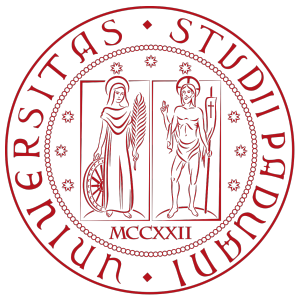


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics of Data

Final Dissertation

Detection and recognition of textual information from drug box images using deep learning and computer vision

Internal advisor

Prof. Marco Baiesi

Internship supervisor

Valentina Vivaldi, PhD

Candidate

Camilla Quaglia

Academic Year 2021/2022

Contents

Introduction

The advent of artificial intelligence is having a major impact in both research and business world. During the last decade machine and deep learning techniques have become a huge research topic in both academia and R&D department of companies. Moreover a fast growing number of start-ups is involved in artificial intelligence, whose application are everywhere. Just to name a few, speech recognition applications are very popular in people daily life, such as *Google Home* or *Amazon Alexa*, together with search engines apps, used for searches based on pictures taken with handheld devices. Nowadays famous automobile manufacturing companies are paying attention to autonomous vehicle, developing working prototypes of autonomous car. Other AI applications are recommendations systems on streaming services or email spam filtering. These are emblematic examples of how artificial intelligence is taking hold in people's lives and its influence is set to grow, not without ethical issues. To mention some numbers, according *Tractica*, the annual global AI software revenue will grow from \$10.1B in 2018 to \$126.0B by 2025, at a rate of 43.41%. Also, the global machine learning market was valued at \$1.58B in 2017 and is expected to reach \$20.83B in 2024, growing at a rate of 44.06% between 2017 and 2024 (from *Forbes*).

Also in science AI is having a strong impact, with applications in high energy physics, medical diagnosis and even climate science. However, there are cases in which the AI revolution does not parallel with a deep understanding of how intelligent algorithms really work. Sometimes they can be considered *black box* models that are not fully interpretable and therefore not reliable for delicate tasks, for instance in the medical field. A physicist's mindset can borrow rigour and *scientific method* to develop and interpret AI techniques. A formal definition of the parameters of a model together with clearly-defined processing techniques, all in a rigorous pipeline, is the value that a scientist and in particular a physicist can bring into this world.

The scope of this thesis work is to implement an OCR pipeline, capable of detecting and recognizing text instances when an image is given as input. The pipeline is divided into two steps: a detector, which scope is to detect the regions where a text is present, and a recognizer, which scope is to recognize and read the detected words and numbers. The work was initially developed during the internship experience in the start-up *PatchAI*, now *an Alira Health* company. The application of the algorithm in this context is the recognition of textual information on drug boxes. The idea is to deploy such pipeline into an app support, in such a way it can be used by patients, who can take a picture of the box and receive information about the medicine, in particular its posology. Also the use of a vocal assistant that reads orally the recognized text is explored, being a interesting application for ederly or visually impaired people.

The thesis is organized in four chapters. The first one explores what is OCR and which is the role of deep learning in the field. Applications of deep learning based OCR are analyzed in chapter two. Use cases in both physics and business world are inspected. Chapter three pays the attention to the choice of the dataset to train the algorithm with and discuss the available open source OCR databases. Therefore the initial exploratory data analysis is presented. The details and the architecture choices behind the implementation are discussed in chapter four, where the results and the performance on medicine boxes are shown.

Chapter 1

Deep Learning for OCR

1.1 What is OCR

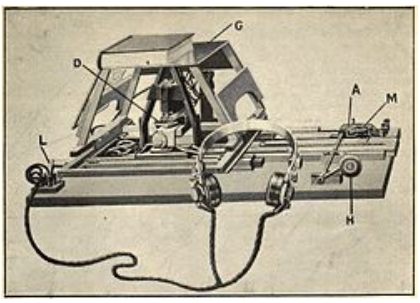
Optical character recognition (OCR) is the method of converting text, both handwritten and printed, present in images or scanned documents, to a machine-readable format. It is one of the earliest addressed computer vision tasks, implemented even before the deep learning boom in 2012.

1.1.1 A bit of history

The roots of OCR are in telegraphy and in the creation of reading devices for blind people. [1] On the First World War eve the physicist Emanuel Goldberg invented a machine that could read characters and convert them into telegraph code. In the late 1920s and into the 1930s he developed what he called a *Statistical Machine* for searching microfilm archives using an optical code recognition system. For his invention, he was granted a USA Patent in 1931. The patent was than acquired by IBM.

During the same period in 1914, Edmund Fournier d'Albe developed the so-called *Optophone*, a scanner that when moved across a printed page produced tones that corresponded to specific letters or characters.

Figure 1.1: *The optophone*



In 1974 Ray Kurzweil started the company *Kurzweil Computer Products, Inc.* to continue the development of omni-font OCR, which could recognize text printed in any font. Kurzweil decided that the best application of this technology would be to create a reading machine for the blind, which would allow blind people to have a computer read text to them out loud. This idea required the invention of two newer technologies: the CCD flatbed scanner and the text-to-speech synthesizer. The aforementioned successful product was unveiled in 1976 by Kurzweil himself and the leaders of the National Federation of the Blind. In 1978, Kurzweil Computer Products began selling a commercial version of the optical character recognition computer program. *LexisNexis* was one of the first customers, and bought the program to upload legal paper and news documents onto its nascent online databases. Two years later, Kurzweil sold his company to *Xerox*, which had an interest in further commercializing paper-to-computer text conversion.

In the 2000s, OCR was made available online as a service (*WebOCR*), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs. With the advent of smart-phones and smartglasses, OCR can be used in internet connected mobile applications that extract text in images captured using the device's camera.

1.2 The role of Deep Learning

As the previous section remarks, there are certain OCR tasks that do not require deep learning. The power of AI applied to such tasks lays in the generality of the developed pipelines, that can be applied in different scenarios, from airports passports checking to traffic signs recognition in self driving cars. For instance, nowadays OCR remains a challenging problem when text is found in natural scenes. The text in such environment could present geometrical distortions, complex background and also diverse fonts. With the advent of deep learning it is possible to deal with these kind of tasks, having the possibility to train algorithms on a significant volume of data.

Moreover using deep learning one can make the OCR algorithm robust to a certain feature according to the scope of the pipeline, exploiting data augmentation techniques or synthetic training data generation. A concrete example is given in the work [2], where the authors use data augmentation methods like gaussian smoothing, perspective distortions, morphological filtering, downscaling and additive noise on training samples, in order to recognize textual information occurring in complex natural environments. To extract features from the samples a convolutional neural network encoder is exploited and to recognize texts a recurrent neural network model. The work presents an OCR system that surpasses the accuracy of major commercial and open source engines on distorted text samples.

A scenario in which the use of deep learning makes OCR reaching performances never seen before is the recognition of texts embedded in videos. The paper [3] proposes a method which makes use of convolutional neural networks and Deep Auto-Encoders. Features are extracted using a multi-scale sliding window scheme. A recurrent approach is then used to recognize characters. The pipeline is tested on a large video database from several Arabic TV channels.
More precisely, as a first step the input image is scanned with a multi-scale window-based scheme, in order to represent the text image by a sequence of learned feature vectors. The paper proposes two ways to perform feature extraction and it worths to discuss them, as they are common to several deep learning OCR pipeline, despite the architecture exploited in this thesis is analyzed deeply in chapter four. The first way is based on deep auto-encoders while the second one is a based on convolutional neural network.

Deep Auto-Encoders

An auto-encoder is an architecture composed by an enconder and a decoder, that in principle can be a feedforward or a convolutional network. Given an input image, the scope of the auto-encoder is to learn to produce an output that is identical to the input. In particular the encoder extracts features from the image and encode them in a *latent space* or *feature space* and then the decoder reconstructs the input. The network, showed in figure 1.2, is trained using the backpropagation algorithm. The images representation in the feature space can be exploited to perform a characters features-based classification, in a totally unsupervised setting.

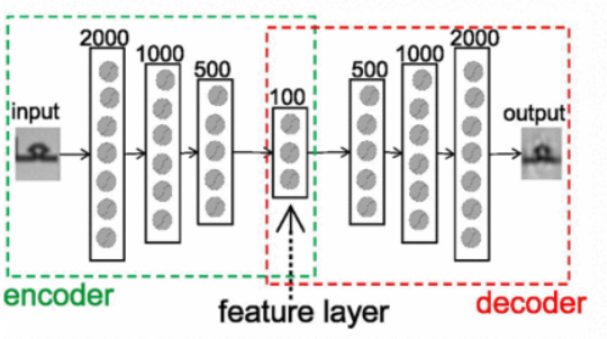


Figure 1.2: *Auto-encoder architecture. Figure from the paper [3]*

Convolutional Neural Network

A Convolutional Neural Network is a hierarchical network that has been successfully used for pattern classification. It is able to extract features from an image using a pipeline of convolutions, sub-sampling and neural layers. The architecture includes a convolutional part, in which a kernel scans the image to produce some *feature maps*, on which the chosen *activation function* act. During training the coefficients of such kernel are learned. The feature maps are then downsampled, through a *pooling* layer, reducing the dimensions and the computational cost and obtaining the input for the next convolutional layer. At the end of the convolutional part, a fully connected part is exploited to obtain one number (the class of the input) out of the image. In the paper [3], a Multy-layer Perceptron with one single hidden layer is used for performing the classification of arabic characters.

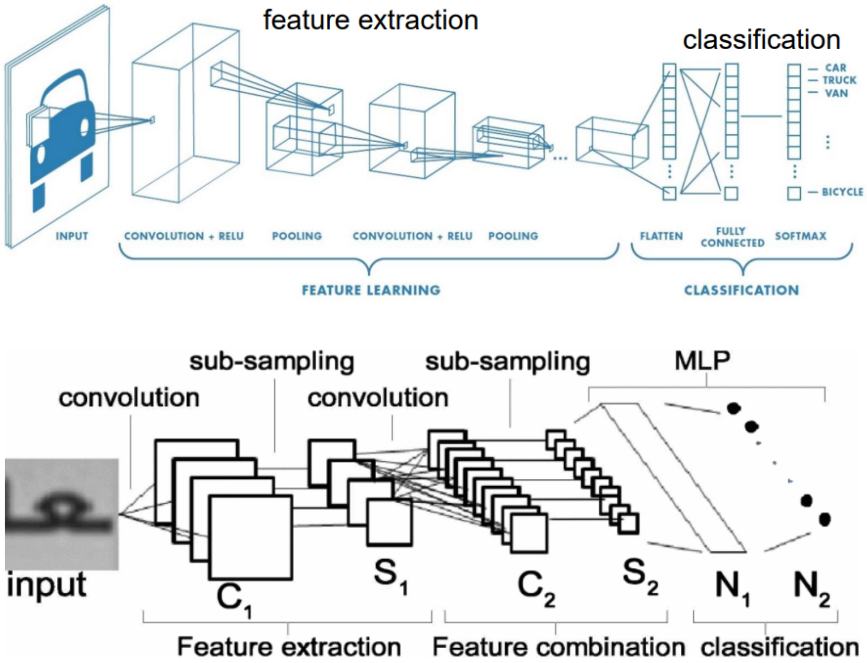


Figure 1.3: *Top: CNN general architecture. Bottom: CNN architecture used in the paper [3]*

Recurrent Neural Networks and LSTM

Regarding the text recognition task, in several works, including the aforementioned and this thesis, recurrent neural networks are exploited, in particular Long Short Term Memory (LSTM) networks.

Without losing generality, RNN are networks in which the temporal order of the inputs matters, since for each hidden layer there is a *context layer*, which is a copy of the hidden at a previous time (figure 1.4). At each time step t, the output of the network takes into account not only the input at time t but also the information at a previous time $t-1$, as the following equation clarifies

$$\begin{aligned} \mathrm{h}_t &= f\,(\,\mathrm{W}_1\mathrm{x}_t + \,\mathrm{W}_3\mathrm{h}_{t-1}) \\ \mathrm{o}_t &= g\,(\,\mathrm{W}_2\,\mathrm{h}_t) \end{aligned} \tag{1.1}$$

where o_t is the output at time t, x_t is the input at time t, h_t is the state of the hidden layer at time t, h_{t-1} is the state of the context layer, f and g are the activation functions.

A RNN with the structure above can learn a sequence of letters. For instance, if the network is trained with texts corpora in a given language, it can learn to predict the next coming letter in a word given as input. Back in the 1990, in the famous work [4], a RNN model is trained with english text with the scope of predicting english words given the onset letters, as figure 1.5 shows.

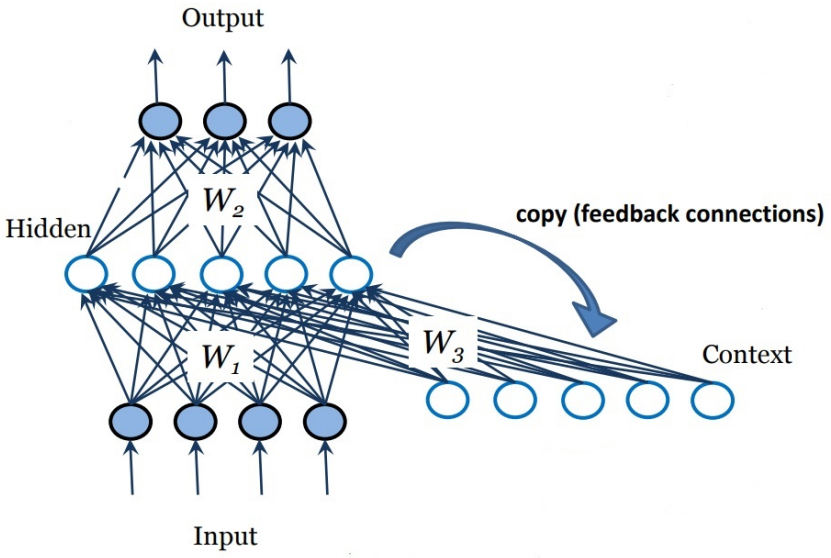


Figure 1.4: *Simplest architecture of a Recurrent Neural Network*

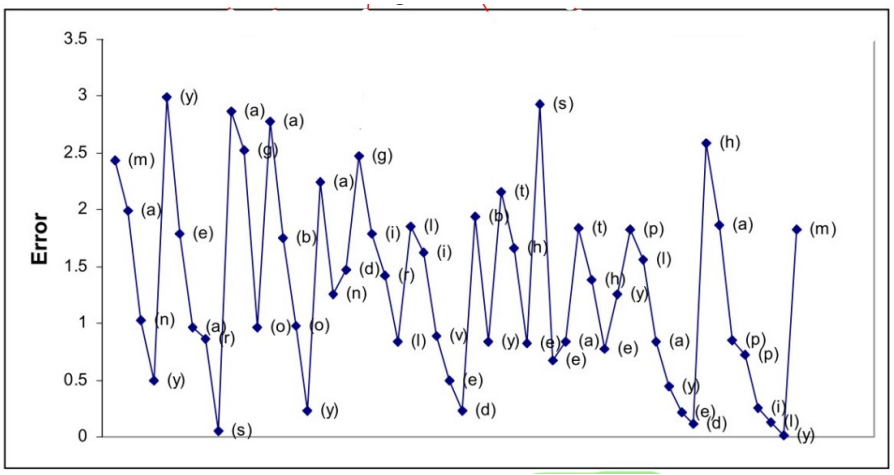


Figure 1.5: *Performance of a Recurrent Neural Network which scope is predicting english words. Note that at the first letter of the word the prediction error is higher and keeps on decreasing as the word goes ahead. Figure from the paper [4].*

The problem of RNN are the long-term dependencies, which means that the model cannot store in memory information referring to arbitrary distance in time. A solution is represented by Long Short Term Memory networks, that are exploited in the recognition tasks of OCR pipelines. The following explanation of LSTM is inspired by this work [5]. are characterized by additional *gated units* with respect to classic RNN, with the scope of learn how much information should be retained in a temporary memory. They were introduced by Hochreiter & Schmidhuber in 1997 [6]. In general RNN can be represented with a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a single *tanh* layer. Also LSTMs have this chain like structure, but the repeating module has a more complex structure. In fact there are four interacting layers, as figure 1.6 shows.

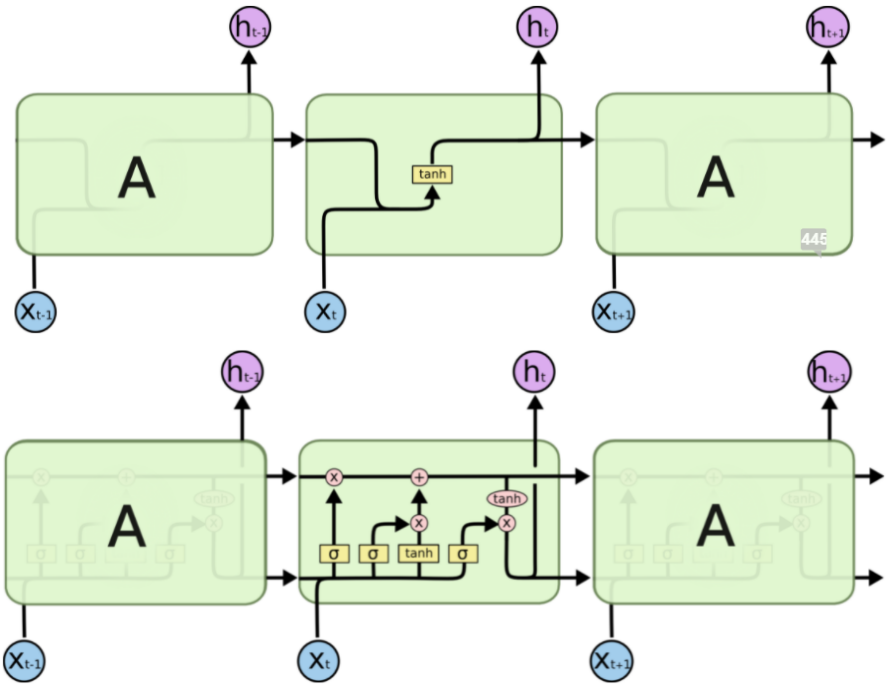


Figure 1.6: *Top: Repeating module of a standard RNN, with one single layer. Bottom: Repeating module in a LSTM, containing four interacting layers.*

The notation to understand the diagram is:

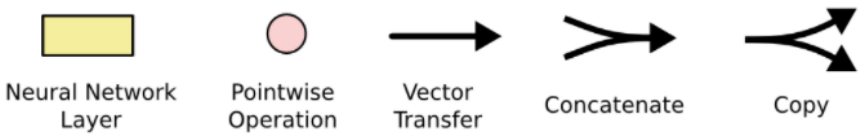


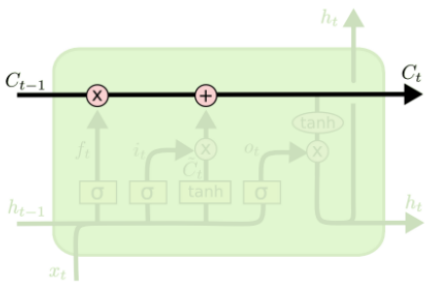
Figure 1.7: *Notation*

In the drawn schema, each line carries an entire vector, from the output of one node to the inputs of others.

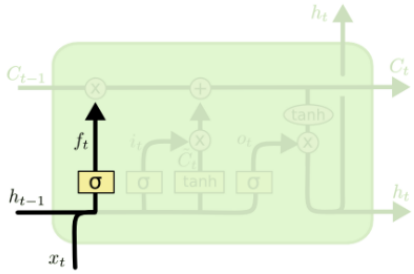
The first piece of the module is the *cell state*, the horizontal line running through the top of the diagram (figure 1.8). It runs along the entire chain, with only some minor linear interactions. It can happen that information just flow along it unchanged.

Information can be removed or added to the cell state, regulated by the so-called gates. They are composed of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, with the meaning of *how much information should be let through.* A value of zero means *let nothing through* while a value of one means *let everything through.*

Figure 1.8: *The cell state*



There are three types of gates: input gate, forget gate and output gate. The *forget gate* is a sigmoid layer that decide what information should be thrown away from the cell state.
It receives an external input x_t and the input from the previous block h_{t-1} and outputs a number in $[0, 1]$.

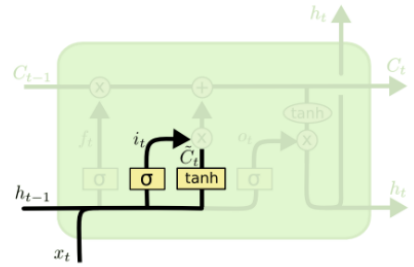


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 1.9: *The forget gate: σ is the sigmoid activation function, W_f is the matrix of weights associated to that gate and b_f is the bias. Recall that $f_t \in [0, 1]$.*

Consider the example of a model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.

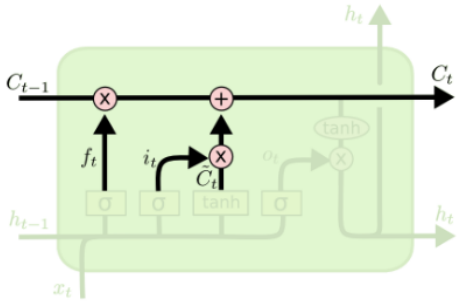
The *input gate* is again composed by a sigmoid layer, with the scope of deciding what new information should be stored in the cell state. Then a tanh layer creates a vector of new candidate values $\tilde{C}_t$. In the example of the language model, the scope is to add the gender of the new subject to the cell state, to replace the old one to forget.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 1.10: *The input gate.*

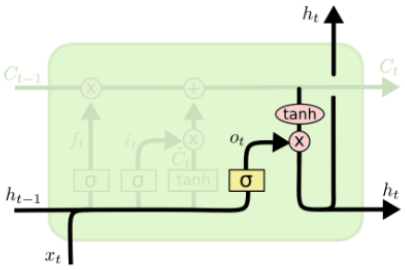
At this point the old cell state, C_{t-1} has to be updated into the new cell state C_t. The new candidate values is obtained by multiplying the old state by f_t, forgetting the information we decided to forget earlier. Then we add $i_t \cdot \tilde{C}_t$. In the case of the language model, this is where the information about the old subject's gender is actually dropped and the new information is added, as decided in the previous steps.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 1.11: *Updating of the cell state.*

Finally, the output will be a filtered version of the cell state. Specifically the *output gate* is formed by a sigmoid layer which decides what parts of the cell state will be the output. Then the cell state is passed through tanh (to push the values to be between 1 and 1) and multiply it by the output of the sigmoid gate o_t, getting the final output h_t.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 1.12: *The output gate.*

For the language model example, since it just saw a subject, it might want to output a verb, since it is likely to come next. For example, it might output whether the subject is singular or plural, so that it knows what form a verb should be conjugated into.

Chapter 2

Applications of Deep Learning based OCR

Despite OCR was born and applied before deep learning era, it is still a challenging problem especially when text images are taken in an unconstrained environment. As already mentioned, nowadays applications require OCR performing in challenging frameworks, including complex backgrounds, noise, lightning, different fonts and geometrical distortions in the image. To overcome such situations the help of deep learning is necessary. In the following chapter it is examined how OCR problems can be treated in a quantum computing framework, after a parenthesis of how AI is changing physics. Then interesting applications of deep learning based OCR are presented, both in physics and in the industry world.

2.1 Applications in physics

A core principle of physics is developing knowledge collecting and analyzing data. The experimental and simulated results are in fact the starting point to validate a physics theory or announce a new discovery. Thus, machine learning and deep learning have instantly entered physics and may become a new paradigm in basic and applied research.

2.1.1 How AI is changing physics

Artificial intelligence and machine learning help physicists in modelling and computing complex problems with greater speed and accuracy than ever before. For instance, recently *NVIDIA* and *IBM* published a paper [7] in Nature Astronomy to show an AI-driven method to detect gravitational waves. Moreover last year *Google Deepmind* brought a revolution in the protein structure modeling world with *AlphaFold*. *AlphaFold* is able to release the structure of every protein known to science, building a database that will provide researchers and biological physicists with a toolkit to improve the drug discovery process and better understand diseases. Another case in which deep-learning, in particular pattern recognition, allows to boost accuracy and speed is weather forecast. The work [8] shows how has been possible to make five-day forecasts of extreme weather with capsule neural networks. Capsule NN do not require as much training data as other neural networks. This was an ideal application due to the limited availability of satellite data and rare occurrences of the events.

An emblematic example of how machine learning helps physicts to extract and organize complex problems that involve a lot of data comes from particle physics. Nowadays the largest and most powerful accelerator ever built is the Large Hadron Collider (LHC) at CERN. In it, beams of protons are accelerated until they reach very high energies and then they are made to collide. The product of these collisions is a huge number of various subatomic particles, which are selected, located and tracked by dedicate hardware for further analysis. The amount of collisions detected at LHC is up to about 1 billion particle collisions every second, of which about 1 in 100'000 events is kept for future

analysis. The use of machine-learning techniques is revolutionizing how humans interpret these data samples. In particular to separate signal from background DNNs model are trained, reaching the most prominent performance with respect to previous approaches.

Also algorithms like Boosted Decision Trees are applied to particle identification in the MiniBooNE experiment operated at Fermi National Accelerator Laboratory (Fermilab) for neutrino oscillations [9]. In the Decision Trees nodes are used to categorize the events as either signal or background. Each node uses only a single discriminating variable to decide if the event is signal-like or background-like. This forms a tree like structure with leaves-nodes at the end. Training of a decision tree is the process to define the "criteria" for each node. Basing on training event sample the corresponding value that gives the best separation between signal and background at this stage is selected. Two new nodes are then created for each of the two sub-samples. The division is stopped once a certain node has reached either a minimum number of events, or a minimum or maximum signal purity. In Boosted Decision Trees, the selection is done on a majority vote on the result of several decision trees, which are all derived from the same training sample by supplying different event weights during the training.

2.1.2 Quantum information based character recognition

The paper [10] proposes a quantum algorithm for character recognition based on quantum superposition and quantum entanglement. The effectiveness of the proposed method is proven even in case of noisy input images. The algorithm works as follows.

A reference image is encoded into the following quantum state:

$$|\alpha_n\rangle = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} \frac{f_n(i,j)}{\sqrt{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} f_n^2(i,j)}}|i\rangle|j\rangle \tag{2.1}$$

where $f(i,j)$ is the pixel value at (i,j) position, N_x and N_y are numbers of pixels of row and column in the image. The idea is to make an entanglement state in each reference image using its featured signals (the regions in which the image can be divided). In fact characters in images are not orthogonal each other. The reference images are given by the following superposition states

$$|r\rangle = \sum_{n=1}^{N} \omega_n \mid \alpha_n\rangle \prod_{m=1}^{M} \mid \alpha_{mn}\rangle \tag{2.2}$$

where α_n is the n-th reference image (equation (2.1)) and $\alpha_{n,m}$ is the m-th featured signal in the n-th reference image. The purpose of the algorithm is to extract the reference image that gives the highest correlation to the input image. In other words, the quantum state to be extracted should have a high probability. The probability of the state to be recognized is increased iteratively by quantum correlation between the featured signals of object and the references. Initially, the pixel value of the object or reference images is coded as phase information or amplitude information in probability amplitude. When the pixel value is given by $x \in [0,1]$, the probability amplitude in phase-encoding and amplitude-encoding are $f(i,j) = exp(i\pi(x-1))$ and $f(i,j) = x$, respectively. Phase-method and amplitude-method are compared.

The dataset used in the paper is composed of roman alphabet images with 128 x 128 pixels as figure 2.1 shows. Featured signals are generated by dividing the images into 8 regions.

Figure 2.2 shows the average probability amplitude of the image as a function of number of correlations of featured signals. One can see that a probability of 1 is obtained after quantum correlation and observation of 8 featured signals. The probability to be in the desired state will be almost 1 by iterative quantum correlation and observation and this proves the effectiveness of the quantum approach.

Figure 2.1: *Samples from the dataset used*

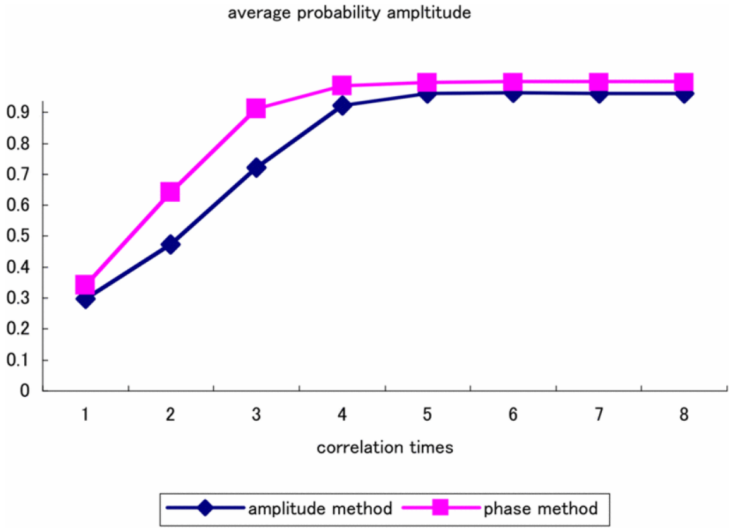


Figure 2.2: *Average probability amplitude vs number of correlation. Figure from the paper [10].*

2.1.3 Recognition of mathematical expressions

An interesting application of deep learning based OCR is the recognition of mathematical expressions, both handwritten and printed. In the physicists community are popular some impressive API, like *Mathpix* and *MyScript*. These tools allow to recognize from a screenshot printed or handwritten formulas and translate them to latex code, being a precious time saver for scientific writing. These systems are closed source and use private datasets for training.

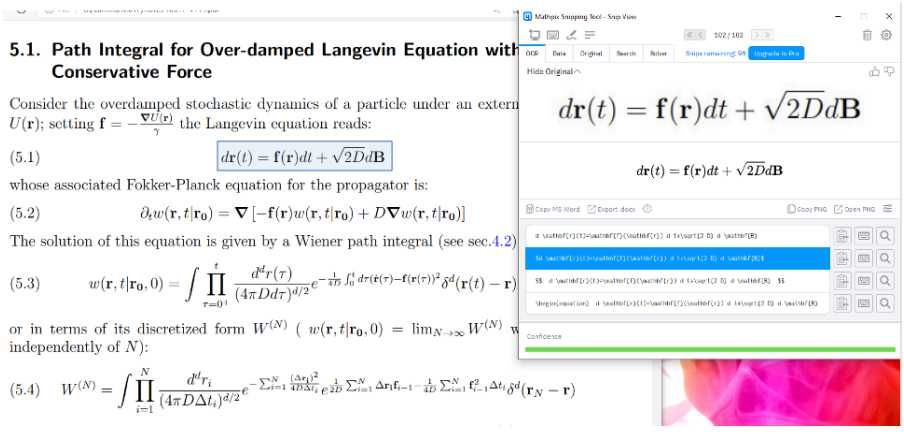


Figure 2.3: *Mathpix example of usage*

The paper [11] presents an approach based on neural network that learns to recognize handwritten mathematical expression (HMEs) in a two-dimensional layout and outputs them as one-dimensional character sequences in LaTeX format. In the work they implement a convolutional neural network encoder that takes HME images as input and exploit a recurrent neural network decoder equipped

with an attention mechanism to generate LaTeX sequences. The proposed approach was trained and validated on benchmarks dataset published by the CROHME (Competition on Recognition of Online Handwritten Mathematical Expression) international competition. The proposed method significantly outperformed the state-of-the-art techiques with an expression recognition accuracy of 46.55% on CROHME 2014 and 44.55% on CROHME 2016. It is worth mentioning the incredible work of Minh Nguyen, 2nd Place of the Japan Demo Day, June 30th, 2021. Following the idea of implementation presented in the aforementioned paper, he develops a system to read, interpret and translate mathematical expressions from images to LaTeX code. Figure 2.4 illustrated the demo he implemented.

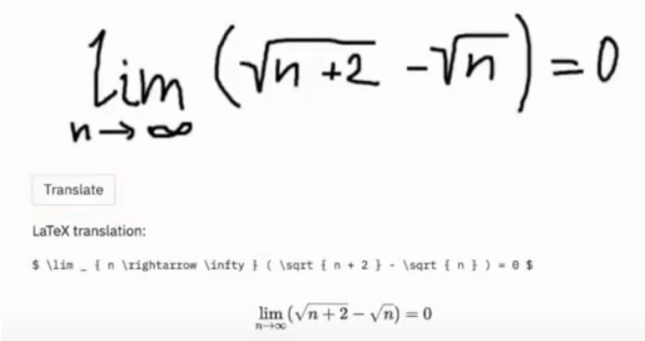


Figure 2.4: *Demo by Minh Nguyen example (`https://www.youtube.com/watch?v=7A-uhg7fwkY`)*

2.2 Applications in the industry

OCR plays a crucial role for many institutions and companies that work with thousand of documents to analyze. This technology represents a business solution for automating information extraction from printed or written text in scanned documents or image files and then converting the text into a machine-readable form, with the aim to store it in databases or search information of interest.

For instance, such technology is exploited in banks to extract costumer's account details. Similarly, passport checking in airports makes use of OCR. Moreover, popular applications aim to retrieve information from receipts, invoices, forms, statements or contracts with the aim of converting handwritten and typed texts to digital format. OCR can be used also to recognize vehicle number plates, with the purpose of vehicle tracking or toll collection.

Nowadays, one of the most fascinating and challenging application of OCR is in self driving car. The aim is to recognize traffic signs, that can be written in different fonts, with different background and curvature and also in different light conditions.

2.2.1 A concrete example: PatchAI - an Alira Health company

PatchAI S.r.l. is a startup founded in 2018 in Padua. The company is aimed at building digital health systems to make clinical research and routine care smart and accessible to patients. Their principle culminates in products like virtual assistants that provide empathetic conversational experience (evidence-based conversations). PatchAi develops also platforms for patient engagement and real-time data collections to make clinical research and standard clinical practice more connected, engaging and humane. To give some numbers that highlight the importance of this kind of efforts, the average deployment time of a new medicine is 15 years, while the cost is around 2 billions of dollars. Clinical research studies present inefficiencies due to delays and abandonment of programs by patients. Is has been estimated that half of the patients involved in a clinical research miss the therapies.
In November 2021 the startup is acquired by *Alira Health*, a global healthcare consulting and technology based in the USA.

This thesis takes its first steps in this context. The OCR pipeline developed in this work is indeed exploited to detect and recognize textual information present on medicine boxes. The idea is to integrate such pipeline into an app support, in such way that the patient can scan the medicine box with the device's camera and the app extracts some basic but important details from that package, like name of that medicine, manufacturing date, expiry date, dosage and target. A possibility is also to integrate information about posology when the app recognizes the corresponding medicine, in order to have a customized solution for every patient, according to the physician's indications. Moreover the use of a vocal assistant that reads orally the recognized information is explored in this work. This can be indeed an useful integration for ederly and blind people. The latter idea is inspired by the work [12]. This paper proposes in fact the use of an OCR model in the identification of medicine boxes for visually impaired people. They use a camera device, available in several popular devices such as computers and smartphones, to identify relevant features on medicine box. After box detection, audio files are played to inform about dosage, indications and contraindications of the medication and also the time indicated in advance by the doctor. Qualitative experiments with blindfolded volunteers indicated an appreciation of the tool.

Figure 2.5: *Medicine box detection using an external camera. Figure from the paper [12].*

Chapter 3

Dataset choice and pre-processing

The dataset we feed a machine learning algorithm with is crucial in determining its performances. The samples in such dataset, i.e. the training dataset, are assumed to be *independently* and *identically* distributed. The latter assumptions are necessary to develop theoretical results [13], but in practice are hard to satisfy, in particular the *independece* constraint. To be as close as possible to the assumption the key is to collect the training set in a smart way, avoiding too correlated samples and too small datasets. Unfortunately there are no theoretical results or rule of thumb that say which is the optimal cardinality of such dataset, being the latter depending essentially on the model. The aim of having a well built dataset is to promote a good generalization performance, once our algorithm is tested. The training dataset has indeed to be a good representation of the real world with respect to our task. For instance, if the goal of our machine learning algorithm is to classify cats and dogs, if our training set is composed only of cats images, the algorithm will be never able to recognize dogs in real world. Despite the latter seems a trivial example, it highlights the importance of the dataset choice and justifies this chapter.

3.1 Datasets available for OCR

There are several datasets publicly available for this task. Since OCR refers to extracting text from every possible image, there are different types of OCR, corresponding to different types of datasets. As already mentioned, this kind of problems spans from receipts information extraction, reading license plates, no-robot captchas to the most challenging natural scene text detection and recognition. Some examples of datasets, open source available and for non-commercial usage, suitable for different tasks, are the followings.

- ***Street View House Numbers dataset (SVHN)***. It is a dataset of house numbers extracted from Google street view. The digits come in various shapes and writing styles and the images can be a little blurred. The dataset is similar to the well known *MNIST*, since it contains 32x32 labeled images of numbers. In particular there are 10 classes available (e.g. digit '1' has label 1, '9' has label 9 and '0' has label 10). SVHN, composed of over 600'000 digit images, can be therefore exploited to train algorithms to detect and classify digits.

Figure 3.1: *SVHN dataset examples*

- ***COCO-Text (2017)***. It contains natural scenes images with text in different positions and orientations, again from street view. It is one of the biggest dataset of this kind, containing 63'686 images with 145'859 text instances.

Figure 3.2: *COCO-Text dataset example*

- ***ICDAR 2015***. It also contains natural scenes images, with texts in English and numbers. The images are blurred and the text is multi-oriented and sometimes not so visible even by a human eye. It contains a total of 1'500 images out of which 1'000 for training and 500 for testing. It also contains 2'077 text instances. The images are obtained from wearable cameras.

Figure 3.3: *ICDAR-2015 dataset example*

- ***ICDAR 2019 - Robust Reading Challenge on Scanned Receipts OCR and Information Extraction (SROIE)***. The dataset contains 1000 scanned receipt images. Each receipt image contains around about four key text fields, such as goods name, unit price and total cost, etc. The text annotated in the dataset mainly consists of digits and English characters. The dataset can be used to train document-intensive processes aimed at office automation in many financial, accounting and taxation areas.

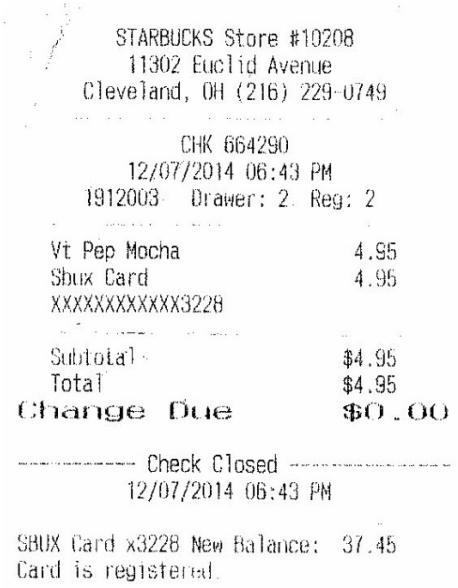


Figure 3.4: *Example of scanned receipt from ICDAR2019-SROIE dataset.*

- ***ICDAR 2019 - Robust Reading Challenge on Multi-Lingual scene Text detection and Recognition***. The dataset is composed of 10'000 images from natural scenes similar to the ones in ICDAR 2015. With respect to the latter, this dataset present less blurred and easier to spot texts. Moreover the texts are in 10 different languages, each one present in a batch of 1000 consecutive images. The languages are Arabic, English, French, Chinese, German, Korean, Japanese, Italian, Bangla and Hindi.

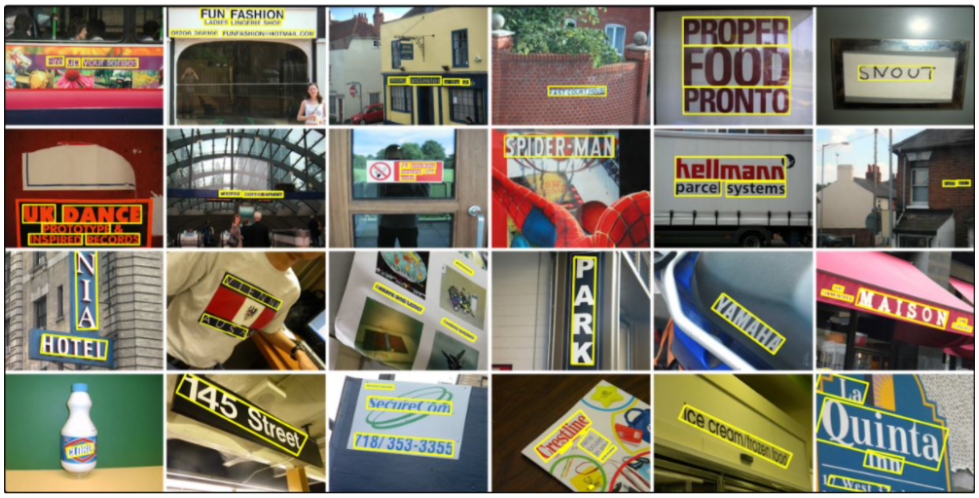


Figure 3.5: *ICDAR-2019 dataset example*

In this work, after an initial test of the pipeline on ICDAR 2015, the training and testing are carried out using images in English and Italian from ICDAR 2019. *ICDAR*, which stands for *International Conference Document Analysis and Recognition* is an international academic conference which is held every two years in a different city, since 1991. It is about character and symbol recognition, both printed and handwritten, document analysis, document based forensics, camera and video based scene text analysis. It is a precious source of open source datasets for OCR tasks, available with ground truth information. In fact for each image a *.txt* file is present, containing the coordinates of the edges of the region where a word or a number is present (the so called *bounding boxes*) and the prediction for that text. In the above images, one can see the bounding boxes plotted around each instance. These ground truths are needed in this work to build the *labels*, used to construct the loss of the model, as explained in the next chapter. Therefore these datasets represent also a benchmark to test if the model is working correctly.


```
441,1548,665,1576,645,1739,407,1744,Latin,La
733,1543,1591,1538,1588,1748,696,1753,Latin,diversità
1663,1548,1768,1548,1775,1748,1649,1744,Latin,è
1863,1600,2087,1586,2110,1748,1859,1744,Latin,un
2148,1562,2748,1557,2782,1748,2229,1763,Latin,valore
```

Figure 3.6: *Example of ground truth information for an image from ICDAR2019. The structure of the file is the same for all the ICDAR datasets. For each word there is the prediction of the text and the corresponding location, annotated as a rectangle, which vertices coordinates are in clockwise order starting from the top. Note that in this dataset also the prediction of the language is present. The text that is not recognized is present in the ground truth file as ###.*

3.2 Exploratory Data Analysis

Before the implementation of the OCR pipeline, composed of detector and recognition, the exploratory data analysis is carried out over the chosen datasets, ICDAR 2015 [14] and ICDAR 2019 [15]. As programming language, *Python* [16] is used and to work with images *OpenCV* [17] is exploited. *OpenCV* is an open-source library for image processing and computer vision tasks, originally written in C++ but with an user friendly wrapper for Python. The version of OpenCV used in this work is 4.1.2.

From ICDARb2019 only 2000 images, the ones in English and Italian, are extracted out of the 10'000. They are further divided in training (1500 images) and test set (500 images), making sure that both the languages are in both sets. Then, thanks to *OpenCV*, the extentions of all the images are trasformed in *.jpg*, since in the original datasets few images are *.png* and *.gif*. It is also checked that all the images have three channels (RGB images). In the ICDAR 2019 datset, it is noted that the heights and widths of the images are randomly distributed, while in the ICDAR 2015 all the images are of the same size, (720, 1280). Before passing them to the detector and building the labels, all the images are resized to be 512 by 512, being careful to resize accordingly also the ground truth i.e. the bounding boxes coordinates. The reason for which all the images have to be 512 by 512 will be clear in the next chapter.

In figure 3.7 there are some examples of the images from the chosen datasets with the bounding boxes around the text instances, plotted with the help of ground truth information. Also the recognized texts are present. Sometimes instead of the word one can see "###", meaning that that word is not recognized.

Figure 3.7: *Images from ICDAR2015 (top) and ICDAR2019 (bottom) with bounding boxes plotted around the text instances and corresponding predictions from ground truth files.*

Chapter 4

OCR pipeline implementation

Most of the OCR tasks, including the one implemented in this work, consist of two sub-tasks:

1. The ***Detection*** part, with the purpose of predicting the coordinates of the area in which the text is present. The edges of this area form the so called *bounding box*.
2. The ***Recognition*** part, with the purpose of predicting texts from cropped images.

In this chapter the aforementioned sub-tasks, together with their implementation, are analyzed in details.

4.1 Detector

As detector *EAST (Efficient Accurate Scene Text Detector)* is implemented. The choice of this fast and accurate scene text detection method is motivated by this paper [18]. The algorithm is indeed quite powerful, since it can find horizontal and rotated bounding boxes and can be combined with any text recognition method. Moreover one can use it to detect text both in images and video.
EAST is a scene text detection method that consists of two parts:

1. A *multichannel fully convolutional network (FCN)*, that takes in input an image and gives in output the so called *geo-map* and *score-map*.
2. A *non-maximum suppression* stage, which takes in input the output of the FCN and generate the final result.

The power of using a deep learning method is that the network learns from the training data which are the important features to discriminate a text from the background.
The implementation is carried out using *Python* [16] and *Keras* [19], an high-level neural network library that runs on top of *TensorFlow* [20], a library for multiple machine learning tasks. The training is performed using a GPU *NVIDIA GeForce RTX 3070*.

	version
Python	3.9.7
Tensorflow	2.7.0

Table 4.1: *Versions of Python and Tensorflow used*

In the following the components of the above pipeline are examined.

4.1.1 The fully convolutional network (FCN)

The structure of the implemented FCN is illustrated in figure 4.1. The picture is taken from the paper [18]. The structure of the network in this implementation differs a bit from the one in the figure, despite it is inspired by this work, as one can see in this section.

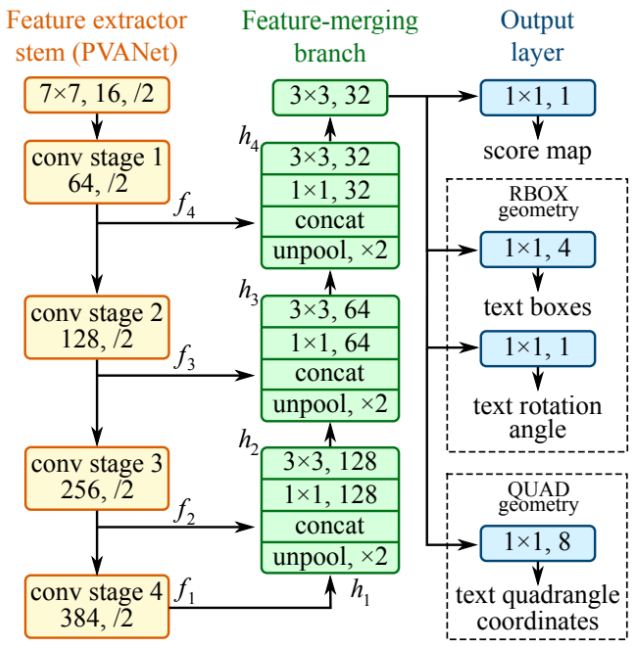


Figure 4.1: *Structure of the text detection FCN. Image from the paper [18]*

As the figure 4.1 illustrates, the model can be divided into three parts: stem feature extraction, feature merging branch and output layer.

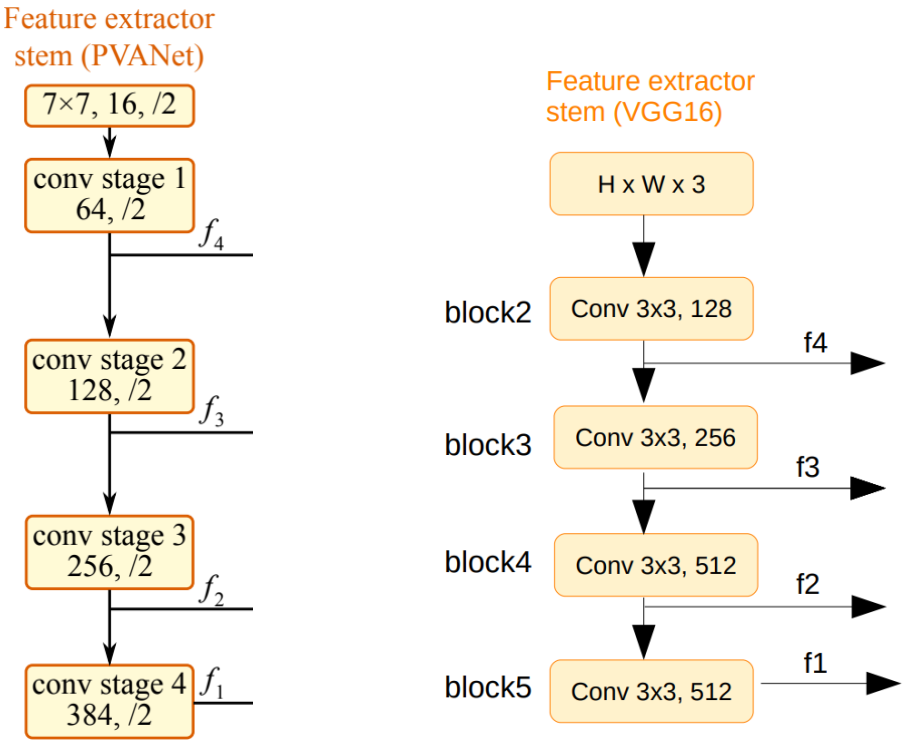


Figure 4.2: *Feature extractor part from the paper (right) vs my implementation (left)*

Feature Extractor

The feature extractor part can be any convolutional neural network: in the implementation of this work, as CNN, VGG-16 is exploited. The VGG model can be loaded and used in the *Keras* deep learning library. The network is pre-trained on ImageNet. In fact using a pre-trained model that is

trained on huge datasets like ImageNet, composed by over 14 million of images, allows us to quickly specialize this architecture to work for our dataset. This process is termed as *transfer learning*. Note that in the paper [18] they used instead PVANet, as figure 4.1 shows. From this network, four levels of feature maps *f1, f2, f3* and *f4* can be obtained. Looking at the figure 4.2, where one can see the structure of the network exploited in this work, H and W (height and width of the input image) are fixed to 512. The reasons behind this lie in the *transfer learning* process, since the model is pre-trained on fixed input image sizes, that are $512 \times 512 \times 3$. This justifies the resizing that was done in the exploratory data analysis. Despite it is not highlighted in the figure, after every convolutional block there is a *maxpooling* layer with the purpose of downsampling the image. A more accurate summary for this part is given by *Keras* and pasted in figure 4.3.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 512, 512, 3)	0	
block1_conv1 (Conv2D)	(None, 512, 512, 64)	1792	input_1[0][0]
block1_conv2 (Conv2D)	(None, 512, 512, 64)	36928	block1_conv1[0][0]
block1_pool (MaxPooling2D)	(None, 256, 256, 64)	0	block1_conv2[0][0]
block2_conv1 (Conv2D)	(None, 256, 256, 128	73856	block1_pool[0][0]
block2_conv2 (Conv2D)	(None, 256, 256, 128	147584	block2_conv1[0][0]
block2_pool (MaxPooling2D)	(None, 128, 128, 128	0	block2_conv2[0][0]
block3_conv1 (Conv2D)	(None, 128, 128, 256	295168	block2_pool[0][0]
block3_conv2 (Conv2D)	(None, 128, 128, 256	590080	block3_conv1[0][0]
block3_conv3 (Conv2D)	(None, 128, 128, 256	590080	block3_conv2[0][0]
block3_pool (MaxPooling2D)	(None, 64, 64, 256)	0	block3_conv3[0][0]
block4_conv1 (Conv2D)	(None, 64, 64, 512)	1180160	block3_pool[0][0]
block4_conv2 (Conv2D)	(None, 64, 64, 512)	2359808	block4_conv1[0][0]
block4_conv3 (Conv2D)	(None, 64, 64, 512)	2359808	block4_conv2[0][0]
block4_pool (MaxPooling2D)	(None, 32, 32, 512)	0	block4_conv3[0][0]
block5_conv1 (Conv2D)	(None, 32, 32, 512)	2359808	block4_pool[0][0]
block5_conv2 (Conv2D)	(None, 32, 32, 512)	2359808	block5_conv1[0][0]
block5_conv3 (Conv2D)	(None, 32, 32, 512)	2359808	block5_conv2[0][0]
block5_pool (MaxPooling2D)	(None, 16, 16, 512)	0	block5_conv3[0][0]

Figure 4.3: *VGG-16 summary given by Keras*

Feature Merging Branch

Looking at the lowest green box in figure 4.1, the feature maps obtained from the feature extractor (*f1=h1*) are first fed to the unpooling layer to double their size, and then concatenated with the current feature map (*f2*). Next, the 1X1 convolution is used to reduce the number of channels and the amount of calculation, followed by a 3X3 convolution to fuse information to produce the final output of every merging stage. The structure is the same for all the merging states (green boxes).
If g_i is the intermediate state before merging one can write

$$g_i = \begin{cases} unpool(h_i) & if \quad i \leq 3 \\ conv_{3x3}(h_i) & if \quad i = 4 \end{cases} \tag{4.1}$$

and the merged feature map h_i is

$$h_i = \begin{cases} f_i & if \quad i = 1 \\ conv_{3x3}(conv_{1x1}[g_{i-1}; f_i]) & otherwise \end{cases} \tag{4.2}$$

Output Layer

The final output from the merged state (highest green box in figure 4.1) is passed through 1X1 convolutional layer with 1 channel which gives a *score map* in the range [0,1]. The final output is also passed through *RBOX* or *QUAD Geometry* which gives a multi-channel geometry map.

Geometry	channels
RBOX: $\mathbf{G} = \{\mathbf{R},\theta\}$ with $\mathbf{R} = \{d_i \mid i \in [1,4]\}$	5
QUAD: $\mathbf{G} = \{(\Delta x_i, \Delta y_i) \mid i \in [1,4]\}$	4

Table 4.2: *Description of RBOX and QUAD Geometry*

In the table above there are two types of *geo map*. For RBOX one has 5 channels, the first 4 are the distances from the pixel position to the rectangular boundaries (vector **R** in the table) and the last channels is the rotation angle, as shown in the figure 4.4.

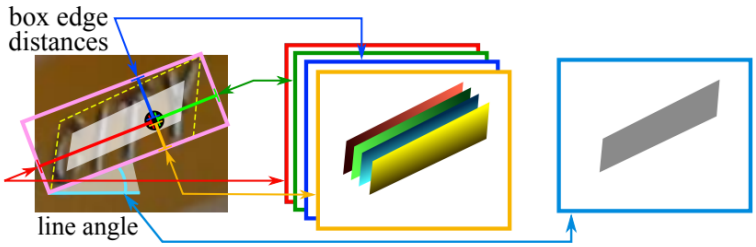


Figure 4.4: *Rbox geo map. Figure from the paper [18]*

Regarding QUAD geo map, there are 8 numbers to represent the displacement from four vertices to each pixel location. Each offset distance contains two numbers (Δx_i and Δy_i) and the geometric output contains 8 channels. In this implementation only RBOX is used.
The score map represents the confidence score/level for the predicted geometry map at that location/pixel. It lies in the range [0,1]. In other words, each pixel has its own score map. For instance suppose that 0.80 is the score map of a pixel. This means that for this pixel we are 80% confident that it will have the predicted geometry map i.e it is part of predicted text region.

4.1.2 Label generation

In order to build the loss of the FCN the idea is to exploit the ground truths i.e. the coordinates of the bounding boxes that are associated to each image in the ICDAR datasets. In particular, since the output of the FCN is a score map and a geo map for each text instance, the scope is to generate *artificially* the two maps starting from the ground truth coordinates. As already state, only RBOX geometry is taken into account.

Score map generation

Only the case in which the geometry around the instance is a quadrangle is considered. Let it be $\mathbf{Q} = \{p_i | i \in \{1, 2, 3, 4\}\}$ a quadrangle, where $p_i = \{x_i, y_i\}$ are vertices on the quadrangle in clockwise order. To generate the score map, the idea is to shrink $\mathbf{Q}$ by moving its vertices inwards by some reference length $0.3 \cdot r_i$. The reference length r_i for each vertex p_i can be computed as

$$r_i = \min\left(\mathrm{D}\left(p_i, p_{(i \bmod 4)+1}\right), \mathrm{D}\left(p_i, p_{((i+2) \bmod 4)+1}\right)\right) \tag{4.3}$$

where $D(p_i, p_j)$ is the L2 distance between p_i and p_j.

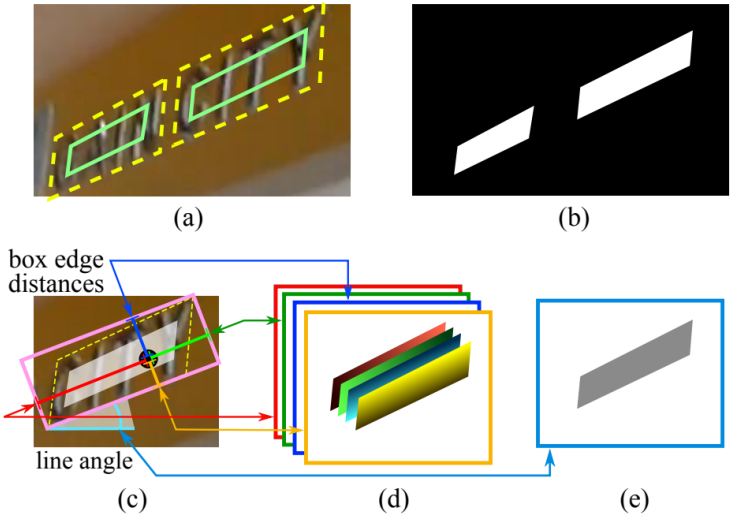


Figure 4.5: *Label generation process. (a) The original quadrangle is the yellow dashed one while the shrunk one is represented by the solid green line. (b) The score map is represented by the white area, which is the area inside the green shape. (c) RBOX geometry generation (d) 4 channels of the RBOX geometry, representing the distances of each pixel to rectangle boundaries. (e) 5th channel of the RBOX geometry, representing the rotation angle, calculated from the horizontal direction to the rectangle. Figure from the paper [18]*

Geo map generation

The idea is to generate RBOX geometry, described in table 4.2. Starting from the coordinates in the ICDAR dataset (vertices of the yellow dashed quadrangle in figure 4.5 (c)) a rectangle that covers the region with minimal area is generated (pink rectangle in figure 4.5 (c)). Then for each pixel in the positive score map (generated through the shrinking procedure), its distances from the 4 pink boundaries of the text box are calculated, forming the 4 channels of RBOX geometry. The 5th channel is formed by the rotation angle.

4.1.3 Loss function

The loss to evaluate the performance of the FCN can be formulated as

$$L = L_s + \lambda_g L_g \tag{4.4}$$

where L_s and L_g are the loss of the score map and of the geo map respectively and λ_g represents the trade off between the two. Both in the paper and in this implementation λ_g is set to 1.

Score map loss

As loss for the score map the Dice coefficient loss function is implemented. Dice loss originates from Sørensen–Dice coefficient, which is a statistic developed in 1940s to gauge the similarity between two

samples. It was brought to computer vision community by Milletari et al. in 2016 for 3D medical image segmentation. The equation for the loss is

$$L_s = 1 - \frac{2\sum_{i=1}^{N} \hat{\mathbf{Y}}_i \mathbf{Y}_i^*}{\sum_{i=1}^{N} \hat{\mathbf{Y}}_i + \sum_{i=1}^{N} \mathbf{Y}_i^*} \tag{4.5}$$

where $\hat{\mathbf{Y}}_i$ and $\mathbf{Y}_i^*$ represent the prediction and the ground truth values on the score map. N is the total number of the elements in the score map. In the EAST paper [18] they used instead binary cross-entropy loss as loss for the score map.

Geo map loss

The RBOX geometry is composed of 5 channels: 4 distances and 1 angle (table 4.2). Accordingly the loss of the geo map is composed of two *weighted* parts:

$$L_g = L_d + \lambda_\theta L_\theta \tag{4.6}$$

where the weight λ_θ is set to 50 in our implementation.
The loss for the distances (first 4 channels of RBOX) is the IoU (Intersection over Union) loss defined as follows

$$L_d = -log[IoU(\hat{\mathbf{R}}, \mathbf{R}^*)] = -log\frac{|\hat{\mathbf{R}} \cap \mathbf{R}^*|}{|\hat{\mathbf{R}} \cup \mathbf{R}^*|} \tag{4.7}$$

where $\hat{\mathbf{R}}$ is the network prediction and $\mathbf{R}^*$ is composed from the ground truth, as described above. Recall that $\mathbf{R}$ represents the vector of distances from the pixel to the four edges of the rectangular box $\mathbf{R} = \{d_i | i \in [1, 4]\}$, where d_1, d_2, d_3 and d_4 represent the distance from a pixel to the top, right, bottom and left boundary of its corresponding rectangle, respectively.

If one consider the intersection $|\hat{\mathbf{R}} \cap \mathbf{R}^*|$, the width and height of the intersected rectangle are

$$\begin{aligned} w_{\mathbf{i}} &= \min\left(\hat{d}_2, d_2^*\right) + \min\left(\hat{d}_4, d_4^*\right) \\ h_{\mathbf{i}} &= \min\left(\hat{d}_1, d_1^*\right) + \min\left(\hat{d}_3, d_3^*\right) \end{aligned} \tag{4.8}$$

Regarding the union one has

$$|\hat{\mathbf{R}} \cup \mathbf{R}^*| = |\hat{\mathbf{R}}| + |\mathbf{R}^*| - |\hat{\mathbf{R}} \cap \mathbf{R}^*| \tag{4.9}$$

Therefore the area of both rectangles, intersection and union, can be computed. It follows that the implementation of L_d, with the above reasoning, is quite straightforward.
Moreover the loss of the angle L_θ can be computed as

$$L_\theta(\hat{\theta}, \theta^*) = 1 - cos(\hat{\theta} - \theta^*) \tag{4.10}$$

where $\hat{\theta}$ is the prediction, while θ^* is the ground truth of the rotation angle.

4.1.4 Training

Once implemented the *labels* i.e. the ground truths of the above defined loss, the latter can be used to measure the performance of the FCN during training. The network is trained using *Adam* as optimizer, with *AMSgrad* option enabled. AMSgrad is a recent proposed improvement to Adam. It computes the current gradient v_t, but then setting $v_t = max(v_t, v_{t-1})$. In this way the influence of large, more informative, gradients is retained, leading to a more satisfactory performance on image recognition tasks. However there is still a lack of verifiability on generic data sets, therefore the use of AMSgrad, as other machine learning techniques, is still empirical based. The training images are divided in minibatches of size 10. The initial learning rate is $1 \cdot 10^{-3}$ and decays of a factor of 0.90 after a patience (number of epochs with no improvement) of 2. Also the so called *early stopping* is implemented: if the loss does not improve after a patience of 5 epochs the learning is stopped.

4.1.5 Non-Maximum Suppression

After training, the geometry maps are converted back to the bounding boxes. In order to remove some low confidence boxes, the idea is to apply thresholding basing on the score map. The threshold is fixed to 0.7. Therefore the geometries corresponding to pixels with a score map lower than 0.7 are thrown away. Then, the remaining boxes are merged using Non-Maximum Suppression.
Non Maximum Suppression (NMS) is a technique used in many computer vision algorithms. It refers to a class of algorithms which scope is to select one single bounding box out of many overlapping ones.

The algorithm used in the implementation, inspired by the EAST paper [18], is *Locally Aware NMS*. The latter works as follows:

1. Sort the bounding boxes for a single instance and start from the topmost
2. Take the next bounding box in the row and find IoU with the area of the previous
3. If the IoU $>$ threshold (fixed to 0.2 in the implementation) merge the 2 boxes by taking the weighted average (where the weight is given by the score) otherwise keep the box as it is
4. Iterate over all boxes

4.2 Recognizer

In this second part the task is to crop out the image inside the bounding box and apply text recognition on it. As recognizer *Tesseract* [21] is exploited through the *pytesseract* python library. The version of the latter is 0.3.8. *Pytesseract* makes use of Recurrent Neural Networks, in particular LSTM (Long Short Term Memory) networks, introduced from a theoretically point of view in the first chapter.

Tesseract is an open-source OCR engine, that began as a Ph.D. research project in Hewlett-Packard Labs, Bristol. It gained popularity and was developed by Hewlett-Packard between 1984 and 1994. In 2005 HP released Tesseract as an open-source software. Since 2006 it is developed by Google. It can be used in conjunction with an external text detector to recognize text from an image of a single text line, as it is indeed exploited in this work. The tool is a C++ implementation of a LSTM recurrent neural network and support more than 100 languages.

In this work the languages used are english and italian and before giving the cropped texts to *pytesseract*, the images are preprocessed. With the help of *OpenCV* the images are firstly converted to gray scale images and then morphological transformations are applied.
Morphological transformations are simple operations based on the image shape. They are normally performed on binary images. This kind of transformation needs two inputs, one is the gray scale image and the second one is called *structuring element* or *kernel* which decides the nature of operation. The aforementioned structuring element is created automatically by *OpenCV*. It is chosen a rectangular kernel, which is a square matrix filled with ones. There exist two basic morphological operators: Erosion and Dilation. The second one is performed. The kernel slides through the image (as in 2D convolution). The pixels are divided by 255 making sure the image is normalized. A pixel in the original image is either 1 or 0 and will be 1 if at least one pixel under the kernel is 1. Therefore the transformation increases the white region (the text) in the image or the size of foreground object increases. The procedure is visually described in figure 4.6.

Figure 4.6: *Left: original image (gray scale). Right: image after dilatation.*

4.3 Results

In this section the results of the training of the detector described above are shown. The training is performed using an *NVIDIA GeForce RTX 3070* GPU, using 1000 images from ICDAR 2015 and 1500 images from ICDAR 2019. The weights of the detector trained on ICDAR 2015 are then used to train the algorithm on the other dataset. The reasons of this transfer learning are the followings:

1. The transfer learning helps in reducing overfitting, increasing the cardinality of training dataset
2. The images of ICDAR 2019 contains much less non recognixed text (marked with ###), therefore is more meaningful to adopt this dataset as benchmark to test the *Tesseract* recognizer
3. The images on ICDAR 2015 are more blurred, since they are taken with wearable cameras, and this is a good point to make the detector robust to this feature

The results of the detector model trained on ICDAR 2015 are shown in figure 4.7. The loss on the test set, composed of 500 images, is evaluated to 0.97 after the 30th epoch.

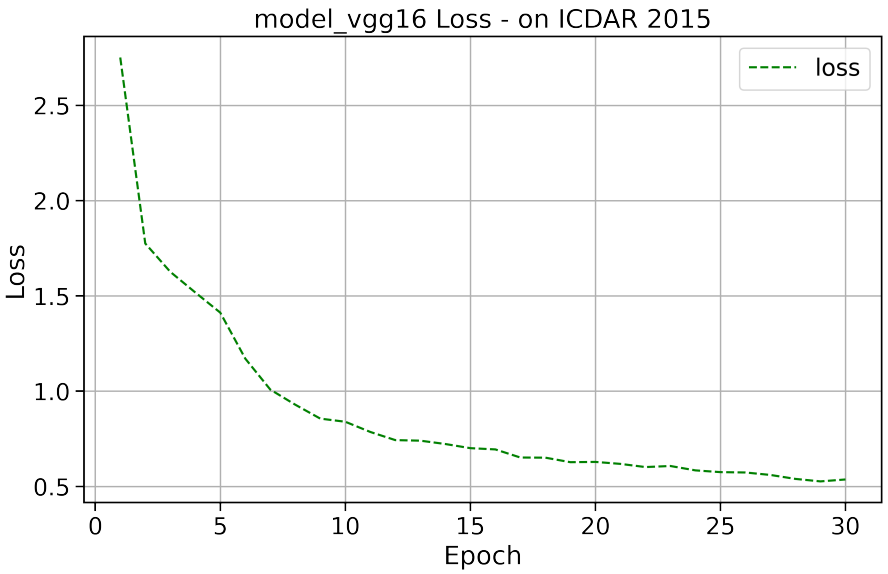


Figure 4.7: *Loss of the vgg16 model (detector) on ICDAR 2015 training set, composed of 1000 images.*

The same model is trained on ICDAR 2019, exploiting the weights trained on ICDAR 2015. Both the training set (1500) images and the test set (500) images on ICDAR 2019 contain images in english and italian. From now on the analysis on performed using such dataset.

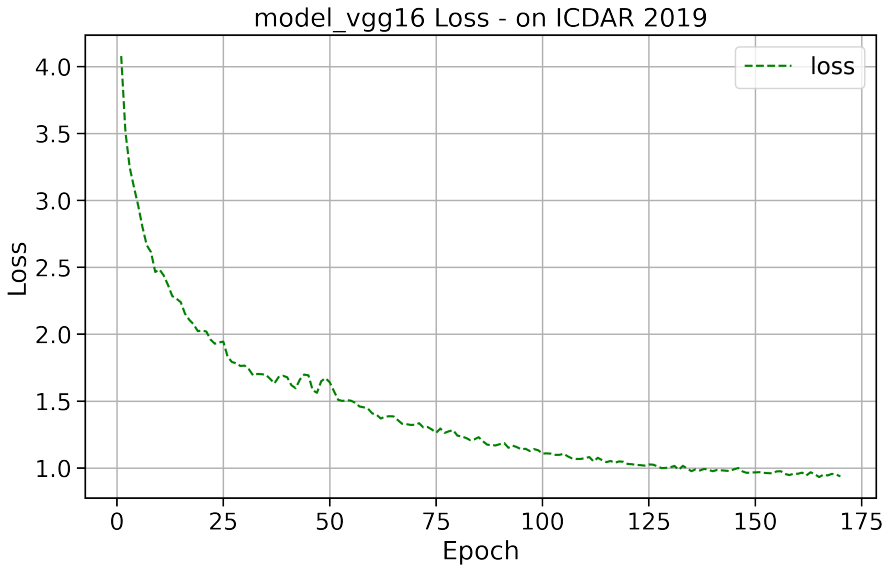


Figure 4.8: *Loss of the vgg16 model (detector) on ICDAR 2019 training set, composed of 1500 images.*

An exploratory analysis of trained model generalization performances follows.

Firstly, the loss for each image in the training and test set has been calculated. The statistics of the two vectors of losses are shown in table 4.3.

	mean	***std***	***min***	***max***
Training losses	1.03	1.06	0.15	18
Test losses	1.9	2.6	0.3	21

Table 4.3: *Losses statistics*

The density plot and the boxplots of the losses are shown in figure 4.9, to visualize their distribution. The boxplots show that the IQR (interquartile range), between 25th percentile and 75th percentile, is more huge for the test losses, as the histogram confirms, being more picked in fewer bins.

Therefore two threshold losses are selected and basing on this selection, the data are categorized into three categories, Good, Average and Bad. The two threshold losses are the 30th and 70th percentile of the training losses distribution. If a loss is below the 30th percentile falls into the Good category, if it is between the 30th and 70th percentile falls into the Average category, otherwise it belongs to the Bad category. The results of the aforementioned categorization are shown in figure 4.10.

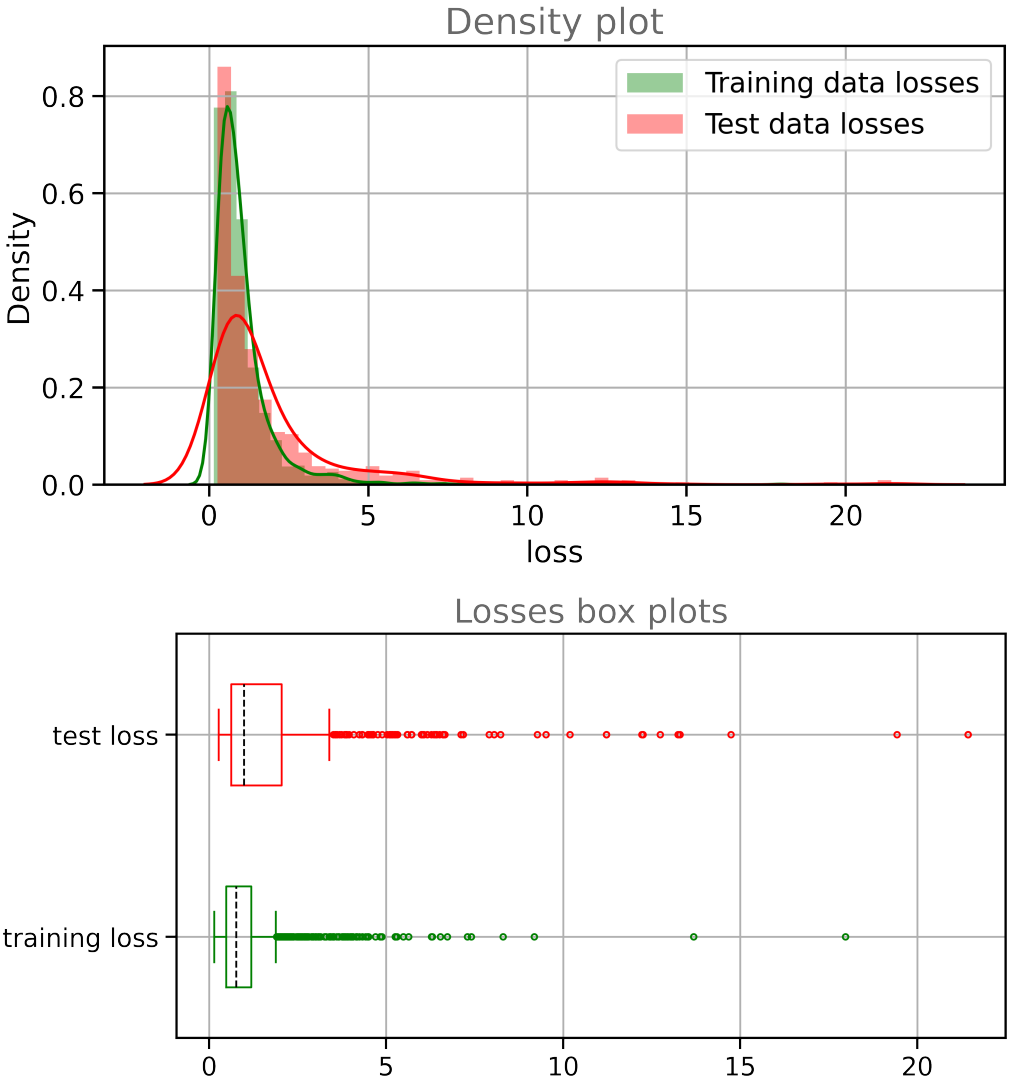


Figure 4.9: *Top: density plot of the losses. Bottom: Box plots of the losses. The dashed black line represent the median of the data.*

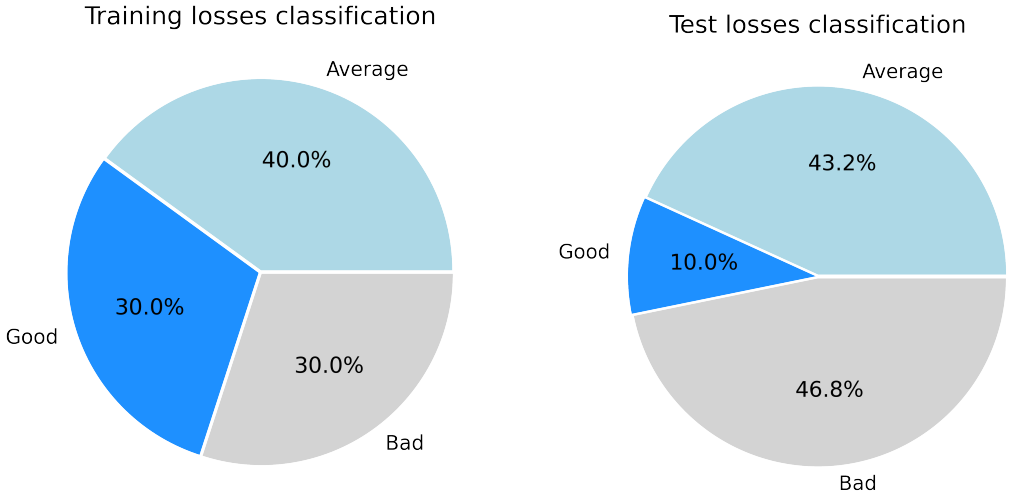


Figure 4.10: *Left: Training losses categorization. Right: Test losses categorization.*

After the training the bounding boxes are sent to thresholding and Locally Aware NMS as described in section 4.1.5.

The results of the predictions on some test images after the aforementioned steps are shown in figure 4.11.

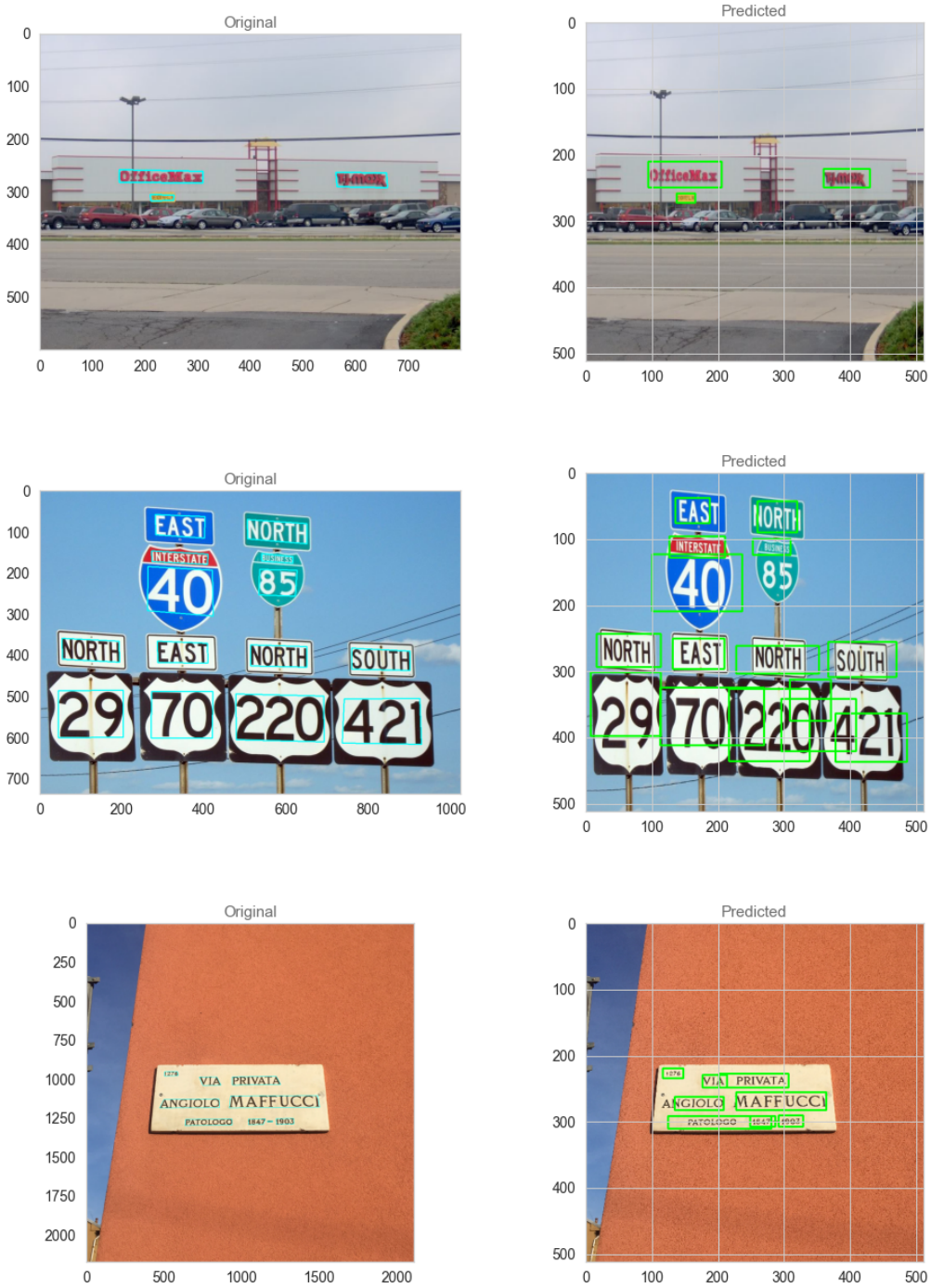


Figure 4.11: *Some predictions on test images after thersholding and NMS. On the left there are images with bounding boxes drawn with the help of ground truth information. On the right there are the predictions of the detector algorithm.*

Figure 4.12 shows also the time taken by the detector to process the test images.

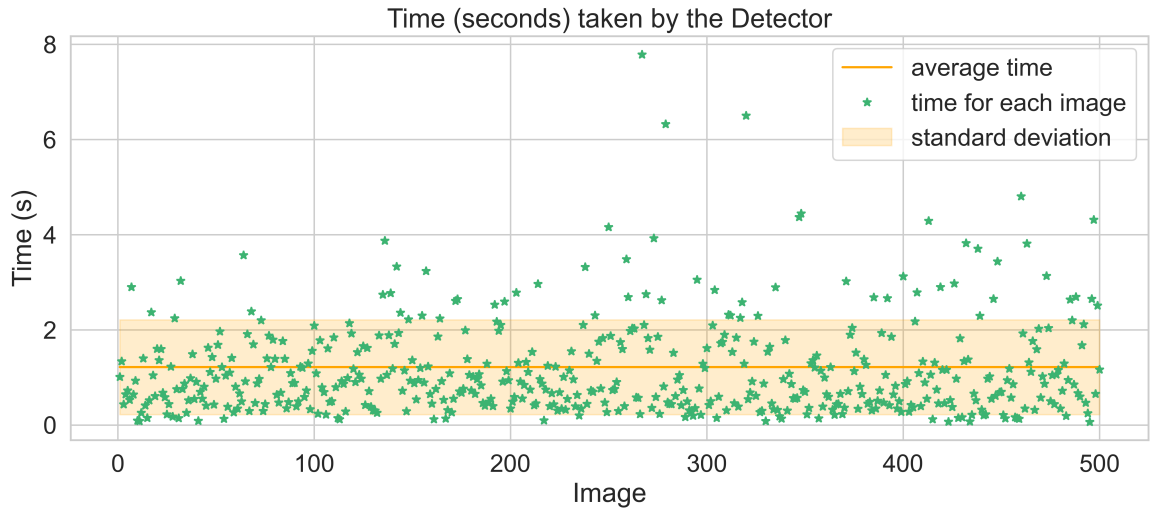


Figure 4.12: *Time taken by the detector on the 500 test images.*

To assess the performance of the recognizer the texts predictions present in the ICDAR 2019 dataset become handy. In fact for each test image the pipeline checks how many predicted text instances correspond to the ICDAR ones. Before doing the comparison, it is inspected which is the ratio of the presence of unrecognized text, marked with ###, in the ground truth files of the test set. Such ratio is defined as

$$ratio = \frac{number\ of\ ###\ in\ a\ image}{total\ texts\ instances\ in\ a\ image} \tag{4.11}$$

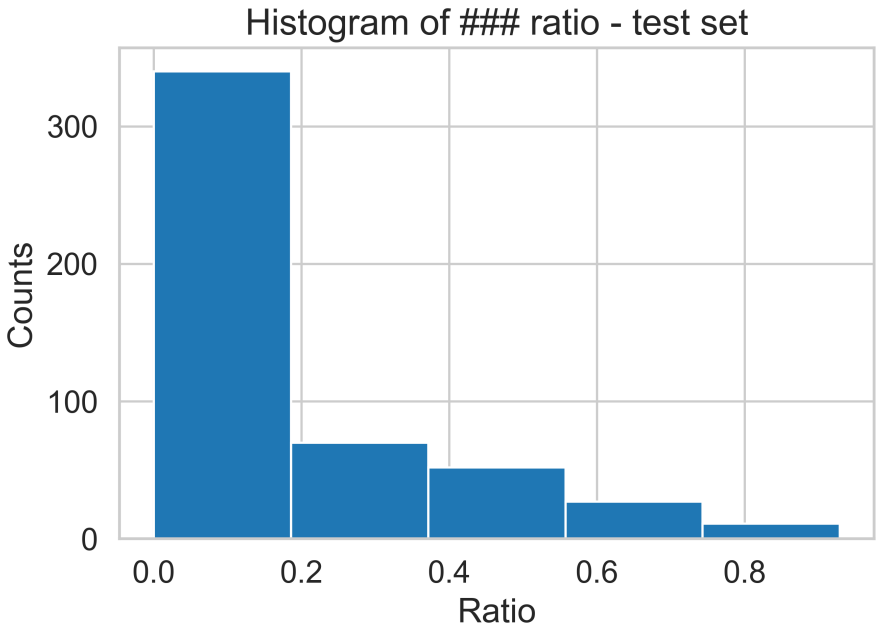


Figure 4.13: *Histogram of the ratio defined above on test set images.*

The unrecognized texts are then removed, to avoid to false the comparison results.
Four categories of images have been defined: Perfect, Good, Bad and Total bad.
An image falls in the Perfect category if all the recognized instances correspond to the ICDAR ones. It falls in the Good category if there is a match for over 50% of the instances. If the recognized instances are between 50% and 30% the image falls into the Bad category and otherwise into the Total bad one.

Figure 4.14 shows the results of this comparison on the test set, equally divided in english and italian images.

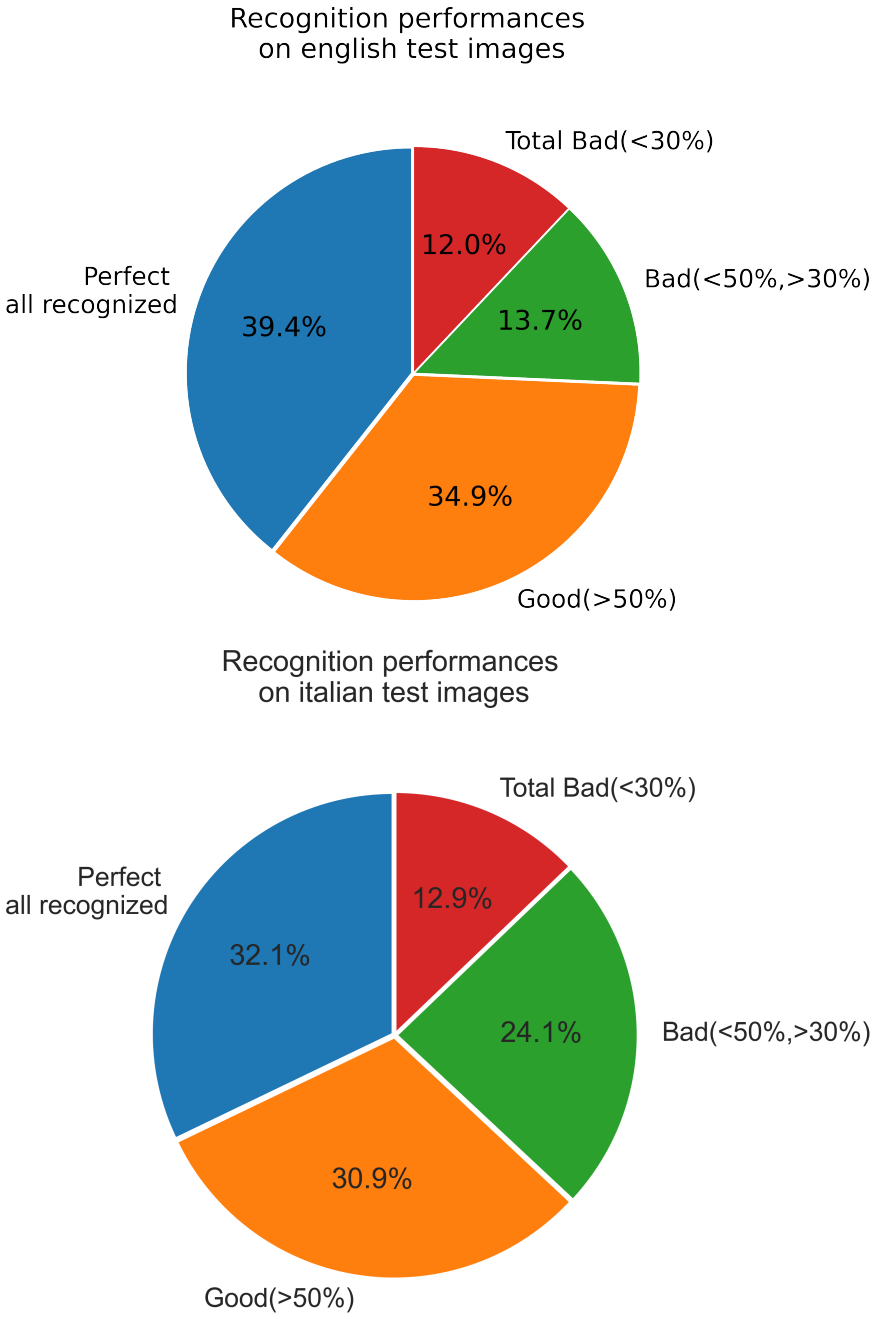


Figure 4.14: *Top: English test images comparison results. Bottom: Italian test images comparison results.*

4.4 Performance on medicine boxes

The OCR pipeline implemented so far and described above in details is tested on medicine boxes, being the latter the application in the internship context. The performance reached by the detector on two example images is shown in figure 4.15. Note that one medicine box is in english and the other one is in italian, since the target of the product is interested in these two languages.

Figure 4.15: *Examples of the detector performance on medicine boxes*

The images inside the bounding boxes are cropped and pre-processed as described in section 4.2. Then exploiting *pytesseract* the texts are recognized. The results are in figures 4.17 and 4.16.
Through the use of *gTTS* (Google Text-To-Speech) python library the recognized texts are read orally, after having specified the language, and saved into audio files.

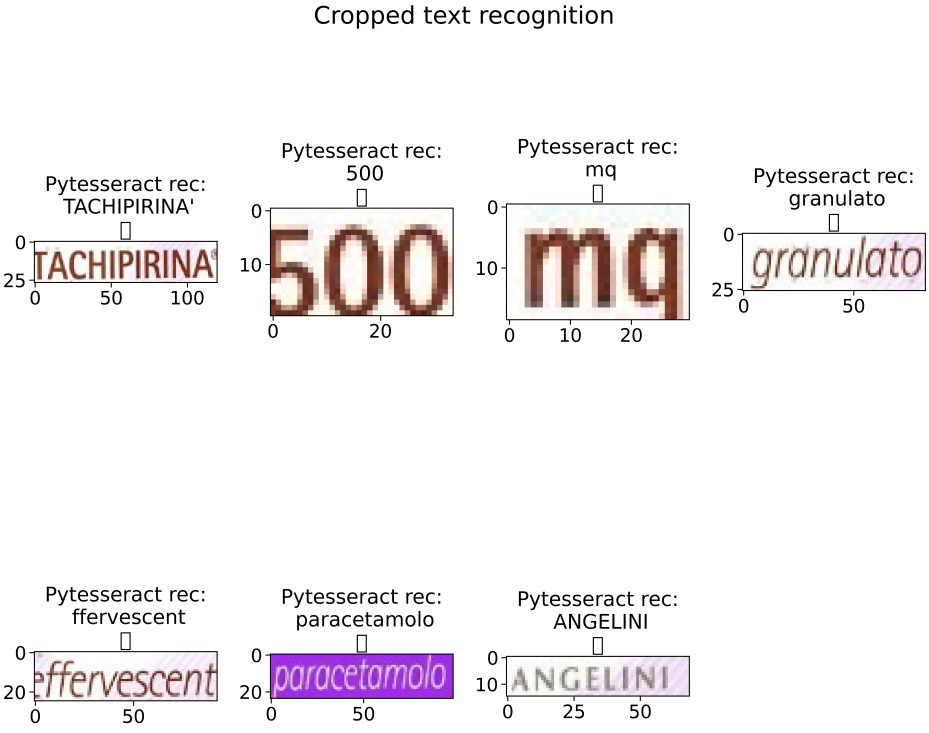


Figure 4.16: *Pytesseract text recognition*

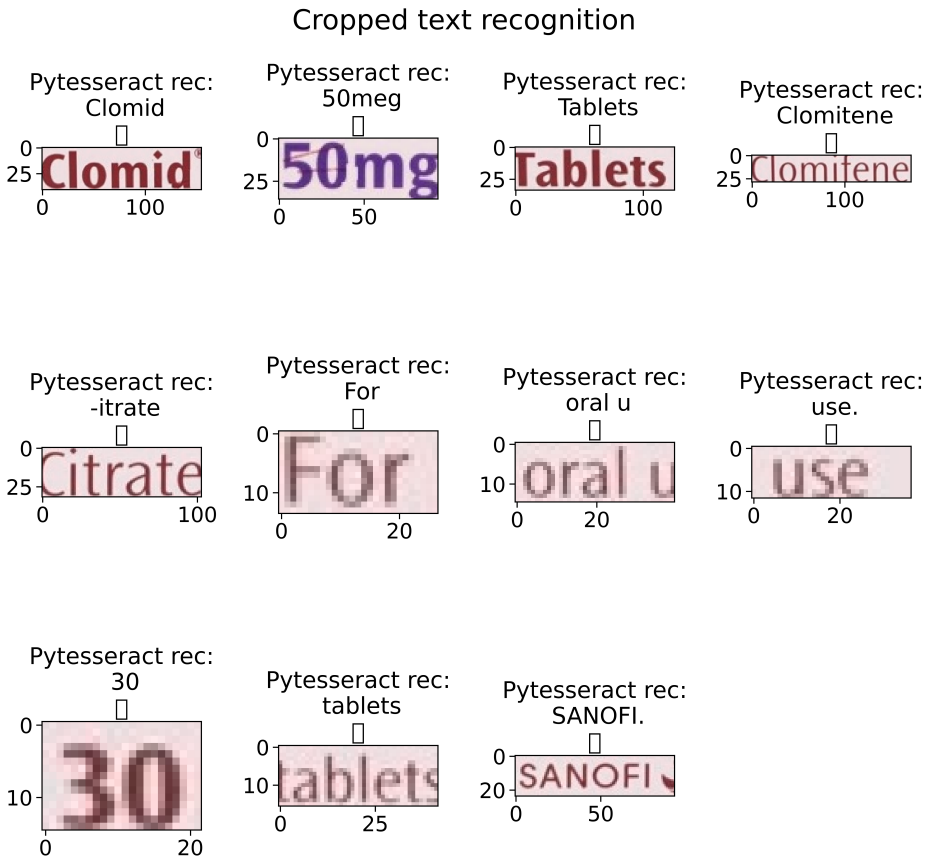


Figure 4.17: *Pytesseract text recognition*

Conclusions and future perspectives

This thesis work is an overview of what is deep learning based Optical Character Recognition and what are the areas in which the popularity of these tools are growing. Despite the old roots, OCR is far from being a solved problem, being still challenging for instance in the task of detecting and recognizing text in natural images. Nowadays extracting and understanding textual information embodied in natural scenes have become increasingly important and popular, as confirmed by the large numbers of participants of the ICDAR series contests.

The work concentrates on the implementation of the EAST algorithm to detect text in scene images. Also, the attention is payed on label construction and on the design of the loss function to evaluate the model. The pipeline is capable of predicting texts instances of arbitrary shapes and orientations through a single neural network. The latter exploits VGG-16 network as features extractor, but one could experiment also with other networks, as ResNet-50 or PVANet, used in the work [18]. The performance of the detector is far from being perfect, but it is satisfactory considering the relatively small dataset used for training. The training of the detector is very computationally demanding and this has held back from trying different network architectures or validation setups, even if different hyperparameters are tried and tuned by hand. A future direction is to train the model with a larger dataset. Also, the algorithm might miss vertical text instances as they are rare in the training set used. However one should consider that the model, once deployed into an app support, can keep on training as long as it is used. In fact the pictures taken by users can become training samples, with more close features to the ones needed for the scope e.g. vertical texts or blurred image. Regarding the specific application on medicine boxes, an interesting possibility is to integrate a vocal assistant, as already explored. In this way it could be an useful device also for ederly or visually impaired peoples. With a view to putting this pipeline on an app for patients, it could be helpful to integrate the recognition of the medicine with information about its posology. The idea is that a patient scans the medicine and receives information e.g. about the appropriate dosage according to the physician's indications. Also an instant translator that can individuate the active ingredient of a drug in a language and interpret it into another language could be extremely useful.

Despite the specific application on medicine boxes, it is important to highlight the generality of the implemented pipeline, that can be adapted to different contexts.

Bibliography

[1] H. F. Schantz, *History of OCR, Optical Character Recognition*. Recognition Technologies Users Association, 1982.

[2] M. Namysl and I. Konya, "Efficient, lexicon-free ocr using deep learning," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 295–301, 2019.

[3] S. Yousfi, S.-A. Berrani, and C. Garcia, "Deep learning and recurrent connectionist-based approaches for arabic text recognition in videos," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1026–1030, 2015.

[4] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, pp. 179–211, 1990.

[5] c. blog, "Understanding lstm networks," 2015.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] E. Huerta, A. Khan, X. Huang, M. Tian, M. Levental, R. Chard, W. Wei, M. Heflin, D. S. Katz, V. Kindratenko, *et al.*, "Accelerated, scalable and reproducible ai-driven gravitational wave detection," *Nature Astronomy*, vol. 5, no. 10, pp. 1062–1068, 2021.

[8] A. Chattopadhyay, E. Nabizadeh, and P. Hassanzadeh, "Analog forecasting of extreme-causing weather patterns using deep learning," *Journal of Advances in Modeling Earth Systems*, vol. 12, Feb 2020.

[9] H.-J. Yang, B. P. Roe, and J. Zhu, "Studies of boosted decision trees for miniboone particle identification," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 555, p. 370–385, Dec 2005.

[10] O. Matoba, T. Murata, K. Nitta, and T. Yoshimura, "Recognition property of quantum character recognition algorithm," in *LEOS 2006 - 19th Annual Meeting of the IEEE Lasers and Electro-Optics Society*, pp. 350–351, 2006.

[11] J. shu Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J.-S. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognit.*, vol. 71, pp. 196–206, 2017.

[12] X. Cavalcante Benjamim, R. Gomes, A. Burlamaqui, and L. Gonçalves, "Visual identification of medicine boxes using features matching," pp. 43–47, 07 2012.

[13] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014.

[14] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. K. Ghosh, A. D. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny, "Icdar 2015 competition on robust reading.," in *ICDAR*, pp. 1156–1160, IEEE Computer Society, 2015. relocated from Tunis, Tunisia.

[15] N. Nayef, Y. Patel, M. Busta, P. N. Chowdhury, D. Karatzas, W. Khlif, J. Matas, U. Pal, J.-C. Burie, C.-l. Liu, and J.-M. Ogier, "Icdar2019 robust reading challenge on multi-lingual scene

text detection and recognition — rrc-mlt-2019," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1582–1587, 2019.

[16] Python Core Team, *Python: A dynamic, open source programming language*. Python Software Foundation, 2019.

[17] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[18] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "EAST: an efficient and accurate scene text detector," *CoRR*, vol. abs/1704.03155, 2017.

[19] F. Chollet *et al.*, "Keras," 2015.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[21] A. Kay, "Tesseract: An open-source optical character recognition engine," *Linux J.*, vol. 2007, p. 2, jul 2007.