: N° d'ordre
: N° de série

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES
Computer Science Department

Thesis
submitted in partial fulfilment of the requirements for the degree of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**
Option: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Presented by :

- **Sakhri Amira**
- **Kadir Imane**

Title

Transforming handwriting and calligraphy into fonts

Defended on June 16th 2022

Examination Commitee :

M.	……	MCA	Chair
M.	……	MAA	Examinator
M.	Chourouk Guettas	MAA	Supervisor

Academic year: 2021-2022

الإهداء

الحمد لله الذي وفقني في مسيرتي هذه لإتمام مذكرتي فلك الحمد و الشكر كثرا مباركا .

اهدي ثمرة جهدي إلى سكني و راحتي **والدي الغالي** ، و الى من بسمتها غايتي و ما تحت قدميها جنتي **أمي الحبيبة** .

و إلى مصدر قوتي و نجاحي أختي"**إيناس** و زوجها(**عماد**)و كتاكيتها (**لمار أسينات، ميرال ترنيم ، ميار سكينة الرحمان ، وليد أمير**)"و إلى من أنارت دربي بنصائحها "**انتصار**" ، و أخيرا إلى حبيب قلبي و سندي أخي "**محمد الأمين**".

كما لا أنسى مصدر الصبر و الإجتهاد "**فتحي**".

و إلى كل العائلة الكريمة ، و زملاء الدراسة متمنية لهم التوفيق.

سخري أميرة.

الإهداء

الحمد لله وكفى والصلاة والسلام على الحبيب المصطفى واهله

اما بعد، الحمد لله الذي وفقنا لتثمين هذه الخطوة في مسيرتنا الدراسية بمذكرتنا هذه ثمرة الجهاد والنجاح بفضله تعالى

مهداة:

الى كل من علمني حرفا في هذه الدنيا الفانية.

الى سندي و ملجئي الامن ،داعمي و مشجعي الدائم اليك **والدي العزيز**.

الى **والدتي** "المثل والقدوة" من ترعرعت الروح بفضلها، ولا نطمع الا برضاها.

-الى كل من هم عزوتي وبهم تكتمل فرحتي اخوتي (**سليمة ، محمد تجاني**)

الى الروح التي عانقت روحي (**رمضان**)

الى كل قسم **الاعلام الالي** وجميع دفعة **2022 جامعة الشهيد حمة لخضر ،الوادي**،والشكر الجزيل الى الاستاذة المشرفه

(**قطاس شروق**)التي رافقتنا طيلة هذا البحث.

قادير إيمان

Abstract

Recognition and generation of written texts in cursive characters, are still a problem in their cursive form due to the nature of English Language connected characters in handwriting. In this thesis, we propose an hybrid RNN approach with the objective of creating an application capable of learning patterns of the user's handwriting and generating similar writing styles. We employed an architecture of two deep LSTM networks, one trained to create sequences of discrete words given a previous context and the other is trained to write with genuine cursive handwriting by predicting real-valued data points. By using the outputs of the second network as inputs for the first, we aim to create a model that can write on its own. The implementation showed promising result in producing different realistic fonts that resemble to the user's handwriting.

Keywords:Deep learning, RNN, LSTM, Cursive handwriting, Font Generation.

Résumé

La reconnaissance et la génération de textes écrits en caractères cursifs, restent un problème dans leur forme cursive en raison de la nature des caractères liés dans la langue Anglaise dans l'écriture manuscrite. Dans cette thèse, nous proposons une approche RNN hybride avec l'objectif de créer une application capable d'apprendre les modèles d'écriture manuscrite de l'utilisateur et de générer des styles d'écriture qui lui ressemblent. Nous avons utilisé une architecture de deux réseaux LSTM profonds, l'un formé pour créer des séquences de mots discrets compte tenu d'un contexte précédent et l'autre est formés pour écrire avec une écriture cursive authentique en prédisant des points de données à valeur réelle. En utilisant les sorties de deuxième réseau comme entrées pour le premier, nous visons à créer un modèle qui peut écrire tout seul. La mise en œuvre a montré des résultats prometteurs en produisant des différents polices réalistes similaires à l'écriture manuscrite de l'utilisateur.

Mots-clés : Apprentissage profond, RNN, LSTM, écriture manuscrite cursive, génération de polices.

الملخص

التعرف على النصوص المكتوبة وتوليدها بأحرف متصلة، لا يزال يمثل مجال بحث علمي نشط نظرا لطبيعة الحروف المتصلة باللغة الإنجليزية خاصة في الخطوط المكتوبة باليد. في هذه الأطروحة، نقترح نهج RNN هجين بهدف إنشاء تطبيق قادر على تعلم أنماط الكتابة اليدوية للمستخدم وتوليد خطوط كتابة تشبهها . استخدمنا بنية لشبكتي LTSM عميقتين، تم تدريب إحدها على إنشاء تسلسل من الكلمات المنفصلة وفقًا لسياق سابق والآخرى تم تدريبها على الكتابة بخط اليد الأصلي من خلال التنبؤ بنقاط البيانات ذات القيمة الحقيقية. باستعمال مخرجات الشبكة الثانية كمدخلات للشبكة الأولى، نهدف إلى إنشاء نموذج يمكن أن يكتب من تلقاء نفسه. أظهرت التجارب الأولية نتيجة واعدة في إنتاج خطوط واقعية تشبه الكتابة اليدوية للمستخدم.

الكلمات المفتاحية :التعلم العميق، الكتابة اليدوية المخطوطة، توليد الخط ، LSTM ، RNN.

CONTENTS

TABLES

LIST OF PLOTS

GENERAL INTRODUCTION

Humans have a lot of things in common, yet we are very unique individual in some aspects and one of them is handwriting. Handwriting is a personal skill that is affected by many particular variables. It has been and still remains an important means of communication and recording information in everyday life. Because of its unique characteristics, Handwriting Recognition and generation is an active area of research in computer science with the major focus of understanding the handwriting and converting it into readable text.

Deep learning, a major subfield of Artificial Intelligence (AI) that has been gaining a lot of attentionin recent years. Taking advantage of its fresh momentum and learning capabilities, Many researchers are focusing on creating artistic humanly projects using the huge available datasets such as poems, music, and even drawings, so why not handwritings?

Our goal in this project is to create new set of fonts, and rather than the classical used fonts, we aim to generate fonts that are aesthetically more pleasing and closer to human natural handwritings. Along with the hope of being able to replicate handwritings of historical important characters eventually.

In this work, we will propose an hybrid network of two RNNs to learn the personal handwriting of any user and generate a similar writing patterns automatically, by importing a personal handwritten image or writing it directly into a web application. This report contains four chapters sealed with a general conclusion and some perspectives:

Chapter 1: It will contain a background to the topic and more details about our motives.

Chapter 2: It will focus on other similar works along with an evaluation.

Chapter 3: Contains the general architecture of our system and thorough explanation.

Chapter 4: The final chapter will describe the system's implementation and the obtained results.

CHAPTER 1

DEEP LEARNING

1.1 Introduction

Daily scenarios deal with uncertainties in various areas, from investment opportunities and medical diagnoses to sports games and weather forecasts, and in all cases with the objective of making decisions based on collected observations and knowledge about uncertain areas. Models developed using machine learning and deep learning are widely used for all types of inference and decision making and generating new data. So what is meant by deep learning and what are its basic concepts? What are the techniques and methods of deep learning?

1.2 Definition

Deep learning or deep learning is a subfield of artificial intelligence (AI). This term designates the set of automatic learning techniques (machine learning), in other words a form of learning based on mathematical approaches, used to model data. To better understand these techniques, we must go back to the origins of artificial intelligence in 1950, the year during which Alan Turning became interested in machines capable of thinking[1].

This reflection will give birth to machine learning, a machine that communicates and behaves according to stored information. Deep learning is an advanced system based on the human brain, which includes a large network of artificial neurons. These neurons are interconnected to process and memorize information, compare any problems or situations with past similar situations, analyze the solutions and solve the problem in the best possible way.

As with humans, deep learning involves learning from lived experiences or, in the case of machines, recorded information.

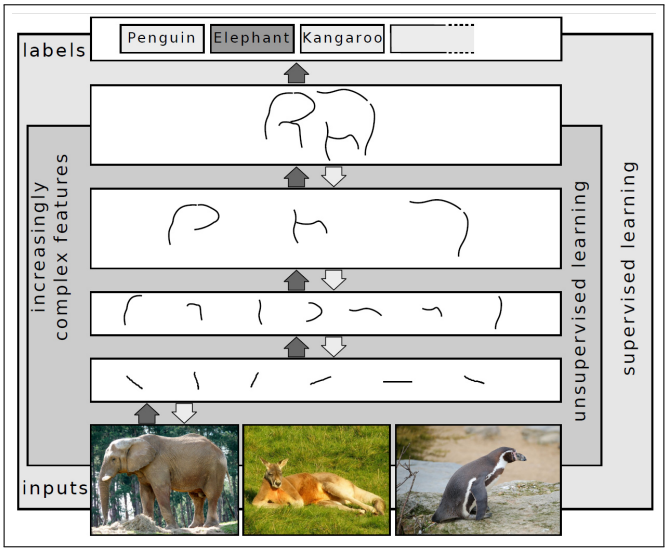

Figure 1.1: deep learning

What is the difference between Machine learning and Deep learning?

Deep learning is machine learning.

But deep learning is considered an evolution of machine learning. It uses a programmable neural network that enables machines to make accurate decisions without the help of humans.

In practice, deep learning is just a subset of machine learning. Technically deep learning is machine learning and works in a similar way.

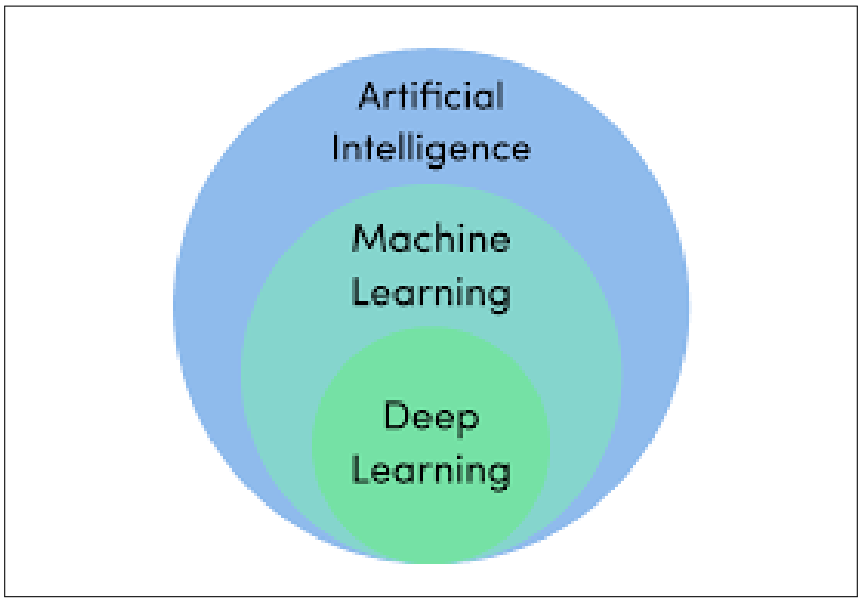

Figure 1.2: Machine Learning vs Deep Learning

Basic machine learning functions are getting better gradually, but it still needs some guidance. If the AI algorithm makes an inaccurate prediction, the programmer must step in and make adjustments.

Whereas deep learning models are designed to continuously analyze data with a logical structure similar to how humans draw conclusions. To achieve this, deep learning applications use a layered architecture of algorithms called an artificial neural network. The design of the artificial neural network is inspired by the biological neural network of the human brain, resulting in a learning process that is much more powerful than standard machine learning models.

The difference between them is summarized:

- In machine learning, we use algorithms to analyze data, learn from that data, and make correct decisions based on what you've learned.
- Deep learning builds algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own.
- Deep learning is a subfield of machine learning. While both fall into the broad category of artificial intelligence.
- Deep learning is what powers the most human-like AI

1.3 Basic concepts

Deep learning techniques are classified into three major categories: unsupervised, partially supervised (semi-supervised) and supervised. Furthermore, deep reinforcement learning (DRL), also known as RL, is another type of learning technique, which is mostly considered to fall into the category of partially supervised (and occasionally unsupervised) learning techniques.

1.3.1 Supervised learning

A machine learning approach that infers a function from a set of labeled training data. Training data consists of a set of real-world examples (eg hospital patient data, stock exchange value, etc.).When considering such a technique, the environs have a collection of inputs and resultant outputs (xt,yt). For instance, the smart agent guesses if the input is xt and will obtain as a loss value. Next, the network parameters are repeatedly updated by the agent to obtain an improved estimate for the preferred outputs. Following a positive training outcome, the agent acquires the ability to obtain the right solutions to the queries from the environs. For DL, there are several supervised learning techniques, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and deep neural networks (DNNs). In addition, the RNN category includes gated recurrent units (GRUs) and long short-term memory (LSTM) approaches. The main advantage of this technique is the ability to collect data or generate a data output from the prior knowledge. However, the disadvantage of this technique is that decision boundary might be overstrained when training set doesn't own samples that should be in a class. Overall, this technique is simpler than other techniques in the way of learning with high performance.[2] .

1.3.2 Unsupervised learning

This technique makes it possible to implement the learning process in the absence of available labeled data (i.e. no labels are required). Here, the agent learns the significant features or interior representation required to discover the unidentified structure or relationships in the input data. Techniques of generative networks, dimensionality reduction and clustering are frequently counted within the category of unsupervised learning. Several members of the DL family have performed well on nonlinear dimensionality reduction and clustering tasks; these include restricted Boltzmann machines, auto-encoders and GANs as the most recently developed techniques. Moreover, RNNs, which include GRUs and LSTM approaches, have also been employed for unsupervised learning in a wide range of applications. The main disadvantages of unsupervised learning are unable to provide accurate information concerning data sorting and computationally complex. One of the most popular unsupervised learning approaches is clustering[2] .

1.3.3 Semi-supervised learning

It is a machine learning approach that combines a small amount of labeled data with a large amount of unlabeled data during training. In this technique, the learning process is based on semi-classified data sets. Occasionally, generative Adversarial networks (GANs) and DRL are used in the same way as this technology. In addition to, RNNs, which include GRUs and LSTMs, are also used for micro-supervised learning. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with labeled training data only). It is a special case of weak supervision.One of the advantages of this technique is to reduce the amount of classified data required. On the other hand On the one hand, one of the drawbacks of this technique is the input feature that is not relevant to the current training data It can make wrong decisions. One of the most popular examples of a text document is a text document classifier Semi-supervised learning application. Because it is difficult to get a large amount of classified text Documents, semi-supervised learning is ideal for the task of classifying text documents.[2]

1.3.4 Reinforcement learning

Reinforcement Learning operates on interacting with the environment, while supervised learning operates on provided sample data. This technique was developed in 2013 with Google Deep Mind [3]. Subsequently, many enhanced techniques dependent on reinforcement learning were constructed. For example, if the input environment samples: xt, agent predict: and the received cost of the agent is , P here is the unknown probability distribution, then the environment asks a question to the agent. The answer it gives is a noisy score. This method is sometimes referred to as semi-supervised learning. Based on this concept, several supervised and unsupervised techniques were developed. In comparison with traditional supervised techniques, performing this learning is much more difficult, as no straightforward loss function is available in the reinforcement learning technique. In addition, there are two essential differences between supervised learning and reinforcement learning: first, there is no complete access to the function, which requires optimization, meaning that it should be queried via interaction; second, the state being interacted with is founded on an environment, where the input xt is based on the preceding actions[4] [5] .

For solving a task, the selection of the type of reinforcement learning that needs to be performed is based on the space or the scope of the problem. For example, DRL is the best way for problems involving many parameters to be optimized. By contrast, derivative-free reinforcement learning is a technique that performs well for problems with limited parameters. Some of the applications of reinforcement learning are business strategy planning and robotics for industrial automation. The main drawback of Reinforcement Learning is that parameters may influence the speed of learning. Here are the main motivations for utilizing Reinforcement Learning:

- It assists you to identify which action produces the highest reward over a longer period.
- It assists you to discover which situation requires action.
- It also enables it to figure out the best approach for reaching large rewards.
- Reinforcement Learning also gives the learning agent a reward function.

Reinforcement Learning can't utilize in all the situation such as:

- In case there is sufficient data to resolve the issue with supervised learning techniques.
- Reinforcement Learning is computing-heavy and time-consuming. Specially when the workspace is large.

1.4 Techniques and methods

1.4.1 Artificial neural networks

Artificial neural networks (ANN) or connectionist systems are computer systems inspired by the biological neural networks that make up the brains of animals. Such systems learn (gradually improve their ability) to perform tasks by considering examples, usually without task-specific programming. For example, in image recognition, they can learn to identify images containing cats by analyzing sample images that have been manually labeled as "cat" or "no cat" and using the analytical results to identify cats in other images. They have found most use in applications that are difficult to express with a traditional computer algorithm using rule-based programming[6].

An ANN is based on a collection of connected units called artificial neurons (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal the downstream neurons connected to it. Neurons can have a state, usually represented by real numbers, usually between 0 and 1. Neurons and synapses can also have a weight that varies as they learn, which can increase or decrease in strength of the signal it sends downstream[6].

Typically, neurons are organized in layers. Different layers can perform different types of transformations on their inputs. Signals travel from the first layer (input) to the last layer (output), possibly after traversing the layers several times.

The original goal of the neural network approach was to solve problems in the same way as a human brain would. Over time, attention has focused on matching specific mental abilities, resulting in deviations from biology such as backpropagation or transmission of information in the reverse direction and adjustment of the network to reflect these informations.

Neural networks have been used for a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, board and video games, and medical diagnosis.

As of 2017, neural networks typically have a few thousand to a few million units and millions of connections. Although this number is several orders of magnitude lower than the number of neurons on

a human brain, these networks can perform many tasks at a higher level than humans (e.g., recognizing faces, playing "Go" [6]).

1.4.2 Deep neural networks

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers[7]. There are different types of neural networks, but they are always made up of the same components: neurons, synapses, weights, biases and functions.These components work similarly to the human brain and can be trained like any other ML algorithm.

For example, a DNN trained to recognize dog breeds will review the given image and calculate the probability that the dog in the image belongs to a certain breed. The user can view the results and select the probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNNs have many layers, hence the name "deep" networks.

DNNs can model complex nonlinear relationships. DNN architectures generate composition models where the object is expressed as a layered composition of primitives[8]. Additional layers allow feature composition from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. For example, it has been proven that sparse multivariate polynomials are exponentially easier to approximate with DNNs than with shallow networks.

Deep architectures include many variations of a few basic approaches. Each architecture has found success in specific areas. It is not always possible to compare the performance of several architectures, unless they have been evaluated on the same datasets.

DNNs are typically feed-forward networks in which data flows from the input layer to the output layer without looping back. First, the DNN creates a map of virtual neurons and assigns random numerical values, or “weights,” to the connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights.[9] In this way, the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

1.4.3 Convolutional Neural Networks

In the field of Deep Learning, the CNN is the most famous and commonly employed algorithm [10] [11]. The main benefit of CNN is that it automatically identifies the relevant features without any human supervision. CNNs have been extensively applied in a range of different fields, including computer vision , speech processing , Face Recognition , etc. The structure of CNNs was inspired by neurons in human and animal brains, similar to a conventional neural network. More specifically, in a cat’s brain, a complex sequence of cells forms the visual cortex; this sequence is simulated by

the CNN . Goodfellow et al[12]. identified three key benefits of the CNN: equivalent representations, sparse interactions, and parameter sharing. Unlike conventional fully connected (FC) networks, shared weights and local connections in the CNN are employed to make full use of 2D input-data structures like image signals. This operation utilizes an extremely small number of parameters, which both simplifies the training process and speeds up the network. This is the same as in the visual cortex cells. Notably, only small regions of a scene are sensed by these cells rather than the whole scene (i.e., these cells spatially extract the local correlation available in the input, like local filters over the input).

A commonly used type of CNN, which is similar to the multi-layer perceptron (MLP), consists of numerous convolution layers preceding sub-sampling (pooling) layers, while the ending layers are FC layers.An example of CNN architecture for image classification is illustrated in Fig.

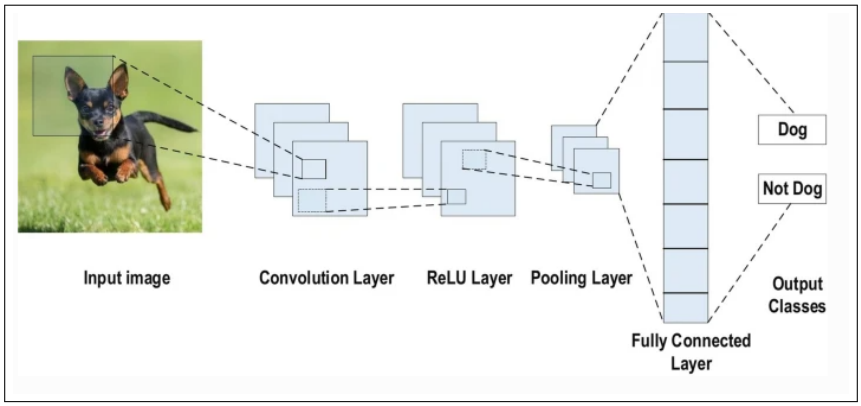

Figure 1.3: An example of CNN architecture for image classification

- The main reason to consider CNN is the weight sharing feature, which reduces the number of trainable network parameters and in turn helps the network to enhance generalization and to avoid overfitting.
- Concurrently learning the feature extraction layers and the classification layer causes the model output to be both highly organized and highly reliant on the extracted features.
- Large-scale network implementation is much easier with CNN than with other neural networks.

CNN layers:

The CNN architecture consists of a number of layers (or so-called multi-building blocks). Each layer in the CNN architecture, including its function, is described in detail below.

1. Convolutional Layer: The convolution layer is the core building block of the CNN.It carries the main portion of the network's computational load. It consists of a collection of convolutional filters (so-called kernels). The input image, expressed as N-dimensional metrics, is convolved with these filters to generate the output feature map[10].

2. Pooling Layer: The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. The main task of the pooling layer is to subsamples from the feature maps. these Maps are created by following convolutional operations. That is, this approach Shrink large feature maps to create smaller feature maps. At the same time, it maintains The majority of information (or features) prevalent at each step of the assembly stage. in similar By the method of the convolutional process, the size of both the step and the kernel are initially determined before performing the assembly process. Several kinds of collection methods are available for Utilize in different grouping layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling. The most familiar and frequently utilized pooling methods are the max, min, and GAP pooling. Figure illustrates these three pooling operations.

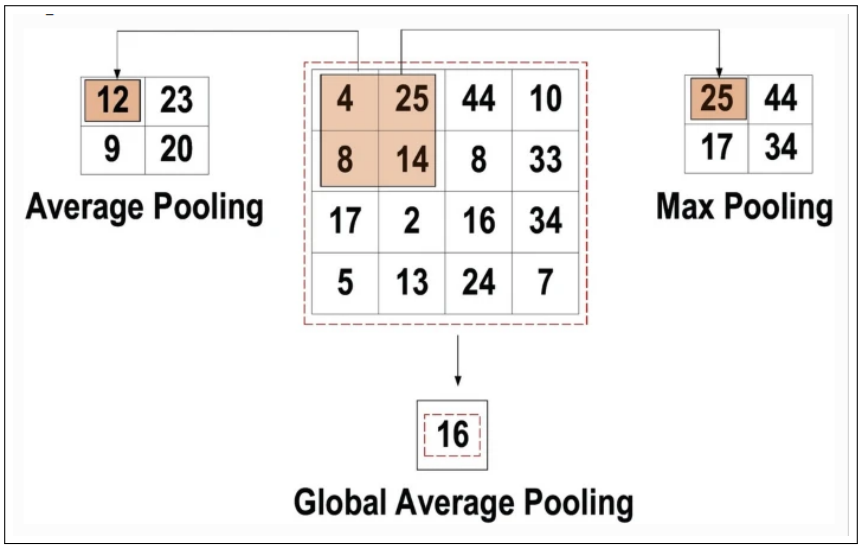

Figure 1.4: Three types of pooling operations

Sometimes, overall CNN performance drops as a result; This is the main Lack of pooling layer, this layer contributes CNN to determine if The feature is available in the particular input image, but it is exclusively focused on verifying The correct location for that feature. Thus, the CNN model is missing relevant information.

3. Activation Function (non-linearity):Since convolution is a linear process and images are far from linear, nonlinear layers are often placed directly after the convolution layer to introduce nonlinearity into the activation map. Assigning inputs to outputs is the basic function of all Types of activation function in all types of neural networks. The input value is determined by Calculate weighted aggregation of neuron inputs with their bias (if any). this is It means that the activation function makes the decision whether or not to fire a neuron Reference to a specific input by creating the corresponding output.

   Non-linear activation layers are employed after all layers with weights (so-called learnable layers,

such as FC layers and convolutional layers) in CNN architecture. This non-linear performance of the activation layers means that the mapping of input to output will be non-linear; moreover, these layers give the CNN the ability to learn extra-complicated things. The activation function must also have the ability to differentiate, which is an extremely significant feature, as it allows error back-propagation to be used to train the network. The following types of activation functions are most commonly used in CNN and other deep neural networks[10].

4. Fully Connected Layer:Generally, this layer is located at the end of each CNN structure. The neurons in this layer have complete communication with all the neurons in the previous and subsequent layer as seen in a normal FCNN, the so-called fully Connected Policy (FC). It is used as a CNN classifier. It follows the basic method of The traditional multi-layer perceptual neural network, as it is a kind of feed-forward ANN. The input to the FC layer comes from the last pool or convolutional layer. This entry is in . format Vector shape, created from feature maps after flattening. output file The FC layer represents the final CNN output, as illustrated in Fig.

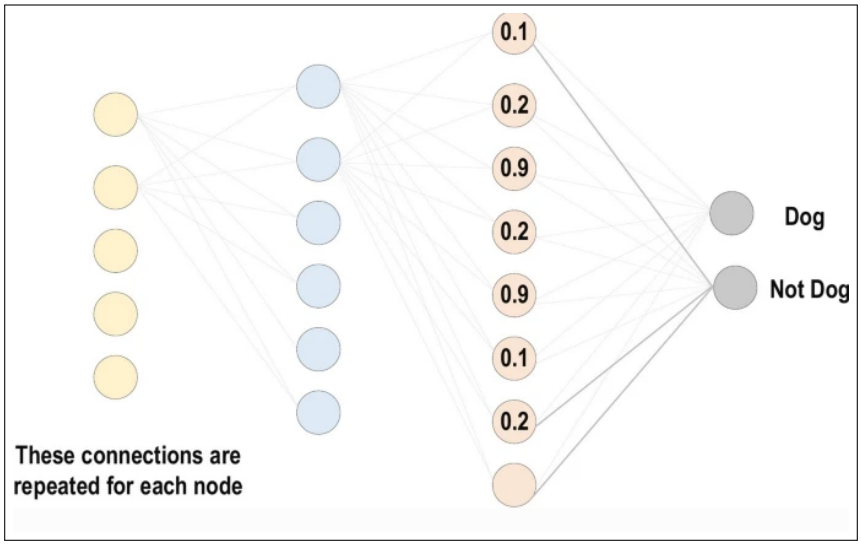

Figure 1.5: Fully connected layer

5. Loss Functions: The previous section has presented various layer-types of CNN architecture. In addition, the final classification is achieved from the output layer, which represents the last layer of the CNN architecture. Some loss functions are utilized in the output layer to calculate the predicted error created across the training samples in the CNN model. This error reveals the difference between the actual output and the predicted one. Next, it will be optimized through the CNN learning process.

Improving performance of CNN:

Based on our experiments in different DL applications [13]. We can conclude the most active solutions that may improve the performance of CNN are:

- Expand the dataset with data augmentation or use transfer learning (explained in latter sections).
- Increase the training time.
- Increase the depth (or width) of the model.
- Add regularization.
- Increase hyperparameters tuning.

1.4.4 Recurrent neural networks

RNNs are a commonly employed and familiar algorithm in the discipline of DL [14]. RNN is mainly applied in the area of speech processing and NLP contexts. Unlike conventional networks, RNN uses sequential data in the network. Since the embedded structure in the sequence of the data delivers valuable information, this feature is fundamental to a range of different applications. For instance, it is important to understand the context of the sentence in order to determine the meaning of a specific word in it. Thus, it is possible to consider the RNN as a unit of short-term memory, where x represents the input layer, y is the output layer, and s represents the state (hidden) layer. For a given input sequence, a typical unfolded RNN diagram is illustrated in Fig . Pascanu et al.[15] introduced three different types of deep RNN techniques, namely “Hidden-to-Hidden”, “Hidden-to-Output”, and “Input-to-Hidden”. A deep RNN is introduced that lessens the learning difficulty in the deep network and brings the benefits of a deeper RNN based on these three techniques.

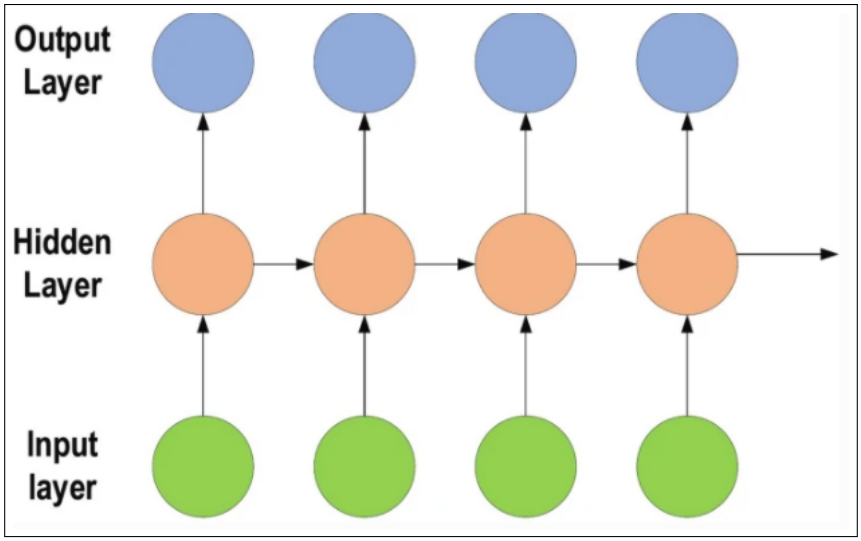

Figure 1.6: Typical unfolded RNN diagram

1.4.5 Recursive neural networks

RvNN can achieve predictions in a hierarchical structure also classify the outputs utilizing compositional vectors. Recursive auto-associative memory (RAAM) is the primary inspiration for the RvNN development. The RvNN architecture is generated for processing objects, which have randomly shaped structures like graphs or trees. This approach generates a fixed-width distributed representation from a variable-size recursive-data structure. The network is trained using an introduced back-propagation through structure (BTS) learning system [16]. The BTS system tracks the same technique as the general-back propagation algorithm and has the ability to support a treelike structure. Auto-association trains the network to regenerate the input-layer pattern at the output layer. RvNN is highly effective in the NLP context. Socher et al. introduced RvNN architecture designed to process inputs from a variety of modalities. These authors demonstrate two applications for classifying natural language sentences: cases where each sentence is split into words and nature images, and cases where each image is separated into various segments of interest. RvNN computes a likely pair of scores for merging and constructs a syntactic tree. Furthermore, RvNN calculates a score related to the merge plausibility for every pair of units. Next, the pair with the largest score is merged within a composition vector. Following every merge, RvNN generates (a) a larger area of numerous units, (b) a compositional vector of the area, and (c) a label for the class (for instance, a noun phrase will become the class label for the new area if two units are noun words). The compositional vector for the entire area is the root of the RvNN tree structure. An example RvNN tree is shown in Fig . RvNN has been employed in several applications[17].

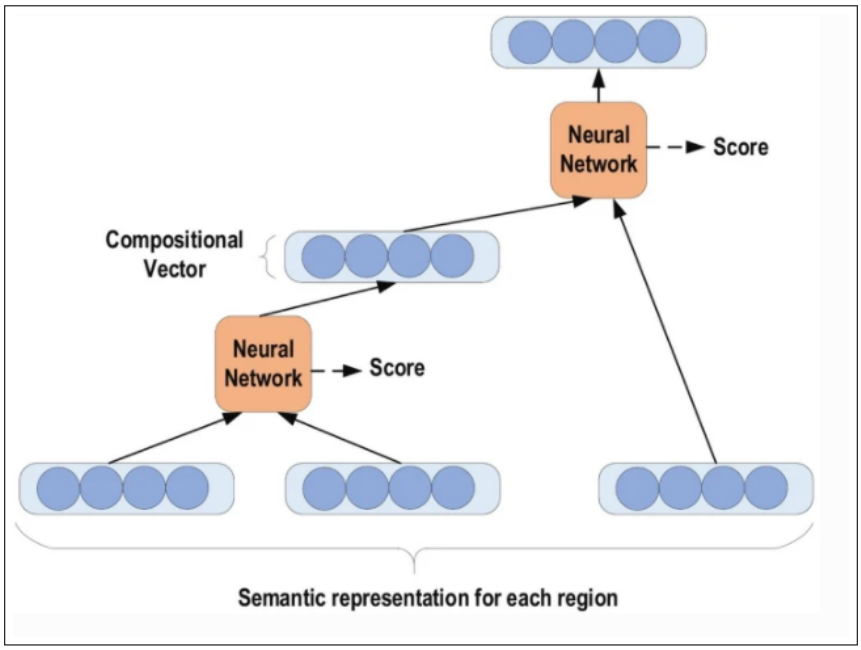

Figure 1.7: An example of RvNN tree

Comparing the Different Types of Neural Networks (ANN vs. RNN vs. CNN)

The table below, summarizes some of the differences among different types of neural networks:

	ANN	RNN	CNN
Data	Tabular data	Sequence data	Image data
Recurrent connections	No	Yes	No
Parameter sharing	No	Yes	Yes
Spatial relationship	No	No	Yes
Vanishing and Exploding Gradien	Yes	Yes	Yes

Table 1.1: the different types of neural networks

1.5 Applications

Deep Learning is used in many areas:

- image recognition.
- automatic translation: Automatic translation: Unsupervised machine translation using only monolingual language, The work of Guillaume Lample, who is French, is completely independent but uses the same technique A computer engineer working in the artificial intelligence department at Facebook. Either way, the Researchers show that a neural network can learn to translate a language without having to dig into the dictionary.

In recent years, machine translation has made tremendous progress thanks to neural networks that Enhanced machine learning performance. However, this type of AI requires large amounts For content previously translated by humans. Translation is the result of supervised learning in which The machine guesses and then receives the correct answer from the human, allowing it to adjust its processing accordingly. This method is effective for widely spoken languages, such as English or French, as there are many parallel documents. But it doesn't work well for the rarity Languages that do not offer such a combination.

- autonomous car[18].
- medical diagnosis: To initiate more effective treatments and better assess the vital prognosis, the challenge is to precisely identify the type of brain tumor tumor with which we are dealing. Classically, the diagnosis is made by histology, that is to say by microscopic observation of the biological tissues that have been taken[19].

This method relies on accurate cellular anomaly analysis, so the final diagnosis can vary Significantly from one doctor to another. In addition, several types of histological tumors may be involved characteristics without evolving in the same way.

That is why David Capper and colleagues turned to another analytical method based on molecular markers: the method of DNA methylation in cancer cells, that is, the addition of methyl groups to DNA. This type of diagnosis, which is characterized as being completely objective, is not carried out systematically, however, due to the lack of resources, despite the efforts made in this direction. Since 2016, the World Health Organization has advised the combination of tissue and molecular analysis to screen for certain types of brain tumors.

Wanting to standardize the process, the researchers trained a computer to analyze and classify brain tumors into different categories according to their methylation profiles, thanks to machine learning, a field of study that falls under artificial intelligence. Their system improves the accuracy of classic histological or molecular diagnostics. In fact, 12% of tumors out of 1104 cases studied were reclassified according to computer recommendations.

- Personal recommendations.
- Auto moderation for social networks.
- Financial forecasting and automated trading.
- identification of defective parts.
- Detecting malware or fraud.
- chatbots (conversational agents).
- space exploration: At a time when researchers are overwhelmed by mountains of astronomical data via ground-based telescopes and space probes, artificial intelligence comes as a saviour. Already included in the search for exoplanets or classification of galaxies, here is trying to identify lunar craters.

Inventorying craters caused by the impact of meteorites is a daunting task, because our satellite contains hundreds of thousands of them. But it does provide information on the dynamics and distribution of matter during the youth of the Solar System. Some of the craters, which are still visible today in the absence of erosion, were actually formed 4 billion years ago.

- smart robots.

1.6 Conclusion

Deeper neural networks are more difficult to train yet they are a powerful technique to solve image or handwritten based problems. In this chapter, we tried to cover some of the principal concepts related to deep learning and as we move forward in chapters, we are going to discuss handwriting recognition and generation using different techniques and highlight our motivation and goals.

CHAPTER 2

IMAGE PROCESSING AND CALLIGRAPHY

2.1 Introduction

Handwriting synthesis is the automatic generation of data that resembles natural handwriting. Although handwriting synthesis has recently gained increasing attention, the area still lacks a stand-alone review. This chapter presents classifications of the various aspects of handwriting composition, and presents applications, techniques, and assessment methods for handwriting composition based on the many aspects that we identify. Then it discusses different synthesis techniques.

2.2 Image processing

Images contain a lot of important information. If they are easy to detect for our accustomed eyes, they represent a real challenge in data analysis. All of these techniques are known as "image processing". In this section, we will discuss the classic algorithms, techniques and tools for processing data in the form of images[20].

What is image processing?

Image processing, as the name suggests, is the process of processing images, which can include many different techniques, and the end result can be another image, a variation, or just another image. a parameter of this image. This result can then be used for further analysis or decision making.

This is the core part of computer vision that plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at once and extract useful information from them.

But what is an image?

An image can be represented as a 2D function F(x,y) where x and y are spatial coordinates. It is therefore an array of pixels arranged in columns and rows. The value of F at a point x,y is known as the intensity of an image at that point. If x, y and the amplitude value are finite, it is called a digital image. An image can also be represented in 3D whose coordinates are x, y and z. The pixels are then arranged in the form of a matrix. This is called an RGB image. If the image is grayscale, there is only one channel: $z = 1$.

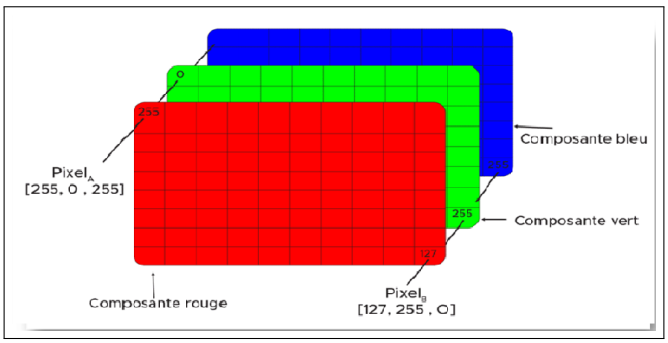

Figure 2.1: illustration of an rgb image

Classic image processing techniques

Historically, images were processed with mathematical analysis methods, we will present some of them below:

- Gaussian blurring, or Gaussian smoothing, is the result of applying a Gaussian function to an image, i.e. to a matrix as defined above. It is used to reduce image noise and soften detail. The visual effect of this blurring technique is similar to looking at an image through a translucent screen. It is sometimes used as a data augmentation technique for deep learning which we will discuss in more detail below.

- The Fourier transform decomposes an image into sine and cosine components. It has multiple applications such as image reconstruction, image compression or image filtering. Since we are talking about images, we will consider the discrete Fourier transform. Consider a sinusoid, it is composed of three elements:
  - Amplitude – related to contrast.
  - Spatial frequency – related to brightness.
  - Phase – related to color information.

Edge detection is an image processing technique for finding the boundaries of objects in images, it works by detecting discontinuities in brightness. More precisely, the edges are defined as the local maxima of the image gradient, i.e. the areas where there is a large variation in values between two pixel areas. The mostcommonedge detection algorithmis that of Sobel.

2.3 Optical Character recognition

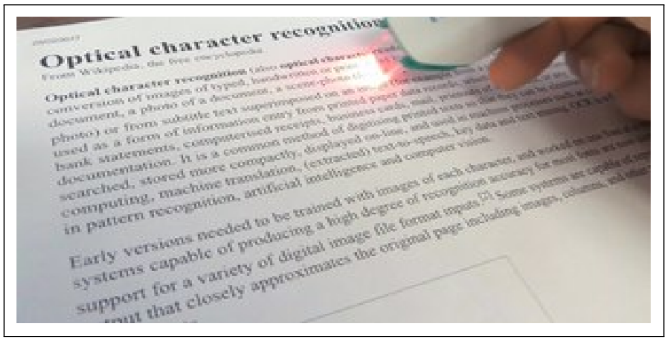

Figure 2.2: .know your imaging on symbols

digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast)[21]. Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision. Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components[22].

History

Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind[23]. In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code[24]. Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters[25].

In the late 1920s and into the 1930s Emanuel Goldberg developed what he called a "Statistical Machine" for searching microfilm archives using an optical code recognition system. In 1931 he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM.

Blind and visually impaired user

In 1974, Ray Kurzweil started the company Kurzweil Computer Products, Inc. and continued development of omni-font OCR, which could recognize text printed in virtually any font (Kurzweil is often credited with inventing omni-font OCR, but it was in use by companies, including CompuScan, in the late 1960s and 1970s[23][26]). Kurzweil decided that the best application of this technology would be to create a reading machine for the blind, which would allow blind people to have a computer read text to them out loud. This device required the invention of two enabling technologies – the CCD flatbed scanner and the text-to-speech synthesizer. On January 13, 1976, the successful finished product was unveiled during a widely reported news conference headed by Kurzweil and the leaders of the National Federation of the Blind.[citation needed] In 1978, Kurzweil Computer Products began selling a commercial version of the optical character recognition computer program. LexisNexis was one of the first customers, and bought the program to upload legal paper and news documents onto its nascent online databases. Two years later, Kurzweil sold his company to Xerox, which had an interest in further commercializing paper-to-computer text conversion. Xerox eventually spun it off as Scansoft, which merged with Nuance Communications.

In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs on a smartphone. With the advent of smart-phones and smartglasses, OCR can be used in internet connected mobile device applications that extract text captured using the device's camera. These devices that do not have OCR functionality built into the operating system will typically use an OCR API to extract the text from the image file captured and provided by the device[27][28] .The OCR API returns the extracted text, along with information about the location of the detected text in the original image back to the device app for further processing (such as text-to-speech) or display. Various commercial and open source OCR systems are available for most common writing systems, including Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), Devanagari, Tamil, Chinese, Japanese, and Korean characters.

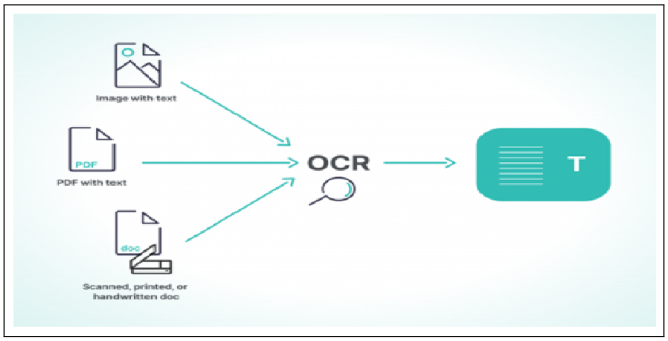

Figure 2.3: optical character recognition (OCR)

Applications

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR. They can be used for:

- Data entry for business documents, e.g. Cheque, passport, invoice, bank statement and receipt.
- Automatic number plate recognition.
- In airports, for passport recognition and information extraction.
- Automatic insurance documents key information extraction.
- Traffic sign recognition.
- Extracting business card information into a contact list[29].
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg.
- Make electronic images of printed documents searchable, e.g. Google Books.
- Converting handwriting in real-time to control a computer (pen computing).
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR.The purpose can also be to test the robustness of CAPTCHA anti-bot systems.
- Assistive technology for blind and visually impaired users.
- Writing the instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time.
- Making scanned documents searchable by converting them to searchable PDFs.

Types

- Optical character recognition (OCR) – targets typewritten text, one glyph or character at a time.
- Optical word recognition – targets typewritten text, one word at a time (for languages that use a space as a word divider). (Usually just called "OCR".).
- Intelligent character recognition (ICR) – also targets handwritten printscript or cursive text one glyph or character at a time, usually involving machine learning.
- Intelligent word recognition (IWR) – also targets handwritten printscript or cursive text, one word at a time. This is especially useful for languages where glyphs are not separated in cursive script.

OCR is generally an "offline" process, which analyses a static document. There are cloud based services which provide an online OCR API service. Handwriting movement analysis can be used as input to handwriting recognition. Instead of merely using the shapes of glyphs and words, this technique is able to capture motions, such as the order in which segments are drawn, the direction, and

the pattern of putting the pen down and lifting it. This additional information can make the end-to-end process more accurate. This technology is also known as "on-line character recognition", "dynamic character recognition", "real-time character recognition", and "intelligent character recognition".

Techniques

1. Pre-processing: OCR software often "pre-processes" images to improve the chances of successful recognition. Techniques include[30]:

   • De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counterclockwise in order to make lines of text perfectly horizontal or vertical

   • Despeckle – remove positive and negative spots, smoothing edges.

   • Binarisation – Convert an image from color or greyscale to black-and-white (called a "binary image" because there are two colors). The task of binarisation is performed as a simple way of separating the text (or any other desired image component) from the background[31]. The task of binarisation itself is necessary since most commercial recognition algorithms work only on binary images since it proves to be simpler to do so[32]. In addition, the effectiveness of the binarisation step influences to a significant extent the quality of the character recognition stage and the careful decisions are made in the choice of the binarisation employed for a given input image type; since the quality of the binarisation method employed to obtain the binary result depends on the type of the input image (scanned document, scene text image, historical degraded document etc.)[33].

   • Line removal – Cleans up non-glyph boxes and lines.

   • Layout analysis or "zoning" – Identifies columns, paragraphs, captions, etc. as distinct blocks. Especially important in multi-column layouts and tables.

   • Line and word detection – Establishes baseline for word and character shapes, separates words if necessary.

   • Script recognition – In multilingual documents, the script may change at the level of the words and hence, identification of the script is necessary, before the right OCR can be invoked to handle the specific script[34].

   • Character isolation or "segmentation" – For per-character OCR, multiple characters that are connected due to image artifacts must be separated; single characters that are broken into multiple pieces due to artifacts must be connected.

   • Normalize aspect ratio and scale[35]. Segmentation of fixed-pitch fonts is accomplished relatively simply by aligning the image to a uniform grid based on where vertical grid lines

will least often intersect black areas. For proportional fonts, more sophisticated techniques are needed because whitespace between letters can sometimes be greater than that between words, and vertical lines can intersect more than one character[36].

2. Text recognition: There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters[36].

   - Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

   - Feature extraction decomposes glyphs into "features" like line.

   - Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded)[36].

   Modern OCR software include Google Docs OCR, ABBYY FineReader and Transym.Others like OCRopus and Tesseract uses neural networks which are trained to recognize whole lines of text instead of focusing on single characters.

   A new technique known as iterative OCR automatically crops a document into sections based on page layout. OCR is performed on the sections individually using variable character confidence level thresholds to maximize page-level OCR accuracy. A patent from the United States Patent Office has been issued for this method . The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress. Other common formats include hOCR and PAGE XML.

   For a list of optical character recognition software see Comparison of optical character recognition software.

3. Post-processing:

   OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document[24]. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy[31].

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

"Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together.For example, "Washington, D.C." is generally far more common in English than "Washington DOC". Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy. The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

4. Application-specific optimizations:

   In recent years, the major OCR technology providers began to tweak OCR systems to deal more efficiently with specific types of input. Beyond an application-specific lexicon, better performance may be had by taking into account business rules, standard expression,or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing.

   The New York Times has adapted the OCR technology into a proprietary tool they entitle, Document Helper, that enables their interactive news team to accelerate the processing of documents that need to be reviewed. They note that it enables them to process what amounts to as many as 5,400 pages per hour in preparation for reporters to review the contents.

2.4 Handwriting analysis (literature review)

Generating personal handwriting fonts with large amounts of characters is a boring and time-consuming task. Take Chinese fonts as an example, the official standard GB18030-2000 for commercial font products contains 27533 simplified Chinese characters. Consistently and correctly writing out such huge amounts of characters is usually an impossible mission for ordinary people.

Researchers adopted different ways to achieve the goal, each of those researches worked to determine the image characteristics to reach the best grades with the accuracy of the feature of the handwriting. Another thing that was debated is the quality of the photos, those images may be not Clear because of the quality of the sensor, Without forgetting to try to ameliorate the quality of security (More secure) and make the system as fast as possible.

• Random selection, In this article the contribution was to design a system that makes a person's new handwriting style on paper. The user studies paper-based documents and studies a persuasive approach to the casual observer. No previous work has done this. Handwriting is simply

erased from paper samples. The tablet can be used, but only for annotation of preserved samples. Samples may be linked (cursive), print (disconnected), or in between. Natural handwriting is used, instead of short isolated/consecutive alphabets. This provides a novel ability to model the handwriting of historic figures. Rendering a sentence typically takes around 8 seconds on a 3.8Ghz i7. This approach generates output that looks real to a casual observer And the authors were first to replicate the writing of a historic figure[37].

- The Store Bank, The model aims to create a Stroke Bank with little human supervision. The grammatical base of the form is used to decompose words in a hierarchical representation. The authors presented a novel approach to Chinese handwriting generation, a dictionary that maps components in standard font to a particular handwriting. The model does not need expert input about the structure of Chinese characters or human supervision in stroke extraction, yet it is able to capture cursiveness, spatial correlation between strokes, and other font characteristics through component mapping based on weighted features. The generated characters resemble the actual handwritings in both structure and cursiveness and are almost indistinguishable from ground truth handwritings[38].

- Automatic Generation of Large-scale Handwriting Fonts via Style Learning, The user only needs to write a small number of characters on a blank sheet of paper and upload the image to a website. The system can then automatically creates a font library for the user's personal style handwriting with huge amounts of Chinese characters. During the offline processing period, the authors, first manually, specify the write path for each stroke of all Characters in the standard "Kaiti" font library that are used as reference data for their system. Then enter character sets they are properly selected to meet the requirements of different situations. while online, stroke paths were extracted for individual character images derived from input text images. Then Leveraging Artificial Neural Networks (ANNs) for Learning and Reconstructing the general style of the user's handwriting. Experimental results showed that the system is capable of automatically creating high-quality handwriting font libraries that include massive amounts of machine-generated characters that are indistinguishable from the original handwriting[39].

- kai standard line, the authors built a user-friendly web interface for crowds to manually label the positions and sizes of components of every Chinese character in the target character set. The Kai font was selected as a reference. they also devised an algorithm to find a small subset of Chinese characters having all required components to synthesize other Chinese characters. In production phase, to create a personal handwriting font, a user has to handwrite the small subset of Chinese characters on an smartphone or electronic pad. One character by one character, the system tracks every stroke and extracts components from the handwriting. Then, every target Chinese character is synthesized from the extracted components, by placing them properly according to their position and size information. To create a personal handwriting font, with commonly-used 3,914 traditional Chinese characters, a user has to hand-write only 400 or so Chinese characters.Style Chinese font,the

model consists of three sub networks, classification network (the classifier), generating network (the generator), and discriminating network (the discriminator): the classifier is used to first identify the most similar font in the pre-collected font sets, the generator implements the learning and transferring from the identified font to our target font using the provided training data, and the discriminator tries to differentiate the generated images from the real ones. The latter two networks are optimized,the model can generate satisfactory results at much smaller size from the volume of training data[40].

- The fonts in connected texts or semi-connected text were a problem. To overcome these problems, improve the network suggesting structure, by introducing predefined network classification to determine the first style information the target line before training. It is more intuitive and natural for the network to learn the handwriting method of known line style nearby, and therefore less training data required to achieve the desired results. Experiences show that we can actually help the training of the required data set and improve learning and generate performance of handwriting lines. As for the detailed training process, we adopted generating based on the FCN network that can learn the general structure of the line and transfer the letter as a whole without any extraction or other prejudices. In addition, a discriminatory network has been tried from the GAN structure to increase the improvement of efficacy and accuracy of the generation network[41].

- The DSD, this work presents a method for representing handwriting strokes online through a Decoupled Style Descriptor (DSD) model. DSD succeeds in creating preferred drawing samples more often in user study than the modern model. Furthermore, the authors demonstrate the capabilities of their model in interpolating samples at different representation levels, retrieving new character representations, and achieving high writer-identification accuracy, even though they are not explicitly trained to perform these tasks. Online synthesis of handwriting remains a challenge, especially when it deduces a stylistic representation from a small number of samples and attempts to create new ones. However, class style factors are shown to have potential, and it is believed that they can also apply to stylistic tasks such as transmission and interpolation in other fields of sequential data, such as speech synthesis, dance movement prediction, and musical comprehension[42].

It's clear that most works in the literature addresses the Chinese language as a target, because it's more complicated and has special characteristics. The table below classifies existing systems, trying to present for each the database used, year, time or success rate, average error rate, the used method and user inputs. These works are classified based on the used technology and in chronological order:

N	Year	Method used	DB used	Time or Success Rate	Average Error Rate
1	2006	rnn	Alpha matted	8 seconds on a 3.8Ghz i7	85%
2	2012	store bank	Kai	/	49,6%
3	/	ANNs	MinSet OptSet	/	51,25%
4	2008	kai standard line	Big5/CNS/GB	/	/
5	2017	cnn	GB/GA/FCN/BD	/	95%
6	/	DSD	BRUSH/GAN	/	89.70%

Table 2.1: literature review

As presented, In all the mention previous works, authors tried to solve handwriting synthesis related to Chinese language due to it's complicated nature and achieved remarkable results.

For our contribution, we aim to generate handwriting fonts in English language. We will try to use deep learning technique (RNN) to create a web interface that can recognizes handwriting and generates styled handwriting fonts that are similar to user inputs.

2.5 Conclusion

In this chapter, we have provided an overview of the most important methods used for handwriting analysis, as well as some effective systems for handwriting generation. We also classified similar works to present their methods and their results. In the next chapter, we will describe the general structure of our system and the techniques we used.

CHAPTER 3

SYSTEM CONCEPTION

3.1 Introduction

In this chapter, we are going to demonstrate the use of a form of recurrent neural network (LSTM) for online handwriting synthesis and text prediction. We will introduce the general architecture of our system and explain how RNN will be employed to achieve the purpose of our project.

3.2 General architecture of our system

In the following diagram, we present the general architecture of our system which is composed of three distinguished parts.

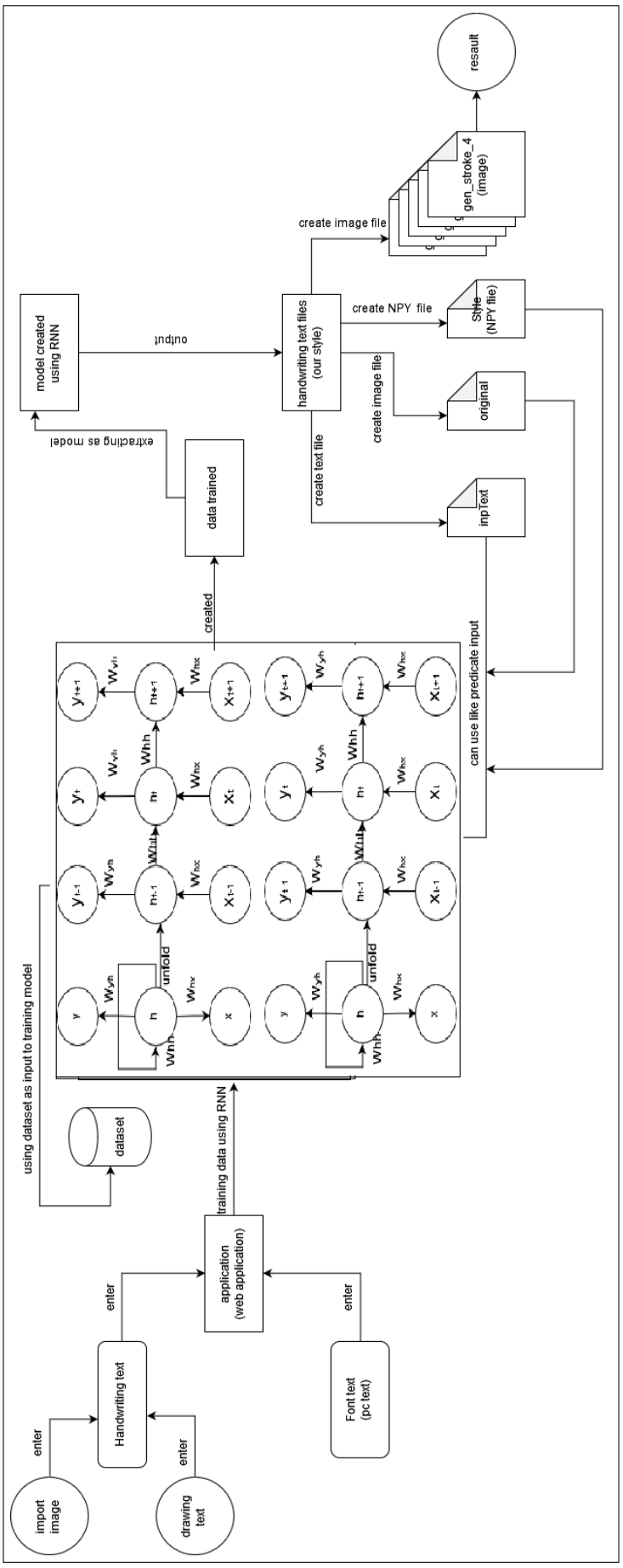

Figure 3.1: General Architecture of our project System

Part 1: Acquiring the user style

Firstly, the application requires two inputs "handwriting" (writing on screen or importing image) and the text written using keyboard as shown above. When the user clicks the button save Style on the application, three files (file.text, image.png, style.npy) are created as shown in the diagram below.

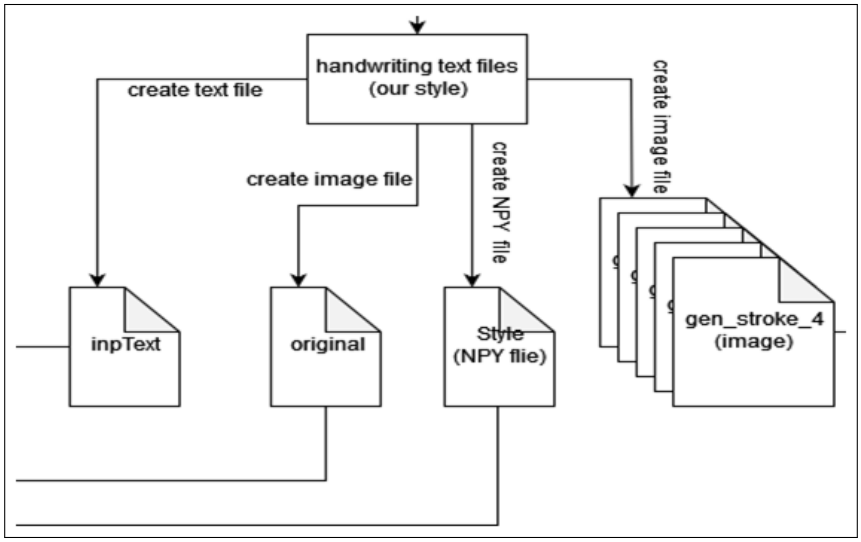

Figure 3.2: The created files in the diagram

file.text, this file contains the sentence entered by the user corresponding to the drawing or a picture, image.png contains the original entred pattern, style.npy, an NPY file is a NumPy array file generated by the Python software package that includes the NumPy module.

Part 2: Traing the RNN

Recurrent Neural Network(RNN):

RNNs are a strong and durable sort of neural network, and they are one of the most promising algorithms in use since they are the only ones that have internal memory. Increased processing power, along with vast quantities of data to work with, and the advent of long short-term memory (LSTM) in the 1990s, has pushed RNNs to the forefront. RNNs can recall critical details about the input they received because of their internal memory, allowing them to anticipate what will happen next with great accuracy. This is why they are the chosen algorithm for time series, voice, text,financial data,audio,video,weather,and other sequential data. When compared to their algorithms,recurrent neural networks can acquire a far deeper knowledge of a sequence and its context[43].

We employed an ensemble of two deep LSTM networks ,one trained to create sequences of discrete words given a previous context and the other trained to write with genuine cursive handwriting by predicting real-valued data points one at a time.

By using the outputs of the second network as inputs for the first, we hope to create a model that can write on its own. Our approach is a hybrid of two seemingly disparate models (by Alex Graves and Mikolovetal[44].) that collaborate to do the more difficult task of creating genuine and discrete

valued sequences with long-range structure.

The used hybrid approach of Alex Graves and Mikolovetal :

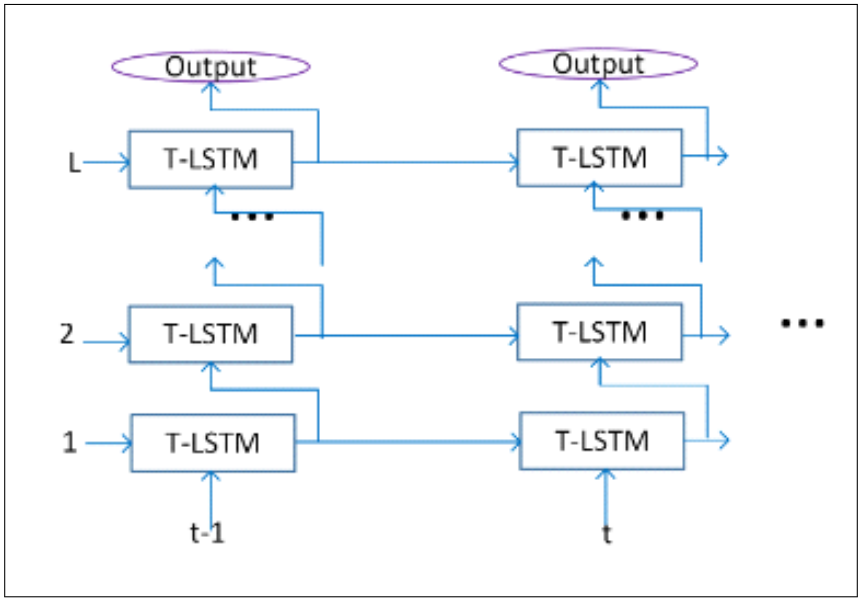

Figure 3.3: Multi-layer time-LSTM flowchart (T-LSTM).

The output of a T-LSTM is utilized as the input of the T-LSTM in the next layer at the same time step, as well as the recurrent input of the T-LSTM in the following time step in the same layer. The standard LSTM is a time-LSTM that performs temporal modulation via time recurrence by using the output of the previous time step as the input of the current time step. Figure 3.3 shows a multi-layer LSTM formed by stacking many layers of LSTM units together to boost modeling power.

At time step t, the vector formulas for computing the l-th layer LSTM units are as follows:

$i_t^l = \alpha(W_{ix}^l x_t^l + W_{ih}^l h_{t-1}^l + p_i^l \odot c_{t-1}^l + b_i^l)(1)$

$f_t^l = \alpha(W_{fx}^l x_t^l + W_{fh}^l h_{t-1}^l + p_f^l \odot c_{t-1}^l + b_f^l)(2)$

$c_t^l = f_t^l \odot c_{t-1}^l + i_t^l \odot (W_{cx}^l x_t^l + W_{ch}^l h_{t-1}^l + b_c^l)(3)$

$o_t^l = \alpha(W_{ox}^l x_t^l + W_{oh}^l h_{t-1}^l + p_o^l \odot c_t^l + b_o^l)(4)$

$h_t^l = o_t^l \oslash \odot(c_t^l)(5)$

where x_t^l

is the input vector for the l-th layer with

$$x = \begin{cases} \mathrm{h}^{l-1} & \text{if l>1} \\ \mathrm{s}_t & \text{if l=1} \end{cases} \quad (6)$$

- st is Handwrting input at time step t .
- the vectors ilt ,olt , flt , clt are the activation of the input , output , forget gates , and mamorycelles , respectively .
- hlt is the output of the time-LSTM .

- wlx and wlh are the weight matrices for the inputs xlt and the recurrent input hlt-1, respectively .
- bl are bias vectors .
- pli ,plt , plf are parameter vectors associated with peephole connections.
- The functions Ø and alpha are the logistic sigmoid and hyperbolic tangent nonlinearity .the operation $\odot represent selement - wise multiplication of vectors$.

According to Figure 3.3 , the output of a time-LSTM is utilized as the input of the time-LSTM at the same time step in the next layer and as the recurrent input of the time-LSTM at the same time step in the same layer. The output of the final hidden layer is utilized to forecast senone labels for senone classification. As a result, the same output is used for temporal model along the time axis and target discrimination along the layer axis.

Part 3: Generating the desired handwritten style

After training the RNN, a model is created, which is a file that has been trained to recognize certain types of patterns.

From the model, we can extract a set of patterns similar to the user's pattern by returning to the trained data and searching whether there is something similar or not. If the user's handwriting is somewhat poor, the system will generate dots instead of letters.

3.3 Conclusion

Deep learning (RNN and LTSM) is a powerful tool that can be used to create models to build systems never thought to be achieved using other techniques. In this chapter, we presented the general architecture of our system along with a detailed explanation of the used RNN. The next chapter will be dedicated to the implementation of our application and the obtained results.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Introduction

In this chapter we are going to define the objectives and requirements of the project. Besides of that, we are going to introduce most of the software we are using in this project. Along with a description of the algorithms implementation and the obtained results, followed with a discussion.

Requirements

The minimum requirements for this project are:

- Knowledge in Python programming language.
- Anaconda.
- dataset knowledge.
- Internet connection.

4.2 Work Environment

The used development environments,both hardware and software are described below

Development device information

For the realization of our project, we used anhp computer characterized by:

- Operating system: Parrot , Windows 10
- Processor: Intel Celeron CPU N3060 @ 1.60 GHZ 1.60GHZ.
- RAM: 4GB.

In order to install and run the application we used a Samsung smart phone with:

- Name of the Device: Samsung Galaxy A21s.
- Android version: 10
- RAM:4GB.
- Internal storage: 64GB
- Connection: LTE, 3G, 2G.
- WIFI:802.11a.

4.3 Definitions

4.3.1 Python

Python is a high-level programming language designed to be easy to read and simple to implement. It is open-source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems and has also been ported to Java and .NET virtual machines. Python is considered a scripting language, like Ruby or Perl and is often used for creating Web applications and dynamic Web content. It is also supported by a number of 2D and 3D imaging programs,

enabling users to create custom plug-ins and extensions with Python. Examples of applications that support a Python API include GIMP, Inkscape, Blender, and Autodesk Maya.

Scripts written in Python (.PY files) can be parsed and run immediately. They can also be saved as a compiled programs (.PYC files), which are often used as programming modules that can be referenced by other Python programs[45].

4.3.2 Parrot

Parrot Security OS is a Linux-based operating system for ethical hackers and penetration testers that was initially launched in 2013. Parrot OS may be viewed as a mobile lab for a variety of cyber security tasks, including pen testing, reverse programming, and digital forensics. It also comes with everything you'll need to safeguard your data and generate applications. Parrot OS is often updated and offers a variety of hardened and sandboxing features. Most devices that employ containerization technologies like Docker or Podman should work with the tools in the package. Parrot OS is extremely light and runs quickly on all PCs, making it a good choice for systems with outdated hardware or limited resources[46].

4.4 Platforms

To Create any system choosing the right platform is a must, and for our system, we chose for the software part :

4.4.1 PyCharm IDE

Definition

PyCharm is an integrated development environment (IDE) for the Python programming language. JetBrains, a Czech firm, created it. It includes code analysis, a graphical debugger, an integrated unit tester, version control system integration, and Django and Anaconda support for web development and data science. PyCharm is available in Windows, Mac OS X, and Linux versions. There is a Community Edition and a Professional Edition with more functionality that is available under a proprietary license.

4.4.2 Anaconda

Definition

Anaconda is a free and open source version of the Python and R programming languages for developing data science and machine learning applications (large scale data processing, predictive analysis, scientific computing), with the goal of making package management and deployment easier [39]. The conda4 package management system is in charge of package releases. Anaconda Browser is

a graphical user interface (GUI) provided with the Anaconda distribution that allows users to manage conda libraries, environments, and channels without having to use the command line. Navigator may also install, execute, and update libraries from the Anaconda Cloud or the local Anaconda repository in an environment. It runs on Windows, Mac OS X, and Linux.

Why Anaconda

- Ease to use.
- Cross-platform, Windows, macOS and Linux versions.

4.5 Implementation

After going through the platform and defining it and the requirement of the project, now we will explain the implementation of the system.

The libraries we have used in our project .

4.5.1 Flask Python

Flask is a web framework and a Python module that makes it simple to create web applications.It has a lot of amazing features, such as url routing and a template engine. It's a web application framework that uses the WSGI protocol.

```
pip install Flask
```

Figure 4.1: code install Flask

4.5.2 Building and Training the Recurrent Neural Network

Among the most important libraries that we will need in our project are the math library and the numpy library, so that the following libraries work to deal with the algorithm correctly to conduct the training process in an arithmetic and semi-accurate manner.

```
pip install numpy
```

Figure 4.2: code install numpy

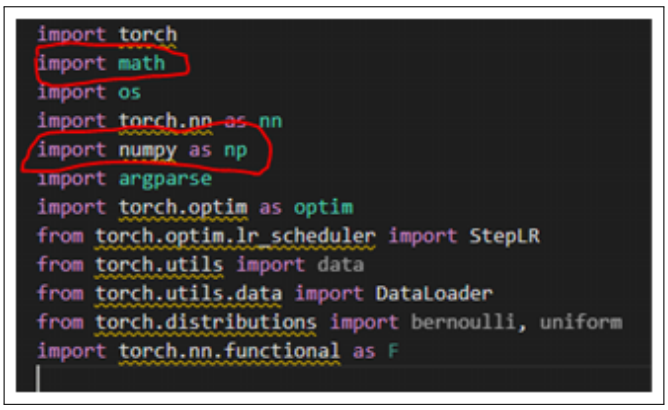

Figure 4.3: libraries used

After we have called the libraries, we will set up the RNN architecture. We need to adjust the learning rate, the sequence length, the maximum number of training intervals, the dimensions of hidden and output layers, the number of iterations we want to return to when doing backpropagation, and the maximum and minimum values that we will allow from our scaling.

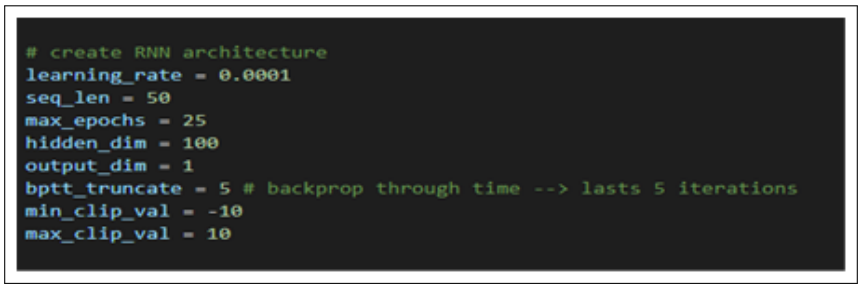

Figure 4.4: structure setting Rnn

4.5.3 Data Preparation for RNN Training and Testing

To acquire the RMSE, we'll need to install sklearn.You may use the following code to install sklearn:

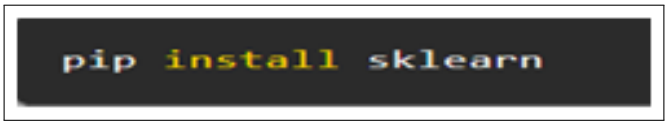

Figure 4.5: code install sklearn

This is the file test simple rnn.py. To begin, we'll import numpy, matplotlib.pyplot as plt, and math. Numpy and math are required for data processing, while matplotlib.pyplot is required for visualizing our series data. We'll also use sklearn.metrics' mean squared error to calculate the root mean square error (RMSE) at the conclusion of the training. Finally, we'll import the train and sigmoid functions from the simple rnn.py file, as well as the hyper parameters for the hidden dimensions, sequence length, and output dimensions.

```
import numpy as np
import matplotlib.pyplot as plt
import math

from sklearn.metrics import mean_squared_error

from training import train, hidden_dim, seq_len, sigmoid, output_dim
```

Figure 4.6: the file test simple rnn.py

4.5.4 RNN Train and Test Split on Sequence Data

On a sine wave, we will train our Recurrent Neural Network. Sine waves are data sequences that oscillate at a frequency of 2$pi. We'll put up our training and testing data when we obtain the sine wave data. Let's start with two empty lists, X for the data in the input sequence and Y for the next data point in the series.

Each X datapoint will be treated as 50 contiguous points in the series, and each Y datapoint will be treated as the next datapoint in the sine wave. The training data will be the first 100 points, and the validation data will be the following 50. After we've created the lists, we'll use np.expand dims to convert them to matrices.

```
sin_wave = np.array([math.sin(x) for x in range(200)])
# training data
X = []
Y = []
num_records = len(sin_wave) - seq_len # 150

# X entries are 50 data points
# Y entries are the 51st data point
for i in range(num_records-50):
    X.append(sin_wave[i:i+seq_len])
    Y.append(sin_wave[i+seq_len])

X = np.expand_dims(np.array(X), axis=2) # 100 x 50 x 1
Y = np.expand_dims(np.array(Y), axis=1) # 100 x 1

# validation data
X_validation = []
Y_validation = []
for i in range(num_records-seq_len, num_records):
    X_validation.append(sin_wave[i:i+seq_len])
    Y_validation.append(sin_wave[i+seq_len])

X_validation = np.expand_dims(np.array(X_validation), axis=2)
Y_validation = np.expand_dims(np.array(Y_validation), axis=1)
```

Figure 4.7: RNN Train and Test Split on Sequence Data

4.5.5 Constructing a Recurrent Neural Network Architecture

Let's put up the Recurrent Neural Network Architecture now that we've built up the training and validation data. All we'll do here is initialize the U, V, and W matrices, which represent the weights for the input to hidden layer, hidden to output layer, and hidden to hidden layer, respectively. We will receive the same result every time we run the test with the same seed. The seed setting method in the numpy.random library is the same as the one in the Python random library.

```
np.random.seed(12161)
U = np.random.uniform(0, 1, (hidden_dim, seq_len)) # weights from input to hidden layer
V = np.random.uniform(0, 1, (output_dim, hidden_dim)) # weights from hidden to output layer
W = np.random.uniform(0, 1, (hidden_dim, hidden_dim)) # recurrent weights for layer (RNN weigts)
```

Figure 4.8: Constructing a Recurrent Neural Network Architecture

4.5.6 RNN Training and Testing on Sequence Data

We must first train our RNN before we can test it. We imported the train function from the simple rnn.py file earlier in our test simple rnn.py file. Now we'll train our function using the randomized weight layers as well as the sine function's inputs and outputs.

```
U, V, W = train(U, V, W, X, Y, X_validation, Y_validation)
```

Figure 4.9: RNN Training and Testing on Sequence Data

4.5.7 RNN on a Sine Function Example (Training Fit)

To obtain predictions from our data, we must loop around each data point and perform a forward pass using the U, V, and W weights that we previously trained. To acquire the predictions, we perform nearly the same thing we did in calculate loss for each data point. We repeat the procedure, taking each dot product for the weights layers, obtaining the activation, and replacing the previous activation. We'll append the final output activation to the predictions after each run of the sequence data.

```
41  # predictions on the training set
42  predictions = []
43  for i in range(Y.shape[0]):
44      x, y = X[i], Y[i]
45      prev_activation = np.zeros((hidden_dim,1))
46      # forward pass
47      for timestep in range(seq_len):
48          mulu = np.dot(U, x)
49          mulw = np.dot(W, prev_activation)
50          _sum = mulu + mulw
51          activation = sigmoid(_sum)
52          mulv = np.dot(V, activation)
53          prev_activation = activation
54      predictions.append(mulv)
55
56  predictions = np.array(predictions)
57
58  plt.plot(predictions[:, 0,0], 'g')
59  plt.plot(Y[:, 0], 'r')
60  plt.title("Training Data Predictions in Green, Actual in Red")
61  plt.show()
```

Figure 4.10: RNN on a Sine Function Example (Training Fit)

The final step in running the data is to plot the training data in green and the actual data in red.

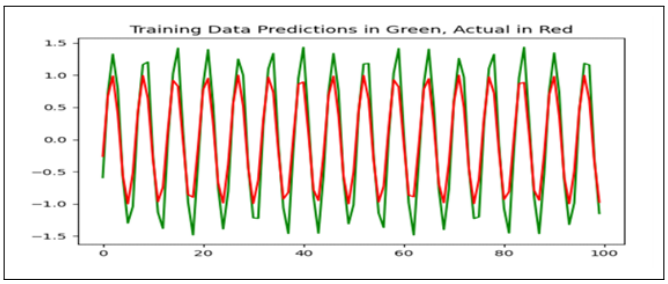

Figure 4.11: Training Prediction from RNN vs Actual Data Points

4.5.8 RNN on a Sine Function (Test Fit) Example

To test the Recurrent Neural Network, we'll perform the same thing we did to plot the training data, but this time we'll use the validation data instead of the training data.

```
# predictions on the validation set
val_predictions = []
for i in range(Y_validation.shape[0]):
    x, y = X[i], Y[i]
    prev_activation = np.zeros((hidden_dim,1))
    # forward pass
    for timestep in range(seq_len):
        mulu = np.dot(U, x)
        mulw = np.dot(W, prev_activation)
        _sum = mulu + mulw
        activation = sigmoid(_sum)
        mulv = np.dot(V, activation)
        prev_activation = activation
    val_predictions.append(mulv)

val_predictions = np.array(val_predictions)

plt.plot(val_predictions[:, 0,0], 'g')
plt.plot(Y_validation[:, 0], 'r')
plt.title("Test Data Predictions in Green, Actual Data in Red")
plt.show()
```

Figure 4.12: RNN on a Sine Function (Test Fit) Example

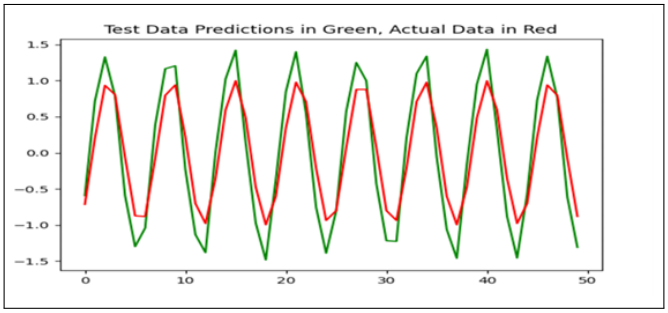

Figure 4.13: Training Prediction from RNN vs Actual Data Points on Validation Data

4.5.9 Examining the RMSE

The final step in evaluating our model is to calculate the root mean squared error of our function. All we need to do is apply the mean squared error function to the validation data results and validation predictions, and then square the result.

```
# check RMSE
rmse = math.sqrt(mean_squared_error(Y_validation[:,0], val_predictions[:, 0, 0]))
print(rmse)
```

Figure 4.14: Examining the RMSE

Our output is shown in the image below, including the training and validation losses.

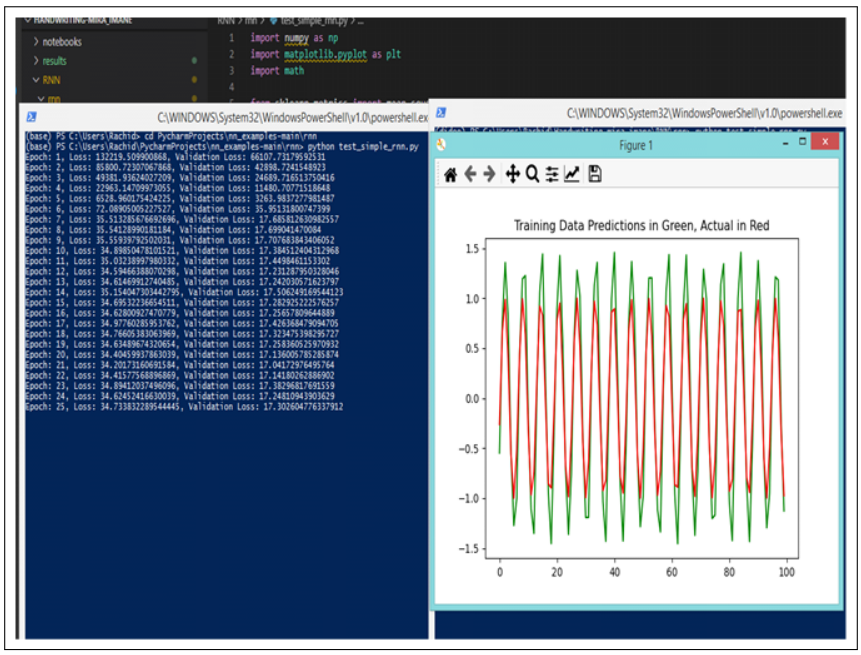

Figure 4.15: output, including the training and validation losses

4.5.10 Software

To manage the system we created a website application. The following descriptions show the interface of the system.

Application interface

Figure 4.16 below is the main interface of our application.

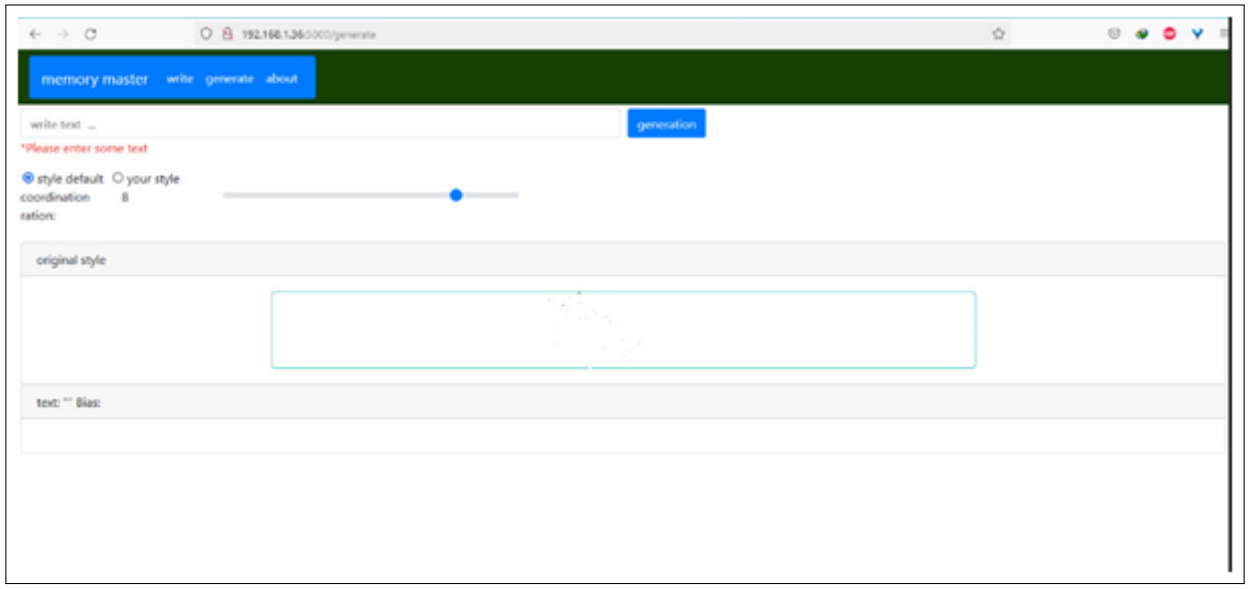

Figure 4.16: Application interface

Enter patterns

The user enters his own pattern either through drawing (figure 4.17, 4.18) or importing a pattern image(figure 4.19, 4.20), as well as entering via the keyboard (text).

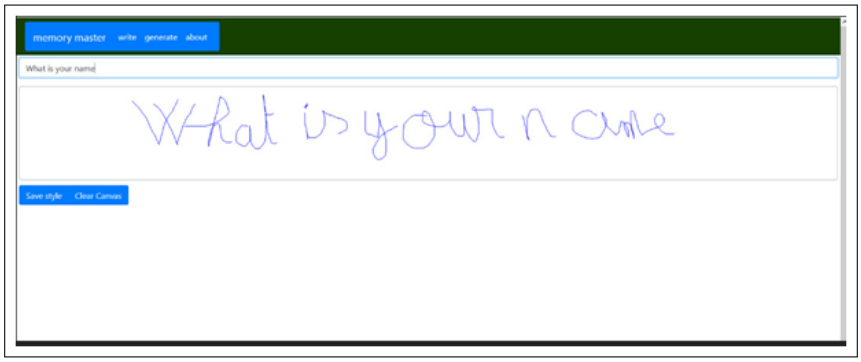

Figure 4.17: drawing font

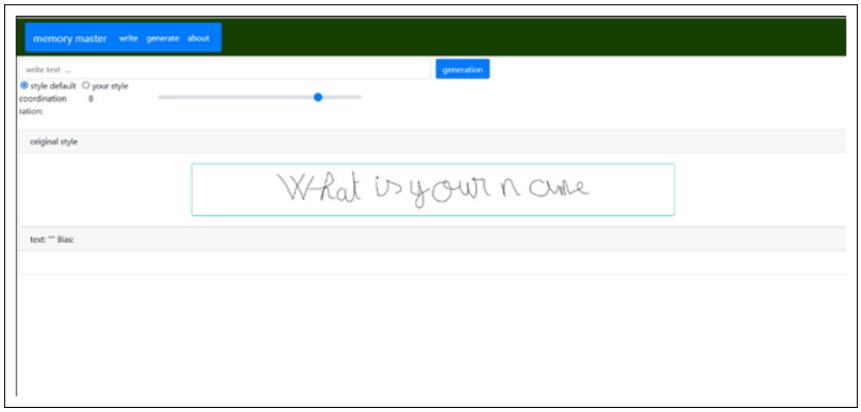

Figure 4.18: Style Show

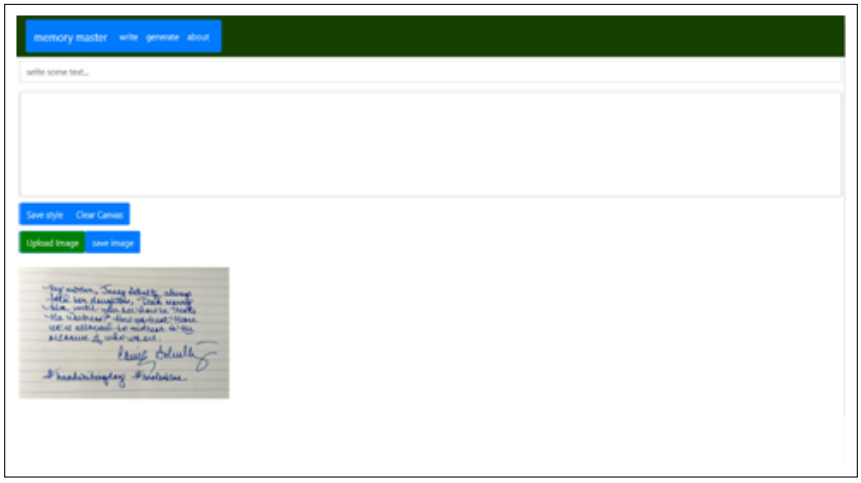

Figure 4.19: import font

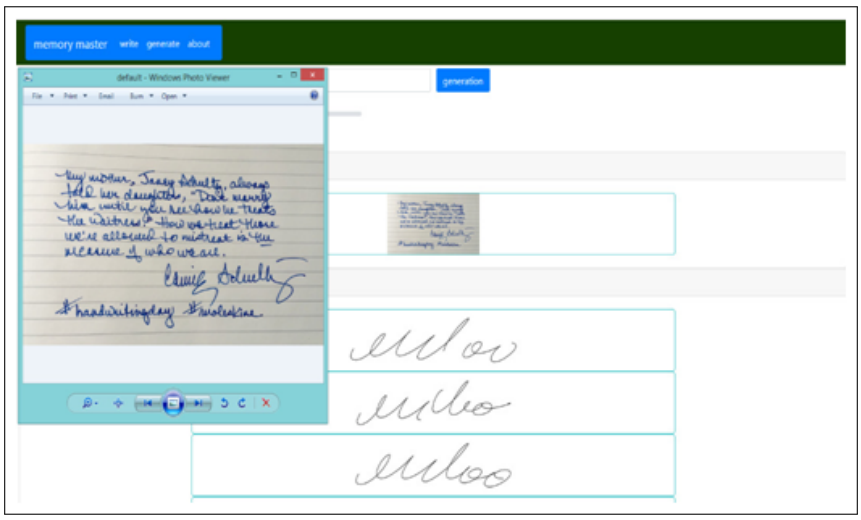

Figure 4.20: Style Show

Then it saves two files: inp.text, a computer-typed file, and style.npy, which is a Num Python file in which information is stored in the form of dtype, meaning that any device can read this file and it opens with the editor only, and it contains handwriting style so that the latter is saved in the app/static/uploads/..

Application result

It outputs a set of models (5 of them) imitating the user's style, as the following image represents(figure 4.21).

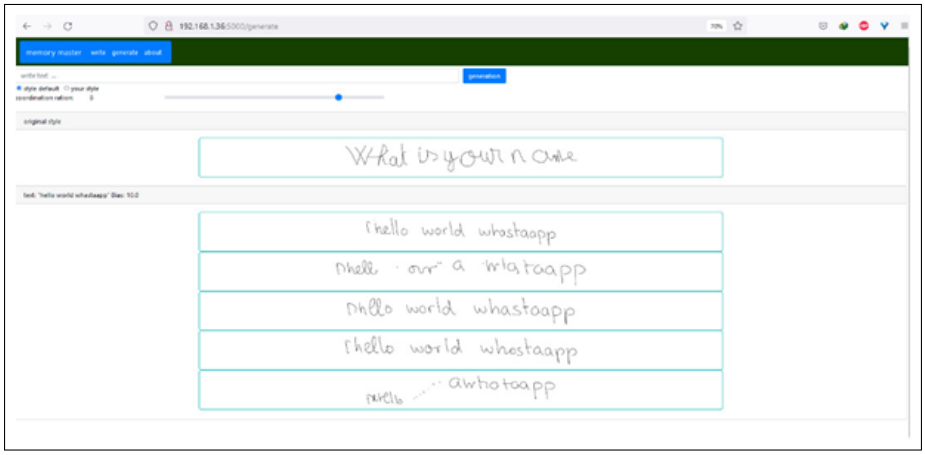

Figure 4.21: Application result

These patterns are saved in a file with the pattern entered by the user and written to the computer as shown in the figure below (figure 4.22).

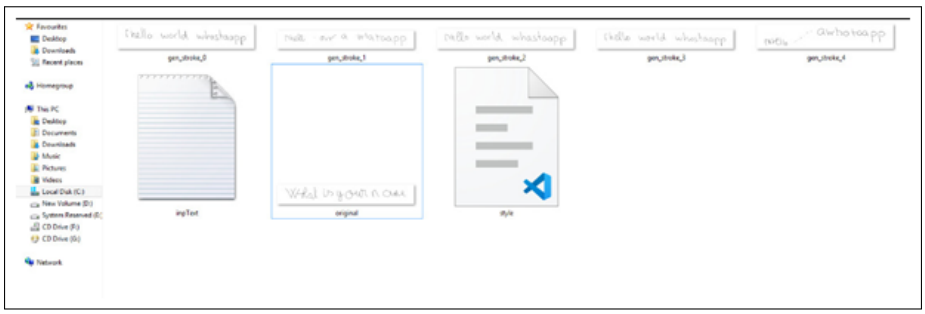

Figure 4.22: Save files

4.6 Results and Discussions

We were able to generate some handwriting using the RNN algorithm that allows to output a set of similar patterns for the user's font to some extent, and we created a data set containing a huge number of sentences.

When the method is combined with the proposed approach proposed in Chapter 3, we obtained good results.

We can note that the proposed method did not work in all fonts, because it is possible that a number of points may appear in the result and that indicates a lack of understanding of the user's font to a certain degree or a lack of training.

4.7 Conclusion

In this chapter, we discussed the platform used and the reason for choosing it. After that, we did go through the implementation. Passing by the result and ending with a discussion, clarifying some of the advantages and inconveniences of the implemented system.

GENERAL CONCLUSION

In this thesis, we were interested in the process of generating handwriting fonts using the RNN algorithm. An ideal system does not exist and a good system requires low rejection rates and very low error rates. The purpose of this system is to create patterns similar to the user's handwritten pattern. The system uses personal input lines, recognizes it, and uses RNN's trained data in order to create a model that generates a set of fonts similar to the user's inputs. Pycharm platform was used to implement the website application. In the end, we have achieved a system capable of:

- Recognizing User's handwriting styles.
- Generating different font's styles.

Through the different chapters, we explained the architecture of the system, the difficulties encountered, and some of the hypothesis and objectives that were achieved. Our system gives encouraging results in terms of the performance and efficiency.

In the future, we hope to adapt our system to manage other different applications such as face recognition, voice recognition and eventually generate humanly pleasing outputs, but before any of that, there are still many lacking in our system to be completed and solved on various levels.

BIBLIOGRAPHY

[1] J.-W. Lin, R.-I. Chang, and C.-Y. Hong. Deep learning for information acquisition in bayesian inferences.

[2] M. Saeed, Z. Al Aghbari, and M. Alsharidah. Big data clustering techniques based on spark.

[3] V. Mnih, K. Kavukcuoglu, D. S. A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, and G. Ostrovski. Human-level control through deep reinforcement learning.

[4] M. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Van Essen, A. Awwal, and V. Asari. A state-of-the-art survey on deep learning theory and architectures.

[5] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath. Deep reinforcement learning.

[6] Sifre, L. Driessche, G. van den; Schrittwieser, and Julian. Mastering the game of go with deep neural networks and tree search.

[7] Bengio and Yoshua. Learning deep architectures for ai.

[8] Szegedy, Christian, Alexander, and Dumitru. Deep neural networks for object detection.

[9] Hof and R. D. Is artificial intelligence finally coming into its own?

[10] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks.

[11] D. Zhou. Theory of deep convolutional neural networks.

[12] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. Deep learning.

[13] L. Alzubaidi, M. Fadhel, O. Al-Shamma, J. Zhang, J. Santamaría, and Y. Duan. Towards a better understanding of transfer learning for medical imaging.

[14] H. Hewamalage, C. Bergmeir, and K. Bandara. Recurrent neural networks for time series forecasting.

[15] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks.

[16] C. Goller and A. Kuchler. Proceedings of international conference on neural networks.

[17] G. Urban, N. Subrahmanya, and P. Baldi. Inner and outer recursive neural networks for chemoinformatics applications.

[18] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification.

[19] B. Alipanahi, A. Delong, and M. T. Weirauch. the underside of a breakthrough technology.

[20] sisu.ut.ee. Accessed: 06-2022. [Online]. Available: https://sisu.ut.ee/imageprocessing/book/1

[21] H. Haven. Ocr document. April 15, 2016.

[22] techtarget. Accessed: 06-2022. [Online]. Available: https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition?amp=1&fbclid=IwAR0CWgyw5c-Mgi9W6vA8iqu31NcamJnQm8ZwMdvQMpTybo0jTLKk0eog7Ss

[23] Schantz and F. Herbert. The history of ocr, optical character recognition. July 1, 2015.

[24] Dhavale and S. Vikrant. Advanced image-based spam detection and filtering techniques. March 10, 2017.

[25] d'Albe. On a type-reading optophone.

[26] Schantz and H. F. "the history of ocr".data processing magazine. June 27, 2015.

[27] d'Albe. Extracting text from images using ocr on android. November 20, 2008.

[28] d'Albe and R. Smith. Ocr on google glass. October 23, 2014.

[29] R. Smith. An overview of the tesseract ocr engine.

[30] Mehdi. How the best ocr technology captures 99.91% of data.

[31] Assefi and Mehdi. "optical character recognition (ocr). December 2016.

[32] D. IMPEDOVO and G. PIRL. Dynamic handwriting analysis for the assessment of neurodegenerative diseases.

[33] G. SINGH and N. SHAH. Handwriting change as a psychiatric symptom.

[34] GONZALEZ and RAFAEL. Digital image processing.

[35] handwriting analysis for mental disorder detection. Accessed: 06-2022. [Online]. Available: https://www.ijert.org/research/handwriting-analysis-for-mental-disorder-detection-IJERTV10IS040233.pdf

[36] data scientest. Accessed: 06-2022. [Online]. Available: https://datascientest.com

[37] L. Zhou and Z. Sun. Learning and generation of personal handwriting style chinese font.

[38] A. Zong and Y. Zhu. Automating personalized chinese handwriting generation.

[39] What is anaconda. Accessed: 06-2022. [Online]. Available: docs.anaconda.com

[40] Y. E. Irfan Ahmad, Radwan Abdel-Aal. Handwriting synthesis.

[41] J.-W. Lin, R.-I. Chang, and C.-Y. Hong. Complete font generation of chinese characters in personal handwriting style.

[42] S. Tellex, J. Tompkin, and A. Kotani. Generating handwriting via decoupled style descriptors.

[43] builtin. Accessed: 06-2022. [Online]. Available: https://builtin.com/data-science/recurrent-neural-networks-and-lstm

[44] medium. Accessed: 06-2022. [Online]. Available: https://medium.com/geekculture/understanding-the-paper-generating-sequences-with-rnns-by-alex-graves-18635cdd32be

[45] techterms. Accessed: 06-2022. [Online]. Available: https://techterms.com/definition/python.

[46] simplilearn. Accessed: 06-2022. [Online]. Available: https://www.simplilearn.com/tutorials/cyber-security-tutorial/parrot-os-vs-kali