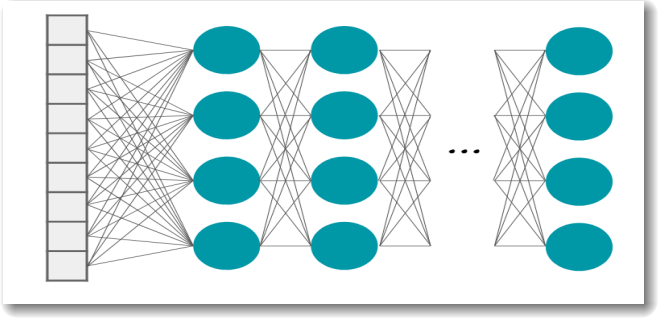


Figure 2.20: Fully connected layer

2.5.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a class of neural networks that are used for sequence modeling such as Natural Language Processing (NLP) and speech processing applications (Pouyanfar et al., 2019).

For example, when we read a text, we don't think of every word alone, but we understand the sequence of words to get the meaning of the sentence. At the same time, we don't forget the meaning of what we had read. The ideas we absorbed remain in our memories to enable us to understand the total meaning of paragraph (Alom et al., 2019). This is a typical case of Recurrent Neural Networks (RNNs).

The cells in RNN store this previous information and use it in hidden state memory, which performs a kind of saving of the inputs that passed before the current inputs. As the value of the hidden state at any time moment is dependent on the value of the hidden state at the previous moment and the inputs at the current moment as shown in Figure 2.21.

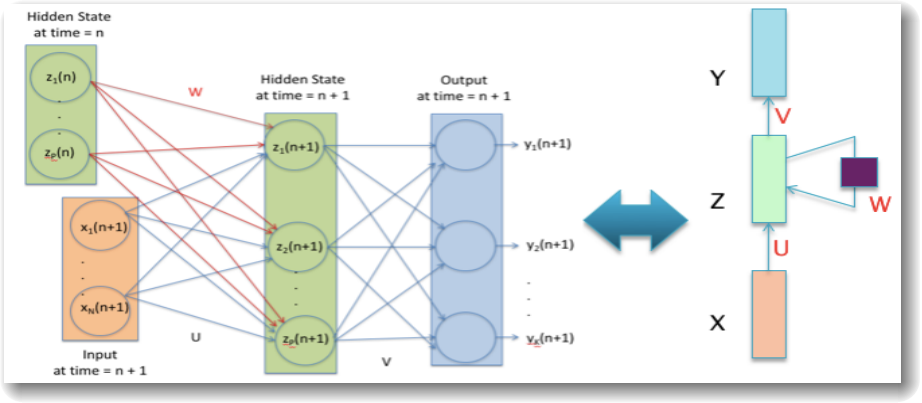


Figure 2.21: Recurrent neural network

The Architecture of Recurrent Neural Networks

We assume that an input sequence X is given by $X = (x_1, x_2, ..x_T)$ with input weight matrix W, and the hidden state vector $H = (h_1, h_2...h_T)$ with input weight matrix U, and output vector $Y = (y_1, y_2...y_T)$. With activation function g. Were b is a bias term (Kim et al., 2016; Alom et al., 2019). As shown in Figure 2.22 the simple RNN has two versions to calculate H and Y, with $t = 1$ to T as follows:

Elman network (Elman, 1990): An Elman network is a MLP. That bases outputs from the hidden layers as an input with the normal input.

$$h_t = g(W_x x_t + U_h h_{t-1} + b_h) \tag{2.17}$$

$$y_t = W_y h_t + b_y \tag{2.18}$$

Jordan network (Jordan, 1997): A Jordan network is a MLP. That using the outputs from the output layers as an input with the normal input.

$$h_t = g(W_x x_t + U_h y_{t-1} + b_h) \tag{2.19}$$

$$y_t = W_y h_t + b_y \tag{2.20}$$

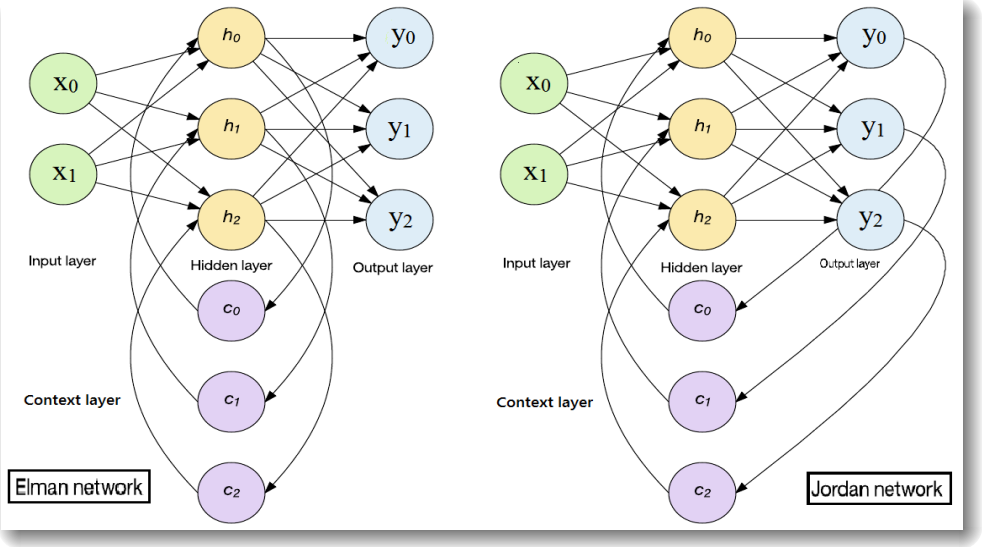


Figure 2.22: Elman architecture / Jordan architecture

The Challenges of Training Recurrent Networks

The architecture of Recurrent Neural Networks (RNNs) makes them very hard to train, especially if the input sequence is long (Aggarwal, 2018). The error gradient calculate to update the network weights, in RNNs two problems can occur (Pascanu et al., 2013):

- *Exploding Gradient*: When those gradients accumulate during an update, the result will be very large.
- *Vanishing Gradients*: When those gradients are small or zero, it will easily vanish.

In order to address these problems, a set of solutions has been proposed, the most important of these effective solutions:

Long Short Term Memory (LSTM): Proposed by Hochreiter and Schmidhuber in 1997. Designed to solve the Vanishing Gradient problem, It's a type of recurrent neural network when the recurrent hidden layer is replaced by LSTM cell (Kim et al., 2016) as shown in Figure 2.23.

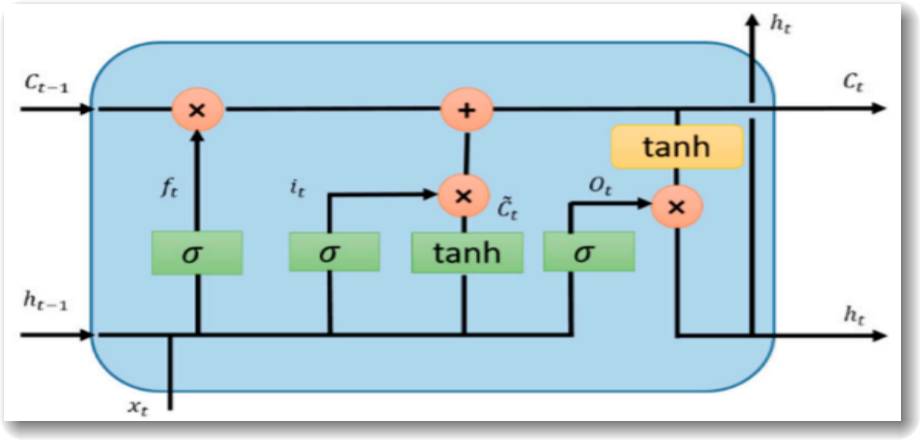


Figure 2.23: LSTM cell

The most important differences that distinguishes LSTM from RNN are:

- Cell Memory C_t: In order to generate the cell memory C_t, the LSTM works on the cell memory C_{t-1} of the previous stage, and it is used to create the next hidden state h_t.
- Gates in LSTM:
  - *The input gate*(i_t): Decides the necessary inputs.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.21}$$

  - *The forget gate*(f_t): Calculates the previous memory important ratio.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.22}$$

  - *The output gate* (o_t): Determines whether memory cell activate or not.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.23}$$

  Update Cell Memory Value C_t and Hidden State vector h_t:

$$C_t = f_t c_{t-1} + i_t \tanh(W_c.[h_{c-1}, x_t] + b_c) \tag{2.24}$$

$$h_t = o_t \tanh(C_t) \tag{2.25}$$

W_{ci} , W_{cf} and W_{co} are denoted weight matrices for peephole connections, and σ is the logistic sigmoid function.

BLSTM Bidirectional recurrent neural networks (RNN) are putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another.

Gated Recurrent Units (GRUs): GRU can be considered as a simplification of the LSTM (Alom et al., 2019). The main difference from LSTM is that GRU does not have a cell memory state c_t. Instead, it uses the Update Gate z_t and the Reset Gate r_t as shown in Figure 2.24 by the following equations:

- *The Update Gate* z_t:

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{2.26}$$

- *The Reset Gate* r_t:

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{2.27}$$

- *Hidden State* $\tilde{h}_t$:

$$\tilde{h}_t = \tanh(W[r_t * h_{t-1}, x_t]) \tag{2.28}$$

- *The new Hidden State* h_t:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.29}$$

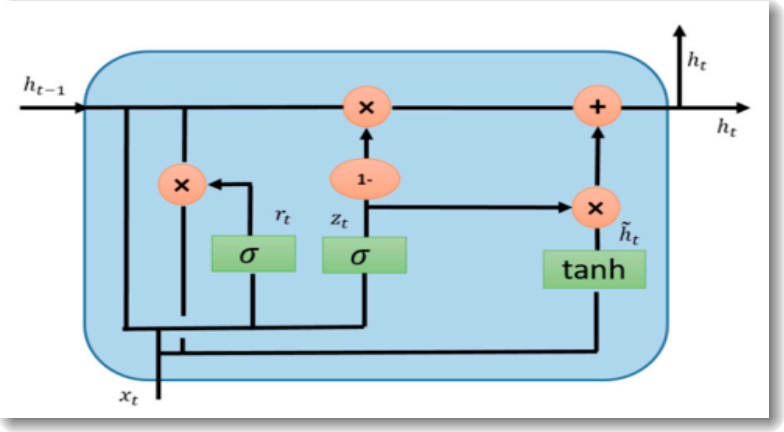


Figure 2.24: GRU

2.5.3 Various Applications of Deep Learning

The applications of deep learning are endless and have covered various fields (Pouyanfar et al., 2018), among which we mention:

Natural Language Processing

One of the most difficult computer challenges is to understand the human natural language, due to its complexity and laborious structure. For this purpose, a set of algorithms and techniques has been designed for typical NLP applications such as sentiment analysis, machine translation, and paraphrase identification.

Visual Data Processing

One of the most important areas that have benefited from deep learning is the computer vision, especially the Convolutional Neural Network (CNN) as it showed great results in this field, especially image classification, object detection, semantic segmentation, and video processing.

Speech and Audio Processing

Understanding human speech is one of the most important steps for an ideal human-computer interaction. The audio processing process works on electrical or analog audio signals. This field has many applications, including speech emotion recognition, speech enhancement, and music classification.

2.6 Conclusion

This chapter mentioned basic concepts of machine learning, we focused particularly on neural networks and concluded with deep learning. The rapid development in these areas and the ability of the computer to process various data, and the emergence of computer programs of negative use, it became necessary to establish a mechanism to distinguish between human and machine, and this is what we will discuss in the next chapter.

CHAPTER 3

TEXT-BASED CAPTCHA BREAKING

3.1 Introduction

None of us today can live without the Internet that most domains now depend on it (commerce, education, communications, etc.), often before benefit from Internet services for example create an e-mail account we first have to pass a test. This test is a CAPTCHA, it's used to prevent the presence of fake accounts created by computer programs (bots) that can make a negative impact on web sites. This test is designed so that a human can pass it, but the machine cannot.

Depending on the weaknesses of machines, CAPTCHA is designed in several types (text, image, audio, and video) in order to make it more difficult to break by bots. And the text-based CAPTCHA is the most used compared to other types, due to its ease of use and low cost of design.

In this chapter, we introduce firstly a brief history of CAPTCHA development followed by a classification of CAPTCHA types. We secondly focus on the text-based CAPTCHA process. In the last section, we report some main state-of-the-art methods on text-based CAPTCHA breaking.

3.2 Emergence

As we mentioned in the previous chapter, the term Artificial Intelligence (AI) is always associated with the Turing test, considering that the Turing test is the first model of machine intelligence (Machinery, 1950). The purpose of the Turing test was to explain the machine's ability to mimic some of the human behavior, which is linked with the intelligence (Brodić and Amelio, 2019). In 1996 and in order to exclude bots from exploiting web services, Moni Naor proposed to rely on Turing tests and on tasks that humans excel at performing than the machine (Naor, 1996). These give rise later to the most types of CAPTCHA.

In 1997, the Alta Vista website needed to prevent the automatic submission of URLs. For this purpose a scheme was developed by Andrei Broder and his colleagues at the **DEC** System Research Center[1]. Their scheme was an image of characters specially designed so that it would not be recognized by automatic vision systems but it is easily recognized by humans (Lillibridge et al., 2001).

In September 2000, there was a challenge between Audie Manber (Chief Scientist at Yahoo!) and Professor Manuel Blum and his students in Carnegie of Mellon University to design a "reverse Turing test" to prevent bots from registering for services of mail, mailbox, Yahoo chat room, etc. (Baird and Popat, 2002). It is a modified Turing test, which differs in that an examiner is a machine and the test provides a distinction between machine and humans rather than a machine intelligence test (Baird et al., 2003).

In January 2002, Prof. Baird, H. S., Popat, K. launched the first workshop about 'Human Interactive Proofs' (HIPs), so that everything related to CAPTCHA test was discussed (Baird and Popat, 2002). HIPs are a set of challenge/response protocols for authenticate the human (vs. machine), herself (vs. anyone else), an adult (vs. child) (Brodić and Amelio, 2019) through a set of discrimination-based tests. The most important is the distinction between human actions and machine activities.

After these years, most institutions and companies that rely on web services began to add or design their CAPTCHA test to their websites to protect from malicious exploitation by bots.

3.3 Definition and Characteristics

The term CAPTCHA for (Completely Automated Public Turing test to tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper, and John

[1]The Systems Research Center was a research lab that was created by Digital Equipment Corporation (DEC) in 1984, in Palo Alto, California.

Langford of Carnegie Mellon University (M. Blum and Langford, 2000), they also provided the first definition of the CAPTCHA as "*a CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot*". It was also defined as a class of Human Interactive Proofs (HIPs) (Baird and Popat, 2002) and a reverse Turing test (Baird et al., 2003).

To evaluate any CAPTCHA, a group of scientists from Carnegie Mellon University (M. Blum and Langford, 2000), Microsoft Research (Rui and Liu, 2004) and the Palo Alto Research Center (Baird and Popat, 2002) presented the most important characteristics that must be included in a CAPTCHA:

- **Accessibility**: Should be available to everyone, it should be as independent as possible from the user's language, age, and educational background.
- **Automated**: "Completely Automated" in the term CAPTCHA means the machine must be able to generate the test, but it cannot solve it.
- **Easy for humans**: Should be solved easily and quickly (take no more than 30 seconds) by human users. The human success rate in solving the CAPTCHA test should be 90% or higher (Bursztein et al., 2011).
- **Hard for machines**: Should be based on a hard artificial intelligence problem where the best current techniques find it difficult to solve by machines. The computer's pass rate for a CAPTCHA test must be 1% or lower (Bursztein et al., 2011).
- **Open**: "Public" in the term CAPTCHA means that the database(s) and algorithm(s) used to generate the test must be public.

3.4 CAPTCHA Classification

The CAPTCHA test is designed in several formats such as text, image, video, etc. So it was classified on the basis of its design.

3.4.1 Linguistic CAPTCHAs

The designers of this type of CAPTCHA relied on that machines confront a problem with Natural Language Processing (NLP) tasks such as understanding the content of a text. Linguistic CAPTCHA appears as plain text on the web page and the user must understand the text content to solve this test (Kluever, 2008). Among Linguistic CAPTCHA tests, we found knock-knock and SS-CAPTCHA.

knock-knock CAPTCHA (Ximenes et al., 2006) is a linguistic CAPTCHA presented in 2006 based on knock-knock jokes, so the user should find the real joke among fake jokes as shown in Figure 3.1.

SS-CAPTCHA (Yamamoto et al., 2010) is a test that shows a set of real and fake sentences that have been automatically created (they relied on google translate to create these sentences) and asks the user to determine the real sentences as shown in Figure 3.2.

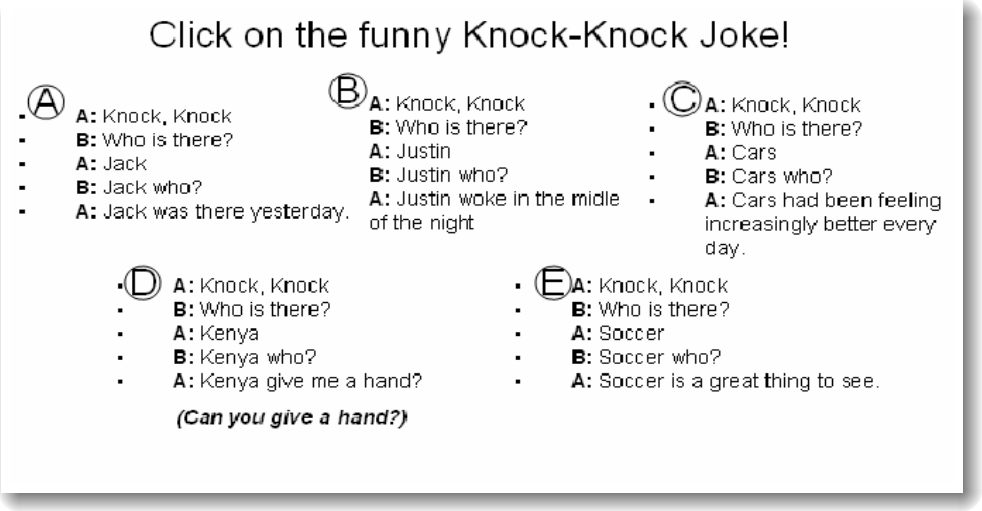


Figure 3.1: knock-knock CAPTCHA (Ximenes et al., 2006)

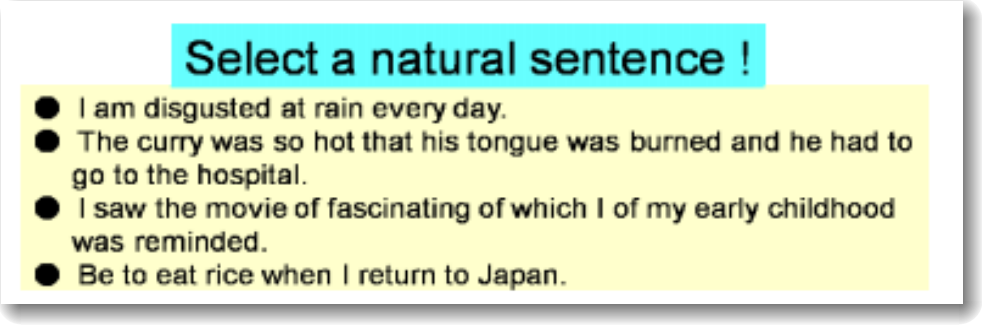


Figure 3.2: SS-CAPTCHA(Yamamoto et al., 2010)

3.4.2 Text-based CAPTCHAs

This type of CAPTCHA is also categorized as OCR-Based Methods (Yadava et al., 2011; Shirali-Shahreza and Shirali-Shahreza, 2008) since its design relies on OCR [2] programs vulnerabilities. Text-based CAPTCHA is an image that contains a combination of distorted characters

[2]Optical Character Recognition (OCR) is the electronic or mechanical conversion of images of typed, handwritten, or printed text into machine-encoded text.

that the user is asked to recognize and enter to pass the test. It is one of the most common types of CAPTCHA due to its easy design (Tang et al., 2018) which led to the presence of many designs.

- **English Text CAPTCHAs,** since English is a universal language, we find that the most used type of text-based CAPTCHA is those which depend on the English characters.

  PessimalPrint (Baird et al., 2003): Also called "reverse Turing test", their design relied on the fact that low-quality words were indistinguishable by the machine. They chose 70 natural English words that they thought were difficult to recognize by OCR programs, and apply distortion mechanisms to decrease their quality as shown in Figure 3.3a.

  ReCAPTCHA (Von Ahn et al., 2008): In their test, they use words found in ancient and historical printed texts, due to the difficulty of recognizing them through OCR programs and they added a little of deformation as shown in Figure 3.3b.

  ScatterType (Baird et al., 2005): It uses about 15,000 meaningless English words randomly generated and 100 font types, cutting and scattering operations are applied to make it more difficult as shown in Figure 3.3c.

(a) PessimalPrint (b) ReCAPTCHA (c) ScatterType

Figure 3.3: English text CAPTCHAs

- **Non-English Text CAPTCHAs,** due to a large number of attacks on the CAPTCHA designed by the English characters, a group of CAPTCHA based on different languages are designed (Roshanbin and Miller, 2013):

  Arabic CAPTCHA (Khan et al., 2013): In their design, the authors benefit from the natural characteristics of the Arabic language (number and position of dots, overlap of words, diacritics) these make it difficult to distinguish by OCR programs with distortion added as well to make it more difficult as shown in Figure 3.4a.

  In DavaCAPTCHA (Yalamanchili and Rao, 2011): This CAPTCHA design based on a writing script called "Devanagari" that is used to write the Hindi language for its difficult natural characteristics (position of the strips, overlap of words) as shown in Figure 3.4b.

Chinese CAPTCHA (Wang and Bøegh, 2013) : One of the most famous languages that contain a large number of letters (3755 characters) is Chinese (Tang et al., 2018), and this is what makes the solution space bigger. The designers of this CAPTCHA relied on creating random words from Chinese characters and application of deformations and Multi-layer mechanisms to make it stronger as shown in Figure 3.4c.

(a) Arabic CAPTCHA (b) DavaCAPTCHA (c) Chinese CAPTCHA

Figure 3.4: Non-English text CAPTCHAs

- **Handwritten Text CAPTCHAs,** identify handwriting was considered a challenge for the machine, and this made them take advantage of this weakness to establish new handwriting based CAPTCHA (Rusu and Govindaraju, 2005). Many of these types were designed, among them we mention the following works:

  The authors of (Rusu et al., 2010) rely on the *Gestalt laws of perception* (Koffka, 1935) and *Geon theory* (Biederman, 1987) which state that humans are able to recognize distorted images and create a complete picture of the partial information "grouping idea" and the change in shape, size, or position of the organism does not affect its identification potential by humans. An example of this type is shown in Figure 3.5.

Figure 3.5: Handwritten text CAPTCHAs (Rusu et al., 2010)

To increase the security and stiffness of the text-based CAPTCHA and make it unbreakable by bots, it was strengthened by a set of resistance mechanisms. But this made it harder even for humans, which led to user dissatisfaction. This led to the emergence of image-based CAPTCHA.

3.4.3 Image-based CAPTCHAs

Chew and Tygar from the University of California proposed the problem of understanding the image semantic and object detection by machine (Kluever, 2008), which led them to image-based CAPTCHAs. This type of CAPTCHA depends on a set of images that are often meaningful this type of CAPTCHA has been successful and used in many forms and is still being used until now, among the most famous forms we mention: object detecting image-CAPTCHAs and subject detecting image-CAPTCHAs.

- **Object Detecting Image-CAPTCHAs,** in order to exploit the problem of the machine in object detection, this test displays a set of images and asks the user to discover one of the objects among the displayed ones. Below are some works in this class.

  ARTiFACIAL (Rui and Liu, 2004): Relies on the human face as the most common thing for human users, so that the user must find the human face by clicking on 6 points, 4 corners of the eyes, and two corners of the mouth as shown in Figure 3.6a.

  ASIRRA CAPTCHA (Elson et al., 2007): This CAPTCHA is based on a collage of images of dogs and cats the user must specify the images of cats as shown in Figure 3.6b.

  Collage CAPTCHA (Shirali-Shahreza and Shirali-Shahreza, 2007a): It randomly displays a set of images of objects with little rotation and asks the user to select an object as shown in Figure 3.6c.

(a) ARTiFACIAL (b) ASIRRA CAPTCHA (c) Collage CAPTCHA

Figure 3.6: Object detecting image-CAPTCHAs

- **Subject Detecting Image-CAPTCHAs,** it is also based on a set of images, but this time the user must know the subject that combines these images, such as:

  The Naming CAPTCHA (Chew and Tygar, 2004): This CAPTCHA is based on 6 images, so the user must type the term describing these images, this is illustrated in Figure 3.7a.

The Anomaly CAPTCHA (Chew and Tygar, 2004): The user must select the image with the subject different from among the 6 images presented to him, this illustrated in Figure 3.7b.

(a) Naming CAPTCHA (b) Anomaly CAPTCHA

Figure 3.7: Subject detecting image-CAPTCHAs

After the text-based CAPTCHA, the most common type of CAPTCHA that is used nowadays is the image-based CAPTCHA. To support it and make it respect the principle of availability to all. It has been augmented by a set of features, among them audibility.

3.4.4 Audio and video-based CAPTCHAs

Humans are distinguished by their high perception of speech compared to the machine even if it is in a noisy environment (Chan, 2003) where this gap was exploited in the design of audio-based CAPTCHAs. One of the most important features of this type of CAPTCHA is the ability of the person with visual impairment to use it so that it can be considered a complement to the CAPTCHAs that we have previously mentioned (Singh and Pal, 2014).

Audio reCAPTCHA (Chew and Tygar, 2004): Its idea is to create a short phrase that mostly contains a set of randomly chosen numbers and words and convert them into audio clips with the addition of noise to make them more difficult. The user is asked to enter the words in the audio clip as shown in Figure 3.8a.

CAPTCHA for illiterate people (Shirali-Shahreza and Shirali-Shahreza, 2007b): This is another example of an audio CAPTCHA where a set of images is displayed and the user is asked (by speech) to click on one of these images. It is considered one of the easiest CAPTCHA tests to be used even by illiterate people and young children as shown in figure 3.8b.

Video-based CAPTCHA (Kluever, 2008): Depending on machine weakness on video classification. In this kind of CAPTCHA, the video provides information to the user (Singh and Pal,

2014). After watching the video, the user must describe it with three words in order to pass the CAPTCHA. An example of this is shown in Figure 3.8c.

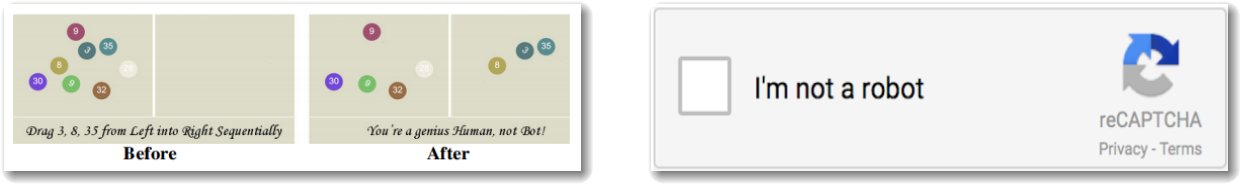
(a) Audio reCAPTCHA (b) Audio CAPTCHA

(c) Video CAPTCHA

Figure 3.8: Audio and Video-based CAPTCHAs

There are other types of CAPTCHA that differ from the previously mentioned so that they are designed to be more difficult to break, we mention :

- Game-based CAPTCHAs: Depending on the interactive intelligence that distinguishes the human, a group of game-based CAPTCHAs are designed (Zhang, 2010). The designers of this CAPTCHA as shown in Figure 3.9a asks the user to place the specified numbers on the right side.
- No CAPTCHA ReCAPCHA: Also known "Checkbox ReCAPTCHA" offered by Google[3] and it is the most widely used CAPTCHA service. A request sent to Google when the user clicks in the checkbox that contains (website key (obtained when registering with reCaptcha), a cookie for google.com, and information generated by the tool's browser checks). This request analyze by the advanced risk analysis system. if the system considers the user has a high reputation, the challenge will consider being solved, if not the system decides what type of CAPTCHA challenge will be presented to the user (text-based ReCAPTCHA or image-based ReCAPTCHA) (Sivakorn et al., 2016).



(a) Zhang's CAPTCHA (b) Checkbox reCAPTCHA

Figure 3.9: Zhang's CAPTCHA and Checkbox reCAPTCHA

[3]https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html

3.5 CAPTCHA Generation

In this section of our work, due to the many types of CAPTCHAs, we focus on the most used and most common types: Text-based CAPTCHA.

3.5.1 Text-based CAPTCHA Generation

Although it is not possible to find an algorithm or a general method for generating text-based CAPTCHAs, it is possible to observe a group of steps that have been followed most of the time and those can be changed slightly or added other steps (Banday and Shah, 2011). These steps are:

1. Often, the first step is to create an image that contains the CAPTCHA text with the appropriate dimensions.
2. Set the background of the image. The background of the image can be varied according to the resistance mechanism used (distorted or simple background).
3. Characters or words are randomly selected to create CAPTCHA text, from a previously chosen set by considering certain characteristics (often the difficulty to recognize by OCR soft wares). The majority use English alphabets and Arabic numbers but this does not prevent the presence of text dependent on other languages such as Arabic, Chinese or Indian, etc.
4. Choose the type, size, and color of the font used and apply a set of distortions. Also, the more fonts used each time, the stronger the CAPTCHA text will be. After that place the CAPTCHA text on the previously created image.
5. Finally, resistance mechanisms are applied to the image to make it more solid and unbreakable.

3.5.2 Resistance Mechanisms

One of the most important features of the CAPTCHA is the hardness to solve by machines. In order to achieve that, it has been strengthened with the following techniques :

Segmentation Resistance

Segmentation resistance mechanisms are often used in the text-based CAPTCHA so that the characters that make up the test are combined with a set of distortions to make them difficult to divide by OCR programs while trying to keep it easy to solve by humans (Tang et al., 2018; Roshanbin and Miller, 2013; Chellapilla et al., 2005). In the table 3.1 below, we mentioned some of these mechanisms.

Table 3.1: Segmentation Resistances (Tang et al., 2018; Roshanbin and Miller, 2013; Chellapilla et al., 2005)

Mechanism	Description	Example
Noise Arcs	Adding arcs to the CAPTCHA text, the arcs can be thin or thick.	
Fragmentation	Creating spaces in characters by splitting it vertically or horizontally.	
Overlapping	Removing the space between characters.	
Complicated Background	Adding images to the background of the test.	
Warp and Rotation	Rotating and twisting characters randomly.	
Hollow Scheme	Relying on the contour lines only to write the characters.	
Two-Layer Structure	Two vertical | horizontal layers of normal CAPTCHA.	

Recognition Resistance

After locating the letters (in the correct order), comes the problem of recognizing them. To make the recognize more difficult a set of mechanisms were applied (Chellapilla et al., 2005; Roshanbin and Miller, 2013).

- Using different fonts: by using a new font every time.
- Make characters different in size, angle, width, and location.
- Make it impossible to distinguish a character by counting its pixels: by making all characters contain the same number of pixels.
- Using random strings instead of word: by using words that are not in the language but rather were created randomly.
- Using a large database: by using a large set of characters that make up the test.

Other Mechanisms

Many other mechanisms are also used to boost the resistance of a CAPTCHA such as limiting the test time or the number of attempts, revealing the IP addresses that provide incorrect answers, use the tests that require human interaction usually similar to the game, using 3D images or structures, etc. (Roshanbin and Miller, 2013).

All these resistances were added, whether against segmentation or recognition in the context of improving the solidity of CAPTCHAs, which made breaking it a challenge.

3.6 Text-based CAPTCHA Breaking

The most common type of CAPTCHA that has been attacked or broken is the text-based CAPTCHA, so we find that most of the breakage techniques were designed to attack this type using traditional methods until machine learning and deep learning techniques.

In order to break any text-based CAPTCHA, we must go through steps as shown in Figure 3.10, The most important step is recognition, while the rest of the steps can be considered as supportive steps (Chen et al., 2017; Bursztein et al., 2011; Roshanbin and Miller, 2013).

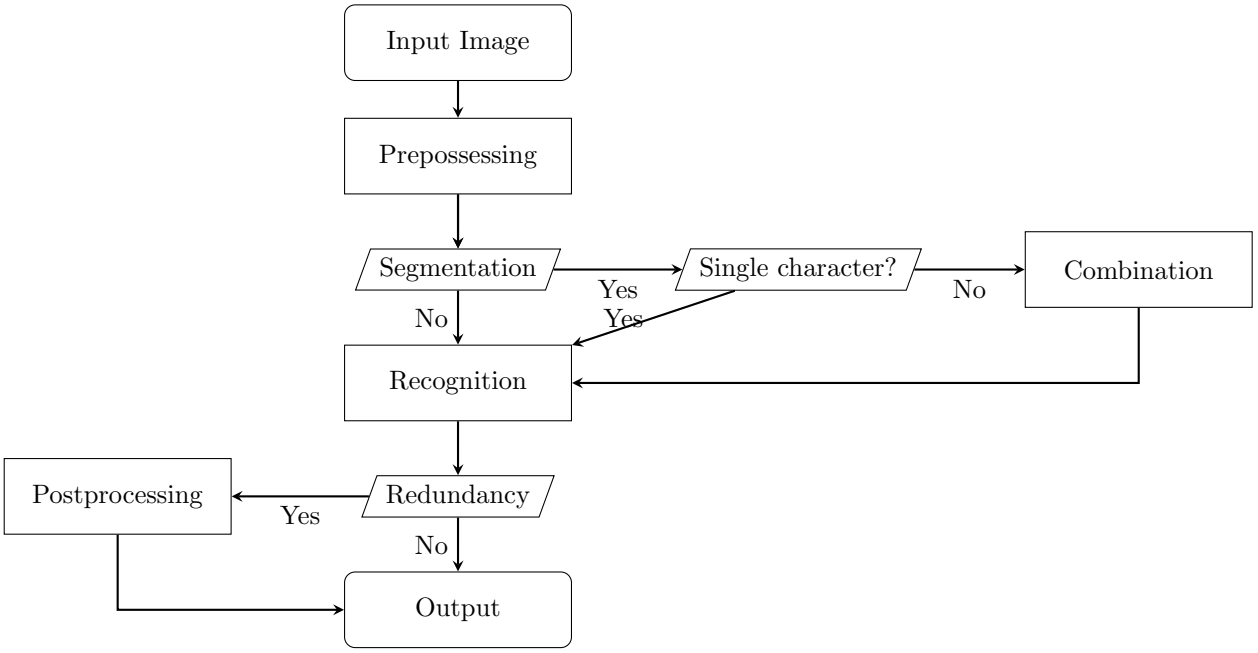


Figure 3.10: Text-CAPTCHA breaking process (Chen et al., 2017)

3.6.1 Preprocessing

This step can be considered as a preparation for the following two steps (segmentation or recognition). The focus is to make the text-based CAPTCHA image as clear as possible, by applying the following procedures:

- **Image Binarization:** This technique is used to remove distorted background and noise from the image, by converting the image to black and white using the threshold (Chen et al., 2017), Sauvola and Pietikainen's method (Sauvola and Pietikäinen, 2000) or Otsu's (Otsu, 1979) are used for this purpose.
- **Image Thinning:** It is considered one of the most prominent methods used in Preprocessing to analyze any image. It also plays an important role in image analysis and pattern recognition. Thinning consists of reducing a thick object in the image into a thin skeleton (Ben Boudaoud et al., 2015). The ZS algorithm (Zhang and Suen, 1984) is used for this purpose.
- **Image Denoising:** Often a group of arcs and lines are added to reinforce the CAPTCHA test against breakage, so to eliminate this noise a group of denoising methods is used. Using filters, Gibbs and Hough transform and Wavelet transforms (Chen et al., 2017).

3.6.2 Segmentation

CAPTCHA attackers often resort to the segmentation before the recognition step to break it, segmentation is one of the most important supportive steps especially when CAPTCHA is supported by segmentation resistance (overlapping, hollow, etc.). However, find some methods that do not use segmentation in breaking the CAPTCHA (Chen et al., 2017; Zhang et al., 2019). So there are two categories:

Text-Based CAPTCHA Breaking Methods Based on Segmentation

These methods rely on segmentation as a basic step in breaking CAPTCHAs. A set of techniques and methods are used for this propose. The aim of it is to obtain individual characters, it varies from CAPTCHA to another according to its segmentation resistance. From these methods we mention:

- Methods based on single (individual) characters: segment a CAPTCHA image to individual characters using a combination of techniques (Zhang et al., 2019; Chen et al., 2017) based on:
  - *Character projection:* An effective method to segment non-overlapping characters. That by the character projection histogram (the projection can be vertical and horizontal)

  - *Character width:* A limited method that is only used when characters are of equal width and uniformly distributed.
  - *Character feature:* Segmentation based on the inside and outside features of characters, such as (dot (i, j), circle (a, b, d, e, g, o, p, q), cross (t, f)).
  - *Character contour:* The appropriate segmentation lines are determined based on analyzing geometric features of character contours.

- Methods based on character components: Instead of producing individual characters, these methods produce multiple character components based on:
  - *Character Structure:* based on the structural features of the characters, often relies on color filling to discover the shared character components.
  - *Filter:* Using Log-Gabor filters to extract character components, it is appropriate for a wide range of text-based CAPTCHAs.

Combination

In contrast to the individually segmented characters, which are directly recognized. Character components need to combined to recognize later and this depends on two techniques (Chen et al., 2017):

- *Combination methods based on redundancy:* This technique is used after using the filter in the segmentation. This is by combining the output from the filter to generate characters.
- *Combination methods Based on non-redundancy:* By using the boundaries of the overlapping area (shared character components) to create a complete character.

Text-Based CAPTCHA Breaking Methods Based on Non-segmentation

With the increase in the use of segmentation resistances in text-based CAPTCHA. They relied on methods that depend on direct recognition without need for segmentation by using a set of techniques, this is what we will discuss in next section.

3.6.3 Recognition

Character recognition is the most important stage in breaking a text-based CAPTCHA, and it is divided into these categories (recognition methods based on matching algorithms, recognition methods based on character feature, and recognition methods based on machine learning and deep learning). In this part, we will try to mention some works in each category.

Recognition Methods Based on Matching Algorithms

These algorithms depend on the similarity of the pixels between the characters, which is based on:

- *Template Matching*: Template matching algorithm is traverse scanning the search area within each pixel and regional matching calculation to find the optimal match point. In (Dai et al., 2013), and after the pre-processing and extraction of characters they depended on Template Matching Algorithm in the recognition with the aim of breaking 100 text-based CAPTCHAs collected randomly from a game website (which are multi-font formats and have adhesion and irregular tilt angle as shown in Figure 3.11) and they achieved a recognition rate of 93%.

Figure 3.11: CAPTCHAs of a game website (Dai et al., 2013)

- *The Shape Context*: In (Mori and Malik, 2003), the authors developed a method based on shape context matching that can identify words in the text-based CAPTCHA image. The first step is to find the locations of characters. After that, match them to extract candidate words and choose the most probable word. They broke EZGimpy and Gimpy (CAPTCHAs used by Yahoo, examples of them are shown in the Figure 3.12) and get a success rate of 92% and 33%, respectively, this method is robust to transform the image it also provides sufficient information about the image but they didn't take into account the rotation of characters.

(a) EZ-Gimpy (b) Gimpy

Figure 3.12: EZ-Gimpy and Gimpy CAPTCHAs (Mori and Malik, 2003)

Recognition Methods Based on Character Feature

Text-based CAPTCHAs designs differ in their composing characters, whether in structural or statistical features. So we find recognition methods based on:

- *Character Structural Feature:* These methods are based on the characteristics of characters such as (shape, number of loops, inflection point, and cross points). In (Gao et al., 2014), they attacked Yahoo!, Baidu CAPTCHA, and reCAPTCHA (a group of CAPTCHAs that use the principle to "connecting characters together (CCT)"), that by using a set of methods that depend on the cut head and tail (CHT), the guide lines and the loop principle as shown in Figure 3.13.

Figure 3.13: Guide lines and loop principle (Gao et al., 2014)

- *Character Statistical Feature:* Based on statistical characteristics of characters such as (pixel feature and contour feature) to break the CAPTCHAs. In (Yan and El Ahmad, 2007) after segmentation and for recognition, the authors use the number of pixels "pixel-count" for each letter (from A to Z) to recognize them and break the CAPTCHA. They achieved a success rate higher than 90%.

Recognition Methods Based on Machine Learning

CAPTCHA tests have always been supported by recognition resistances that make them unbreakable, especially with the standard methods mentioned above. Therefore, given the sophistication of machine learning algorithms, breaking CAPTCHA is one of the challenges that can be solved using machine learning by correctly classifying CAPTCHA characters (Chen et al., 2017). As we mentioned previously, recognizing the characters of CAPTCHA using machine learning is a classification problem, the most common methods used for this purpose are Support Vector Machine (SVM), k-nearest neighbor (KNN), and Artificial Neural Network (ANN).

In (Starostenko et al., 2015), the authors break reCAPTCHA 2011 and 2012 versions (shown in Figure 3.14), after the pre-possessing and segmentation steps. The recognition step was solved as a classification problem using Support Vector Machine (SVM), it is a supervised machine learning algorithm used in classification. SVM separates classes via a hyperplane by using a kernel function. They train 1000 reCAPTCHA images segmented, this model gets a success of 31% for reCAPTCHA 2011 version and 56.3% for reCAPTCHA 2012 version. The double characters (nn, vv, mm, etc.) and those are formed by thin double characters (il, li, ii, ll, it, ti, 11, etc.) was the reason to get this result. So they created additional classes for these combinations. The addition of these classes has significantly improved the precision of

the classification process. They got success of 82% for reCAPTCHA 2011 version and 95.5% for reCHAPTCHA 2012 version.

Figure 3.14: reCAPTCHA 2011 and 2012 versions (Starostenko et al., 2015)

In (Gao et al., 2016), the authors attack CAPTCHAs deployed by the top 10 most popular websites (Google, Microsoft, Yahoo!) using two models of recognition k-nearest neighbor (KNN) and convolutional neural network (CNN) in which KNN achieved higher success rates.

The Recognition step in (Baecher et al., 2010) was solved as a clustering problem by using K-Means. This model was used for three different data set (shown in Figure 3.15). Finanzagentur and Umweltpramie are data-set with 100 images, get a success rate of 70% and 68% respectively. Sparda Bank has 330 images and get a success rate of 87%.

(a) Umweltpramie (b) Finanzagentur (c) Sparda Bank

Figure 3.15: Attacking text-CAPTCHAs (Baecher et al., 2010)

(Bostik and Klecka, 2018), compared four classification algorithms to recognize the characters of CAPTCHA. ANN, SVM, KNN, and Decision tree. The CAPTCHA dataset that they used is generated by a PHP script. ANN achieved the best success rate of 100%, SVM 98.80%, KNN, and Decision tree get 98.99 %.

Through these works, we have seen that ML methods have given better results than the standard methods, but are still limited in extracting the features and segmentation that the recognition model is based on.

Recognition Methods Based on Deep Learning

We know that DL has achieved great success in simulating human ability and this facilitates the process of breaking the CAPTCHA. It was used in attempts to break text-based CAPTCHA and achieved great success compared to traditional methods. In this section, we

present two ways of breaking with DL. The first is similar to the previous techniques (standard and using ML) is to pass through the segmentation stage and recognition of characters separately. The second which is more practical, by use of the complete image of the CAPTCHA Without the need for segmentation or pre-processing.
Convolutional neural networks (CNNs) is most deep learning technique using to break CAPTCHAs. Where it is used in two ways, either to recognize characters after segmentation or to recognize directly (without segmentation).

Among those who used CNN after the segmentation step (Wang et al., 2017), where they trained CNN on images of the characters composing the CAPTCHA. The CAPTCHA image that they use is shown in Figure 3.16 (contains Chinese and English characters with Arabic numbers). They segment the CAPTCHA image to images of size 32x32 and rotate each character from -50° to 50° to get more data. Their CNN contains four convolution layers and two fully connected layers, the output layer contains 82 neurons (class) 10 for digits '0'-'9', 52 of for 'a'-'z' and 'A'-'Z', 10 of each for simplified and traditional Chinese digits. They achieved rate of 85.72%.

Figure 3.16: Examples from (Wang et al., 2017) dataset CAPTCHAs

In (Karthik and Recasens, 2015), the authors utilized tow way to break Microsoft CAPTCHAs. The first one is by using the template method with data set of 144 character images, they achieved with this method an individual character success rate of 60%, with an overall CAPTCHA success rate of 5,56%. The other by using a LeNet-5 CNN architecture (consists of three convolutional layers followed by three fully connected layers) and they added one extra layer to create a deeper structure. Their CNN network learned by character images after segmentation of 600 CAPTCHAs, each image was divided into six equally distributed segments without losing part of the target character, the recognition success was 57.05% for this method.

We also mention (Tang et al., 2018), after the preprocessing and segmentation, they propose an attack based on CNN in recognition to determine what each single character is, and they test their attack on the top 50 most popular text-based CAPTCHAs. CNN architecture is a LeNet-5 with an extra convolutional layer which gets a success rate of 10.1% to 90%. They also used three Chinese CAPTCHA as an example of CAPTCHA which is based on a large character set and they archived success rates of 93.0%, 32.2% and 28.6%. They considered their attack the beginning of the end of text-based CAPTCHAs.

We note that the use of CNN in the recognition step only is similar to the use of one of the traditional methods of ML. Despite its speed and good results, but using it to recognize the word directly without segment the CAPTCHA image is more typical.

In (Hu et al., 2018), generated a data-set of CAPTCHA images with 5 characters contain 71000 images (50000 for training set, 20000 for validation set and test set include 1000) the Figure 3.17 show some images that they generated. The network structure that they use depends on the advantages of VGG-Net (convolutional neural network developed by Oxford visual geometry group (Simonyan and Zisserman, 2014)) which deepens the convolutional layer and reduces the size of the convolution kernel. The output layer contains 310 neurons each 62 neurons predicts a character, they defined a bijection $\theta(x)$ Equation 3.1 that maps a character to numbers. They achieved recognition accuracy reached to 96.5%.

$$\theta(x) = \begin{cases} 0 \sim 9, x =' 0' \sim' 9' \\ 10 \sim 35, x =' a' \sim' z' \\ 36 \sim 61, x =' A' \sim' Z' \end{cases} \tag{3.1}$$

Figure 3.17: CAPTCHAs generated by (Hu et al., 2018)

(Wang et al., 2019) also utilized CNN without the segmentation step they compare three CNN architecture (ResNet50, DenseNet-121, and DFCR that they built), they applied this architecture with three dataset (shown in Figure 3.18). 1st had a five-character CAPTCHA composed of digits and English letters, 2nd composed the same character but four-character with noise, the last dataset composed of Chinese characters, and one character rotated by 90°. Their result showed that DFCR has better recognition accuracy (reached to 99%).

(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 3.18: Examples from (Wang et al., 2019) dataset

The most of the attacks on text-CAPTCHA were limited to a specific type or group that shares the same protection features. In (Ye et al., 2018) they provided a CAPTCHA attack base on the Generative Adversarial Network (GAN). The first step of their approach is to generate a large number of synthetic CAPTCHAs together with their labels (using a data set of real CAPTCHAs contain 500 images), the next step is training the solver (CAPTCHA breaker) that contain LeNet-5 CNN architecture by the set auto generated. They evaluate their approach on 33 text CAPTCHA schemes at the same time they compared the results with (Bursztein et al., 2011) and (George et al., 2017) their results were even better, and it gets 100% success rate in some types.

Another DL technique used in (Wang et al., 2020), they combined residual network (ResNet) and recurrent neural network (RNN 2.5.2) to recognize CAPTCHA characters. ResNet for extracting the feature of CAPTCHA image and LSTM (2.5.2) converts the feature extracted into a single text string. And to take the text strings of different lengths without segmentation they utilized a spatial attention mechanism from (Wojna et al., 2017). They used also transfer learning to reducing training time. This attack achieved a success rate over 90%.

We note that with the development of deep learning techniques, it became possible to break a text-based CAPTCHAs without segmentation. In addition the possibility of breaking any type of text-CAPTCHA (difference in the number of characters and distortion) with a single model.

3.6.4 Post processing

Some result of the recognition step need post processing to get the final results (Chen et al., 2017), there are the post-processing methods selection or rejection based.

Post-processing Methods Based on Selection

Often the previous steps result in extra individual characters and this requires optimization to determine the final result, this strategy includes the local and the global optimization to select the final recognition. Local optimization depends on the recognition confidence optimally of an individual character, while the global optimization strives for the best results for all characters in an image.

Post-processing Methods Based on Rejection

This strategy determines whether a candidate character recognition results should be rejected or no, depending on multiple features such as confidence, string length, character spaces, and the first and the last character of a string. This strategy is also key to ensuring the high reliability of CAPTCHA recognition.

3.7 Conclusion

In this chapter, we first provided an overview of CAPTCHA types. Later we focused on text-based CAPTCHA, its generation, and the resistance mechanisms added to make it unbreakable. In addition to the process and state-of-the-art in breaking this type from the standard methods to the deep learning techniques that have achieved high rates in this field.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENT

4.1 Introduction

The aim of this chapter is an experimental study of an attempt to break a text-based CAPTCHA based on deep learning using a convolutional recurrent neural network. the dataset is generated with python CAPTCHA library where each image contain between four to seven characters from the 26 upper case English alphabets.

After going through the segmentation of CAPTCHA image, convolutional neural networks achieved good results in recognizing characters and breaking text-based CAPTCHA test. But in our attempt, we try to break text-based CAPTCHA without the need for the segmentation step using a model based on an End-to-End CRNN-CTC network used for handwriting and text recognition.

We first introduce some details of the experiment: the dataset, the setup environment, followed by the network architecture that we rely on. We conclude by discussing the obtained result.

4.2 Implementation setup

In this section, we will first introduce the dataset used in our experiment, then the environment setups that we worked on.

4.2.1 Dataset

We use python CAPTCHA library to generate our dataset. This library can generate audio and image CAPTCHAs. It creates an image that contains a number of characters with different colors and a little warp, rotation, overlapping sometimes and noise such as a number of dots, arcs, and lines. That to make it simulate an unbreakable real text-based CAPTCHAs, resistant to attacks.

- We first create a chart that contains the characters we would like to insert in the CAPTCHA image. In this case, the upper case English alphabet is used as the charset.
- The CAPTCHA image is set to 128 pixels width and 32 pixels height.
- Each image contains between four to seven characters string randomly selected from the charset.

As training dataset needs a large amount of data to construct the recognition model, more then 8000 is set to the size of the training dataset and the size of the test dataset is set to 2672. Examples from our text-based CAPTCHA dataset are shown in Figure 4.1. The string that the CAPTCHA contains is used as the image name, and the label for each CAPTCHA is the image name without suffix (.jpg).

(a) SBGZH.jpg (b) ZNFP.jpg (c) YLDKVSN.jpg

Figure 4.1: Examples from our dataset.

Grayscale and normalization are applied to the CAPTCHA image dataset as preparation to use it as input (X-train/X-test) to the network model. On the other hand, the labels are encoded to use it as (Y-train/Y-test) to the network model.

4.2.2 Environment

The implementation environment used in this experimentation is **Google Colaboratory**[1] who offers a free Jupyter notebook environment, access powerful computing resources and supports free GPU training acceleration. Google Colab allows us to write and execute arbitrary python code through the browser and is especially well suited to machine learning and data analysis.

For this experimentation, we use:

Python as a programming language. It is a high-level programming language that contains libraries that facilitate the construction of machine learning and deep learning models.

GPU using GPU (Graphics Processing Unit) that is available in Google Colab to reduce the time spent for calculating and speed up the model learning.

Libraries to build, train, and test our model we use the following libraries:

- TensorFlow 2.0[2]: is an open source platform for machine learning and deep learning developed by Google. It has a comprehensive list of tools and libraries that help to build machine learning and deep learning applications.
- Keras[3]: is the high-level API of TensorFlow 2.0, an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions.
- NumPy[4]: is a scientific computing library. It provides comprehensive mathematical functions with high level syntax that makes it accessible and versatile.
- OpenCV[5]: Open Source Computer Vision Library is library used for machine perception and computer vision applications.
- Matplotlib[6]: is a comprehensive library for creating static, animated, and interactive visualizations in Python.

[1] `https://research.google.com/colaboratory`
[2] `www.tensorflow.org`
[3] `https://keras.io`
[4] `www.numpy.org`
[5] `www.opencv.or`
[6] `https://matplotlib.org/`

4.3 Network architecture

According to the following works: text line recognition in natural scenes (Tong et al., 2019) handwriting text recognition (Jaramillo et al., 2018) and text recognition (Baek et al., 2019), as they depend on End-to-End CRNN-CTC Network as shown in Figure 4.2 in their experiments and they achieved good results. This encouraged us to use it to break the text-based CAPTCHA.

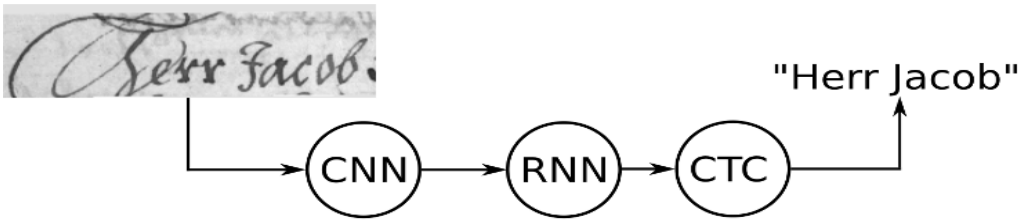


Figure 4.2: CRNN-CTC Network

The network structure combining CNN, RNN, and Connectionist Temporal Classification (CTC). As shown in Figure 4.3, the network can be divided into three parts from left to right: convolutional layers, recurrent layers, and transcription layer. The beginning of the entire network is the convolutional neural network layers. The CNNs are used to automatically extract the feature sequences for each input image. The RNNs (Bidirectional LSTM) then predict each vector of the feature sequence extracted from the CNN. The last part of the network is the transcription layer, which is responsible for converting the predictions from RNNs to real labels.
Figure 4.6 shows the flowchart of the model.

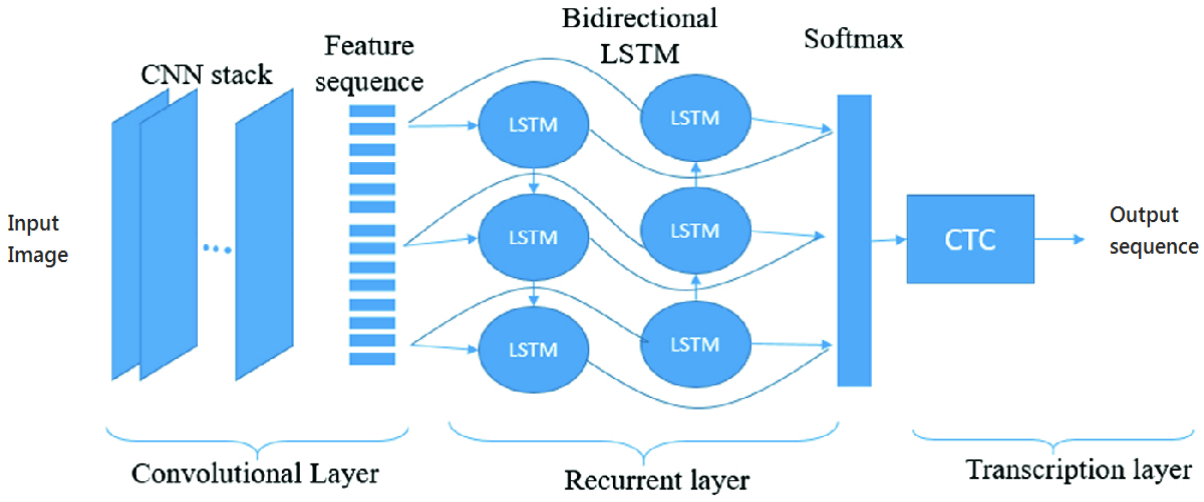


Figure 4.3: The CRNN-CTC Network parts

Here is a description of the constituent layers of the network we have relied on :

- **Convolutional layers**: Features extraction part, its structure is a normal convolutional neural network without the fully connected layer, it is composed of:
  - The text-based CAPTCHA image of size (32×128) in grayscale as an input.
  - 6 convolutional layers all with a kernel of 3×3 and 1×1 stride, the numbers of filters are 16, 32, 64, 64, 64 and 64, respectively, and one extra convolutional layer with a kernel of 2×2 and 64 filters. ReLU is the activation function.

    Conv2D(16, (3,3), activation = 'relu', padding='same')
  - A 2×2 and 2×1 max-pooling is also applied, with the aim of reducing the size of the extracted features.

    MaxPool2D(pool_size=(2, 2), strides=2)
  - Batch Normalization is also used after both convolutional layers before the last one to stabilize the learning process. and to avoid the overfitting dropout is used.

    BatchNormalization()
    Dropout(0.3)

After this first convolutional stage, the feature maps are transformed from the 3D size (width × height × depth) into 2D of size (width × (height × depth)) vectors. Finally, a sequence of feature vectors is extracted from the feature maps produced by the CNNs as shown in Figure 4.4, which is the input for the recurrent layers.

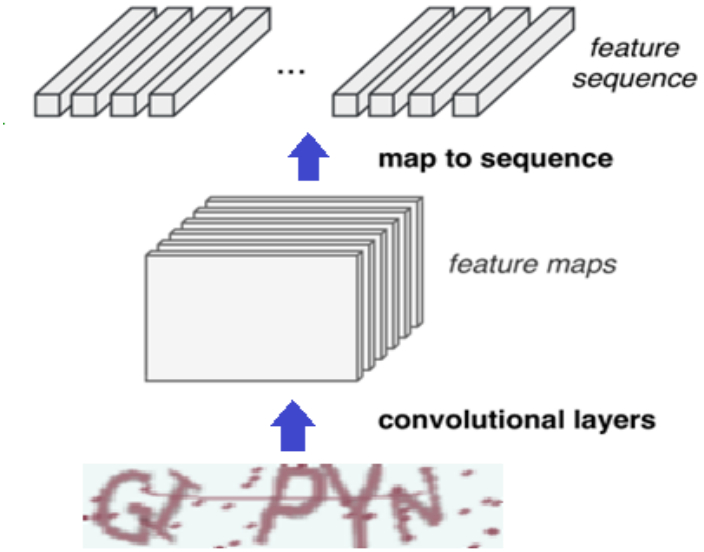


Figure 4.4: Convolutional layer

- **Recurrent layers**: After feature extraction part, two Bidirectional LSTM recurrent layers of 128 units are applied, which predict a label distribution P_t for each vector x_t in the feature sequence $x = x_1...x_t$.

  Bidirectional(CuDNNLSTM(128, return_sequences=True))

  Followed by a full-connect layer as shown in Figure 4.5 . The final results, which take the form of probability distributions, are obtained through a softmax layer. $L + 1$ labels used, where L is the number of characters that appear in our database (26 characters). The additional dimension has needed for the blank symbol of the CTC. Figures 4.7 and 4.8 depict this bloc.

  Dense(len(char_list) + 1, activation = 'softmax')

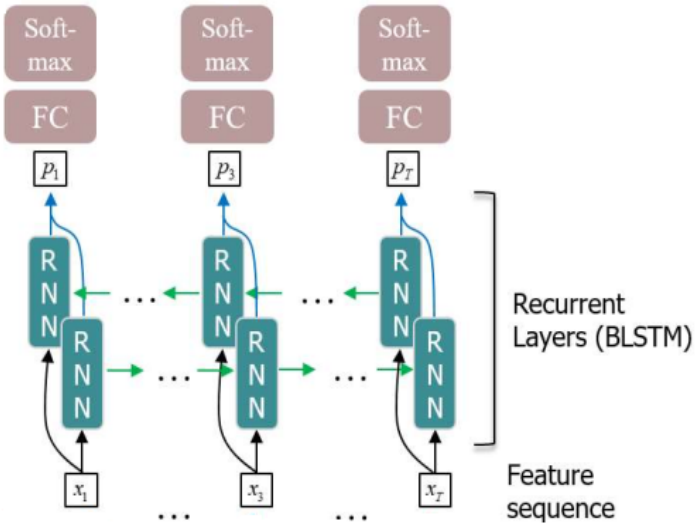


Figure 4.5: Recurrent layer

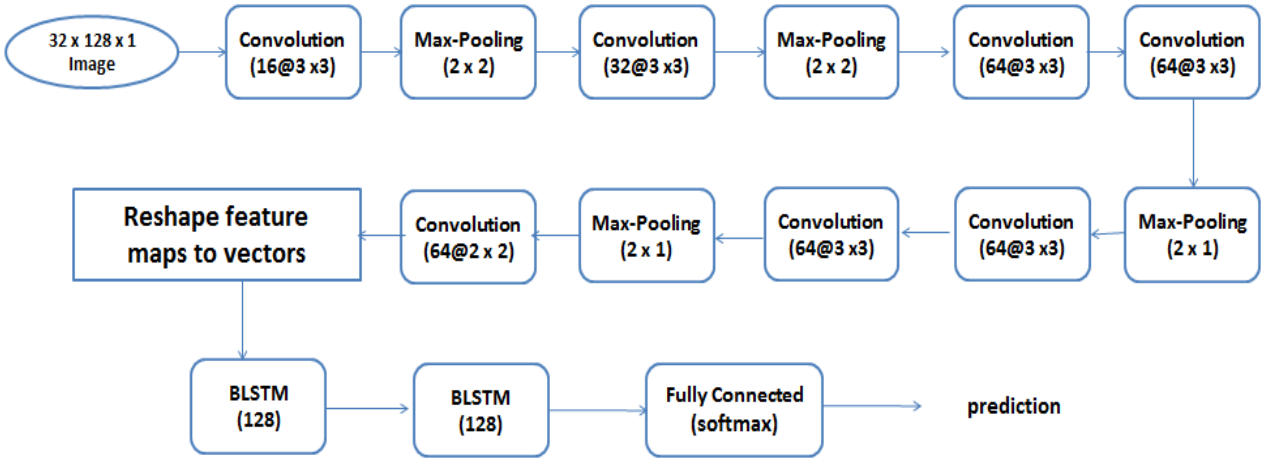
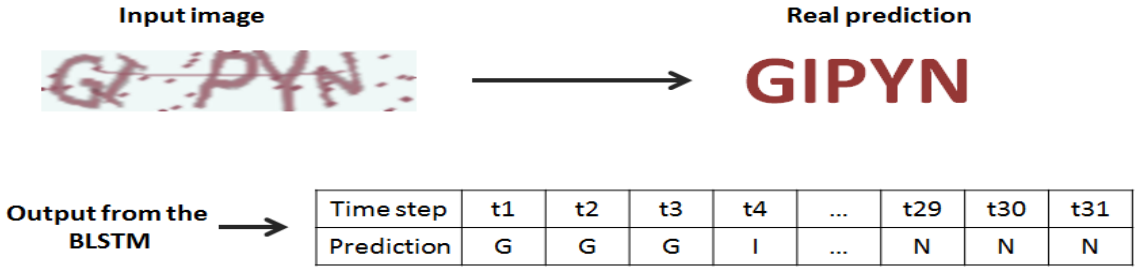


Figure 4.6: Flow chart of the CRNN

- **Transcription layer**: Transcription use to convert the predictions (of each prediction) made by the Bidirectional LSTM into a label sequence. This by the CTC (Connectionist Temporal Classification) loss function. *"The CTC loss function aims to model the conditional probability of label sequences given probability distribution of each predicted label."*(Feng et al., 2019).

AS shown in Figure 4.7, when the character takes more than one time-step, the result will be in duplicate characters. CTC solves this problem.



Time step	t1	t2	t3	t4	...	t29	t30	t31
Prediction	G	G	G	I	...	N	N	N

Figure 4.7: Recurrent layers output

CTC works on three major concepts :

- **Encoding**: CTC merges all the repeating characters into a single character. In the case that the real label contains a repeating character, CTC introduces a pseudo-character called blank denoted as "-". While encoding, if a character repeats, then a blank is placed between the characters in the output label sequence.
- **Loss calculation**: by summing over the probability of all possible alignments (paths) between the input and the label.
- **Decoding**: After the model trained, the best path algorithm uses in which calculate the best path by considering the character with max probability at every time-step. This step involves removing blanks and duplicate characters, which results in the actual label. An example is shown in Figure 4.8.

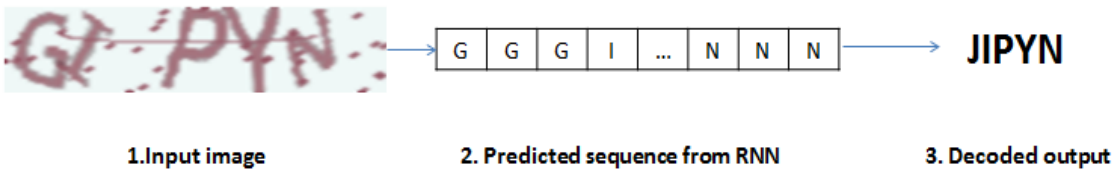
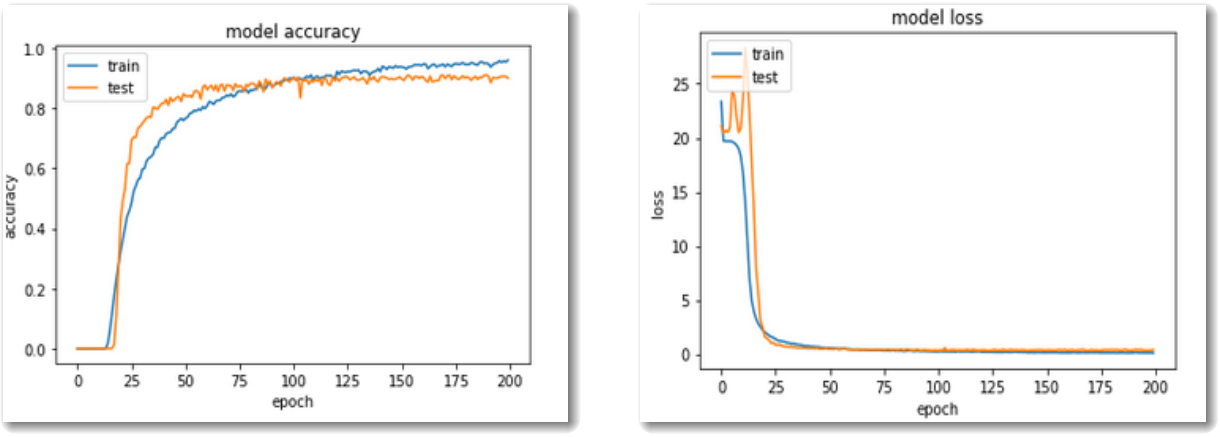


Figure 4.8: Transcription layer

4.4 Results and Discussion

In order to show the results obtained from our model, we explain below the results in terms of the accuracy and the loss of our model in breaking text-based CAPTCHA to get the real label.

After the model training, we note the following remarks from the curves of accuracy and loss in Figures 4.9a and 4.9b, which represent the development of accuracy and loss results, according to the epochs. Where the accuracy of the model in training and test set is 0.96 and 0.91, respectively. And the loss is 0.11 in training, 0.28 in test. The table 4.1 represents some text based CAPTCHA breaking using our model (real label vs our model prediction).



(a) Model accuracy (b) Model loss

Figure 4.9: Model accuracy and loss

Through the results obtained by our trained model, which achieved an accuracy of 91% in the test dataset. It is considered a good result and threatens the security of the text-based CAPTCHA test.

Although we did not reach the results of the previous works. However, we have achieved some goals such as:

- We didn't need to segment the text-based CAPTCHA image and recognize each character separately. The model uses the complete CAPTCHA image as an input and that reduces the breaking process complexity by skipping costly stages, such as preprocessing and segmentation.
- We were not limited to a specific CAPTCHA text length. Our model is able to break a CAPTCHA with different text lengths.

- Our model based on an End-to-End CRNN-CTC Network used for handwriting and text recognition, and it gave good results in breaking the text-based CAPTCHA taking into account the differences between the normal text and the CAPTCHA text, which is often distorted, twisted and overlapping in each other.
- We can also improve our model results by augment the training dataset and training it for a longer time.

This makes the text-based CAPTCHA vulnerable to deep learning breaking methods and useless to use in websites to protect them from computer malicious programs.

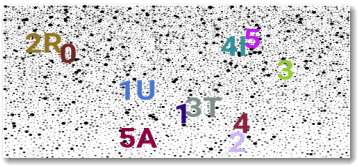
Table 4.1: A sample of real vs predicted labels

CAPTCHA	Real label	Predicted label
	VRXIX	VRXIX
	TXNCOSZ	TXNCOSZ
	CONP	CONP
	NTRACDY	VTRACDY

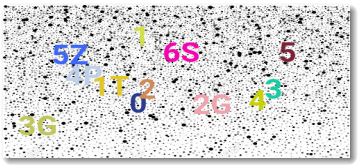
4.5 Recommendation

From what we have presented, we have noticed that there is no difficulty with deep learning techniques to break any text-based CAPTCHA. However, we noticed that most of the existing text-based CAPTCHAs depend on sequential characters, and that facilitates the process of breaking them. That is what we will rely on, by making the characters that make up the CAPTCHA text not to be sequenced.

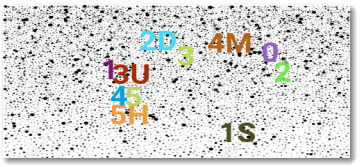
As a contribution, we suggest a model that may make it difficult for the machine to pass than a human. The idea is simple, we create an image containing characters of each in a random place accompanied by an index number. The user must enter the characters in the correct order. For misleading, a blurry background is used and single random numbers are added to the image. The Figure 4.10 below shows examples of our proposed CAPTCHA.



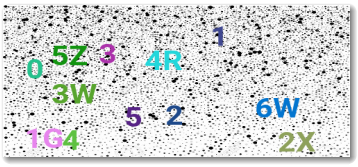
(a) URTLA



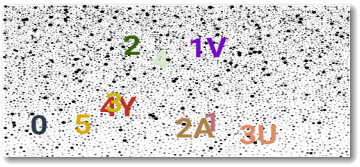
(b) TGGPZS



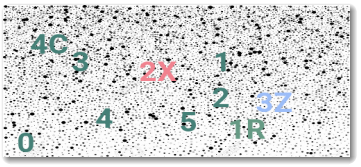
(c) SDUMH



(d) GXWRZW



(e) VAUY



(f) RXZC

Figure 4.10: Examples from our proposed CAPTCHA

4.6 Conclusion

In this chapter, we conducted an experiment to break text-based CAPTCHA using End-to-End CRNN-CTC network and CAPTCHA images datset generated by python. The results obtained show that it is not difficult to attack and break the text-based CAPTCHA through deep learning techniques, even without the need to segment the CAPTCHA image.

CHAPTER 5

CONCLUSION

The purpose of this work is to break text-based CAPTCHA using deep learning techniques. We start with an overview of machine learning and deep learning with its convolutional and recurrent neural network. Then we introduced a generality of CAPTCHA test and their types. Thereafter, we presented the state-of-the-art of breaking the text-based CAPTCHA, starting by matching algorithms until the machine learning and deep learning techniques.

Subsequently, we conducted an experimental study to break text-based CAPTCHA using a model based on an End-to-End CRNN-CTC network used for handwriting and text recognition, and text-based CAPTCHA images dataset generated by python. The result of the experiments reveals that our model is able to break the text-based CAPTCHA generated whit accuracy of 91% which can be improved by augmenting the dataset and training it for a longer time.

In the perspective of ameliorating the text-based CAPTCHA breaking we propose to improving the CRNN-CTC network parts such as using ResNet or VVG-Net in the convolutional layer and GRU in the recurrent layer, also to use Generative Adversarial Network (GAN) to generate more CAPTCHAs image datasets.
To improve the CAPTCHA test and making it unbreakable there are several types discussed as an alternative to the text-based CAPTCHAs such as image, audio, and video based-CAPTCHAs. We also propose to investigate the Arabic CAPTCHA, mathematical/geometric CAPTCHA requires also more studies and evaluate extensively the proposed CAPTCHA.

However, breaking CAPTCHA remains an active field due to advances in deep learning techniques, and generation an unbreakable CAPTCHA is still a challenge.

"A step backward for CAPTCHA is still a step forward for AI - every defeat is also a victory" (Luis von Ahn 2006).

BIBLIOGRAPHY

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning - A Textbook*. Springer.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE.

Algwil, A., Ciresan, D., Liu, B., and Yan, J. (2016). A security analysis of automated chinese turing tests. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 520–532.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3):292.

Azad, S. and Jain, K. (2013). Captcha: Attacks and weaknesses against ocr technology. *Global Journal of Computer Science and Technology*.

Baecher, P., Fischlin, M. G. L., Langenberg, R., Lützow, M., and Schröder, D. (2010). Captchas: the good, the bad, and the ugly. In Freiling, F. C., editor, *Sicherheit 2010. Sicherheit, Schutz und Zuverlässigkeit*, pages 353–365, Bonn. Gesellschaft für Informatik e.V.

Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S. J., and Lee, H. (2019). What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723.

Baird, H. S., Coates, A. L., and Fateman, R. J. (2003). Pessimalprint: a reverse turing test. *International Journal on Document Analysis and Recognition*, 5(2-3):158–163.

Baird, H. S., Moll, M. A., and Wang, S.-Y. (2005). Scattertype: A legible but hard-to-segment captcha. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 935–939. IEEE.

Baird, H. S. and Popat, K. (2002). Human interactive proofs and document image analysis. In *International Workshop on Document Analysis Systems*, pages 507–518. Springer.

Banday, M. T. and Shah, N. A. (2011). A study of captchas for securing web services. *arXiv preprint arXiv:1112.5605*.

Ben Boudaoud, L., Sider, A., and Tari, A. (2015). A new thinning algorithm for binary images. In *2015 3rd International Conference on Control, Engineering Information Technology (CEIT), Tlemcen, Algeria*, pages 1–6.

Biederman, I. (1987). Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115.

Bostik, O. and Klecka, J. (2018). Recognition of captcha characters by supervised machine learning algorithms. *IFAC-PapersOnLine*, 51(6):208–213.

Brabazon, A., O'Neill, M., and McGarraghy, S. (2015). *Natural computing algorithms*. Springer.

Brodić, D. and Amelio, A. (2019). *The CAPTCHA: Perspectives and Challenges: Perspectives and Challenges in Artificial Intelligence*, volume 162. Springer Nature.

Bursztein, E., Martin, M., and Mitchell, J. (2011). Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 125–138.

Chan, T.-Y. (2003). Using a test-to-speech synthesizer to generate a reverse turing test. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 226–232. IEEE.

Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. (2005). Building segmentation based human-friendly human interaction proofs (hips). In *International Workshop on Human Interactive Proofs*, pages 1–26. Springer.

Chellapilla, K. and Simard, P. Y. (2005). Using machine learning to break visual human interaction proofs (hips). In *Advances in neural information processing systems*, pages 265–272.

Chen, F. (2016). Ai, deep learning, and machine learning: A primer. *presentation, Andreessen Horowitz, June*, 10.

Chen, J., Luo, X., Guo, Y., Zhang, Y., and Gong, D. (2017). A survey on breaking technique of text-based captcha. *Security and Communication Networks*, 2017.

Chew, M. and Tygar, J. D. (2004). Image recognition captchas. In *International Conference on Information Security*, pages 268–279. Springer.

Chowdhury, A. and Vig, L. (2018). An efficient end-to-end neural model for handwritten text recognition. *arXiv preprint arXiv:1807.07965*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., and dos Reis Alves, S. F. (2017). Artificial neural networks. *Cham: Springer International Publishing*, page 39.

Dai, F., Gao, H., and Liu, D. (2013). Breaking captchas with second template matching and bp neural network algorithms. *JIPM*, 4(3):126–133.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387.

Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2007). Asirra: a captcha that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security*, volume 7, pages 366–374.

Feng, X., Yao, H., and Zhang, S. (2019). Focal ctc loss for chinese optical character recognition on unbalanced datasets. *Complexity*, 2019.

Fiot, J.-B. and Paucher, R. (2009). The captchacker project. *Ecole Centrale Paris*.

Flasiński, M. (2016). *Introduction to artificial intelligence*. Springer.

Fukushima, K. (1989). Analysis of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2(6):413–420.

Gao, H., Wang, W., Fan, Y., Qi, J., and Liu, X. (2014). The robustness of" connecting characters together" captchas. *J. Inf. Sci. Eng.*, 30(2):347–369.

Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X., and Li, J. (2016). A simple generic attack on text captchas. the internet society, `http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/simple-generic-attack-text-captchas.pdf`.

Gao, H., Yao, D., Liu, H., Liu, X., and Wang, L. (2010). A novel image based captcha using jigsaw puzzle. In *2010 13th IEEE International Conference on Computational Science and Engineering*, pages 351–356. IEEE.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., Wang, H., et al. (2017). A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368):eaag2612.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

Guo, T., Dong, J., Li, H., and Gao, Y. (2017). Simple convolutional neural network on image classification. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, pages 721–724. IEEE.

Gurney, K. (1997). *An introduction to neural networks*. CRC press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G. (2018). Deep learning—a technology with the potential to transform health care. *Jama*, 320(11):1101–1102.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Hu, Y., Chen, L., and Cheng, J. (2018). A captcha recognition technology based on deep learn-

ing. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 617–620. IEEE.

Ivakhnenko, A. and Ivakhnenko, G. (1995). The review of problems solvable by algorithms of the group method of data handling (gmdh). *Pattern Recognition And Image Analysis C/C Of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 5:527–535.

Jaramillo, J. C. A., Murillo-Fuentes, J. J., and Olmos, P. M. (2018). Boosting handwriting text recognition in small databases with transfer learning. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 429–434. IEEE.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier.

Karthik, C.-P. and Recasens, R. A. (2015). Breaking microsoft's captcha. *Technical report*.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954.

Khan, B., Alghathbar, K., Khan, M. K., AlKelabi, A. M., and Alajaji, A. (2013). Cyber security using arabic captcha scheme. *Int. Arab J. Inf. Technol.*, 10(1):76–84.

Khan, B., Alghathbar, K. S., Khan, M. K., AlKelabi, A. M., and AlAjaji, A. (2010). Using arabic captcha for cyber security. In *Security Technology, Disaster Recovery and Business Continuity*, pages 8–17. Springer.

Kim, J., Kim, J., Thu, H. L. T., and Kim, H. (2016). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)*, pages 1–5. IEEE.

Kluever, K. A. (2008). Evaluating the usability and security of a video captcha.

Koffka, K. (1935). Principles of gestalt psychology, 481-493.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumar, N. S. and Supriya, L. (2019). Survey on text error detection using deep learning.

Lancashire, L. J., Lemetre, C., and Ball, G. R. (2009). An introduction to artificial neural networks in bioinformatics—application to complex microarray and mass spectrometry datasets in cancer studies. *Briefings in bioinformatics*, 10(3):315–329.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE.

Li, S., Shah, S. A. H., Khan, M. A. U., Khayam, S. A., Sadeghi, A.-R., and Schmitz, R. (2010). Breaking e-banking captchas. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 171–180.

Lillibridge, M. D., Abadi, M., Bharat, K., and Broder, A. Z. (2001). Method for selectively restricting access to computer systems. US Patent 6,195,698.

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Liu, X., Deng, Y., Sun, Y., and Zhou, Y. (2018). Multi-digit recognition with convolutional neural network and long short-term memory. In *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1187–1192. IEEE.

M. Blum, L. A. v. A. and Langford, J. (2000). The captcha project, "completely automatic public turing test to tell computers and humans apart" dept. of computer science, carnegie–mellon univ, and personal communications. `http://www.captcha.net`.

Machinery, C. (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mori, G. and Malik, J. (2003). Recognizing objects in adversarial clutter: breaking a visual captcha. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I.

Moy, G., Jones, N., Harkless, C., and Potter, R. (2004). Distortion estimation techniques in solving visual captchas. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II.

Naor, M. (1996). Verification of a human in the loop or identification via the turing test. *Unpublished draft from http://www. wisdom. weizmann. ac. il/˜ naor/PAPERS/human abs. html*.

NUNES, I. and DA SILVA, H. S. (2018). *Artificial neural networks: a practical course*. Springer.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., and Iyengar, S. (2019). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):92.

Rao, M. and Singh, N. (2012). Random handwritten captcha: Web security with a difference. *International Journal of Information Technology and Computer ScienceIJITCS*, 4(9):53–58.

Rawat, W. and Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.

Rosenberg, C. R. and Sejnowski, T. J. (1986). The spacing effect on nettalk, a massively parallel network. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 72–89. Citeseer.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory.

Roshanbin, N. and Miller, J. (2013). A survey and analysis of current captcha approaches. *J. Web Eng.*, 12(1&2):1–40.

Rui, Y. and Liu, Z. (2004). Artifacial: Automated reverse turing test using facial features. *Multimedia Systems*, 9(6):493–502.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,.

Rusu, A. and Govindaraju, V. (2005). Visual captcha with handwritten image analysis. In *International Workshop on Human Interactive Proofs*, pages 42–52. Springer.

Rusu, A., Thomas, A., and Govindaraju, V. (2010). Generation and use of handwritten captchas. *International Journal on Document Analysis and Recognition (IJDAR)*, 13(1):49–64.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Sauvola, J. and Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern recognition*, 33(2):225–236.

Shanmuganathan, S. and Samarasinghe, S. (2016). *artificial neural network modiling*. Springer Publishing Company, Incorporated, 1st edition.

Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., and Stanley, M. (2017). A brief survey of machine learning methods and their sensor and iot applications. In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–8. IEEE.

Shi, B., Bai, X., and Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304.

Shirali-Shahreza, M. and Shirali-Shahreza, S. (2007a). Collage captcha. In *2007 9th International Symposium on Signal Processing and Its Applications*, pages 1–4. IEEE.

Shirali-Shahreza, M. H. and Shirali-Shahreza, M. (2006). Persian/arabic baffletext captcha. *J. UCS*, 12(12):1783–1796.

Shirali-Shahreza, M. H. and Shirali-Shahreza, M. (2007b). Localized captcha for illiterate people. In *2007 International Conference on Intelligent and Advanced Systems*, pages 675–679. IEEE.

Shirali-Shahreza, S. and Shirali-Shahreza, M. H. (2008). Bibliography of works done on captcha. In *2008 3rd International Conference on Intelligent System and Knowledge Engineering*, volume 1, pages 205–210. IEEE.

Sibi, P., Jones, S. A., and Siddarth, P. (2013). Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology*, 47(3):1264–1268.

Silva, I. d. et al. (2017). Artificial neural networks: A practical course. [sl]: Springer international publishing, 2017. Technical report, ISBN 978-3-319-43162-8,978-3-319-43161-1. Citado 3 vezes nas.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, V. P. and Pal, P. (2014). Survey of different types of captcha. *International Journal of Computer Science and Information Technologies*, 5(2):2242–2245.

Sivakorn, S., Polakis, J., and Keromytis, A. D. (2016). I'm not a human: Breaking the google recaptcha. *Black Hat*, pages 1–12.

Stansbury, D. (2014). A gentle introduction to artificial neural networks. *The Clever Machine*.

Stark, F., Hazırbas, C., Triebel, R., and Cremers, D. (2015). Captcha recognition with active deep learning. In *Workshop new challenges in neural computation*, volume 2015, page 94. Citeseer.

Starostenko, O., Cruz-Perez, C., Uceda-Ponga, F., and Alarcon-Aquino, V. (2015). Breaking text-based captchas with variable word and character orientation. *Pattern Recognition*, 48(4):1101–1112.

Su, X., Xu, H., Kang, Y., Hao, X., Gao, G., and Zhang, Y. (2019). Improving text image resolution using a deep generative adversarial network for optical character recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1193–1199. IEEE.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Takenaka, F., Hashimoto, T., and Hongo, T. (2017). *Artificial Neural Networks A Practical Course*. Springer.

Tang, M., Gao, H., Zhang, Y., Liu, Y., Zhang, P., and Wang, P. (2018). Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. *IEEE Transactions on Information Forensics and Security*, 13(10):2522–2537.

Tong, G., Li, Y., Gao, H., Chen, H., Wang, H., and Yang, X. (2019). Ma-crnn: a multi-scale attention crnn for chinese text line recognition in natural scenes. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1–12.

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer.

Von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 294–311. Springer.

Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066.

Wang, J., Qin, J. H., Xiang, X. Y., Tan, Y., and Pan, N. (2019). Captcha recognition based on deep convolutional neural network. *Math. Biosci. Eng.*, 16(5):5851–5861.

Wang, P., Gao, H., Shi, Z., Yuan, Z., and Hu, J. (2020). Simple and easy: Transfer learning-based attacks to text captcha. *IEEE Access*, 8:59044–59058.

Wang, T. and Bøegh, J. (2013). Multi-layer captcha based on chinese character deformation. In *International Conference on Trustworthy Computing and Services*, pages 205–211. Springer.

Wang, Y., Huang, Y., Zheng, W., Zhou, Z., Liu, D., and Lu, M. (2017). Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based captcha. In *2017 IEEE International Conference on Industrial Technology (ICIT)*, pages 980–985. IEEE.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

Weng, H., Zhao, B., Ji, S., Chen, J., Wang, T., He, Q., and Beyah, R. (2019). Towards understanding the security of modern image captchas and underground captcha-solving services. *Big Data Mining and Analytics*, 2(2):118–144.

Wojna, Z., Gorban, A. N., Lee, D., Murphy, K., Yu, Q., Li, Y., and Ibarz, J. (2017). Attention-based extraction of structured information from street view imagery. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 844–850.

Ximenes, P., dos Santos, A., Fernandez, M., and Celestino, J. (2006). A captcha in the text domain. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 605–615. Springer.

Yadava, P., Sahu, C., and Shukla, S. (2011). Time-variant captcha: generating strong captcha security by reducing time to automated computer programs. *Journal of Emerging Trends in Computing and Information Sciences*, 2(12):701–704.

Yalamanchili, S. and Rao, M. K. (2011). A framework for devanagari script-based captcha. *arXiv preprint arXiv:1109.0132*.

Yamamoto, T., Tygar, J. D., and Nishigaki, M. (2010). Captcha using strangeness in machine translation. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 430–437. IEEE.

Yan, J. and El Ahmad, A. S. (2007). Breaking visual captchas with naive pattern recognition algorithms. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 279–291. IEEE.

Yan, J. and El Ahmad, A. S. (2007). Breaking visual captchas with naive pattern recognition

algorithms. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pages 279–291.

Yan, J. and El Ahmad, A. S. (2008). A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 543–554.

Ye, G., Tang, Z., Fang, D., Zhu, Z., Feng, Y., Xu, P., Chen, X., and Wang, Z. (2018). Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 332–348.

Zhang, T. and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239.

Zhang, W. (2010). Zhang's captcha architecture based on intelligent interaction via ria. In *2010 2nd International Conference on Computer Engineering and Technology*, volume 6, pages V6–57. IEEE.

Zhang, Y., Gao, H., Pei, G., Luo, S., Chang, G., and Cheng, N. (2019). A survey of research on captcha designing and breaking techniques. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 75–84. IEEE.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237.