FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Scripty: Cursive Handwriting Recognition using Deep Learning Techniques

Prepared by

Noura Abu Laban	1161803
Mahmoud Hussain	1160778
Rawan Ghanem	1160351

Supervised by

Dr. Mohammed Hussain

Graduation Project submitted to the Department of Electrical And Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Systems Engineering

BIRZEIT
January -2021

المستخلص

من بين أكثر المجالات الشائعة في المجالات العلمية هو الذكاء الاصطناعي، وهو مجال من ضمن مجالات علوم الحاسوب الذي يركز على أتمتة القرارات والتفكير والعمل كالدماغ البشري. وقد تم استخدامه في عدة حقول سواءً في الطب أو التجارة الإلكترونية أو حتى مجال الترفيه ،وقد أُجري الكثير من البحث والتطوير في هذا المجال، والذي يركز في معظم الأحيان على مفهومين شائعين جداً و هما التعلم الآلي والتعلم العميق، وقد حاز التعلم العميق على شعبية بدرجة أكبر بسبب تميز أدائه خاصة في التعامل مع كميات هائلة من البيانات.

يجد الشخص العادي صعوبة كبيرة في فهم المخطوطات اليدوية المكتوبة بخط الآخرين خاصة عندما تكون هذه المخطوطة هي وصفة طبية حيث تعتمد صحتهم عليها ؛ نظراً لأن من المتفق عليه على نطاق واسع أن الوصفات الطبية عادة ما تكون مزيجًا من الاختصارات التي قد تكون اختصارات لاتينية ومصطلحات طبية التي قد يجد الشخص العادي صعوبة في فهم محتوى الوصفة الطبية المكتوبة له، الأمر الذي يجعله قلقاً فيما إذا كان الدواء الذي يحصل عليه هو ما يحتاجه حقًا.
عادةً ما يتم الاعتماد على الصيدلاني في قراءة الوصفة الطبية المعطاة والتي تكون مكتوبة بخط اليد غير المقروء، لتوفير الدواء المناسب. لكن ماذا لو لم يكن الشخص في الصيدلية صيدلانياً مُعتمداً ؟، حيث سبق ونتج عن ذلك أضراراً خطيرة كما تشير الدراسات إلى وفاة أكثر من 7000 مريض سنوياً.

لطالما كانت نماذج تحليل الخطوط سواءً المطبوعة والمكتوبة في الأفق منذ فترة طويلة من الوقت، و قد ثبت أنها قادرة على كشف الحروف المكتوبة بخط اليد غير المتواصل ، بيد أنها لم تكن كافية للتعرف على خط اليد المتواصل، ولا سيما في الوثائق الكبيرة، دون وجود سياق واضح.

في هذا التقرير، نستكشف سلسلة من نُهج التعلم العميق في الكشف عن الكلمات المكتوبة بخط اليد، حيث توصلنا إلى نموذج نهائي يتعرف على الوثائق المكتوبة بخط اليد بطريقة ابتكارية سريعة. يُعتبر Scripty نموذجاً للتعرف على خط اليد المكتوب باستخدام التعلم العميق وأنماط تعلم الآلة من أجل توفير طريقة فعالة لحل المشكلة المذكورة أعلاه.

Abstract

One of the most spoken about scientific fields has been Artificial intelligence (AI), it is the field of computer sciences that emphasize the development of intelligence machines, thinking and working like humans. It has recently been a key factor in development and automatic decision making in any scientific field. Those fields may be medicine, e-commerce or even the entertainment field. Countless research and development has been done on this field, but it really concentrates on two very popular and important concepts Machine Learning and Deep Learning. However, deep learning has been more popular due to its higher performance especially with huge amounts of data.

The average person may find it very difficult to understand other people’s cursive handwriting. And this may be very critical if it's a doctor’s prescription where their health depends on it. For example, in cursive very complicated medical prescriptions, we rely on the pharmacist to read the given prescription and to provide the right medicine. However, what if the person in the pharmacy is not a certified pharmacist who knows a medicine right away, or what if they could not read the illegible handwriting of the doctors, serious damage has been resulting of the latter, as statistics done by non-profit organization Medscape India showed that the sloppy handwriting of physicians is responsible for more than 7,000 deaths each year [1].

OCR models have been around for a while now, and they have proven to be able to detect typed or separated handwritten characters. However, they have fallen short to recognize cursive handwriting, especially in big documents, with little to no context at all. Finding a deep learning approach to recognize cursive handwriting using object detection algorithms would open new doors in OCR systems.

In this report, we explore a series of deep learning approaches in handwritten word detection and we finally reached a model that best recognizes cursive handwritten documents in a fast innovative way. Scripty introduces a deep learning cursive handwriting recognition model using Faster RCNN object detection algorithm in order to provide an efficient way to solve the problem mentioned above. Scripty is able to take a picture of the handwritten document, process the image, analyze it and then predict the correct words in this document.

Table of contents

List of Figures

List of Tables

List of Abbreviations

Original Sequence	**Abbreviation**
Artificial Intelligence	AI
Support Vector Machine	SVM
Convolutional Neural Networks	CNN
Optical Character Recognition	OCR
Region Based Convolutional Neural Networks	R-CNN

Region Proposal Network	RPN
Artificial Neural Networks	ANN
Character Recognition	CR
Weighted Finite State Transducer	WFST
Convolutional Recurrent Neural Network	CRNN
Recurrent Neural Networks	RNN
Long Short-Term Memory	LSTM
Rectified Linear Unit	RELU
Application Program Interface	API
National health insurance	NHI
International Conference on Document Analysis and Recognition	ICDAR
Hidden Markov Model	HMM
International Association for Pattern Recognition	ICPR
Visual Object Classes	VOC
Average Precision	AP
mean Average Precision	mAP
Region of Interest	RoI
Intersection of Union	IoU
Non-Maximum Suppression	NMS
Hierarchical Data Format 5	HDF5
Generative adversarial networks	GAN

Chapter 1: Introduction

1.1. General introduction

With the rapid development of smart devices, Artificial Intelligence (AI) is being used widely to develop the internet services, the transmission and storage of information, sensors and microcontroller devices, automation of decision making, etc. Deep learning which is an essential branch of AI can be used in object detection, translation of speech-to-text, object recognition, information retrieval from media and data analysis with multiple aspects and it has been extensively used for handwriting character recognition over the past few years.

Despite the wide use of technology nowadays, people around the world tend to take their notes manually: with pen and paper. In this digital world, saving notes written in cursive handwriting is very inefficient and difficult to access them, search through them or share them with others .This is especially important when the notes are doctors prescriptions or bank cheques.

Reading a handwritten document is a very important thing especially if this document is a medical prescription for example. A misread prescription can lead to mistreatment and may cause death, according to non-profit organization Medscape India, which spearheads an expedition to address the issue, the number of fatal incidents due to unreadable prescriptions is internationally on the rise, and moreover, they reported that the sloppy handwriting of physicians is responsible for 7,000 deaths each year [1].

Bank cheques are another example of cursive handwritten documents that can be very confusing for the reader and misleading in some cases. Having the amount of money written on the cheque or the signature of any party involved being written in cursive handwriting makes it impractical in the current world.

So, a lot of important information gets lost or does not get reviewed because of the reason that documents never get digitalized (transferred to digital format). We have thus decided to try to find a reasonable solution for this problem in our project because we believe the significant great ease of management of digital text compared to written text, which will help people to access, search, share their documents more effectively and analyze their records, while still using their preferred writing method.

Learning through deep models has gained a very big interest. Many deep learning architectures such as deep neural networks, recurrent neural, and deep convolutional neural networks have been applied to a vast range of fields like computer vision, natural language processing, or automatic speech recognition.

The big advantage of deep learning appears in discovering the features in the input data in multiple levels of representation, which means, higher level features represent more abstract semantics of the data. Deep learning techniques have been used to produce state-of-the-art results on many difficult tasks. For example, computer vision tasks such as visual content-based image classification are very complicated and challenging tasks , mainly because of intra-class variation, variation of perspective, variation of size ,occlusion.

Many trials have also made to recognize handwritten digits using machine learning techniques such as K-nearest neighbors, linear / nonlinear classifier, and support vector machines (SVMs) on the MNIST dataset [2], or deep learning models such as deep convolutional neural networks (CNN) and deep neural networks [3-5]. Deep learning algorithms took the top place in the object recognition field because of the great performance improvement that they have provided [6], [7]. These algorithms made it possible for machines and computers to model our environment very well to show intelligence.

In our project, we are introducing a cursive handwriting recognition model using deep learning techniques to solve the problems mentioned above. Scripty is able to use object detection algorithms to take a picture of the handwritten document, process the image, analyze it and then predict the correct words in this document with mean Average Precision (mAP) of 30%.

1.2. Motivations and problem statement

For this project we will be introducing a new model for cursive handwriting recognition using deep learning techniques. Considering that deep learning has become very important and very practical, especially in detection models.

We believe the problem of Cursive handwriting recognition is really important to tackle for many reasons; firstly, Optical Character Recognition (OCR), or the process of identifying letters and words for images of handwritten or typed characters, is a heavily researched area. While OCR is widely used, there is not a generally accepted, contemporary method for performing OCR. Moreover, commercial OCR platforms often take in a large amount of training data on an individual before testing the individual's handwritten text. It is uncommon for OCR platforms to perform well on handwriting from an individual for which training data is nonexistent. [8]

One other important reason, and which we will be focusing much on later in our future works, is the detection and recognition of Cursive doctors handwritten prescriptions. There have been 300 experiments around this topic, and from the 300 measurements, 88% of the doctors read the prescriptions correctly, compared with 82% of the nurses and 75% of the pharmacists. A potentially fatal error was lorazepam injection 4 mg, which was read as 40 mg (lethal dose) by 20% of healthcare workers [8].

Preventable medication errors affect more than 1.5 million Americans annually. These errors are caused by unclear abbreviations and doses, and illegible handwriting. Therefore, a plea has been made for doctors to use digital notes and prescriptions to prevent these errors [9]

When it comes to neural networks, they may seem like there are many options to choose from, and for our approach towards detecting cursive handwriting but we were motivated to choose the faster region based convolutional neural networks (R-CNN). This is because we wanted to use the most effective algorithm, which means the most efficient and fastest way to train and test our data.

One clear option for such models may be CNNs, however CNNs require training data to classify images but often perform well on new images due to their complex model weights. However thought, In order to achieve word recognition, neural networks must also perform object detection in conjunction with object classification.

Table 1.1 shows the differences in speed between the main object detection algorithms that will be explained further later on in this report. This shows that Faster RCNN is the fastest and most efficient.

	R-CNN	Fast R-CNN	Faster R-CNN
Test Time per Image	50 Seconds	2 Seconds	0.2 Seconds
Speed Up	1x	25x	250x

Table 1.1 Speed comparison for object detection algorithms

Based on the reasons mentioned above, we were motivated to develop an innovative model to solve the problem. In this report, we go beyond the accomplished works to answer the following research questions:

• To what extent can a Faster RCNN model with random cursive handwritten images detect the words in a picture?
• Instead of relying on a complex detecting process, is there any possibility to detect cursive handwritten documents such as doctor's prescriptions in a faster, more accessible way?

1.3. Contributions

For this project we have chosen to use an international open-source dataset for cursive handwritten words and sentences called IAM. In addition to cursive handwritten letters, a dataset called EMNIST.

Initially we wanted to make the dataset ourselves by getting a huge amount of prescriptions from local hospitals and clinics. However, we find it quite hard to get our hands-on local datasets especially at those uncertain times where the coronavirus has hit hard everywhere in the world. This means visiting hospitals and clinics to get those prescriptions may not be the best choice. Which makes using already created datasets by previous researches will be very useful.

In conducting our data exploration we are trying to think in the categories of classification, clustering, regression, and ranking; for classification, we want to make a multiclass classification algorithm for different handwritten letters and characters.as for clustering, we want an algorithm to find the rules of classification and the number of classes. This usually happens when you need to segment your customers and tailor a specific approach to each segment depending on its qualities. Regression is used for an algorithm to yield some numeric value. Ranking is used because some machine learning algorithms just rank objects by a number of features. And will be very helpful for our future alternative medicine recommender feature [10].

The IAM dataset was used, in addition to the EMNIST dataset. Both datasets needed to be prepared in a certain format to be usable in Scripty. We used python programming language for labeling and formatting. Then made specific annotation files for each dataset, in order to be able to train and test on Scripty. Further explanation on this process will be explained in chapter 3.

As we have a big data set, we will use deep learning techniques instead of classical machine learning techniques since it gives higher accuracy as the figure1.1 shows.

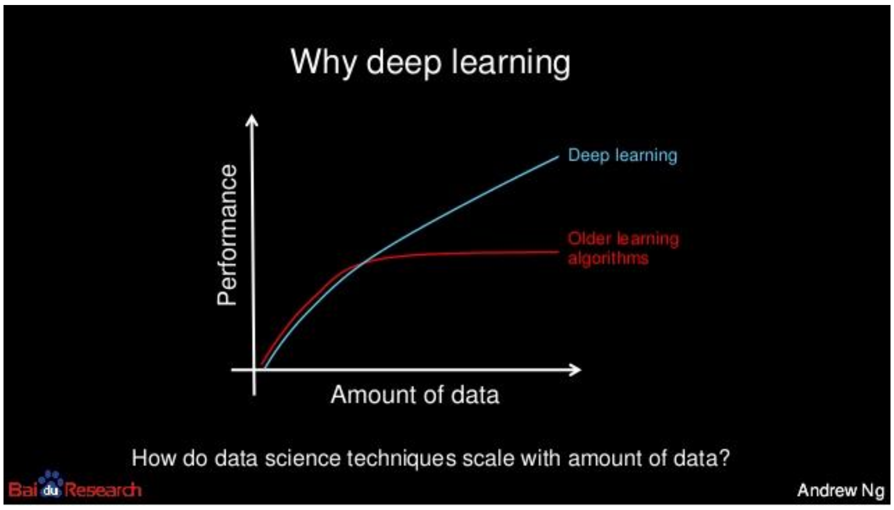


Figure 1.1. Deep Learning Performance [11]

Many deep learning algorithms will be tested in order to compare between their performance and accuracy, the following algorithms that were tested:

1- CNN Architecture: a primary class of deep neural networks, most commonly applied to analyzing visual images. As known, CNN is a good algorithm for classification but we faced many problems when using it in our project, since CNN wasn't able to localize each letter correctly, especially when there was a large over -lapping between characters.

2- RCNN: A group of machine learning algorithms for object detection that uses selective search to extract boxes in the image and checks whether those boxes contain an object.

3- Fast RCNN is based on RCNN however in fast RCNN we feed the image to a CNN model to find regions in which the boxes of objects may be in.

4- Faster RCNN is a modified version of fast RCNN in which we will be using in Scripty. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses "Region Proposal Network (RPN).

1.4. Thesis Organization

The rest of this report is organized as follows. Chapter 2 presents related work with discussing their methodology, features, experiments and limitations. Chapter 3 describes our datasets, object detection algorithms, Scripty architecture and methodology through detailing the collaborative framework, defining and formalizing the problem we address. Chapter 4 discusses the procedure of Scripty and the evaluation of our results. Chapter 5 concludes the thesis with providing more details about challenges faced and the future work plan.

Chapter 2: Background and Related Works

Handwriting recognition is known as one of the most leading applications of pattern recognition and machine learning. Although having some limitations, handwriting recognition systems have been used as an input method of many electronic devices and assists in the automation of many manual tasks that require processing of handwriting images.

So, the first step taken before designing and extracting the features of Scripty is collecting useful information from different resources. This step was helpful in detecting the features that should be included in the application to guarantee that it will serve users with the satisfying results they need and with the best efficiency.

We dealt with various types of resources, such as related papers, websites, mobile applications, questionnaires and asked people from the category we are interested in, which are pharmacists.

2.1. Questionnaire

A questionnaire was made using Google form in order to get the general pharmacist's opinion , since medical prescriptions are a good example of cursive handwriting documents that we develop Scripty to read . The questionnaire includes various subjects in order to develop a model that will satisfy all their needs and desires, where 70 pharmacists filled the questionnaire. The questionnaire contained the questions shown in figures 2.1, and 2.2, it provided us with the proper platform the users are comfortable working with and the features they require.

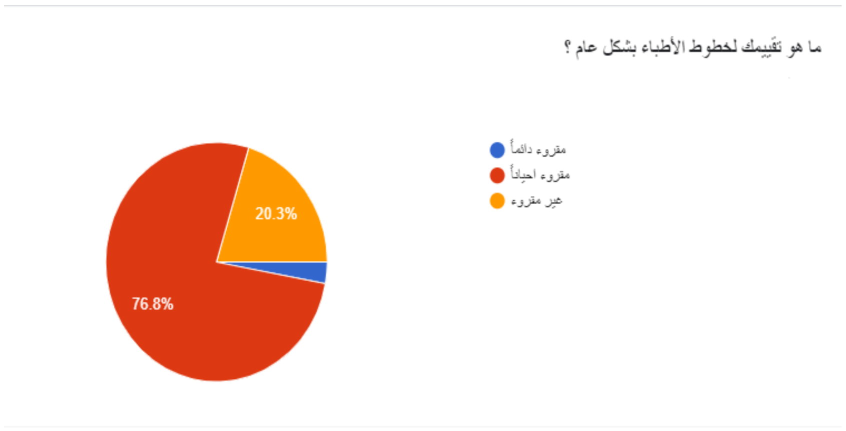


Figure 2.1. Questionnaire question 1

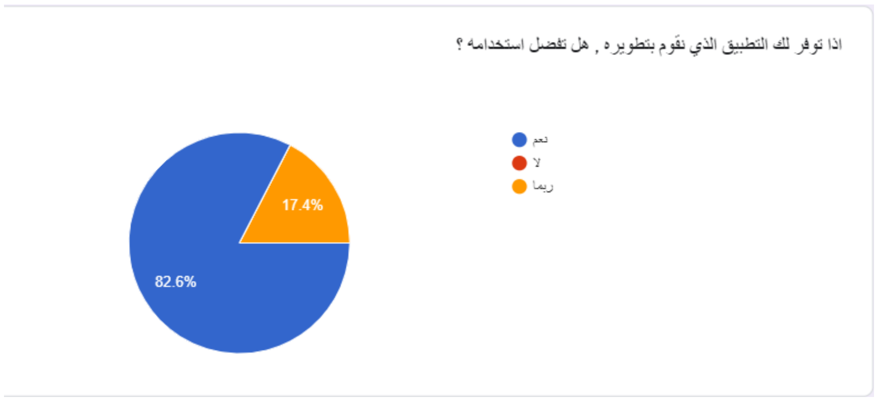


Figure 2.2. Questionnaire question 2

2.2. Related works

In this section, we will present some applications and papers that's related to our application, and explain the differences between our solution and the already accomplished work.

2.2.1 Websites and mobile applications

OCR Text Scanner [12] is an application that lets you turn images to live text in a flexible way. It detects text from an image with high accuracy. Moreover, OCR technology provides a way to recognize text on an image.

Moreover, the application edits perspective and borders, applies filters, adjusts brightness and contrast. Also, it offers many features such as: PDF converter, Image to Text converter, Extracts texts from a single page and Supports text editing & sharing

Pen to print [13] is another app that looks into handwriting OCR. This app converts and scans in addition to recognizing handwritten notes into digital text. This application targets students, businessmen, secretaries and more users. It specifically needs an iOS version of 9.0 or later device to run the app smoothly. Pen to print also offers an In-app Purchases for more features.

MediPic [14] is another android application that uses OCR techniques to scan the handwritten and return the input image's text as a string. Using the app will decrease the cases of misinterpretation of medicine names, decrease the pharmaceutical errors, and it will be helpful for patients, because it will help them know more about the medicines they are about to take

2.2.2 Reviewed papers

In a paper **by Attigeri**, **et al [15]**. Attempts at recognizing English handwritten characters without any feature extraction are done. The attempted system here uses a multilayer feed-forward neural network. The character set here basically contains 26 alphabets. This neural network will be trained on 50 different character data sets.

In this system, every character would be resized into 30x20 pixels exactly to then be used for training. The results of this system show good recognition rates compared with systems using feature extraction for handwritten character recognition.

The main objective of this research is to find a new solution for handwritten text recognition of different fonts and styles by improving the design structure of the traditional artificial neural networks (ANNs) [16]. The resulting recognitions will be fed into OCR as an electronic translation of images. ANNs have proved in past studies to be very good at recognizing printed digits and characters, however, it proved to be not effective when it comes to word recognition.

Scripty in this paper states that a typical handwriting recognition system consists of pre-processing, segmentation, classification, and post-processing stages. The general schematic diagram of the recognition system is shown in Figure 2.3.

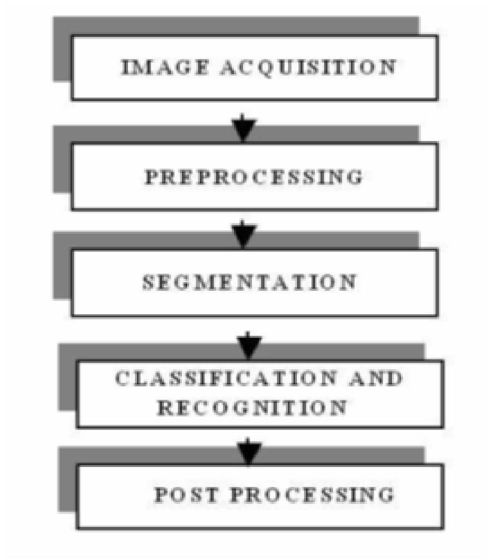


Figure 2.3. Attigeri's Proposed System Architecture

.

To start off this research suggests that Image acquisition should be done first which normally means that the systems acquire an input of a scanned image. Secondly, pre-processing is done. Which is basically a series of operations performed on the scanned input image.

When preprocessing is done, one of the most important stages is next. Which is segmentation, the full image of a sequence of characters will be decomposed into sub images of individual characters. And in this particular system, this segmentation will be done by assigning a number to each character using a labeling process.

Following up will be classification and recognition, this stage is the decision-making stage of the whole system. A feed-forward backpropagation neural network is used.

The final stage will be post-processing which is basically printing the corresponding recognized characters in a text. The latter will be done by calculating the equivalent ASCII values using the recognition index of the test samples.

In this proposed system, the raw data is subjected to a number of preliminary processing steps to make it usable in the descriptive stages of character analysis. Preprocessing aims to produce data that are easy for OCR systems to operate accurately.

It is also worth mentioning that the main objectives of pre-processing would be noise reduction, in addition to binarization, edge detection, dilation, filling, stroke width normalization, skew correction, slant removal, and segmentation of the processed image as shown in Figure 2.4.

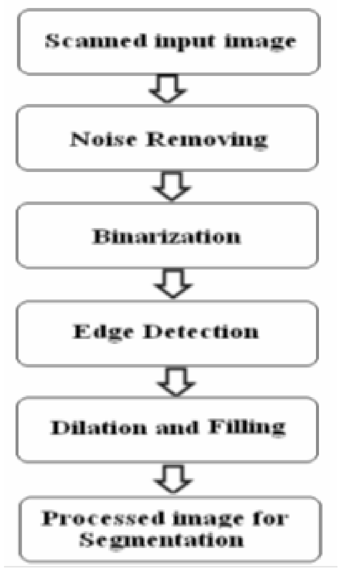


Figure 2.4.pre-processing steps

This paper also sheds light on the two main types of segmentation in character recognition, the external and the internal segmentation.

External segmentation decomposes the page layout into its logical units. On the other side, Internal Segmentation is an operation that seeks to decompose an image of a sequence of characters into sub-images of individual symbols.

As a result, Scripty will be less complex compared to the offline methods using feature extraction techniques.

Another paper by **Dhande, et al** [16]. Provides a general view about character recognition for cursive handwriting recognition. It provides how machines can observe the environment, distinguish character of interest from their background and make correct decisions about the character, which is a simulation of the human reading process.

The provided system identifies the text stored in an image in specific format. The text is converted into machine readable format. Where each pixel representation of a letter is converted into the equivalent character representation using Character Recognition (CR).

The mathematical model of the system is build based on the following function

$$F(x) = F(I \rightarrow O \mid P) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

Where f(x) represents a function, which is to map a set of inputs represented by I to a set of outputs represented by O given a set of operations.

Figure 2.5 shows the main stages and sub stages of the system’s architecture, where each stage represents an operation.

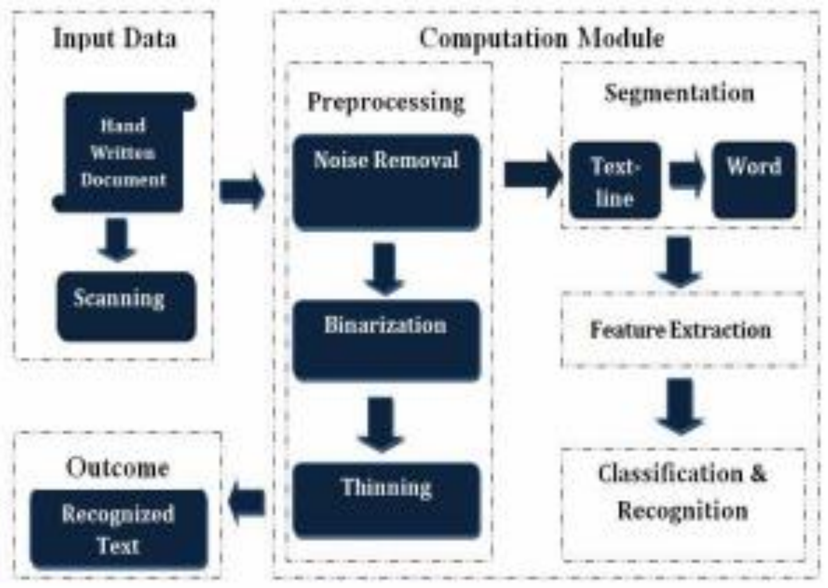


Figure 2.5. Main Stages of System Architecture

The System is executed in many stages. First of all, Image Acquisition, where in this stage, the system will get the text document, or the image contains the text, where the image is in specific format.

Second stage is preprocessing, which is an important stage of input before starting any system. At this stage, the system will perform some actions over the input image, it removes the noise using median filters, then it will convert the image into a binary image, and a thinning process using Guo-Hall Algorithm is applied over the binary image to reduce the extra information.

This paper also presents two methods to recognize words and lines in the Segmentation stage, where this stage will carry out on the pre-processed image. The horizontal projection histogram method for line segmentation, and the vertical projection histogram method used for word segmentation. Horizontal projection method is used to segment a line from a paragraph, where the most rising section is assumed as threshold, then, the heights of other sections are checked whether it's great or equal to threshold, then each line is segmented from the binary image. The output of line segmentation will be the input of word segmentation. Inter-word and intra-word distances are calculated, then, the vertical projection histogram will calculate the threshold values, and then it will compare the distances and decide how to treat it.

After that, the features of vertical projection results will be extracted using the Convex-Hull algorithm. Using Convex-Hull, 25 features are designed. Six different topological features will be calculated, and the total feature count as 125 i.e. 25 features for the overall image and 100 features in all four sub-images

In order to get a clear result, the system will classify and recognize the words using SVM, and do some post processing on the results, where it will be matched with the dictionary to improve the accuracy, and it will be displayed in specific format.

This work deals with various techniques, which is used in character recognition for English handwriting, the accuracy achieved by the system is 85% in recognition.

One more paper by **Yann Soullard, et al [17]**, proposed a model for handwriting recognition by the implementation of the optical model and the language model. Their approach was to evaluate the READ dataset and they obtained good results.

The system architecture is shown in Figure 2.6. First, a generic handwriting recognition system made up of an optical model (PG(O|W)) and a language model (PG(W)). This model is trained on the generic data set, to model the variance between different writing styles, languages and document images background. Both generic models -optical and language- render as initial models for the adaptation process intended to design specific optical (PS(O|W)) and language models (PS(W)).When decoding, the outputs of the optical model, whether generic or specific, are analyzed by the suitable language model by exploring a search graph due to a Weighted Finite State Transducer (WFST) and using the Viterbi algorithm. The method selects the sentence ŵ maximizing the a posteriori probability P(W|O) among all possible sentences W by using the Bayes formula:

$$\hat{W} = \arg\max P(W|O) = \arg\max P(W|O)\, P(W) a\, \beta|w| \ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

where O is the observation sequence coming from the input image, P(O|W) is the probability of the observation sequence given the sentence W computed due to the optical character model and P(W) is the prior probability of the sentence computed using the language model. The two hyper-parameters, αand β, are the language model.

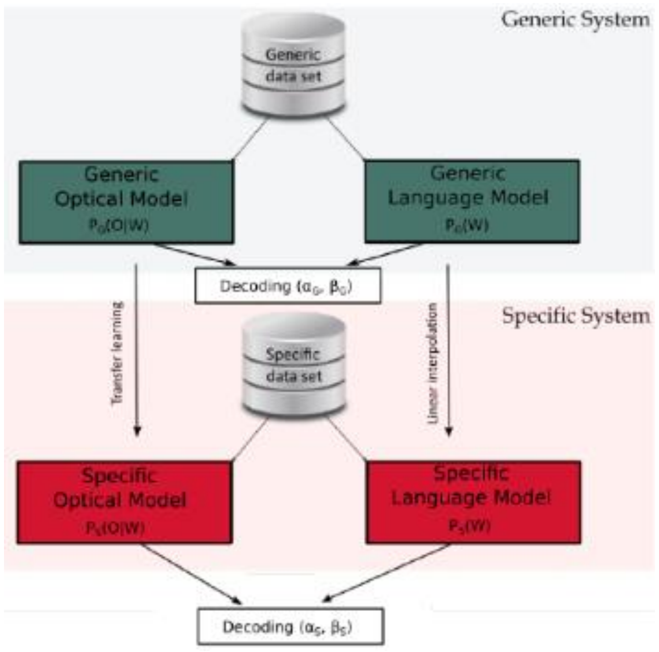


Figure 2.6.System architecture for writer adaptation

R. Achkar,et all[18] describes in a paper regarding doctors handwritten prescriptions. It basically describes how neural network or Convolutional Recurrent Neural Network (CRNN) technology is used specifically to detect medical prescriptions and translate the handwritten text into a digital one. The paper includes the methods used to write the python-based code, the training process, and the results obtained. The training is done using short texts, since prescriptions usually consist of 2 or 3 words as shown in figure 2.7.

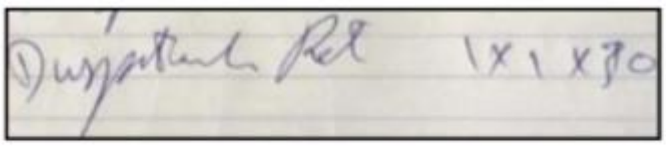

Duspatalin Ret 1x1 x30

Figure 2.7. Prescription Sample

This project is based on CRNN, which is inspired by VGG16 architecture that is used for image recognition noting that CNN differs from Recurrent Neural Networks (RNN) and Traditional Neural Networks. Traditional Neural Networks lack the use of memory, and use the result of previous training in order to predict the outcome, while CNN have the ability or benefit to develop an internal representation of a two-dimensional image. This allows the network to detect the position and scale of letters in the input data images. As for RNN, it works with sequence prediction problems, specifically the Long Short-Term Memory (LSTM), a type of RNN; it overcomes the problems of training a recurrent network that uses long-term data dependency problems. This network has the ability to remove or add information to the layer state by a decision made by a sigmoid layer, the output is to be based on said layer state. Finally, the output of the state is put through a tan function and multiplied by the output of the sigmoid gate

A hybrid network model such as CRNN is the perfect choice of combining the benefits of both RNN and CNN networks to be used to tackle the problem of extracting letters in handwritten medical prescriptions.

The program is not able to detect the start and end of the prescription, so the text is required to be manually inserted on a line-by-line basis, in other words, prescription by prescription. This will help the program to yield more accurate results with less computational time.

The training focuses on image processing by using a sliding window of ‘2’ width in order to check the written text by taking 2 or 3 characters at a time. In order to validate data, a minimum of 10,000 training samples need to be processed.

The input of the neural network is the image of different medical prescriptions, the desired output is specified in two ways. The first way is by creating a text with the output of every image. The second way is by creating a label for every character in the text. Figures 2.8.A,2.8. B, shows training error versus the validation error for three types of handwriting.

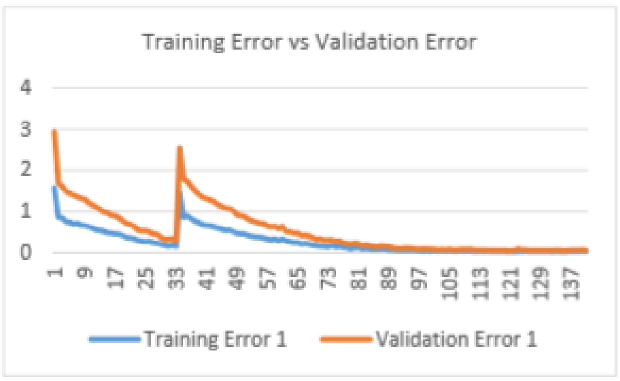


Figure 2.8.A Results of Normal Handwritten Text.

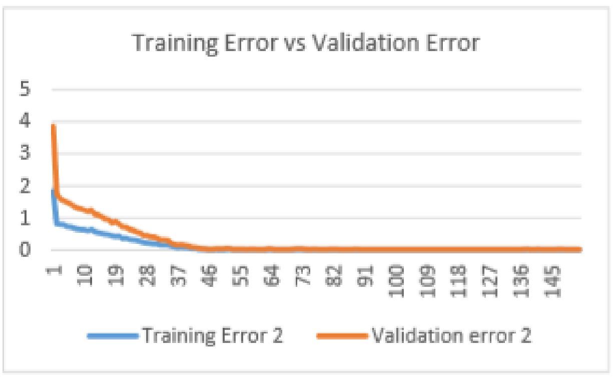


Figure 2.8.B Results of Cursive Handwritten

Figure 2.8. Results of different types of text

This combination of normal handwriting and that of doctors' prescriptions generated accurate results. This program achieved a score of 95% accuracy, which is really great, taking into account training time and the amount of data input.

Dr. E.Kamalanabhan , et al [19] provides research into a way to recognize the doctor's prescription using deep machine learning, by using the RNN neighborhood predicted character..

Regarding the data set, IAM dataset is used to sequence the handwriting, cropped images randomly sized 113x113 are used.

RNN is used in predicting the next character from images, for this project, the LSTM cell is used from RNN. For each time one word is processed by LSTM and the forthcoming letter in the name of drugs is found by calculating the probabilities of the possible values.

The used model was building a CNN in Keras with TensorFlow as backend,
A standard CNN is used with multiple convolutions and max pool layers, a few dense layers and a final output layer with SoftMax activation. Rectified Linear Unit (ReLU) activation was used between the convolution and dense layers. The size of the model needs to be proportional to the size of the data. Model summary is provided in Table 2.1.

Layer (type)	**Output Shape**	**Param #**
Zero_padding2d_2	(None,115, 115, 1)	0
Lamda_2(Lambda)	(None,56, 56, 1)	0
Conv1(Conv2d)	(None,28, 28, 32)	832
Activation_7 (Activation)		
Pool1(MaxPooling2d)	(None,14, 14, 32)	0
Conv2(Conv2D)	(None,14, 14, 64)	18496
Activation_8 (Activation)	(None,14, 14, 64)	0
Pool2(MaxPooling2d)	(None,7, 7, 64)	0
Conv3(Conv2D)	(None,7, 7, 128)	73856
Activation_9 (Activation)	(None,7, 7, 128)	0
Pool3(MaxPooling2d)	(None,3, 3, 128)	0

Flatten_2 (Flatten)	(None, 1152)	0
Dropout_4 (Dropout)	(None, 1152)	0
Dense1 (Dense)	(None, 512)	590336
Activation_10 (Activation)	(None, 512)	0
Dropout_5 (Dropout)	(None, 512)	0
Dense2 (Dense)	(None, 256)	131328
Activation_11 (Activation)	(None, 256)	0
Dropout_6 (Dropout)	(None, 50)	0
output (Dense)	(None, 256)	12850
Activation_12 (Activation)	(None, 50)	0
Total Params: 827 698 Trainable Params 827 698 Non-Trainable Params: 0		

Table2.1.: CNN model Summary

One more paper by **Y.Ou, S. et all[20]** ,provides a general overview of the Prescription Recognition System.it describes a system designed to help pharmacists identify medical information in the prescription. One of the main offered features in this system, recognize medicines and their usage, routes of administration, dosage, quantity and day. Moreover, the system makes patients have more time to ask questions and understand the treatment, and ensure patients' medication safety. As well, it provides a lot of features that were adopted in the main aim of our project.

The architect of the system is divided and executed in many stages. First stage, Prescription Image. The input prescription image will be obtained by scanner and sent to google text detection Application Program Interface (API). The results are in the format of strings and bounding boxes. After that, the Pre-Processing, where some processing is done with results obtained from the Prescription Image stage. This stage will prevent the difficulties of finding information in some separate word sequence by merging texts horizontally and combining text parts based on its relative position and font size. The National health insurance (NHI) code will be detected, where Each medicine has a different NHI code. The similarity between input NHI and other NHIs contained in the database will also help in detecting medicine names. After that the system will do some grouping, this grouping process will help find the rest of the information. After that, routes of administration and usage will be detected, where the route of administration is composed of English capitalization. For each string in the text group that consists of capital letters in English, where it changes any abbreviation found in the text, with the full medicine name. at least, the system will read other prescription information like dosage, quantity, and days and display it in special format.

The paper by Lisa Yan. **et al [21]** introduces a stateless approach to word recognition by using Faster R-CNN's region proposal network in conjunction with a heuristic inorder to do word labels from region proposals.The main model used in this project was Faster R-CNN model for object detection and classification.

To evaluate the word recognition system , a Faster R-CNN architecture based on VGG CNN M 1024 was trained on Pascal VOC 2007 for 70,000 iterations, then trained the model on the hybrid IAM and Chars74K training dataset for 40,000 iterations. Finally, the trained model was tested on the hybrid IAM and Chars74K test dataset. All methods were implemented with Caffe [31] and were trained on a NVIDIA GRID K520 GPU. Figure 2.9 shows the training loss on the word image dataset; the final network took approximately 12 hours to train.

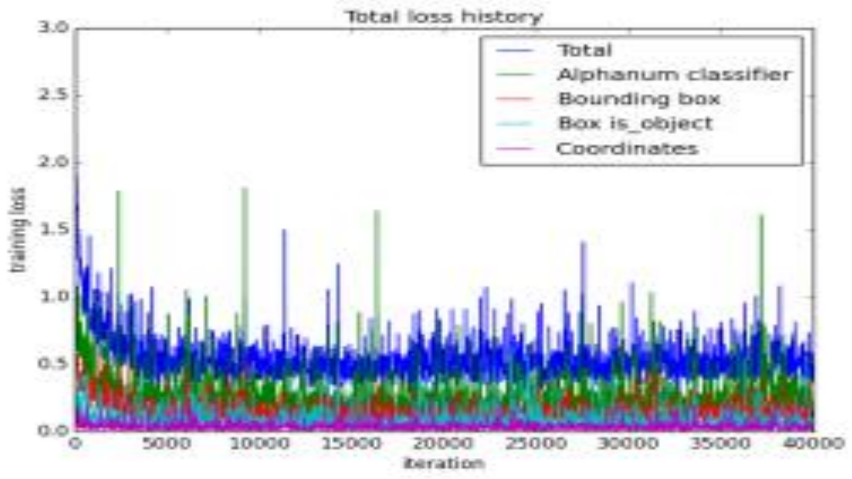


Figure 2.9. Loss function over 40k iterations

The uniqueness and importance of Scripty comes from the fact that we used a robust approach to recognize handwritten words and sentences by using Faster R-CNN which uses region proposal networks and CNNs to make the process of localization and recognition easier and more efficient. As we've seen from the related works above, the results are good and promising for word images that have characters with little overlap, while they don't give the desired results for cursive word images, where individual characters require surrounding context to be recognized. So, we used the Faster R-CNN algorithm which is based on object detection instead of doing the character segmentation in the traditional way even in over-lapped characters. Moreover, we used more than one dataset in training and testing of Scripty which is un-annotated and we did the annotation by ourselves.

Chapter 3: System implementation and design

3.1. Datasets

To build the model, two datasets will be used. IAM and EMNIST characters dataset. The IAM handwriting database contains the largest collection of English handwriting images which can be used to train and test handwritten text and to perform writer identification and validation experiments EMNIST characters database contains a huge number of handwritten characters, split into training, validating and testing.

The EMNIST character dataset is a part of hand printed document and character number recognition, it contains a huge amount of unannotated character images. Where each image has fixed height and width with 128 pixels. Our work includes labeling 26000 character images with a fixed bounding box using python, where the chars are equal in size in writing, so we might have some problems with chars that have the same shape in capital and small cases, since we have 52 classes, with nearly 16 characters that have same shape in capital case as the small case. Figure 3.1 below provides samples of EMNIST dataset.

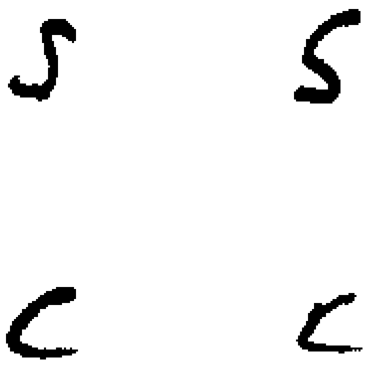


Figure 3.1. Samples from EMNIST dataset

The IAM database was first published at the International Conference on Document Analysis and Recognition (ICDAR) 1999. Using this database, a Hidden Markov Model (HMM) based recognition system for handwritten sentences was developed and published at the International Association for Pattern Recognition (ICPR) 2000.

The segmentation scheme used in the second version of the database is documented in and has been published in the International Commission for the Protection of the Rhine

(ICPR) 2002. We use the database extensively in our own research, see publications for further details.

The dataset items are scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels, dataset is annotated in PASCAL visual object classes (VOC) format, where all the annotation is written in xml file. The dataset annotated up to word level, a small part of the dataset have annotation to character level, which made us using the python to label images, we was able to label up to 7000 images in PASCAL VOC format, we used Labeling application which written in python programming language for this mission.. Figure 3.2 below provides samples of a complete form, a text line and some extracted words [22].

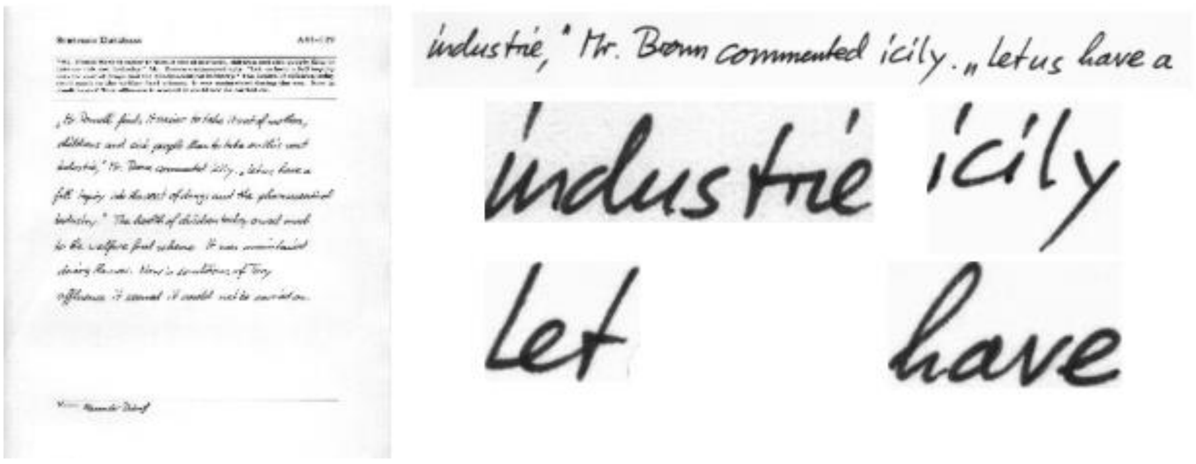

industrie, " Mr. Brown commented icily. " let us have a
industrie
icily
let
have

Figure 3.2. Samples from IAM dataset

In general, the occurrences of characters in English vary from character to other, so the IAM dataset will afford us unbalanced datasets since it contains only words. We had to mix the two datasets to achieve the class balancing, make the calculating of mAP valid and feed the deep learning model with a big number of data for each class. Table 3.1 shows the number of photos used from each dataset and number of characters from each dataset respectively.

Dataset	Words	Characters
IAM	7000	23 084
EMNIST	-	30 000

Table 3.1. Datasets words and characters statistics

Scripty aims to train over the characters inside the images annotated in Tensorflow record format, where each line inside the record contains the image name, bounding box coordinates, and the class of the object. The dataset is split with 70% for the training phase and 30% for the testing phase.

3.2. Deep Learning

Our handwriting recognition system will be based on object detection algorithms which is a deep learning technique. In our system, Faster RCNN to be specific is the chosen method of object detection algorithms.

3.2.1. Neural networks

In Neural networks a computer learns to perform a certain task by analyzing some training samples. The samples will be hand labeled to be used in the system. For instance, in our system, thousands of labeled handwritten prescription images will be fed to the neural network. Then this neural will be able to find visual patterns in those images that correlate to some certain labels.

To put it simply, we can model it as millions of processing nodes working interconnected. These nodes are organized in a layered matter. The data then will move forward from one layer to the other in what is called "feed forwarding". Which essentially means that a node may be connected to a lot of other nodes in the layer below and it will receive data from them. At the same time, this node will send the data after processing it to the nodes connected to it in the layer above.

To each of its incoming connections, a node will assign a number known as a "weight." When the network is active, the node receives a different data item — a different number — over each of its connections and multiplies it by the associated weight. It then adds the resulting products together, yielding a single number. If that number is below a threshold value, the node passes no data to the next layer. If the number exceeds the threshold value, the node "fires," which in today's neural nets generally means sending the number — the sum of the weighted inputs — along all its outgoing connections [23].

When a neural net is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the bottom layer — the input layer — and it passes through the succeeding layers, getting multiplied and added together in complex ways, until it finally arrives, radically transformed, at the output layer. During training, the weights and thresholds are continually adjusted until training data with the same labels consistently yield similar outputs [24].

3.2.2. CNN

CNNs are a primary class of deep neural networks which generalize multilayer perceptron's i.e.: feed-forward networks. CNNs are mainly based on convolution operations between data input which can be seen as a matrix, and a filter also can be seen as a matrix. The convolution operation can be represented as a replacement of the matrices product. The result can be viewed as a regularization or smoothing of the input data. Figure 3.3 shows the architecture of CNN.

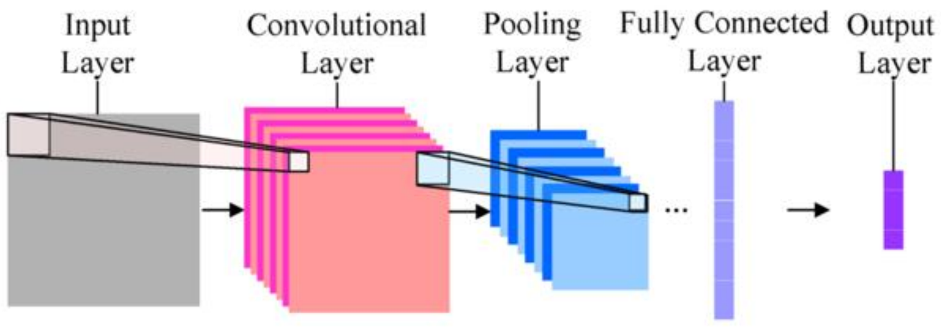


Figure 3.3. Architecture of CNN

The architecture is split into many stages, First one is the Input layer where the model acquiset the input image. The second stage is the convolutional layers, in CNN convolutional layers are used, which contain a set of learnable filters and these filters are adjusted to a region of the input image to scale down the image dimensions. Without any loss of main features of the image. Figure3.4 shows the work principle of the convolution layer [25].

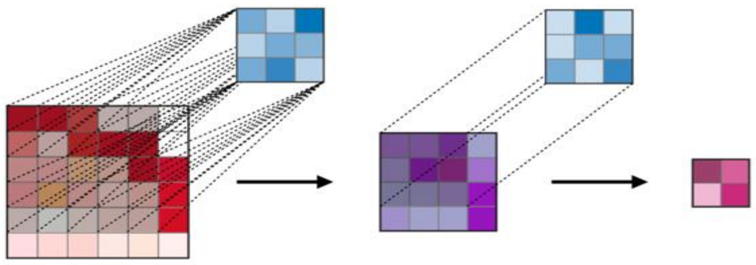


Figure 3.4. Convolutional layer in CNN

Third stage in CNN architecture is the pooling layer, which is used to reduce the number of dimensions (width, height), but maintains the most important information. A well-known technique used in the pooling layer is max pooling. Max pooling is a kind of non-linear sampling, it divides the input image into non-overlapping rectangles. Moreover, it accelerates computation and makes some of the detected features more powerful. Figure.3.5 shows the work principle of the pooling layer.

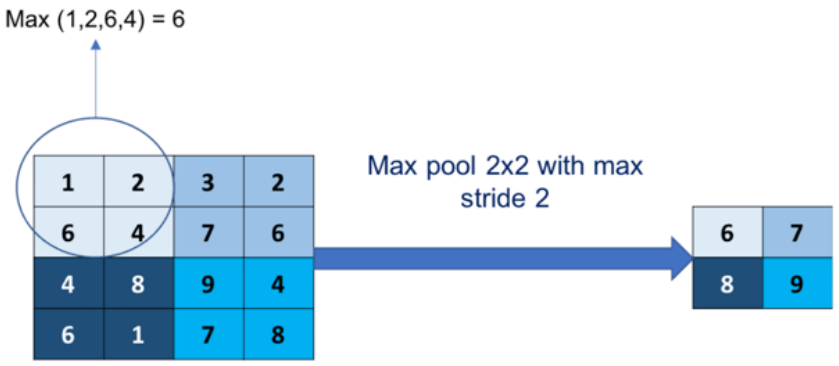


Figure 3.5. Pooling layer in CNN

Fully connected layer and dropout layer is the last stage of CNN, where neurons are connected to all activations in the past layer in order to generate class predictions. Dropout layers in CNN are used to prevent overfitting by increasing testing accuracy. We generally place the dropout layer with p = 0.5 since it is always better to only switch off the neurons to 50%. If we switched off more than 50% then there can be chances when the model leaning would be weak and the predictions will be bad. Figure.3.6 shows the work principle of the Fully-Connected layer [26].

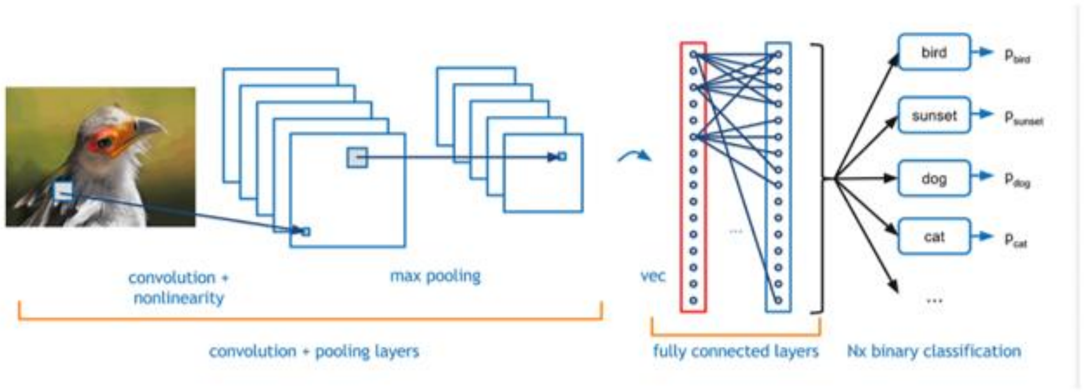


Figure 3.6. Fully connected layer in CNN

3.2.3. Object Detection Algorithms

Object detection algorithms are automated methods used for locating objects within an image. It's considered as an integral part of computer vision, and one of its hot topics these days.

Object detection algorithms are different from classification algorithms. The main aim of object detection algorithms is to draw a bounding box around all objects of interest within an image and locate them. Moreover, the number of occurrences of objects isn't fixed and this tells us that the output layer length in object detection algorithms isn't fixed, so we can’t build it using a standard convolutional network followed by a fully connected layer.

The major idea of object detection algorithms is to take the regions of interest (RoI) inside an image, and use the CNN to classify each object inside each region. But, the main problem faced by the object detection algorithms is the huge number of regions selected.

Some of the most well-known object detection techniques used with computer vision are the following; R-CNN, Fast RCNN, Faster RCNN. We will go into further details about the techniques we researched later on in this chapter.

A. R-CNN

RCNN is an object detection algorithm used to solve the huge regions selected in object detection algorithms, this approach uses 2k regions to classify, the selection of regions is based on a selective search, where the initial region of interests is generated, and then a greedy algorithm is used to combine the similar regions into larger ones. Figure 3.7 and Figure 3.8 both shows the main stages of R-CNN

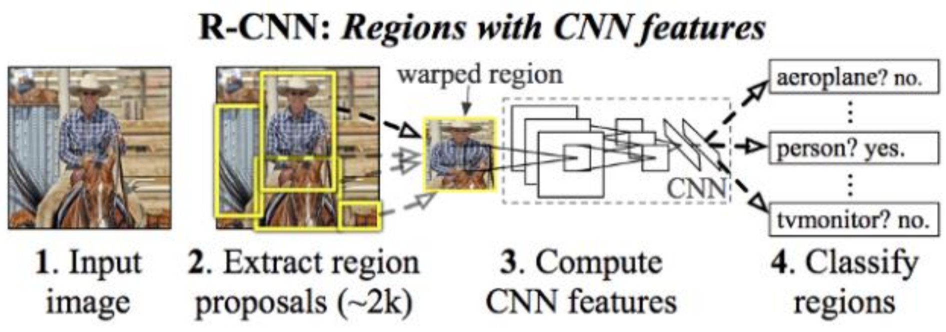


Figure 3.7. CNN stages

The two thousands regions are wrapped into one image, this image is fed to CNN which works as a feature extractor and produces $2^{\wedge 12}$ dimensional vectors as an output. The extracted features then fed a SVM algorithm to classify the objects inside each region. The algorithm produces four values with percentages for each region, which are the offsets to increase the precision of bounding boxes.

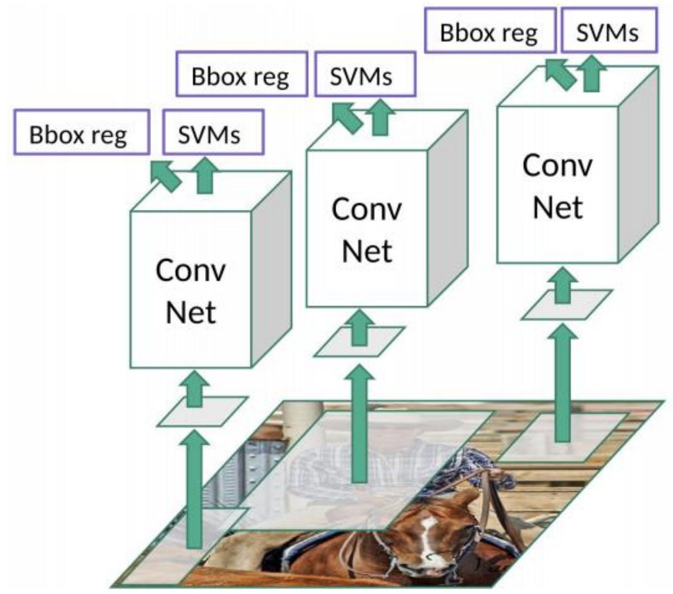


Figure 3.8. another view of R-CNN workflows

There are some problems with this approach, time to train data is too high, where the model needs to train 2000 regions per image, and also it takes 47 second for the testing side. In addition, the selective search method used for selecting the 2k regions is fixed, which means there's no learning happening at that stage. Finally, there is no guarantee for unselecting bad regions in the selective search method used [27].

B. Fast R-CNN

Fast RCNN is an improvement of the RCNN algorithm, which takes place to solve drawbacks of RCNN to build a faster object detection algorithm. The approach of Fast RCNN is similar to RCNN, but, instead of feeding two thousands regions for the CNN, we feed the input image and generate convolutional feature maps. Figure 3.9 shows the architecture of Fast R-CNN.

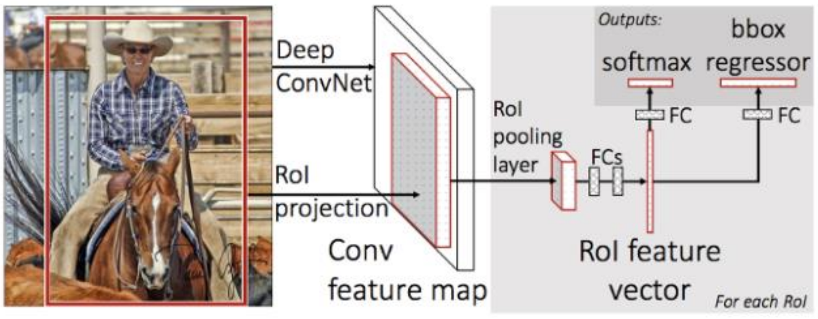


Figure 3.9. Fast R-CNN architecture

From the convolutional feature map, the region of proposals is identified and wrapped into one image. The image wrapped is fed to a RoI pooling layer which reshape the image into a fixed size divided into grids, and the max pooling is applied over each grid, and that produces the RoI feature vector, which we can feed to a fully connected layer that uses softmax function to predict the classes of the proposed regions and a fully connected layer that predict the offsets of bounding boxes relative to the original RoI for each class.

Using this approach, RoI is selected based on the convolutional feature map extracted from the CNN, so you don't have to train 2000 regions per image in Scripty, which makes this algorithm much faster than RCNN, where it takes 8.75 seconds to train each image and 2.3 seconds for the testing.

Fast RCNN model is optimized for a loss combining classification and localization, where the loss function is presented in Equation 3.1.

$$1[u >= 1] = \begin{cases} 1 & \text{if } u \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \mathcal{L}(p, u, t^u, v) &= \mathcal{L}_{\text{cls}}(p, u) + 1[u \geq 1]\mathcal{L}_{\text{box}}(t^u, v) \\ \mathcal{L}_{\text{cls}}(p, u) &= -\log p_u \\ \mathcal{L}_{\text{box}}(t^u, v) &= \sum_{i \in \{x,y,w,h\}} L_1^{\text{smooth}}(t_i^u - v_i) \end{aligned}$$

Equation Number 3.1

Where the total loss is the summation of classification cost and bounding boxes prediction costs, where u represent the true class label, the value 0 represents the background which we neglect inside our classification. p is the discrete probability distribution per RoI over classes, v is the true bounding box and t^u is the predicted bounding box correction. The bounding box loss L_{box} uses a robust loss function to measure the difference between the true and predicted bounding box values, where the smooth L1 loss is claimed to be less sensitive to outliers [28].

C. RPN

In object detection using R-CNN methods, Region Proposal Networks abbreviated as RPN is basically the backbone of the new Faster RCNN algorithm and have proven to be very efficient till now.

To put it simply, Regions where the objects lie in an image are called “proposals”. To generate those proposals, a small network is slid over a convolutional feature map that is the output by the last convolutional layer. Figure 3.10 shows the workflow of RPN.

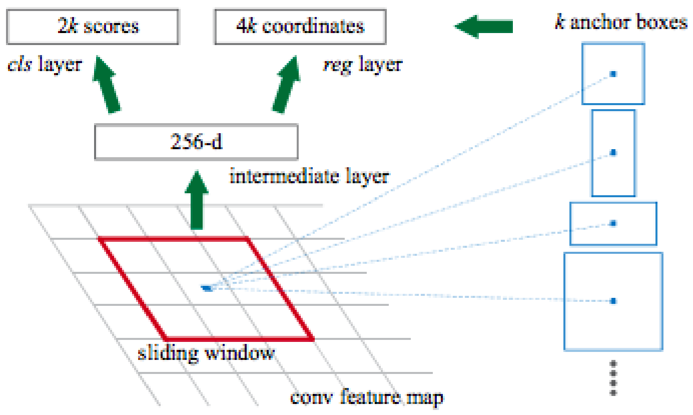


Figure 3.10. RPN workflow

The concept of an anchor has been introduced with RPN, an Anchor is the central point of the sliding window. In RPN there is a classifier and a regressor. The classifier determines the probability of a proposal having the target object. While regression regresses the coordinates of the proposals.

For any image, scale and aspect-ratio (aspect ratio = width of image/height of image), are two important parameters. In the paper introducing RPN, The developers chose 3 scale and 3 aspect-ratio. So, a total of 9 proposals are possible for each pixel, this is how the value of k is decided, K=9 for this case, k being the number of anchors.

For the whole image, number of anchors is W*H*K. This algorithm is robust against translations, therefore one of the key properties of this algorithm is translational invariant. And how it works exactly is that there anchors are assigned labels based on two things; the anchors with highest Intersection-over-union (IoU) overlap with a ground truth box. And the anchors with IoU Overlap higher than 0.7.

Ultimately, RPN is an algorithm that needs to be trained. So there is a loss in regression and classification parts. Equation 3.2 shows the loss function of regression and classification parts.

$$L(\{p_i\},\{t_i\}) = (1/N_{cls}) \times \Sigma L_{cls}(p_i,p_i^*) + (\lambda/N_{reg}) \times \Sigma p_i^* L_{reg}(t_i,t_i^*)$$

$$L_{reg}(t_i,t_i^*) = R(t_i - t_i^*)$$

Equations number 3.2

Where i represents the index of anchor, p is the probability of being an object or not, is a vector of four parameterized coordinates of predicted bounding box, the star represents the ground truth box, which is the real bounding box around the object, and N_{cls} and N_{reg} represents the normalization. The constant λ has a default value equal to 10, so that the classifier and regressor are scaled to the same level.

We must mention that L in the classifier represents the log loss function. Moreover, p* with regression term in the loss function ensures that if and only if the object is identified, then only regression will count, otherwise p* will be zero, so the regression term will become zero in the loss function [29].

D. Faster R-CNN

The Faster RCNN model for object detection and classification is the major model used in this project. It is one of the most popular object detection architectures which uses CNNs.

Faster RCNN as Figure 3.13 shows can be simply considered as the combination of RPNs and Fast R-CNN, it has integrated feature extraction, proposal extraction, bounding box regression, classification into a network, achieving end-to-end training in a great meaning.

The way of handwriting text recognition mainly includes two parts: First, character segmentation. Second, character recognition. The accuracy rates of recognition based on traditional OCR is relatively poor to these two parts for complex conditions, so it is very desirable to solve this problem based on Faster RCNN. Figure. 3.11 shows the main architecture of Faster RCNN.

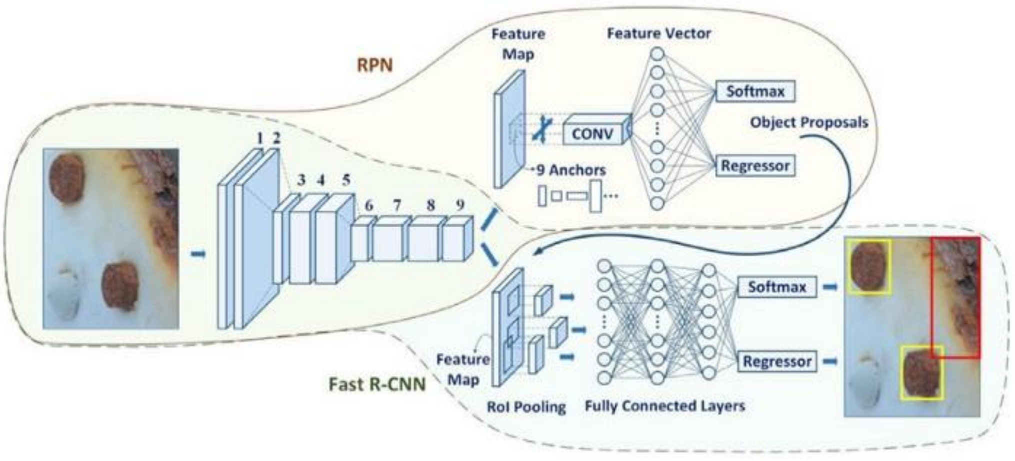


Figure 3.11. Faster R-CNN architecture

From figure 3.11, the architecture of Faster RCNN is composed from three parts as shown. The first part is the convolutional layers. In these layers filters are trained to extract the proper features of the image, for example if we are going to train those filters to extract the proper features for a human face, then those filters are going to learn through training shapes and colors that only exist in the human face. In Scripty, we are going to train the model to learn through character, so we focused on shapes and edge detection since these are the most two factor matters in character recognition, and for this purpose we used the VGG neural networks.

The second part is the RPN, which is a small neural network sliding on the last feature map of the convolution layers and predicting if there is an object or not and it also predicts the bounding box of those objects. Figure 3.13 in the RPN section shows the workflow of RPN used in Faster-RCNN.

The RPN in Faster R-CNN takes the output from the last convolutional layer and slides a n × n window across to make m sliding windows. After that, k anchors are assigned per sliding window, where each anchor determines a different scale and aspect ratio for region proposals which are centered at that sliding window. In our Faster R-CNN architecture, n = 3 and k = 9. Next, the sliding window is passed through a layer to generate a lower resolution middle layer, which finally outputs the region proposal illustrated by two layers, classification (cls) and regression (reg). The cls layer produces a score of whether or not an object exists within each of the k anchors, while the reg produces the bounding box for the region proposal, defined by 4 coordinates xmin, ymin, xmax and ymax respectively.

The third part is Fast R-CNN or the classification and bounding box prediction stage. In this stage, we pass the mk region proposal outputs from the RPN into the Fast R-CNN network to generate mk softmax classifications in every image. The RPN output is first passed through a RoI pooling layer that scales each region proposal into equally sized inputs to feed into the fully connected layers for classification. We use the near joint training procedure to train RPN and Fast R-CNN at the same time.

The RPN and Fast R-CNN share convolutional features, whose backward passes are calculated as the combination of the RPN and Fast R-CNN loss. For each forward pass, the region proposals are treated as fixed and pre-computed. The RoI pooling layer, whose input is both the convolutional feature and the bounding box proposals, thus avoids computing the nontrivial derivative with regards to the bounding box proposals.

In general, at first we scale the input image, and enter it into the CNN (convolution layers and pooling layers) to extract the feature map. Secondly, the feature map is fed into the RPNs to generate a set of possible bounding boxes for the target. Finally, the original feature map and all possible candidate boxes of RPNs output are fed into the RoI pooling layer for extraction and collection of proposal, then calculation of the feature maps are fed into the full connection layer, finishing output object classification and coordinate regression. Figure 3.12 shows the complete flow of the Faster R-CNN model [30].

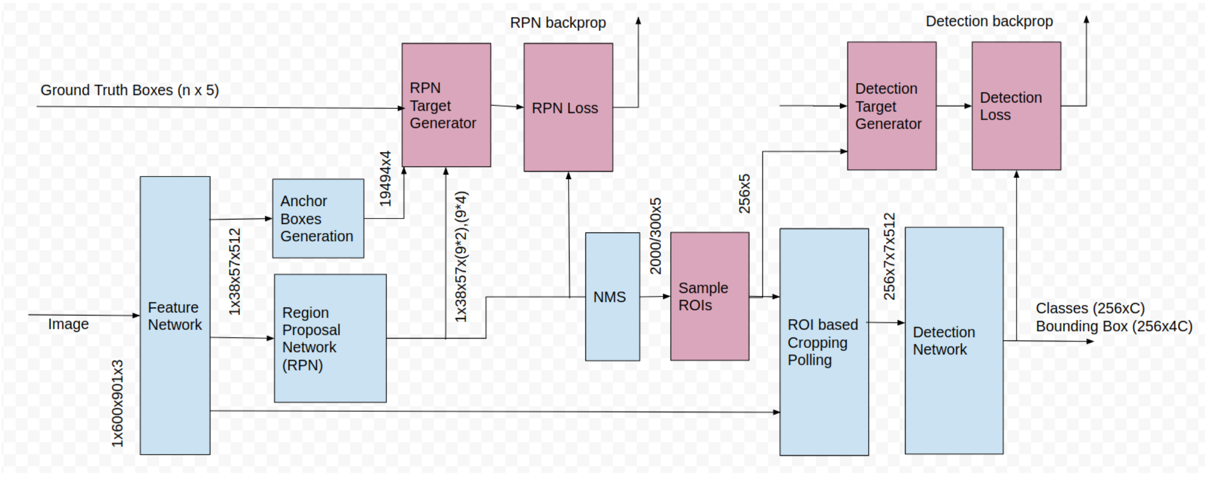


Figure 3.12.Faster R-CNN model flow

Most of the complexities of Faster RCNN lies on the training of RPN and Detection network. The training of RPN can be summarized as follows: Firstly, a mechanism called anchor boxes generates a number of bounding boxes. This mechanism considers every pixel of the featured image as an anchor. Each anchor correlates to a larger set of squares of pixels in the original image, for better results in feature extraction some reshaping is usually done on the original image. As can be noticed in Figure.3.13.a, anchors are located uniformly across both dimensions of the reshaped image. To generate anchor boxes it's required to take the shape of the tensor as input from the feature generation layer, these anchors generated will have different shapes and sizes of rectangular boxes generated at the center of each anchor. Commonly 9 boxes are generated per anchor (3 sizes x 3 shapes) as shown in Fig 3.13.b hence, there are tens of thousands of anchor boxes per image.

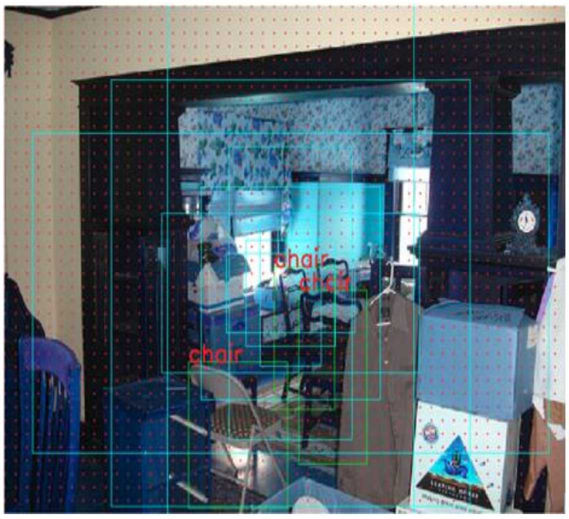


Figure 3.13..A Anchor generation Example

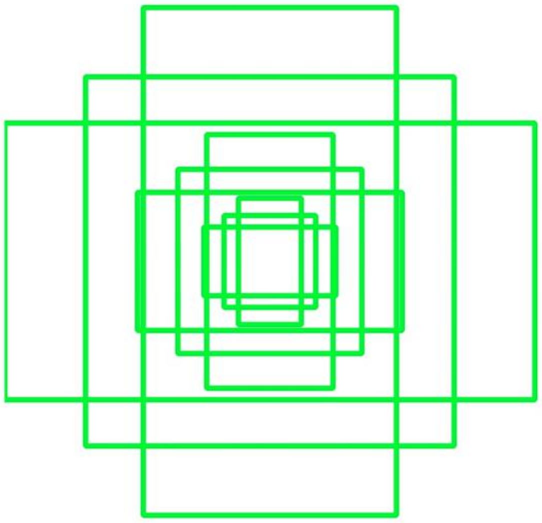


Figure 3.13.B Anchor boxes of 3 different shapes

Figure 3.13. Anchors generated examples

An operation called Non-Maximum Suppression (NMS) is used as the first step of reduction. NMS discards boxes that overlap with other boxes that have higher scores, where scores are still unnormalized probabilities before softmax is applied to normalize them. About 2000 boxes are extracted through the training phase where the number is about 300 in the testing phase.

During the testing phase, those boxes with their scores go to the detection network. In the training phase, the 2000 boxes are reduced through sampling to about 256 before entering the detection network.

In order to generate labels for RPN classification, i.e.: foreground, background, and ignore, IoU of all the bounding boxes against all the ground truth boxes are taken. Then, IoUs are used to label the 256 RoIs as foreground, background and ignored. These labels are then used to calculate the cross-entropy loss, after first removing the ignored class boxes.

Along with classification, the RPN also attempts to narrow the center and the size of the anchor boxes around the target, which is called the bounding box regression. For achieving this, targets need to be generated, and losses need to be calculated for back propagation. The distance vector from the center of the ground truth box to the anchor box is taken and normalized to the size of the anchor box, which is the target delta vector for the center. The size target is the logarithm of the ratio of size of each dimension of the ground truth over the anchor box. The loss is calculated by using an expression called Smooth L1 Loss .Figure 3.14 shows a plot of smooth L1 loss vs the norm(d).

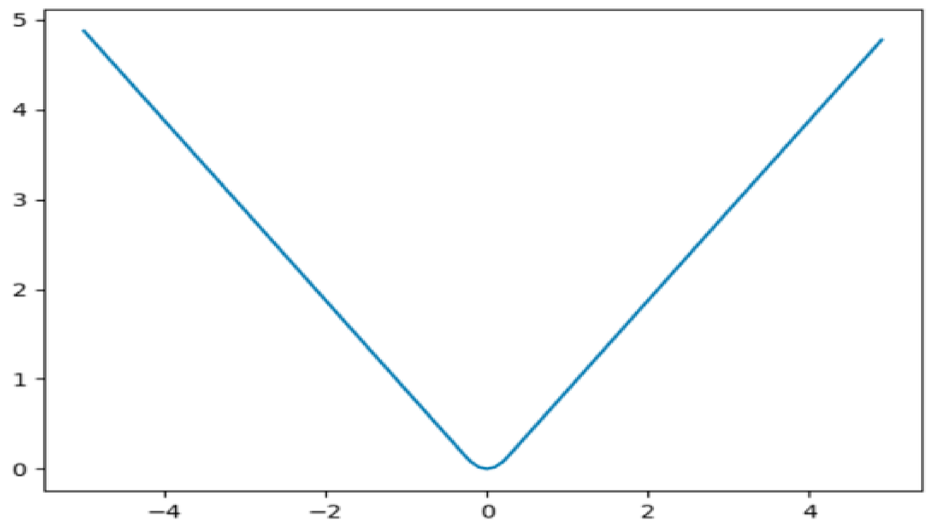


Figure 3.14. RPN loss function

From the figure 28, it's clear that the regular L1 loss (the norm or absolute value) is not differentiable at 0. So, smooth L1 Loss conquers this by using L2 loss near 0. The range of L2 loss is tuned by a parameter called sigma. Mathematically the formula looks like the following pseudo-code shown in figure 3.15.

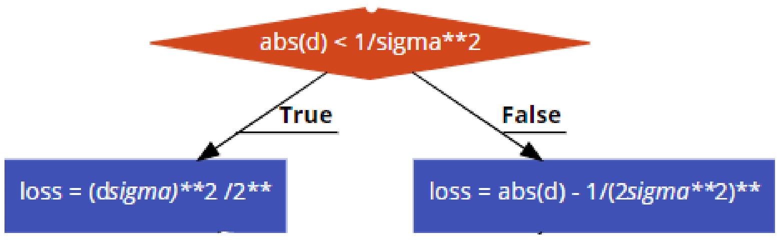


Figure 3.15. pseudo code for loss function in RPN

The losses are back propagated as the usual way to train RPN. The RPN can be trained alone, or jointly with the detection network.

The next stage of training is the detection network training. Detection Network can be considered as the top layers of the classification network which is used for features generation. So the starting weights can be pre-loaded from that network before training. Training the Detection Network is very similar to the training of RPN. Firstly, IoUs of all the 2000 bounding boxes against all the ground truth boxes are taken. So RoIs generated by the NMS following RPN against each ground truth bounding box is calculated. After that, the RoIs are labeled as foreground or background based on the corresponding threshold values. Then a fixed number i.e: 256 RoIs are selected from the foreground and background ones. If there are not enough foreground and/or background RoIs to fill the fixed number, then some RoIs are duplicated randomly.

The features are cropped and scaled to 14x14 but it’s max-pooled to 7x7 at the end before entering the Detection Network, according to the size of the RoIs, for this, RoI width and heights are scaled to the feature size.

In order to generate labels for Detection Network classification, IoUs of all the RoIs and the ground truth boxes are calculated. Based on IoU thresholds e.g. foreground above 0.5, and background between 0.5 and 0.1, labels are generated for a subgroup of RoIs. The difference between RPN and here is that here there are more classes. Classes are encoded in sparse form, instead of one-hot encoding.

Following a similar approach to the RPN target generation, bounding box targets are also generated. But these targets are in the compact form as mentioned previously, hence are expanded to the one-hot encoding for calculation of loss.

The loss calculation is also similar to that of the RPN network. For classification sparse cross-entropy is used and for bounding boxes, Smooth L1 Loss is used. The only difference with RPN loss is that there are more classes (20 including background) to consider instead of just 2 (foreground and background) [31].

Figure 3.16 shows the main differences in architecture between the main object detection algorithms which we explained before.

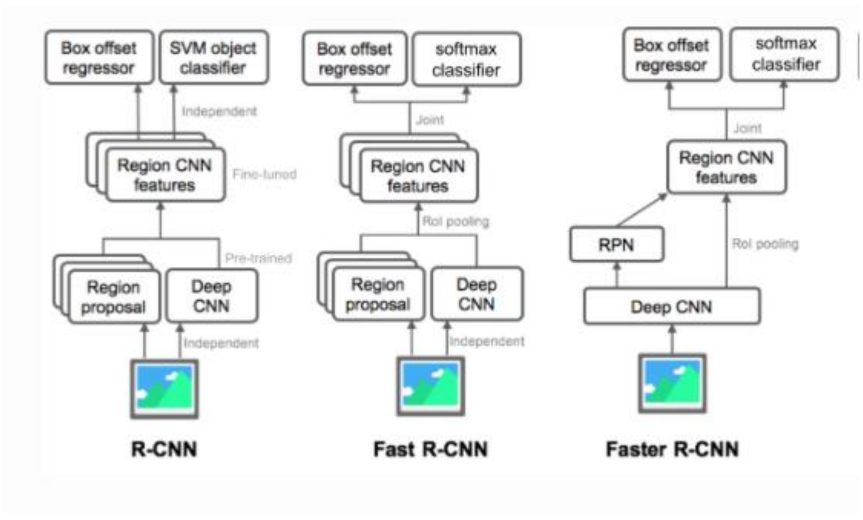


Figure 3.16. object detection algorithms architecture compariso

3.3. Methodology

For the mission of complex handwriting text recognition in this project, there are four main steps that will be detailed, Figure 3.17. Shows the flow diagram we used in building Scripty.

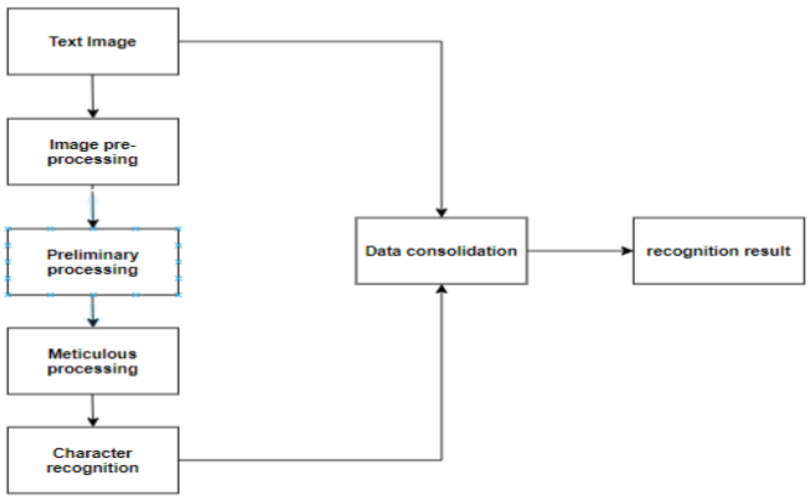


Figure 3.17. Flow chart of Scripty

1) Image pre-processing: This step includes data labelling, noise removing, binarization, edge detection, data augmentation so the dataset would be ready for the segmentation stage. We separated the dataset into different folders containing words, characters and merged between words and characters to make the training stage easier and more efficient.

2) Preliminary processing: This step includes obtaining the image of each word as the output to the next network, at the same time the locative information of the words in the text can be obtained using character segmentation of the words depending on faster R-CNN.

3) Meticulous processing: This step includes features extractions of the images depending on the VGG16 network and the proposals of the RPN, this works by obtaining the images from the input and then passed to VGG to implement the first feature extraction step, also, another feature extraction process will done over the proposals generated by the RPN and the output will passed to the classification part.

4) Character recognition: This step includes classification and recognition of each letter in the input image based on CNN as discussed in the Fast RCNN section. The output from this step is the recognition results as easy and usable data and saving the data to the next step.

5) Data consolidation: This step aims to achieve data consolidation depending on the spatial information of the word in the text, finishing visualization of results data in the text image through covering the original text area.

Each step mentioned above will be discussed in details in the next chapter.

Chapter 4: Performance Evaluation and results.

4.1 Simulation Tools

4.1.1 Deep learning platform

After searching and studying the requirements of Scripty and time provided for the completion of this project, we decided to choose Google Colab platform where we developed Scripty using python programming language.

Google Colab is a free online cloud based Jupiter notebook environment that allows us to train our machine learning and deep learning models on CPUs, GPUs and TPUs. We choose Google Colab to be the chosen framework for a few reasons. First, Google Colab offers pre-installed libraries, such as Pandas, Numpy, Matplot in addition to offering many pre-installed machine learning libraries such as Keras, TensorFlow and PyTorch, in which we actually needed and used in Scripty. In addition, having a development environment where everything can be saved on the cloud and can be accessed from any device at any given time, it helps massively in the development of a machine learning model like ours. Another very important feature is its collaboration feature where everyone in the team is able to collaborate on the coding part and see instant changes made while working together on this project. Besides, you can also share your completed work with other developers.

The most important reason may be the fact that Google Colab offers free GPU and TPU use, which you can run for 12 consecutive hours. The GPU and TPU acceleration make a huge difference even for some small projects. considering that training Scripty may need huge memory and speed, having free GPUs to use on a browser based app , gives the capabilities to train for long hours without the stress of buying a GPU and worry about our cooling fan work, for Scripty we used the GPU option to train and test our model. Which can get up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB Ram.

4.1.2 Programming Language

The python version used for this model is python 3, as it was the version that accommodates Scripty’s code needs and works well for the libraries we used.

4.1.3 Dataset

In addition to Google Colab, we used another google product which was Google Drive that offers free cloud space to save our dataset in order to be used later for training and testing on the model. It was between the whole team.

4.1.4 Framework

As mentioned above, many pre-installed machine learning libraries and packages were used in our deep learning model to simplify the overall programming of Scripty, we are going to introduce the most important ones here which are: TensorFlow and Keras libraries, MatPlotLib, Pandas and OpenCV.

TensorFlow is an end-to-end open source platform for machine learning. It's a comprehensive and flexible ecosystem of tools, libraries and other resources to provide a high level APIs workflow.

Keras, on the other hand, is a high-level neural networks library that is running on the top of TensorFlow, CNTK, and Theano. Using Keras in deep learning allows for easy and fast prototyping as well as running seamlessly on CPU and GPU. This framework is written in Python code which is easy to debug and allows ease for extensibility.

In addition to Keras and TensorFlow, we used MatPlotLib; it is a comprehensive library for creating static, animated, and interactive visualizations in Python. We needed this library to plot training and testing results. We also used Pandas; it is used for Data Analysis, providing tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc. Pandas also allow the handling of missing data. It was mostly used for dealing with CSV files in Scripty for records, also for handling the annotation files when training and testing. It's also worth noting that Pandas library is built on top of Numpy, meaning Pandas needs Numpy to operate.

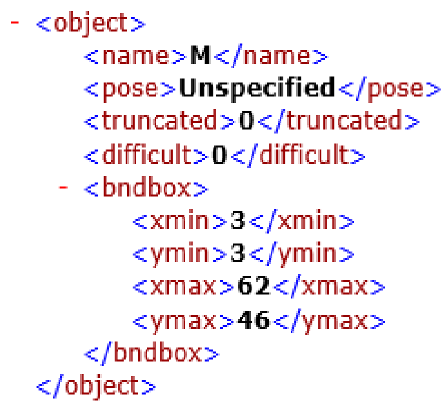
OpenCV is another important framework that is almost used in every deep learning model focused on recognition of images, objects and video processing applications. OpenCV is an open-source computer vision and machine learning software library developed by Intel. It provides a common infrastructure for applications related to computer vision and its associated fields and it is used to speed up the use of real-time machine recognition of objects. It has been a very important factor in Scripty since it helped with all image processing operations done on this model such as image filtering, geometric image transformations, color space conversion, histograms, augmentation, etc.

To understand OpenCV better we have to introduce Computer Vision first; Computer vision is a field of computer science that aims to achieve the human-like vision in detecting , identifying and processing different objects in images and videos , in order to enable computers to reach the level of complexity. Without Computer vision and OpenCV a lot of applications and models as Scripty here wouldn't be available.

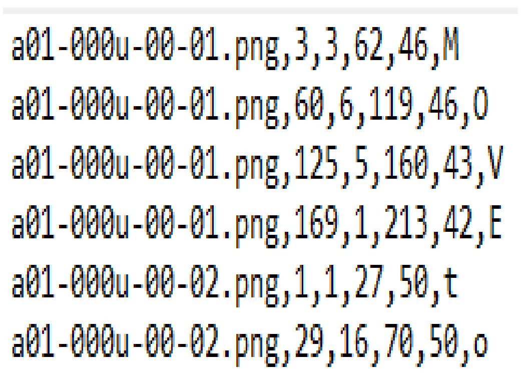
4.2 Faster RCNN

4.2.1 Data Preparation

The first stage of Scripty workflow is data preparation. Since our datasets are not annotated, and the IAM dataset contains more than one character in the image, we used LabelImage scripts which is written in python programming language to label characters in about 7000 IAM images in Pascal VOC format, which is written in xml files, then we used the python again to extract the annotations to tensorflow records (image, x1, y1, h, w, class) which is written in text files. Since the EMNIST dataset contains a single object with fixed size in each image, we used python to label characters directly without using LabelImage. Figure 4.1 shows the format of annotations file generated.



```
- <object>
    <name>M</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
  - <bndbox>
      <xmin>3</xmin>
      <ymin>3</ymin>
      <xmax>62</xmax>
      <ymax>46</ymax>
    </bndbox>
  </object>
```



```
a01-000u-00-01.png,3,3,62,46,M
a01-000u-00-01.png,60,6,119,46,O
a01-000u-00-01.png,125,5,160,43,V
a01-000u-00-01.png,169,1,213,42,E
a01-000u-00-02.png,1,1,27,50,t
a01-000u-00-02.png,29,16,70,50,o
```

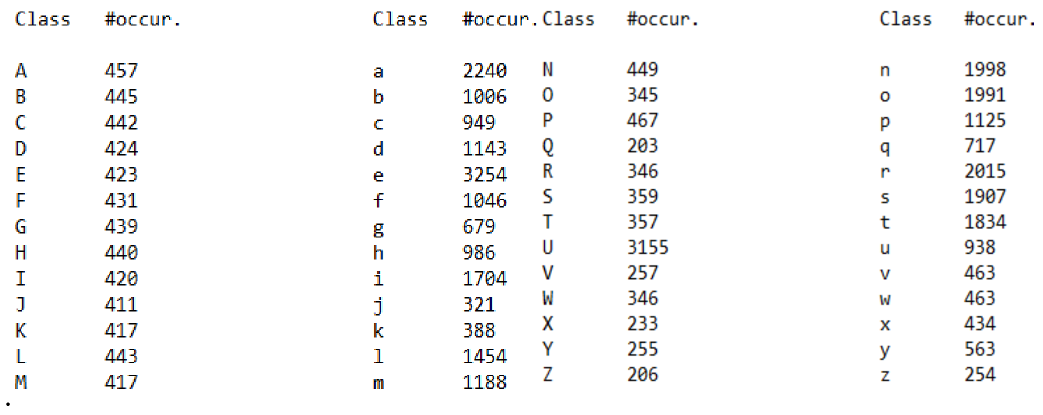
Figure 4.1..A Pascal VOC format *Figure 4.1.B tensorflow records*

Figure 4.1. generated annotations files format

We applied some image processing phases over our data. In order to prepare our data for VGG feature extraction, we applied some resizing over the EMNIST dataset to get bigger images from the dataset to fit into the VGG.

During the training process, each bounding box will augmented based on its occurrences number in the annotation file in order to significantly increase the diversity of data, this process will augment the low occurrences characters in our dataset dynamically using OpenCv library in python.

Table 4.1 shows the character occurrences before starting the training and augmenting bounding boxes.



Class	#occur.	Class	#occur.	Class	#occur.	Class	#occur.
A	457	a	2240	N	449	n	1998
B	445	b	1006	O	345	o	1991
C	442	c	949	P	467	p	1125
D	424	d	1143	Q	203	q	717
E	423	e	3254	R	346	r	2015
F	431	f	1046	S	359	s	1907
G	439	g	679	T	357	t	1834
H	440	h	986	U	3155	u	938
I	420	i	1704	V	257	v	463
J	411	j	321	W	346	w	463
K	417	k	388	X	233	x	434
L	443	l	1454	Y	255	y	563
M	417	m	1188	Z	206	z	254

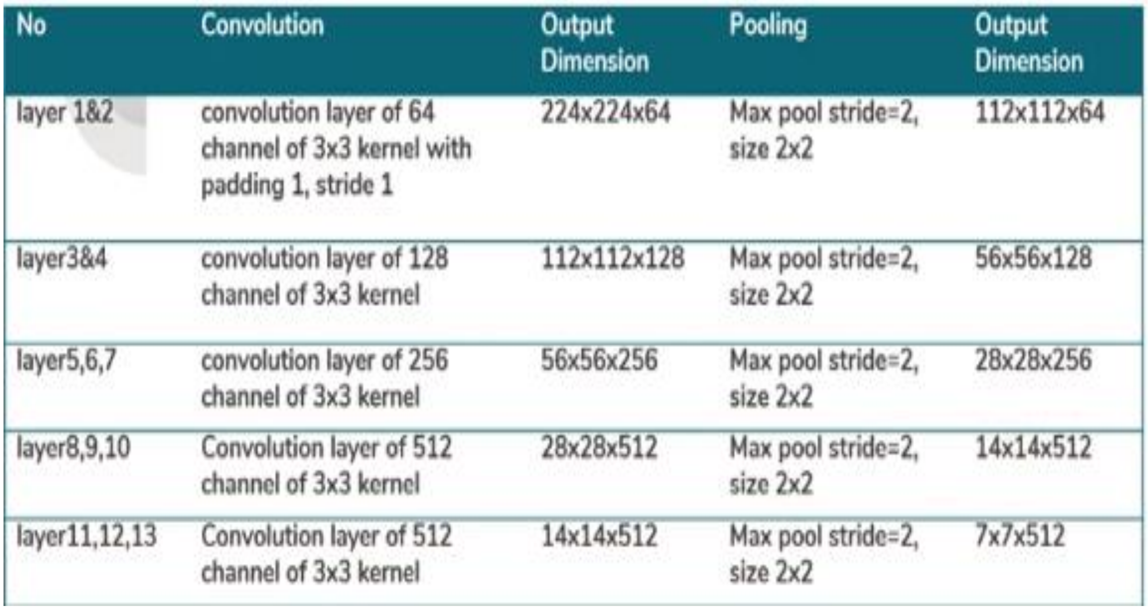
Table 4.1. Classes Statistics

We used Google Colab to train Scripty, and due to limited disconnectivity, where it allows train up to 12 hours, and the runtime disconnects after 90 minutes when Colab isn't open, we used checkpoints to keep saving our model after train each epoch in a file with Hierarchical data format (HDF5), so we can avoid restarting the training if any runtime interruptions happens while training the model.

4.2.2 Features Extraction using VGG

VGG-16 is a pre-trained CNN that was used as the base network of Scripty. It's since it is considered as the one of the excellent vision model architecture till date in extracting and classification of features related to shapes.

In general, some English letters have very similar shapes so we need a strong network in this field to give the most acceptable results. Moreover, the most popular thing about VGG-16 is that instead of having a large number of hyper-parameters they have focused on having convolution layers of 3x3 filter with a stride 1 and always used the same padding and max pool layer of 2x2 filter of stride 2. At the end it has 2 fully connected and two dropout layers followed by a soft max for output. The 16 in VGG16 refers to it as 16 layers that have weights. This network is a large network and it has about 138 million parameters. Table. 4.3 summarizes each layer of VGG16 used in Scripty and its main functionality.



No	Convolution	Output Dimension	Pooling	Output Dimension
layer 1&2	convolution layer of 64 channel of 3x3 kernel with padding 1, stride 1	224x224x64	Max pool stride=2, size 2x2	112x112x64
layer3&4	convolution layer of 128 channel of 3x3 kernel	112x112x128	Max pool stride=2, size 2x2	56x56x128
layer5,6,7	convolution layer of 256 channel of 3x3 kernel	56x56x256	Max pool stride=2, size 2x2	28x28x256
layer8,9,10	Convolution layer of 512 channel of 3x3 kernel	28x28x512	Max pool stride=2, size 2x2	14x14x512
layer11,12,13	Convolution layer of 512 channel of 3x3 kernel	14x14x512	Max pool stride=2, size 2x2	7x7x512

Table 4.2:VGG-16 layers and its main functionalities

As can be noticed from table 4.3, the input image was resized to 224x224x64, in order to be used in the VGG-16. The first and the second layers are convolutional layers that have max pool stride=2, so the output dimension will be 112x112x64. The same goes with the layers from 3-13, which its max pool stride=2 so the output dimensions are divided by 2 in each time. Finally, the output dimension from the last layer is 7x7x512 which will be used in the next stages.

4.2.3 Generating Anchors in RPN layer

RPN is applied on the feature maps generated from the VGG16. This returns the object proposals along with their objectness score. The RPN in Scripty is a simple network with 3 convolutional layers. There is one common layer which feeds into two layers — one for classification and the other for bounding box regression.

To create this RPN layer we firstly pass the feature maps from the base layer to a convolutional layer with kernel size 3x3 and 512 channels, keeping the size of the feature map the same, then we proceed to pass the first step to a convolutional layer with kernel size 1x1 to replace the fully connected layer. Then we create the classification layer and the regression layer for our RPN layer.

For the classification layer, 9 anchors give 0 or 1 output because it uses sigmoid activation, where the values 0 or 1 indicate if the anchor contains an object or not. Whereas 36 anchors channel for computing the regression bounding boxes with linear activation. We also defined lambda_rpn_regr = 1.0 and lambda_rpn_class = 1.0, as they will be used in loss formulas later on.

In order to create those three convolutional layers, we use (conv2D) function from Keras, this function is able to create a 2D convolution layer which creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. When creating this 2D convolution layer it is important to pass the activations argument, in order to specify the activation layer used. In our case, we used ‘relu’ for our RPN layer. And we used “sigmoid” for our classification layer, and ‘linear’ activation for our regression layer.

To explain the activations used Relu and Sigmoid, Relu activation applies the rectified linear unit activation function. With default values, this returns the standard ReLU activation: max(x, 0), the element-wise maximum of 0 and the input tensor. While Sigmoid activation applies Sigmoid activation function. Sigmoid is equivalent to a two-element softmax, where the second element is assumed to be zero. The sigmoid function always returns a value between 0 and 1. In order to calculate the RPN loss in classification, we used the categorical_crossentropy.

Linear regression is used here to predict the bounding boxes for the identified objects because what it generally does is generate tighter bounding boxes for each identified object in the image. All these offsets and scores are combined with the anchors to generate the proposals which is the input for the RoI pooling layers. Figure 4.2 shows examples of the proposals with highest probability around each object inside some training images.

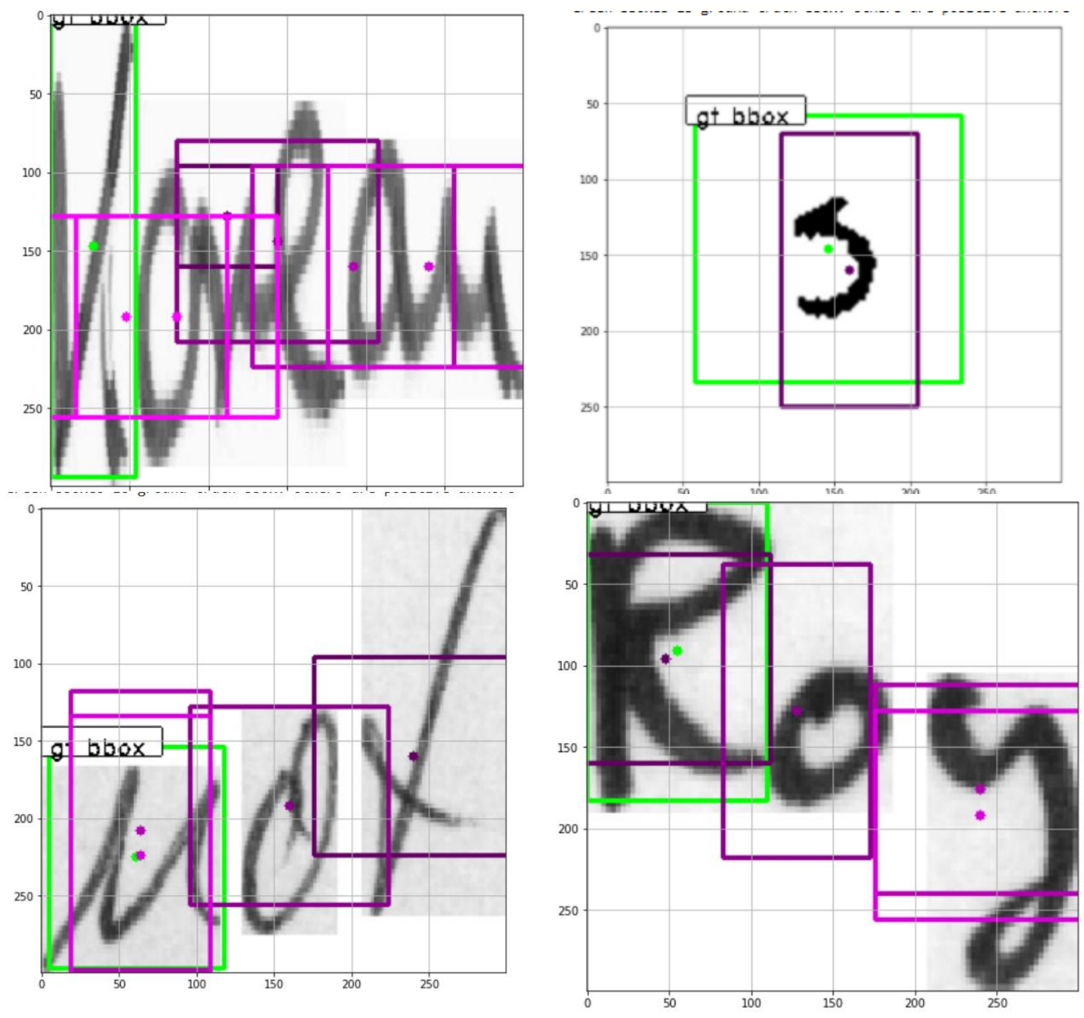


Figure 4.2. examples of highest probability proposals around characters

4.2.4 Calculating IoU and NMS layer

This layer of Scripty will take the proposals from the RPN layer as inputs. At this layer we'll filter out the unwanted generated boxes using NMS technique. First, the IoU is calculated between all boxes, where it represents the similarity between boxes, where each two boxes with similarity near to 1 will be merged into one box.

NMS in Scripty is designed as follows, first, it sorts all the bounding boxes based on their scores, where it selects the highest confidence box, which is considered as part of the output, and IoU is calculated based on the selected box. We mainly used the Numpy libraries APIs in this section since it is completely based on mathematical calculation on union between boxes.

This stage output is the following, all boxes drawn outside the feature map will be ignored and dropped. Also, the boxes that have IoU over the overlapping threshold will be ignored. In Scripty, the overlapping threshold plays an important role, since the cursive handwriting is strongly overlapped. After many experiments over Scripty, we achieved the best case at overlapping threshold=0.7.

4.2.5 Extracting Features based on RPN proposals.

At the RoI Pooling layer, Scripty will take the proposals from the NMS layer and the feature maps from VGG as an input, and it will extract the features from backbone based on the proposals generated by RPN and filtered by NMS. We used RoI to utilize a single feature map for all the proposals generated by RPN in a single pass. Which solves the problem of variety in dataset images shapes.

Our layer will produce fixed-size feature maps from non-uniform inputs by doing max-pooling on the RPN proposals and filtered by NMS stage, where it applies max pooling with pool size of 7x7, which means the height and the width of the proposal will be divided by the pool width and pool high. Since the proposals define the regions which most likely contain an object, the features extracted from RoI based on these proposals will be the main features to focus on inside the image, which we get after applying max pooling. The features extracted from this stage will be the input for the last stage which is Fast R-CNN or the detection network.

4.2.6 Detection Network

The features extracted from the RoI pooling stage is the input to our detection network, which is mainly a Fast RCNN model. The Detection network mainly built from three convolutional layers, one fully connected layer with Relu activation functions is shared between the classification and regression, and an independent two fully connected layers used to build classification and regression stages.

as mentioned before, the shared fully connected layer build using Conv2D from keras with 3x3 kernel size and ReLu as activation function, for the classification stage, a fully connected layer is also build with Conv2D with softmax as activation function, the reason that the softmax function is used is to normalize the outputs, converting them from weighted sum values into probabilities that sum to one, where each value in the output of the softmax function is interpreted as the probability of membership for each class, the categorical cross entropy is used to calculate then loss in the detection network classification. The regression stage consists from a fully connected layer, built with Conv2D with a linear regression, where the regression predicts the bounding box correction, with reference to ground truth box.

Most of the previous stages are used in the training and testing phases. In the training phase, we've trained Scripty over 40 epochs, where each epoch consists of 1000 batches, so Scripty is trained over 40 thousands batches, where each batch has the size of the training dataset. The number of epochs is the number of complete passes through the training dataset. Number of batches represents the number of samples processed before the model is updated.

In order to train Scripty, we used the alternating training approach, where we train the RPN and detection network each one separately. First, RPN is trained to generate region proposals. The weights of the shared convolutional layer are initialized based on a pre-trained model on VGG16, and the other weights of RPN are initialized randomly.

After the RPN produces the region proposals, the weights of both the RPN and the shared convolutional layers are tuned. The generated proposals by the RPN are used to train the Fast R-CNN module. In this case, the weights of the shared convolutional layers are initialized with the tuned weights by the RPN. The other Fast R-CNN weights are initialized randomly. While the Fast R-CNN is trained, both the weights of Fast R-CNN and the shared layers are tuned. The tuned weights in the shared layers are again used to train the RPN, and the process repeats. The model will save all the trained values in the checkpoint and record files and file with HDF5 format after training each epoch, this helps to continue the training process if any interruption happens.

4.3 Experiments

A series of experiments were applied to choose the deep learning algorithm we used. Since as known, there are no specific rules in deep learning that we should follow in order to have the best results, it highly depends on running many trials and making many changes either on the dataset such as data augmentation or on the model such as changing some hyper-parameters. We thought at first to do the characters segmentation phase and then enter those letters to CNN model in order to recognize them but we realized that this way isn't very efficient in our case so we decided to use another algorithm, which is Faster RCNN.

Many experiments were done to test our implemented Faster RCNN model. The experiments were as follows:

1- Characters segmentation and CNN

In this experiment we tried to build our deep learning model in two phases: characters segmentation then characters classification. We tried using a watershed segmentation algorithm, and many different bounding windows implementations to do the words segmentation but we found it difficult to segment the words into characters since there is a very large overlap between characters and the handwriting font was very cursive.

The aim was to apply segmentation algorithms to get segmented characters to train and test over, and since IAM contains words with different occurrences for each character, we used the EMNIST dataset to increase the size of trained images for each character.

We tried to build different CNN models by changing the numbers and types of layers and noticed the effect of this change on accuracy and loss function. Also, we tried to solve the problem of overfitting so we applied by class weighting since the classes were unbalanced, i.e.: there was some classes contains 20,000 photos and some contains 7,000 photo, so, by using this technique we gave higher weight for classes that have less number of photos, we also tried to resize the image and see this effect on accuracy. Finally, we achieved an accuracy of 88% on 10 epochs.

2- Faster RCNN over IAM dataset only

After a lot of research, we decided to use the Faster RCNN algorithm on the IAM dataset, which contains 7000 images labelled using LabelImg, where each image consists of a single word.

Initially, we built our first version of Faster RCNN with default parameters mentioned in the original paper of Faster RCNN, after visualizing the results, we tried to improve our model via controlling the hyper-parameters. After many tries we came to get the best results for IAM dataset when we set the values of anchor box scales to 128,256,512, the overlapping threshold used in NMS to 0.7, RPN min and max overlap threshold to 0.3,0.7 respectively and classifier min and max overlap threshold to 0.1,0.5 respectively. FigureX shows a sample output of this experiment, which is an image containing the word “which”.

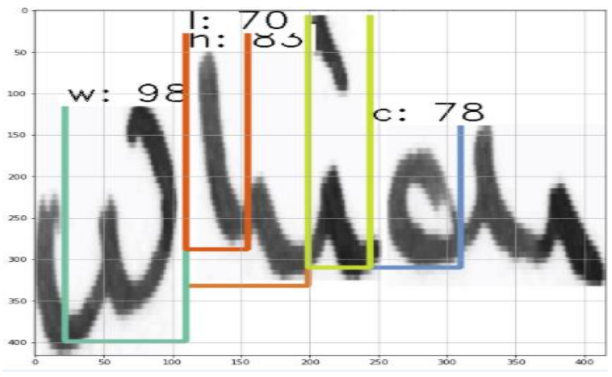


Figure 3.3. Output samples from experiment 2

Based on our results for this experiment, we noticed that the model still suffering from overlapping issues, and the low number of training samples for each char, for i.e Figure 4.3 shows that the model has defined the first ‘h’ and doesn’t recognize the second one, and the reason may be the lack of training samples for ‘h’ or the thresholds defined in the NMS where it neglects any bounding box crossed outside the image boundaries.

For improving reasons, we decided to merge the characters dataset EMNIST with the IAM in order to enhance Scripty to be able to recognize all characters in any position and improve the mAP for each character. We must mention that we achieved mAP = 19 % in this experiment.

3- Faster RCNN over IAM and EMNIST

Deep learning algorithms as known, are always data hungry. So, as we add more data to the dataset, the model will have a better performance and therefore will give better results. In this experiment, we tried to solve the issues related to the lack of dataset by merging the EMNIST with the words dataset IAM in order to avoid some problems that resulted in the previous experiment. We used the same parameters used in the second experiment.

Since the EMNIST dataset has fixed sizes for each image, and upper case has the same shape as lower case for most of the characters, the classifier faced issues in classifying these characters, we must mention that overlapping issues occurred in our results and we achieved mAP = 22%

In order to increase the precision, and handle cursive character more efficiently, we applied data augmentation, which means sharing the images and rotating images with different angles in order to enhance the model and make it more efficient in recognizing the letters that have more than one way of writing.

4- Faster RCNN over IAM and EMNIST with data augmentation

In this experiment, we applied augmentation steps during the training phase, in order to increase the number of characters for each class. We applied augmentation over all characters, but more on characters with less occurrences in the dataset, for e.g. Capital letters, Table 4.1 shows the statistics used in this operation.

We applied two main operations in data augmentation, the shearing and rotation processes. Shearing is applied over characters, so one part of the image is shifted like a parallelogram, this process helps us to create a new augmented image for the original character, which shapes differently and tends to shape like cursive characters. We also applied horizontal rotations with very small angels in order to handle more cases like italic cases. This experiment was the final experiment of our work, and we will discuss the results in details in section 4.4.

4.4 Results and Evaluation

Per our last experiment, we have trained Scripty on a merged dataset with augmentation during the training process. Scripty trained using the alternating training approach of Faster RCNN, so RPN and detection networks are trained separately. The RPN and detection network perform classification and regression operations before they pass their outputs, so the main aim is to train the latter to perform efficiently.

Figure 4.4 shows RPN classification and regression operation loss values, and how it changed due to increasing total epochs trained. We clearly notice that the classification loss was the highest in the first epoch, around 0.55, and it decreases exponentially with the increases of total epochs trained. Which is predicted since the more epochs trained on the images the more this classification layer will learn and be able to detect the existence of an object in a specified box. The loss of RPN regression is also decreasing with more epochs trained, this predicts that the RPN will generate proposals around objects with the highest probability of containing an object.

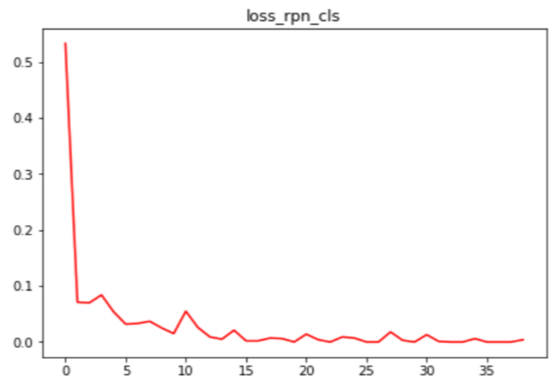


Figure 4.4.A RPN classification Loss

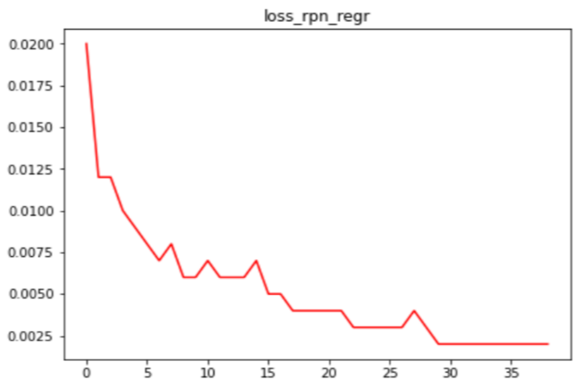


Figure 4.4.B RPN Regression Loss

Figure 4.4. RPN Loss

On the other hand, the loss of the detection network classification and regression is decreasing exponentially, which something predicted. The detection network will be trained over proposals from the RPN which will become more confident about proposals generated while the number of epochs trained increases. Figure 4.5 shows how the values of classification and regression loss change due to an increased number of epochs trained. The figures show that the detection network is good at classification since the loss decreased from 2.25 to 0.33 after 40 epochs trained, figure 4.5.c shows that the network has an accuracy of 87% over the 53 classes.

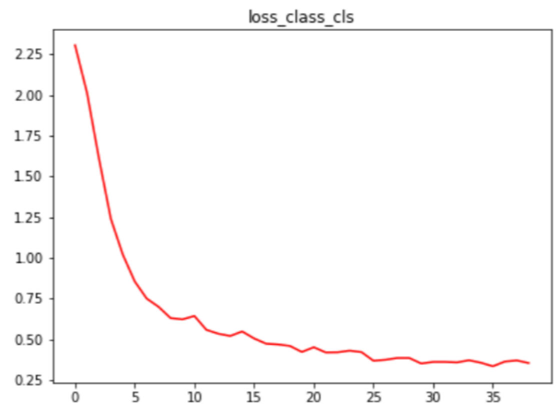


Figure 4.5.a Detection network classification Loss

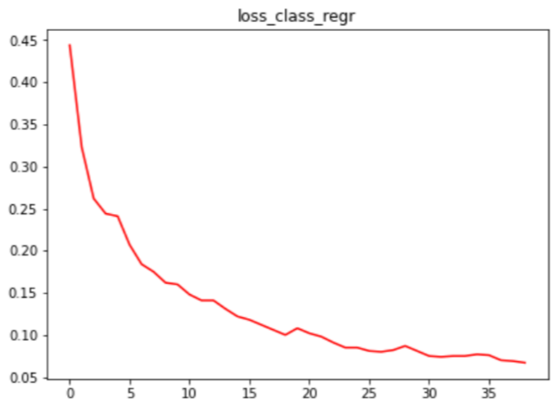


Figure 4.5.b Detection Network Regression Loss

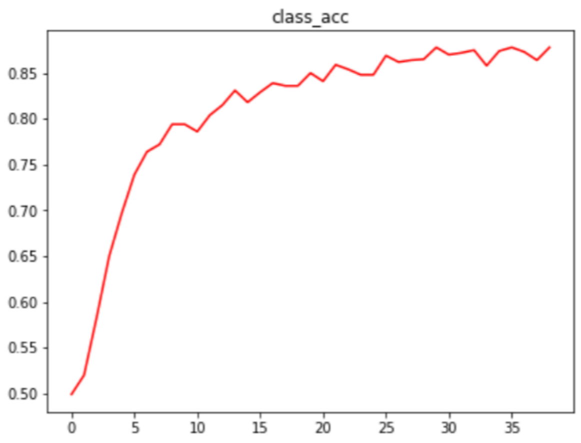


Figure 4.5.c Classification accuracy of Detection Network

Figure 4.5. Detection Network Losses

The term total loss defined in Faster R-CNN to be the summation of the loss of the RPN and the loss detection network stages, we can tell, the more the loss is decreasing, the more Scripty will become efficient in recognizing and detecting characters. Figure 4.6 shows how the total loss is decreasing when the number of epochs trained is increasing, which represents the summation of losses from training the RPN and detection network.

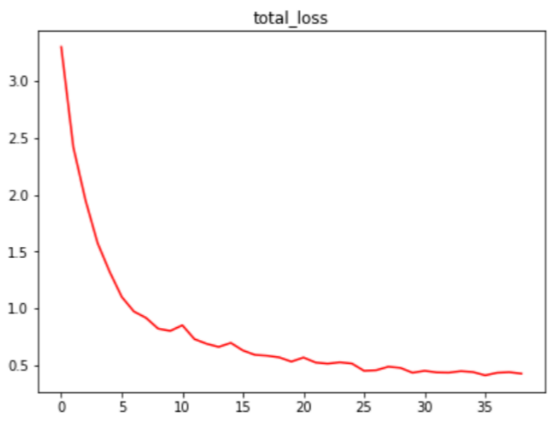


Figure 4.6. Scripty training's total Loss

The Faster R-CNN testing evaluation is mainly measured by mAP, which presents the mean value for APs for the 53 classes. The AP relates the order and confidence of the classification proposals for each bounding box to the classes across all images. The AP of each capital letter class for our test data is shown in Figure 4.7.a and for small letter classes in Figure 4.7.b. The mAP over all 53 classes is 30%, with lowest AP being 'x' 10% and the highest APs being 'c' 95%.

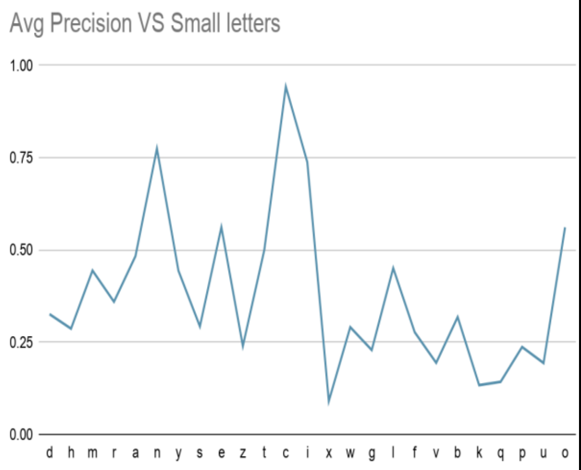


Figure 4.7.a AP for each small character

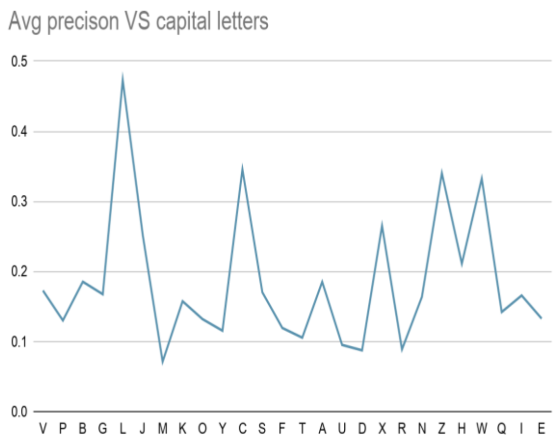


Figure 4.7.b AP for each capital character

Figure 4.7. AP for each character

One very important factor for getting the achieved results is the number of training images for each letter, this highly affects the training model and probability of predicting each letter. Put simply, letters that are used more in the English language, such the letter 'e', would be more common in word images from IAM dataset, which means the number of training images for this letter would be more.

Having more images for training raises the probability of detecting this letter when testing. This can be seen in our results as the letters with higher number of training images got higher mAP, which means it was predicted correctly most times it appeared in a tested image. e.g. the small letter ‘e’ had 3570 training images as can be seen in figure 4.8 and got 56 %as can be seen in ,this is one of the highest AP we got for small letters.

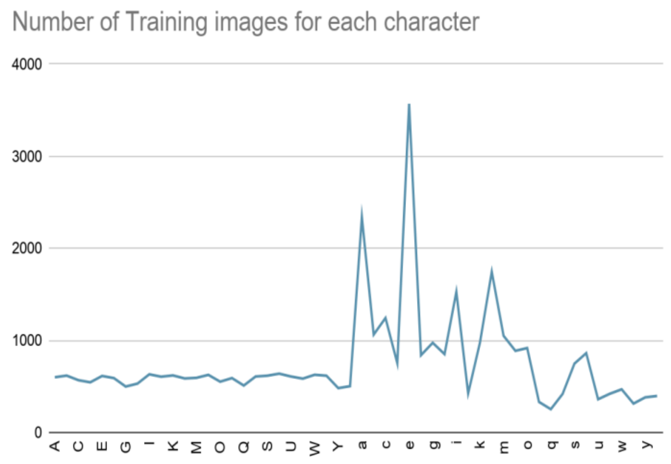


Figure 4.8. Number of Training images for each character

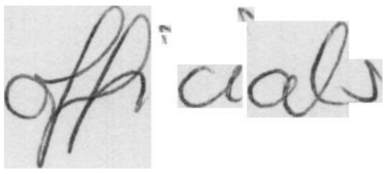
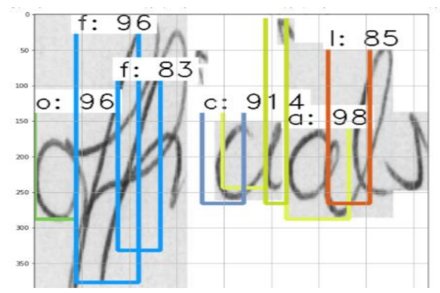
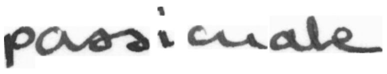
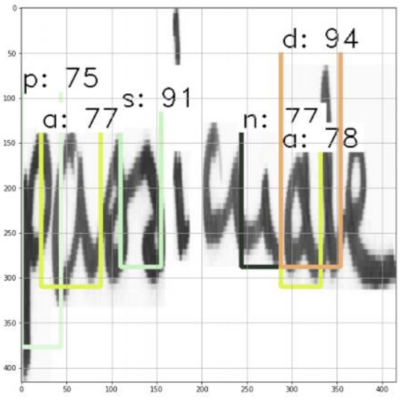
Figure 4.8 shows that all capital letters have the same number of trained images, this is mainly because of the lack of use of capital letters in English writing, since its mostly only used when the letter comes at the beginning of the sentence; in IAM dataset, most of the words are taken from a long paragraph, so they don't usually start with a capital letter. This also contributes to the low APs for capital letters as seen in figure 4.7.b, where the highest AP was for the letter ‘L’ and it was 47%.

So while trying to balance the dataset and increase the number of images for training the capital letters, we added images from EMNIST dataset, and they were mostly the same number of images for capital letters. Images in EMNIST dataset as mentioned before, have the exact size for capital and small letters, so when classifying those images into their right classes, if the capital and small letter has almost the same shape, it will be confused. In addition, augmentation was done on capital letters images with the same number of generated images for each. All those reasons led to the numbers of capital letters being almost the same.

We can also conclude from figure 4.7.a that the small letter ‘c’ got the highest AP and this is due to the fact that the shape of the letter c is very distinctive and isn't very similar to other letters, other than the capital letter ‘C’. we notice that letters which have similar shapes to other letters or some combination of letters, had less accuracy and AP. e.g. The letters ‘at’ in the word ‘passionate’ in table 4.9 in cursive handwriting can have a similar shape to the letter ‘d’, which explains how Scripty predicted it to be a‘d’ with 97% probability, which means some of the overlapping issues still occur in Scripty.

Another important conclusion from the training results is the small letters had better AP that capital letters. This purely due to the fact the small letters appear way more in words and in our dataset the images for small letters were noticeably higher than those for capital letters, and it is known that the more data the model consumes the better the prediction is going to be.

Table 4.3 contains some samples from the output of Scripty, each cell shows the tested image, the bounding boxes around the objects inside the image with their percentages and the elapsed time in testing.

Tested Image	Predicted letters and their percentage	Elapsed time
 **{officials}**	 [('a',98), ('a',84),('l',85),('o',96), ('f',96),('f',83),('c',91),('i', 79)]	2.810664077
 **{passionate}**	 [('p',75),('a',78),('s',91), ('n',77) ,('a',77),('d',94)]	2.720199346

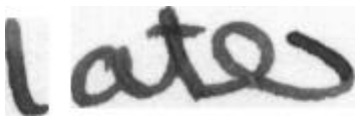
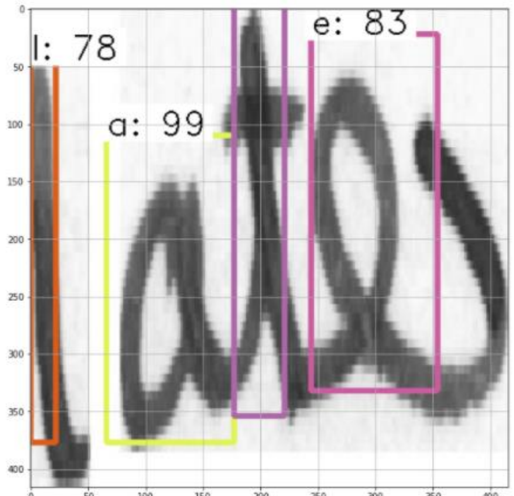
 **{late}**	 [('l',78),('a',99),('t',77), ('e',83)]	2.9524142742
Results for Sentence		

Table 4.3 Scripty output samples

Chapter 5: Conclusion, Challenges, and Future Work

5.1. Conclusion

Scripty is a deep learning model that recognizes cursive handwriting in an easy and efficient way which will be very helpful in the recognition process of any cursively handwritten document especially the important ones such as the medical prescriptions and bank cheques. So, it will be very useful for pharmacists, bank employees or anyone who would like to read any handwritten document clearly.

Scripty deploys neural networks implementing a faster RCNN approach to object detection, which is a new innovation in the deep learning field. Scripty takes the object detection route instead of the ordinary techniques used for most OCR models. During the development of Scripty, many experiments were made to finally arrive at the best approach for cursive handwriting recognition. Scripty is able to detect, localize and classify alphabetic characters in a given image that contains any cursive handwriting.

Scripty mainly shows that neural networks such as CNN, RNN, CRNN and RPN alone are insufficient in detecting cursive handwriting, while they may be a good approach for detecting handwriting when it's separate and with context, it falls short for detecting cursive and context free handwriting. Furthermore, Scripty proves that object detection algorithms, especially faster RCNN, are the way to go for cursive handwriting recognition as they evidently integrate all stages needed in recognizing handwritten words in one fast and efficient way.

5.2. Challenges

During the development of Scripty, many challenges were faced. One of the hardest challenges we had was the processing of the dataset in order to be used in Scripty. The first Dataset dilemma we faced is that we initially thought of making our own dataset by collecting cursive handwritten images, mostly from doctor's offices, clinics and from local pharmacies. However, this option was impossible because of the global crisis that we're facing these days which is coronavirus pandemic.

The second dataset dilemma was that even the online datasets that we managed to have IAM and EMNIST were un-labelled datasets, so we had to label those datasets by ourselves using labelling and python codes.

Moreover, the unbalanced dataset was a major problem that we faced while we were training Scripty , for example : “a" occurs in almost every word in English , for that reason the model has recognized it much better than “z” which has very less percentage of occurrence in English words.

The upper-case letters were also a major issue in Scripty since the upper-case letters occur only at the beginning of the word. We tried to solve those issues using many data manipulation techniques such as augmentation that was mentioned before. However, those techniques didn’t give perfect results since it’s the nature of language which we can’t control, we may solve this issue in the future by generating our own words which mainly contain the least frequently used letters (z, w, v) and words that only contain upper-case letters in order to train Scripty in detection of even the rare characters.

One more challenge we faced was related to the Faster RCNN algorithm which is a relatively new algorithm and doesn’t have so many resources compared to traditional neural networks like CNN or RNN, Etc., especially in handwriting recognition. So, it was a totally new topic for us, since none of us has worked with it before. So, we spent a lot of time trying to understand the architecture by attending online courses and reading a very large number of papers which was a very challenging but interesting experience.

5.3. Future works

We are looking to train Scripty on a larger dataset when we are able to get our hands on one, either by making it or by developing a model to generate more cursive-like word images and use it for training; since a deep learning model is always data-hungry. The bigger the dataset, the better accuracy we will get. The latter may be done using Generative adversarial networks (GANs), which are an exciting recent innovation in machine learning that we are looking forward to starting to research into and maybe use for Scripty in the future. GANs are generative models: they create new data instances that resemble your training data. Thus we could be able to generate new cursive words based on the images we already have.

We are also considering applying a post-processing stage on our predicted letters in order to find the best corresponding word for those letters. We plan to do this using English language models/algorithms that take the predicted letters and make all possible combinations according to their order and predict the most accurate words.

This would be very helpful for our next plan of developing a mobile application based on Scripty to be able to scan a cursive handwritten paper like a doctor's prescription and then process it and save everything written in this picture in a document that is later edited or shared by the user.

We are also considering making this project possible to use by Arabic speaking users, this may require us to make our own Arabic dataset. However, this would be a very new innovation for the Arabic handwritten documents since it's all cursive.

References

[1] Anonymous. 'Fact of the matter'. Bangalore Times (a supplement of The Times of India, Accessed:2020-03-15 5:13pm.

[2]MNIST. http://yann.lecun.com/exdb/mnist/

[3]. LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11, pp. 2278-2324, 1998.

[4]. Ciresan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 3642-3649. IEEE, 2012.

[5]. Ciresan, Dan Claudiu, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. "Deep, big, simple neural nets for handwritten digit recognition." Neural computation 22, no. 12, pp. 3207-3220, 2010.

[6]. G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Li and T. Hospedales, "When Face Recognition Meets with Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition," Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 142–150, Santiago, Chile, 13–16 December 2015.

[7]. K. Younis and A. Alkhateeb, "A New Implementation of Deep Neural Networks for Optical Character Recognition and Face Recognition," Proc. of the New Trends in Information Technology (NTIT-2017), The University of Jordan, 25-27 April 2017.

[8]:Brits, H., et al. "Illegible handwriting and other prescription errors on prescriptions at National District Hospital, Bloemfontein." South African Family Practice 59.1 (2017): 52-55.

[9] Suddath, Claire. "Mourning the death of handwriting." Time Magazine 174.4 (2009).

[10] Rodriguez-Vera FJ, Marin Y, Sanchez A, et al. Illegible handwriting in medical records. J R Soc Med. 2002;95(11):545–6. PMID: 12411618, PMCID: PMC127925

[11] https://www.slideshare.net/ExtractConf/andrew-ng-chief-scientist-at-baidu

accessed on 23/3/2020 6:45pm

[12]
https://play.google.com/store/apps/details?id=com.fourtechsolutions.ocr_reader_ocr_scanner_ocr_text_scanner&hl=en
accessed on 18/5/2020 10:51pm

[13] https://play.google.com/store/apps/details?id=p2p.serendi.me.p2p&hl=en
accessed on 18/5/2020 10:55pm
[14]https://www.semanticscholar.org/paper/MediPic%3A-A-mobile-application-for-medical-Alday-Pagayon/19bd8c7957b32f0ccc613eba07b6ec9d6958916a
accessed on 18/5/2020 10:51pm

[15] Attigeri, Savitha. "Neural Network based Handwritten Character Recognition system." International Journal of Engineering and Computer Science 7.03: 23761-23768, 2018.

[16] Dhande, Pritam S., and Reena Kharat. "Character Recognition for Cursive English Handwriting to Recognize Medicine Name from Doctor's Prescription." 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA). IEEE, 2017.

[17] Soullard, Yann, et al. "Improving text recognition using optical and language model writer adaptation." 2019.

[18] R. Achkar, K. Ghayad, R. Haidar, S. Saleh and R. Al Hajj, "Medical Handwritten Prescription Recognition Using CRNN," 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Beijing, China, 2019, pp. 1-5.

[19] Kamalanaban, E., M. Gopinath, and S. Premkumar. "Medicine Box: Doctor's Prescription Recognition Using Deep Machine Learning." International Journal of Engineering & Technology 7.3.34 (2018): 114-117.

[20] Y.Ou, S. Tseng, J. Lin, X. Zhou, J. Wang and T. Kuan, "Automatic Prescription Recognition System," 2018 International Conference on Orange Technologies (ICOT), Nusa Dua, BALI, Indonesia, 2018, pp. 1-4.

[21] http://cs231n.stanford.edu/reports/2016/pdfs/269_Report.pdf accessed 15/10/2020

[22] https://fki.tic.heia-fr.ch/databases/iam-handwriting-database
accessed on 25/5/2020 11:55pm

[23] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." *Icdar*. Vol. 3. No. 2003.

[24] http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 accessed on 24/5/2020 10:01pm

[25]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 accessed on 24/5/2020 10:49pm

[26]. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

[27]https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e accessed on 24/10/2020 3:45pm

[28]Ross Girshick. “Fast R-CNN.” In Proc. IEEE Intl. Conf. on computer vision, pp. 1440-1448. 2015.

[29] https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9 accessed on 25/10/2020 on 7:35pm

[30] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016): 1137-1149.

[31]https://whatdhack.medium.com/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd?fbclid=IwAR1Us_-Uc5LhWxZB7Ycim9WQVW1ZwcxXv3cMl_ACpVN5OsmxyaSAFywcl8M accessed on 27/10/2020 on 6:30pm