

Thales Levi Azevedo Valente

**Method for Automatic Detection of Stamps in Scanned
Documents Using Deep Learning and Synthetic Data
Generation by Instance Augmentation**

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em
Informática of PUC-Rio in partial fulfillment of the requirements
for the degree of Doutor em Ciências - Informática

Advisor: Prof. Marcelo Gattass

Co-advisor: Dr. Paulo Ivson Netto Santos



Thales Levi Azevedo Valente

Method for Automatic Detection of Stamps in Scanned Documents Using Deep Learning and Synthetic Data Generation by Instance Augmentation

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the Examination Committee:

Prof. Marcelo Gattass

Advisor

Departamento de Informática – PUC-Rio

Dr. Paulo Ivson Netto Santos

Co-Advisor

Tecgraf – PUC-Rio

Prof. Karla Tereza Figueiredo Leite

UERJ

Prof. Geraldo Braz Junior

UFMA

Prof. Alberto Barbosa Raposo

Departamento de Informática – PUC-Rio

Prof. Waldemar Celes Filho

Departamento de Informática – PUC-Rio

All rights reserved.

Thales Levi Azevedo Valente

Graduated in Computer Science from Universidade Federal do Maranhão – UFMA in 2015, he obtained the degree of Mestre at Universidade Federal do Maranhão – UFMA in Electrical Engineering in 2017

Bibliographic data

Valente, Thales Levi Azevedo

Method for automatic detection of stamps in scanned documents using deep learning and synthetic data generation by instance augmentation / Thales Levi Azevedo Valente ; advisor: Marcelo Gattass ; co-advisor: Paulo Ivson Netto Santos. – 2022.

101 f. : il. color. ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2022.

Inclui bibliografia

1. Informática – Teses. 2. Detecção de carimbos. 3. Aprendizagem profunda. 4. Faster R-CNN. 5. Documentos digitalizados. 6. Aumento de instâncias. I. Gattass, Marcelo. II. Santos, Paulo Ivson Netto. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD:004

Acknowledgments

My Family, for all the support you have given to me. They provided all the necessary structures to move forward, even during diversities. In special, to my parents Aquiles and Leonilde Valente, and to my brothers Thalita and Thiago Valente.

I want to thank my advisor, Prof. Marcelo Gattass, and co-advisor, Prof. Paulo Ivson, for supporting my Ph.D. study and related research and giving me the teachings and opportunities.

I want to thank my committee members, Prof. Karla Tereza Figueiredo Leite, Prof. Geraldo Braz Junior, Prof. Alberto Barbosa Raposo, and Prof. Waldemar Celes Filho, for their insightful comments and suggestions. Thanks to their kindness and sincere interest in my research, my defense was an enjoyable moment that I will not forget.

I want to thank my friend Pedro Bandeira for the discussions and essential support in several moments, the sleepless nights we were working together before deadlines, and all the fun we have had in the last years. To my friends Romeu de Oliveira and Eduardo Reis for following me closely for most of this journey. To Lillian Garcês because she actively showed me that she was cheering for me at various times.

I want to thank PUC-Rio. Especially to the professors in which I participated in the disciplines. To my Tecgraf colleagues and classmates. To the members of the Informatics Department, in special to Cosme. And to my other research colleagues.

Finally, I would like to thank Tecgraf/PUC-Rio, Capes, and CNPQ for their financial support.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) - Finance Code 001.

Thanks for everyone's contribution. I'm very grateful for everything.

Abstract

Valente, Thales Levi Azevedo; Gattass, Marcelo (Advisor). Santos, Paulo Ivson Netto **Method for Automatic Detection of Stamps in Scanned Documents Using Deep Learning and Synthetic Data Generation by Instance Augmentation**. Rio de Janeiro, 2022. 101p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Scanned documents in business environments have replaced large volumes of papers. Authorized professionals use stamps to certify critical information in these documents. Many companies need to verify the adequate stamping of incoming and outgoing documents. In most inspection situations, people perform a visual inspection to identify stamps. Therefore, manual stamp checking is tiring, susceptible to errors, and inefficient in terms of time spent and expected results. Errors in manual checking for stamps can lead to fines from regulatory bodies, interruption of operations, and even compromise workflows and financial transactions. This work proposes two methods that combined can address this problem, by fully automating stamp detection in real-world scanned documents. The developed methods can handle datasets containing many small sample-sized types of stamps, multiples overlaps, different combinations per page, and missing data. The first method proposes a deep network architecture designed from the relationship between the problems identified in real-world stamps and the challenges and solutions of the object detection task pointed out in the literature. The second method proposes a novel instance augmentation pipeline of stamp datasets from real data to investigate whether it is possible to detect stamp types with insufficient samples. We evaluate the hyperparameters of the instance augmentation approach and the obtained results through a Deep Explainability method. We achieve state-of-the-art results for the stamp detection task by successfully combining these two methods, achieving 97.3% of precision and 93.2% of recall.

Keywords

Stamp Detection; Deep Learning; Faster R-CNN; Scanned Documents; Instance Augmentation.

Resumo

Valente, Thales Levi Azevedo; Gattass, Marcelo (Advisor). Santos, Paulo Ivson Netto **Método para Detecção Automática de Carimbos em Documentos Escaneados Usando Deep Learning e Geração de Dados Sintéticos Através de Instance Augmentation**. Rio de Janeiro, 2022. 101p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Documentos digitalizados em ambientes de negócios substituíram grandes volumes de papéis. Profissionais autorizados usam carimbos para certificar informações críticas nesses documentos. Muitas empresas precisam verificar o carimbo adequado de documentos de entrada e saída. Na maioria das situações de inspeção, as pessoas realizam inspeção visual para identificar carimbos. Assim sendo, a verificação manual de carimbos é cansativa, suscetível a erros e ineficiente em termos de tempo gasto e resultados esperados. Erros na verificação manual de carimbos podem gerar multas de órgãos reguladores, interrupção de operações e até mesmo comprometer fluxos de trabalho e transações financeiras. Este trabalho propõe dois métodos que combinados podem resolver esse problema, automatizando totalmente a detecção de carimbos em documentos digitalizados do mundo real. Os métodos desenvolvidos podem lidar com conjuntos de dados contendo muitos tipos de carimbos de tamanho de amostra pequena, com múltiplas sobreposições, combinações diferentes por página e dados ausentes. O primeiro método propõe uma arquitetura de rede profunda projetada a partir da relação entre os problemas identificados em carimbos do mundo real e os desafios e soluções da tarefa de detecção de objetos apontados na literatura. O segundo método propõe um novo pipeline de aumento de instâncias de conjuntos de dados de carimbos a partir de dados reais e investiga se é possível detectar tipos de carimbos com amostras insuficientes. Este trabalho avalia os hiperparâmetros da abordagem de aumento de instâncias e os resultados obtidos usando um método Deep Explainability. Foram alcançados resultados de última geração para a tarefa de detecção de carimbos combinando com sucesso esses dois métodos, alcançando 97.3% de precisão e 93.2% de recall.

Palavras-chaves

Detecção de Carimbos; Aprendizagem Profunda; Faster R-CNN; Documentos Digitalizados; Aumento de Instâncias

Table of Contents

1. Introduction	14
1.1. Objective.....	16
1.1.1. General Objective.....	17
1.1.2. Specific Objectives.....	17
1.2. Contributions.....	17
1.3. Work Organization.....	18
2. Related Works.....	20
2.1. Stamp Detection.....	20
2.2. Logo Detection.....	21
2.3. FPN and Deformable Modules.....	22
2.4. Conclusions Obtained from Related Works.....	22
3. Theoretical Background	24
3.1. Review of Detection and Classification Deep Networks.....	24
3.1.1. Multi-Layer Perceptron.....	24
3.1.2. Traditional and Deformable Convolution Networks.....	25
3.1.3. Residual Networks.....	28
3.1.4. Feature Pyramid Networks.....	29
3.1.5. Faster R-CNN.....	29
3.1.6. Knowledge Transfer.....	34
3.2. Deep Explainability.....	35
3.3. Image Processing Operations.....	38
3.3.1. Thresholding.....	38
3.3.2. Mathematical Morphology.....	39
3.3.3. Image Rotation.....	40
4. Materials and Methods	42
4.1. Development and Experimental Setup.....	42
4.2. Software for Annotation Process.....	43
4.3. Method Proposed for Stamp Detection.....	46
4.3.1. Image Acquisition.....	46
4.3.2. Preprocessing and Splitting.....	46
4.3.3. Network Design.....	48
4.3.4. Evaluation.....	52
4.3.5. Experiments.....	53
4.4. Method Proposed for Stamp Instance Augmentation.....	56
4.4.1. Instance Augmentation Procedure.....	58
4.4.2. Experiments.....	60

5. Results	62
5.1. Stamp Detection.....	62
5.2. Instance Augmentation.....	67
6. Discussion.....	70
6.1. Stamp Detection.....	70
6.2. Instance Augmentation.....	73
6.3. Comparison with Related Works.....	75
7. Conclusion	77
References	78
Appendix 1	85
1. Supervised Training	85
1.1. LinearRegression.....	85
1.1.1. Univariate Linear Regression.....	86
1.1.2. Multivariate Linear Regression.....	88
1.2. Classification.....	88
1.3. Train, Validating and Test Sets.....	89
1.4. Error Analysis.....	90
2. Artificial Neural Networks	91
2.1. Introduction.....	91
2.2. Activation Function.....	92
2.3. Loss Function.....	94
2.4. Hyperparameters Tuning.....	96
□ Learning rate.....	96
□ Momentum.....	97
□ Regularization.....	97
2.4.1. Batch Normalization.....	98
Appendix 2	100
1. Extra Experiments	100

List of Figures

Figure 1: Architectural graph of an MLP.	25
Figure 2: Convolution layer illustration. Source: [38].	26
Figure 3: Deformable convolution layer illustration. Adapted from [15]. ...	27
Figure 4: Residual learning: a building block. Source: [42].	28
Figure 5: Schematic of the architecture from feature pyramid network (FPN). Adapted from: [15].	29
Figure 6: Schematic of the architecture from Faster R-CNN.	30
Figure 7: Relative encoding of the proposal.	31
Figure 8: Illustration of RPN anchors parametrizations.	31
Figure 9: RPN general architecture.	32
Figure 10: The main screen of the software developed for dataset annotation.	43
Figure 11: Illustration of the most frequent stamp types highlighted in screen 2 (on the right). The stamp segmented and the stamp selected in the gallery are shown for visual comparison before saving the data (on the screen left).	45
Figure 12: Search screen illustration.	45
Figure 13: Flowchart of each main phases and its respective steps of the data preparation.	47
Figure 14: Method proposed for dataset analysis. The qualitative analysis step serves as a guide for choosing network architecture.	49
Figure 15: Proposed network architecture designed using our qualitative analysis.	51
Figure 16: Graphic produced on the Tensorboard for the AP50 metric obtained from the application of the network on validation set. (a) Experiment shows best value on epoch 1800. (b). Experiment shows best value on epoch 4100.	56
Figure 17: Proposed method for Stamps Instance Augmentation.	58
Figure 18: Illustration of the distribution of the draw of angles for 1000 attempts.	60
Figure 19: 3D visualization illustrating the four instances selected (on yellow color) for the Instance Augmentation experiments.	61
Figure 20: Results for the AP50 metric in the 1699 epoch based on validation for 7 experiments with several Instance Augmentation settings. The epoch is highlighted in the chart.	62
Figure 21: Loss curves about the training base and the validation base for six different experiments using the proposed architecture.	64

Figure 22: Illustration in a 3D graph the distribution of test set instances. Instances represented by a red circle belong to classes absent from the training group, and instances represented by a blue square belong to classes present in the training group.	66
Figure 23: Illustration of the reduced characteristics of detected and undetected instances plotted on a three-dimensional graph. In blue, the instances detected by the network. Undetected instances are red, and instances selected for synthetic data generation experiments are yellow.	67
Figure 24: (a) Location of selected stamp instance. (b) Stamp not detected after generating two clean instances per page. (c) Stamp detected after generating 12 clean instances per page. (d) Stamp not detected after generating 12 instances, no cleaning, per page.	68
Figure 25: Single instance not detected by the network after generating synthetic data of selected stamps.	69
Figure 26: First point of view of case studies of the application of the neural network for stamp detection, where we highlight cases of success within the context of some simple (shape, intra-similarity, location, multiplicity) and complex (overlap/occlusion, quality, and rotation) features in the dataset.	70
Figure 27: Second point of view of case studies of the neural network application for stamp detection, where we highlight divergences between the thresholds applied in the test group. Cases (a) and (b) consider scores above 90% and cases (c) and (d) consider scores above 70%.	71
Figure 28: Second point of view of case studies of the application of the neural network for stamp detection, where we highlight cases of types in which the network had difficulties in detecting stamps	72
Figure 29: Example of a dataset with Heteroskedasticity. Adapted from: [70].	86
Figure 30: Model of an artificial neuron.	91
Figure 31: Illustration of bias modifying a potential neuron activation by an affine transformation.	92
Figure 32: (a) Identity activation function. (b) Sigmoid activation function. (c) Relu activation function.	93
Figure 33: (a) Identity activation function. (b) Sigmoid activation function. (c) Relu activation function.	94
Figure 34: Cross entropy conditions. (a) $t_n = 1$. (b) $t_n = 0$	95

List of Tables

Table 1: Main parameters tested in the experiments.....	54
Table 2: Distribution of training, validation, and test sets.	63
Table 3: COCO evaluation metrics and Af-score achieved by the best model from experiment.....	64
Table 4: Precision and recall values achieved using several thresholds for the IOU and score measures. We highlight the two best configurations used.	65
Table 5: Values achieved for f-score metric using several thresholds for the IOU and score measures. We highlight the two best configurations used.	65
Table 6: Results in detecting instances of test group stamps, which belong to types present and absent in the training group.	66
Table 7: Results in the detection of stamp types from the test group, and which belong to types present and absent in the training group	66
Table 8: Comparison between the previous and new the results achieved for validation and test sets using the proposed method for Instance Augmentation.	69
Table 9: Network confidence in detecting each instance of the type stamp that the network failed in detection.....	74
Table 10: Precision and recall values achieved after applying the data augmentation proposed method.....	76
Table 11: Comparison of the results for stamps detection using different network architectures and cost functions from extra experiments...	101

List of acronyms

AP - Average Precision

AR - Average Recall

CNN - **Convolutional Neural Network**

DCN – Deformable Convolutional Network

DPI - dots per inch

Faster R-CNN - Faster Region-Based Convolutional Neural Network

Fast R-CNN - Fast Region-Based Convolutional Neural Network

FC- Fully-Connected

FP - False positive

FPN – Feature Pyramid Network

FN - False Negative

GUI – Graphical User Interface

IOU - Intersection over union

MLP - Multilayer Perceptron

MBSGD – Mini-batch stochastic gradient descent

Resnet - Residual Network

ROI - Region of Interest

RPN - Region Proposal Network

SVM - Support Vector Machine

TN - True negative

TP - True positive

“So do not throw away your confidence; it will be richly rewarded”

Hebrews 10:35

1. Introduction

Several sectors of society use stamps to authenticate documents, such as security, industrial, governmental, educational institutions, medical prescriptions, sales, bank checks, and postal mail [1-6]. The entities issue their documents through printing on solid paper, and stamps guarantee the authenticity of the content [1]. Nevertheless, it has been a common practice for these entities to migrate to paperless offices by digitizing documents, storing and maintaining them in large databases [2]. With digitization, institutions preserve their physical documents and provide information access, retrieval, and indexing services, including content extracted from stamps [1, 2, 7].

Stamps combine textual and graphical components [2-3]. Text components provide information such as location or who validated information. Graphic components are geometric shapes that usually vary depending on the type of stamp. Stamps can contain variable fields to manually fill in dynamic information, such as dates or signatures, and serve to authenticate the identity of an authorized professional or an organization [5]. Stamped official documents are often accepted without question, and the owner cannot deny the legal effects stated in the paper [5, 8].

In most situations, people use their own eyes to identify stamps [4, 8]. The manual stamping checking is tiring, susceptible to errors, and inefficient time spent and expected results. Furthermore, some processes can take place in parallel and involve hundreds/thousands of pages that are immediately scanned. Errors in manual checking can lead to fines from regulatory bodies, interruption of operations, or even compromise of workflows and financial transactions. Often, there is no time or resources to perform manual, even predictive, checking [1]. Automatic stamp detection can reduce labor costs and alleviate all these presented problems.

The literature presents several challenges to be overcome to perform automatic stamp detection. Stamps can have arbitrary orientation and any position within a page, which requires detection throughout the entire image [2-3, 5]. Stamps may have missing parts or overlapping areas with other elements on the page (text, manuscripts, or even other stamps) [1-2]. Stamps contain unpredictable patterns due to poor ink conditions, uneven surface contact, noise, or characteristics of the page itself [1-2, 4]. Thus, two impressions of the same stamp can look significantly different [5, 7]. Types of stamps can vary in color, shape, aspect ratio, size.

For the past several years, object detectors based on Convolutional Neural Networks (CNN) have shown great results. The definition of the object-detection problem is to determine the location of the object in a given image (object location) and to which category each object belongs (object classification) [8]. Therefore, the stamp detection task can be reduced to the object detection task. So far, surveys found in the literature point to 2 main types of object detectors [9–12]. One-stage detectors give up accuracy to the detriment of lower consumption of computational resources. On the other way, two-stage detectors usually achieve better evaluation metrics, but require more computational power.

Several techniques have been inserted into object-detection frameworks to reduce or even solve the effects of the challenges of this task. Dai et al. [12] developed the DCN (Deformable Convolutional Network), in which they improved convolutional networks by introducing the operations of deformable pooling and convolutions. These techniques increase the convolutional kernels' receptive field sampling and pooling operation through additional offsets automatically learned from the target task. Although this technique improves the object pose robustness, point of view, non-rigid deformations, occlusion, and overlapping [12-13], we found few works that explored this [8, 14–18].

Another module not much explored in the literature on object detection tasks is the Feature Pyramid Networks (FPN) [18]. FPN produces and combines feature maps at different hierarchical levels and spatial resolutions in-network. In other words, this module creates a rich semantic pyramid, with features at various scales and hierarchical levels, from a single input image scale and a single convolutional backbone. Thus, this technique circumvents the bias of high computational power in multi-scale image analysis, since at first, each image scale would have to be processed by the neural architecture backbone, increasing resource consumption. The literature highlights that FPN handles problems involving multi-scale, small objects or objects of varying sizes as well [12, 19].

Despite the improvements in neural networks and the application of knowledge transfer techniques from pre-trained weights, the lack of representative data is still a problem that negatively affects many researchers that use deep learning [12, 20]. A solution found in the literature is the data augmentation of images. However, Ribani & Marengoni [20] point out that although several promising image augmentation techniques have been developed in the literature, increasing data with very low representativeness can lead to overfitting. They also point out that randomly increasing data can intensify intrinsic imbalance and that defining which amounts are satisfactory is a challenge.

Recently, instance augmentation research has been developed as another tool to address the lack of data [22-23]. Instance augmentation consists of

synthetically generating data from instances of objects of interest and real images. Changes that occur locally are new instances of the objects, which are “pasted” at the pixel level. Thus, instance augmentation provides a greater degree of fidelity at the global and local levels of the data. Instance augmentation can be wholly targeted to specific types of instances, taking new training data from a few images to a combinatorial level. In other words, with a single image containing a single object, it is possible to generate N new objects in a training image using instance segmentation. On the other hand, to create N objects with data augmentation, generating N images and changing the entire image is necessary.

In addition to a reasonable amount of data, computational power is another prerequisite for developing solutions based on deep learning. The available resources and the number of experiments directly affect the time needed to develop a solution. Nevertheless, Deep learning experiments usually have a high computational and time cost. In this sense, a concrete way to assess how a model can be improved or what effects additional training data would have been to use Deep Explainability techniques. Deep Explainability can improve debugging processes of models by using tools to recognize and understand failure cases or emphasize discovering problems that limit learning and inferring networks [23]. For example, visualization techniques in the space of features can help gain insights into changes in the model's behavior and how its predictions are affected in developing and testing a solution [24]. In this research, we use Deep Explainability to guide the performance of experiments using instance augmentation.

Some state-of-the-art works found in the literature use FPN and DCN for detection tasks in different domains [8,14-15,17]. However, these jobs typically perform extensive testing to find the most suitable architecture, do not analyze failure cases, and do not exploit data augmentation. This research designs and develops a deep learning-based solution for stamp detection that uses FPN and DCN based on the characteristics of the data used. Model failure cases can be evaluated using visualization techniques from Deep Explainability. This assessment guides a proposed solution to work around the previously observed failure cases based on instance augmentation. The research was successful in all these steps, and the lessons learned and presented can be used in other object detection problems.

1.1. Objective

This section presents the general and specific objectives to be achieved during the development of this work.

1.1.1. General Objective

The general objective of this work is to present a new computational method for the automatic detection of stamps in images of digitized documents, investigate which types of stamps are more difficult in detecting and propose a new method for data augmentation to solve the most difficult cases found.

1.1.2. Specific Objectives

To achieve the intended general objective, it was necessary to fulfill the following specific objectives:

- Design the network architecture based on the advances found in the Deep Learning and Object Detection literature, but taking into account the limitations of computational resources and time to run the experiments;
- Conduct experiments on the defined architecture evaluating the model through evaluation metrics commonly used in the literature;
- Investigate which types of stamps the final model had more difficulty detecting through a Deep Explainability method.
- Design and apply an instance augmentation algorithm over the types of stamps selected by the Deep Explainability to generate synthetic data from real pages and stamps. Also, verify if data augmentation can help to improve the detection of these types of stamps.

1.2. Contributions

We propose a state-of-the-art computational method capable of fully automating the detection of stamps in digitized documents, using a convolutional framework based on two stages for object detection. We also explore the new version of deformable modules proposed by Zhu et al. [25] and the use of FPN, modules that, as far as we know, have never been used for the stamp detection task.

Through our method, we design a new neural network oriented to the difficulties found in stamps detection task problem. The method evaluated 469 stamps distributed in 251 types. We only use 80 types of stamps for training. Still, we achieved 97.3% precision and 93.2% recall. We showed that our network could generalize knowledge and detect more than 3x of the types of stamps present in the training set.

Among other points, this work presents the following contributions:

- We illustrate a path that replaces the “brute force testing” of architectures to solve object detection tasks. The network architecture design is guided by the relationship between the problems identified in our dataset and the challenges and solutions of the object detection task pointed out in the literature. With the strategy, we significantly minimized the amount of extensive testing;
- As far as we know, this work is the first to propose a model combining Faster Region-Based Convolutional Neural Network (Faster R-CNN), DCN version 2, Residual Networks (Resnet) as resource extractor backbone and pyramidal network of features for the stamp detection task. FPN has been added to Faster R-CNN to use features in the top layers as well as in the shallow layers for detecting stamps of different sizes in scanned document images.
- We propose an innovative greedy strategy for data splitting considering the distribution of stamps and their types globally (across the dataset) and locally (per document page image), which can also be helpful for other problems involving detecting multiple object types in images;
- As far as we know, this work is the first to evaluate results obtained by the network through a Deep Explainability method that combines network feature extraction and dimensionality reduction techniques to generate a 3D visualization;
- We propose a method for instance augmentation of stamp datasets from real data to investigate whether it is possible to detect stamp types with insufficient number of samples.
- We achieved state-of-the-art results for the stamp detection task through the successful combination of: (1) classic and novel Data Engineering methods, (2) a novel Object Detection method that combine the recent advances of the Deep Learning literature and (3) a new Deep Explainability method used to debug the network and guide the hyperparameters selection.

1.3. Work Organization

In addition to this introductory chapter, there are 6 more chapters, which complete this thesis and are structured as follows:

Chapter 2 presents work related to the detection of stamps and logos, which

are similar in nature. This chapter also presents works that involve Object Detection, FPN networks and deformable networks.

Chapter 3 presents theoretical review underlying this work to familiarize the reader with the main concepts used to build the method.

Chapter 4 shows all the stages of development of this research, starting with the acquisition of documents, followed by the method developed to detect stamps in documents. We describe general information about the dataset, its construction process, challenges, our data analysis performed to guide the network design, the greedy strategy proposed for dividing the dataset, the performed experiments and the evaluation metrics. This chapter also presents the proposed method for generating synthetic stamp data.

Chapter 5 presents the results achieved by applying the proposed method on our dataset, the discussions, and some case studies of success and failure.

Chapter 6 discusses some case studies of success and failure obtained by the proposed methods.

In Chapter 7, the conclusions inferred about the methods are presented, together with the suggested future work to improve the research.

Finally, the Appendix 1 briefly describes the concepts and nomenclature employed in this work in relation to neural networks and Appendix 2 presents extra experiments performed in this work.

2. Related Works

This chapter presents and briefly describes some of the works found in the literature related to the task of detecting stamps in documents. Since we found few works related to the problem of this research, we also investigated some works with the theme of stamp recognition and logo detection in images to add insights from these studies. Logos are graphic objects that have many properties in common with stamps. We also present works that used FPN networks of deformable modules to solve other tasks and that showed promising results.

2.1. Stamp Detection

Usually, computational techniques aimed at detecting and recognizing stamps involve two significant areas of computational research: image processing and machine learning. We can find research involving template matching, image registration, morphological operations such as skeletonization and thinning of binary images, invariant transformations based on edges, color space transformation, hough transform, grouping, connected components, extraction of geometric features, and classification using Support Vector Machines (SVM) [1–3, 5, 27].

Chen [26] conducted one of the first survey in stamp recognition. The work was limited to specific regions of the image and considered only circular and rectangular shapes, stamps without imperfections, 4 types, and 4 orientations. The detection step was performed manually. In Nourbakhsh et al. [5] the authors considered only one stamp class, with specific sizes, dimensions, font, and style, only 4 specific orientations. Also, they did not consider cases of stamps overlapping with other stamps. They achieved 82% accuracy when detecting the presence of stamps in 1,200 images.

Roy et al. [4] the authors considered 12 types of stamps and circular and rectangular shapes. They did not assess how much they achieved in stamp detection task, but they did get 92.03% accuracy for the stamp recognition task on 127 documents. Another work of Roy et al. [3] considered 12 types of stamps with circular or rectangular shapes. The method achieved 92.42% accuracy for recognizing stamps on 530 documents. The detection step achieved 100% precision at the cost of a very low recall, i.e., 20%. Their method is based on handcraft low features.

The authors Micenková & Beusekom [1] evaluated their stamp detection method at two different image resolutions: 200 and 300 dots per inch (DPI). Its best result was achieved at 300 DPI: 83.4% recall and 83.8% accuracy on 320 images.

In this evaluation, they considered a single stamp class, black and without the presence of overlapping. When considering overlapping cases, his method achieved 69% accuracy and 68% recall on 52 stamps.

Concerning works that used deep learning, in Sharma et al. [2] the authors evaluated the use of one-stage and two-stage object detection frameworks. For the first type, they evaluated the application of the Yolov2 network. For the second type, they evaluated the application of the Faster R-CNN network with three different backbones (VGG16, VGG_M and ZF), separately. The best result was using the Faster R-CNN, achieving 89.2% average precision (AP) and 89.6% accuracy in detecting stamps in 60 test images. In Jun et al. [27] the authors proposed the use of Fast Region-Based Convolutional Neural Network (Fast R-CNN) to detect elements in documents. Their method achieved 97% AP metric on 53 pages of documents.

2.2. Logo Detection

Logo detection in real-world scenes has several similarities to the stamp detection problem. Logos have well-defined geometric shapes, may or may not have strings, and usually do not have well-defined locations and quantities in the image. However, logos usually have more textures and colors, and overlapping between them is rarer since they are usually associated with another object in the image to which they belong. Nevertheless, some conclusions obtained in this research can be used for the stamp detection problem.

Palecek [28] applied, in different logo datasets, two-stage object detection frameworks such as Faster R-CNN and Mask R-CNN, and a one-stage object detection framework (RetinaNet). Their final evaluation obtained better results using object detection frameworks based on two stages than frameworks based on one stage and segmentation networks. Song & Kurniawati [29] also compared the same types of object detection frameworks (two-stage and one-stage). They also evaluated a training base composition with only synthetic data, real data, and synthetic data. The best results were achieved using frameworks based on two stages in all experiments. Regarding the composition of the training base, the best result was composed using real and synthetic data.

Bhunja et al. [30] evaluated the application of several deep network architectures in logo detection such as Yolo, Faster-RCNN, U-Net, SiameseFCN, CoFCN, SG-One. They observed that traditional frameworks have limited performance with little data in the training base, even using pre-trained weights, tending to overfit a few times. From there, they proposed a new architecture based on multi-scale feature analysis and residual connections, like networks such as FPN and ResNet. They also evaluated the performance of networks in a more open

scenario, that is, where the training set and the test set had utterly different samples and surpassed the results of previous architectures.

Bhunja et al. [30] also evaluated the performance of networks in a more open scenario, that is, where the training set and the test set had utterly different samples. Although his method outperforms previous work in this more challenging scenario, his work achieved 89.2% AP in logo detection. Other limitations pointed out by the authors were problems with the imbalance between background and foreground information and detecting small size logos. Guo et al. [31] also had problems with small size logos in their work, although he achieved better results after applying data augmentation techniques and Faster R-CNN network architecture.

2.3. FPN and Deformable Modules

Works that combine FPN with deformable modules are not yet typical in any field. However, we found some works in different areas that used this combination. Overall, FPNs bring the advantage of multiscale hierarchical feature analysis, and deformable modules bring greater robustness to geometric transformations, overlap, and occlusion.

Ren et al. [13] proposed a method based on Faster R-CNN combining multiscale feature analysis and deformable modules to perform object detection in remote sensing optical images and achieved better results than traditional architectures. Shi et al. [16] evaluated different architectures of deep networks for detecting marine organisms in videos. The authors concluded that the use of FPN brings much better results than object detectors that do not use it. They also point out that DCN has the advantage of making the model invariant to geometric transformations since the network generalizes the learning about these transformations based on the data itself.

In their studies, Deng et al. [14] also found better results when combining FPN with deformable modules in their visual classification task of concrete cracks. Finally, Han et al. [6] compared one-stage and two-stage-based frameworks and FPN and DCN to analyze of airport remote sensing images. They got better results combining a 2-stage framework with FPN and DCN.

2.4. Conclusions obtained from related work

In general, works aimed at detecting stamps are difficult to compare. Authors usually present the results achieved in metrics but provide what they achieved in the number of images, not in numbers of stamps. The works usually use private data and, even in cases where they used the same database, the process of

dividing the data into training, validation, and testing groups (proportion and specific instances for each group) are different. We also have examples of works with minimal scope.

The works focused on the other themes, logo detection and the use of FPN networks and deformable modules agree with the studies of Zhao et al. [9], Zou et al. [10] and Liu et al. [12], extensive object detection surveys. Object detection frameworks usually achieve better metrics, feature analysis at a hierarchical and multiscale level brings excellent gains, and the use of deformable modules brings greater robustness when the problem presents overlapping and geometric variations. Finally, we also verified that the literature points out significant gains in using synthetic data with real data in the training base

3. Theoretical Background

This chapter presents a theoretical review of the main topics related to artificial neural networks, deep explainability and image processing that are employed throughout this thesis. Appendix 1 contains an additional overview of more foundational concepts.

3.1. Review of Detection and Classification Deep Networks

This section introduces the main concepts about neural networks employed in our framework for stamp detection.

3.1.1. Multi-Layer Perceptron

In multilayer neural networks, the neurons are arranged in a layered fashion, in which a group of hidden layers separates the input and output layers [32]. Multilayer perceptron (MLP) is a fully connected multilayer neural network in its general form. A neuron in any network layer connects to all the neurons (nodes) in the previous layer. Signal flow through the network progresses in a forward direction, from left to right and on a layer-by-layer basis. The following three points highlight the basic features of MLP [33]:

- The model of each neuron in the network includes a nonlinear activation function that is differentiable;
- The network contains one or more layers that are hidden from both the input and output nodes;
- The network exhibits a high degree of connectivity, which is determined by the synaptic weights of the network.

The output neurons constitute the output layer of the network. The remaining neurons constitute hidden layers of the network. Thus, the hidden units are not part of the output or input of the network—hence their designation as “hidden.” The first hidden layer is fed from the input layer made up of sensory units (source nodes); the resulting outputs of the first hidden layer are in turn applied to the next hidden layer, and so on for the rest of the network [33].

However, if the activation functions of all the hidden units in a network are linear, then for any such network, we can always find an equivalent network without hidden units. This follows from the fact that the composition of successive linear transformations is itself a linear transformation [34]. Equation 1 describes the output of a layer.

$$y^l = \varphi(W^l y^{l-1} + b^l) \quad (1)$$

Where y^l is the output vector, W^l the weights matrix of each neuron pair of layer l and $l-1$, and b^l the bias term vector of each neuron in layer l . The Figure 1 illustrates an architectural graph of an MLP.

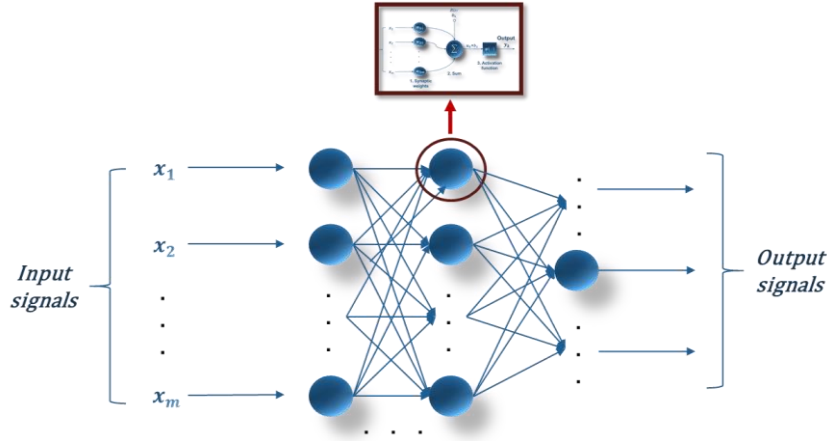


Figure 1: Architectural graph of a MLP.

3.1.2. Traditional and Deformable Convolution Networks

CNN are deep supervised machine learning algorithms. Two significant advantages can be highlighted from this deep network model: (1) excellent learning capacity, that is, making strong and mainly correct assumptions about the nature of the data and (2) being easy to train compared to neural networks with a similar number of layers [35]. CNN are applied in many problems, including extensive data analysis, computer vision and image analysis, speech recognition, natural language processing, and recommendation systems.

CNN forgoes designing and extracting a handcrafted set of features and, instead, feeds data directly into the network. These networks mainly consist of many convolutional layers, interspersed with pooling layers that reduce the dimensionality of the input signal, and usually a few fully connected layers and a final classification layer. The convolutional layers can be thought of as a feature extraction subsystem, not designed or selected by algorithm developers but learned explicitly for the task at hand during the training process [36].

Neurons that belong to the convolutional and completely connected layers are often combined with a bias and an activation function. The activation function of the neurons presented at the end of the completely connected layer can vary according to the type of problem in which the network is applied. However, the relu is the most used activation function in hidden layers since Krizhevsky et al. [35] proved that its mathematical simplicity allows the stochastic gradient descent to

converge up to 6 times faster compared to the sigmoid and tanh functions.

Each convolutional layer consists of groups of 1D (for signals or sequences), 2D (for images or spectrograms), or 3D (for videos or volumetric images) neurons called kernels. Sets of kernels in the convolutional kernel form filters, which extract characteristics whose internal values are adjustable synaptic weights. In other words, the convolutional filters are trainable features extractors. Signal units that pass from one layer to another are organized on feature maps.

The filters define a small area (3x3, 5x5, 7x7 pixels). Each unit is connected to local patches in feature maps of the previous layer through the convolutional filters and is then passed through a nonlinearity (activation function). All units in a feature map share the same convolutional filter, and different feature maps in a layer use different convolutional filters. The shared filters reduce the number of connections, reducing training time and chances of overfitting. These factors speed up the learning and reduce the memory requirements for the network. Figure 2 illustrates how convolution occurs in an image.

DCNs are designed to handle critical challenges in visual recognition as geometric variations or model geometric transformations in object scale, pose, viewpoint, and part deformation [12]. By adding 2D offsets to the regular convolution grid in the standard convolution, deformable convolution sample features from flexible locations instead of fixed locations, allowing for the free deformation of the sampling grid. The spatial sampling locations in deformable convolution modules are augmented with additional offsets, learned from data, and driven by the target task [14].

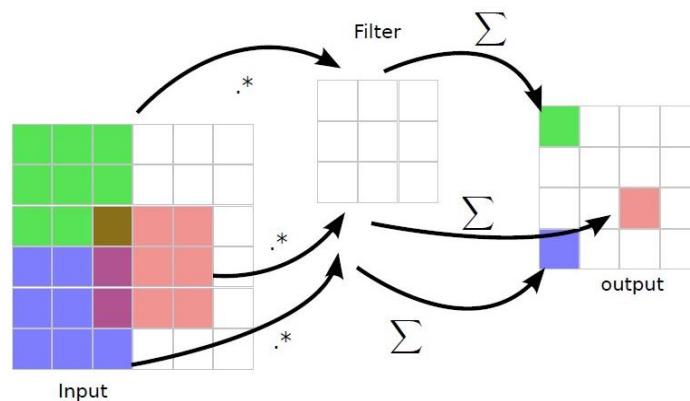


Figure 2: Convolution layer illustration. Source: [37].

A standard convolution consists of two steps: (1) Sampling using a regular grid R over the input feature map X ; and (2) summation of sampled values weighted by W . The grid R defines the convolution filter by size and dilation. For example, $R = \{(-1,1), (-1,0), \dots, (0,1), (1,1)\}$ defines a 3x3 filter with dilation 1. We can

derive the standard convolution output of each position p on the output feature map Y , according to the following formula:

$$Y(p) = \sum_{k=1}^k w_k \cdot x(p + p_k), \quad (2)$$

In Zhu et al. [26], deformable convolution was defined by augmenting the regular grid R with 2D offsets, where given a convolutional filter of K sampling locations, w_k and p_k denote the weight and pre-specified offset for the k -th location, respectively. For example, $K = 9$ and $p_k \in \{(-1, -1), (-1, 0), \dots, (1, 1)\}$ defines a 3×3 convolutional filter of dilation 1. Let $x(p)$ and $y(p)$ denote the features at location p from the input feature maps x and output feature maps y , respectively. The modulated deformable convolution can then be expressed as:

$$Y(p) = \sum_{k=1}^k w_k \cdot x(p + p_k + \Delta p_k) \cdot \Delta m_k, \quad (3)$$

where Δp_k and Δm_k are the learnable offset and modulation scalar for the k -th location, respectively. The modulation scalar Δm_k lies in the range $[0, 1]$, while Δp_k is a real number with unconstrained range.

Figure 3 illustrates the difference between regular and deformable convolutions: the former's sample matrix is fixed and regular, whereas the latter's is unfixed and malleable. As a result, the receptive field used to perform dot product with the kernel is regular for the former, while for the latter, it is irregular. It is worth noting that the deformable convolution's sample matrix offset is chosen by an algorithm that can better learn the geometrical properties of the objects to be recognized.

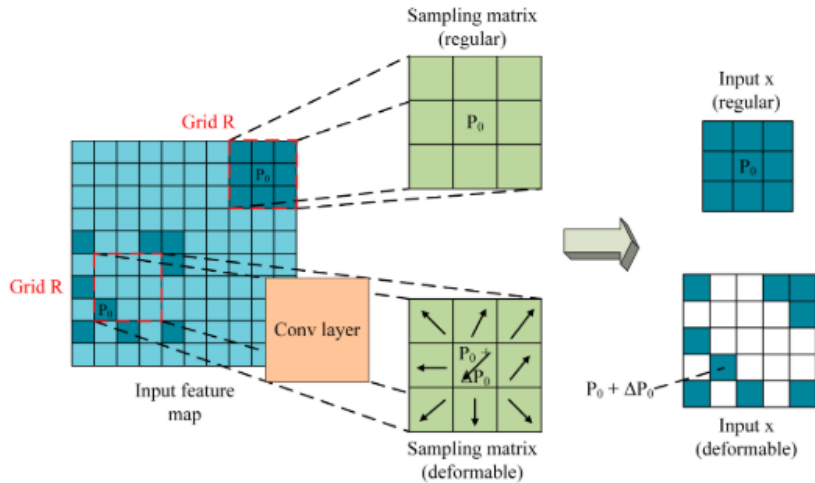


Figure 3: Deformable convolution layer illustration. Adapted from [14].

3.1.3. Residual Networks

In a CNN, the first layers represent low-level local features, such as edges, while deeper layers capture more complex and specific shapes [38]. The researchers believed that deeper networks should provide better results. However, experiments showed that the network performance did not behave as expected after adding a certain number of layers. The backpropagated error disappearance in higher levels of the network led some neurons to lose the ability to learn due to the lack of updating their connections. Repeated multiplication of small numbers made the backpropagated error infinitely small.

The researchers have experimentally proven that adding layers made the method a complex optimization problem: when the model introduces more parameters, it becomes more challenging to train the network [38]. They observed that when the depth of the network increases, the precision reaches a saturation point and then degrades quickly. Unexpectedly, this degradation was not caused by overfitting, as adding more layers led to a more significant error even in the network training stage [39].

In 2015, He et al. [40] proposed the Resnet, a CNN developed by Microsoft and submitted to the Large-Scale Visual Recognition Challenge 2015 an object classification competition. The Resnet authors also proved that (1) extremely deep Resnets are easy to optimize, but “simple” counterpart networks (which stack layers) exhibit a higher training error when their depth increases; (2) Resnets can easily enjoy precision gains from increasing depth, producing results substantially better than previous networks.

Resnet consists of stacked residual blocks linked via shortcut connections. Feedforward neural networks with "shortcut connections" can implement the formulation of $F(x) + x$. (Figure 4). Connections that bypass one or more layers are known as shortcut connections. The shortcut connections merely conduct identity mapping; their outputs are appended to the stacked layers' outputs. Shortcut identity links do not add any more parameters or computational complexity. Stochastic gradient descent can still train the complete network end-to-end [41].

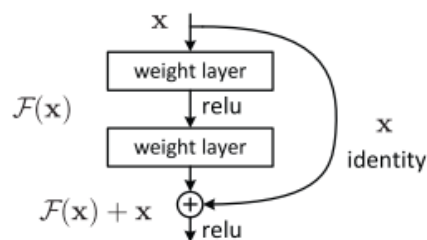


Figure 2. Residual learning: a building block.

Figure 4: Residual learning: a building block. Source: [41].

3.1.4. Feature Pyramid Networks

FPN is designed to improve the detection of multiscale objects by fusing the output of each layer of the backbone network in a top-to-down manner. It takes a single-scale image of arbitrary size as input and outputs proportionally sized feature maps at multiple levels in a fully convolutional fashion. Features are computed on each image scale independently, enabling a model to detect objects across an extensive range of scales. This process is independent of the backbone convolutional architectures [16-17, 19].

FPN combines the upper layer (used to acquire low-resolution but powerful semantical features) and the lower layer (used to acquire high-resolution but weak semantical features) through lateral connections to improve feature extraction capability. Specifically, the feature pyramid architecture is composed of five feature maps denoted as $\{P2, P3, P4, P5, P6\}$, among which $P2, P3, P4$, and $P5$ are calculated by feature maps $\{C2, C3, C4, C5\}$ with the lateral connection, respectively, and the max pooling operation of $P5$ generates $P6$ [8, 15].

The overall network framework consists of a bottom-up pathway, a top-down pathway, and lateral connections, as shown in Figure 5. The Bottom-up pathway is the feed-forward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2 [18]. On the top-down pathway, the lateral connections merge feature maps by element-wise addition from the bottom-up pathway. The coarser-resolution feature maps are upsampled by a factor of 2 (using nearest neighbor upsampling for simplicity). Convolutions 1×1 and 3×3 are applied in the ways.

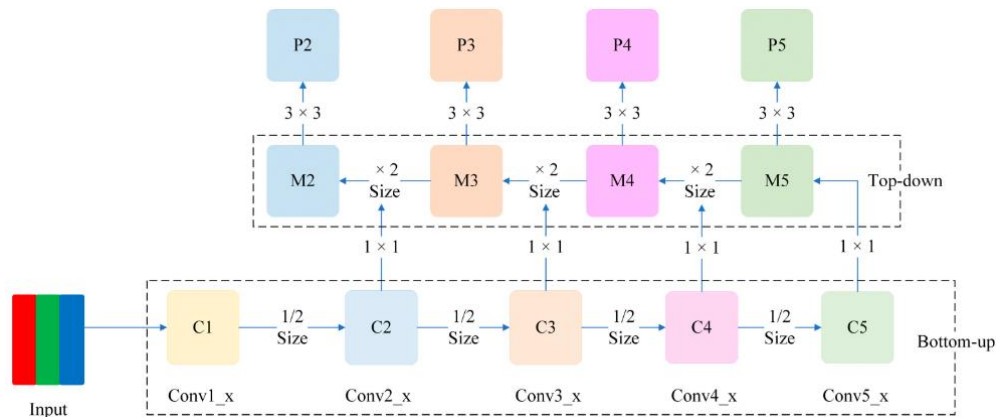


Figure 5: Schematic of the architecture from feature pyramid network (FPN). Adapted from: [14].

3.1.5. Faster R-CNN

We can define a spatial location on computational vision as the smallest rectangle aligned to the axis that entirely involves an object. A good object detector

must be able to identify multiples and partially occluded objects in arbitrary scenes, be invariable to the scale, point of view, and orientation of the object, and on several locations. Object detection is a type of supervised learning problem that consists of predicting the class of one or more objects in the same image and delimiting the fairer bounding box that encompasses them.

The object detection task has been by an increasing amount of attraction due to three main fronts: application, data, and technological developments [10, 12, 32, 52–54]. Literature divides the ConvNets based approaches to object detection into two categories: two-stage detection framework and one-stage detection framework [10, 12, 32, 54]. The two-stage detection framework is slower in more accuracy, while a phase is faster but less accurate [9, 11, 54].

Faster R-CNN is currently one of the usual representative methods in object detection [15]. This object detection framework is the first two-stage end-to-end unified deep learning detector that enables nearly cost-free region proposals. It shares convolutional features maps unifying Region Proposal Network (RPN) and Fast R-CNN algorithms, generating marginal cost for computing region proposals [45].

This framework can be divided into four main parts: the backbone, Region Proposal Network (RPN), Region of Interest (ROI) Pooling layer, and a Fully-Connected (FC) layer (Figure 6). The backbone serves as a feature extractor, extracting the semantic features from the input image and producing a feature map for the subsequent steps [15]. The backbone can be any convolutional network, and it processes the whole image with several convolutional and max-pooling layers to produce a convolutional feature map.

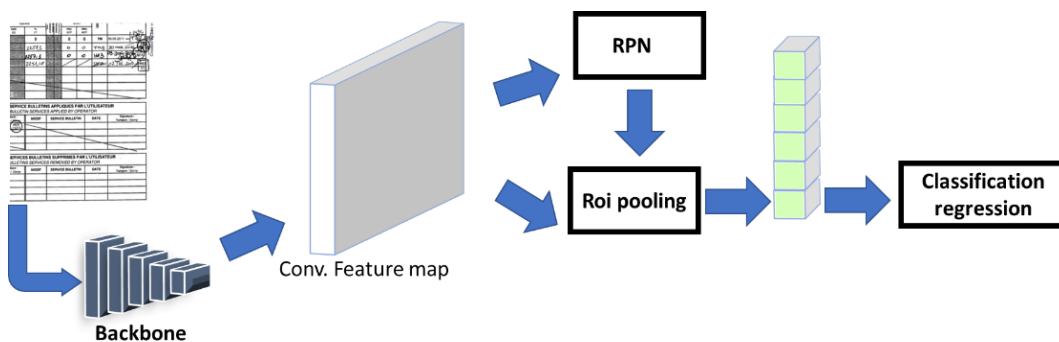


Figure 6: Schematic of the architecture from Faster R-CNN.

The goal of RPN is to produce possible object regions called object proposals. RPN uses structs called anchors. Anchors are references boxes for encoding proposals. Let (x, y, w, h) and (d_x, d_y, d_w, d_h) be the bounding boxes of a proposal and an anchor, respectively. A proposal is parameterized as $(d_x,$

d_y, d_w, d_h) relative to an anchor, where (d_x, d_y) is the displacement vector from anchor center (x_a, y_a) to proposal center (x, y) divided by anchor width and weight, respectively. The proposal's scaling factors in width and height concerning the anchors are (d_x, d_y) . The Figure 7 illustrates this relative encoding of proposal.

RPN generates proposals via anchors by placing nine anchors centered at each point of the convolutional feature map. Three aspect ratios and three scales are used to recognize objects with multiple dimensions and add scaling variance, resulting in these nine anchors. The RPN predicts one proposal concerning each anchor based on 6 parameters to describe it: (d_x, d_y, d_w, d_h) relative to bounding box parameters and (d_{obj}, d_{bg}) relative to object/background class probabilities. Thus, nine proposals relative to these nine anchors centered at each point are predicted by giving 9×4 -d relative box parameters and 9×2 -d class probabilities (Figure 8).

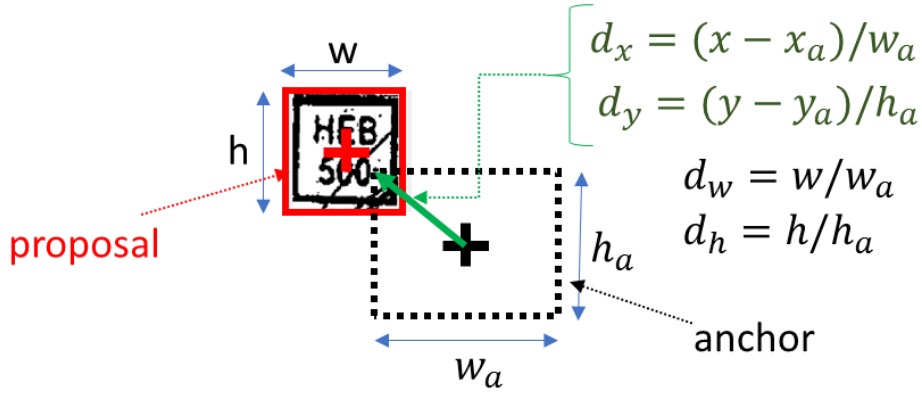


Figure 7: Relative encoding of the proposal.

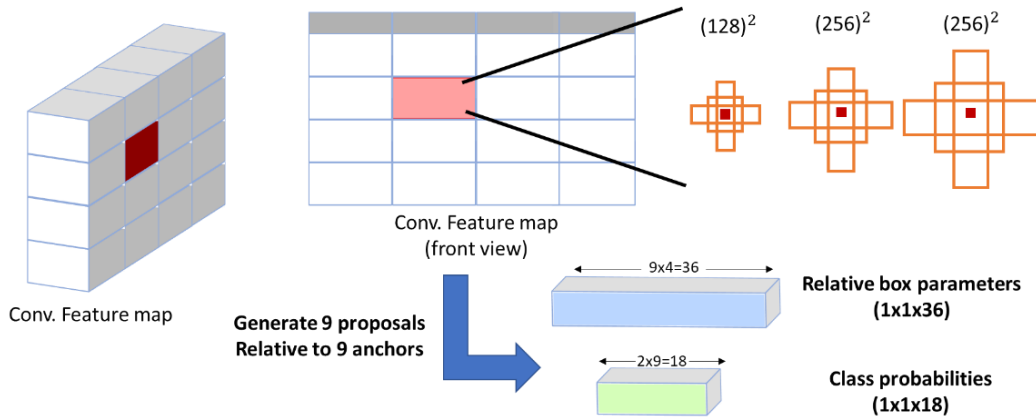


Figure 8: Illustration of RPN anchors parametrizations.

After explaining the concept of anchors, we can describe the overview of the RPN algorithm. Firstly, RPN receives as input the convolutional feature map

computed by the backbone. Then, RPN predicts nine proposals relative to 9 anchors for all points of the convolutional feature map. Finally, it generates 36 and 18 channels for box parameters and class probabilities, respectively. The final object proposals are generated for further processing by suppressing non-maximum proposals.

The RPN architecture consists of three convolutional layers, as illustrated in Figure 9. The intermediate layer converts the input convolutional feature map to the one specifically for proposal generation. The regression layer predicts the box parameters of all proposals. Finally, the classification layer predicts the object/background probabilities of all proposals.

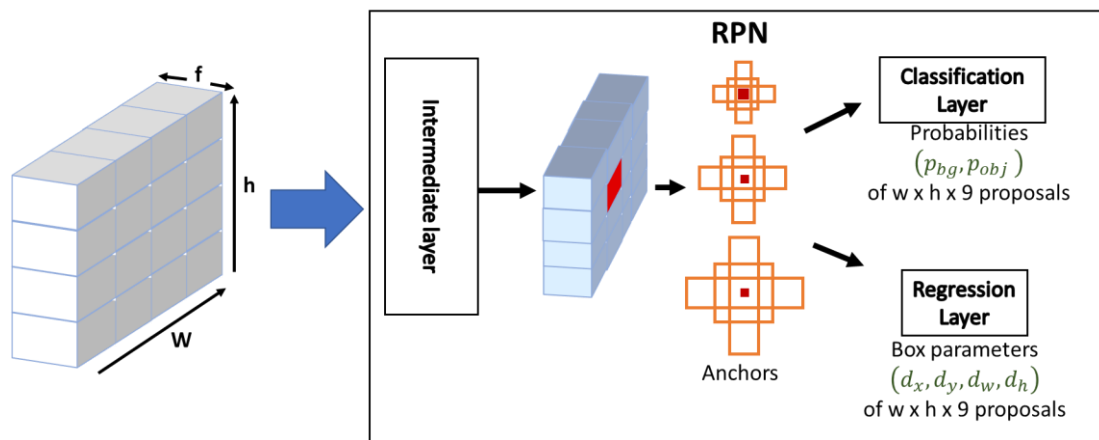


Figure 9: RPN general architecture.

The training overview for RPN is following:

- Assign a label to each anchor.
 - Use Intersection over Union (IoU) for measuring box overlap to define the labels.
 - Positive anchors have $\text{IoU} > 0.7$ with any object.
 - Negative anchors have $\text{IoU} < 0.3$ for all ground-truth boxes.
- Form a mini batch consisting of 256 anchors.
 - 128 positive (object) anchors.
 - 128 negative (background) anchors.
- Minimize the defined loss function defined for Equation 4.
 - Optimization: stochastic gradient descent

$$L(d^{(i)}, p^{(i)}) = \sum_i \hat{p}_{(obj)}^i \times L_{reg}(d^{(i)}, \hat{d}^{(i)}) + \sum_i L_{cls}(p^{(i)}, \hat{p}^{(i)}) \quad (4)$$

where $\hat{d}^{(i)}$ is the box parameters of the ground-truth box associated with i^{th} anchor, $\hat{p}^{(i)}$ is the ground-truth class probability of the i^{th} anchor, $(d^{(i)}, p^{(i)})$ is the proposal parameters predicted by RPN via anchors, and $L(.)$ evaluates the loss of classification and regression of the parameters.

After generating object proposals, we will go through how to use ROI pooling to extract the feature map of each proposal, which consists of three steps: ROI clipping, ROI division, and max pooling. The convolutional feature map is cropped according to the object proposal box using ROI clipping. The ROI feature clip is then divided into 7×7 grids using ROI division. To create a proposed feature map, max pooling is done to each channel of the grid. We acquire the $7 \times 7 \times 512$ convolutional feature map for an item proposal, which is flattened to a feature vector.

Finally, like RPN, the FC layer has the objective of the predict regression offset and class probabilities for every proposal. It reduces feature dimensions to 4096 by using two hidden layers. Then, there are two wholly linked branches for estimating regression offset and class probabilities. The regression branch predicts $N+1$ classes target boxes, each for a class label, including a background label. For example, if the number of classes is 2, the regression branch predicts 12 regression offsets: 8 for the classes and 4 for the background. The classification branch predicts the class probabilities for all class labels.

The training overview for Faster R-CNN is following:

- Select an image with its labels from the dataset
- Flow through Faster R-CNN network to obtain:
 - RPN: regression offset map and classification map (background or foreground)
 - FC Layer: object regression offset and class probabilities
- Computer prediction loss for updating parameters
 - RPN: convolutional kernels using Equation 4
 - FC layer: weight matrices using Equation 5

$$L(d^{(i)}, p^{(i)}) = \sum_i (1.0 - \hat{p}_{(bg)}^{(i)}) \times L_{reg}(d^{(i)}, \hat{d}^{(i)}) + \sum_i L_{cls}(p^{(i)}, \hat{p}^{(i)}) \quad (5)$$

3.1.6. Knowledge Transfer

The literature shows that transferring knowledge obtained by one network to another improves the performance of the latter. This improvement happens if the networks are designed and trained for different tasks. This knowledge transfer consists of getting the weights of neurons adjusted for a previous task and training on top of them to adjust the network to another task. Yosinski et al. [42] presents that even if this transfer occurs between training for different tasks, the results are still better than using random weights.

In addition, to further improve the results, these weights can be readjusted in the training of the new task. These new settings allow the network to better adapt to the new input patterns. Two strategies were developed to transfer knowledge: (a) fine-tuning and (b) transfer learning. The first strategy consists of using the parameters of the pre-trained network as initial parameters for training with a new dataset rather than using random parameters. The second strategy consists of freezing a defined number of layers of the network in the training process.

Deciding which techniques to transfer knowledge depends primarily on the difference between the nature of the pre-training data and the nature of the objective task data or the goal task. For example, transfer learning may be a good idea when transferring the style from one image to another image [46]. However, if the target dataset is small and the number of parameters is huge, fine-tuning the whole network may result in overfitting [32, 52]. Alternatively, the last few layers of the deep network can be fine-tuned while freezing the parameters of the remaining initial layers to their pre-trained values [32, 57, 58].

The literature encouraged us to use knowledge transfer to initialize the network weights. For example, Akçay et al. [49], in their task of detecting firearms in X-ray images, freezes the first layers and fine-tunes in the other layers of an AlexNet based network pre-trained with natural objects and achieved superior results to previous work. The experiments of Tajbakhsh et al. [48] considered several medical imaging applications and segmentation from three imaging specialties (radiology, cardiology, and gastroenterology). The authors assessed the performance of deep CNNs trained from scratch to CNNs that had been fine-tuned layer by layer and have proven that fine-tuning in a deep network is equal to or greater than training a network from scratch with images of the exact nature.

The latter brings us greater motivation since the difference between medical

images and natural objects is proportional to our dataset with scanned documents and natural objects. Finally, these bibliographies also show us no definitive rule on what layers should be used in transfer learning or fine-tuning, even though it provides intuition.

3.2. Deep Explainability

Researchers and model developers have a strong understanding of deep learning techniques and a well-developed intuition surrounding model building. Their knowledge expedites key decisions in identifying the which types of models perform best on which types of data. These individuals wield mastery over models, e.g., knowing how to vary hyperparameters in the right fashion to achieve better performance. Their expertise helps them make quick judgments about which sorts of models work well with various types of data. These people have a knowledge of models, for example, understanding how to modify hyperparameters in the proper way to improve performance.

However, the internal complexity and nonlinear structure of deep neural networks do the underlying decision-making processes for why these models are achieving such performance are challenging and sometimes mystifying to interpret [50]. It makes them opaque and black box models with an accuracy and interpretability tradeoff, i.e., more performing models are less interpretable [51]. Due deep networks has a black-box nature, researchers are developing methods focused into “open them” to produce better explanations and “see through the black-box” using phrases such as “opening and peering through the black-box”, “transparency,” and “interpretable neural networks” [50].

Because the underlying functioning of the deep networks is not evident, it becomes difficult to justify the outcomes of such models. It is necessary to bring in the explainable AI techniques to understand and explain such methods working and processes [35, 36]. As deep learning spreads across domains, it is of paramount importance that we equip users of deep learning with tools for understanding when a model works correctly, when it fails, and ultimately how to improve its performance [50].

In general, systems are interpretable if humans understand and interpret their working mechanism and decision-making process by asking questions like why the system made a particular prediction? Why answer the interpretability aspect, and how justifies how the system came up to a specific decision answer the explainability part. “Interpretability is the degree to which a human can understand the cause of a decision and can consistently predict the model’s results”. Deep neural nets lack interpretability [35, 37].

Unfortunately, there is no universally formalized and agreed upon definition for explainability and interpretability in deep learning, which makes classifying and qualifying interpretations and explanations troublesome [50]. However, the literature often uses the keywords interpretability and explainability referring to similar concepts [24-25, 34]. Some works consider them to be concepts pointing in the same directions and as being interchangeable [33, 36]. For these reasons and for the sake of simplification, the minor variations between those terminologies are not highlighted in this study. We consider interpretability to be the foundation of explainability, and we use the terms interpretability, explainability, and understandability interchangeably.

The system's explanation should be human interpretable and understandable, mapping the human mental model to build trust, transparency, reliability for success and failure, and robust. Gaining meaningful knowledge and understanding of how and why the model arrived at a particular decision or outcome is crucial in model explainability, making it one of the important evaluation metrics [52].

Explainability can facilitate the understanding of various aspects of a model, leading to insights that can be utilized by various stakeholders. Data scientists can be benefited when debugging a model or when looking for ways to improve performance. And, model risk analysts can challenge the model, in order to check for robustness and approving for deployment [24]. An explainable system can make potential failures easier to detect (with the help of domain knowledge). It can help engineers pinpoint the root cause and provide a fix accordingly. Explainability does not make a model more reliable or its performance better, but it is an important part of formulation of a highly reliable system [53].

Belle & Papantonis [24] explain that through explainability developed approaches can help contribute to the following critical concerns that arise when deploying a product or taking decisions based on automated predictions. The authors list the following items:

- **Correctness:** Are we confident all and only the variables of interest contributed to our decision? Are we confident spurious patterns and correlations were eliminated in our outcome?
- **Robustness:** Are we confident that the model is not susceptible to minor perturbations, but if it is, is that justified for the outcome? In the presence of a missing or noisy data, are we confident the model does not misbehave?
- **Bias:** Are we aware of any data-specific biases that unfairly penalize

groups of individuals, and if yes, can we detect and correct them?

- **Improvement:** In what concrete way can the prediction model be improved? What effect would additional training data or an enhanced feature space have?
- **Transferability:** In what concrete way can the prediction model for one application domain be applied to another application domain? What properties of the data and model would have to be adapted for this transferability?
- **Human comprehensibility:** Are we able to explain the model's algorithmic machinery to an expert? Perhaps even a lay person? Is that a factor for deploying the model more widely?

The literature points that Explainability in deep neural nets can be introduced in three different model training and development stages: Before, during and after neural model training [25, 35, 38]. Explainability approaches applied on after training stage (defined with post-hoc or post-modelling) reflects the fact that inspect a model after the training is completed, thus they do not influence or interfere with the training process, they only audit the resulting model to assess its quality [24]. Methods applied in this stage are often data-driven or application-driven and the internal workings of a model are not illustrated, but the focus is on the intuitive presentation and exploration of model output. Machine learning visual analytics has recently emerged as one of the most intriguing areas to make models more explainable, trustworthy, and reliable. Visual analytics plays a vital role in understanding the deep neural net models through several methods proposed for dimensionality reduction, line charts, and instance-based analysis [52]. In visual analytics explainability is provided through visual representations and feature visualization approaches to support model explanation, interpretation, debugging, and improvement.

One application for visual analytics methods is the suggestion of potential directions for the model developer to explore [50]. It is essential to understand when a given instance can fail and how it fails because thereby researchers and developers can choose better directions to speed up solution improvement, quickly identify and fix problems within a model or dataset to improve overall performance. Developers can using visual analytics with tools on instance-level analysis using instances as unit tests for deep learning models testing a handful of well-known data instances to observe performance and acquiring insights to explains misclassified instances [33, 39, 40].

Image features are mathematically represented as large tensors or 2D

matrices where each row may correspond to an instance and each column a feature. The most common technique to visualize these features is performing dimensionality reduction by projecting the features onto two or three dimensions - would mean computing (x, y) or (x,y,z) coordinates - for every data instance. The features dimensionality reduction can be realized using reduction technique used, e.g., principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE)[33, 41].

Sometimes, when viewing the features of misclassified objects, they appear deceptively like true positives in the feature space, even though the classification is incorrect in the image space. In other circumstances, features of misclassified objects may appear in the feature space to be different to true positives, leading to a false negative classification. Therefore, by seeing feature spaces of all classified objects (correctly classified and misclassified objects), developers may have a better intuitive understanding of recognition algorithms.

This work developed an Explainability method after the model training stage based on Visual Analytics to provide potential insights and directions in choosing stamp types to be used in data augmentation experiments. The method is based on instance-level analysis, in which a trained model extracts the features of each instance, and the features are reduced using the PCA technique. The reduced characteristics are plotted on a graph, and the failure cases are selected to be used in our instance augmentation method to fix problems within a model or dataset to improve overall performance.

3.3. Image Processing Operations

This section presents some techniques of image processing used in our instance augmentation methodology.

3.3.1. Thresholding

Thresholding is one of the most basic forms of image segmentation. The key advantages of thresholding are its simplicity and minimal processing power requirements. It can be used to separate two regions (background and object) of an image with highly different histograms when the image contains a histogram with a bimodal distribution. The graphic shows a histogram with two peaks and one valley.

The thresholding operation can be defined mathematically as follows. A threshold value T is defined for an input image. As a result, the image will be divided into two groups: one with gray levels less than or equal to the threshold, which will receive values of 0; and another with gray levels greater than or equal to

the threshold, which will receive values of 1. According to Gonzalez & Woods [59], given an image $f(x, y)$ and the threshold T , we can obtain the thresholded image $g(x, y)$ by the Equation 6:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) < T \end{cases} \quad (6)$$

where pixels labeled 1 correspond to objects and pixels labeled 0 correspond to background and T is a predefined threshold value.

3.3.2. Mathematical Morphology

Mathematical morphology is a tool used to extract image components helpful in representing and describing the shape of a region, such as borders, skeletons, and convex closure, through pre or post-processing, such as morphological filtering and thinning, and pruning. Morphology is related to the shape, and mathematical morphology describes or analyzes the shape of a digital object, most often rasterized [60-61].

Mathematical morphology uses the geometry of small connected sets of pixels to perform tasks useful in processing regions within images. These sets called structuring elements, interact with the objects in the image, modifying their shapes. Then, this technique also is used to count, or mark connected regions in images, fill in small holes, and smooth or reduce borders.

The principles that define the nature of the transformation to which objects are submitted when applying morphological operations are shape, dimensions of the structuring element, and type of operation performed. In a binary image where the background is black, the objects consist of sets of connected white pixels, and each pixel is an element represented by its coordinates (x, y) . Then, basic set theory operations are applied between these objects, and a structuring element translates onto the image.

The simplest morphological operations are the erosion and dilation operations. In binary images, the dilatation operation images can be defined by Equation 7, in which the erosion of A by B is the set of all points z so that B , translated by z , is contained in A [59]. In other words, the erosion of A by B is then the set of all x displacements such that A overlaps with at least one non-null element, which A is the original image and B are called structuring element.

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \quad (7)$$

Let \hat{B} be the reflection of B around its origin, followed by a translation of this reflection into z . The dilation operation on binary images is defined by the

Equation 8, in which the dilation of A by B is the set of all displacements, z, of so that \hat{B} and A overlap by at least one element [59].

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \quad (8)$$

The dilation operation (Equation 7) increases the volume of objects present in the image by expanding their borders, filling gaps, or connecting objects. On the other hand, the erosion operation (Equation 8) reduces the volume of objects, being widely used to remove small objects or disconnect objects.

There are still other morphological operations, which consist of applying sequences of the morphological operators of erosion and dilation. An opening is the execution of an erosion followed by dilation using the same structuring element in an image and is defined by Equation 9. The other operation, known as closing, consists of an expansion sequence followed by erosion and is defined by Equation 10.

$$A \circ B = (A \ominus B) \oplus B \quad (9)$$

$$A \cdot B = (A \oplus B) \ominus B \quad (10)$$

3.3.3. Image Rotation

Simple methods of rotating sometimes cropped/cut sides of an image, which leads to a half image. This work uses a method to safely rotate an image without cropping/cutting sides of an image so that the entire image will include in rotation. To achieve it, the method consists of the following step-by-step:

- Firstly, get the height and width of the image.
- Locate the center of the image.
- Compute the 2D rotation matrix using Equation 11 and Equation 12.
- Extract the absolute sin and cos values from the rotation matrix.
- Calculate the new height and width of the image and update the values of the rotation matrix to ensure that there is no cropping.
- Apply the matrix rotation on the image.

The Equation 11 is defined by:

$$M = \begin{bmatrix} \alpha & \beta & (1 - \alpha)c_x - \beta c_y \\ -\beta & \alpha & \beta c_x + (1 - \alpha)c_y \end{bmatrix} \quad (11)$$

where:

$$\begin{aligned}\alpha &= scale \cdot \cos\theta \\ \beta &= scale \cdot \sin\theta\end{aligned}\tag{12}$$

and (c_x, c_y) are the coordinates along which the image is rotated or the center of image, θ is the rotation angle and *scale* is a scale factor to resize the image as well.

4. Materials and Methods

This section presents the materials used in this research. We will explain the details of the hardware, software, and technologies and present the dataset used to develop our method.

4.1. Development and Experimental Setup

With the evolution and popularization of Graphics Processing Units (GPU), they have been the leading hardware for executing deep learning-based techniques. Following this technological trend, in this research, the following hardware configuration was used for the experiments:

- Computer1

- Processor: Intel® Core™ 2 Extreme CPU X9500, 3.0Ghz;
- Graphics cards: Integrated to the motherboard;
- Persistent storage: 1000GB, 3,5" 7200 RPM, 64MB Cache, SATA III;
- Volatile storage: 4GB RAM;
- Operational system: Windows 10 Enterprise x64.

- Computer2

- Processor: Intel(R) Xeon(R) CPU @ 2.20GHz, 3.0Ghz;
- Graphics cards: NVIDIA® Tesla P100® 16GB, NVIDIA® Tesla® K80 12 GB, NVIDIA® Tesla® P100 12GB;
- Volatile storage: 13GB RAM;
- Operational system: Ubuntu 18.04.3 LTS.

The Computer1 was used to develop the image labeler software, dataset building, and evaluation of the results. The Computer was used on Google Colaboratory, more commonly referred to as “Colab”. Colab is a research project for prototyping machine learning models on powerful hardware options such as GPUs. It provides a serverless Jupyter notebook environment for interactive development. Colab is free to use like other G Suite products [84]. We used Colab to execute the network (training, evaluation, and test steps).

In Computer1, we use MATLAB [85] along with its Computer Vision System Toolbox for the segmentation software development, dataset segmentation,

data preparation, and evaluation of the results. This toolbox provides algorithms and applications for designing and simulating computer vision systems and image and video processing.

In Computer2, the network was executed using the Facebook AI Research's (FAIR) Detectron2 [86]. Detectron2 is a flexible and extensible platform implemented in Pytorch [87] and available under the Apache 2.0 license. Detectron2 provides fast training on single or multiple GPU servers and includes high-quality implementations of state-of-the-art object detection algorithms. PyTorch is an open-source machine learning framework that allows researchers and practitioners to iterate rapidly on model design and experiments. It shows that PyTorch has been one of the most widely used frameworks in academia and that its use has seen a marked growth in its use in recent years.

4.2. Software for Annotation Process

The annotation process of the objects of interest is performed manually. A system for 2D image labeling was developed in this work using a Matlab tool. The software developed was used in the dataset annotation process. On the main screen (Figure 10), the user can perform seven main actions. They are:

G/EASA	FAA	AD	RT/TA	OGAC/EASA	FAA	AD	R
/	/	/	/	/	/	/	/
/	/	/	/	/	/	/	/
/	/	/	/	/	/	/	/
/	/	/	/	/	/	/	/
/	/	/	/	/	/	/	/

C D'ESSAI MAIS NON PRIS EN COMPTE DANS L'ETAT DE DISPONIBILITE ACCORD AVEC LE CCT 6100 ou RTC.
 It not count up into availability status, in accordance with CCT 6100 or RTC.

HEURES	CYCLES	GN	GG	TL	FT

Date - Signature - Tampon / Stamp	Date - Signature - Tampon / Stamp
07 AUG 2017	07 AUG 2017

ENR 0696 E

Search: c3

Save edit

☒ Edit annotation?

Figure 10: The main screen of the software developed for dataset annotation.

- Open image: when selecting the option to open an image, the user selects a document page in .jpg format to which he wants to segment the objects of interest.
- Select object type: through this action, the user indicates which object type will be segmented: stamp or signature.
- Enable Region Selection: The user will move or resize the region selection rectangle when performing this action. The enclosed area corresponds to the targeting area.
- Enable zoom/translation: The user will be allowed to perform scale and translation transformations on the image when selecting this option. This option is helpful to assist in object segmentation accuracy.
- Save Object: When selecting this option, the coordinates of the selected region and the selected object type are saved in json format in a file in the same folder on the page. The file name will be the same as the page name. If the file does not already exist, the software creates it. If it already exists, the software adds the new data to the file. In addition, a message indicating the saved data and file folder is displayed to the user on a message screen.
- Perform a search of all pages that contain a specific stamp
- Make changes to annotations. Users can correct labeling or readjust segmentation windows.

Once the stamp location is indicated, memorizing, or even typing the stamp type is impractical due to the number of types within the dataset. A screen for selecting the stamp type was developed, where the user visually indicates the type of stamp annotated in a kind of image gallery. The software highlights the most frequently annotated types to facilitate the user's visual search of the type.. In addition, the user can compare the stamp located on the page and the type of stamp selected in the labeling software. Figure 11 illustrates the scenario described. It is easy to notice that some stamps are demarcated in blue and have thicker demarcation than others. The thickest demarcation is the stamp "c3" (HEB 136).

The software has a third screen: the search screen. The primary function of the search screen is to reuse the knowledge gained in previous moments in the base annotation process. This screen was widely used in the process of data cleaning and marking correction. Figure 12 illustrates the screen. In (1), the user selects which

subset of the database he wants to view. Region (2) presents the page's index, and the total of pages containing the object searched within a database subset.

Region (3) presents the visualization window and interaction with the selected page. Region (4) presents the category and subcategory of an object selected by the user. Finally, regions (5) and (6) are for navigating through the pages contained in a selected database and displaying the name of the subcategory of the closest object searched.

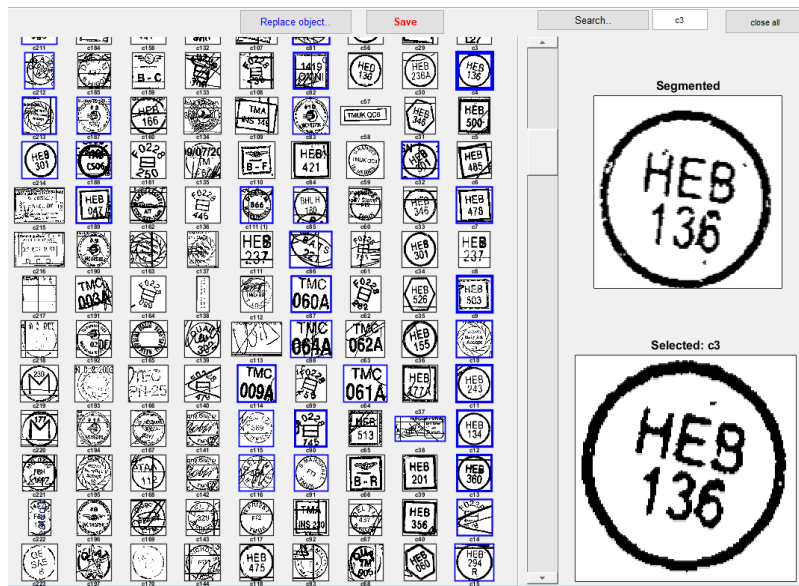


Figure 11: Illustration of the most frequent stamp types highlighted in screen 2 (on the right). The stamp segmented and the stamp selected in the gallery are shown for visual comparison before saving the data (on the screen left).

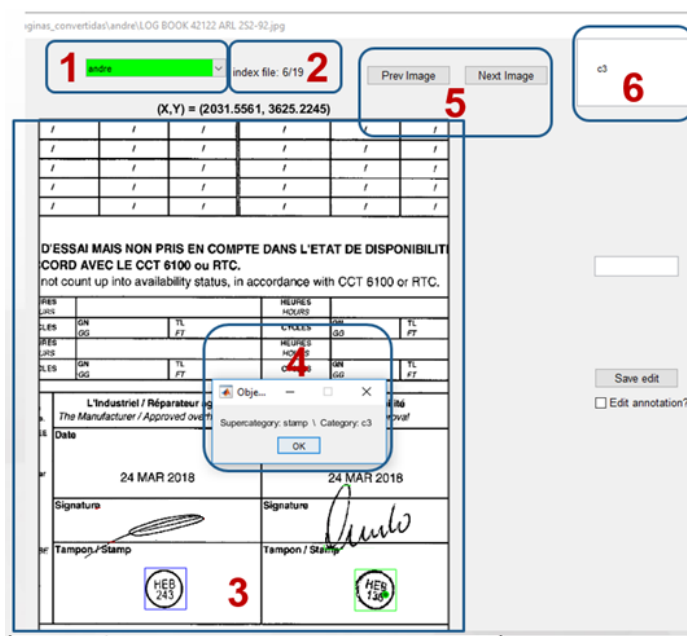


Figure 12: Search screen illustration.

4.3. Method Proposed for Stamp Detection

4.3.1. Image Acquisition

We used documents provided by a private multinational manufacturing company. The dataset has 33 scanned documents in PDF format that together contain a total of 2230 pages. We divide each document and convert its pages to image files. The final configuration chosen was 300 DPI in RGB color and jpg format. Therefore, each page has dimensions 2438 x 3542 on portrait orientation and 3542 x 2438 on landscape orientation, width, and height, respectively, and 24 bits depth.

Although the dataset contains structured pages, most of the pages are semi-structured. Structured data has a regular structure and semi-structured data arises when the source does not impose a rigid structure [61]. However, in practice, stamps are often applied with some randomness, even in cases where there is a specific region. Our dataset contains 1880 stamps distributed on 251 classes. Page images usually have more than one instance of stamps (multiplicity at the instance level). Besides, pages with more than one example contain multiple stamp types (multiplicity at the type of level). About 8 people participated in the annotation process using software developed in this proposal. Each person is responsible for a different portion of the dataset, and, at end of process, 1 additional people performed a final evaluation on all annotations.

4.3.2. Preprocessing and Splitting

Data collection and annotation procedures may generate many object detection problems because they are stressful and error susceptible tasks. The literature points that missing, incorrect, and duplicate values directly affect the performance of several machine learning algorithms. Thus, data cleaning is a necessary, labor-intensive, and time-consuming procedure [63-64].

This method follows a dataset preparation pipeline to provide detailed inspection, cleaning, and validation of the data. This pipeline contains three main phases and nine stages. Figure 13 illustrates the steps flowchart.

The first phase consists of class cleaning (steps (1) to (4) in Figure 13), which aims to select duplicate and "fake classes". First, we manually cluster the stamps from the shape and then map the codes within the clusters. Then we perform a new clustering manually - this time from stamps checker code. Next, we remove the annotations and classes corresponding to duplicates and fake classes identified. These two clusters are necessary since stamps in our dataset can have, at the same time, both the same distinct code and shape as well as specific codes and identical

shapes. It is, one stamp type is identified by both shape and code.

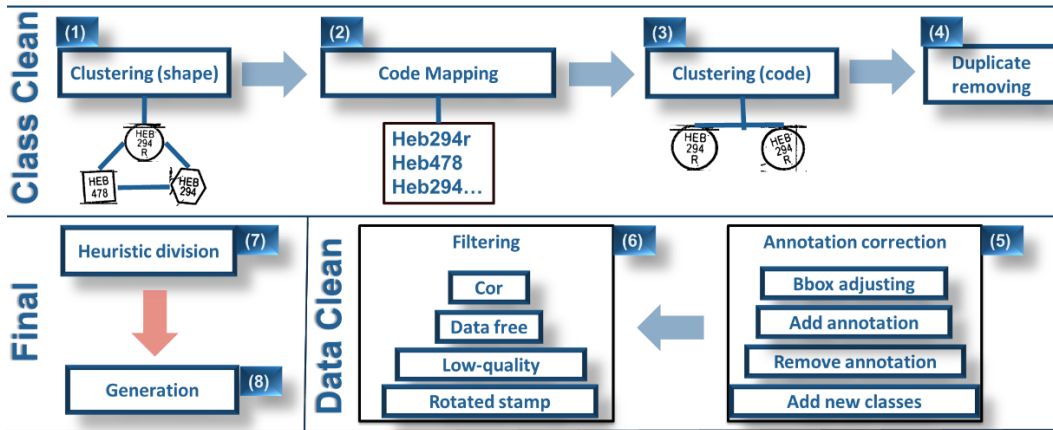


Figure 13: Flowchart of each main phases and its respective steps of the data preparation.

On data cleaning (steps (5) to (6) in Figure 13), we perform the annotations correction, which consists of adjusting the annotation bounding box, observing not annotated stamps or classes not identifying, and removing wrong annotations. We filter color pages, "free" of objects of interest, extreme low-quality pages, and rotated pages or with rotated stamps (stamps with angles greater than 0 degrees). Rotated pages are adjusted according to the complementary rotation angle (i.e., 90, 180, or 270 degrees).

The final phase (steps 8-9 in Figure 13) consists in splitting the dataset into the training, validation, and testing sets. We designed a greedy algorithm to split the dataset to overcome these difficulties and used it in other tasks that present similar problems. The algorithm has three main goals. The first objective is to maximize the number of instances in the training group since the training group should have the most examples. The second objective is maximizing the diversity of stamp types in the test group. We desire to observe the capacity of the proposed method to generalize the detection of stamps for types and situations not yet seen by the network, a scenario certainly expected in real cases. Finally, we want the validation group to have the diversity of stamp types as close as possible to the test group, ensuring that both groups are close concerning diversity.

However, achieving these three goals simultaneously is not a simple task, and dividing the pages randomly is not the most suitable option given the stamps multiplicity in type and instance per page, imbalance, and lack of data. These facts make the process of dividing the data set into a combinatorial problem with infinite possibilities. Our algorithm overcomes these difficulties and guarantees the three desirable goals. Besides, we ensure that the test group covers all superficial characteristics such as shape, size, color, similarity, location, and multiplicity. We

maintain the imbalance in the training set since one of the objectives of this study is to observe how our method behaves, considering this complexity.

In summary, the algorithm works as follows. We first work on the pages with more instances and different types of stamps, placing them in the test, validation, and training groups, respectively. We keep a record of the classes already inserted in each group, and we use intersection operation to check if any stamp types belonging to the current page have been added or not in the groups. Thus, we guarantee the highest number of the kinds for the two first groups, obeying the order of priority. After filling the first two groups, we insert the remaining pages in the training group. Our dataset splitting algorithm is present in Algorithm I.

Algorithm 1: Heuristic dataset division algorithm proposed.

Result: Dataset split method for train, validation and test sets.

Sort pages in ascending order by number of annotations and number of classes, respectively;

```

while While page in reversed_order(list_pages) do
    instructions;
    if not(page.classes.intersection(test_group.classes)) then
        test_group.add_page(page);
        test_group.classes.union(page.classes);
    else
        if not(page.classes.intersection(validation_group.classes)) then
            validation_group.add_page(page);
            validation_group.classes.union(page.classes);
        else
            page.classes.intersection(train_group.classes);
            train_group.add_page(page);
            train_group.classes.union(page.classes);
        end
    end
end
end

```

4.3.3. Network Design

Testing network architectures is a time-consuming step. We follow a procedure to optimize the network architecture design based on the characteristics of the dataset used. The data characteristics refer to the qualitative property-data resolution, diversity of examples of the same class, well-defined acquisition protocol, occluding, noisy, and missing values/parts. In the following sections, we present our method for qualitative analysis, which is possible to apply to any object detection task and the projected architecture based on the proposed analysis.

4.3.3.1. Qualitative Analysis

Qualitative analysis has two steps: simple and complex features analysis. In this step, we verify and describe the data properties and associate them with the

potential of the deep learning resources to direct the network architecture choice. In other words, we searched the literature for the most suitable techniques to deal with each challenging characteristic of our dataset. We propose an architecture to contemplate all the challenge from the solutions found.

Figure 14 summarizes our proposed method, while the following sections demonstrate its application to our dataset. The challenging characteristic of our dataset and its respective suitable techniques are listed below:

1. **Color.** We consider color pages the pages without white background. We consider color stamps those with color edges. Stamps are poor in texture, limiting colors to their contours and/or page color. As shown in Figure 14 (1), stamps and pages can appear in any combination of shades of gray and color. The literature shows that transfer learning handles with it [64].
2. **Shape** is a low-level feature almost as simple as edges. A stamp can contain at least two of a simple shape (i.e., circle, triangle, square), a figure, or a string of characters (see Figure 14 (2)). We use transfer learning by freezing the initial layers of a pre-trained CNN as a proper low-level feature extractor. To use DCNs also handles well with it [12].

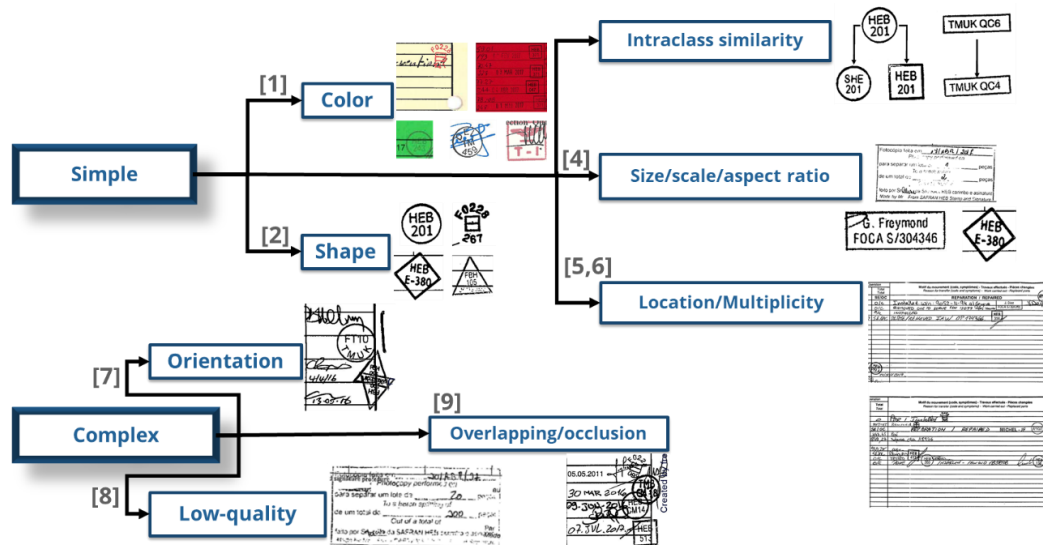


Figure 14: Method proposed for dataset analysis. The qualitative analysis step serves as a guide for choosing network architecture.

3. **Intra-class similarity.** Stamps can have identical shapes but only a similar string of characters. The shape corresponds to different roles, while the string distinguishes between unique individuals (see Figure 14 (3)).
4. **Size/scale/aspect ratio.** Since the acquisition of documents does not vary

in distance, zoom, or point of view, these characteristics are invariant between stamps of the same class. However, size can vary between different classes of stamps, as illustrated in Figure 14 (4). Most stamp classes occupy regions of approximately 100x100 pixels and aspect ratio 1, reaching values of approximately 360x170 pixels and aspect ratio more than 2. Literature shows that FPN and anchor-based deep frameworks can successfully handle objects of different sizes, scales, and aspect ratios [12, 65].

5. **Location.** The large diversity of page templates and the fact that people do not always stamp in the expected field make it impractical to rely on consistent stamp locations. A stamp can appear in any position on the page. For example, Figure 14 (5) illustrates two pages with the same template but stamped in different ways. Both RPN-based and Classification/Regression-based frameworks handle location [9, 65].
6. **Multiplicity.** It is not known a priori which or how many pages have stamps. When stamps occur, a priori both the amount is unknown and which classes exist. Figure 14 (6) illustrates a multiplicity case, where two pages with the same template present different amounts of stamps. Again, RPN-based frameworks can adequately handle this feature [9, 65].
7. **Overlap / occlusion.** The literature highlights it as a recurring problem that increases detection complexity [9, 12, 54, 65]. A stamp can be isolated, overlapping another stamp, or overlapping other structures (i.e., text blocks, general text, manuscripts), as illustrated in Figure 14 (7). DCNs handles well with it [8, 11].
8. **Orientation.** Peoples usually do not pay attention to orientation when stamping a document. However, it is reasonable to expect that stamps are not upside down, that is, at an angle between 90 and 270 degrees. Other angles may occur, but only rarely. Figure 14 (8) illustrates rotated examples of triangle stamps.
9. **Low-quality.** Image degradation in visual recognition tasks is a well-known problem in the literature [12, 66-67]. Stamps in this condition can have missing parts due to reduced acquisition quality, noise, or several overlaps. These problems cause intra-dissimilarity between examples of the same class and can lead to the miss classification. We characterize as

noise any element on an object stamp that does not belong to its body. Figure 14 (9) illustrates cases of low-quality stamps which have overlapping and missing parts.

4.3.3.2. Network Architecture

As a result of the analysis of the qualitative characteristics of the dataset, a deep neural network architecture never applied to the stamp problem is proposed in this work. Figure 15 illustrates the proposed architecture.

As opposed to using common CNNs, working with deep frameworks for object detection brings the advantage of not working with sliding windows manually. Therefore, firstly our network receives as input the entire image. Then, a CNN backbone performs hierarchical features extraction. Our backbone consists of a pre-trained Resnet model on the COCO dataset to use all benefits of knowledge transfer (Section 3.2.6). Residual connections from Resnets are used for addressing vanishing/exploding gradients in profound models [41].

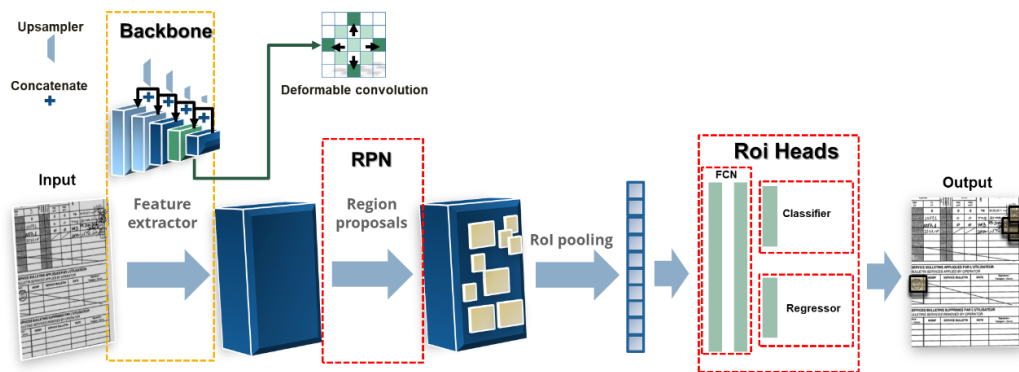


Figure 15: Proposed network architecture designed using our qualitative analysis.

The two first modules from Resnet are freezing to address qualitative problem 2 (shape), and the others are fine-tuned for the extractor parameters to better adapt to the problem we propose to solve. We use deformable convolution on the 4th module to address qualitative problems 2 and 7 (shape, occlusion/overlapping). Then, the hierarchical features extracted from all modules are combined using FPN to get richer semantics by building multi-scale features at various semantic levels - and handling quality problem 4 (dimensions problems).

After the process performed by the backbone, the RPN uses the feature map extracted from the entire image and automatically learns to propose regions and dimensions of promising to object bounding boxes, handling with quality problems 4, 5 and 6 (dimensions, location, and multiplicity). The proposed regions are dimensioned using ROI pooling, which solves the problem of fixed image size

requirements for the FC module. After passing them through two FC layers, the features are fed into the sibling classification and regression branches. The classification branch calculates the probability of a proposal belonging to the stamp type, and the regression layer coefficients are used to improve the predicted bounding boxes. Section 4.3.5 provides information about the experiments and the reason for the specific configuration of the backbone used.

4.3.4. Evaluation

We evaluated the results achieved by the proposed method using several metrics commonly employed in the literature related to stamp detection and object detection tasks. These metrics aim to measure the performance and robustness of the proposed architecture as satisfactory or not, in addition to helping to identify positive and negative points for future improvements of this work in the training, validation, and testing phases.

The metrics use the concepts of true positives (TP - stamps correctly detected), false positives (FP – background incorrectly detected as a stamp), and false negatives (FN – stamps not detected). It is important to note that a true negative (TN) result does not apply in the object detection context, as there is an infinite number of bounding boxes that should not be detected within any given image [68].

- **Score.** Value in the range [0,1] obtained by the classification branch of the network, showing the probability of an object belonging to a particular type.
- **IoU.** Value representing the area of overlap (intersection) between the bounding box B_p provided by the network and the ground-truth bounding box B_{gt} , divided by the union area.

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (13)$$

- Precision is expressed as the number of TP divided by the total number of predicted cases (TP and FP).

$$Precision = \frac{TP}{TP + FP} \quad (14)$$

- Recall is expressed as the number of TP divided by the number of positive cases (TP and FN)

$$Recall = \frac{TP}{TP + FN} \quad (15)$$

- F-score is the harmonic mean of accuracy and recall

$$F - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (16)$$

- Average Precision (AP) is the most commonly used metric on object detection tasks [11]. Fixed a single IOU threshold; it summarizes the precision x recall curve obtained from analyzing different score values into a single value. The AP is obtained by interpolating the precision at each level, taking the maximum precision whose recall value is greater or equal than R_{n+1} [68]. Mathematically, we have:

$$AP = \sum_n (R_{n+1} - R_n) P_{inter}(R_{n+1}) \quad (17)$$

where,

$$P_{inter}(R_{n+1}) = \max_{R: R \geq R_{n+1}} P(R) \quad (18)$$

- Average Recall (AR) is the maximum recall given a fixed number of detections per image, averaged over all categories and IOU thresholds [68].
- AF-score is a custom metric of this work, evaluate AP and AR equally f-score evaluate precision and recall.

4.3.5. Experiments

Hyperparameters settings control the behavior of the learning algorithm, but they are not adapted automatically. Adjusting hyperparameters on the training set is not desirable because the learning process always chooses the maximum possible model capacity, resulting in overfitting. Additionally, tuning hyperparameters model several times based on performance's model test set can quickly result in overfitting to this set, even though the model is never directly trained on it. This phenomenon is known as information leaks because when the hyperparameters are chosen based on the model's performance on the evaluation set, some information about the evaluation data leaks into the model [69].

Simple hold-out validation solves this problem, which a portion of the training set is separated for model evaluation on the hyperparameters adjustment process. This new division must adhere to the same guidelines as the previous one. Thus, evaluating a model using this strategy divides the available data into three sets: training, validation, and test. Each of the three sets must be chosen independently: The validation set must be different from the training set to obtain good performance in the optimization stage and, and the test set must be different from both to obtain a reliable estimate of the valid error rate [62]. Once the model is ready, it is evaluated one final time on the test data [69].

Supervised training was conducted using mini-batch stochastic gradient descent (MBSGD) and standard backpropagation algorithm to determine the global minimum. We used as regularization methods the mini-batch stochastic gradient descent (MBSGD), batch normalization and weight decay, techniques widely discussed in Neapolitan & Jiang [32]. We fixed the designed architecture network for all experiments. In applying the framework, we select the default configuration as the start point and conduct several experiments varying the parameters. The best results were achieved using the values shown in Table 1.

The hyperparameters values are selected for testing using grid search [70], the most well-known technique in which a set of values is selected for each hyperparameter. In the most straightforward implementation of the grid search, all combinations of selected values of the hyperparameters are tested to determine the best choice [32]. For reducing the number of tests, the parameters are tested one to one: when testing a value for a parameter, the result gets worse, the previous value is fixed, and a new parameter is testing. We search a balance between bias (underfitting) and variance (overfitting). The model with the optimal predictive capability is the one that leads to the best balance between bias and variance, which gives the smallest average training and generalization error at the same time [42–44, 71]. We use simple hold-out validation [69].

Table 1: Main parameters tested in the experiments.

Parameters	Values
Minimum size image train	1200
Minimum size image test	1200
Learning rate	0.01
Momentum	0.9
Weight decay	0.3
Freeze backbone stages	1,2
Deformed backbone stages	4
Mini-Batch	512

We control the weights update step experimenting learning rate on values 0.1, 0.01 and 0.001, values suggested in Andrew [88]. For weight decay technique, we use the values 0.1, 0.3 and 0.5. Weight decay acts as a capacity hyperparameter: increasing this parameters decreasing the complexity of the model leading to a simpler model and preventing overfitting [32]. We experiment momentum values 0.9 and 0.999 for penalizes useless “sideways” oscillations (steep steps). The literature point that this allows the use of more significant steps in the correct direction without causing overflows or “explosions” in the lateral direction, resulting in an accelerated learning process [32].

We used batch normalization technique to address the vanishing and exploding gradient problems and reduce covariate shifts. In covariate shift, the parameters change of the hidden inputs change during training from early layers to last layers, and it causes slower convergence during training because the training data for later layers are not stable. Batch Normalization adaptively normalizes data even as the mean and variance change over time during training. It works by internally maintaining an exponential moving average of the batch-wise mean and variance of the data seen during training and allows dropout technique to be omitted [42, 69-70]. The batch normalization layers are tested frozen, fine-tuning from pre-trained model, and full training all of them.

The literature points that the common values used in MBSGD technique are powers of 2 as the size of the mini-batch, because this choice often provides the best efficiency on most hardware architectures [32]. So, we experiment values in the range of 32 to 512. For stages using DCN and transfer learning, we experiment with all possibilities. We experiment with the interval of 2 to 4 stages for DCN and 1 to 5 stages for transfer learning. The experiments are conducted individually, first the transfer learning stages for finding the best freezing layers configuration. Then, the DCN stages are tested. To conduct the experiments, firstly, using DCN would combinatorically increase the number of tests. For each one, the stages are testing one to one and combining them.

We use 10k epochs on all experiments rather than using early stopping because the literature points that in this technique the iterative optimization method is terminated early without converging to the optimal solution on the training data [32]. Another disadvantage of the early stopping technique is that the number of epochs is more of a parameter to be adjusted. In the first stage of our analysis, we empirically observed that the models did not achieve significant improvements in the results in around 10k (a large number) of epochs.

We observe AP and AR metrics to define the best network settings and hyperparameters value using hold-out evaluation. Analyzing the experiments evaluation process using graphics generated by Tensorboard, we observe a

significant interval of epochs for network converging and several peaks configurations (Figure 16(a) and 16(b)).

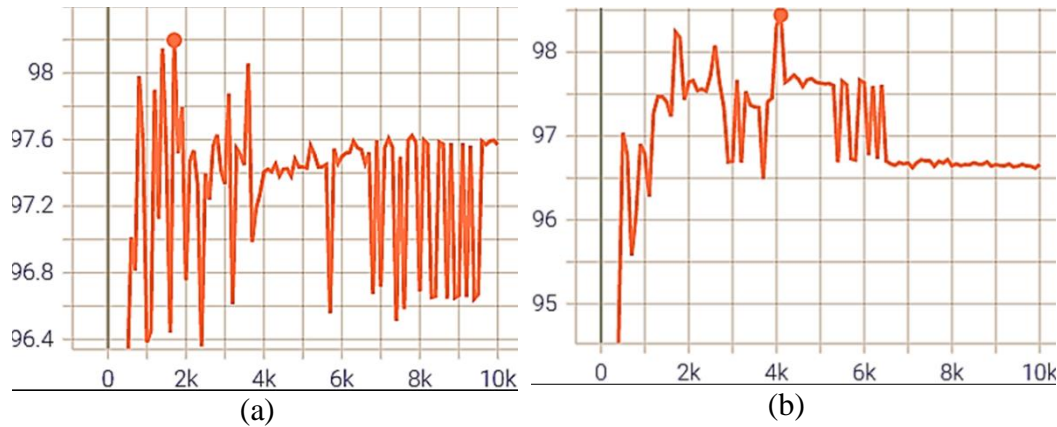


Figure 16: Graphic produced on the Tensorboard for the AP50 metric obtained from the application of the network on validation set. (a) Experiment shows best value on epoch 1800. (b). Experiment shows best value on epoch 4100.

For tiebreaker criteria, we evaluate the AF-score evaluation metric, and we choose a single model. We consider all object sizes and 100 detections per image. We base our choices on the following points:

- The pages have stamps in several sizes.
- The same page can contain more than ten stamps.
- A minimum of 50% IOU is sufficient for the stamp detection problem.

Then, we apply the model again on the validation group, and we evaluate the predictions obtained using precision and recall metrics considering different values for the score and IoU. In this stage, we choose the two best possible configurations concerning each of the metrics.

The final evaluation for measuring the network performance is conducted on possible fixed thresholds as default in a decision-making situation. We experiment all combinations considering the values 0.5, 0.7 and 0.9 for score and 0.05, 0.5 and 0.7 for IoU. We selected two (score, IoU) values: one prioritizing correct inference and one prioritizing detected stamps even with a higher false-positive index. In the end, we apply to the test based on both the model and the thresholds selected. The following section presents and discusses the results obtained from the experiments.

4.4. Method Proposed for Stamp Instance Augmentation

One of the most significant drawbacks of using a state-of-the-art detection

system is the number of annotations needed to train it because finding a large labeled dataset containing instances in a particular task is often unlikely. One of the ways found in the literature to overcome this bias is using data augmentation techniques to generate synthetic data to feed the training of neural networks. However, models trained with this synthetic data have difficulty generalizing to real data due to changing image statistics [21].

A more challenging situation is when there are too few certain classes in the training base. In this case, data augmentation can become ineffective due to the lack of representation of these classes. A simple way to significantly improve the data efficiency of object detection is using an augmentation procedure that is more object-aware, both in terms of category and shape.

Instance augmentation is a form of synthetic data generation based on data augmentation by generating objects of interest instead of complete images. This method has several advantages. It combines information from multiple images in an object-aware manner by copying objects from one image and pasting them onto another image. This method can lead to a combinatorial number of new training data, with multiple possibilities for:

- Choices of the pair of source images from which instances are copied, and the target image on which they are pasted;
- Choices of object instances to copy from the source image;
- Choices of where to paste the copied instances on the target image.

The large variety of options when utilizing this data augmentation method, the large variety of options allow for lots of exploration on how to use the technique most effectively. Instance augmentation has the potential to create challenging and novel training data for free.

The critical insight for using instance augmentation is that state-of-the-art detection methods-based regions, like Faster-RCNN, care more about local region-based features for detection than the global scene layout. For example, a stamp detector cares about the stamp's visual appearance and blending with the background, not where the stamp occurs on the page. [22] shows that while global consistency is essential, only ensuring patch-level realism while composing synthetic datasets should go long to train these detectors. They use the term patch-level realism to observe that the bounding box containing the pasted object looks realistic to the human eye. In this section, this work proposes a instance augmentation by only copying the exact pixels corresponding to an object instead of all pixels in the object's bounding box to ensure the patch-level realism of the object.

The proposed approach for generating new data using Instance

Augmentation is very simple. Instances of the classes to be augmented are manually segmented from their original pages. Then, binary masks of the selected instances are generated using image processing techniques thresholding and image negation. Then instances and masks transform based on the addition of noise, rotation, or morphological operations in some combination. Finally, the object's pixels are pasted into the image using the binary mask as a reference. Figure 17 illustrates the steps described here. The following section details the procedure of the proposed method for Stamps Instance Augmentation.

4.4.1. Instance Augmentation Procedure

After applying the stamp detection method on the images of document pages, we get all the detected stamps. We get the undetected stamps by comparing the network responses to dataset manual annotations. Thus, the first step of the proposed synthetic data generation procedure is to observe undetected stamp types containing few samples in the training set. Types with few training samples are more difficult for the network to detect due to the lack of representation.

To identify the types of stamps with few samples, the model obtained through the stamp detection method analyzes the document pages and extracts the characteristic maps them. Then, the regions corresponding to each stamp are extracted, identifying those detected and those not detected through the combination of the obtained characteristic maps and the manual markings.

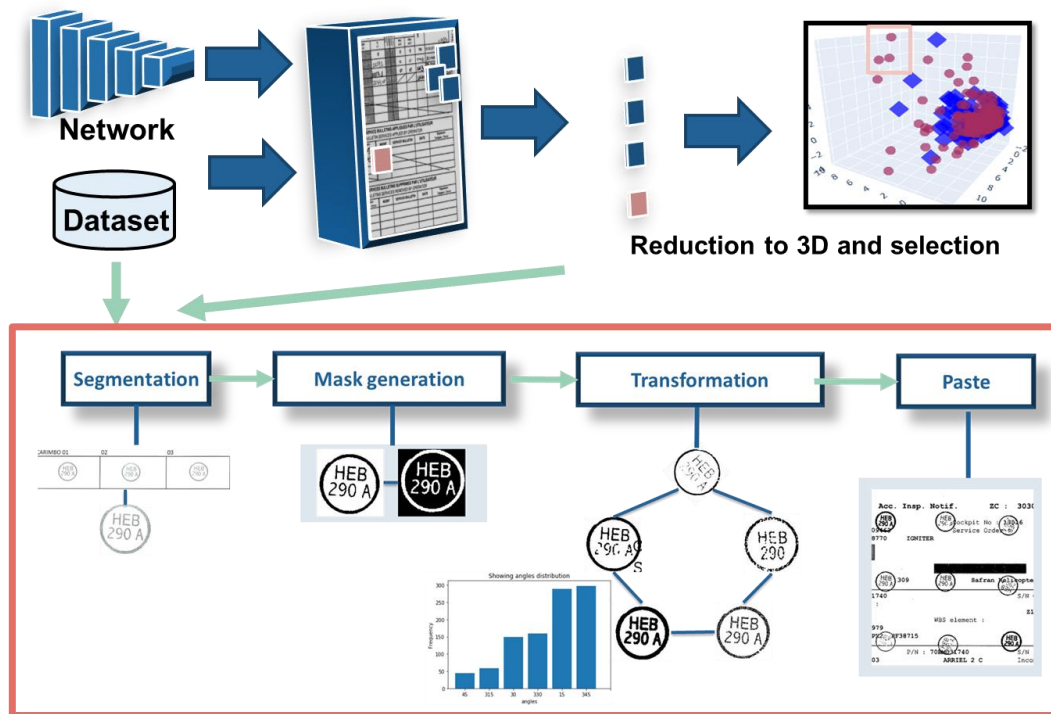


Figure 17: Proposed method for Stamps Instance Augmentation.

After obtaining characteristics of the detected and undetected stamps, the PCA [72-73] technique reduces the dimension of the characteristics to the three-dimensional space. The reduced instances feature of all stamps are plotted to perform a visual analysis of the stamps. Undetected stamps are identified as more isolated in the plot, representing the most difficult stamps for the obtained model to detect.

Once the most challenging types are selected, the generation process starts. First, some cases of stamps from the selected types are extracted based on manual annotations. Once segmented, noises are manually removed (structures that do not belong to the stamp, such as table lines, parts of other stamps, and handwritten words). This cleaning is necessary since when performing the gluing process, the noises not removed will join with the noises already present in the region to be glued and may mischaracterize the new stamp generated.

After extracting the stamp instances, the masks are generated using image processing techniques such as thresholding and image negation. Then, a lottery decides which perturbations or combinations of perturbations will be applied to generate each new stamp. These disturbances are performed to add statistical variability in the synthetic instances. The method generates datasets for every possible combination of transformations, that is, all combinations involving the addition of Gaussian noise, rotation, erosion, and dilation.

The method applies Gaussian noise with a mean of 0.1 and a variance of 0.3. The masks used for the morphological transformations were 3x3 masks, using only one interaction on the instances. These values were established empirically by observing the generation of several datasets visually. The rotations were established for the angles [345, 350, 355, 5, 20, 25] and with probabilities of $p = [0.05, 0.15, 0.30, 0.30, 0.15, 0.05]$ for each angle, respectively. This probability setting prioritizes angles close to the object's natural angle (angle 0) since larger rotation angles are less likely to occur in documents based on observations made in the dataset. Figure 18 illustrates the angle draw distribution for 1000 trials.

The data generation process for each type of stamp follows the following step-by-step:

- For each type of stamp
 - For each page image
 - Draw a stamp instance
 - Draw the probability of transformations taking place. If it is favorable, while it is favorable
 - Apply a transformation from the list of transformations

- Divide the chance of a new transformation happening by 2.
- Remove the applied transformation from the current list

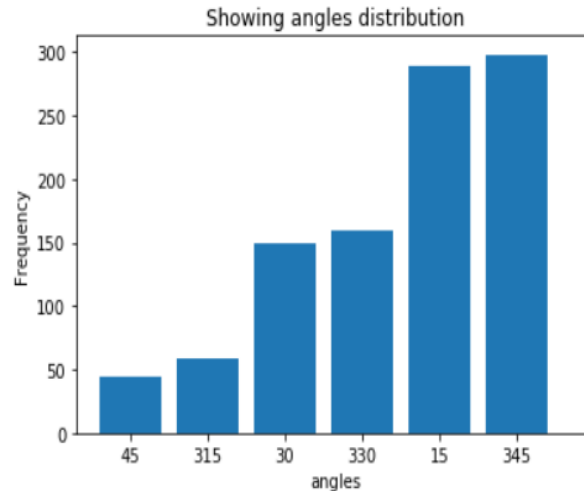


Figure 18: Illustration of the distribution of the draw of angles for 1000 attempts.

Finally, the process of pasting the instances is performed. Each page is gridded to have 12 stamps. We only use stamp-free pages to ensure full control over data generation. The AND binary operation is applied between the instance and the mask to extract only the stamp pixels. The method extracts a region of interest (ROI) – (grid cell) contained in the page, and then it applies an AND binary operation using the ROI and the negative of the mask to extract the background pixels. Then, it adds the pixels extracted from the ROI to the pixels extracted from the stamp. Finally, it pastes the resulting image into the cell corresponding to the ROI. The method repeats this process until it fills the page.

4.4.2. Experiments

The results are analyzed by applying the model to the validation base and 3D visualization. The three most isolated points in the generated graph and one of the random undetected instances were selected. More isolated points on the graph represent the most difficult instances for the network to detect. Figure 19 illustrates the scenario discussed. The points highlighted in yellow represent the stamp instances selected for studies and experiments using the proposed Instance Augmentation method.

We evaluated different scenarios, as several stamp quantities per page, page quantities, transformation combinations, and stamp locations. Experiments were also performed cleaning the instances, not cleaning the instances, and combining them. The experiments were controlled, adding the generation of only one stamp

type at a time. In other words, given stamps A, B, and C, experiments are first conducted until stamp A is successfully detected. After that, stamp B is added. This process is repeated until the synthetic base contains all the selected stamp types.

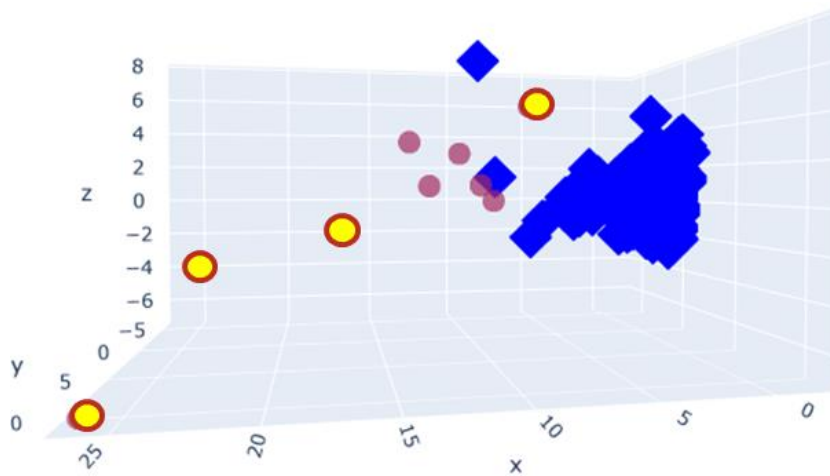


Figure 19: 3D visualization illustrating the four instances selected (on yellow color) for the Instance Augmentation experiments.

We observed two issues in our experiments. The first question is whether the generated instances helped the network to detect similar instances in the validation base and how much improvement there was in the detection. We assess this “how much” by assessing whether the network provided a higher score. The score represents the percentage of how much such an instance is a stamp for the network. The second point is whether there was an improvement in the values of the evaluation metrics used. Figure 20 illustrates the results for the AP50 metric in the 1699 epoch based on validation for 7 experiments. Each of the 7 illustrated experiments has different Instance Augmentation settings. Curves have been smoothed at a rate of 50% for better visualization. The experiment with name 1 achieved the best AP50 value for this epoch.

5. Results

5.1. Stamp Detection

This section presents and discusses the results obtained with the proposed method regarding the detection of stamps.

After the data acquisition, annotation, and pre-processing steps, we apply the algorithm proposed in Section 4.3.2 to perform the dataset division. In the training set, we use 87 types of stamps distributed on 1173 instances. In the validation set, we use 130 types of stamps distributed on 234 instances. Finally, in the test set, we use 251 types of stamps distributed on 473 instances of stamps. The proportion of stamp types among sets is about 1/1.5/3. The proportion of stamp instances among sets is about 5/1/2. In the training set, we used the number of stamp types 3x more than the test set. Table 2 shows the final division and proportions of the dataset.

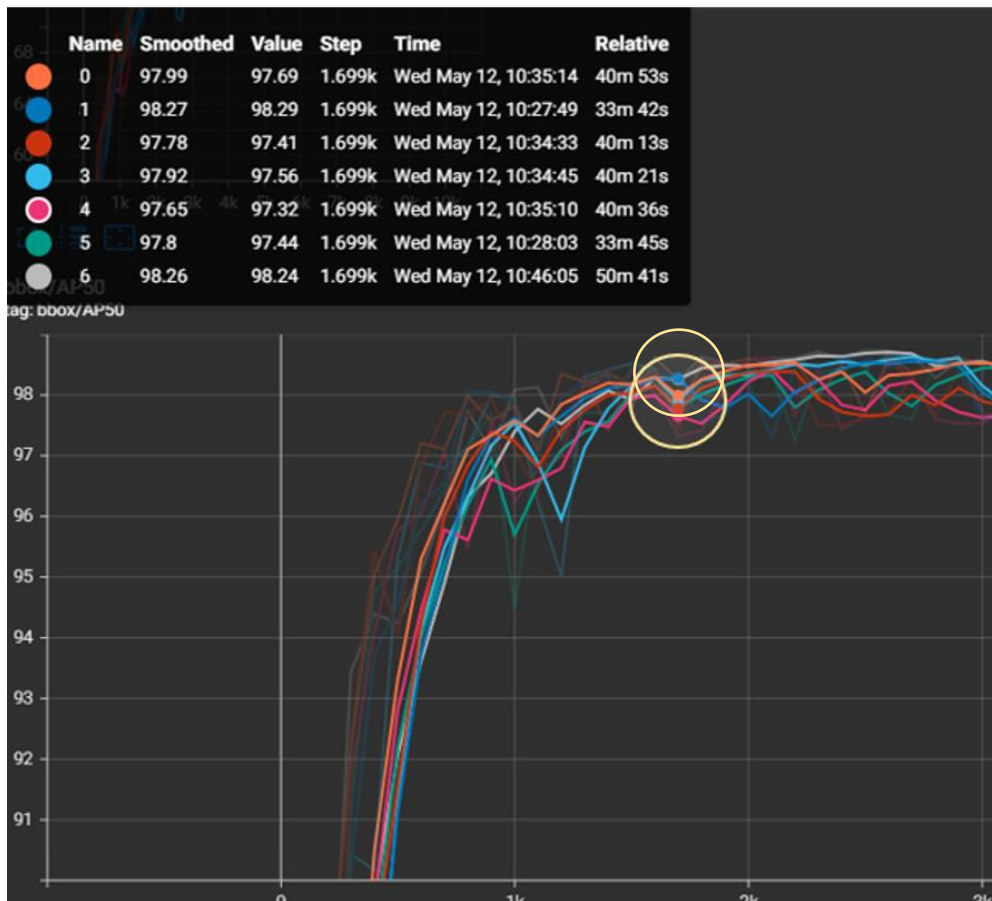


Figure 20: Results for the AP50 metric in the 1699 epoch based on validation for 7 experiments with several Instance Augmentation settings. The epoch is highlighted in the chart.

Table 2: Distribution of training, validation, and test sets.

	Total	Train	Validation	Test
Types	251	87	130	251
Instances	1880	1173	234	473

In the best benchmark, we reduce the input images to approximately three times smaller than the original size, keep the batch normalization layers frozen, and use convolutional deformable layers only in stage 3 of the resnet. We apply transfer learning using the pre-trained weights on the COCO dataset in all backbone layers. We freeze the first two stages of the backbone and fine-tune the remaining parameters. Early stopping is not the most suitable option for stopping criteria in our experiments. The experiments also noticeably showed that the proposed architecture has good generalization capacity and excellent performance concerning overfitting. The cost of training time per experiment is about six hours using Tesla K80 graphics card and three hours using Tesla P100 graphics card.

Figure 21 illustrates the loss curves about the training base and the validation base for six different experiments using the proposed architecture. It is possible to observe that the loss on the validation base decays until a point and after it stabilizes. The curvature of validation has fewer oscillations because it is calculated over fewer frequently than the loss on the training database. This strategy allows us to study the behavior in more epochs of training and perform a greater number of experiments since training takes much time.

Table 3 presents the best results obtained from the validation and test sets using COCO evaluation metrics [11] on network training time. We show the number of classes, the number of stamps, the AP50, the AR, and the AF-score. In general, Table 3 shows that the network achieves promisors AP50 and AR. What explains the bias between the two measures is that the AR measure analyzes results considering more IoU proportions than the AP50 measure. This fact is also one of the reasons for we chose to analyze the model using the metrics precision and recall considering fixed values of IoU and Score.

We perform this last evaluation through different combinations of thresholds for the scores belonging to the predictions and IoU with the manual markings, which we establish based on empirical assessments. Finally, we select the two best thresholds (the best precision and the best recall obtained) and apply the model, configuration, and thresholds selected on the test group.

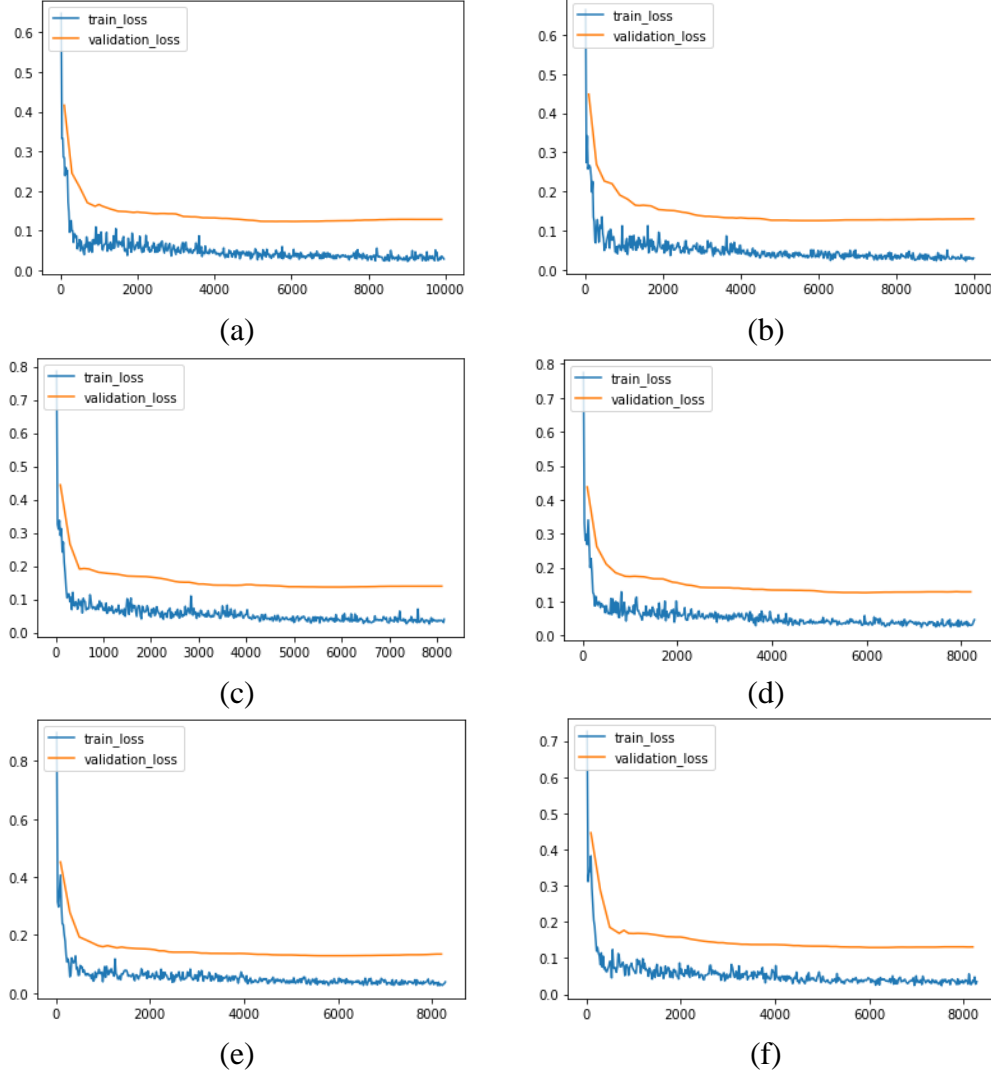


Figure 21: Loss curves about the training base and the validation base for six different experiments using the proposed architecture.

Table 3: COCO evaluation metrics and AF-score achieved by the best model from experiments. The hyperparameters values are specified in Table 1.

	Types	Instances	AP	AR	AF-score
Validation	130	234	98.4	80.9	88.8
Test	251	473	92.8	75.5	83.7

Table 4 shows the results achieve for precision and recall metrics and their respective thresholds in which we test after the choice of the best model. We highlight the best results for each metric. The model achieves better accuracy when considering at least IoU 5% and Score 90%, and better recall when considering at least IoU 5% and Score 70%. The Table shows a direct relationship between the precision and recall metrics and the score and IoU thresholds: precision is directly

proportional to the value of the score, and the recall is inversely proportional to the amount of the IoU. Additionally, we observe a trade-off between precision and recall, where maximizing one of these metrics usually means decreasing the other. The table shows that considering 5% for IoU and 70% for recall the method achieved a more equilibrated relationship between precision and recall. These observations are essential when analyzing the requirements for a future commercial application.

Finally, Table 5 presents the values achieved by the network for the F-score metric using the thresholds for Score and IoU. We highlight the two best results and the averages obtained on the validation and test sets. The Table shows that we consider all the thresholds we reach the average values above 95% for the metric f-score on validation dataset. On test set, we reach 95% for the metric f-score considering 5% of IoU and 70% of score.

Table 4: Precision and recall values achieved using several thresholds for the IOU and score measures. We highlight the two best configurations used.

		%	Precisão-Recall		
Group	IoU/Score	50	70	90	
Validation	5	95.4 – 97.0	96.6 – 96.2	99.1 – 95.3	
	50	95.0 – 96.6	96.1 – 95.7	99.0 – 95.0	
	70	95.0 – 95.2	96.0 – 95.0	99.1 – 94.0	
Test	5	-	<u>98.0 – 92.1</u>	<u>99.3 – 90.0</u>	

Table 5: Values achieved for f-score metric using several thresholds for the IOU and score measures. We highlight the two best configurations used.

		%	F-Score		
Grupo	IoU/Score	50	70	90	
Validation	5	96.2	96.4	97.2	
	50	95.8	95.9	97.0	
	70	95.1	96.2	96.5	
Test	5	-	<u>95.0</u>	<u>94.4</u>	

Table 6 shows the results achieved by best model applied in the test group when analyzing the detected and the undetected stamps belonging to types present in the training group. We also show detected and undetected stamps belonging to absent types in the training group. We achieved detection of 97% of the 236 stamps belonging to types present in the training group, and we detected 90% of the stamps

belonging to absent types in the training group. Figures 22(a) and 22(b) illustrate in a 3D graph the distribution of test set instances presented, in numbers, in Table 6. Figure 22(a) shows the distribution of detected stamps, and Figure 22(b) shows the distribution of undetected stamps. Instances represented by a red circle belong to classes absent from the training group, and instances represented by a blue square belong to classes present in the training group.

Table 6: Results in detecting instances of test group stamps, which belong to types present and away in the training group.

Training set	Detected stamps		Total	%
	Yes	No		
Present	229	7	236	97%
Absent	212	23	235	94%

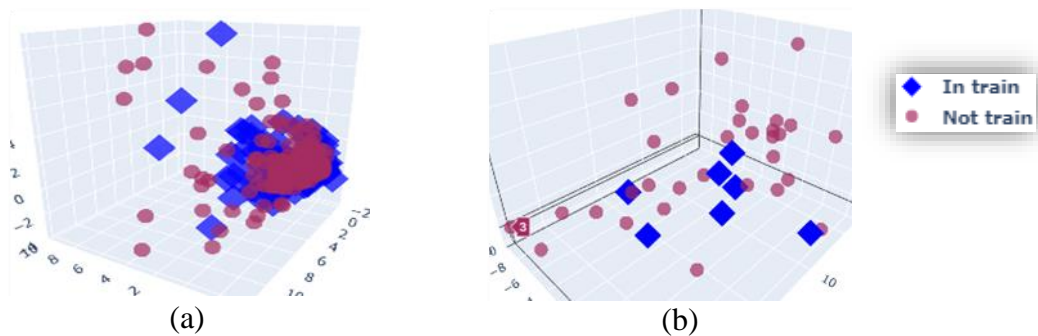


Figure 22: Illustration in a 3D graph the distribution of test set instances. Instances represented by a red circle belong to classes absent from the training group, and instances represented by a blue square belong to classes present in the training group.

Table 7 shows the same analysis as Table 6 concerning stamp types instead of stamp instances. We detected 98% of the 87 types present in the training group. Regarding the absent types in the training group, we detected 93% of the 152 types present in the training group.

Table 7: Results in the detection of stamp types from the test group, and which belong to types present and absent in the training group.

Training set	Detected stamps types			
	Yes	No	Total	%
Present	85	2	87	98
Absent	152	12	164	93

With the results illustrated in Tables 6 and 7, we conclude that our approach can detect stamps and stamp types if they belong to the training base, even if there are few instances. Even in stamp types never seen before by the network, our approach achieved good generalization ability. In addition to these results, we have conducted further experiments with recently proposed network architectures. Our findings do not show any significant improvements in terms of the evaluated metrics. For a detailed overview of these experiments, please refer to Appendix 2.

5.2. Instance Augmentation

Figure 23 illustrates the reduced features of the stamp instances using PCA corresponding to the validation set and plotted on a three-dimensional graph. Blue-colored objects represent detected instances. Yellow objects represent instances whose types are undetected and used for synthetic data generation. In our experiments, the synthetic data generated fed training set only. We can see that the undetected instances are more isolated on the graph, while the detected instances tend to be centered on a point.

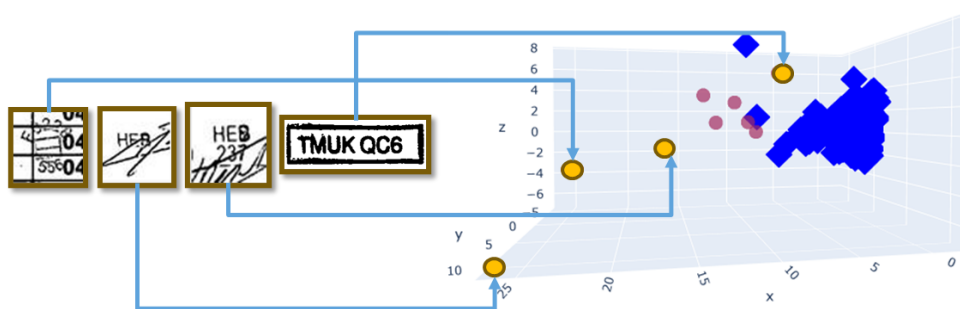


Figure 23: Illustration of the reduced characteristics using PCA of detected and undetected instances plotted on a three-dimensional graph. In blue, the instances detected by the network. Undetected instances are red, and instances selected for synthetic data generation experiments are yellow.

The experiments using synthetic data start after selecting the first type of stamp. Figure 24 illustrates the type of stamp selected for the initial experiments and the respective instance in the graph. Figures 24(a), 24(b), 24(c), and 24(d) illustrate the results obtained. In the experiments, ten pages are generated, varying the amount, and using clean and noisy stamps. Considering the biggest stamp and the portrait and landscape layout of the pages, we experiment to generate 2, 6 and 12 stamps per page. Our experiments found that generating 12 instances per page helped the network to detect the selected class. Then, setting the number of instances per page to 12, we verified that cleaning up the instances to be pasted helped the network to detect the selected class.

Successful setup of experiments performed with the first selected stamp type

is applied to the other stamp types. After application to all types, the method did not obtain satisfactory results for only one of the types. New experiments using double the number of pages (and instances) for this type showed improved results. In other words, we generated 20 synthetically pages for the type that failed and ten pages for the others—however, the method neither achieved success nor any improvement in detection for a specific instance. A compelling reason for this stamp instance is not detected because it has several missing parts (referring to digit 237). Figure 25 highlights the undetected instance in yellow and presents the new results. Apart from this case, all stamps corresponding to the synthetically generated types were detected.

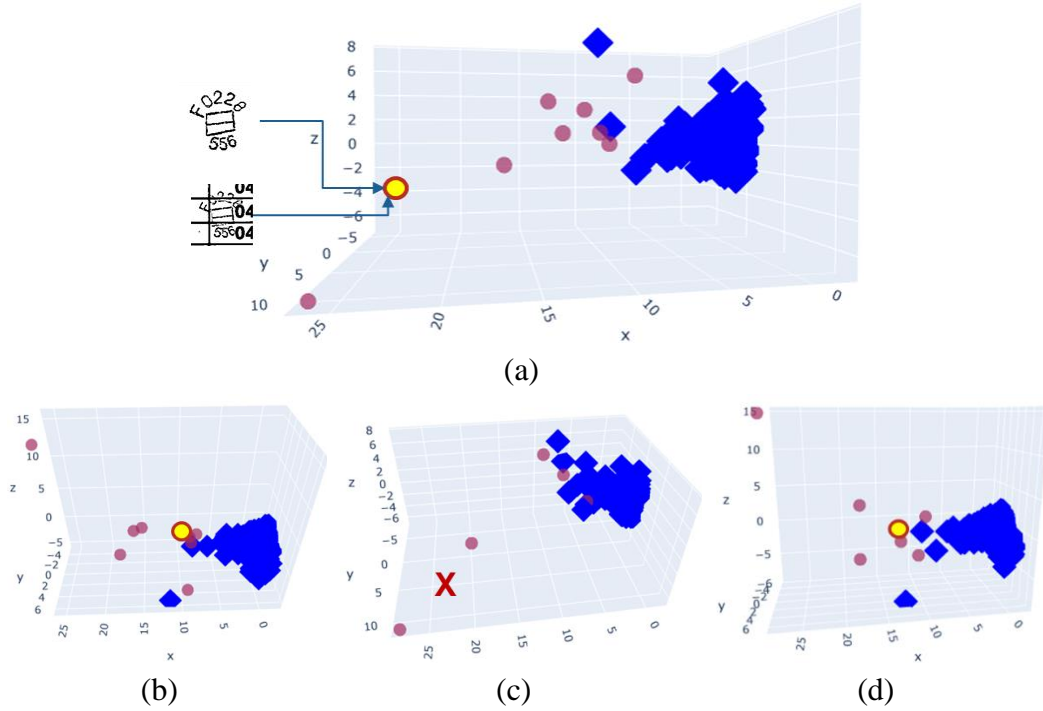


Figure 24: (a) Location of selected stamp instance. (b) Stamp not detected after generating two clean instances per page. (c) Stamp detected after generating 12 clean instances per page. (d) Stamp not detected after generating 12 instances, no cleaning, per page.

We relied on the results achieved and shown in Tables 4 and 5 to evaluate the experiments performed with Instance Augmentation. We evaluated the new experiments using synthetic data generation considering the setting of thresholds in which we obtained a more balanced relationship between the values of precision and recall, that is, 5% of IoU and 70% of Score, together with the best F-score achieved. Table 8 shows the ten best results achieved based on validation using the new experiments' precision, recall, and f-score metrics. The table also compares the result obtained in the test. Overall, all experiments using the proposed augmentation method had better results.

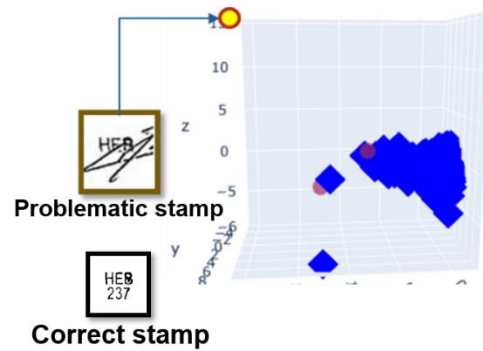


Figure 25: Single instance not detected by the network after generating synthetic data of selected stamps. Highlighted in yellow color the problematic stamp instance. Highlighted in black color an example of the same type without any problems.

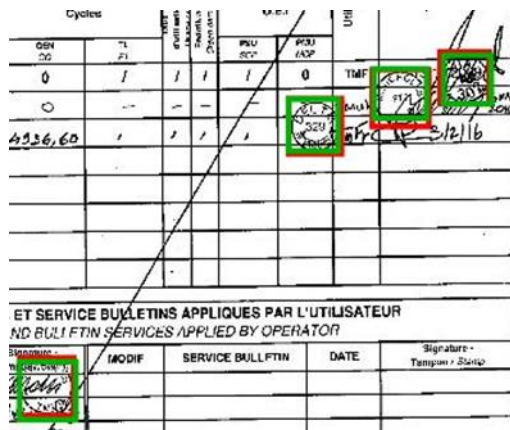
Table 8: Comparison between the previous and new the results achieved for validation and test sets using the proposed method for Instance Augmentation.

Set	Experiment	Precision	Recall	F-score
Validation	Experiment 0 (no augmentation)	96.2	96.2	96.4
	Experiment 1	98.3	98.3	98.3
	Experiment 2	98.7	97.9	98.3
	Experiment 3	97.5	98.7	98.1
	Experiment 4	97.5	98.7	98.1
	Experiment 5	97.9	97.9	97.9
	Experiment 6	98.3	97.4	97.8
	Experiment 7	98.3	97.4	97.8
	Experiment 8	97.4	97.9	97.6
	Experiment 9	97	98.3	97.6
	Experiment 10	97	97.9	97.4
Test	No augmentation	98	92.1	95
	augmentation	97.3	93.8	95.5

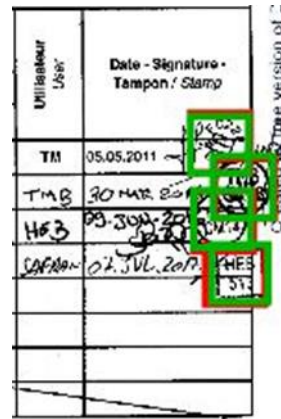
6. Discussion

6.1. Stamp Detection

We evaluate different situations in the test set to perform a detailed analysis of the network's performance. We consider three different points of view, illustrated in Figures 26, 27 e 28. The manual marking is the highlight in red and the network marking in green. In lilac and blue colors, we highlight undetected stamps and false positives detected by the network, respectively. Stamps with only green marking mean 100% of IoU.

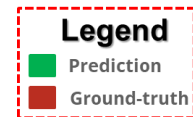


(a) Illustration of a page where the network detects multiple stamps of different geometric shapes, and similar types such as the stamps heb292, heb294, and heb294r.



(b) Illustration of a page where the network detects stamps with geometric shapes, overlays/occlusions, and low visual quality.

N°	Inspeção de Presença Data e Assinatura Data e Assinatura Data e Assinatura	Calibração com (sem) Perfuração Calibração	Data e Assinatura Data e Assinatura Data e Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura	Data e Assinatura Data e Assinatura Data e Assinatura	Inspeção de Presença Data e Assinatura Data e Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura	Assinatura Subsequente (sem?) Subsequente Assinatura
1	HEB 292	0.05	HEB 485	71.49	HEB 486	HEB 294	OK	OK	OK	OK	OK
2	HEB 292	0.11	HEB 485	70.64	HEB 478	HEB 478	OK	OK	OK	OK	OK



(c) Illustration of a page where the network detects stamps with the presence of overlapping and distant locations.

Figure 26: First point of view of case studies of the application of the neural network for stamp detection, where we highlight cases of success within the context of some simple (shape, intra-similarity, location, multiplicity) and complex (overlap/occlusion, quality, and rotation) features in the dataset.

In Figure 26, we illustrate case studies of the application of the neural network for stamp detection, where we highlight cases of success within the context of some simple (Figure 26(c) - shape, intra-similarity, location, multiplicity) and complex features in the dataset (Figures 26 (a) and (b) - overlap/occlusion, quality,

and rotation). Another particular case is the successful hexagon-shaped stamps detection (Figure 26 (c)). Despite this stamp group having a low number of types in the dataset, it has two similar features to the circular group: the proximity between the shape and similar internal texts. This fact explains the success in detecting hexagon-shaped stamps.

In our second analysis, we compare the application of the two thresholds for Score in the test group. Figure 27 illustrates 2 cases of pages applying different thresholds of scores. This figure presents the same conclusions obtained from Table 3. That is, when we increase the score threshold, the precision increases. However, if we choose to be more flexible and consider stamps where the network scored lower, we increase the recall at the cost of decreasing the precision. This can be observed by comparing Figure 27(a) against Figure 27(c) and Figure 27(b) against Figure 27(d).

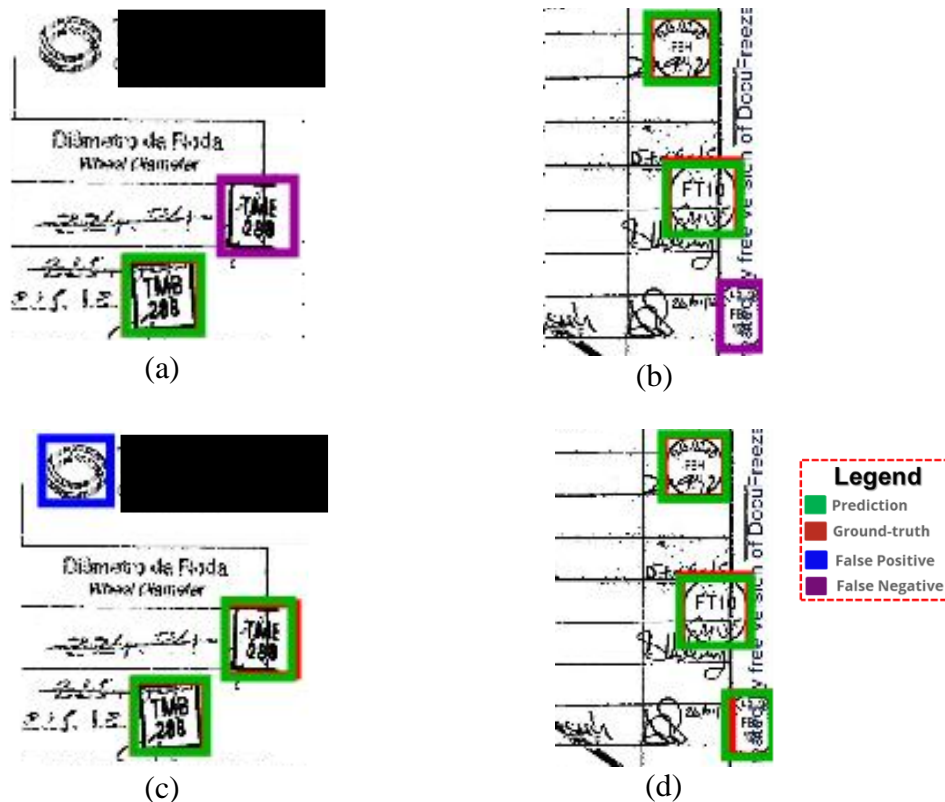


Figure 27: Second point of view of case studies of the neural network application for stamp detection, where we highlight divergences between the thresholds applied in the test group. Cases (a) and (b) consider scores above 90% and cases (c) and (d) consider scores above 70%.

In the last point of view of case studies, we observe some cases of stamp types that our model has difficulty detecting. The first case consists of stamps that do not have surrounding or internal geometric shapes (Figures 28(a) and Figure 28(c)), which can be easily confused with a background (example: other texts

printed on pages). The second case consists of triangular-shaped stamps. However, stamps with the shape "triangle" have a low number of types and instances within the dataset, which also are distributed between the three sets: training, testing, and validation. Furthermore, there is no other stamp type in which the "triangle" shape has a high degree of similarity for take feature sharing.

Figure 28 (c) points to some cases. This Figure highlights 3 specific instances plotted using our Deep Explainability method using PCA (Section 4.4.1). These 2 cases do not have examples in the training or validation set. The default behavior we observe is that cases with few examples or low similarity with other stamp classes tend to appear further away in the graph. Figure 28(c) also illustrates that the highlighted cases are present as examples of undetected instances in Figures 28 (a) and (d).

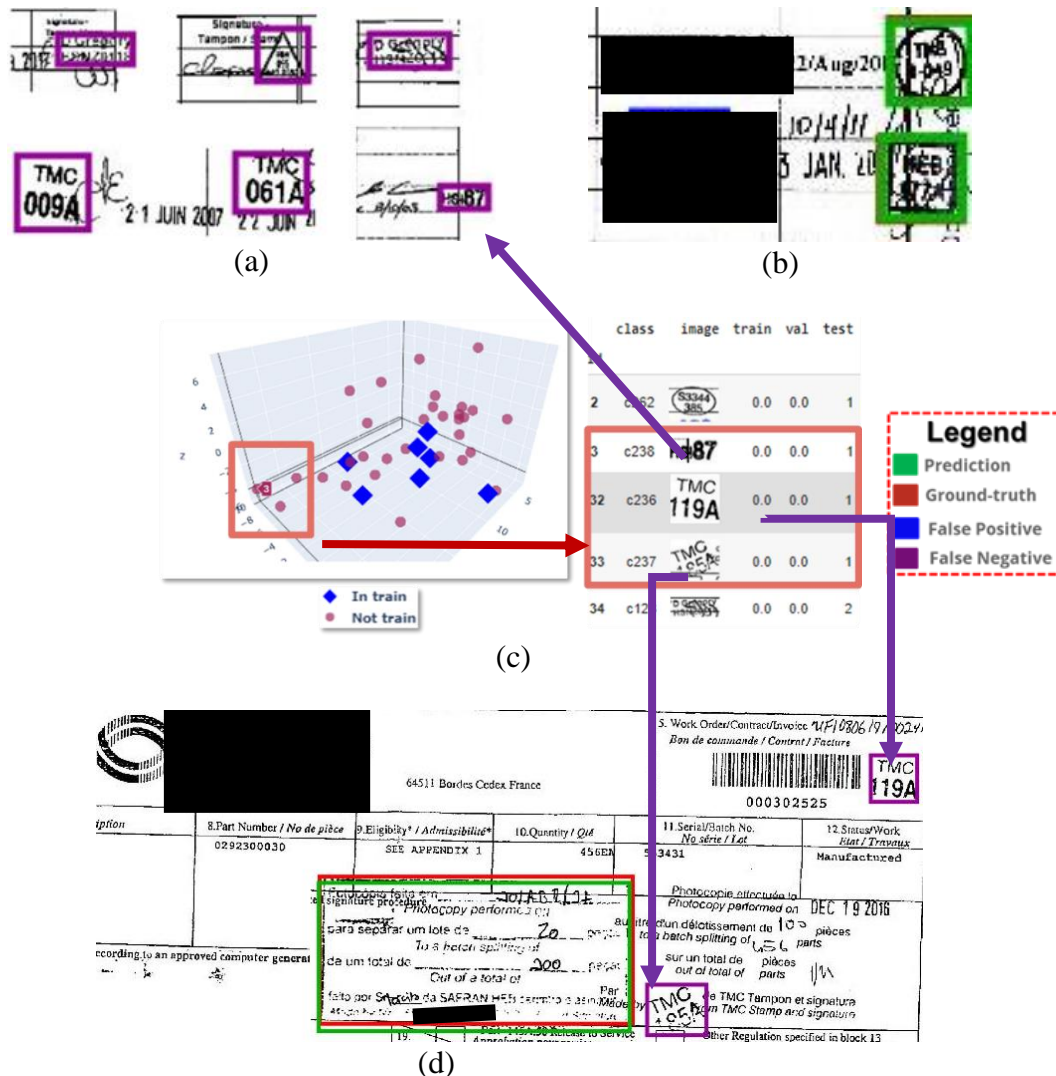


Figure 28: Second point of view of case studies of the application of the neural network for stamp detection, where we highlight cases of types in which the network had difficulties in detecting stamps.

Finally, we also observe cases in which our model mistook a few manuscripts containing the string "heb" with stamps, as shown in Figure 28(b). We believe that it occurs due to the large number of stamps that have this string internally. Stamps containing the string "heb" are in the most stamp types within the dataset (circle, rectangle, triangle, or square). We observe that the network has learned not only the geometric features but also the textual patterns that belong to stamps.

6.2. Instance Augmentation

The proposed method of generating synthetic stamp data based on instance augmentation showed promising and satisfactory results. The most difficult types of stamps were selected to generate data and perform the experiments. The methodology had problems detecting only one type of stamp selected in the experiments performed. For this case, new experiments are conducted.

Table 9 presents the Score metric in detecting each instance of the type of stamp that the network failed in detection. The higher the score, the better the network confidence in predicting a stamp. We evaluated the use of more pages for generation and the influence of different combinations of transformations in the instances. The network confidence when detecting the stamp's instances in which the detection failed is indicated for each experiment scenario.





The results obtained show that applying only rotation in synthetic instances does not significantly improve the results. Best results are achieved by applying data transformations through morphological operations. Overall, the morphological erosion operation proved to be the most promising. Increasing the number of pages brought significant improvements in most experiments.

The new investigations conducted did not reach the detection of the specific instance that has missing parts. The numbering contained in the stamps are striking characteristics, and the network is not trained using instances without the numbering that make up the body of the stamp. This fact explains why increasing the number of pages, or any combination of data transformations does not change the network's trust for this instance. In a real scenario, the correct thing to do is to consider this case as invalid since the omission of the check digit should not occur.

Another point to consider is related to the number of instances generated. We show that it may be necessary to have a more significant number of instances for certain types of stamps. The following factors can explain this fact. First, some types of stamps share characteristics, such as shape or strings, which one of the main advantages of deep networks: the sharing of features. Thus, the network may not have difficulties detecting a new type of stamp or one with few instances if it is like some dense base training. A small additional amount can be evaluated if data

generation is needed to improve results.

Table 9: Score in detecting each instance of the type of stamp that the network failed in detection.

Transformation/ Instance	10 pages		20 pages	
				
Dilatation	11.9%	0%	95.2%	0%
Erosion	95.4%	0%	95.7%	0%
Rotation	9.0%	0%	6.3%	0%
Dilatation, Erosion	9.5%	0%	87.0%	0%
Dilatation, Rotation	5.2%	0%	75.2%	0%
Erosion, Rotation	52.8%	0%	44.4%	0%
Dilatation, Erosion, Rotation	27.5%	0%	97.8%	0%

However, as shown in Table 9, cases may require a more considerable amount of synthetically generated data, as it belongs to a type of stamp that configures itself as an outlier: there is no shape to share, and the digits can be easily confused with typed texts. Another case that would require a more significant number of generated instances is the stamp that presents a triangular shape in Figure 28 (a). This type does not have any example in the training base or other similar classes, so it was not possible to perform experiments with this specific type.

Therefore, the number of instances generated by type must account for prior knowledge of the dataset, considering characteristics such as the number of groups and instances that share certain features and the realization and evaluation experiments. The generation must have a balanced character since a generation tending to infinity can cause the data to be unbalanced concerning certain types of stamps, leading the learning to a bias about the majority type.

Table 8 presents the new results obtained for the validation and test groups after applying the proposed data augmentation method. The Table shows that overall, the network obtained improvements with greater emphasis on the validation group. This result is expected for several reasons. First, the experiments analyze the results obtained using the validation group. Although we hit these same types in the test group, we cannot guarantee that other types of stamps will not be

harmed concerning the selected training period. The ideal scenario is to generate synthetic data balancing the distribution of all types of stamps.

Another justification is that the types of stamps for generating synthetic data were selected based on our knowledge of the validation basis. This makes it possible to target the network better to hit the types contained in the validation base. However, it is still very hard to successfully detect types of stamps in the test base that are not contained in either the validation set or the training set by generating synthetics. However, this work shows the feasibility of generating synthetic instances for the types of stamps in which the network fails to detect in controlled environments. The method even allows commercial applications to detect stamps in documents with types of standardized stamps.

6.3. Comparison with Related Works

In this section, it is presented a comparison of the results of this work with other related works. These are shown in Table 10, that presents information about the databases used by other researchers and the performance metrics about their methods.

It is difficult to compare the works due to the variability in the databases used, the number of classes, distribution of instances by classes, and metrics used.

However, we can observe that our work has several highlights concerning related works. First, we provide information about the number of types present in our dataset. Our dataset comprises a quantity of types much higher than the number of types in the related works that provide this information. This shows how difficult our task is. We use a reasonable number of instances compared to the other works. However, our work has a much lower instance/class ratio, another point that shows our task's difficulty.

We build our database from documents that leading with information, acquisition process, and distributions from a world real business scenario. Overall, even when looking at more metrics we achieved superior values. Finally, we are the only work that provided a method for dataset augmentation based on Instance Augmentation.

Table 10: Precision and recall values achieved after applying the data augmentation proposed method.

Authors	Object	Types	Instances	AP50	Precision	Recall	Acc.
[4]	Stamp	12	127				92
[3]	Stamp	19	530		100	20	
[1]	Stamp		400		84	83	
[74]	Stamp				22.54	97.61	
[2]	Stamp			89.6			
[27]	Stamp			89.2			
[75]	Stamp		918	81			97
[28]	Logo	32	6810	65.8			
[29]	Logo	32	3940	66.35			
[30]	Logo	32	2240	66.9	92.8	96.5	
Ours	Stamp	251	1878	92.8	98	92.1	
Ours (augmentation)	Stamp	251	1878	94.3	97.3	93.8	

7. Conclusion

This work presented a computational method capable of fully automating the stamps detection in scanned documents. First, we described a detailed procedure for the exploratory dataset analysis. Second, we proposed an innovative greedy strategy to generate training data that considers issues with dataset imbalance. We presented the successful application of transfer learning and fine-tuning techniques using a pre-trained deep network over a complex real-world dataset. The results demonstrate the importance of dataset analysis to guide the choice of object detection framework and the data splitting strategy proposed in this work.

The neural network could generalize knowledge to up to 3x more classes than those present during training through our method. The algorithm achieved up to 99% precision, 94% recall, and 95.0% average f-score in the final test procedure. We can conclude that the developed solution can successfully improve the efficiency and accuracy of an otherwise manual and labor-intensive verification process.

This work paves the way for additional research in object detection, imbalanced datasets, and stamp recognition. Future work could apply our proposed dataset analysis method to improve the performance of learning algorithms in other domains. Another investigation would be to generate synthetic data using image processing and Siamese networks to improve dataset imbalance. We also intend to continue this research to recognize different types of stamps using one or more networks for different classes and/or groups of classes.

This work also presents a methodology for generating synthetic stamp data on pages based on instance augmentation. We combine morphological operations, noise addition, and rotation operation. The method first pinpoints the most complex cases for the trained model to detect. After analyzing and selecting the most difficult instances, the method generates instances, pixel by pixel, through image processing algorithms. The method is successful in detecting all selected stamp types.

New experiments were performed for the more complex stamp type to evaluate the combinations of the transformations applied to the instances and the increase in the number of pages in the generation. Overall, the method achieved better results by increasing the number of pages. We note that the least essential operation among those evaluated for synthetic data generation is the rotation operation. A single instance was not detected. However, due to the instance has uncharacterized, the ideal is it be considered invalid and disregarded in future experiments. The final evaluation showed that the method works, and instances should be generated in adequate quantities for the type of stamp chosen. Synthetic instances must conform to the selected stamp type.

References

1. B. Micenková and J. Van Beusekom, "Stamp detection in color document images," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, pp. 1125–1129, 2011, doi: 10.1109/ICDAR.2011.227.
2. N. Sharma, R. Mandal, R. Sharma, U. Pal, and M. Blumenstein, "Signature and logo detection using deep CNN for document image retrieval," *Proc. Int. Conf. Front. Handwrit. Recognition, ICFHR*, vol. 2018-Augus, pp. 416–422, 2018, doi: 10.1109/ICFHR-2018.2018.00079.
3. P. P. Roy, U. Pal, and J. Lladós, "Document seal detection using GHT and character proximity graphs," *Pattern Recognit.*, vol. 44, no. 6, pp. 1282–1295, 2011, doi: 10.1016/j.patcog.2010.12.004.
4. P. P. Roy, U. Pal, and J. Lladós, "Seal detection and recognition: An approach for document indexing," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, pp. 101–105, 2009, doi: 10.1109/ICDAR.2009.128.
5. F. Nourbakhsh, P. B. Pati, and A. G. Ramakrishnan, "Automatic Seal Information Reader," pp. 1–4, 2007.
6. Y. Han, S. Ma, F. Zhang, and C. Li, "Object detection of remote sensing airport image based on improved faster R-CNN," *J. Phys. Conf. Ser.*, vol. 1601, no. 3, 2020, doi: 10.1088/1742-6596/1601/3/032010.
7. Y. Cheng, N. Pu, and P. Tung, "Seal Recognition Using the Shape Selection Algorithm," *2006 IEEE Int. Conf. Electro/Information Technol.*, no. 1, pp. 544–547, 2006.
8. Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," pp. 1–21, 2019, [Online]. Available: <https://arxiv.org/pdf/1807.05511.pdf>.
9. Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," pp. 1–39, 2019, [Online]. Available: <http://arxiv.org/abs/1905.05055>.
10. P. Soviany and R. T. Ionescu, "Frustratingly Easy Trade-off Optimization Between Single-Stage and Two-Stage Deep Object Detectors," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1, pp. 366–378, 2019, doi: 10.1007/978-3-030-11018-5_33.
11. L. Liu *et al.*, "Deep Learning for Generic Object Detection: A Survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020, doi: 10.1007/s11263-019-01247-4.
12. J. Dai *et al.*, "Deformable Convolutional Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 764–773, doi: 10.1109/ICCV.2017.89.
13. Y. Ren, C. Zhu, and S. Xiao, "Deformable faster R-CNN with aggregating

- multi-layer features for partially occluded object detection in optical remote sensing images,” *Remote Sens.*, vol. 10, no. 9, 2018, doi: 10.3390/rs10091470.
14. L. Deng, H. H. Chu, P. Shi, W. Wang, and X. Kong, “Region-based CNN method with deformable modules for visually classifying concrete cracks,” *Appl. Sci.*, vol. 10, no. 7, p. 2528, 2020, doi: 10.3390/app10072528.
 15. J. Peng, C. Bao, C. Hu, X. Wang, W. Jian, and W. Liu, “Automated mammographic mass detection using deformable convolution and multiscale features,” *Med. Biol. Eng. Comput.*, vol. 58, no. 7, pp. 1405–1417, 2020, doi: 10.1007/s11517-020-02170-4.
 16. Z. Shi *et al.*, “Detecting Organisms for Marine Video Surveillance,” in *2020 Global Oceans 2020: Singapore - U.S. Gulf Coast*, 2020, pp. 1–7, doi: 10.1109/IEEECONF38699.2020.9389458.
 17. A. Körez and N. Barişçi, “Object detection with low capacity GPU systems using improved faster R-CNN,” *Appl. Sci.*, vol. 10, no. 1, 2019, doi: 10.3390/app10010083.
 18. T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 936–944, 2017, doi: 10.1109/CVPR.2017.106.
 19. R. Ribani and M. Marengoni, “A Survey of Transfer Learning for Convolutional Neural Networks,” *Proc. - 32nd Conf. Graph. Patterns Images Tutorials, SIBGRAPI-T 2019*, pp. 47–57, 2019, doi: 10.1109/SIBGRAPI-T.2019.00010.
 20. C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
 21. D. Dwibedi, I. Misra, and M. Hebert, “Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection,” *arXiv*, 2017.
 22. G. Ghiasi *et al.*, “Simple copy-paste is a strong data augmentation method for instance segmentation,” *arXiv*, 2020.
 23. N. Xie, G. Ras, M. van Gerven, and D. Doran, “Explainable Deep Learning: A Field Guide for the Uninitiated,” 2020, [Online]. Available: <http://arxiv.org/abs/2004.14545>.
 24. V. Belle and I. Papantonis, “Principles and Practice of Explainable Machine Learning,” vol. 4, no. July, pp. 1–25, 2020, doi: 10.3389/fdata.2021.688969.
 25. X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable convnets V2: More deformable, better results,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 9300–9308, 2019, doi: 10.1109/CVPR.2019.00953.
 26. Y. S. Chen, “Automatic identification for a Chinese seal image,” *Pattern Recognit.*, vol. 29, no. 11, pp. 1807–1820, 1996, doi: 10.1016/0031-

3203(96)00032-5.

27. C. Jun, Y. Suhua, and J. Shaofeng, "Automatic classification and recognition of complex documents based on Faster RCNN," *2019 14th IEEE Int. Conf. Electron. Meas. Instruments, ICEMI 2019*, pp. 573–577, 2019, doi: 10.1109/ICEMI46757.2019.9101847.
28. K. Palecek, "Deep learning for logo detection," *2019 42nd Int. Conf. Telecommun. Signal Process. TSP 2019*, pp. 609–612, 2019, doi: 10.1109/TSP.2019.8769038.
29. J. Song and H. Kurniawati, "Exploiting trademark databases for robotic object fetching," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2019-May, pp. 4946–4952, 2019, doi: 10.1109/ICRA.2019.8793829.
30. A. K. Bhunia, A. K. Bhunia, S. Ghose, A. Das, P. P. Roy, and U. Pal, "A deep one-shot network for query-based logo retrieval," *Pattern Recognit.*, vol. 96, p. 106965, 2019, doi: 10.1016/j.patcog.2019.106965.
31. H. Guo, V. Swaminathan, and S. Mitra, "A Scalable Data Augmentation and Training Pipeline for Logo Detection," *Proc. - 2019 IEEE Int. Symp. Multimedia, ISM 2019*, pp. 48–55, 2019, doi: 10.1109/ISM46123.2019.00016.
32. R. E. Neapolitan and X. Jiang, *Neural Networks and Deep Learning*. Springer International Publishing, 2018.
33. S. Haykin, *Neural Networks and Learning Machines Third Edition*, 3rd ed., no. 114–115. Pearson, 2006.
34. C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.
35. A. Krizhevsky, I. Sutskever, and H. Geoffrey E., "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.* 25, pp. 1–9, 2012, doi: 10.1109/5.726791.
36. B. Van Ginneken, A. A. A. Setio, C. Jacobs, and F. Ciompi, "Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans," *Proc. - Int. Symp. Biomed. Imaging*, vol. 2015-July, pp. 286–289, 2015, doi: 10.1109/ISBI.2015.7163869.
37. L. G. Hafemann, "An Analysis of Deep Neural Networks for Texture Classification," 2014.
38. M. S. Ebrahimi and H. K. Abadi, "Study of Residual Networks for Image Recognition," *Lect. Notes Networks Syst.*, vol. 284, pp. 754–763, 2018, doi: 10.1007/978-3-030-80126-7_53.
39. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
40. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision*

and pattern recognition, 2016, pp. 770–778.

41. K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778, Accessed: Aug. 07, 2019. [Online]. Available: <http://image-net.org/challenges/LSVRC/2015/>.
42. J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in *Advances in Neural Information Processing Systems 25*, 2014, pp. 3320–3328, Accessed: Jun. 21, 2018. [Online]. Available: <https://arxiv.org/pdf/1411.1792.pdf>.
43. K. B. Ahmed, L. O. Hall, D. B. Goldgof, R. Liu, and R. A. Gatenby, “Fine-tuning convolutional deep features for MRI based brain tumor classification,” no. March 2017, p. 101342E, 2017, doi: 10.1117/12.2253982.
44. L. Jiao *et al.*, “A survey of deep learning-based object detection,” *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.
45. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
46. Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, “Neural Style Transfer: A Review,” *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 11, pp. 3365–3385, 2020, doi: 10.1109/TVCG.2019.2921336.
47. H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, “Factors of Transferability for a Generic ConvNet Representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, 2016, doi: 10.1109/TPAMI.2015.2500224.
48. N. Tajbakhsh *et al.*, “Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016, doi: 10.1109/TMI.2016.2535302.
49. S. Akçay; M. E. Kundegorski; M. Devereux; T. P. Breckon, “Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery,” in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1057–1061.
50. F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 8, pp. 2674–2693, 2019, doi: 10.1109/TVCG.2018.2843369.
51. E. Tjoa and C. Guan, “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 11, pp. 4793–4813, 2021, doi: 10.1109/TNNLS.2020.3027314.
52. G. Joshi, R. Walambe, and K. Kotecha, “A Review on Explainability in Multimodal Deep Neural Nets,” *IEEE Access*, vol. 9, pp. 59800–59821,

- 2021, doi: 10.1109/ACCESS.2021.3070212.
53. Y. Zhang, P. Tino, A. Leonardis, and K. Tang, “A Survey on Neural Network Interpretability,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 5, pp. 726–742, 2021, doi: 10.1109/TETCI.2021.3100641.
 54. L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, “Explaining explanations: An overview of interpretability of machine learning,” *Proc. - 2018 IEEE 5th Int. Conf. Data Sci. Adv. Anal. DSAA 2018*, pp. 80–89, 2019, doi: 10.1109/DSAA.2018.00018.
 55. J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, “A survey of visual analytics techniques for machine learning,” *Comput. Vis. Media*, vol. 7, no. 1, pp. 3–36, 2021, doi: 10.1007/s41095-020-0191-7.
 56. M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau, “ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models,” *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 88–97, 2018, doi: 10.1109/TVCG.2017.2744718.
 57. A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren, “Do Convolutional Neural Networks Learn Class Hierarchy?,” *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 152–165, 2018, doi: 10.1109/TVCG.2017.2744683.
 58. C. R. García-Alonso, L. M. Pérez-Naranjo, and J. C. Fernández-Caballero, “Multiobjective evolutionary algorithms to identify highly autocorrelated areas: The case of spatial distribution in financially compromised farms,” *Ann. Oper. Res.*, vol. 219, no. 1, pp. 187–202, 2014, doi: 10.1007/s10479-011-0841-3.
 59. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Pearson Prentice Hall, 2010.
 60. J. R. Parker, *Algorithms for Image Processing and Computer Vision*, 2nd ed. Wiley Publishing, 2010.
 61. A. Madani, O. Boussaid, and D. E. Zegour, “Semi-structured documents mining: A review and comparison,” *Procedia Comput. Sci.*, vol. 22, pp. 330–339, 2013, doi: 10.1016/j.procs.2013.09.110.
 62. I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. 2011.
 63. D. Jared, *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. Wiley, 2014.
 64. N. Tajbakhsh *et al.*, “Convolutional neural networks for medical image analysis: Full training or fine tuning?,” *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.
 65. K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, “Imbalance Problems in Object Detection: A Review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2020, doi: 10.1109/tpami.2020.2981890.
 66. D. Prasad, “Survey of the problem of object detection in real images,” *Int.*

- J. Image Process.*, no. 6, pp. 441–466, 2012, [Online]. Available: <http://www.cscjournals.org/csc/manuscript/Journals/IJIP/volume6/Issue6/IJIP-702.pdf>.
67. J. M. Johnson and T. M. Khoshgoftaar, “Survey on deep learning with class imbalance,” *J. Big Data*, vol. 6, no. 1, p. 54, 2019, doi: 10.1186/s40537-019-0192-5.
 68. R. Padilla, S. L. Netto, and E. A. B. Da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” *Int. Conf. Syst. Signals, Image Process.*, vol. 2020-July, pp. 237–242, 2020, doi: 10.1109/IWSSIP48289.2020.9145130.
 69. F. Chollet, *Deep Learning with Python*, 1st ed. Manning, 2018.
 70. A. Goodfellow, I. and Bengio, Y. and Courville, *Deep Learning*. MIT press, 2016.
 71. C. M. Bishop, *Neural Networks for Pattern Recognition*. OXFORD, 1995.
 72. D. R. Okada and R. Blankstein, *Digital Image Processing for Medical Applications*, vol. 52, no. 4. Cambridge, 2009.
 73. K. Adrian and B. Gary, *Computer vision in C++ with the Opencv Library*. O’Reilly, 2014.
 74. A. Alaei, P. P. Roy, and U. Pal, “Logo and seal based administrative document image retrieval: A survey,” *Comput. Sci. Rev.*, vol. 22, pp. 47–63, 2016, doi: 10.1016/j.cosrev.2016.09.002.
 75. P. Forczmanski, A. Smolinski, A. Nowosielski, and K. Malecki, “Segmentation of Scanned Documents Using Deep-Learning Approach,” pp. 360–369, 2020, doi: 10.1007/978-3-030-19738-4.
 76. G. and A. L. Seber, *Linear Regression Analysis*, Second. Wiley Publishing, 2003.
 77. G. B. Renher, A. C. and Schaalje, *Linear Models in Statistics*, Second., vol. 96, no. 455. Wiley, 2008.
 78. A. Schneider, G. Hommel, and M. Blettner, “Linear Regression Analysis,” *Dtsch. Arztebl.*, vol. 107, no. 44, pp. 776–782, 2010, doi: 10.3238/arztebl.2010.0776.
 79. M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, Second. 2003.
 80. C. Szegedy *et al.*, “Going Deeper with Convolutions,” in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 2014, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
 81. H. Rezatofighi, N. Tsoi, J. Y. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “Generalized intersection over union: A metric and a loss for bounding box regression,” *arXiv*, pp. 658–666, 2019.
 82. X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: Deformable Transformers for End-to-End Object Detection,” pp. 1–16,

- 2020, [Online]. Available: <http://arxiv.org/abs/2010.04159>.
83. T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, 2020, doi: 10.1109/TPAMI.2018.2858826.
 84. Google Colaboratory, Colab, <https://colab.research.google.com/>, accessed in: 20/12/2020.
 85. Matlab, MathWorks, <https://www.mathworks.com/products/matlab.html/>, accessed in: 20/12/2020.
 86. Detectron2, Facebook IA, <https://github.com/facebookresearch/detectron2/>, accessed in: 20/12/2020.
 87. Pytorch, Pytorch, <https://pytorch.org/>, accessed in: 20/12/2020.
 88. Andrew Ng., Coursera Class Notes: Gradient Descent in Practice II – Learning Rate, <https://www.coursera.org/lecture/machine-learning/gradient-descent-in-practice-ii-learning-rate-3iawu>, accessed in: 20/12/2020.
 89. Rajput V., R Programming: Linear Regression, <https://medium.com/aiguys/r-programming-linear-regression-15b32464737>, accessed in: 20/12/2020

Appendix 1

1. Supervised Training

Supervised learning algorithms are learning algorithms that learn to associate some input with some output, given a training set of examples of x inputs and y outputs [70]. The algorithm receives data formed by expected input and output pairs and automatically looks for a function that maps the inputs to their respective outputs. The search is adjusted through feedback signals resulting from measurements that determine the distance between the current output of the algorithm and the expected output [69].

The literature categorizes supervised learning problems mostly are categorized into "regression" and "classification" problems [34, 35]. In a regression problem, the algorithm seeks to predict the results on a continuous output, which means that it is looking for a function that maps the input variables to some continuous function. In a classification problem, the algorithm seeks to predict the results in a discrete output. In other words, this is looking for a function that maps input variables into discrete categories.

1.1. Linear Regression

The primary objective of regression analysis is to calculate a model that tries to represent the underlying relationship between continuous (dependent) variables and independent (explanatory) variables. Linear regression is a type of regression that uses a linear function to predict this relationship [76].

The linear regression is considered parametric in nature which means that it makes assumptions about the dataset. These assumptions are as follows:

- The dependent variables (y) and independent variables (x) have a linear and additive relationship. The term "linear" refers to the fact that the change in y caused by a unit change in x is constant. The term "additive" refers to the fact that the influence of x on y is unaffected by other variables.
- No correlation between x is permitted. Correlation between x result in multicollinearity. When variables are correlated, the model's ability to discern the true influence of x on y becomes extremely challenging. For example, the size of a house in meters and feet are correlated.

- The error terms ϵ must have constant variance. When the variance of ϵ , monitored over different values of an independent variable (x), is non-constant, we say that the dataset suffers from Heteroskedasticity. The Figure 29 shows an example of dataset with Heteroskedasticity.
- The ϵ must be uncorrelated, i.e., the error ϵ_t at the time t cannot be used to determine the error ϵ_{t+1} at the time $t+1$. The presence of correlation in error terms is called Autocorrelation. Autocorrelation drastically affects the regression coefficients and standard error values since they are based on the assumption of uncorrelated error terms.
- The distributions of y and ϵ must be normal.

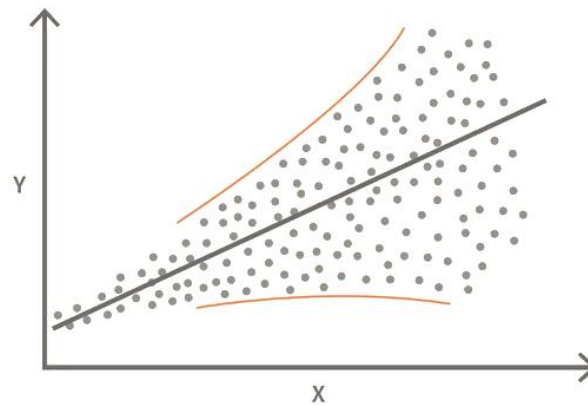


Figure 29: Example of a dataset with Heteroskedasticity. Adapted from: [89].

Due to the presence of these assumptions, linear regression is quite constraining. That is, a linear regression model's performance depends on the fulfillment of these assumptions. If the dataset satisfies them, the model produces satisfactory results. Otherwise, it struggles to achieve good results.

The literature classifies linear regression into two types: univariate or simple linear regression and multivariate or multiple linear regression [37-38]. They will be explained in more detail in the following sections.

1.1.1. Univariate Linear Regression

The simplest type of linear regression is univariate linear regression. As the name univariate suggests, it is used to determine the relationship between a single independent variable and a single dependent variable [78]. In this case, the model that represents the relationship between these variables is a linear function.

Given a dataset of m samples, where the i th sample is composed of a single

independent variable x_i and a single dependent variable y_i that varies as x_i does, the univariate linear regression model is as follows:

$$\alpha + \beta \times x_i + \epsilon_i \quad (19)$$

where:

- Y_i is the estimated value of the i th sample of the dataset. It should be as close as possible to the real value of the dependent variable y_i
- α is known as the constant term or the intercept (also is the measure of the y-intercept value of the regression line)
- β is the coefficient term or slope of the intercept line
- ϵ_i is the error: a random component of the regression handling the residue, i.e. the lag between the estimation and actual value of the dependent parameter.

α and β are known as coefficients. That said, Y_i is estimated by two parameters:

1. The core parameter term, not random in nature, $\alpha + \beta \times x_i$.
2. The random component, ϵ_i .

After hypothesizing that y is linearly related to x , the next step would be estimating the parameters α and β . By doing this, we try to make Y the best possible estimate of the real data y . By mentioning the “best possible” and not the “perfect” estimate, we acknowledge that there is an error (ϵ) in this estimation.

It is important to know that error (ϵ) is an inevitable part of the prediction-making process. No matter how powerful the algorithm we choose, there will always remain an irreducible error. Although we can't eliminate the ϵ term, we can still try to reduce it to the lowest.

To do estimate the best α and β , and thus reduce ϵ to a minimum, the first step is to calculate the prediction error, i.e a measure of how different the estimated values Y_i from the real values y_i is. A common measure is the sum of squares of error of the estimation Y , i.e. sum of squares of ϵ_i values. The equation is as follows:

$$E(\alpha, \beta) = \sum \epsilon_i^2 = \sum_{i=1}^n (Y_i - y_i)^2 \quad (20)$$

The prediction error E must be minimized so that the estimated values Y must be as close as possible of the real values y . This is achieved by finding the best parameters α and β_i . So how do we find the best parameters? There are several techniques to do this, but for the means of the method of this doctoral dissertation

the optimization technique used to minimize the prediction error is the gradient descent. The gradient descent method will be explained in the neural networks section.

1.1.2. Multivariate Linear Regression

A dependent variable guided by a single independent variable is usually not enough in real-world scenarios. For example, if we want to estimate the price of a house, we won't use a single variable like the number of rooms. There are other factors like how old the house, its size, location, etc. For such scenarios, we have the multivariate linear regression.

The multivariate linear regression is quite like the univariate linear regression model, but with multiple independent variables contributing to estimate the dependent variable. Hence, there are multiple coefficients to determine and more complex computation due to the added variables.

The equation of multivariate linear regression is not so different from the univariate one, but it considers more independent variables. It can be represented by:

$$Y_i = \alpha + B_1x_i^{(1)} + B_2x_i^{(2)} + \dots + B_nx_i^{(n)} \quad (21)$$

where:

- Y_i is the estimate of i_{th} sample of the dependent variable y
- x_{ij} denotes the j_{th} independent variable/feature of the i_{th} sample of the dataset
- n is the number of independent variables

Similarly, the cost function is as follows,

$$E(\alpha, B_1, \dots, B_n) = \frac{1}{2m} \sum_{i=1}^m (y_i - Y_i) \quad (22)$$

As we can see, the equation or the cost function is a simple generalization of the univariate linear regression. Now the error must be minimized to find the best estimate of Y . As explained in the section of univariate linear regression, the neural networks section will explain in more details how this is done using gradient descent.

1.2. Classification

The goal in classification is to take an input vector x and assign it to one of

K discrete classes K where $K = 1, \dots, C$. The input space is divided into decision regions called decision boundaries [34]. For now, we will focus on the binary classification problem in which a single target $t \in \{0, 1\}$ such that $t = 1$ represents class C_1 and $t = 0$ represents class C_2 . In other words, the classification model predict discrete class labels using posterior probabilities that lie in the range $y = (0, 1)$ [34]. To achieve this, we consider a generalization of the model described in Equation 1, in which we transform the linear function of w using a nonlinear function $\varphi(\cdot)$ so that:

$$y(x) = \varphi(y(x, w)) \quad (23)$$

where $y(x)$ is the probability of an input x belongs to class $k = 1$.

1.3. Train, validating and test sets

The central challenge in machine learning is that we must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization [70]. However, after just a reasonable number of epochs, machine learning models began to overfit the present data, and their performance on never-before-seen data started stalling (or worsening) compared to their performance on the training data. It is the reason that training data is unable to evaluate the model.

Therefore, the desirable dataset generating method uses a probability distribution to create the training and test data subsets. We commonly establish a series of assumptions referred to as the i.i.d assumptions. These assumptions include that each dataset's examples are unrelated to one another and that the train and test sets are equally dispersed, taken from the same probability distribution [70]. After that, the machine learning algorithm optimizes the model using training data, and the model final is evaluated using the test data. Under this process, the factors that determine the model's performance are its ability to make the training error small and make the gap between training and test error small [70].

However, the model tuning process involves optimizing additional machine learning algorithms settings called model hyperparameters. Hyperparameters settings control the behavior of the learning algorithm, but they are not adapted automatically. Adjusting hyperparameters on the training set is not desirable because the learning process always chooses the maximum possible model capacity, resulting in overfitting. Additionally, tuning hyperparameters model several times based on performance's model test set can quickly result in overfitting to this set, even though the model is never directly trained on it. This phenomenon is known as information leaks because when the hyperparameters are adjusted

based on the model's performance on the evaluation set, some information about the evaluation data leaks into the model [69].

Simple hold-out validation solves this problem, which a portion of the training set is separated for model evaluation on the hyperparameters adjustment process. This new division must adhere to the same guidelines as the previous one. Thus, evaluating a model using this strategy divides the available data into three sets: training, validation, and test. Each of the three sets must be chosen independently: The validation set must be different from the training set to obtain good performance in the optimization stage and, and the test set must be different from both to obtain a reliable estimate of the valid error rate [62]. Once the model is ready, it is evaluated one final time on the test data [69].

1.4. Error Analysis

Model evaluation provides assessing how appropriate the model is to gain insight into the real-world system. Therefore, system designers have to strike the right balance between learning the training set and minimizing the difference between training and the test errors [79]. The capacity of a model is its ability to accommodate a wide range of functions. Models with limited capacity may struggle to fit the training set, while models with high capacity may be overfitted by memorizing training set features that are not useful on the test set.

Two central challenges in machine learning are underfitting and overfitting, and we control whether a model is more likely to overfit or underfit by altering its capacity. Underfitting occurs when the model cannot obtain a sufficiently low error value on the training set. Overfitting occurs when the training and test errors gap is too large [70]. The model is underfitting at the beginning of training because the algorithm has no model fit patterns in the training data. After a certain number of iterations, the model starts to memorize all the training data patterns, and generalization stops improving: the model is starting to overfit [69].

Overfitting and underfitting are often understood in the trade-off between bias and variance in machine learning. To improve the performance of the algorithm further, we need to be able to reduce the bias while at the same time also reducing the variance. The bias is the error caused by the model's simplifying assumptions, whereas variance increases with the model complexity. Therefore, the model with the optimal predictive capability is the one that leads to the best balance between bias and variance, which gives the smallest average training and generalization error at the same time [42–44, 71]

2. Artificial Neural Networks

2.1. Introduction

The human nervous system contains cells, which are referred to as neurons. The neurons are connected to connecting regions called synapses. The strengths of synaptic connections often change in response to external stimuli, and these changes are how learning takes place in living organisms. Artificial networks are models designed as abstractions of brain theory in understanding different aspects of biological neural network learning. An artificial neural network computes a function of the inputs by propagating the computed values from the input neurons to the output neuron(s) and using the weights as intermediate parameters. Learning occurs by changing the weights connecting the neurons [42-43, 79].

The most straightforward neural network is referred to as the perceptron. This neural network contains a single input layer and an output node. We identify three essential elements of the neural model: (1) synapses are characterized by weight or strength of its own; (2) linear combiner sum the input signals, weighted by the respective synaptic strengths; (3) activation function limits the amplitude of the output of a neuron [42-43]. Figure 30 illustrates an example of a neural model.

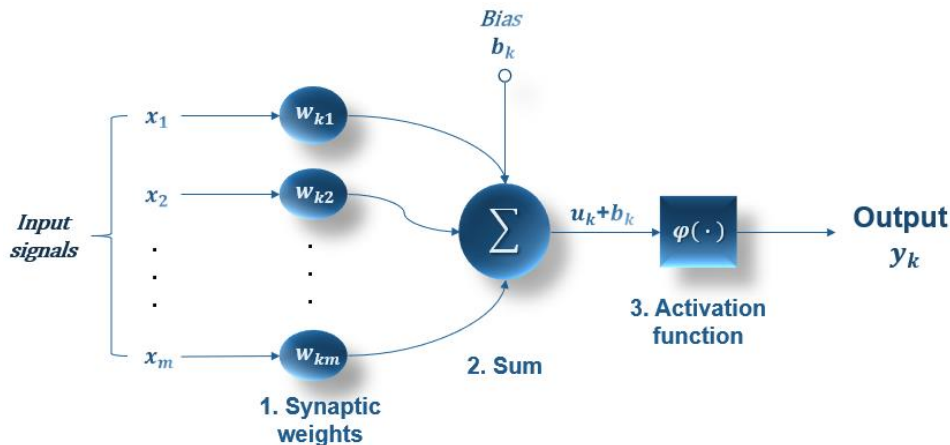


Figure 30: Model of an artificial neuron.

The signal x_j is the input signal x of the synapse j connected to neuron k . The u_k value is the linear combination of each input signal multiplied by respectively synaptic weight w_{kj} . The result of the linear combination is called potential activation. $\varphi_k(\cdot)$ is the activation function in the neuron k , which control the output behavior. And y_k is the final output signal of the neuron. In mathematical terms, we may describe the neuron model as:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (24)$$

and:

$$y_k = \varphi_k(u_k + b_k) \quad (25)$$

A bias b_k applies the effect of increasing or lowering the potential activation of neuron, modifying u_k by an affine transformation in the manner illustrated in Figure 28. We can observe several that Equation 21, in Linear Regression Section, and Equation 23 are equivalents; they perform the same computation. We can observe it also in Equations 22 and Equation 24. Artificial neurons can compute regression or classification and choosing the appropriate activation function is enough for it.

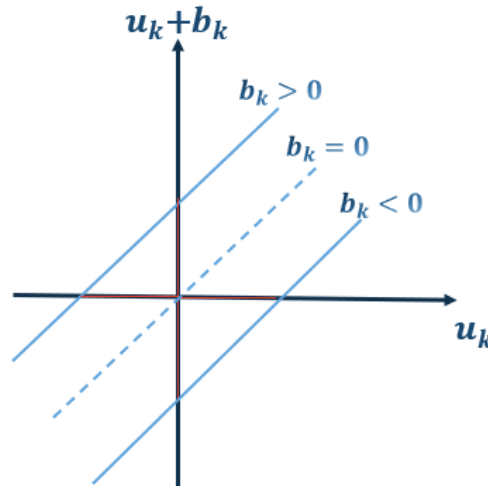


Figure 31: Illustration of bias modifying a potential neuron activation by an affine transformation.

Perceptron is the simplest artificial network, which $k = 1$. It is worth noting that the perceptron has two layers, even though the input layer does not do any calculation and merely communicates the feature values. The number of layers in a neural network does not include the input layer. Because the perceptron has just one computational layer, it is classified as a single-layer network [32].

2.2. Activation Function

The differentiable activation functions enabled the use of the backpropagation method for computing the gradient of the error function concerning the weights, which enabled the use of the gradient descent algorithm to compute the optimal weights [63]. Different choices of activation functions can be used to simulate different types of models used in machine learning. If the target variable to be predicted is real, then it makes sense to use the identity activation

function, and the resulting algorithm is the same as regression. If it is desirable to predict a probability of a binary class, it makes sense to use a sigmoid function for activating the output node so that the prediction \hat{y} indicates the probability that the observed value, y , of the dependent variable is 1 [32].

The most basic activation function $\varphi_k(\cdot)$ is the identity or linear activation often used in the output node when the target is a real value (regression tasks). The sigmoid activation outputs a value in $(0, 1)$, which helps perform computations that should be interpreted as probabilities (binary classification tasks). The relu activation the activation is thresholded at zero, but is the activation function most used in neurons of more complex networks because was found a greatly accelerated time (e.g. a factor of 6) in the converge of these networks [42, 45, 70].

Taking $v_k = u_k + b_k$ for simplifications, the identity, sigmoid and relu activation functions can be expressed the Equation 25, 26, and 27, respectively. The Figures 32(a), 32(b), and 32(c) illustrate the behavior these functions.

$$\varphi_k(v_k) = v_k \quad (26)$$

$$\varphi_k(v_k) = \frac{1}{1 + e^{-v_k}} \quad (27)$$

$$\varphi_k(v_k) = \max(0, v_k) \quad (28)$$

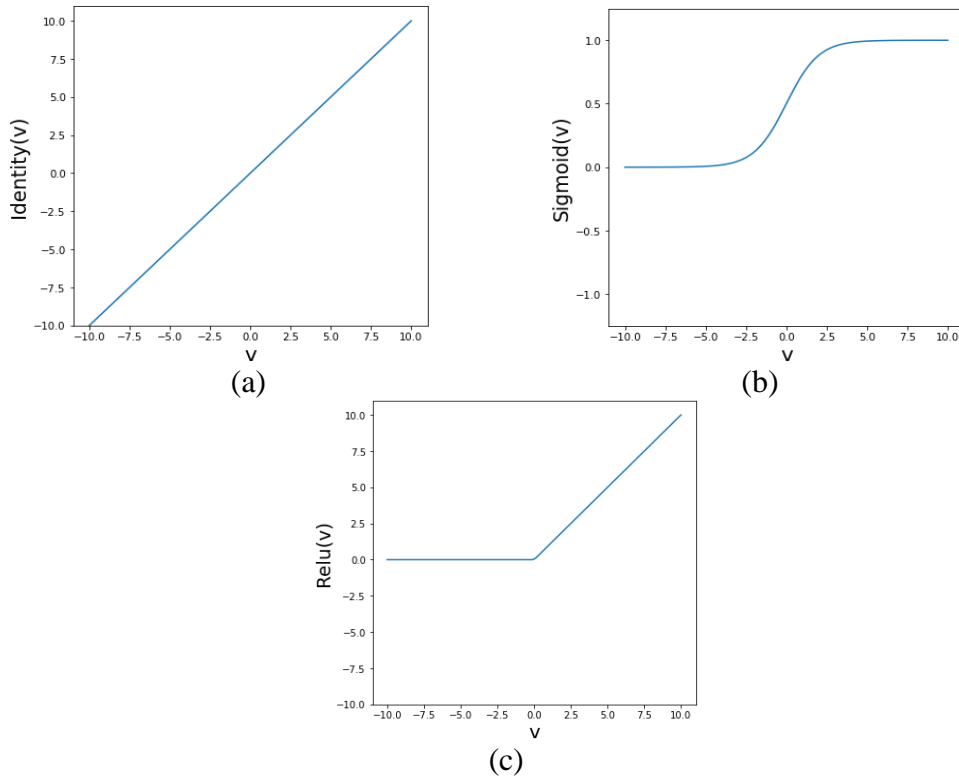


Figure 32: (a) Identity activation function. (b) Sigmoid activation function. (c) Relu activation function.

2.3. Loss Function

Regression and classification algorithms need to adjust their params when the input processing result is not equal to the output target value. These machine learning algorithms calculate the difference between input and respectively desirable output and use it for parameter adjusting to accomplish the adjusts. The goal is to minimize the error obtained. The algorithm calculates the error through functions called loss, error, or cost function [43, 70-71].

The minimum for which the value of the error function is smallest is called the global minimum, while other minima are called local minima [71]. The loss function needs are continuously differentiable concerning the weight vector w , and this should have few or no local minima and be a convex function [33]. Loss function ability the machine algorithm to measure its performance and decide how it evaluates its parameters. The loss function localization on artificial neural network parameters adjusting is illustrated in Figure 33.

There are many other possible choices of error function which can also be considered, depending on the application. For regression problems, the fundamental goal is to model the conditional distribution of the output variables conditioned on the input variable. Therefore, the use of a sum-of-squares error function is motivated. For classification problems, the goal is to model the posterior probabilities of class membership conditioned on the input variables. For it, more appropriate error functions can be considered [71].

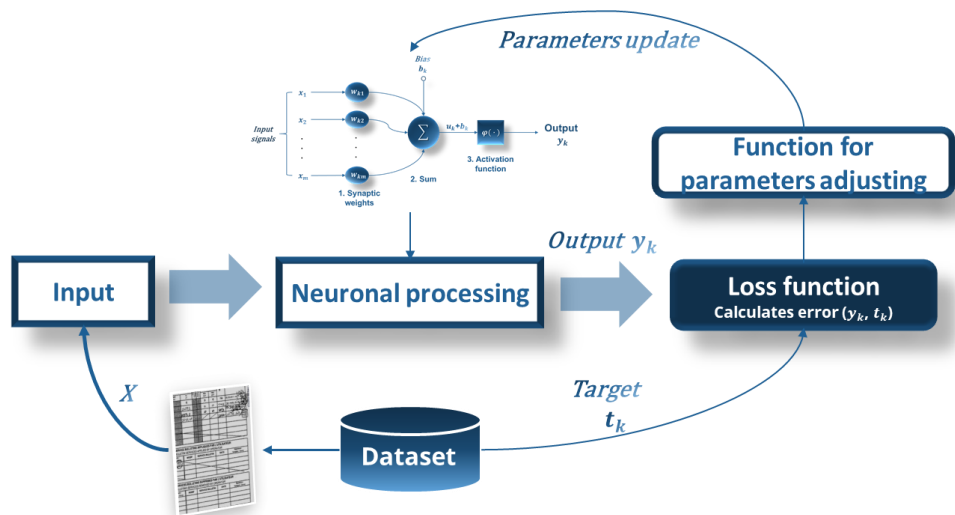


Figure 33: (a) Identity activation function. (b) Sigmoid activation function. (c) Relu activation function.

For example, consider a simple linear regression task with numeric outputs. It requires a simple sum-of-squares error function for an input with prediction y and target t . Let us take the Equation 28 in a simplified form, and we calculate it

for an entire dataset as:

$$E = \frac{1}{2N} \sum_{n=1}^N (y_n - t_n)^2 \quad (29)$$

On binary classification problems, a standard loss function used is cross-entropy. Cross-entropy is a quantity from the field of Information Theory that measures the distance between probability distributions or, in this case, between the ground-truth distribution and the predictions [69]. However, it is essential to mention that different choices of error function arise from different assumptions about the form of the conditional distribution [71]. The more suitable may be chosen according to the task.

Using cross-entropy loss by an instance n , we can calculate the error as:

$$E_n = \begin{cases} -\log(y_n) & \text{if } t_n = 1 \\ -\log(1 - y_n) & \text{if } t_n = 0 \end{cases} \quad (30)$$

The Figure 34(a) and 34(b) illustrate the behavior of first and second conditions, respectively. The intuition is that when $y_n = 0$ but $t_n = 1$ the resulting cost is large, and little if both are equals. The same logic is true when $y_n = 1$ but $t_n = 0$.

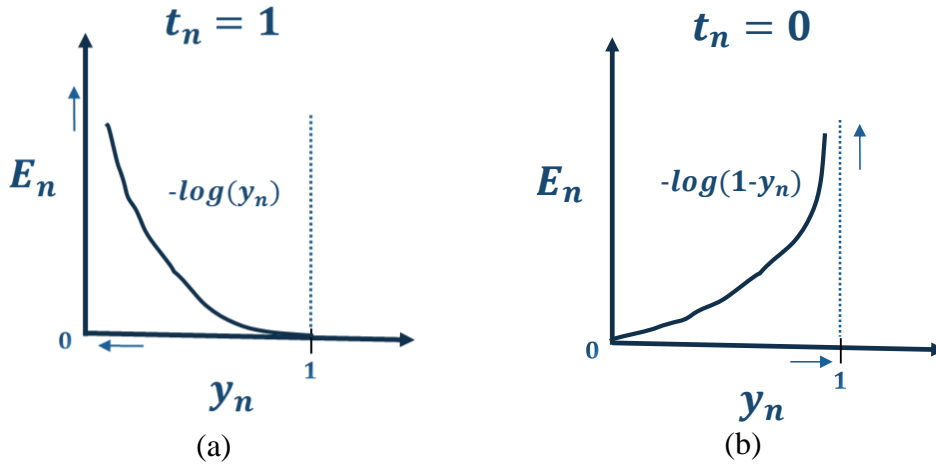


Figure 34: Cross entropy conditions. (a) $t_n = 1$. (b) $t_n = 0$.

For all samples of a dataset D , we can combine these two conditions as:

$$E = \frac{1}{N} \left[\sum_{i=1}^N t_n \log y_n + (1 - t_n) \log(1 - y_n) \right] \quad (31)$$

The intuition this way is that if $t_n = 0$, only the second term of the sum influences in the loss. If $t_n = 1$, only the first term of the sum influences in the loss.

2.4. Hyperparameters Tuning

One challenging and time-consuming step of designing neural networks is defining and tuning their hyperparameters to find the optimal configuration. Hyperparameters are settings that can be used to control the behavior of the learning algorithm. Its values are adapted manually or by external algorithms since it is not appropriate to learn on the training set because it is challenging to optimize [70]. Networks designers spend many times repeatedly modifying the model, train it, evaluating validation data (not the test data, at this point), modifying it again, and repeating until the model is as good as it can get. The process of optimizing hyperparameters typically looks like this [69].

- Choose a set of hyperparameters;
- Build the corresponding model;
- Fit it to the training data and measure the final performance on the validation data;
- Choose the next set of hyperparameters to try;
- Repeat;
- Eventually, measure performance on your test data.

It is essential to observe that hyperparameters should not be tuned using the same data used for gradient descent. Instead, a portion of the data is held out as validation data, and the model's performance is tested on the validation set with various choices of hyperparameters. This type of approach ensures that the tuning process does not overfit the training data set (while providing poor test data performance)[32]. Some common hyperparameters to set are:

▪ Learning rate

It controls the weights update step. When it is small, the transient response of the algorithm is overdamped. When employing high learning rates, a positive feedback loop might occur in which big weights cause large gradients, which then cause a significant update to the weights. When it exceeds a specific critical value, the algorithm becomes unstable (i.e., it diverges) [43, 70].

Another adverse event where a high learning rate is used is the problem of dying neurons on relu activation functions. In such a case, the pre-activation values of the relu can jump to a range where the gradient is 0 irrespective of the input. In

other words, high learning rates can “knock out” relu units. In such cases, the relu might not fire for any data instance. Once a neuron reaches this point, the loss gradient concerning the weights just before the relu will always be zero. In other words, the weights of this neuron will never be updated further during training [32].

▪ Momentum

Momentum has beneficial effects on the algorithm learning behavior, addressing local minima and convergence speed. Local minimum can occur because of small learning rate, small but consistent gradients, or noisy gradients, and in the face of high curvature. The momentum term preventing the learning process from terminating in a local minimum by moving each step based not only on the current slope value but also on the past updates. By accumulating an exponentially decaying moving average of past gradients, the momentum term sometimes acts as a friction parameter, smoothing zigzagging moves [42-43, 69-70].

Therefore, the learning process is moment-based is better because it gives greater preference to consistent directions over multiple steps (horizontal steps) and penalizes useless “sideways” oscillations (steep steps). This allows the use of more significant steps in the correct direction without causing overflows or “explosions” in the lateral direction, resulting in an accelerated learning process [32]. Equation 31 can express the weights update using gradient descent with momentum:

$$\begin{aligned} V_n &= \beta V_{n-1} + (1 - \beta)dw \\ W &= W - \alpha V_n \end{aligned} \tag{32}$$

▪ Regularization

Regularization is one of the central concerns of machine learning, and it consists of any modification we make to a learning algorithm intended to mitigate overfitting [70]. In general, it is more desirable to use complex models (e.g., more extensive neural networks) with regularization rather than simple models without regularization. Weight regularization is a type of regularization that constrains on a network's complexity by forcing its synaptic weights to take values close to zero [43, 69]. It penalizes large (absolute) values of the parameters more than small values [32].

This work uses a particular weight regularization called weight decay or L2 regularization. The cost added is proportional to the square of the value of the weight coefficients (the L2 norm of the weights) [69]. The λ value is used to control the strength of weight decay, which acts as a capacity hyperparameter. Increasing or decreasing the value of λ controls the complexity of the model leads to a simpler

model. This parameter provides greater flexibility by providing a tunable parameter chosen in a data-driven manner [32].

This way we can update the Equations COST by:

$$E = \frac{1}{N} \left[\sum_{i=1}^N t_n \log y_n + (1 - t_n) \log(1 - y_n) \right] + \lambda \sum_{j=0}^D w_j^2 \quad (33)$$

where D is the dimensionality of network parameters, w_j is the value of parameter j , and λ is the control hyperparameter of regularization. For any given weight in the neural network, the updates are defined using gradient descent:

$$w_j = w_j(1 - \alpha\lambda) - \alpha dw_j \quad (34)$$

2.4.1. Batch Normalization

Batch normalization is a method of adaptive reparameterization, motivated by the difficulty of training profound models, and can be applied to any input or hidden layer in a network. This method can address the vanishing and exploding gradient problems and reduce covariate shifts. In covariate shift, the parameters change of the hidden inputs change during training from early layers to last layers, and it causes slower convergence during training because the training data for later layers are not stable. Batch Normalization adaptively normalizes data even as the mean and variance change over time during training. It works by internally maintaining an exponential moving average of the batch-wise mean and variance of the data seen during training [42, 69-70].

In batch normalization, the idea is to add additional “normalization layers” between hidden layers that resist this type of behavior by creating features with somewhat similar variance [32]. According to Goodfellow et al. [70], Let H be a mini batch of activations of the layer to normalize. To normalize H , we replace it with:

$$H' = \frac{H - \mu}{\sigma} \quad (35)$$

where μ is a vector containing the mean of each unit and σ is a vector containing the standard deviation of each unit. Within each row, the arithmetic is elementwise, so $H_{i,j}$ is normalized by subtracting μ_j and dividing by σ_j . The rest of the network then operates on H' in the same way that the original network operated on H . At training time,

$$\mu = \frac{1}{m} \sum_i H_i, \quad (36)$$

And

$$\sigma = \sqrt{\delta + \frac{1}{m} \sum_i (H - \mu)_i^2}, \quad (37)$$

where δ is a small positive value such as 10^{-8} imposed to avoid encountering the undefined gradient of \sqrt{z} at $z = 0$. Crucially, we back-propagate through these operations to compute the mean and the standard deviation and apply them to normalize H . At test time, μ and σ may be replaced by running averages collected during training time. This allows the model to be evaluated on a single example without using definitions of μ and σ that depend on an entire mini batch [70].

Appendix 2

1. Extra Experiments

We performed new tests combining more complex architectures and other cost functions to assess whether the latter could improve the results. We keep the rest of the network parameters unchanged. The L2 and GIOU cost functions did not improve the results in our experiments. Regarding the new architectures used, we experimented with deeper architectures based on resnet100, cascade architectures, and inceptions modules (configuring cascade and resnexts in Table 3, respectively). The literature points out that waterfall architectures can improve results in an object detection task by evaluating different IoU thresholds between network predictions and manual markings [65]. The inceptions modules [80] combines the extraction of different resolutions from feature maps. The optimized combination of these modules offers greater power to extract features and, at the same time, reduce the number of parameters to be trained by the network.

We believe that the L2 cost function did not bring better results due to the complexity of our database, especially concerning the different cases of overlap and the variety of classes with few samples, which are configured in outliers. According to [65], the L2 cost function is not a good choice when there are outliers in the database. The GIOU cost function was designed for the network to evaluate better cases in which there is no overlap between the network predictions and the manual markings belonging to the dataset [81]. However, we did not obtain better results using this metric. Regarding the other architectures tried, the best result obtained only 2% more than our previous result.

Deformable Transformers [82] is an end-to-end object detector that is efficient and fast converging. It converges on a few epochs compared to other object detectors and uses multi-scale deformable attention modules, an efficient mechanism for processing image feature maps. Focal loss [83] is pointed out in the literature as an efficient cost function addressed to the extreme foreground-background class imbalance. However, we believe that this cost function did not show significant improvement due to RPN internally minimizing this imbalance problem's effects [12], [56].

Table 11 shows our best result achieved without using Instance Augmentation and the results obtained from the new experiments performed. We compare the results on the application of the different experiments based

on validation. We evaluated the AF-score metric by combining COCO AP50 and Recall metrics. At first, the results showed us that the new experiments did not provide significant improvements. In our preliminary analysis performed visually, we verified that the new experiments were not successful, almost always related to the same classes of the previous experiments: those that do not have examples in the training group. We will further investigate this fact by generating similar synthetic data from original examples in future works.

Table 11: Comparison of the results for stamps detection using different network architectures and cost functions from extra experiments.

			Metrics (%)				
		backbone / loss	Priority	AP50	Recall	F1	
Baseline	Backbone size	resnet50 / L1	AP50	0.984	0.809	0.888	
			Recall	0.984	0.809	0.888	
New tests	ResNet50	resnet50 / L2	AP50	0.984	0.778	0.869	
			Recall	0.982	0.803	0.884	
		resnet50 / GIOU	AP50	0.977	0.769	0.861	
			Recall	0.973	0.799	0.877	
		resnet50_cascade / L1	AP50	0.983	0.788	0.875	
			Recall	0.975	0.822	0.892	
		resnet50_cascade / GIOU	AP50	0.978	0.757	0.853	
			Recall	0.965	0.794	0.871	
		resnet50_deformable transformers / Focal	AP50	0.970	0.815	0.886	
			Recall	0.970	0.815	0.886	
		ResNet100	resnet100 / L1	AP50	0.983	0.786	0.874
				Recall	0.975	0.815	0.888
			resnetx100 / L1	AP50	0.985	0.785	0.874
				Recall	0.976	0.819	0.891
			resnext101 / GIOU	AP50	0.986	0.791	0.878
				Recall	0.808	0.966	0.880
			resnetx100_cascade / L1	AP50	0.987	0.806	0.887
				Recall	0.986	0.841	0.908