Ca' Foscari
University
of Venice

Master's Degree in Digital and
Public Humanities

Final Thesis

Machine Learning and Computer Vision in the Humanities

Supervisor
Ch. Prof. Massimo Warglien

Assistant supervisor
Ch. Prof. Giorgia Minello

Graduand
Federico Dassiè
Matriculation Number 861867

Academic Year
2021 / 2022

List of Figures

Table of Contents

Abstract

We will discuss the ways in which the world of Machine Learning and Computer Vision are presenting themselves nowadays and how, in our case, they have been of fundamental help in improving the work experience and above all guaranteeing better technical efficiency. Preserving heritage, whether it will be a physical or digital aspect, is and always will be a difficult goal to reach. The main problem is given by the vastness of materials of different types, which make treatment difficult and lengthen the times. Since it is impossible or at least exhausting to carry out all the processing work by hand, image by image, it is necessary to be able to automate the process. The thesis focuses on the work carried out at the Giorgio Cini Foundation, which includes a chain of automatic steps that starts from the raw image and arrives at its complete final analysis, passing through a post-production phase and the creation and use of an Object Detection model. The final result are the same starting images, now corrected according to different transformations and parameters, with a series of statistical information about their contents, both ready to be published for a broader public.

Keywords: Deep learning, Machine learning, Convolution neural network (CNN), Deep neural network architectures, Deep learning applications, Object detection, Transfer learning, Computer Vision, Residual Learning, Detection

1 Introduction

Everything started from the need, on the part of the ARCHiVe center (the digital laboratory for the artifacts preservation of the Cini Foundation), of a new automated system for post-processing the product images of scanned objects, in order to publish them in the digital library of the Foundation. These images are organized in funds (groups of images belonging to the same author) based on their origin and timeframe, which can contain from a few hundred to tens of thousands of files. They contain many different kind of materials, including printed texts, manuscripts, passports, photographs, multimedia files and so on. If the scanning phase is a part that necessarily requires manual work, since it is the physical use of digital imaging machines to obtain the digital conversion of real artifacts, the post-production of the images can be carried out automatically, thus aiming to reduce time and stress on the part of the operators, also reducing as much as possible any margin of human error.

But what is the post-production phase of these images based on? The shots, although they are made in the best possible reproducible environment, are sometimes imperfect as they do not guarantee 100%, for example, the presence of the original colors (altered by imperfections in the lights). Other problems concern the containment of the main object within the photograph, often placed in the middle of a too large background, or the need to carry out further operations of a practical nature such as compression (for previews and thumbnails) and the management of metadata, including measurements of the object.

It was therefore necessary to organize a workflow and understand how to overcome the individual needs, in order to create a product capable of effectively replacing the constant presence of users involved in image processing.

In addition to this, this work is accompanied by the creation of a model of identification of objects - reusing the same images once automatically processed -, able to distinguish different features and allow us to know their number and their position (ideally the input is represented by the ordered pages of a book just processed).

1.1 Cini Foundation and ARCHiVe

The Giorgio Cini Foundation was created in 1951 by Vittorio Cini to commemorate his son Giorgio, who had died tragically in an air crash on August 31, 1949. The Foundation is the earliest example in Italy of a private organization whose principal aims included humanistic studies at a time when collective interest was focused on economics, science, and technology. Vittorio Cini's decision to make the Island of San Giorgio Maggiore the setting for his Foundation was a remarkably inspired move.

In fact, by choosing the island as the headquarters of the Foundation and including its restoration and redevelopment among the Foundation's statutory

aims, Vittorio Cini made his creation an heir to a centuries-old tradition: he gave it the historical role and vocation of the monastery whose legacy he wished to perpetuate.

The Foundation's initial projects were focused on solving some of the urgent problems afflicting Italy and Venice in the postwar period. On the one hand, there was a need to train young people professionally and, on the other, there was a dearth of resources and facilities dedicated to scientific and cultural research. For these purposes, two institutions were created: the Centro Marinaro (Nautical Center) and the Centro Arti e Mestieri (Crafts and Trades Center). For many years these institutions provided hundreds of young men with professional training, especially for some of the more specific careers associated with the economic needs of the city. Around the same time, a Center for Culture and Civilization was created. It was dedicated to promoting high-standard scientific and academic events, such as international conferences focused on studying cultural, artistic, social, scientific and economic issues. Its activities also included organizing exhibitions and performances of plays and concerts.

Today the Giorgio Cini Foundation is an internationally recognized cultural institution which continues to draw inspiration from its original vocation and at the same time plays a leading role as a center for studies and a venue for meetings and debate on current issues. Its academic activities – research projects and events aimed at improving our "knowledge of the world" – continuously interact with thinking on the current political and social scene aimed at promoting multidisciplinary approaches and experimenting with exchanges between forms of knowledge and various scientific and professional cultures. The intention is thus also to provide innovative tools for analysis and comprehension, as well as propitious opportunities for "building a new world."

The Giorgio Cini Foundation is not only an invaluable store of literary, artistic, musical, and archive treasures, but also a crossroads for cultures and ideas and a place devoted to the quest for truth and the spread of knowledge, thus perpetuating the tradition of the Benedictine abbey and the Island of San Giorgio Maggiore. In line with the policy traced out by its founder, the Foundation has initiated a new strategy for the development of the island through the creation of the Vittore Branca International Center for the Study of Italian Culture.

The Center is dedicated to the memory of the great Italianist and the Foundation's first Secretary General who, together with Vittorio Cini, was the main driving force of the Venetian institution in its first fifty years of activity.

The Vittore Branca Center is inspired by the great schools of the past and at the same time provides a place for meeting and study both for young researchers and senior scholars interested in furthering their knowledge in a field of Italian culture (especially the culture of the Veneto), such as the visual arts, history, music, literature, drama, the comparative civilisations and spritualities. With specially designed residential facilities on the island of San Giorgio, the Vittore Branca Center offers scholars from all over the world the opportunity to work and stay at length in Venice in a privileged setting particularly conducive to reflection and study. Thanks to the Branca Center, the Island of San Giorgio Maggiore and the Giorgio Cini Foundation have strengthened their bonds in

a mutual, inextricable relationship and have renewed their ultimate mission, which is to encourage the free circulation of ideas and knowledge and all forms of learning.

The ARCHiVe Center (Analysis and Recordings of Cultural Heritage in Venice), founded in 2018 by the Giorgio Cini Foundation, the Factum Foundation for Digital Technology in Conservation and the Digital Humanities Laboratory of the École Polytechnique Fédérale de Lausanne (EPFL-DHLAB), was born with the aim to pay particular attention to the development and use of new technologies for digital preservation and for the enhancement of the immense heritage present on the island of San Giorgio Maggiore in Venice, but not only.

ARCHiVe aims to raise awareness among the international community on the importance of digital technology as an indispensable means for the conservation of cultural heritage in the twenty-first century for the future of knowledge. The Center is a hub for academic and cultural institutions wishing to apply and develop emerging technologies and intelligent computer vision software. These tools are at the service of its main objective, that of collecting a large archive of digital data and keeping them safe for future generations.

In collaboration with the research institutes of the Cini Foundation, ARCHiVe is responsible for enhancing and converting the rich documentary heritage of the Foundation itself into digital, to study - involving other institutions and scholars - new methods to preserve and transmit it thanks to digital technologies and, finally, to propose training activities with a professionalizing value in the Digital Cultural Heritage sector. [15]

2 Literature Review

Humanities and Machine Learning are not the most obvious combination for many reasons: frameworks have made sure that ML algorithms are easy to use and they actually have a super-low entry level programming-wise thanks to all those libraries, but the actual thing about ML is that it needs to be understood in order to be useful. In the Digital Humanities, a result is only a result if we can explain how we got there using classical Humanities hermeneutics. However, most computational methods don't exactly lend themselves to being explained in classical Humanities terms. So technically, ML results aren't really results in the context of the Humanities. In order to be able to actually use ML in Digital Humanities research, we need to learn how we can validate the results ML gives us in terms of the hermeneutics of the Humanities and understand the algorithms it uses.

Machine Learning, text mining and object detection approaches appear to offer a compelling complement to traditional analysis, by having the computer shift through massive amounts of information looking for "suggestive patterns". The power of modern machine learning systems to uncover patterns in large amounts of data has led to their widespread use in many applications, from spam filters to analyzing sequences. And the potential for using these sophisticated algorithms to find meaningful patterns has been recently observed. Our current work is to design sets of relatively constrained experiments using those systems on specific problems in order to examine what works, what does not work, and just what such results might mean [43].

There has been a relatively long history of the use of computational technologies to facilitate the analysis and understanding of cultural and heritage objects. This can be accomplished through digital imaging in two ways: either the nature of the capture of the digital image can be adjusted through advanced digitization, allowing the creation of digital images, which expand our understanding by capturing information the human eye or brain is not capable of processrng independently; or digital images can be analysed and interrogated computationally through image processing to reveal patterns or features in the image, which would not otherwise be noticeable. The two procedures are often linked, with specific digitization processes carried out to provide high resolution digital images, which can then be used as the basis for image processing analysis. Image processing can also be used to help sort collections of images, in order to search for images which have specific features. Various image processing techniques have been used within the arts, heritage and culture domain since the early 1980s, when early pilot digitization projects explored the technicalities and possibilities of digitization and image processing [50]. Reflecting on projects from that period, research tended to be focused on a specific object or collection, and the techniques employed varied, according to the problem in hand.

If Image processing has been applied to humanities research for so long, why is it not better known, or part of most syllabi, for teaching Digital Humanities?

Why have most people undertaking DH research never undertaken any image processing? Why are these techniques not better understood, or better known, within the DH community?

Firstly, unlike technological approaches, such as text encoding or digitization, there are no guides to good practice in image processing; there can be no guides, as each application is bespoke, depending on the research question and the individual image requiring analysis. This means that there is no simple 'painless introduction'. To apply image processing methods to a research question, an individual must have experrence in mathematics, programing and image processing applications. The tools currently available to facilitate research in this area are either limited — tending to be tied to specific research projects, in particular research groups — or fairly technical, with a high barrier to entry. Image processing thus remains sequestered to the pantheon of tools available to computer and engineering scientists only, as most humanities scholars do not have the technical background to apply the techniques themselves.

Secondly, the previously mentioned projects all involved interdisciplinary research teams, where computing and engineering scientists advised humanities experts, curators and archaeologists on how to apply the image processing techniques in question. There can still be some resistance from traditional humanities scholars to interdisciplinary research teams, but for this kind of research, collaboration is essential. Those in the humanities who are interested in pursuing this area are advised to make contact with individuals undertaking 'vision' or 'imaging' research within the computational sciences, in order to establish partnerships. Finally, much of the use of image processing in the arts, humanities, library, archive and museum sectors borrows tools and understanding from computing and engineering science, which can be a problem for the latter working in those areas, if there is no room in a research project to develop new techniques and approaches. Some projects do develop novel imaging processing algorithms specific to their own application, which can, in turn, be applied in other areas. A humanities research project will be attractive to scientists only if there is room for a scientific research contribution. Finding interested partners who are willing to collaborate, and can see the potential of this type of research, is part of the problem [49, 47].

Machine Learning and Image processing usage examples will be now explained through some projects going on in the domain of Digital Humanities, With the aim of highlighting the importance of collaboration between Computer Science and Humanities.

2.1 Object Detection

An important aspect of ML regards Object detection in images. This project seeks to understand the purpose and origins of legal gestures depicted in medieval manuscripts, based on four illustrated manuscripts of Eike von Repgow's

Sachsenspiegel (Mirror of the Saxons), one of the oldest manuscripts on German law.

Figure 1: Labeled gestures

In Fig. 1 a set of gestures used to evaluate the detection system are shown. Pointing and swearing gestures differ by only a finger. The system differentiates between the front and back views of speaking gestures. After recognizing all the possible existing gestures follows the development of a detection system based on gestures squared contours, which have been preemptively labeled. The created detection system relies on the Laplacian of Gaussian to create high contrast responses along contours (Fig. 2).

Figure 2: Laplacian of Gaussian responses used for the detection

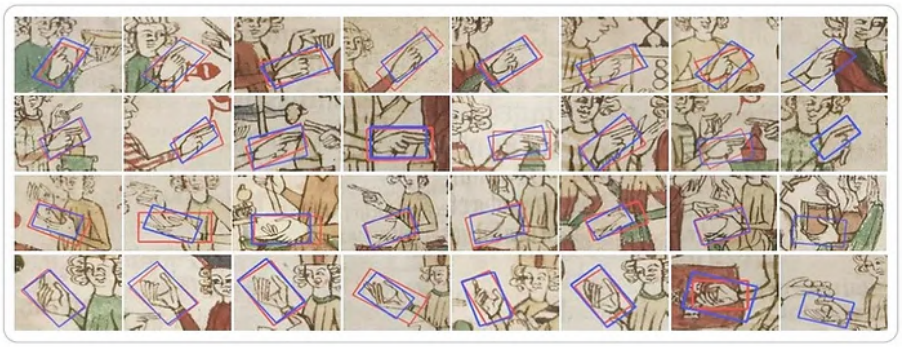

Figure 3: Final detection on the validation set

The model gets then trained with plenty of images, applying also data augmentation by scaling and rotating the gestures. Finally, testing the model on a validation set, made up with images not used for the model training, we can see the results of the gestures detection. In Fig. 3 are shown example of detected gestures (blue) compared to ground-truth bounding boxes (red). Each row shows detections from one type of gesture. The last column of each row gives an example false detection [41].

2.2 Text Recognition

One of the most notable use of ML technology is, undoubtedly, text-recognition. Starting from the first challenges that led to creation of the MNIST in 1998, the most famous database of handwritten digits that is commonly used for training various image processing systems [12], interest has grown over time expanding to many different fields, including archaeology.

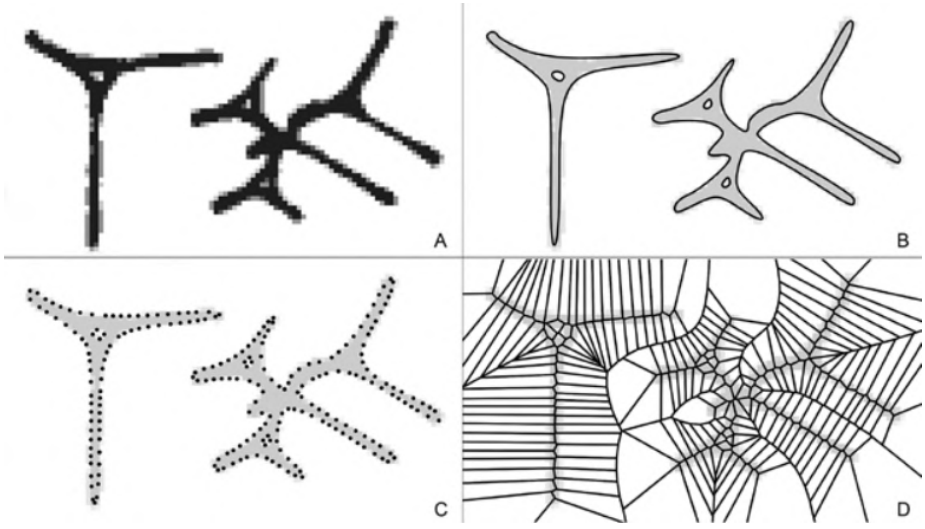

Figure 4: Vectorization and pattern matching

It has been the case, for example, of the analysis of cuneiform tablets. Cuneiform tablets are one of oldest textual artefacts comparable in extent to texts written in Latin or ancient Greek. Since those tablets were used in all of the ancient Near East for over three thousand years, many interesting research questions can be answered regarding the development of religion, politics, science, trade and climate change. These tablets were formed from clay and written

on by impressing a rectangular stylus, require novel methods in the field of computer science for documentation and analysis than methods used for flat objects i.e. ink on paper. Additionally, there are scarcely any methods in the field of Optical Character Recognition (OCR) that are applicable to language written in cuneiform.

The Assyrian language also has complicated grammar and numerous derivations where postfixes, prefixes and even infixes are added to verbs. There is no whitespace between cuneiform characters. It is therefore very hard to infer word boundaries since any sequence of words may look very similar to any other sequence of differently stemmed words. Common methods of word segmentation are thus not applicable.

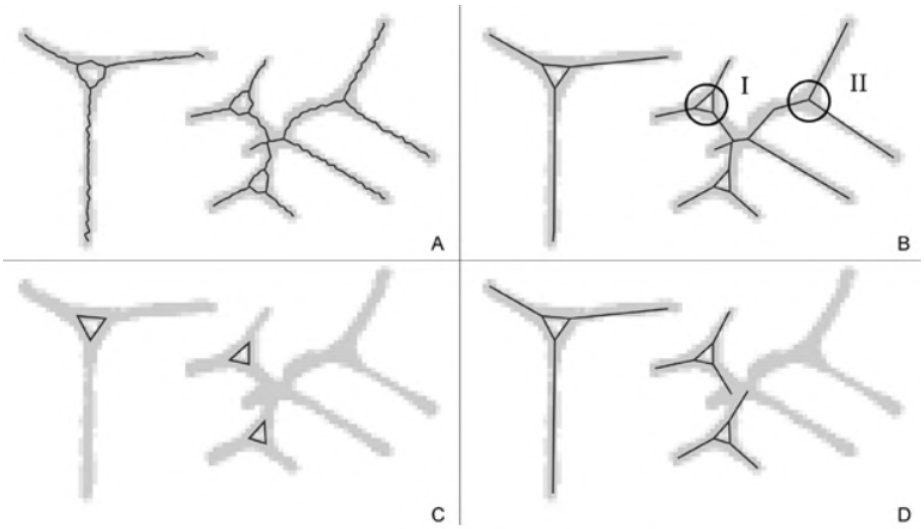

Figure 5: Building a simplified skeleton detecting wedge heads

The chosen workflow unifies retro-digitized tracings and born-digital tracings as well as 3D scanned meshes of cuneiform tablets. The result is a common and simplified representation of wedges and cuneiform characters:

- Retro-digitized tracings are scanned and then vectorized and skeletonized (Fig. 4-5). Wedges are extracted by detecting two types of wedge tracings, and then following the wedge-arms to points of high curvature;
- Born-digital tracings do not require any skeletonization and wedges are extracted directly. A conflict resolution step allows for significant oversegmentation of wedges, simplifying the extraction algorithm, and reduces the count of false positive wedges;

- Mesh data acquired by scanning cuneiform tablets with a structured light 3D-scanner is filtered using the MSII algorithm with the GigaMesh Software Framework. In the resulting feature space wedge configurations are extracted through thresholding and segmentation in 3D. The MSII filter is additionally used to skeletonize and extract wedges from the wedge-groups.

Despite the project is still in development, the team working on this is currently working on automatically classifying the wedges without prior domain knowledge, applying a pattern mining algorithm on the configuration of cuneiform character types to extract frequent configuration of wedges, cuneiform characters, usable as anchors for a subsequent cuneiform character search engine. The long-term goal of this work is the unification of source of cuneiform tablets and the creation of a database that can be queried with cuneiform character keywords [9].

2.3 Image Classification

Technological advances in digitization, automatic image analysis, and information management are enabling the possibility to analyze, organize, and visualize large cultural datasets. Being able to distinguish the representation of an image through its shape has been key for the analysis and recognition of different Mayan hieroglyphs.

The Maya civilization is one of the major cultural developments in ancient Mesoamerica. The ancient Maya language created uniquely pictorial forms of hieroglyphic writing [44]. Most Maya texts were written during the Classic period (AD 250–900) of the Maya civilization on various media types, including stone monuments, architectural elements, and personal items. A special class of Maya texts was written on bark cloths, made from the inner bark of certain trees (the main being the ficus tree), as folding books from the Post-Classic period (AD 1000–1519). Only three such books (namely, the Dresden, Madrid, and Paris codices) are known to have survived the Spanish conquest. A typical Maya codex page contains icons, main sign glyph blocks, captions, calendric signs, etc.

Unknown glyphs are considered as queries to match with a database of known glyphs (retrieval database), in which shape and context information are considered. To make sure errors are minimized, The clean raster images are produced by manually removing the background area from the raw images, whereas the reconstructed forms are generated by further carefully reconstructing the broken lines and missing strokes (Fig. 6). Glyph blocks are typically composed of combination of individual signs. The database contains 1,487 glyph examples of 892 different sign categories. Each category is usually represented by a single

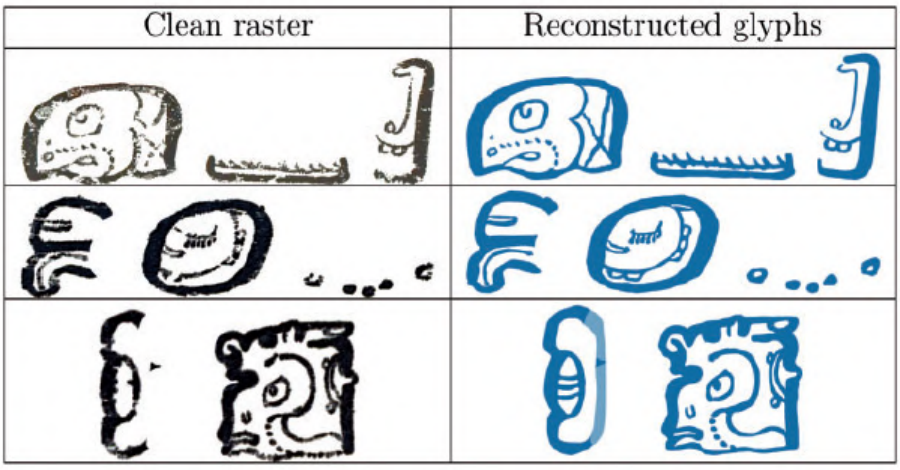

Figure 6: Example glyph strings generated from blocks

example image. Sometimes multiple examples are included; each illustrates a different visual instance or a rotation variant.

Feature extraction and similarity matching are the two main steps for the shape-based glyph retrieval framework. Glyphs are first preprocessed into thin lines (Fig. 7). To do so, an input glyph (A) is first converted into a binary shape (B). Thin lines (C) are then extracted through mathematical morphology operations. D and E shows the high-quality reconstructed binary image and the extracted thin lines. A local shape descriptor, called HOOSC, is then computed at a subset of uniformly sampled pivot points along the thin lines. It combines the strength of Histogram of Orientation Gradient [7] with circular split binning from the shape context descriptor [30]. Given a pivot point, the HOOSC is computed on a local circular space centered at the pivot's location, partitioned into rings and evenly distributed angles. Fig. 7 G-K shows different sizes of the circular space (referred to as spatial context), partitioned into two rings and eight orientations. A Histogram of Orientation gradient is calculated within each region. The HOOSC descriptor for a given pivot is the concatenation of histograms of all partitioned regions.

Follows a BoW approach (bag-of-words) where descriptors are quantized as visual words based on the vocabulary obtained through K-means clustering on the set of descriptors extracted from the retrieval database. A histogram representing the count of each visual word is then computed as a global descriptor for each glyph.

For each image the HOOSC descriptors are extracted and then glyph simi-

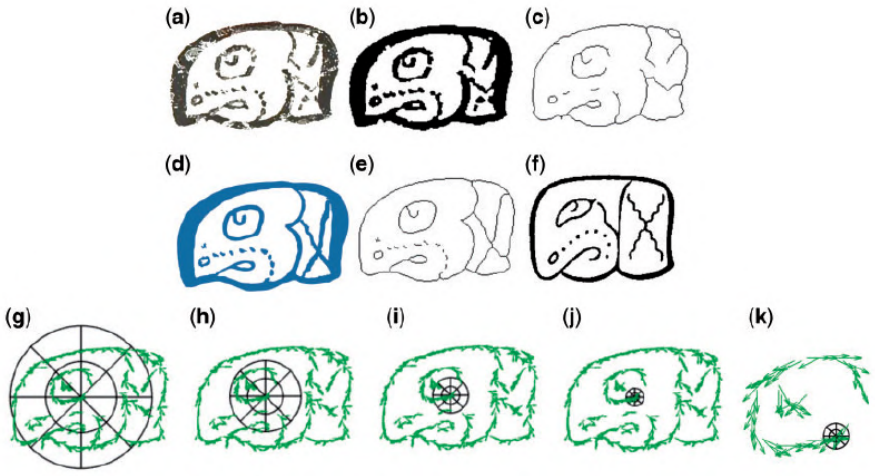

Figure 7: Glyphs processing

larity scores are computed between image pairs. To recap, the similarity measurement follows the following pattern:

- Each glyph image is preprocessed into thin lines; 400 evenly distributed points along the lines are randomly selected as pivot points. The HOOSC descriptor for each pivot point is then computed on a circular region centered at the pivot location, with a diameter equal to the mean of the pairwise distance between pivot points, which is partitioned into two rings and eight equally distributed orientations;
- The BoW pipeline is then applied to compute a global histogram representation indicating the count of each visual word. Finally, the pairwise glyph visual similarity is computed based on the L1 norm distance measure;
- To reduce the complexity of the final graph, the number of edges got reduced by applying a threshold to the similarity scores. Two glyph images are only considered to be similar if their visual similarity score is larger than a predefined threshold, and therefore their corresponding nodes in the graph are connected. Otherwise, the two glyphs are not considered to be sufficiently similar, and their corresponding nodes in the graph are not connected. A higher threshold indicates that fewer glyph pairs are considered to be visually similar, which results in fewer edges in the graph. In contrast, a smaller threshold indicates that more glyph pairs are considered to be visually similar, which leads to a more connected graph (more edges).

Given a glyph image database, the graph is initialized with random node positions and the precomp weights. The force-directed graph optimizes to posi-

Figure 8: Graph-based visualization interface

tion nodes of the graph so that there are as few crossing edges as possible, and all the edges are of roughly similar length. As a result, nodes in the graph often fall in several groups, each containing glyphs of visually similar patterns (Fig. 8). It can be seen that glyphs are grouped into visually similar clusters, such as vertical, horizontal, squared, and knotshaped patterns. This could naturally function as a shape-based glyph indexing system, which could help users quickly identify an unknown glyph based on its visual features [16].

2.4 Image Processing

Incised texts are often very difficult to read. Deciphering such texts draws on a wide range of expertise that papyrologists develop throughout their career. These skills are as much related to visual perception of the artefact and text as they are to palaeographical, linguistic and historical knowledge. Computational

tools can be used to aid papyrologists in their complex task, by providing alternative visual clues and evidence regarding textual incisions on documentary material. This case study presents methodologies that have been adopted to enhance the legibility of the Roman wooden stylus tablets from Vindolanda, on Hadrian's Wall, which consists of incisions left in wood through a now-perished coat of wax. The first step in deciphering these three-dimensional texts is by using advanced digitization techniques, called Shadow Stereo, and reflectance transformation imaging to capture and encode multiple images of the text under varying illumination conditions for further processing and visualization. ln addition, image processing algorithms were developed to isolate the features of the text and aid in their interpretation.

Digitization of such artefacts is the first step in the development of an interpretation of the document. Observing the experts trying to interpret the stylus texts informed us as to how the tablets could be digitized, whilst retaining, in particular, one major piece of information that papyrologists exploit when deciphering it — namely, the volumetric nature of the text. Experts who have access to the actual tablet they intend to transcribe have developed a very specific and intuitive strategy to enhance the visibility of the incisions that the stylus left in the wood. They lay the tablet flat on their hand, lift it up at eye level against a light source and apply pitch-and-yaw motions to the artefact. This effectively enhances the visibility of the incisions, by accentuating the highlights and shadows that the raking light generates; the lower the light, the longer the shadows projected by the text in (inverted) relief, created by the incisions carved on the surface of the tablet. The principle that is put into application here is the 'shadow-stereo' principle, by which concave shapes are revealed from shading, and the motion of the shadows exposes the location of the incisions [53]. This process can be digitally imitated: a digital camera is affixed above a tablet. and a high resolution digital picture is taken for each one of a set of pre-established positions of a light source around the tablet. Each light position is described by an elevation angle and an azimuth angle, where the azimuth angle corresponds to an angle deviation from the horizontal in the plane of the tablet and the elevation angle describes the height of the light, with respect to the plane of the tablet. Both angles are measured from the centre of the tablet. The collected data for each tablet was further used to digitally recreate the shadows and their motion. By adopting an appropriate model of image formation, not only can one store the set of images efficiently, without having to store each image, one can also interpolate between light positions, thus simulating lighting conditions, where a picture was not actually captured. The image model that is adopted relies on this technique of 'reflectance transformation' [25, 57]. This digital model can be remotely used by experts who do not have access to the original text, allowing them to digitally pitch-and-yaw this avatar of the artefact to aid them in interpreting the text: by exploiting the play of light with the 3D nature of the textual representation, we are able to enhance the text's legibility.

Image processing algorithms were developed to isolate textual features. Noise removal homogenizes the background of images by removing patterns or behaviours that are irrelevant; we can increase legibility. by neutralizing some of

the distracting noise present in the images, through homomorphic filtering and wood-grain removal. Image segmentation algorithms were then used to extract the meaningful areas or features of the image and, thereby, identify where the text is located on the complex, abraded surface of the texts. There is no room to provide adequate explanation of these techniques in this case study, but to summarize: background correction is first performed; then ways to achieve text feature extraction. Phase congruency, which exploits the fact that visual features are detected for some properties of the local phase of the image in the Fourier Domain, and Markov Random Fields, which take a statistical approach to region labelling for image segmentation, were employed. Explanation of these techniques can be found in Tarte et al. [46], for those interested in further detail.

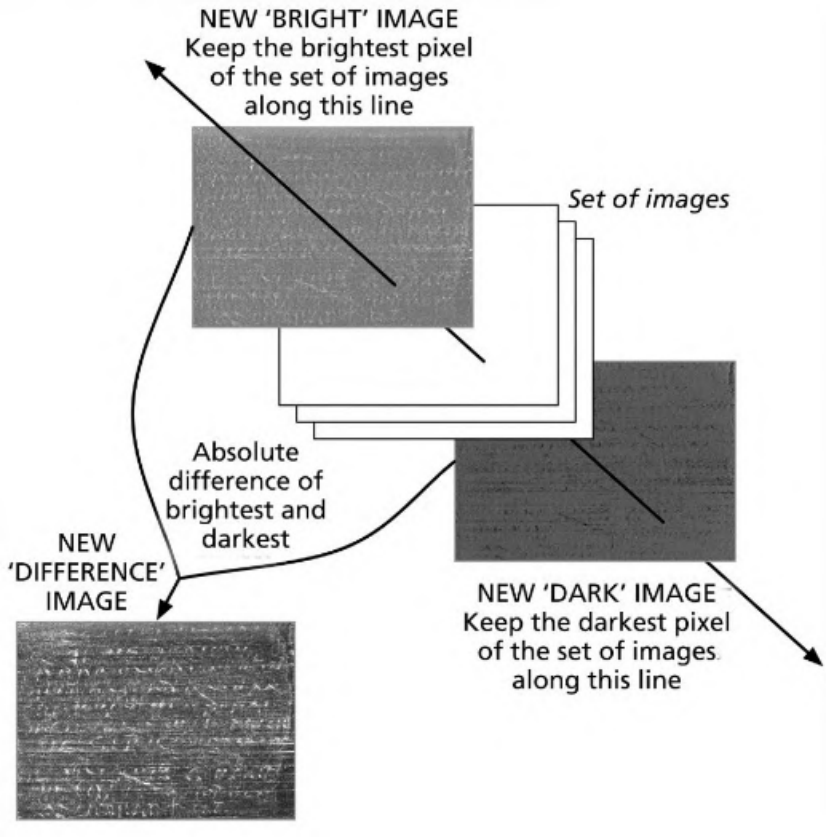

Figure 9: Example of image manipulation technique on a set of images of wooden Roman stylus tablets

Fig. 9 is an example of a simple image manipulation technique on a set of images of a wooden Roman stylus tablet, where the textual information is made

of the incisions in the wood. From a set of images captured from the same vantage point and with different light positions, one can enhance the visibility of the text, by extracting for every pixel position: the brightest pixel (where the tablet is lit the strongest) and the darkest pixel (where the shadows are the darkest). Due to the properties of light shone upon a 3D object, the pixel values on the flat textless areas remain approximately constant, whereas the pixel values at incisions vary widely. Consequently, subtracting the brightest and darkest images yields an image where the incised text (which is where the variations in light are the greater) is made more visible. Most techniques utilized in this project were inspired by approaches adopted by medical image processing. However, these methods had to be largely adapted to this specific application; in general the images of artefacts and their features of interest are not only different from, and noisier than, medical images, but, also, the type of visual expertise required to detect the textual features differs greatly from that of radiologists. The challenge of image processing and capture for ancient incised documents is multiple and specific to both the imaging method and the artefact. This research suggests that by observing and understanding the nature of the classicist's visual expertise, we will be able to integrate prior knowledge into a model of visual perception adapted to the classicist's needs, hence. supporting them in building meaning out of a pure signal — in building an interpretation of an artefact. Digital imaging and image processing can create novel, alternative visual clues as to writing contained within incised texts, assisting papyrologists in reading damaged, ancient texts [48].

2.5 Neural Network

Impressive classification results are obtained on painting databases by using convolutional neural networks (CNNs)[2, 51]. These results occur in a general context were methods of transfer learning [19] (changing the task a model was trained for) and domain adaptation (changing the nature of the data a model was trained on) are increasingly applied. Classifying and analysing paintings is of course of great interest to art historians, and can help them to take full advantage of the massive artworks databases that are built worldwide.

More difficult than classification, and at the core of many recent computer vision works, the object detection task (classifying and localising an object) has been less studied in the case of paintings, although exciting results have been obtained, again using transfer techniques [3, 58, 36]. Several works have tried to transfer the tremendous classification capacity of convolutional neural networks to perform cross-domain object recognition. In [2], it is shown that recycling CNNs directly for the task of recognising objects in paintings, without fine-tuning, yields surprisingly good results. More generally, the use of CNNs opens the way to other artwork analysis tasks, such as visual links retrieval [22], scene classification [11], author classification [21] or possibly to generic artwork

content representation [45].

The problem of object detection in paintings, that is, being able to both localise and recognise objects, has been less studied. In [3], it is shown that applying a pre-trained object detector (Faster R-CNN [34]) and then selecting the localisation with highest confidence can yield correct detections of classes.

Weakly supervised detection refers to the task of learning an object detector using limited annotations, usually image-level annotations only. Often, a set of detections (e.g. bounding boxes) is considered at image level, assuming it is known if at least one of the detection corresponds the category of interest. In order to perform transfer learning a Faster R-CNN (a detection network trained on photographs) is first applied, which is used as a feature extractor, in the same way as in [3]. This results in a set of candidate bounding boxes. For a given visual category, the goal is then, using image-level annotations only, to decide which boxes correspond to this category. Fig. 10 illustrates the situation faced at training time. For each image, a set of bounding boxes is given which receive a label +1 (the visual category of interest is present at least once) or -1 (the category is not present in this image).

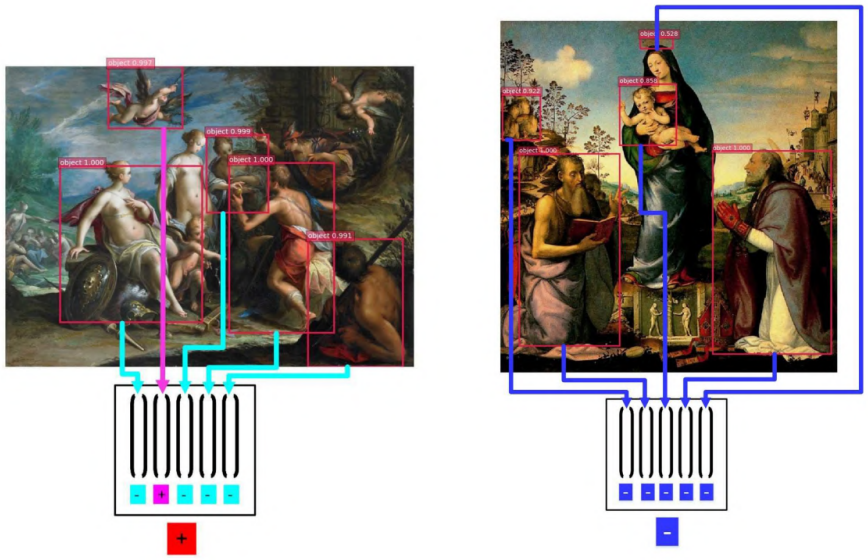

Figure 10: Illustration of positive and negative sets of detections (bounding boxes) for the "angel" category

In order to yield an efficient and flexible learning of new classes, retraining and even fine-tuning are avoided. Faster R-CNN is a meta-network in which a pre-trained network is enclosed. The quality of features depends on the enclosed

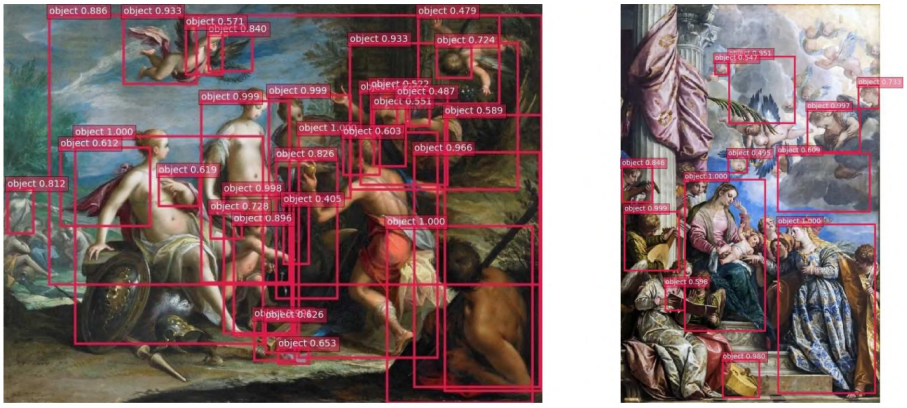

Figure 11: Some of the regions of interest generated by the region proposal part (RPN) of Faster R-CNN

network and several possibility in the experimental part are compared. Images are resized to 600 by 1000 before applying Faster R-CNN. Only the 300 boxes having best "objectness" scores are kept, along with their high-level features. An example of extracted boxes is shown in Fig. 11. About 5 images per second can be obtained on a standard GPU. This part can be performed offline since the network isn't fine-tuned. As mentioned in [66], residual network (ResNet) appears to be the best architecture for transfer learning by feature extractions among the different ImageNet models.

When a new class is to be learned, the user provides a set of weakly annotated images. Both the database and the library of classifiers can be enriched very easily. Indeed, adding an image to the database only requires running it through the Faster R-CNN network and adding a new class only requires a new training.

3 Background

3.1 Computer Vision

Perception is essential in order for any entity to interact in a meaningful way with its environment. Humans draw on many senses (such as sight, sound, touch and smell) to perceive the world. Most machines can only receive input through simple input devices, such as keyboards and mice, or through wired and wireless communication channels. However, in recent years, cameras and microphones have been added as standard parts of computers and mobile devices (such as phones and tablets). At the same time, the speed of these devices has increased significantly, making it possible to start to process this data in a meaningful manner. Computer Vision is about how we can automate image or video understanding on machines. It covers the techniques used to automate tasks ranging from industrial inspection (where the image understanding problem is constrained to one which we could easily address 20 years ago) to video understanding in order to guide autonomous robots so that they can interact in a meaningful and safe manner in a world designed for humans.

Computer vision has many applications in industry, particularly allowing the automatic inspection of manufactured goods at any stage in the production line. On the factory floor, the problem is a little simpler than in the real world as the lighting can be constrained and the possible variations of what we can see are quite limited. Computer vision is now solving problems outside the factory, like automatic reading of car licenses, augmenting sports broadcasts providing statistics, biometric security checks in airports, augmenting movies inserting visual objects into sequences, driving assistance, 3D model reproduction from photographs, advanced interfaces for gaming applications, plant classifications, buried landmines detection through infrared images and so on [6].

3.1.1 Images

An image is a picture (generally a 2D projection of a 3D scene) captured by a sensor. It is a continuous function of two coordinates in the image plane; usually expressed as (i,j) or (column, row) or (x,y). To process such an image on a digital computer it must be both:

- Sampled into a matrix (M rows and N columns);
- Quantised so that each element in the matrix is given an integer value. In other words, the continuous range is split into some number of intervals (k) where most commonly k = 256.

Digital images are created by sampling a continuous image into discrete elements. Digital image sensors consist of a 2D array of photosensitive elements, and each element (pixel/image point/sampling point) in the array has

some fixed area over which is it photosensitive. Typically, between elements there is some small (border) area which is not photosensitive, so it is possible (but not likely) that objects in the real world could be completely missed by the sensor if the light from those objects fell only in the border regions. The bigger issue, however, with sampling is that pixels represent the average value (luminance/chrominance) over a discrete area which in the real world could be projected from a single object, but equally (particularly at the boundaries of objects) could be a summation of light reflected from multiple objects. The number of samples in an image limits the objects that can be distinguished/recognised in the image. Hence, it is essential that the resolution (the number of pixels) is sufficient for our purpose (whatever that may be). At the same time, too high a resolution will have more detail than we need which may make processing harder and will definitely make it slower.

Figure 12: Four different samplings of the same image; top left 256x192, top right 128x96, bottom left 64x48 and bottom right 32x24

Each pixel in a digital image $f(i, j)$ is a function of scene brightness. The brightness values are continuous, but we need to represent them discretely using digital values. Typically, the number of brightness levels per channel is $k = 2^b$ where b is the number of bits (often 8). The question which must be asked is: how many bits are really necessary to represent the pixels? The more bits that are used the more memory the image will require; but as progressively fewer bits are used it is clear that information is being lost. Considering Fig. 12, you should perceive little different between the 8-bit and the 6-bit images although

the latter requires 25% fewer bits. There are clearly issues with the 4-bit and 2-bit images, although it is still possible for us to recognise many objects in these images. So it appears that the number of bits required actually depends on the purpose to which the image is to be put. If a machine is to automatically interpret a scene, though, we typically require more quantisation levels than might be expected, as otherwise false contours (e.g. in the sky in the 2-bit and 4-bit images in Fig. 13) and incorrect segmentation occurs.

Figure 13: Four different quantizations of the same grey-scale image; top left 8 bits, top right 6 bits, bottom left 4 bits and bottom right 2 bits

Images are central to computer vision, as they are the representation that we obtain from imaging devices (such as cameras). They provide us with a representation of the visual appearance of a scene, which we can process to enhance certain features of interest, before we attempt to abstract information. Images generally exhibit some degree of noise which can be attenuated by various simple image processing techniques [60].

Colour

Colour (multispectral) images have multiple channels, grey-scale (monochromatic) images (sometimes, incorrectly, referred to as black and white images) have only one channel. A grey-scale image represents the luminance (Y) at every point a scene. A colour image represents both luminance and chrominance (colour information) within the scene. This information can be represented in a number of ways but in all cases requires multiple channels of image data. Hence,

colour images (with the same sampling and quantisation) are bigger and more complex than grey-scale images, as we must somehow decide how to process each of the channels of information. Note that much of image processing was developed specifically for grey-scale images and its application to colour images is often not very well defined.

Computer vision was for many years based on grey-level images, judged easier to understand and less heavy compared to colored ones. However, colour does provide more useful information that can assist with many tasks, such as segmentation of the image into physically separate entities (objects, surfaces, etc.) [67, 69].

Colour images are more complex than grey-scale images and are typically represented using a three-channel colour space, a number of which are described below:

- RGB: the most common representation for colour images is to use three channels corresponding, roughly, to the red, green and blue wavelengths. What this means in effect is that the photosensitive elements in the camera are spectrally sensitive to wavelengths which are centred on those colours;
- CMY: The CMY model is based on the secondary colours (RGB are the primary colours), and is a subtractive colour scheme; that is the values of the C, M and Y are subtracted from pure white in order to get the required colour. For this reason, it is often employed as a colour model within printers where white is the starting point;
- YUV: The YUV colour model is used for analogue television signals and is comprised of luminance (Y) together with two colour components: blue minus luminance (U) and red minus luminance (V);
- HLS: The HLS model is frequently used in computer vision because, as well as separating the luminance and the chrominance, it separates the chrominance into hue and saturation, giving us quantities which we can talk about easily (e.g. dark blue, light red, etc.) [5].

Noise

One of the most commonly encountered type of noise is the Gaussian Noise. Gaussian noise is a good approximation to much real noise, modelled as having a Gaussian distribution around some mean (μ), which is usually 0, with some standard deviation (σ).

Noise must be joined with the image data in some way. The way in which we model this depends upon whether the noise is data independent or data dependent.

Figure 14: Colour and grey-scale images (left) with Gaussian noise added

In the case of data independent noise (i.e. noise where the amount of noise is not related to the image data itself), an additive noise model is appropriate:

$$f(i,j) = g(i,j) + v(i,j) \tag{1}$$

where $g(i,j)$ is the ideal image, $v(i,j)$ is the noise and $f(i,j)$ is the actual image.

In the case of data dependent noise (i.e. noise where the amount of noise is related to the image data itself), a multiplicative noise model is appropriate:

$$f(i,j) = g(i,j) + g(i,j).v(i,j) \tag{2}$$

In order to evaluate noise, we often need to simulate noise so that it can then be removed/reduced and the extent to which we are successful assessed. Assume that we are generating noise with a Gaussian distribution with a 0 mean and a standard deviation of σ. We first determine the probability distribution p(k) and the cumulative probability distribution $p_{cum}(k)$ of all possible values of the noise from the greatest possible negative change to the greatest possible positive change (typically k = -255..255).

$$p(k) = e^{-k^2/2\sigma^2}/\sigma\sqrt{2\pi}; k = -(G-1), ..., -1, 0, 1, ..., G-1 \tag{3}$$

$$p_{cum}(k) = p_{cum}(k-1) + p(k) \tag{4}$$

$$p_{cum}(-(G-1)) = p(-(G-1)) \tag{5}$$

Once the cumulative distribution has been determined, we can then compute a noise value for each pixel in the image. For every pixel (x,y):

$$f(i,j) = g(i,j) + argmin_k(rand() - p_{cum}[k]) \tag{6}$$

if $f(x,y) < 0$

$$f'(x,y) = 0 \tag{7}$$

if $f(x,y) > G-1$

$$f'(x,y) = G-1 \tag{8}$$

otherwise

$$f'(x,y) = f(x,y) \tag{9}$$

Note the argmin function gives the index of the smallest value, in this case choosing the k within the cumulative distribution whose value is closest to the random number. Also note that the truncation (to ensure that the values remain within 0...255) alters the Gaussian nature of noise somewhat.

Smoothing

Removing or, more likely, reducing noise can be achieved by a variety of methods, a number of which are presented here. Note that different techniques are appropriate in different circumstances, often due to assumptions underlying the techniques or the nature of the image data. The most commonly used approaches to noise reduction are linear smoothing transformations, where the computation can be expressed as a linear sum. Noise suppression using linear smoothing transformations typically results in the blurring of sharp edges. A number of nonlinear transformations (which cannot be expressed as a simple linear sum) have been proposed to deal with this. Generally, these transformations require that some logical operation/test is performed at every location in the image. None of the techniques described here can eliminate the degradations if they are large blobs or thick stripes. In such cases, image restoration techniques (in the frequency domain) may provide a solution.

3.1.2 Binary Vision

Grey-scale images generally have 8 bits per pixel. While processing these images is easier in some ways than processing colour images, there is a simpler form of image, the binary image, in which processing is even more straightforward. In fact, a significant portion of practical applications of computer vision have been developed using binary vision [42].

A binary image is one in which there is only a single bit per pixel (i.e. black or white). These images are created by thresholding where the thresholds used are determined in a variety of ways. The resulting binary images are often post-processed using mathematical morphology and the resulting segmented binary regions are extracted from the image using connected components analysis. It is

worth mentioning that binary images can be created from many types of image, such as intensity images, gradient images, difference images, and so on.

Thresholding

A binary image is created from a grey-scale image by thresholding. The binary thresholding algorithm is simply:

For all pixels (i, j)

$$\begin{aligned} f'(i,j) &= 1 \text{ where } f(i,j) >= T \\ &= 0 \text{ where } f(i,j) < T \end{aligned} \tag{10}$$

Often grey-level 255 is used instead of binary 1 (so that the resulting image can be represented using a 8-bit format and displayed/processed in the same manner as the original grey-scale image). The most efficient means of implementing this operation is to use a lookup table. In other words, create a one-dimensional array (the lookup table) with the same number of elements as there are grey scales. Each cell in the array should then be initialised to 1 or 0 depending upon whether its index is above or below the threshold.

The thresholding operation is generally used in order to separate some objects of interest from the background. Most typically the object(s) of interest are represented by 1 (or 255), but sometimes the binary image would have to be inverted for this to be the case.

Arguably, the most important thing to note about binary imaging is that the foreground and the background that are being separated need to be distinct in the image being thresholded. If they are not distinct then it will be difficult (or even impossible) to accurately segment them using thresholding. However, there are a number of techniques (e.g. adaptive thresholding) which are used to try to deal with situations where the distinction between foreground and background is not clear, and there are a number of techniques (e.g. erosion, dilation, opening, closing) which aim to improve a binary image where the segmentation is imperfect. Fig. 15 shows examples of what happens when the threshold is not chosen correctly. However, even in the 'correctly' thresholded binary image there are some errors (e.g. much of the tip of the pencil is missing and the shadow above the highlighter pen has been erroneously included).

Morphology

Mathematical morphology is an approach to image processing based on set operations typically describing the algebra of nonlinear operators operating on object shape. It provides a unifying framework for describing many image processing and analysis operations and has allowed the development of techniques which otherwise are relatively hard to describe [26].

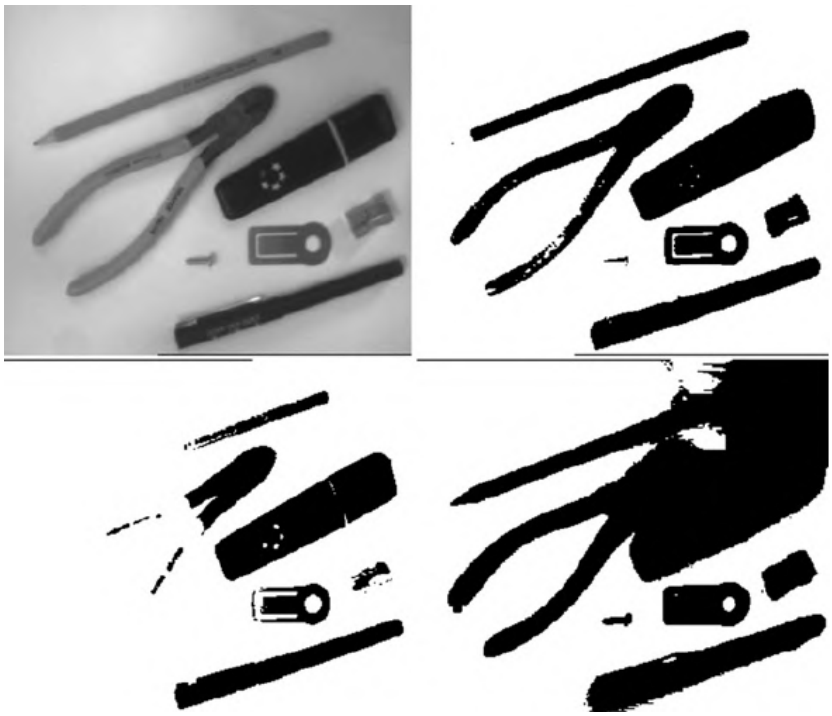

Figure 15: A grey-scale image (top left) together with a binary version where the threshold is 'correct' (top right), a version where the threshold is too low (bottom left) and a version where it is too high (bottom right)

We will look at a set of morphological operations (erosion, dilation, opening and closing) aimed primarily at enhancing the binary images. We will also look briefly at the application of morphological operators in grey-scale and colour images. Morphological operations work by performing a logical test in every possible position in an image between a structuring element and the corresponding portion of the image. In other words, the structuring element is effectively translated to each possible position in the image, the logical operation is applied (comparing the structuring element to the image in some fashion) and the result is stored in a separate output image. Binary images are treated as 2D sets, where the object points are the points in the sets. These sets use the same origin as the original image. Structuring elements are typically small sets of object points defined (typically symmetrically) around their own origin.

- Dilation: dilation is a technique for expanding the number of object pixels, typically in all directions simultaneously:

$$X \oplus B = \{p \in \epsilon^2; p = x + b, x \in X \text{ and } b \in B\} \tag{11}$$

This operation results in small holes being filled and fills in narrow gaps

between larger groups of set points. It also increases the size of objects (i.e. the number of points in the set). Note that normal dilation in an imaging context uses an isotropic structuring element.

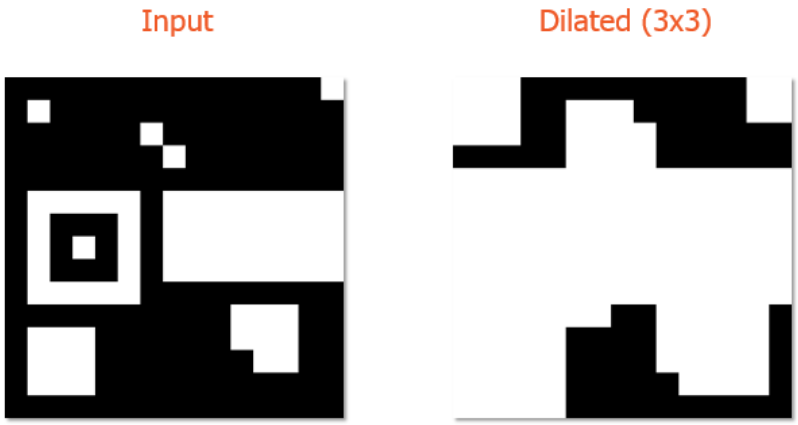

Figure 16: Binary image before (left) and after dilation (right) with a 3×3 isotropic structuring element

- Erosion: erosion is a technique for shrinking object shapes by removing pixels from the boundary:

$$X \ominus B = \{p \in \epsilon^2; p = x + b \in X \text{ for every } b \in B\} \tag{12}$$

A point p is an element of the eroded output set if, and only if, that point is a member of the input image set X when translated by each (and all) of the structuring element points/vectors b. It can also be thought of as a matching problem where the structuring element is compared with the input image set at every possible location, and the output is marked only where the structuring element and the image match perfectly. This operation results in any small points of noise and any narrow features being removed. It also reduces the size of objects (i.e. number of points in the set). Similar to dilation, normal erosion in an imaging context uses an isotropic structuring element.

- Opening and closing: erosion is the mirror concept of dilation and vice versa. While one expands the object pixels, the other shrinks them, and if we combine these operations together we get some interesting and useful effects.

  An opening is an erosion operation followed by a dilation operation with the same structuring element (B):

$$X \bigcirc B = (X \ominus B) \oplus B \tag{13}$$

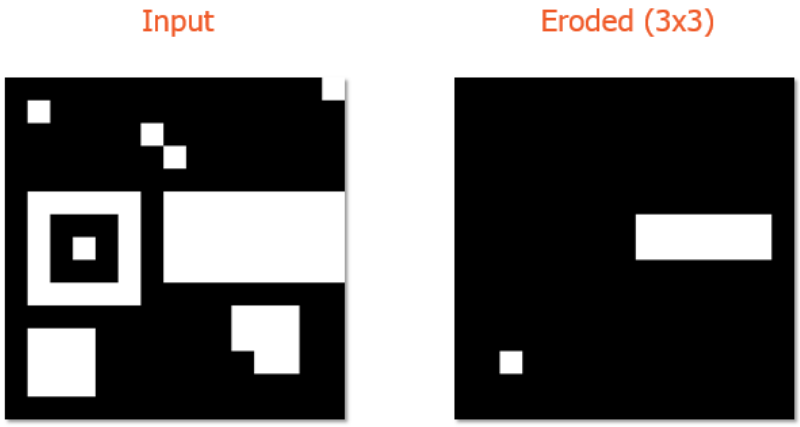

Figure 17: Binary image before (left) and after erosion (right) with a 3×3 isotropic structuring element

An opening removes noise (i.e. eliminates image details smaller than the structuring element), narrow features (such as bridges between larger groups of points), and smoothes the object boundaries. Unlike erosion and dilation though, it maintains the approximate size of the objects.

A closing is a dilation operation followed by an erosion operation, again with the same structuring element (B):

$$X \cdot B = (X \oplus B) \ominus B \tag{14}$$

A closing joins objects which are close together and fills in holes within objects. It tends to distort the shape of objects somewhat. Like opening, closing roughly maintains the size of objects.

3.1.3 Edges

The segmentation of an image is the process of dividing the image into pieces such that different objects or parts of objects are separated. This is an essential processing step in image understanding (where the goal is to understand the content of the images). There are two main ways of addressing the segmentation of images:

- Edge processing, where we identify the discontinuities (the edges) in images;

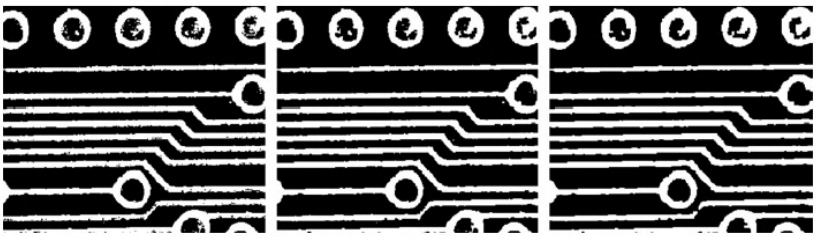

Figure 18: A binary image (left) which has been closed to join close objects (middle) and then opened to break thin connections (right). Note that one of the tracks (bottom middle) of the printed circuit board has broken during the opening operation, which may indicate that there is a problem with the board

- Region processing, where we look for homogeneous regions (or sections) of the images. Binary vision is a very simple example of region processing and much more complex approaches to this problem exist.

These representations can or should be complementary, with the edges delineating the homogeneous regions. Unfortunately it is very difficult to decide where the edges are (and hence to get a unique edge image) and it is equally hard to make a unique interpretation of an image in terms of regions. There are many techniques in both edge-based vision and region-based vision, all of which have the goal of determining the best representation of the scene in terms of edges or regions. For most images there is no absolute correct answer as the answer is typically subjective (it will depend on the observer and will depend upon the purpose of the segmentation).

Edge Detection

Edge detection was developed and is normally presented for single channel (grey-scale) images. Edges are the locations where brightness changes abruptly and hence are often considered from a point of view of 2D derivatives. Because we perform our processing in a discrete domain we typically represent the locations where brightness changes abruptly as edge pixels (although the brightness change will often be between pixels rather than on them).

An edge pixel has a gradient (rate of change) and a direction/orientation (which is taken to be the direction of the largest growth). Typically, edge detection is performed using either first derivative or second derivative operators (or some combination of the two). A first derivative operator results in a local maximum at an edge (where the rate of change is highest). The second derivative results in a 'zero-crossing' at an edge (where the function changes from +ve to –ve or vice versa).

At this point, it is worth pointing out that derivatives work on continuous functions, whereas the images we are processing are in the discrete domain. To approximate the derivates (both first and second derivatives), differences between image pixels are used.

Without diving too much into maths, we'll take into consideration three kind of edge detection which have been or will be mentioned in this paper:

- Edge image thresholding: Simply binary thresholding applied to a gradient image. Unfortunately, edges typically result in gradient responses which are several pixels wide and as a result edge image thresholding normally results in multiple responses across the edges. To address this issue, we need to add an additional stage prior to edge image thresholding in which all gradient responses outside the central maximum (for any given edge point) are suppressed. This technique is referred to as non-maxima suppression, which uses the gradient and orientation information to determine which edge points are the central ones (i.e. the main response at each point along an edge contour). It does this by using the orientation information from each point to determine which points are to each side of the current edge point, and then suppresses the current point if its gradient is less than either of its two neighbours;

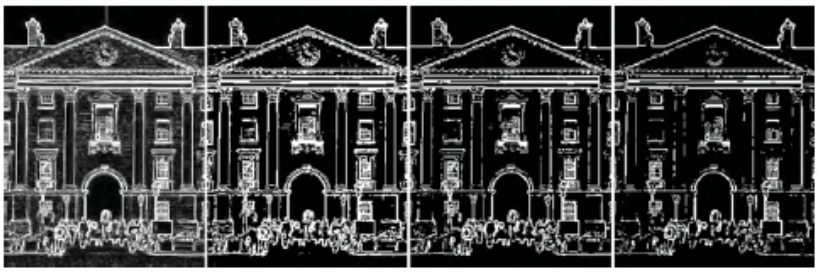

Figure 19: An edge gradient image (a) together with three thresholded versions (b), (c) and (d) thresholded at 120, 160 and 200 respectively

- Laplacian of Gaussian: The Laplacian of Gaussian is a second derivative edge detector and is still one of the most widely used. This edge detector is susceptible to noise, so to detect edges robustly we need to reduce the level of noise by smoothing before applying the edge detector. The smoothing filter has to satisfy two criteria: (1) that it is smooth and band limited in the frequency domain (i.e. it must limit the frequencies of edges in the image) and (2) that it exhibits good spatial localisation (i.e. the smoothing filter must not move the edges or change their spatial relationships). The optimal solution to this problem is to use Gaussian smoothing where σ^2 indicates the width of the Gaussian function:

$$G(i,j) = e^{-\frac{i^2+j^2}{2e^2}} \tag{15}$$

To compute the second derivative, then, the Gaussian smoothing operator should be applied to the image first and then the Laplacian operator should be applied. However it is possible to combine these two operators and apply them simultaneously to the image as a single 'Laplacian of Gaussian' operator. The convolution filter for this operator is defined as follows:

$$h(i,j) = \frac{1}{\pi\sigma^4}\left(\frac{i^2+j^2}{2\sigma^2} - 1\right)e^{-\frac{i^2+j^2}{2e^2}} \tag{16}$$

The shape of this convolution mask is positive in the centre and negative surrounding that returning to zero further from the centre. As a result it is often referred to as the Mexican hat filter.

One advantage of this filter is that it can take a much larger area into account (when compared to normal edge operators) when computing edges. This is not always a good thing though – say if a number of edges are present within that area. In addition, it sometimes does too much smoothing – losing fine details such as corners. Another frequently postulated advantage is that the Laplacian of Gaussian operator gives guaranteed closed loops of edges – which was presented as an important benefit but causes problems for some applications (or more to the point certain types of post-processing). It has been shown using neuro-physiological experiments that the human vision system (in the form of ganglion cells) performs operations which are very similar to Laplacian of Gaussian [17];

- Canny Edge Detection: The Canny edge detector combines first derivative and second derivative edge detection to compute both edge gradient and orientation. It was designed to optimise the following three criteria:
  - Detection – edges should not be missed;
  - Localisation – distance between actual and located edges should be minimised;
  - One response – minimises multiple responses to a single edge.

  Canny detection algorithm:
  - Convolve with Gaussian with some standard deviation σ;
  - Estimate the edge normal direction (i.e. orientation) using the first derivative of the Gaussian convolved image;
  - Find the edges, suppressing non-maxima values. This is done by searching for zero-crossings in the directional second derivative of the Gaussian convolved image, where those derivatives are taken with respect to the edge orientations which were computed in the previous step;

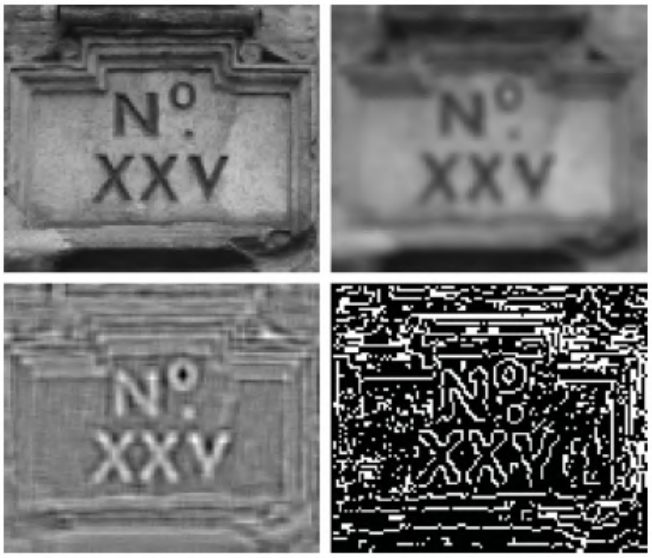

Figure 20: Grey-scale image of a sign (top left), smoothed by a Gaussian smoothing filter (top right), convolved with the Laplacian operator (bottom left), and a binary version showing zero crossing located in a simple (i.e. inaccurate) manner (bottom right)

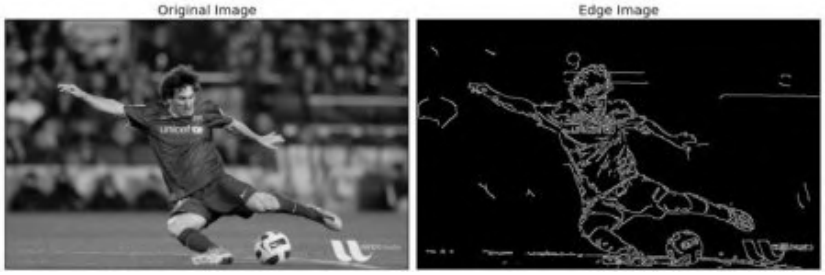

Figure 21: Original gray-scale image vs Canny edge detection application

- The edge gradient is computed based on the first derivative of the Gaussian convolved image;
- Thresholding edges with hysteresis. The idea here is that edge points are considered in contours of connected points. Two thresholds are used – a high gradient threshold over which all points are definitely classified as edge points and a lowthreshold over which points which are connected to definitive edge points are considered edge points.

This attempts to get over the problem of edge contours which break up into unconnected segments, due to the use of a simple threshold;

– Similar to the Laplacian of Gaussian detector we can do this processing for multiple scales (using different Gaussians) and then use 'feature synthesis' to combine edges from different scales [31].

3.2 Machine Learning

In recent times, machine learning (ML) has become very widespread in research and has been incorporated in different kind of applications, including text mining, spam detection, video recommendation, image classification, and multimedia concept retrieval. [52] Deep Learning (DL) is one of the most commonly used ML algorithms, especially in the mentioned situations. [23] The continuing appearance of novel studies in the fields of deep and distributed learning is due to both the unpredictable growth in the ability to obtain data and the amazing progress made in the hardware technologies, e.g. High Performance Computing (HPC). [24] DL employs transformations and graph technologies simultaneously in order to build up multi-layer learning models. The most recently developed DL techniques have obtained good outstanding performance across a variety of applications, including audio and speech processing, visual data processing, natural language processing (NLP), among others. [1]

Usually, the effectiveness of an ML algorithm is highly dependent on the integrity of the input-data representation. It has been shown that a suitable data representation provides an improved performance when compared to a poor data representation. Thus, a significant research trend in ML for many years has been feature engineering, which has informed numerous research studies. This approach aims at constructing features from raw data. In addition, it is extremely field-specific and frequently requires sizable human effort. For instance, several types of features were introduced and compared in the computer vision context, such as, histogram of oriented gradients (HOG) [8], scale invariant feature transform (SIFT) [20], and bag of words (BoW) [54]. As soon as a novel feature is introduced and is found to perform well, it becomes a new research direction that is pursued over multiple decades [37].

Machine intelligence is useful in many situations which is equal or better than human experts in some cases [56], meaning that DL can be a solution to the following problems:

- Cases where human experts are not available;
- Cases where humans are unable to explain decisions made using their expertise (language understanding, medical decisions, and speech recognition;
- Cases where the problem solution updates over time (price prediction, stock preference, weather prediction, and tracking);
- Cases where solutions require adaptation based on specific cases (personalization, biometrics);
- Cases where size of the problem is extremely large and exceeds our inadequate reasoning abilities (sentiment analysis, matching ads to Facebook, calculation webpage ranks).

Several are also the advantages that DL brings to the table:

- Universal Learning Approach: Because DL has the ability to perform in approximately all application domains, it is sometimes referred to as universal learning;
- Robustness: In general, precisely designed features are not required in DL techniques. Instead, the optimized features are learned in an automated fashion related to the task under consideration. Thus, robustness to the usual changes of the input data is attained;
- Generalization: Different data types or different applications can use the same DL technique, an approach frequently referred to as transfer learning (TL) which explained in the latter section. Furthermore, it is a useful approach in problems where data is insufficient;
- Scalability: DL is highly scalable. ResNet [32], which was invented by Microsoft, comprises 1202 layers and is frequently applied at a supercomputing scale. Lawrence Livermore National Laboratory (LLNL), a large enterprise working on evolving frameworks for networks, adopted a similar approach, where thousands of nodes can be implemented [10].

3.2.1 Exploration of a CNN

In the field of ML, DL, due to its considerable success, is currently one of the most prominent research trends.

Convolutional neural network (CNN) is one of the most popular and used of DL networks. The main advantage of CNN compared to its predecessors is that it automatically detects the significant features without any human supervision which made it the most used, [20] being extensively applied in a range of different fields, including computer vision [40], speech processing [59], Face Recognition [28], etc. Its structure was inspired by neurons in human and animal brains, similar to a conventional neural network. More specifically, in a cat's brain, a complex sequence of cells forms the visual cortex; this sequence is simulated by the CNN [65].

The benefits of using CNNs over other traditional neural networks in the computer vision environment are basically the following:

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

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

3.2.2 Architecture

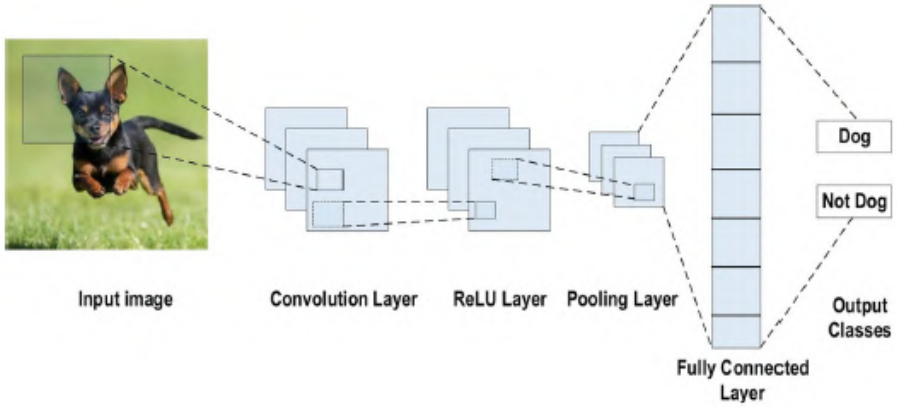

Figure 22: Example of a CNN architecture for image classification

A commonly used type of CNN, which is similar to the multi-layer perceptron (MLP), consists of numerous convolution layers preceding sub-sampling (pooling) layers, while the ending layers are FC layers. An example of CNN architecture for image classification is illustrated in Fig. 22. The CNN architecture consists of a number of layers (or so-called multi-building blocks). Those are the layers and their function:

- Convolutional Layer: In CNN architecture, this is the most significant component. It consists of a collection of convolutional filters (so-called kernels). The input image, expressed as N-dimensional metrics, is convolved with these filters to generate the output feature map;
- Activation Function (non-linearity): Mapping the input to the output is the core function of all types of activation function in all types of neural

network. The input value is determined by computing the weighted summation of the neuron input along with its bias (if present). This means that the activation function makes the decision as to whether or not to fire a neuron with reference to a particular input by creating the corresponding output. Non-linear activation layers are employed after all layers with weights (so-called learnable layers, such as FC layers and convolutional layers) in CNN architecture. This non-linear performance of the activation layers means that the mapping of input to output will be non-linear; moreover, these layers give the CNN the ability to learn complicated things. The activation function must also have the ability to differentiate, which is an extremely significant feature, as it allows error back-propagation to be used to train the network. Some of the most commonly used in CNN and other deep neural networks are Sigmoid, Tanh and ReLU;

- Pooling Layer: The main task of the pooling layer is the sub-sampling of the feature maps. These maps are generated by following the convolutional operations. In other words, this approach shrinks large-size feature maps to create smaller feature maps. Concurrently, it maintains the majority of the dominant information (or features) in every step of the pooling stage. In a similar manner to the convolutional operation, both the stride and the kernel are initially size-assigned before the pooling operation is executed. Several types of pooling methods are available for utilization in various pooling layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling. The most familiar and frequently utilized pooling methods are the max, min, and GAP pooling, which are illustrated in Fig. 23. Sometimes, the overall CNN performance is decreased as a result; this represents the main shortfall of the pooling layer, as this layer helps the CNN to determine whether or not a certain feature is available in the particular input image, but focuses exclusively on ascertaining the correct location of that feature. Thus, the CNN model misses the relevant information;

- Fully Connected Layer: Commonly, this layer is located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach. It is utilized as the CNN classifier. It follows the basic method of the conventional multiple-layer perceptron neural network. The input of the FC layer comes from the last pooling or convolutional layer. This input is in the form of a vector, which is created from the feature maps after flattening. The output of the FC layer represents the final CNN output, as illustrated in Fig. 24;

- Loss Functions: The final classification is achieved from the output layer, which represents the last layer of the CNN architecture. Some loss functions are utilized in the output layer to calculate the predicted error cre-

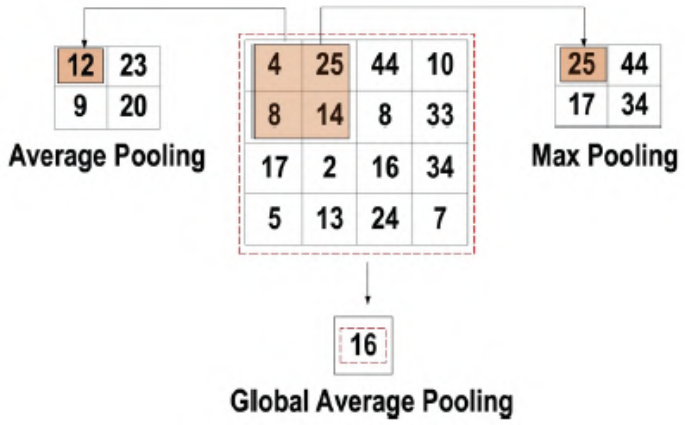

Figure 23: Three different types of pooling operations

ated across the training samples in the CNN model. This error reveals the difference between the actual output and the predicted one. Next, it will be optimized through the CNN learning process. However, two parameters are used by the loss function to calculate the error. The CNN estimated output (referred to as the prediction) is the first parameter. The actual output (referred to as the label) is the second parameter. Several types of loss function are employed in various problem types, such as the Cross-Entropy (or Softmax), the Euclidean and the Hinge ones.

The input x of each layer in a CNN model is organized in three dimensions: height, width, and depth, or m $\times$ m $\times$ r , where the height (m) is equal to the width. The depth is also referred to as the channel number. For example, in an RGB image, the depth (r) is equal to three. Several kernels (filters) available in each convolutional layer are denoted by k and also have three dimensions (n $\times$ n $\times$ q), similar to the input image; here, however, n must be smaller than m, while q is either equal to or smaller than r. In addition, the kernels are the basis of the local connections, which share similar parameters (bias b^k and weight W^k) for generating k feature maps h^k with a size of (m - n - 1) each and are convolved with input, as mentioned above. The convolution layer calculates a dot product between its input and the weights as in Eq. 17, similar to NLP, but the inputs are undersized areas of the initial image size. Next, by applying the nonlinearity or an activation function to the convolution-layer output, we obtain the following:

$$h^k = f(W^k x + b^k) \tag{17}$$

The next step is down-sampling every feature map in the sub-sampling layers. This leads to a reduction in the network parameters, which accelerates the training process and in turn enables handling of the overfitting issue. For

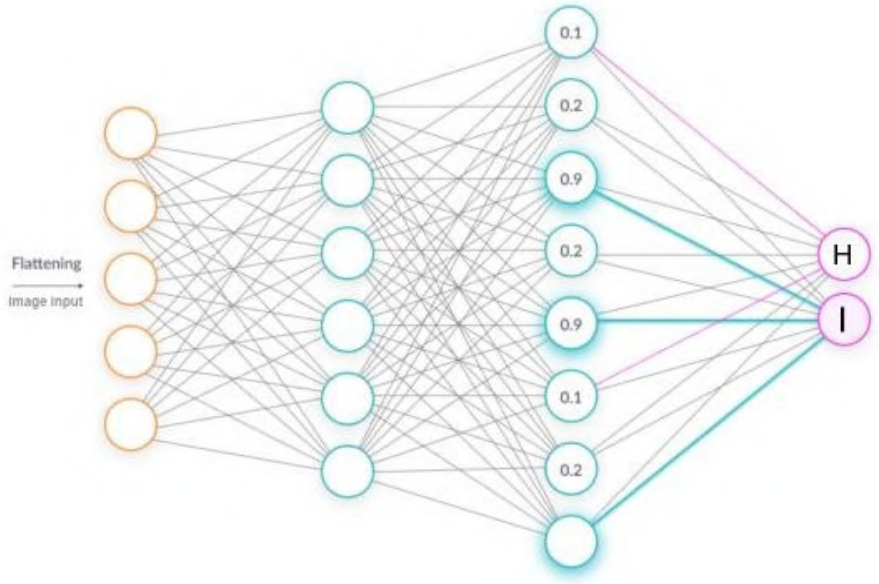

Figure 24: Fully connected layer

all feature maps, the pooling function (e.g. max or average) is applied to an adjacent area of size p × p , where p is the kernel size. Finally, the FC layers receive the mid- and low-level features and create the high-level abstraction, which represents the last-stage layers as in a typical neural network. The classification scores are generated using the ending layer [e.g. support vector machines (SVMs) or softmax]. For a given instance, every score represents the probability of a specific class [37].

3.2.3 Regularization

Model fitting is the measure of how well a machine learning model generalizes data similar to that with which it was trained. Fitting refers to adjusting the parameters in the model to improve accuracy. The process involves running an algorithm on data for which the target variable ("labeled" data) is known to produce a ML model. Then, the model's outcomes are compared to the real, observed values of the target variable to determine the accuracy.

In a CNN model, overfitting represents the main issue when trying to get a well-behaving model. The model is entitled overfitted in cases where the model executes especially well on training data and does not succeed on test data (unseen data). An underfitted model is the opposite; this case occurs when the

model does not learn a sufficient amount from the training data. The model is referred to as "just-fitted" if it executes well on both training and testing data. These three types are illustrated in Fig. 25.

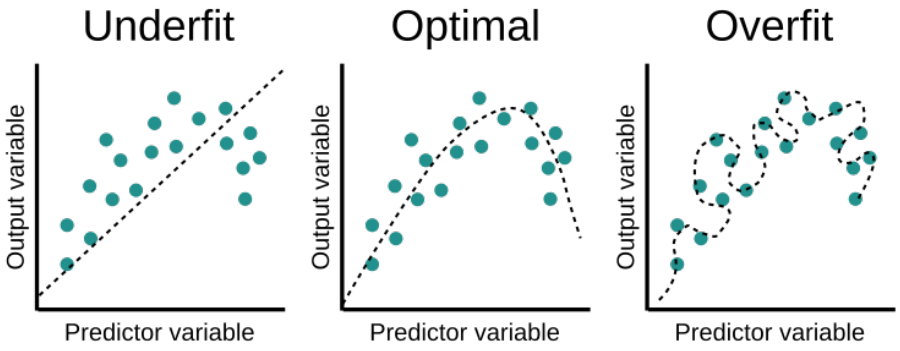

Figure 25: Fitting a model

Various intuitive concepts are used to help the regularization to avoid over-fitting:

- Dropout: This is a widely utilized technique for generalization. During each training epoch, neurons are randomly dropped. In doing this, the feature selection power is distributed equally across the whole group of neurons, as well as forcing the model to learn different independent features. During the training process, the dropped neuron will not be a part of back-propagation or forward-propagation. By contrast, the full-scale network is utilized to perform prediction during the testing process;
- Drop-Weights: This method is highly similar to dropout. In each training epoch, the connections between neurons (weights) are dropped rather than dropping the neurons, being the only difference between drop-weights and dropout;
- Data Augmentation: Training the model on a sizeable amount of data is the easiest way to avoid over-fitting. To achieve this, data augmentation is used, thanks to several techniques which artificially expand the size of the training dataset, which will be discussed later;
- Batch Normalization: This method is used to make training of artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling. Each layer of a neural network has inputs with a corresponding distribution, which is affected during the training process by the randomness in the parameter initialization and the randomness in the input data. The effect of these sources of randomness on the distribution of the inputs to internal layers during training

is described as internal covariate shift. Although a clear-cut precise definition seems to be missing, the phenomenon observed in experiments is the change on means and variances of the inputs to internal layers during training. During the training stage of networks, as the parameters of the preceding layers change, the distribution of inputs to the current layer changes accordingly, such that the current layer needs to constantly readjust to new distributions. This problem is especially severe for deep networks, because small changes in shallower hidden layers will be amplified as they propagate within the network, resulting in significant shift in deeper hidden layers. Therefore, the method of batch normalization is proposed to reduce these unwanted shifts to speed up training and to produce more reliable models [37].

3.2.4 Optimization

This part regards the CNN learning process, which includes two major issues: choosing the learning algorithm (optimizer) and the use of enhancements along with the algorithm to enhance the final output.

Loss functions, which are founded on numerous learnable parameters (e.g. biases, weights, etc.) or minimizing the error (variation between actual and predicted output), are the core purpose of all supervised learning algorithms. The techniques of gradient-based learning for a CNN network appear as the usual selection. The network parameters should always update though all training epochs, while the network should also look for the locally optimized answer in all training epochs in order to minimize the error.

The learning rate is defined as the step size of the parameter updating. The training epoch represents a complete repetition of the parameter update that involves the complete training dataset at one time. Note that it needs to select the learning rate wisely so that it does not influence the learning process imperfectly, although it is a hyper-parameter.

One of the most used learning algorithm is Gradient Descent or Gradient-based: to minimize the training error, this algorithm repetitively updates the network parameters through every training epoch. More specifically, to update the parameters correctly, it needs to compute the objective function gradient (slope) by applying a first-order derivative with respect to the network parameters. Next, the parameter is updated in the reverse direction of the gradient to reduce the error. The parameter updating process is performed though network back-propagation, in which the gradient at every neuron is back-propagated to all neurons in the preceding layer. This operation is represented in Eq. 18:

$$w_{ij^t} = w_{ij^t-1} - \Delta w_{ij^t}, \Delta w_{ij^t} = \eta \times \frac{\partial E}{\partial w_{ij}} \tag{18}$$

The final weight in the current training epoch is denoted by w_{ij^t}, while the

weight in the preceding (t-1) training epoch is denoted w_{ij^t-1}. The learning rate is η and the prediction error is E.

The following example further explains the situation:

> *Gradient decent can be explained as below:*
> *A helicopter dropped a blind person randomly at some location and his aim to find a lowest valley we could reach. From his current position he checks the slope of the land with a stick and follows the slope to achieve his goal. This location may contain many valleys but may not be the lowest valley. Sometimes the blind person can get stuck in a valley surrounded by mountains which is not the lowest valley as he only check the slope at his current location, thus he assumes that this is the lowest valley he could find.*
> *Reason why multiple epochs:*
> *The best way to solve this problem is to drop this person again randomly at some other location and try to find the lowest valley more than once. Record all the lowest valley and find the lowest among it.*

Some alternative methods include the Batch, the Stochastic and the Mini-batch Gradient Descent, the Momentum and the Adaptive Momentum Estimation (Adam) [61, 38, 35], with the last one being one of the latest trends in deep learning optimization since it has been specifically designed for training deep neural networks, being also more memory efficient and asking for less computational power. [71]

3.2.5 Data manipulation

In most supervised Machine Learning tasks, best practice recommends to split the data into two independent sets: a training set and a validation set.

- Training Set: The dataset that we feed our model to learn potential underlying patterns and relationships;
- Validation Set: The dataset that we use to understand our model's performance across different model types and hyperparameter choices.

The training set is this dataset that our model uses to learn any underlying patterns or relationships that will enable making predictions later on. The training set should be as representative as possible of the population that we are trying to model. Additionally, we need to be careful and ensure that it is as unbiased as possible, as any bias at this stage may be propagated downstream during inference.

The validation set acts as an independent, unbiased dataset for comparing the performance of different algorithms trained on our training set, used to

validate our model performance during training.. This validation process gives information that helps us tune the model's hyperparameters and configurations accordingly. It is like a critic telling us whether the training is moving in the right direction or not.

The model is trained on the training set, and, simultaneously, the model evaluation is performed on the validation set after every epoch. The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e., the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before [55].

DL is extremely data-hungry considering it also involves representation learning [33, 70]. DL demands an extensively large amount of data to achieve a well-behaved performance model, i.e. as the data increases, a good performance model can be achieved. In most cases, the available data are sufficient to obtain a good performance model. However, sometimes there is a shortage of data for using DL directly [62]. To properly address this issue, three suggested methods are available:

- The first involves the employment of the transfer-learning concept after data is collected from similar tasks. Note that while the transferred data will not directly augment the actual data, it will help in terms of both enhancing the original input representation of data and its mapping function [27]. In this way, the model performance is boosted. Another technique involves employing a well-trained model from a similar task and fine-tuning the ending of two layers or even one layer based on the limited original data;

- In the second method, data augmentation is performed [64]. This task is very helpful for use in augmenting the image data, since the image translation, mirroring, and rotation commonly do not change the image label. Conversely, it is important to take care when applying this technique in some cases such as with bioinformatics data. For instance, when mirroring an enzyme sequence, the output data may not represent the actual enzyme sequence;

- In the third method, simulated data can be considered for increasing the volume of the training set. It is occasionally possible to create simulators based on the physical process if the issue is well understood. Therefore, the result will involve the simulation of as much data as needed.

We will now take into detail the main two of those methods: Transfer learning and Data augmentation.

The use of CNNs has recently widespread, offering plenty of support for answering many classification problems. Generally speaking, deep CNN models require a sizable volume of data to obtain good performance. The common challenge associated with using such models concerns the lack of training data.

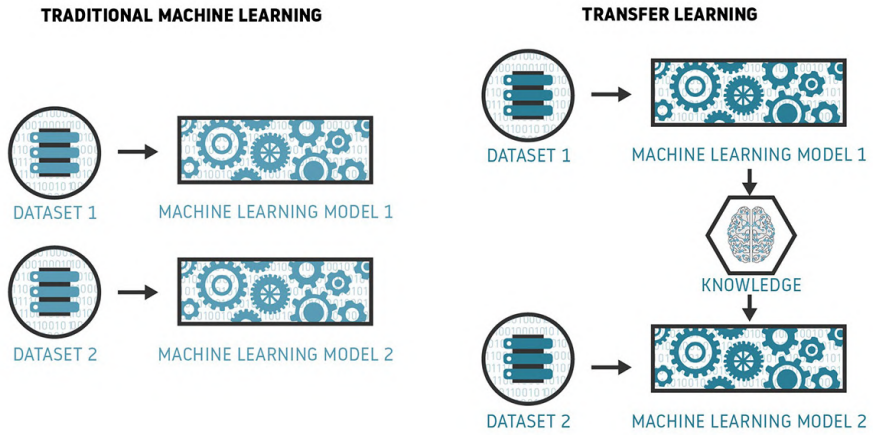

Figure 26: Classic ML vs Transfer learning

Indeed, gathering a large volume of data is an exhausting job, and no successful solution is available at this time. The undersized dataset problem is therefore currently solved using the Transfer Learning technique [13, 18], which is highly efficient in addressing the lack of training data issue. The mechanism of TL involves training the CNN model with large volumes of data.

The student-teacher relationship is a suitable approach to clarifying TL. Gathering detailed knowledge of the subject is the first step [68]. Next, the teacher provides a "course" by conveying the information within a "lecture series" over time. Put simply, the teacher transfers the information to the student. In more detail, the expert (teacher) transfers the knowledge (information) to the learner (student). Similarly, the DL network is trained using a vast volume of data, and also learns the bias and the weights during the training process. These weights are then transferred to different networks for retraining or testing a similar novel model. Thus, the novel model is enabled to pre-train weights rather than requiring training from scratch. The conceptual design is illustrated in Fig. 26.

In a real case scenario, the "teacher" we've mentioned is basically a pre-trained model. Many CNN models have been trained on large datasets such as ImageNet for image recognition purposes. These models can then be employed to recognize a different task without the need to train from scratch. Furthermore, the weights remain the same apart from a few learned features. In cases where data samples are lacking, these models are very useful. There are many reasons for employing a pre-trained model. First, training large models on sizeable datasets requires high-priced computational power. Second, training large models can be time-consuming, taking up to multiple weeks. Finally, a pre-trained model can assist with network generalization and speed up the

convergence.

Pre-trained models, however, have their downsides:

- Training a good model requires a massive number of images, for which multiple layers must be included. Achieving excellent outcomes in image classification or recognition applications, with performance occasionally superior to that of a human, becomes possible through the use of deep convolutional neural networks (DCNNs) [32, 13, 39]. However, the accuracy of the model becomes insufficient in the case of the utilized model has fewer layers, or if a small dataset is used for training due to over- or under-fitting problems;

- The majority of the crowdsourcing workers are unable to make accurate notes, for example, on medical or biological images due to their lack of medical or biological knowledge. Thus, ML researchers often rely on field experts to label such images; however, this process is costly and time consuming. Therefore, producing the large volume of labels required to develop flourishing deep networks turns out to be unfeasible. Natural images from public datasets are completely dissimilar from the raw medical images, meaning that the model performance is not enhanced.

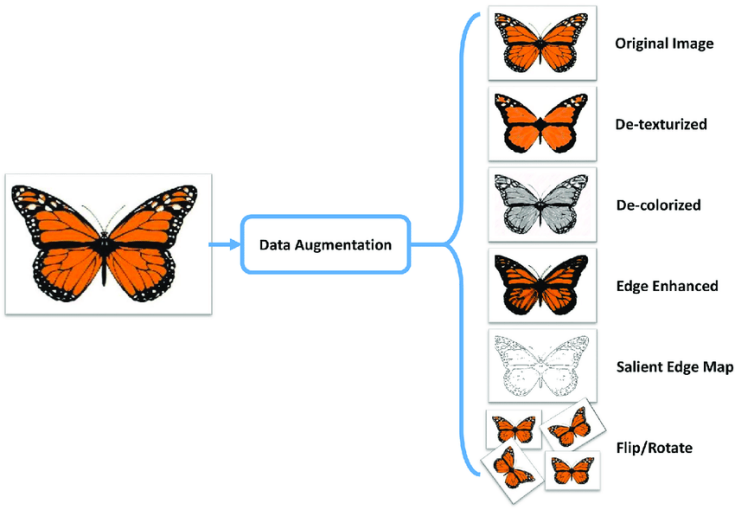

Figure 27: Some examples of data augmentation on an image

Data augmentation techniques provide solutions to overfitting issues, increasing the amount of available data [64, 63, 29]. These techniques are data-space solutions for any limited-data problem. Data augmentation incorporates a collection of methods that improve the attributes and size of training datasets, allowing DL networks to perform better when these techniques are employed.

- Flipping: Flipping the vertical axis is a less common practice than flipping the horizontal one. It is also very simple to implement and it is not a label-conserving transformation on datasets that involve text recognition (such as SVHN and MNIST).

- Color space: Encoding digital image data is commonly used as a dimension tensor (height × width × colorchannels). Accomplishing augmentations in the color space of the channels is an alternative technique, which is extremely workable for implementation. A very easy color augmentation involves separating a channel of a particular color, such as Red, Green, or Blue. A simple way to rapidly convert an image using a single-color channel is achieved by separating that matrix and inserting additional double zeros from the remaining two color channels. Furthermore, increasing or decreasing the image brightness is achieved by using straightforward matrix operations to easily manipulate the RGB values. By deriving a color histogram that describes the image, additional improved color augmentations can be obtained. Lighting alterations are also made possible by adjusting the intensity values in histograms similar to those employed in photo-editing applications;

- Cropping: Cropping a dominant patch of every single image is a technique employed with combined dimensions of height and width as a specific processing step for image data. Furthermore, random cropping may be employed to produce an impact similar to translations. The difference between translations and random cropping is that translations conserve the spatial dimensions of this image, while random cropping reduces the input size [for example from (256, 256) to (224, 224)]. According to the selected reduction threshold for cropping, the label-preserving transformation may not be addressed;

- Rotation: Rotation augmentations are obtained when rotating an image left or right from within 0 to 360 degrees around the axis. In digit recognition tasks, small rotations (from 0 to 20 degrees) are very helpful. By contrast, the data label cannot be preserved post-transformation when the rotation degree increases; Translation: To avoid positional bias within the image data, a very useful transformation is to shift the image up, down, left, or right. For instance, it is common that the whole dataset images are centered; moreover, the tested dataset should be entirely made up of centered images to test the model. Note that when translating the initial images in a particular direction, the residual space should be filled with Gaussian or random noise, or a constant value such as 255s or 0s. The spatial dimensions of the image post-augmentation are preserved using this padding;

- Noise injection: This approach involves injecting a matrix of arbitrary values, which is commonly obtained from a Gaussian distribution. Injecting

Figure 28: Example of a noise-injected image (b) vs original (a)

noise within images enables the CNN to learn additional robust features (Fig. 28).

4 Cini Foundation Projects

In the previous chapters we provided an introduction to Machine Learning and Computer Vision and other related topics such as convolutional neural networks, along with their potential and practical use in the world of humanities. In this section we want to present the project carried out with the active collaboration with the ARCHiVe center of the Giorgio Cini Foundation, having been the direct client and participant in the thinking processes and activities.

First, we are gonna introduce our data, followed by the explanation of the development of the Image Processing application. Then, given the results, we will also explain the procedures that lead to the images analysis done through the Object Detection model.

4.1 Data

The data we are going to work with will differ according to the type of application. In fact, the following project is divided into two parts: one concerning the automatic post-production of images and the other concerning the identification of particular elements within a series of images. However, if in the first case the range of types is very wide, in the second case we will work exclusively with pages of books. As far as the first part is concerned, entire collections are taken into consideration, collected by origin, which often concern old drawings/drafts of artifacts to be made in glass. Here the numbers are potentially huge, since single collections can count up to 20000-30000 images, still having to consider that every raw image (in .CR2 format, which is a heavy one) weighs around 50-60 MB.

Project	Number of images	Type	Format
Image processing	20000-30000	Artworks	CR2
Object detection	1200	Book pages	JPG

Figure 29: The data used in the project

The identification of classes within the pages of the books, on the other side, concerns a different series of books and manuscripts, with a diversification capable of optimizing the effectiveness of the final model as much as possible. At the end, for the training of the model discussed in this thesis, we have got a total of around 1200 images. We will also explain in detail in the dedicated part,

soon, how those images got transformed and ready to be used for the training phase.

Figure 30: Sample image processing 1

Figure 31: Sample image processing 2

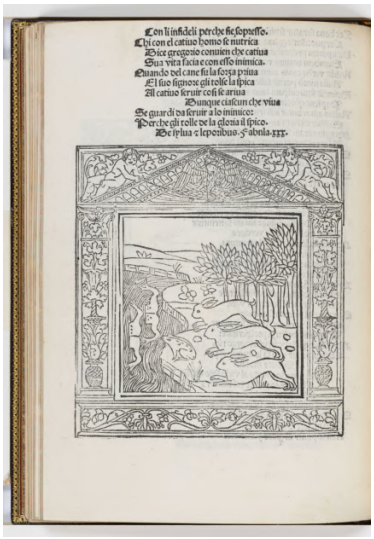

Figure 32: Sample object detection 1

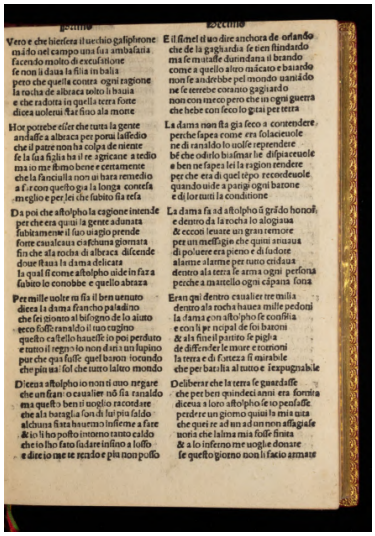

Figure 33: Sample object detection 2

4.2 Image Postprocessing

Planning and development

The goal is to create an application (possibly as user-friendly as possible, in such a way as to ensure simple use by everyone) capable of performing certain operations in a completely automatic way once giving as an input the images of a fund and having set a series of parameters.

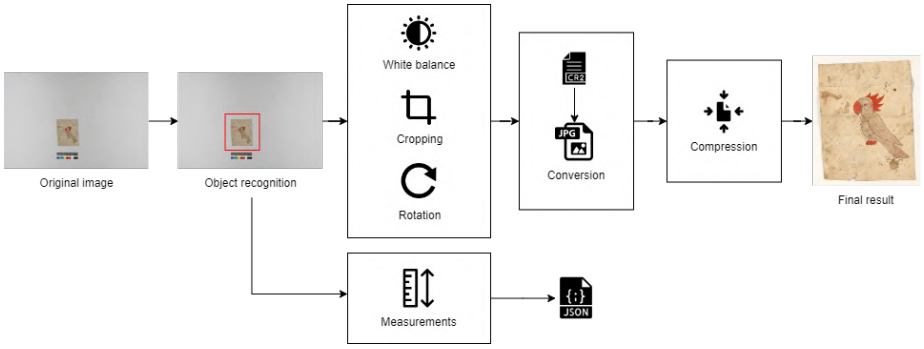

Figure 34: The designed application procedure

The application that had to be developed had to meet mainly the following requirements, which we will better explain later:

- Recognize the main object and distinguish it from the rest of the image;
- Make a rectangular cutout with a slight margin starting from the edges of the object;
- Apply a color adjustment through white balance, standard operation to restore the authenticity of the image;
- Rotate the image according to its real orientation, in case it is not correctly positioned;
- Convert the new cropped file to a lighter format such as JPG, since the original images are provided in the camera's RAW format (.CR2), which is large in size;
- Perform further compression in order to reduce the new JPG file to a reduced size, especially for use in previews and thumbnails;
- Measurement of the dimensions (width, length) of the object, exported as metadata in JSON format.

The project have been developed in Python, being the most flexible and accessible programming language for such operations. Various libraries have been used to carry out the many functions of the application, such as OpenCV, Numpy and Pillow for image manipulation, Matplotlib for visualization and Gooey for the graphic UI of the application. The application of the color profile to images is based on the use of the open-source application RawTherapee, therefore necessary for the use of this function, called up for our needs via the CLI (Command Line Interface). The whitebalancing function, also, is called into the application from an external batch script.

Technical description

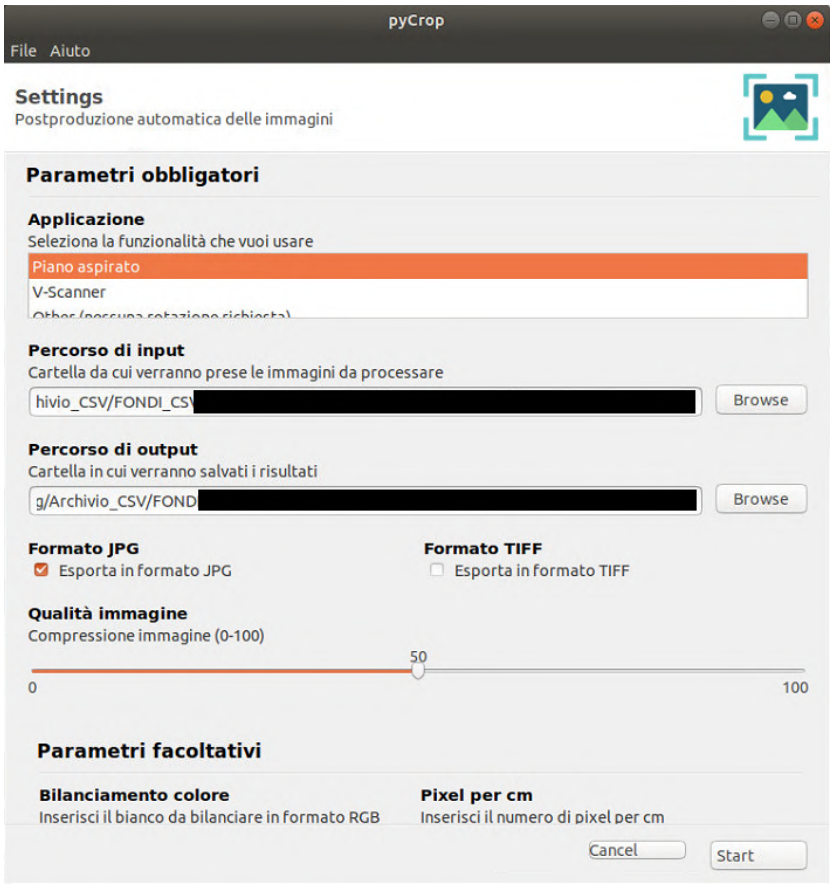

Figure 35: The applications' main menu 1/2

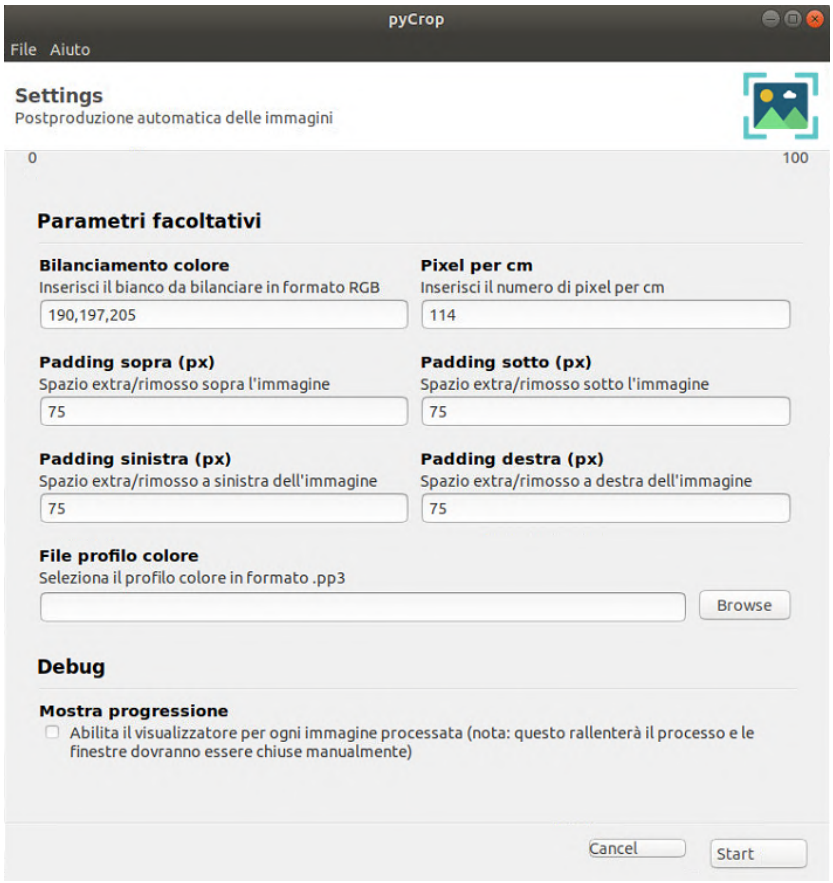

Figure 36: The applications' main menu 2/2

The application comes with a screen showing all the features, starting with the minimum mandatory parameters. First of all it is possible to set the tool originally used for image acquisition (which determines its rotation); this is followed by the insertion of an input path (from which the images to be processed will be taken) and an output path in which to enter the results of the process; the format decision (.JPG, .TIF) and the compression of the output images (due to the need to have, sometimes, lower quality images to be used as previews/icons) closes the section of mandatory parameters. The optional parameters (for which it is therefore not strictly necessary to fill in the empty field), on the other hand, concern style additions or changes, such as white balance (via PHP script after acquiring the RGB value on which to balance), the insertion pixel/cm valueto measure the object (value acquired through another tool created specifically for the application) and the application of a color profile (.PP3, an operation that requires the call of RawTherapee via CLI). In the lower part of the application

there are customized clipping parameters (in each direction it is possible to add a custom padding) and the possibility of enabling a viewer (via Matplotlib) in order to show all the recognition phases of the object and cropping made by the software, image by image.

Here is how the processing works according to the mentioned features:

- A selection of the tool used for the original image acquisition is needed since, according to the selected tool, we will determine the rotation the image has to do. In fact, in the case of a scan carried out with the so-called "v-scanner" (a scanner with a V shape, used for books and manuscripts since allows to acquire pages more easily), the program knows in advance that, based on the name of the file (renamed on the right or left page during shooting), it will have to be rotated accordingly. Other tools present include the vacuum plane and a simple option that allows you not to rotate the final image;

- We then clearly need two basic information: the folder from which to take the input images and the one in which to create the new output images. Those are just taken as parameters in the main application screen;

- At this point the images are identified and analyzed. Once imported into OpenCV, the library that we will use for the main post-production operations, it is necessary to convert them to a grayscale and create a kernel. Kernels in computer vision are matrices, used to perform some kind of convolution in our data. Convolutions are mathematical operations between two functions that create a third function. In image processing, it happens by going through each pixel to perform a calculation with the pixel and its neighbors. The kernels will define the size of the convolution, the weights applied to it, and an anchor point usually positioned at the center. So in a 3x3 matrix, each pixel is affected only by the pixels around it, wherein a 7x7 farther pixels would change it. To apply it to an image, we would position it over a given pixel using its anchor point, perform some operation between the values in the kernels and the values of the pixels, define a new value for that pixel, and move to the next. We can now apply different filters, and we just need to pass the image and the size of the kernel [14].

  As an example image to explain how the process works, we will use a drawing of a very cute parrot. First, we dilate the image. As explained on p. 44-45, dilation is a morphological operation in which, given a kernel overlapped with the image, it adds bright pixels to the boundaries of objects in an image, covering holes and expanding the number of pixels of the highlighted objects. Now, we apply thresholding to binarize the image based on the new pixel intensities. Specifically, we will use an Otsu tresholding, which will automatically calculate the threshold value to use (see p. 43).

Figure 37: Original image of the parrot drawing

This is a first way in which the application processes images. However, not every object is that easy to recognize. This is the reason why a second method has been developed, in case the first fails or is not sufficient to achieve the predetermined objectives. The new process starts the same way: grayscale conversion, kernel setup and tresholding, with the difference that the latter is no longer detected automatically but is kept on a fixed threshold, making it more effective on a certain category of images (those that in fact very often they were not adequately processed with the first method). To these steps is now added the application of a Gaussian blur (or smoothing), the result of blurring an image by a Gaussian function (of which we have seen something on p. 40-41). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Follows another method we have seen after that, the Canny edge detector, applied on the just blurred image. However, we need to clean up all the noise made by the blur and then by the Canny functions. To do that, but also to better highlight the contours of the image, we will perform a particular operation: we will first apply a dilation and then an erosion. By doing this we are able to close all the holes generated by the noise and, at the same time, to eliminate the extra pixels that are not part of the original image. The OpenCV object detection system, through the contours-finding function, is now able to identify our parrot drawing and we're now able to easily crop it off.

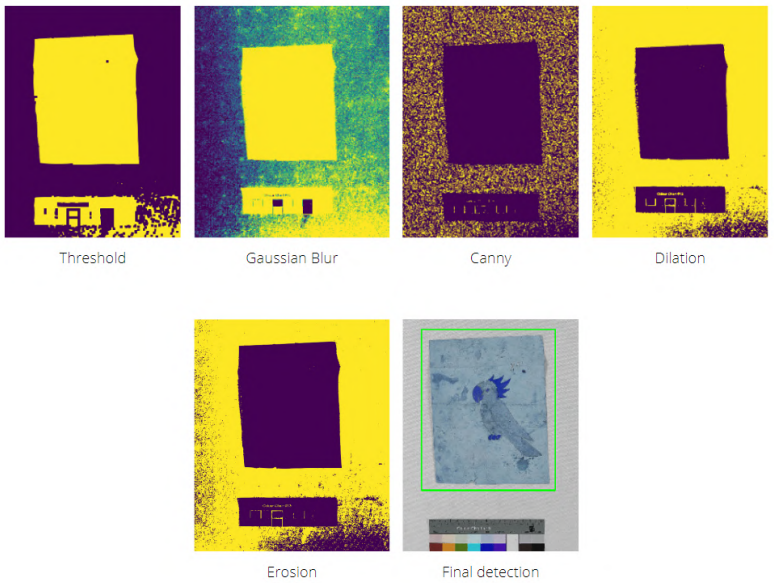

Figure 38: Post-processing phase

We have chosen an image that is very easy to see and manipulate, to the point that some steps may seem inappropriate. However, this is a very effective method on a large scale of images, albeit with limitations due to the potential vastness of combinations in image structures;

- The source images have a raw extension, .CR2, not particularly compatible with an easy use of the contents also due to the excessive weight of the format. Consequently, the image first needs to be converted. The most useful formats in such circumstances, especially in the context of an online publication, are JPG and TIF. A further step is represented by the compression of the images, often useful in the case of thumbnails or images of secondary importance, for which it is not necessary to maintain a high quality. This process is done through Pillow (PIL), the Python Imaging Library that adds image processing capabilities to our Python interpreter, providing extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It basically provides a solid foundation for a general image processing tool;

The steps just seen concerned the minimum parameters required to process an image, to which it is possible to add further customizations. Now we will see what we can do to enhance our image processing.

Figure 39: Post-processed parrot

- White balancing, one of the custom options of the application, is a process that leads to the adjustment of the intensity of colors on the basis of a "pure" white present in the photos themselves (through the use of color charts or color checkers) or acquired externally, on its own, but still in the same shooting conditions as the images taken in analysis. In fact, it happens that the environment in which these are acquired is never particularly perfect, not making the original colors of the objects at their best.

  There were two paths considered viable: resorting to an automatic recognition algorithm or a safer method, based on a previous acquisition of the RGB white value that we want to use as a balancing pivot. In the first case we use what is referred to as the Greyworld method, a simple mechanism in which the main premise is that, in a normal photo with sufficiently balanced colors, the average of all colors turns out to be a neutral gray. Therefore, it is possible to estimate the illuminating color by observing the average color and comparing it with the gray. In the second case,

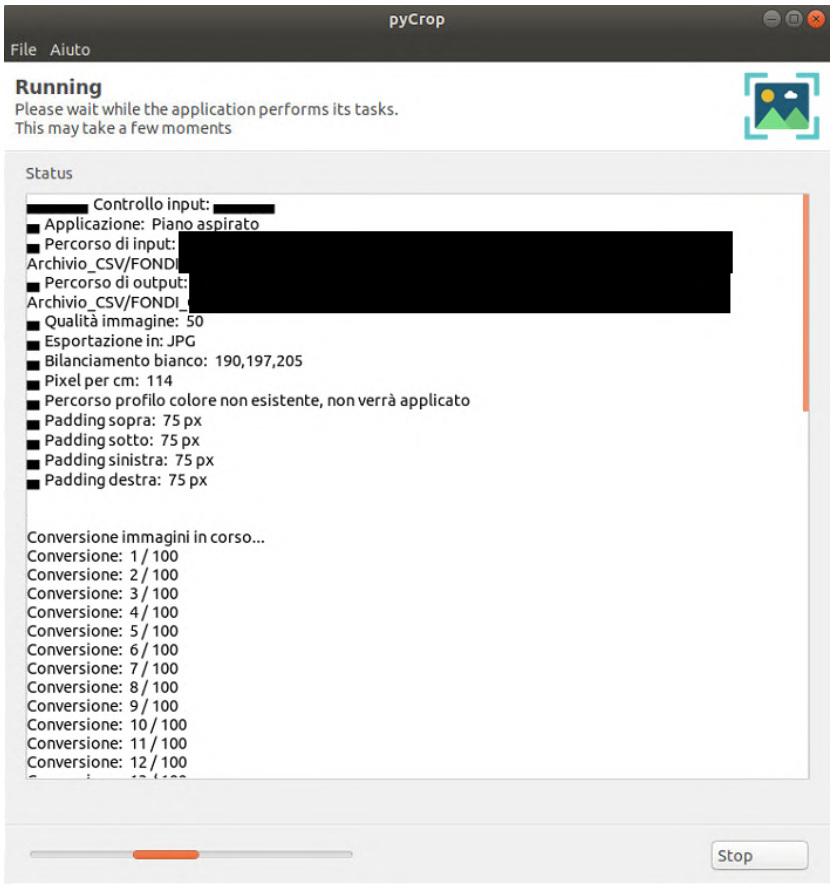

Figure 40: Processing example

however, it is possible to achieve maximum precision with minimal effort, considering that it is sufficient to acquire the correct color value only once for the entire set of images to be balanced.

In the provided examples (it is possible to notice the differences between the original photo and the application of both methods, being able to agree as a way is a slight difference between the automatic method and the manual one.

Algorithms for automatic adjustments attempt to automatically improve the perceptual quality of an image and are commonly used, for example, by digital cameras. Manual adjustments and adjustments are often required to achieve the best possible results, but for normal use the automatic enhancements make it easy to achieve higher quality. In our case we want to obtain the maximum possible yield, so the implementation of a manual

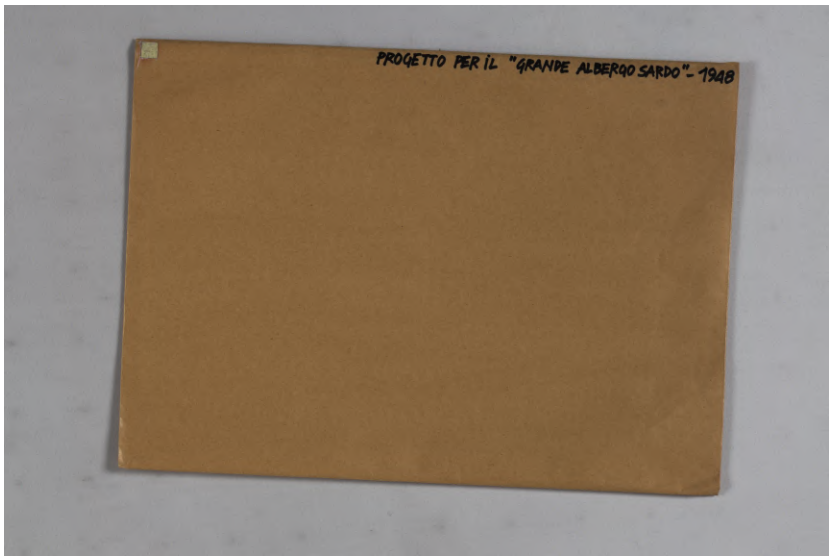

Figure 41: Original image (manually cropped for the example)

balancing system is fundamental;

- Another optional feature is the measurement of the identified object. The operation is very simple: once the measurements in pixels of the bounding box of the object have been taken, this is converted into centimeters after entering the pixel per centimeter value. This value can be obtained through a small external tool that allows, given an image, to establish with extreme precision the size in pixels of one centimeter positioned on the color. Once the value has been entered in the dedicated field and the procedure has been started, the new measurement will return to the user in the form of a JSON file, which is later reused for inserting metadata in the digital library;

- Sometimes color balancing is done via pre-prepared external color profiles. In our specific case, these are .pp3 files created through the opensource application RawTherapee, an image editing software. These files contain numerous stylistic changes that allow, once adopted within our application, to make the same changes in an image dataset. For this option to work, however, a prior installation of RawTherapee is required, since the application of the color profile is carried out by calling the software via the CLI (Command Line Interface);

- Finally, the application contemplates the possible addition of artificial margins. It is simply a numerical value in pixels (one for each side) that can be positive or negative, therefore corresponding to an addition or removal of a part of the image. In the image scanning phase, in fact, the

Figure 42: Automatic balancing (Greyworld)

Figure 43: Manual balancing (RGB pick)

minimum bounding box is identified as close as possible to the edges of the object. However, this is not the final result we want to achieve. For this reason it is possible to add a custom margin on each side, to ensure a good visual rendering to each image. Another clever use of this function is related, in a specific case faced by myself in the past, to cases in which an entire set of images has a common problem on one of the sides. This was the case of a book, whose scanning platform, given a reflection it created, could not be distinguished from the rest of the book and was consequently included in the clipping. By applying a reduction of the lower edge of the images, they were successfully cropped while maintaining the integrity of the pages and achieving the desired result;

- The last practical feature, but not linked to the image processing process, is given by the possibility of enabling, for each processed image, a viewer of the various steps to which it is subjected. It is mostly a tool for the developer, in order to understand any missing steps or highlight problems and look for corresponding solutions. This is simply an option that can be checked at the bottom of the application, however not recommended in case you have to process many images as it constantly slows down and interrupts the process.

Results

Figure 44: Original image

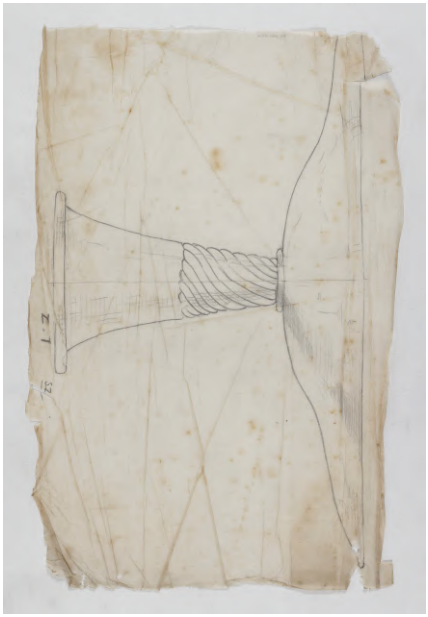

Figure 45: Processed image

At the end of the post production operations we finally have our finished product: the main object of the image has been cropped and, if indicated among the optional parameters, the image has probably also been balanced in terms of colors and a margin has been added. This tool has already been officially used for the post-production of some funds. Some examples are therefore attached below.

If further needs emerge and therefore there is a request to add additional features to the program, these would represent a first point on which to turn attention. In the meantime, it is certainly possible to broaden our horizons to try to look beyond the processes currently under consideration, further expanding the potential of the application.

A first expansion is already represented by the next sub-chapter, "Analyzing books", which deals with reusing the processed images with the purpose of carrying out an analytical investigation on them, recognizing the presence of particular elements through object detection models.

Another of the proposals under attention concerns the image deskewing, that is the "straightening" of the image perpendicular to the axes of the plane. The scanned images do not always turn out to be perfectly straight or aligned, which is why, in order to avoid manual retouching as much as possible, it can be a very useful feature. The program already tries to recognize the contours of the main

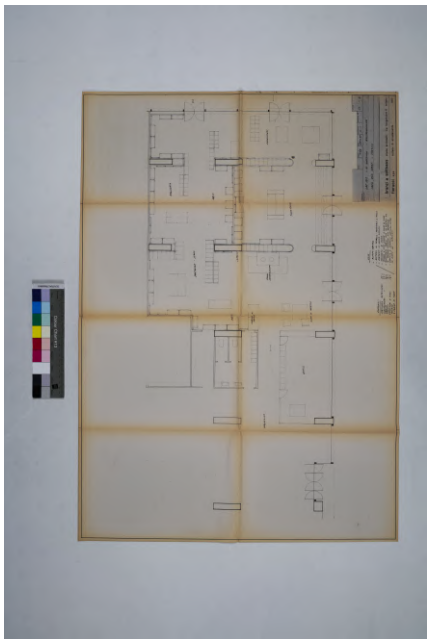

Figure 46: Original image

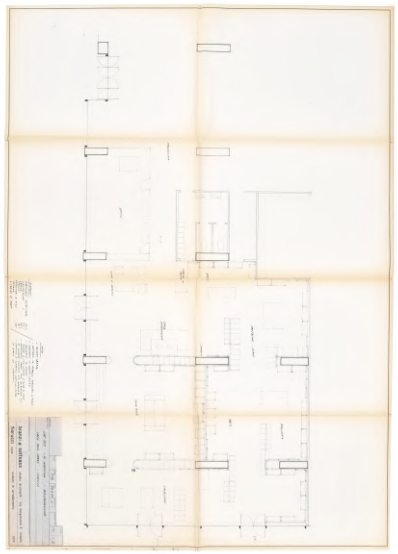

Figure 47: Processed image

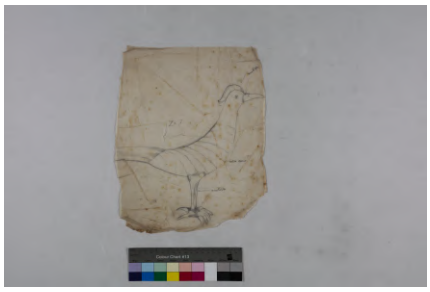

Figure 48: Original image

Figure 49: Processed image

Figure 50: Original image

Figure 51: Processed image

object, identifying its position, and cutting it out from the rest of the context, however it does not make any angular correction.

4.3 Features Detection

As mentioned in the last chapter, the images are processed in order to be published in the Foundation's digital library. However, the desire to expand horizons and explore the world of machine learning led to adding a further articulation to the project. After a series of dialogues, we found that one of the Foundation's interests was based on the automatic recognition, within the post-processed images, of particular elements. In detail, in the case of ancient books and manuscripts, the possibility of identifying the presence of drop caps and illustrations. The main goal was so to create an object detection model that was able to recognize these elements once sufficiently trained and organized.

At this point the way to go was identifying the easiest but better performing model to use and train, which we found in using Detecto. Detecto is a Python package that allows to build fully-functioning computer vision and object detection models. It can be used on still images and videos, transfer learning on custom datasets and serialization of models to files. Detecto is also built on top of PyTorch, which is another big library used in ML. Along with Detecto we are using, for some functions, PyTorch and Matplotlib.

Method

Now we take into consideration every single step that led to the development of the model:

- We first consider the potential dataset. We start from a base of numerous images belonging to different funds, however without any reference of our interest. Our object detection model, in fact, cannot identify something for which it has not been previously trained. For more details about the data, see Data;
- Once our expectations are clear in mind, we must first separate the group of images into two sections: a training dataset and a validation dataset. This subdivision is carried out to allow the model to train on a part of the images (it is normally advisable to use about 80% of the total images in training), being able to then check the results of your training on unpublished images, never seen before. (20% remaining images);

```
labeled
|__ training: [0044.jpg, 0048.jpg, 0075.jpg ....]
|__ validation: [0003.jpg, 0010.jpg, 0045.jpg ...]
|__ train_labels: [0044.xml, 0048.xml, 0075.xml ....]
|__ val_labels: [0003.xml, 0010.xml, 0045.xml ...]
```

Figure 52: Dataset structure tree

- Now, the most manual and, unfortunately, tedious part: the labeling to be done individually, image by image. Assuming that we want our model to identify elements such as drop caps and illustrations, it is necessary to establish what a drop cap or illustration actually represents within an image. To do this, we will rely on a program commonly used in machine learning, LabelImg. This, in fact, allows you to easily create "bounding boxes", squares around the area of our interest, creating a label with the name of the corresponding feature (in our case, drop cap or illustration);
- The work performed by LabelImg is fundamental, creating an XML file for each modified image containing all the metadata relating to our bounding boxes. From a code process point of view, however, going through numerous files can drastically lengthen program times. For this reason it is necessary to merge and convert the XML files into two CSV files, which will become our training labels and validation labels, i.e. those references that will allow our model to understand how to train and if the detection it will make will be correct;

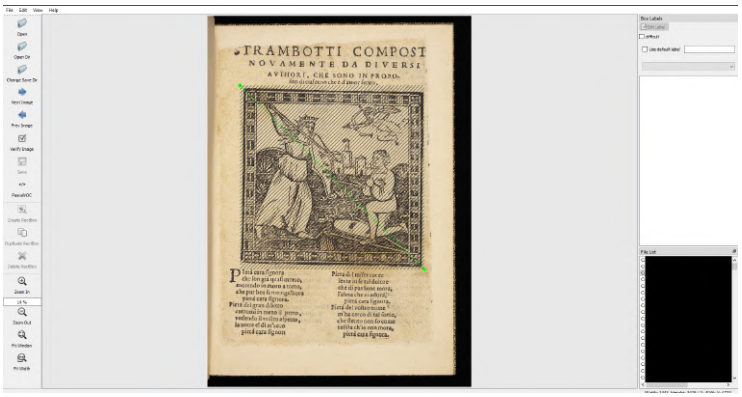

Figure 53: Example of image labeling with LabelImg

- It is recommended, in order to expand our dataset and make it more effective, to use data augmentation. As explained in the previous chapters, this is a technique capable of enriching our training dataset with new images modified through different types of alterations, reusing the same images already present. In our case, assuming we do not end up with completely crooked or translated pages, We will just apply a flip on the horizontal axis, essentially turning the image upside down;

- Finally we can create the model, and we want to be sure to correctly enter all the parameters that are right for us. In fact, the training of the model requires the following data: the training dataset, which we have just completed with the latest data augmentation additions, and the validation dataset, including labels for both; a batch size, which defines the number of samples that will be propagated through the network (how many samples are processed at a time); the number of epochs, i.e. a parameter that defines the number times that the learning algorithm will work through the entire training dataset. To describe what epochs and batches are, we can think of a for-loop over the number of epochs where each loop proceeds over the training dataset. Within this for-loop is another nested for-loop that iterates over each batch of samples, where one batch has the specified batch size number of samples. But why are we iterating our model for a defined amount of epochs? Since our model uses a gradient descent method, one of the most used algorithms in deep learning, we can say that each epoch finds the approximate direction that each parameter needs to be adjusted but it's just approximate. Adjusting too much in one single epoch will overshoot the set of parameters we're actually trying to find. Further explanation can be found in the example on p. 31.

  In this phase we will use a pre-trained model of PyTorch, that is Faster-RCNN_ResNet50_FPN_Coco-258fb6c6 (or just ResNet50, a CNN provided

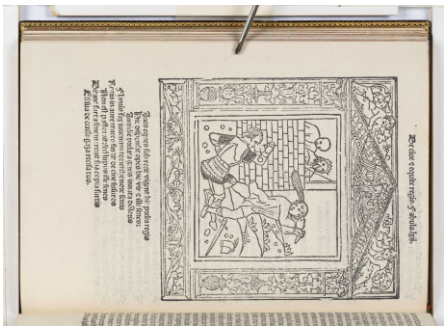

Figure 54: Original image

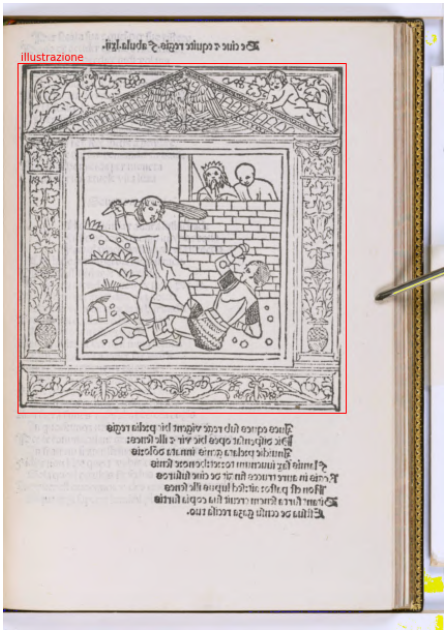

Figure 55: Augmentation example

by Keras, a Python Deep Learning library), a neural network with already starting provisions regarding the identification of a wide range of objects. Simply put, we are applying transfer learning.

Residual Network (ResNet) is an efficient framework in training deeper neural network. Like any Deep Learning framework the layers are nothing but nonlinear processing units for feature extraction and transformation. They also include hidden layers of an artificial neural network and sets of propositional formulas.

In general, in a deep convolutional neural network, several layers are stacked and are trained to the task at hand. The network learns several low/mid/high level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some $(n + x)$th layer). It has proved that training this form of networks is easier than training simple deep convolutional neural networks and also the problem of degrading accuracy is resolved. In plain networks, as the number of layers increases, even after thousands of iterations, training error will become worse and worse. In theory, we expect having a deeper network should only help but in reality, the deeper network has higher training error, and thus test error. In fact, when deeper networks are able to start converging, a degradation problem

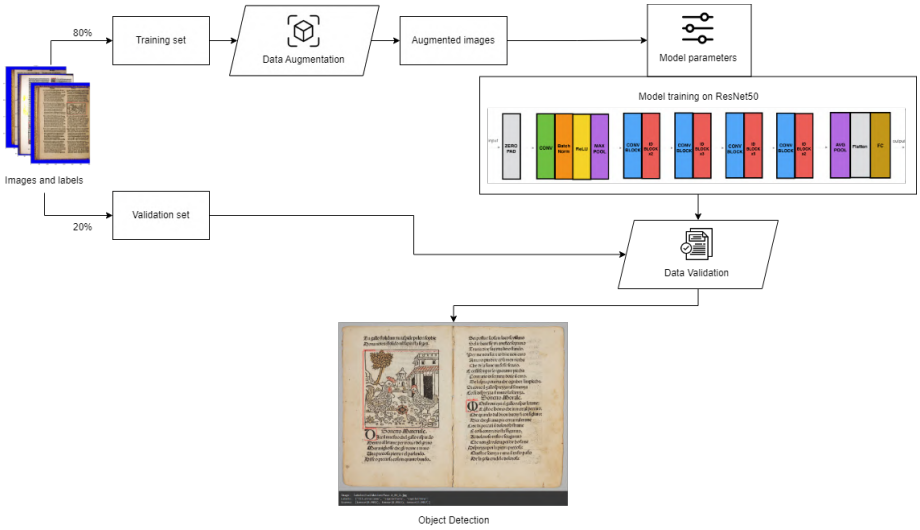

Figure 56: Workflow of the process, from the starting images to the final detections, which are shown and explained in the following pages (the model image is taken from https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33, by Priya Dwivedi)

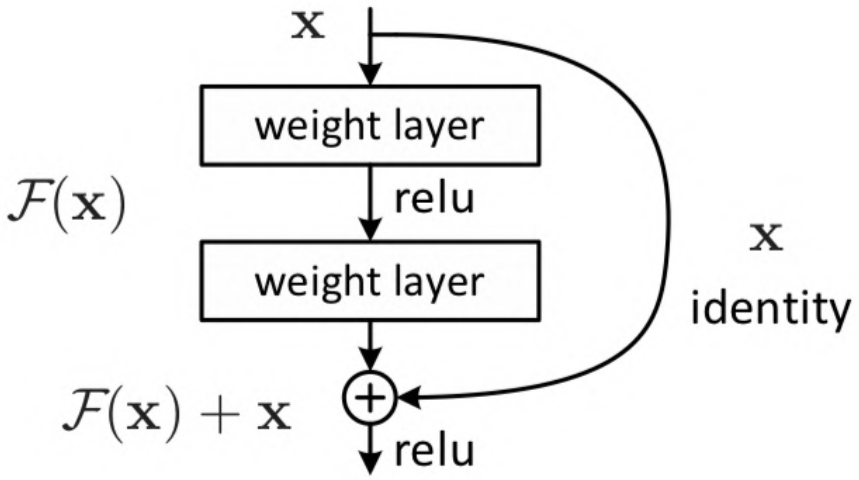

Figure 57: Example of a residual block, skipping "useless" layers

gets exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. The approach is to add a shortcut or a skip connection that allows information to flow more easily from one layer to the next "useful" layer, i.e., bypassing data along with the normal CNN flow from one layer to another (Fig. 57).

Two main types of blocks are used in a ResNet, depending mainly on whether the input/output dimensions are same or different:

The Identity block: the standard block used in ResNets and corresponds to the case where the input activation has the same dimension as the output activation;

The Convolutional block: we can use this type of block when the input and output dimensions don't match up. The difference with the identity block is that there is a CONV2D layer in the shortcut path.

The ResNet-50 model consists of 5 stages each with a Convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. [4].

Cross validation accuracy

Accuracy test made on 10 random different subdivisions of the same dataset.

K-fold	1	2	3	4	5	6	7	8	9	10	Mean	Std
Images accuracy	96,52%	96,09%	97,39%	97,83%	97,83%	94,35%	95,22%	94,78%	96,09%	98,26%	96,44%	15,18
Drop caps accuracy	90,48%	93,94%	92,78%	91,55%	95,71%	88,19%	88,04%	85,71%	90,80%	95,16%	91.24%	3,10
Illustrations accuracy	95,06%	92,68%	100,00%	98,59%	97,40%	96,25%	95,12%	95,31%	93,51%	95,89%	95.98%	2,10

Creato con Datawrapper

Figure 58: Cross validation on 10 different folds - Object Detection model (Our method)

Cross validation accuracy

Accuracy test made on 10 random subdivisions of the same dataset.

K-fold	1	2	3	4	5	6	7	8	9	10	Mean	Std
Training accuracy	69,36%	71,21%	76,99%	72,96%	68,16%	67,94%	70,77%	76,23%	80,37%	84,41%	73,84%	5,52
Validation accuracy	63,32%	63,84%	62,45%	66,38%	68,56%	71,18%	66,81%	67,25%	63,32%	65,07%	65,82%	2,75

Creato con Datawrapper

Figure 59: Cross validation on 10 different folds - Image Classification model (Compared method); Validation accuracy here equals to the "Images accuracy" of our method

Having relied on an Object Detection process instead of resorting to Image Classification (see the comparison in proved to be fundamental for various reasons: the type of analysis we wanted to carry out had the aim of focusing on the classes of elements to be identified, rather than considering the entire images as belonging to individual classes. In fact, in the coexistence of several classes within the same image, the image classification system stops working for us. Assuming also how it could be useful in any case, it was statistically found that focusing on object detection was also much more efficient and effective in terms of results and timing, with much higher accuracy.

The Image Classification model, made using TensorFlow/Keras, consists of three convolution blocks with a max pool layer in each of them. There's a fully connected layer with 512 units on top of it that is activated by a ReLu activation function. The model outputs class probabilities based on binary classification by the sigmoid activation function. As we did in creating the object detection model, also in this case data augmentation parameters were applied. We also used an optimizer and a loss function for our model. The role of the optimizer is to measure how good our model predicted output is when compared with true output. If the loss is high optimizers change the attributes of the neural network such as weights and learning rate in order to reduce the losses. For this model, we chose the ADAM optimizer and binary cross entropy loss function.

After doing some tests, we can confirm how getting into more than 8-10 epochs will lead us to overfitting the model, since the well smoothed curve we get tells us that our path is correct and that we got a good result.

Since we need to be sure that our model and dataset are valid enough, we need to perform cross validation tests. It is a validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations.

Different training and validation sets, within the same dataset, have given quite similar results. At the end, the accuracy of the model on a the cross-validation test, made up with 10 randomized trainings, gave a result between 95-98% with a standard deviation of 1-1.5, while the accuracies of the single classes stood between 85-95% for the drop caps and 92-100% for the illustrations (. At this point we can think of not having to use dropout or other regularization methods, since the use of data augmentation proved sufficient to avoid overfitting and the results are what we were aiming for.

In the case of the Image Classification model, after 10 epochs the validation accuracy (so the capacity of our model to classify never-seen-before images) average on 10 folds does not go higher than 71%. Even after trying going up to 20 epochs (double those used in the object detection process), the ability of the model to distinguish the images illustrated from those in which only text is present is still particularly complicated, up to an average accuracy around "only" 79%, compared with the values close to 95-100% obtained with the other method. We must also take into account the fact that it is not possible in the case of image classification to apply a threshold to improve the results, since for each image we will always have only one result, the estimate accuracy of

belonging of the image to a certain category, without giving space to other interpretations.

Model	Processing duration
Object detection	~50 minutes
Image classification	~21 minutes

Figure 60: Comparison of processing timings between the two models

We must point out, however, that the Image Classification model is much faster than the Object Detection model, given the same number of epochs. But as mentioned previously, even doubling the timing of the fastest model is not enough to obtain the same results as the Object Detection model, confirming the latter model best effectiveness.

At the end of the process, our model will be ready and we will be able to ascertain its accuracy based on the counter-test carried out with the validation dataset during the training phase. At this point, if we are satisfied with the result, it is possible to save it and keep it for any kind of future operation. Conversely, if we are not happy with the final outcome, we can work again on the training dataset or model parameters to look for a better solution. In any case, our model will now be able to make detections on new images it has never seen, establishing the presence or absence of objects on which it has been trained.

Results

In this final part, we will present some examples of detection. Currently, given the needs, it is possible to make single detections (using manually selected or random images from a list) or detections on entire folders, returning the number of elements identified for each predetermined label, the number of "empty" images, i.e. of identified elements. For each subject to detect, the output of the code returns the image itself with red identification boxes with the detection label (, so what has been identified, and the detection accuracy percentage attached (only in the case of single images, otherwise it would be extremely time consuming to show all the images of a group, especially in they're a lot), together with a small summary out-of-the-image composed of the image name, labels

Figure 61: Results of image classification distinguishing text-only ("solo testo") images from illustrated ("illustrata") ones

of the detections made and the corresponding scores, that is the percentage of accuracy. Since from this process we can draw off many information, we can also compare the results with the ground-truth values (information that is known to be real or true, which we can effectively use to compare with detections, and how many features occur in how many pages (.

One of the most important elements in the model analysis phase was represented by its accuracy level. As we saw in several trials on the same randomly mixed dataset returned more than positive results. However, those results were only possible by excluding all those detections that were identified by the model with very low percentages. As can be seen in different threshold levels applied to the results of the dataset analysis made it possible to understand how, below a certain threshold, the results were completely distorted by detection with very low estimation. This made it necessary, therefore, to evaluate the right threshold to avoid mainly two circumstances: having an excessively high number of false positives and having a number of false negatives equal to 0, thus allowing

Figure 62: Example of single image detection outcome

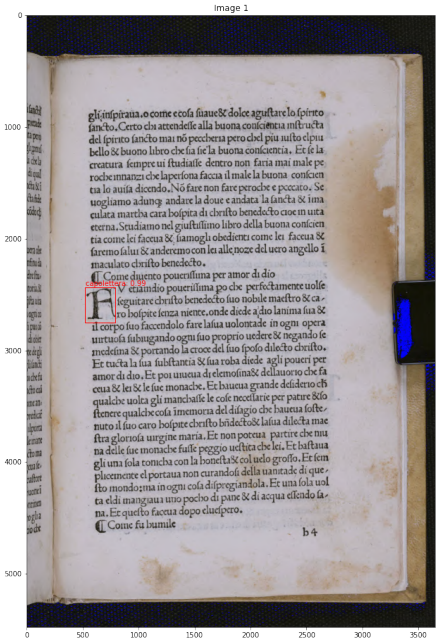

Figure 63: Detection example 1

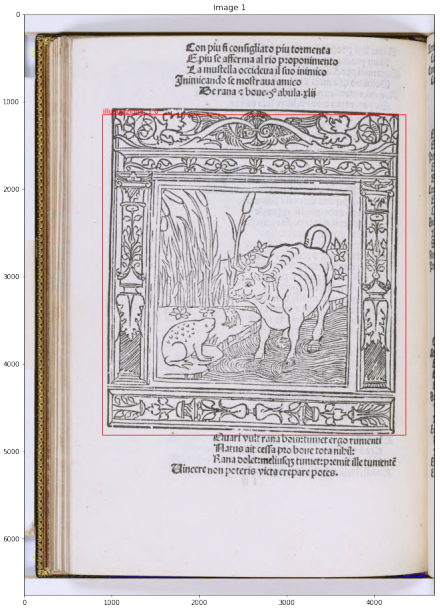

Figure 64: Detection example 2

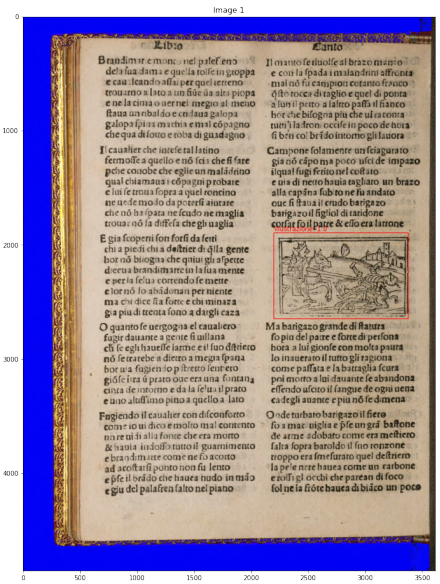

Figure 65: Detection example 3

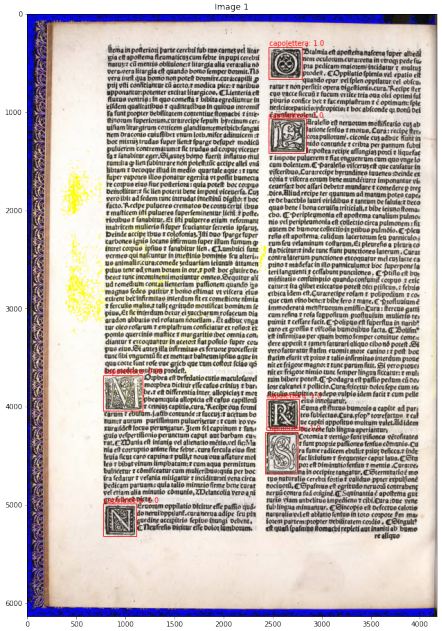

Figure 66: Detection example 4

Number of detected occurrences of drop caps vs illustrations

Given the validation set of images, the images present the following detected combinations of drop caps and illustrations.

Number of drop caps	Number of illustrations: 0	1	2	3
0	116	64	1	2
1	27	5	1	0
2	6	5	0	0
3	1	1	0	0
4	0	0	0	0
5	0	0	0	0
6	1	0	0	0

Figure 67: Contingency table regarding the number of images that have drop caps and/or illustrations. The sum corresponds to the number of images of our validation set, 230

Detection filtering

By applying a threshold to our detections we can discard all the false positives given by very low accuracy results.

Threshold	95%	90%	70%	50%	30%	20%	10%	0%
Drop cap errors	6	5	4	5	8	11	72	100
Illustration errors	0	0	2	2	2	3	57	61
False negatives	2	1	0	0	0	0	0	0
False positives	6	5	6	7	10	14	139	161
Precision	**98%**	**98,3%**	**98%**	**97,7%**	**96,7%**	**95,4%**	**54,5%**	**47,4%**
Recall	**99,3%**	**99,6%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

The sum of drop cap and illustration errors makes up to the number of false positives. False negatives are the missed ground truth detections.

Figure 68: Precision and Recall results based on different thresholds

us to have a recall value equal to 100%, equivalent to the complete acquisition of the ground truth values of the dataset. To recap, false negatives are the drop caps/illustrations that are present among the ground truth labels but are not found by the model, while false positives are detections made by the model which are not corresponding to a ground truth label.

Based on the fact that we wanted to ensure maximum recall with maximum precision, it was found that the 70% threshold is the most balanced value, thus minimizing losses.

5 Conclusions and future works

After considering alternative possibilities we will now see possible future developments.

As for the postproduction of the images, the system may seem quite cumbersome but sufficiently accurate (an estimate is between 85 and 90% accuracy), which however may seem inconvenient on large numbers (out of 25000 images over 2500 will be wrong, a considerable number anyway). While still widely used in the world of digital photography, classic image processing has its limitations and difficulties when we find ourselves with a wide range of images available. In order to be able to correctly process as much as possible, it is necessary to adjust the code in order to allow the application to correctly distinguish the various parts of the image.

However, today's immense growth in the use of artificial intelligence in this field has led to rethink what has been developed so far. For example, just think of the very recent birth and growth of StableDiffusion, an open-source machine learning, text-to-image model able to generate digital images from natural language descriptions. Reason that prompted us, on the wave of that, to think of creating a double intelligent model able to first recognize the object to be extracted and then analyze it in detail and recognize its various peculiarities, clearly on the basis of the pre-established classes during the development phase. However, we remind you that the current system works very well and is able to guarantee excellent results, even if they can always be improved in order to minimize errors and, consequently, manual corrections as much as possible.

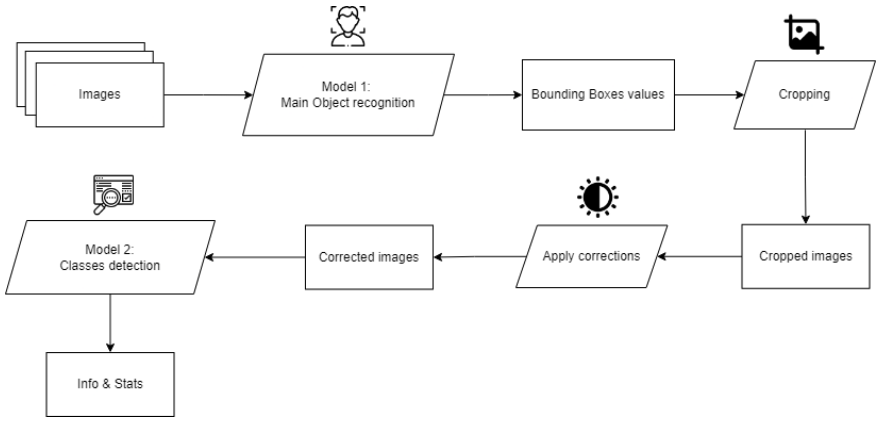

Figure 69: Potential workflow of the double model system

This new system will obviously imply the need to train two different models based on our needs, first using a dataset of RAW images to be labeled, indicating for each image the presence of the main object and the color chart, always

present. It will be essential, although objectively useless for the purposes of our acquisitions, to indicate the presence of the latter, in order to avoid that the model can confuse this element with the object we want to extract. A second model will then follow, this time created from a dataset of images already fully processed, capable of recognizing the classes of elements that we will label on each image on which the model will be trained.

The hope is to obtain two extremely effective models, able to minimize the margins of error considering the already important success obtained with the already existing object detection model.

Acronyms

ADAM Adaptive Momentum Estimation.

BOW Bag of Words.

CLI Command Line Interface.

CMY Cyan Magenta Yellow.

CNN Convolutional Neural Network.

CONV2D Convolutional 2D layer.

CR2 Canon Raw version 2.

CSV Comma-separated Values.

DCNN Deep Convolutional Neural Network.

DL Deep Learning.

FC Fully Connected.

GAP Global Average Pooling.

GPU Graphics Processing Unit.

HLS Hue Saturation Lightness.

HOG Histogram of Oriented Gradients.

HOOSC Histogram of Orientation Shape Context.

HPC High Performance Computing.

JPG Joint Photographic Experts Group.

JSON JavaScript Object Notation.

LLNL Lawrence Livermore National Laboratory.

MB MegaBytes.

ML Machine Learning.

MLP MultiLayer Perceptron.

MNIST modified National Institute of Standards and Technology.

MSII Maximum Segmented Image Information.

NLP Natural Language Processing.

OCR Optical Character Recognition.

PHP PHP: Hypertext Preprocessor.

PIL Python Imaging Library.

PP3 PlanetPress Suite Form File.

R-CNN Region Based Convolutional Neural Networks.

ReLU Rectified Linear Unit.

ResNet Residual Network.

RGB Red Green Blue.

RPN Region Proposal Part.

SIFT Scale Invariant Feature Transform.

SVHN Street View House Numbers.

SVM Support Vector Machine.

TIF Tagged Image File Format.

TL Transfer Learning.

UI User Interface.

XML eXtensible Markup Language.

YUV Luminance (Y) Blue projection (U) Red projection (V).

References

[1] Adeel A; Gogate M; Hussain A. "Contextual deep learning-based audio-visual switching for speech enhancement in real-world environments". In: *Inf Fusion* (2020).

[2] Crowley E J; Zisserman A. "In search of art". In: *Workshop at the European Conference on Computer Vision* (2014).

[3] Crowley E J; Zisserman A. "The Art of Detection". In: *European Conference on Computer Vision* (2016).

[4] Kaushik A. *Understanding ResNet50 architecture*. 2022. URL: https://iq.opengenus.org/resnet50-architecture/.

[5] Koschan A; Abidi M A. "Digital Color Image Processing". In: *Wiley & Sons, Inc* (2007).

[6] Cyganek B. "Object Detection and Recognition in Digital Images: Theory and Practice". In: *CVPR* (2013).

[7] Dalal N; Triggs B. "Histogram of oriented gradients for human detection". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005).

[8] Dalal N; Triggs B. "Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition". In: *(CVPR'05), vol. 1. IEEE* (2005).

[9] Hubert M; Bogacz B. "A Bridge to Digital Humanities: Geometric Methods and Machine Learning for Analysing Ancient Script in 3D". In: *CAA2015. Keep The Revolution Going: Proceedings of the 43rd Annual Conference on Computer Application and Quantitative Methods in Archaeology* (2015).

[10] Van Essen B; Kim H; Pearce R; Boakye K; Chen B. "Lbann: livermore big artificial neural network HPC toolkit". In: *Proceedings of the workshop on machine learning in high-performance computing environments* (2015).

[11] Florea C; Badea M; Florea L; Vertan C. "Domain transfer for delving into deep networks capacity to de-abstract art". In: *Scandinavian Conference on Image Analysis* (2017).

[12] LeCun Y; Cortes C; JC Burges C. *THE MNIST DATABASE of handwritten digits*. 1998. URL: http://yann.lecun.com/exdb/mnist/.

[13] Tan C; Sun F; Kong T; Zhang W; Yang C; Liu C. "A survey on deep transfer learning". In: *International conference on artificial neural networks: Springer* (2018).

[14] Thiago Carvalho. *Basics of Kernels and Convolutions with OpenCV*. 2020. URL: https://towardsdatascience.com/basics-of-kernels-and-convolutions-with-opencv-c15311ab8f55.

[15] Fondazione Giorgio Cini. *Who we are*. URL: https://www.cini.it/en/who-we-are.

[16] Hu R; Gayol C P; Odobez J; Gatica-Perez D. *Analyzing and visualizing ancient Maya hieroglyphics using shape: From computer vision to Digital Humanities*. 2017. URL: https://academic.oup.com/dsh/article/32/suppl_2/ii179/3943649.

[17] Marr D. "Vision: A Computational Investigation into the Human Representation and Processing of Visual Information". In: *Freeman* (1982).

[18] Weiss K; Khoshgoftaar TM; Wang D. "A survey of transfer learning". In: *J Big Data* (2016).

[19] Donahue J; Jia Y; Vinyals O; Hoffman J; Zhang N; Tzeng E; Darrell T; Decaf. "A deep convolutional activation feature for generic visual recognition". In: *Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32* (2014).

[20] Lowe DG. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision, vol. 2. IEEE* (1999).

[21] Van Noord N; Postma E. "Learning scale-variant and scale-invariant features for deep image classification". In: *Pattern Recognition 61* (2017).

[22] Seguin B; Striolo C; DiLenardo I; Kaplan F. "Visual Link Retrieval in a Database of Paintings". In: *Computer Vision – ECCV* (2016).

[23] Liu W; Wang Z; Liu X; Zeng N; Liu Y; Alsaadi FE. "A survey of deep neural network architectures and their applications". In: *Neurocomputing* (2017).

[24] Potok TE; Schuman C; Young S; Patton R; Spedalieri F; Liu J; Yao KT; Rose G; Chakma G. "A study of complex deep learning networks on high-performance, neuromorphic, and quantum computers". In: *ACM J Emerg Technol Comput Syst (JETC)* (2018).

[25] Malzbender T; Gelb D; Wolters H. "Polynomial Texture Maps". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001).

[26] Najman L; Talbot H. "Mathematical Morphology: From Theory to Applications". In: *John Wiley Sons, Inc* (2013).

[27] Yosinski J; Clune J; Bengio Y; Lipson H. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems: San Mateo: Morgan Kaufmann Publishers* (2014).

[28] Li HC; Deng ZY; Chiang HH; "Lightweight and resource-constrained learning network for face recognition with performance optimization". In: *Sensors* (2020).

[29] Takahara T; Ueda T Hirahara D; Takaya E; "Effects of data count and image scaling on deep learning training". In: *PeerJ Comput Sci* (2020).

[30] Belongie S; Malik J; Puzicha J. "Shape matching and object recognition using shape contexts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002).

[31] Canny J. "A computational approach to edge detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986).

[32] He K; Zhang X; Ren S; Sun J. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016).

[33] Karimi H; Derr T; Tang J. "Characterizing the decision boundary of deep neural networks". In: *arXiv preprint* (2019).

[34] Ren S; He K; Girshick R; Sun J. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems 28* (2015).

[35] Hinton G; Srivastava N; Swersky K. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent". In: (2012).

[36] Inoue N; Furuta R; Yamasaki T; Aizawa K. "Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation". In: *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

[37] Alzubaidi L; Zhang J; AJ Humaidi; Al-Dujaili A; Ye Duan; Al-Shamma O; J Santamaría; Mohammed AF; Al-Amidie M; Farhan L. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *Journal of Big Data* (2021).

[38] Bottou L. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010: Springer* (2010).

[39] Deng J; Dong W; Socher R; Li LJ; Li K; Fei-Fei L. "Imagenet: a large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition.* (2009).

[40] Fang W; Love PE; Luo H; Ding L. "Computer vision for behaviour-based safety in construction: a review and future directions". In: *Adv Eng Inform.* (2020).

[41] Schlecht J; Bell P; Carqué B; Ommer B; Saurma LE. *Computer-Assisted Detection and Analysis of Medieval Legal Gestures.* 2012. URL: https://sabinelang254.wixsite.com/digital-humanities/gesture-detection.

[42] Marchand-Maillet S; Sharaiha Y M. "Binary Digital Image Processing: A Discrete Approach". In: *Academic Press* (1999).

[43] Piotrowski M. *Machine Learning for the Humanities: A very short introduction and a not-so-short reflection.* 2020. URL: https://latex-ninja.com/2020/10/25/machine-learning-for-the-humanities-a-very-short-introduction-and-a-not-so-short-reflection/.

[44] Stone AJ; Zender M. "Reading Maya Art: A Hieroglyphic Guide to Ancient Maya Painting and Sculpture". In: *Thames & Hudson Limited Publisher* (2011).

[45] Strezoski G; Worring M. "OmniArt: Multi-task Deep Learning for Artistic Data Analysis". In: *arXiv:1708.00684* (2017).

[46] Tarte S M. "Digitizing the Act of Papyrological Interpretation: negotiating spurious exactitude and genuine uncertainty". In: *Lit Linguist Computing* (2011).

[47] Terras M. "Artefacts and Errors: acknowledging issues of representation in the digital imaging of ancient texts". In: *Codicology and Papyrology in the Digital Age, II* (2012).

[48] Terras M. "Image processing in the Digital Humanities". In: *Digital Humanities in Practice* (2012).

[49] Terras M. "The Digital Classicist: disciplinary focus and interdisciplinary vision". In: *Digital Research in the study of Classical Antiquity* (2010).

[50] Thaller M. "Images and Manuscripts in Historical Computing". In: *Proceedings of a workshop on 15 November 1991, organized by the International University Institute, Firenze* (1992).

[51] Yin R; Monson E; Honig E; Daubechies I; Maggioni M. "Object recognition in art drawings: Transfer of a neural network". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference* (2016).

[52] Rozenwald MB; Galitsyna AA; Sapunov GV; Khrameeva EE; Gelfand MS. "A machine learning framework for the prediction of chromatin folding in Drosophila using epigenetic features". In: *PeerJ Comput Sci* (2020).

[53] Brady M; Pan X; Schenck V; Terras M; Robertson P; Molton N. "Shadow Stereo, Image Filtering and Constraint Propagation". In: *Images and Artefacts of the Ancient World, Vol. 4* (2005).

[54] Wu L; Hoi SC; Yu N. "Semantics-preserving bag-of-words models and applications". In: *IEEE Trans Image Process* (2010).

[55] Baheti P. *Train Test Validation Split: How To Best Practices*. 2022. URL: https://www.v7labs.com/blog/train-validation-test-set.

[56] Carvelli L; Olesen AN; Brink-Kjær A; Leary EB; Peppard PE; Mignot E; Sørensen HB; Jennum P. "Design of a deep learning model for automatic scoring of periodic and non-periodic leg movements during sleep validated against multiple human experts". In: *Sleep Med.* (2020).

[57] Goskar T A; Earl G P. "Polynomial Texture Mapping for Archaeologists". In: *British Archaeology* (2010).

[58] Westlake N; Cai H; Hall P. "Detecting people in artwork with cnns". In: *ECCV Workshops* (2016).

[59] Palaz D; Magimai-Doss M; Collobert R. "End-to-end acoustic modeling using convolutional neural networks for hmm-based automatic speech recognition". In: *Speech Commun* (2019).

[60] Sonka M; Hlavac V; Boyle R. "Image Processing, Analysis, and Machine Vision". In: *Cengage Learning* (2007).

[61] Ruder S. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint* (2016).

[62] Alzubaidi L; Fadhel MA; Al-Shamma O; Zhang J; Santamaría J; Duan Y; Oleiwi SR. “Towards a better understanding of transfer learning for medical imaging: a case study”. In: *Appl Sci* (2020).

[63] Saleh AM; Hamoud T. “Analysis and best parameters selection for person recognition based on gait model using CNN algorithm and image augmentation”. In: *J Big Data* (2021).

[64] Shorten C; Khoshgoftaar TM. “A survey on image data augmentation for deep learning”. In: *J Big Data* (2019).

[65] Hubel DH; Wiesel TN. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *Physiol* (1962).

[66] Kornblith S; Shlens J; Le Q V. “Do Better ImageNet Models Transfer Better?” In: *arXiv:1805.08974* (2018).

[67] Plataniotis K N; Venetsanopoulos. “Color Image Processing and Applications”. In: *Springer Verlag* (2000).

[68] Pan W. “A survey of transfer learning for collaborative recommendation with auxiliary data”. In: *Neurocomputing* (2016).

[69] Gevers T; Gijsenij A; van de Weijer J; Geusebroek J-M. “Color in Computer Vision: Fundamentals and Application”. In: *Wiley & Sons, Inc* (2012).

[70] Li Y; Ding L; Gao X. “On the decision boundary of deep neural networks”. In: *arXiv preprint* (2018).

[71] Zhang Z. “Improved Adam optimizer for deep neural networks”. In: *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)* (2018).