Visual Inspection Using Deep Learning Techniques for Industrial Manufacturing Processes with Class Imbalance and Limited Labeled Data

Nicolás Barrero Lizarazo

Department of Industrial Engineering

Pontifical Xavieran University

This document is submitted as Partial Fulfillment of the Requirements for the Degree of

Master of Science in Industrial Engineering

School of Engineering December 2020

I would like to dedicate this work to my loving parents and family.

To the students out there passionate for technology and *experimentation*. *cacharrear*

To the whole team of the CTAI laboratory.

To Lenny, Luc and other robots involved in my academic and learning process at CTAI.

To dreamers.

Acknowledgements

I would like to acknowledge PhD Carol Martinez, for teaching me the “dark side” of industrial engineering at PUJ. I am so thankful for her constant support and guidance throughout these years.

Also, I would like to thank PhDs Ivan Mondragon and Marta Manrique and, Msc Wilson Hernandez for their trust letting me experiment with the finest state-of-the-art technology at the *Centro Tecnológico de Automatización Industrial* CTAI.

Abstract

In recent years computer vision have been used to perform visual inspection in industrial processes. However, conventional image processing is limited to high-controlled scenarios and specific tasks such bar-code reading and identification, gauging and measurement, basic material inspection and, location and counting. In industrial processes other visual inspection tasks require well-developed techniques, for instance, advanced defect detection and segmentation, feature location, and assembly verification, amongst others. Nowadays, these tasks implement machine learning techniques based on deep learning to achieve outstanding performance over manual inspection. Usually, it is believed that the deep learning approach requires many labeled data to generate an acceptable model, this often hinders the widespread adoption of deep learning in industries which are limited to the availability of labeled data. Additionally, as it is natural in visual inspection, defects are present less frequent than regular pieces, thus creating a class imbalance that could complicate the training of a robust deep learning model. On the other hand, other machine learning techniques applied to visual inspection require expertise in feature engineering to create a model and often the results are limited to very specific characteristics that cannot be extrapolated to a different environment, material or characteristic to be identified. The purpose of this project is to deploy a deep learning algorithm for visual inspection that deals with class imbalance and data availability in a case of study (steel sheet surface inspection), presenting different approaches used to achieve the final model.

Table of contents

List of figures

Nomenclature

Other Symbols

E	Experience
P	Performance metric
T	Task

Acronyms / Abbreviations

AI	Artificial intelligence
API	Application programming interface
CNN	Convolutional Neural Network
CPPS	Cyber physical production system
CPS	Cyber physical system
CPU	Central processing unit
CSV	Comma-separated values
CTAI	*Centro Técnologico de Automatización Industrial* (Technological Center of Industrial Automation)

CV Computer vision

DSC Sørensen–Dice coefficient

GAN Generative Adversarial Networks

GPU Graphics processing unit

HOG Histogram of oriented gradients

IaaS Infrastructure as a service

ICT Information and communications technology

IoT Internet of things

IoU Intersection over union

LSTM Long short-term memory

MAPI Manufacturers alliance for productivity and innovation

ML Machine learning

MLP Multi layer perceptron

PaaS Platform as a service

RAM Random access memory

ReLu Rectified linear unit

RGB Red, blue, green

RLE Run-length encoding

SaaS Software as a service

SGD Stochastic gradient descend

SIFT Scale-invariant feature transform

SME Small and median enterprise

SMOTE Synthetic Minority Over-sampling Technique

Chapter 1

Problem Context

1.1 Background & Precedents

"Most engineering and manufacturing problems are data-rich but knowledge-sparce."[1]

Since the apparition of different disruptive technologies at the end of the 1990's, industry has experienced a constant growing towards digitization around the world. These technologies, that initially appeared independently one from the other, are usually referred by different authors as triggers of what is currently known as industry 4.0 [2].

According to [2, 3] was the German government who first defined the fourth industrial revolution under the name of *Industrie 4.0*, declaring it as a key strategy for the year 2020. However, since the 2011 Hanover technology Fair, we can recognize the first approaches to this paradigm [2], even under the American-styled names *Smart Manufacturing* or *Smart factory* [4].

Industry 4.0 gathers some digital technologies, that once they are merged and integrated, they will transform the way quotidian activities are done [2, 3, 5]. Particularly in industry, the impact of the CPS (Cyber Physical systems) will be essential to adopt this new technological paradigm [2].

These systems aims to integrate the physical world with its virtual counterpart, in order to perform interrelated jobs with a high degree of coordination [2], focusing in an intelligent production, in which real time relationships among people, products and devices exist along a productive process. The application of a CPS in industry is formally known as a CPPS (cyber Physiscal Production System) [3].

According to [5] three main technologies have been identified as main enablers of this industrial revolution; *Internet of things* also known as IoT, *Cloud Computing* and *Predictive Analytics* [5]. However [2] outlines more than eleven technologies, some of them included in the main three categories above described. However, both authors agree that this is not an extensive list of the digital technologies involved in industry 4.0.

Although [2, 3, 5] agree that these technologies are key in order to adopt industry 4.0 through the CPPS paradigm, [5] identifies the importance of the transformation not only from a technological point of view but also directly from the business model which will impact the future of companies [5]. He proposes that, a transformation of the industries that are product-oriented to industries service-oriented is necessary. However, without leaving out the manufacturing processes. Under three schemes; SaaS (Software as a Service), IaaS (Infrastructure as a service) y PaaS (Platform as a Service) the companies of the future shall leverage on these transformation in order to maintain their competitiveness.

These innovative business models require a vertical and horizontal integration of data and information in which in a mid-term will become in a global network. [2, 3, 5] state that security and data protection are key to implement any data driven model. Also, [5] identifies that the greatest challenge is to close the gap between industry and academy, and states that the leak of research in technical and specific technologies hinders the widely adoption of these technologies.

No matter the different names there exists for the same concept; Industrie 4.0 in Germany, smart manufacturing in the USA and smart manufacturing in South Korea, manufacturing systems

faces more complex, dynamic and even chaotic behaviours. [6]. The never previously seen increase in data availability [4] referred also as Big data proposes a challenge for industries.

This amount of data hinders the ability of a company to identify valuable information from raw data. In fact as [7] proposes, when machine learning techniques are inappropriate integrated to a process it could lead not to only a negative impact, but also could create a distraction from main issues or lead to delayed or wrong conclusions.

In this new context one can summarize the challenges proposed by [6] in these key demanding tasks for industries:

- Adoption of advanced manufacturing technologies.
- Growth of the importance of manufacturing of high value added products.
- Utilization of advanced knowledge, information management and AI systems.
- Sustainable manufacturing (processes) and products.
- Agile and flexible enterprise capabilities including supply chain innovation in products, services, and processes.
- Close collaboration between industry and research to adopt new technologies.
- Apparition of new manufacturing management paradigms.

As [8] states, knowledge acquired by ML techniques could be used by process owners in their decision making or directly to improve the system. In other words one can derive knowledge out of data.

However, the field of machine learning is diverse and many algorithms, theories and methods are available. Besides, one can find that large amount of data adds a challenge due to its complexity, variety, high dimensionality as well as the natural NP-complete optimization problems present in

manufacturing scenarios [6]. This may represent a barrier regarding the adoption of these tools for the manufacturing industries and as [8] emphasizes, stored data becomes useful only when it is analyzed and turned into information.

ML has been successfully utilized in process optimization, monitoring and control applications in manufacturing, and predictive maintenance in different industries. Although, these applications are found to be limited on specific processes instead of the whole manufacturing program or manufacturing system [9].

The advantage of ML is that it handles high dimensional, multivariate data, extracting implicit relationships within large data-sets, in a complex and dynamic, even chaotic environments [10]. However, depending on the characteristic of the ML algorithm (supervised, unsupervised, reinforcement learning), the requirements towards the available data may vary. Particularly, the first approach to apply ML in manufacture would be the derivation of patterns from existing data sets [6].

A very common challenge of a ML implementation in manufacturing is the acquisition of relevant data, this includes its quality, composition metadata, labels etc. [6], and the ability to understand data in order to apply ML. Compared to traditional statistical methods where a lot of time is spent to extract information, in ML a lot of time es spent on preparing the data [8].

In manufacturing there is a common problem that data is not available for certain attributes [11], thus hybrid-approaches are becoming more common, promising better results than individual single algorithm applications [6].

For instance [12] proposes a system based on ensembles with deep convolutions neural networks for surface defect classification. They integrated different networks with data augmentation, and training them using various optimization methods, to finally combine the three models and create the predictor.

Since these methods rely on large-scale labeled samples [13] proposes a semi-supervised method in which they use enough label samples to train the model and unlabeled data is used to assist the learning process of the convolutional neural network, that is to say the authors used a technique called pseudo-label. They achieve an accuracy of 90.7% with this method which represents an improvement of 17.53% versus the ensemble method.

Other approach of automated surface inspection performed by [14] is based on transferring learning, where they extract characteristics for a pre-trained network and then use them to perform two tasks: image classification and image segmentation. This represent and improvement of 0.66%-25.5% in accuracy. This application can detect seven defect types.

Other ML technique frequently used is Support Vector Machines, in this case [15] proposes a SVM classificator that could identify defects on a glass surface, however with this approach the features are previously extracted using image processing and then the feature vector is used in the SVM. That's to say this approach implies more manual feature engineering in order to train the ML algorithm used.

Finally, other authors orientate their research towards unsupervised learning, thus dealing with the problem of the availability of labeled-data. [16] proposed a method based on a variance model, they state that since the steel surface inspection task is challenging for computer vision due to the patterns of defects, low contrasts, and pseudo defects they propose an anisotropic diffusion model to eliminate these pseudo-defects, and a Harr-Weibull-Variance model to characterize the regular texture of the steel surface and then detect arbitrary types of defects. They claim to reach 96.2% of average detection rate, however it is important to notice that this approach does not classify different type of defects only detects them.

The most recent approach found in unsupervised learning implies the utilization of Generative Adversarial Networks [17] in order to create artificial data. This could deal with data-shortage and also could be a possible solution for class imbalance problems, by keeping the intrinsic features

of the phenomena one would like to characterize. Then one could use these artificial data to train another model that would perform the defect classification or detection.

The complex structure and the diverse nature of the current ML techniques, and the natural behaviour of an industrial process implies, according to [6], that one have to take into account the following aspects when developing a ML model in manufacturing:

- Multi-variate scenario.
- High dimensional data sets.
- Adaptability to changing environments.

1.2 Problem Statement

"The advent of AI will lead to a significant reorganization of tasks in the manufacturing environment, displacing humans from repetitive and routine tasks, while creating new opportunities for them to focus on value-generating activities."[18]

Currently, we stand on the brink of a technological revolution that will fundamentally alter the way we live, work, and relate to one another. Industry 4.0 introduces what has been called the "smart factory". It is characterized by a fusion of technologies that "are blurring the lines between the physical, digital, and biological spheres"[19]. This model proposes an industry where automatic solutions will adopt the execution of versatile operations, which consist of operational and analytical components such as "autonomous" manufacturing cells, which independently control and optimize production [20].

The digitization of manufacturing is changing how products are designed, fabricated, used, and serviced, just as it's transforming the operations, processes, and energy footprint of factories and supply chains [21]. This transformation merges different technologies that had matured

enough to be applied in an industrial environment. Amongst them one can find Cloud computing, IoT, Block-chain, Collaborative Robotics and Data analytics.

Particularly, data analytics uses the data captured by different sensors in machines across processes to find useful information for decision-making at operational, tactical or strategic levels. As [2] proposes, unprocessed data may not provide meaningful value to decision-making in a cyber-physical production network unless these data are effectively analysed and utilised for manufacturing decisions.

In order to analyse the data generated in these environments, data science and data analytics techniques should be developed and employed [22]. However, building practical applications in which big data from different sources are integrated can be a challenging task [2].

Part of data analytics includes predictive analysis, that is to say, to evaluate and analyse what would happen in different scenarios based on historical data. Then, we could define that some of the more advanced predictive analytics techniques, that are able to recognize patterns by themselves, are also well known under the name of machine learning algorithms which are part of a greater paradigm called Artificial intelligence (AI).

As industries try to dive into AI solutions, data scarcity has emerged as a major challenge, collecting massive training sets in manufacturing is often not feasible [18]. If we consider industries where Six Sigma practices are implemented one can find three to four defects per million parts. The rarity of these defects makes it challenging to have sufficient defect data to train visual inspection models [23].

As machine learning algorithms depend on the ability to train on data-sets, that relevant data has to exist within the company in order to enable the AI application [18]. In 2019, the Manufacturers Alliance for Productivity and Innovation foundation (MAPI), created a report about the impact that AI will have in manufacturing industries. They report, as shown in figure 1.1, that “data maturity” to implement machine learning techniques has not been achieved yet

by the majority of companies in the United States. Only 11% are fully data mature, while over one-quarter are only partly so. About half of companies could become data mature within five years, but 14% of companies will not become so within the foreseeable future [18].

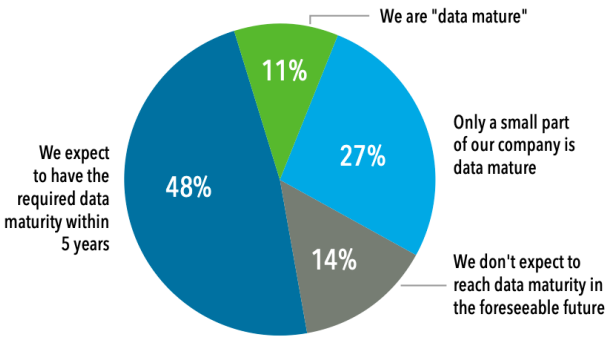
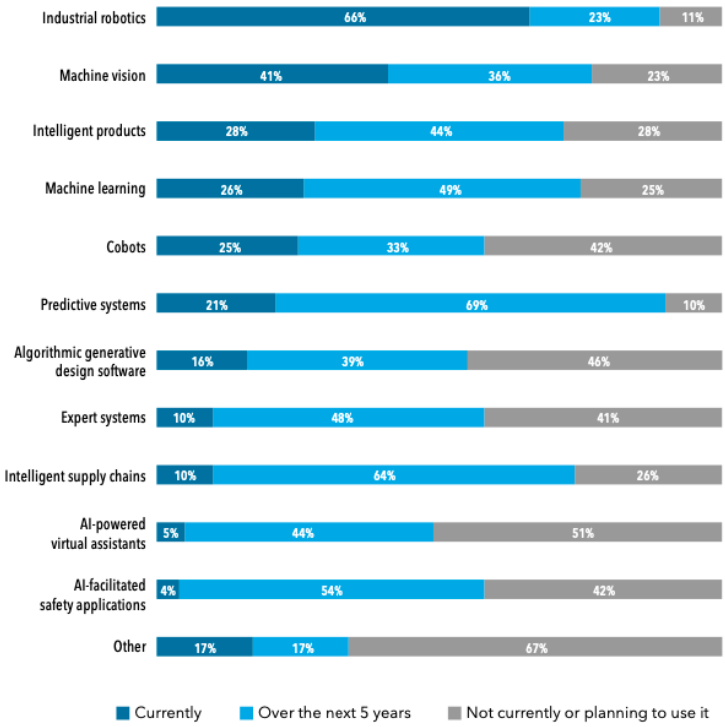


Fig. 1.1 “Data maturity” for manufactures according to [18]

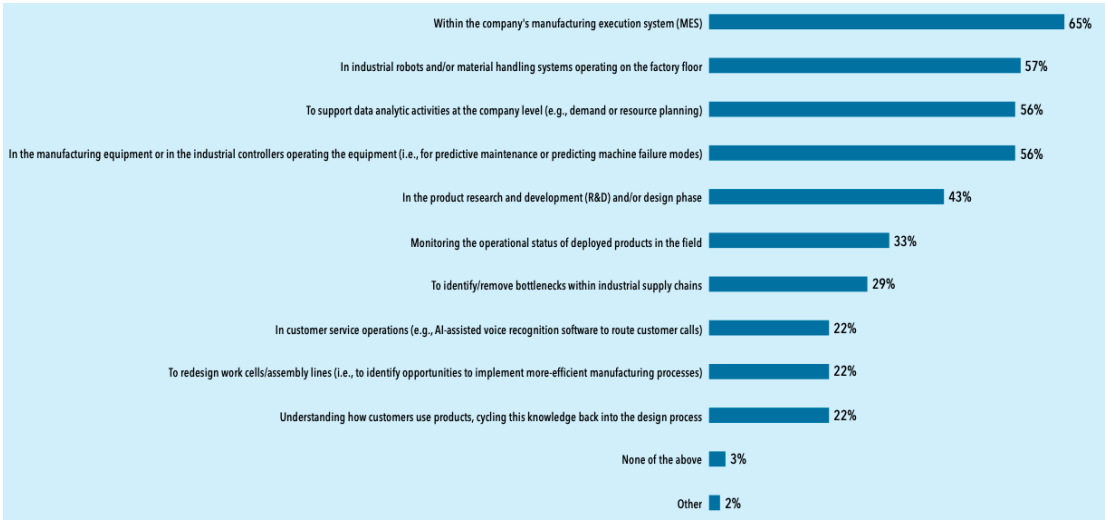
Although, some AI/Industry 4.0 technologies have been somehow adopted by companies, amongst others we find Industrial robotics, Machine vision, Intelligent products, Machine learning and Cobots. According to the MAPI report, predictive systems are growing to 90% penetration while the use of AI to manage intelligent supply chains, is growing to almost three-fourths penetration, in part with the use of AI in machine learning, machine vision, and intelligent robotics [18].

As the deployment of AI has spread, based on the MAPI report we can identify in figure 1.2a that machine vision is currently adopted in around 46% of industries and machine learning in 26% with a huge potential of both technologies in the near future (5 years). Examining the different operational domains we identify that applications at a floor level are expected with 57% of chances as seen in figure 1.2b.

The impact of the digital transformation in manufacturing process will change the way industries currently perform their activities, however there are barriers that difficult the adoption of these technologies not only in big companies but also in small and median enterprises.



(a) AI deployment levels vary for different applications



(b) AI applications expected to deploy across operational domains

Fig. 1.2 Current and expected deployment of AI. Source: [18]

These barriers are not only limited to know-how, qualified human talent or technological lacking but also for strategical decisions in companies and the uncertainty caused by the lack of awareness about AI capabilities and impact of its implementation. Although, executive decisions are based on profit and currently they are based on the limited knowledge the executive level has on AI, other technical limitations are present as shown in figure 1.3, particularly in this project we want to deal with data availability.

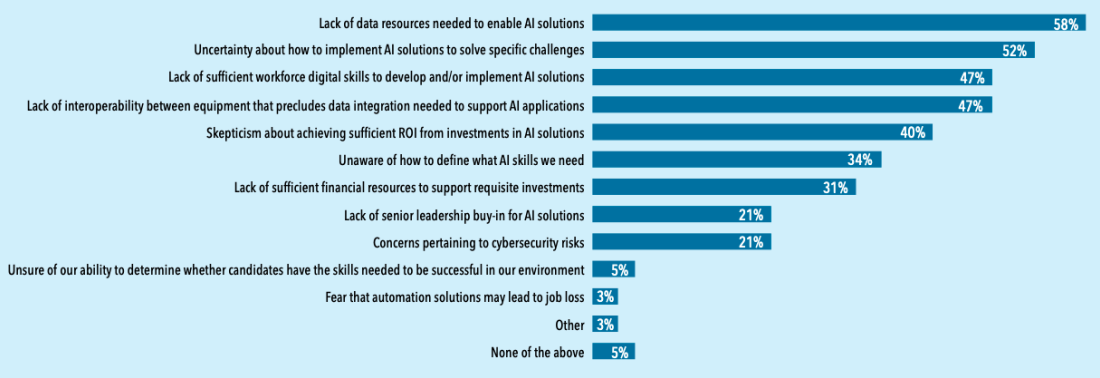


Fig. 1.3 Top Barriers to AI Adoption. Source: [18]

Figure 1.3 also shows that the lack of workforce with digital skills is also an important problem around the world. In fact governments around the world have taken actions to train people in these skills as part of big government development plans for digital transformation towards industry 4.0. In Colombia, ministry of technology and communication has launched a digital transformation policy to make companies more competitive and productive using industry 4.0 technologies [24].

These policies are mainly orientated to the Colombian industry. However, in this country most of industries are SME's [25], which cannot access high-tech solutions to be adopted in their processes, thus hindering their development in short term and impacting directly in their competitiveness in long term.

Currently, governments including the government of Colombia are taking the first steps in order to regulate the development and implementation of AI [26]. The first approach that

companies should take is to create a post of ethical responsibility on AI and design tests to validate that the models created are unbiased. However, these first steps are more orientated to data science and analytics, specifically to data privacy and safety.

On the other hand, the adoption of digital technologies in Colombia depends on two factors; the size of the enterprise and the maturity of the technology to be implemented. For instance, cloud computing is widely being implemented and is expected that near 90% of enterprises will be using cloud services for the year 2022 [27].

According to IBM Latam, it is estimated that nearly 82% of enterprises are considering to adopt an AI solution into their processes, however the lack of knowledge or trained people hinders the real implementation of the solution [27].

In Colombia AI implementations are almost exclusively for big companies and usually for commercial and support services, for instance chat-bots for contact centers, Robot process automation algorithms and search engines [27]. Nevertheless, approaches of machine learning techniques exists particularly in agriculture and crops management powered by IBM's Watson.

Microsoft states that for Colombia the adoption of AI could eventually triplicate the country productivity, thus representing an increase of 6.8% on the country GDP. This estimate considers a maximum adoption of AI in the next ten years, creating a demand of high qualified employees. According to their research new posts will be created in these areas: Government and state 1.1 million, commerce and tourism 800.000 posts, building trade 400.000 and manufacturing 300.000 [28].

According to the report developed by the Chamber of Commerce of Bogotá in 2017, only 0.8% of the manufacturing enterprises have adopted AI in their processes, and the sector with the most adoption of AI was financial services with 1.4%. The most adopted technologies in manufacturing are cloud computing and cybersecurity which are mature technologies. However, in "advanced" technologies the most representatives are IoT with 10.4% and 3D printing with

3.0%. This report also concludes that only 6% of SME's have implemented any kind of a digital technology [29].

The first steps in the adoption of AI technology in manufacturing processes in Colombia are orientated to food sector that implements it for quality check purposes, specifically for visual inspection of agricultural products using optical sensors [30].

Even though the adoption of machine learning techniques for manufacturing industries is trending worldwide, and is expected to keep growing within the next five years, references show that globally there are barriers that still limit its adoption by companies.

By developing digital skills for digital transformation and AI implementation in manufacturing this project would like to answer the following question:

Is it possible to create an algorithm based on machine learning techniques for surface visual inspection in a manufacturing process characterised by data shortage and class imbalance?

1.3 Consistency With Master Syllabus

Since most of the supervised machine learning techniques have the goal of minimizing a fitness or cost function based on a error metric, they use optimization methods to find optimum or near-optimum solutions, thus reducing errors and creating a good model. One can say that most of the machine learning algorithms can be used as part of an operations research application, since these advanced techniques aid in the process of intelligent and informed decision making, which is the key thematic of the Master in Industrial engineering at *Pontificia Universidad Javeriana*.

Besides, genetic algorithms, evolutive algorithms, mimetic algorithms, particles swarm and others bio-inspired methodologies for decision making problems are part of the artificial intelligence paradigm. However, the techniques that are capable to "learn" by themselves, are enclosed

by the machine learning paradigm. Hence there one can say that there is an intrinsic relationship between machine learning and operations research.

When both, operations research and machine learning are data-driven with the purpose of decision making, they can be seen as part of the wider domain of data science. Operations Research is the study of how to make decisions efficiently, while machine learning makes predictions and inferences [8], they complement each other.

On the other hand, visual inspection is important in industrial processes, since is part of quality specification fulfillment, some courses of the Industrial Engineering master are about quality assurance as six sigma and lean manufacturing.

1.4 About The Case Study

The case of study is focused in visual surface inspection using machine learning techniques, as stated above, different approaches have been used to solve this task. However, in Colombia research in the field of ML integrated with computer vision has not been widely adopted yet, thus presenting an opportunity to import or create knowledge about the feasibility to develop and eventually implement these techniques in Colombia manufacturing processes.

The purpose is to focus in a quality check station of a material or product that require visual inspection, particularly to identify surface defects, and even classify them according to the specifications of the process. Since the retrieval of this data represents a huge challenge and is time consuming, the case study consider a labeled-data shortage and also the utilization of an available data set.

Due to the unexpected events caused by the COVID-19 pandemic it was necessary to manage a way to recreate an industrial process using available data. Since it was not possible to in-site capture raw data for the project, nor feasible for the lack of time, a search for available images

related to an industrial process that was meant to be used for a visual inspection system was conducted.

In order to search for an available data set the following items where considered to select the data for the case study:

- To be a general industrial process which its study will be valid and valuable for Colombian industry.
- The predominant data type should be unstructured data in form of images, whether they could be RGB or gray-scale of any resolution.
- Class imbalance and data shortage should be present in the data set, however the data set could be changed to emulate this behaviour.
- The data set should pose a challenge difficult enough to require the utilization of machine learning techniques.
- A labeled data set is desirable.

A data set published by [31] fitted all the previous conditions and was chosen as study case. The data consists of multiple images of different defects present on steel sheets as shown in figure 1.4. Given that steel is an important material used for different purposes particularly in construction and, since this material is processed in Colombia this study case will serve as an approach to emulate a real process in a Colombian industry and, to propose a machine learning solution to the automatic visual inspection.

The images where originally taken by [31] which is a Russian steel manufacturing company. They recently created a large industrial data lake, with petabytes of data that were previously discarded and it is now looking to machine learning to improve automation, increase efficiency, and maintain high quality in their production [32].

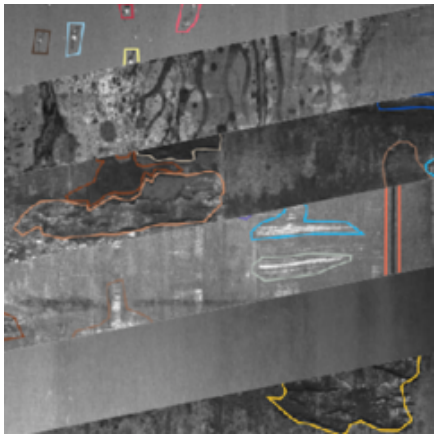


Fig. 1.4 Surface defects on steel for visual inspection. Source: [32]

Although the information comes from an international company, the process of steel manufacturing could be extrapolated to local companies due to metals universal physical properties and standardised conformation processes, so the solution proposed could be applied or adapted for local producers. Particularly, this process consist in the production of flat sheets of steel which includes some classic metallurgic techniques such as heating, rolling, drying and cutting.

The data set, which is available at [33], consist of $256X1600$ gray-scale images from high frequency cameras that aims to localize and classify surface defects on a steel sheet, the data set includes also its corresponding ground truth for training and testing. Further information about the case study and its characterization is described in section 3.1.1.

Chapter 2

Research Proposal & Methodology

2.1 Research Objective

To develop a surface visual inspection algorithm based on machine learning techniques, dealing with labeled-data shortage and class imbalance for an industrial process.

2.1.1 Specific Objectives

1. To explore and discover the latest and recommended techniques based on machine learning considering labeled-data shortage and class imbalance for visual inspection, in order to identify the ones which could best fit for the algorithm development.

2. To propose an algorithm based on machine learning techniques and computer vision to develop a visual inspection task under the conditions of labeled-data shortage and class imbalance.

3. To Propose and describe a case study that will be recreated in a laboratory, to be used as a test-bed to validate and test the proposed algorithm.

4. To emulate and evaluate, under controlled conditions, the ability of the visual inspection algorithm to identify and locate superficial defects.

2.2 Methods & Methodology

Machine Learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on providing confidence intervals around these functions [34].

In order to methodologically develop the project, the following guidelines stated by [6] were used since they are the generally accepted approach to develop a machine learning application:

1. To look at the available data and how it is described (labeled, unlabeled, expert knowledge availability, etc.) then one chooses between a supervised, unsupervised, reinforcement, or hybrid learning approach.

2. Analyze the general applicability of available algorithms with regard to the research problem requirements (e.g. able to handle high dimensionality). A specific focus has to be laid on the structure, the data types, and overall amount of the available data, which can be used for training and evaluation.

3. Previous applications of the algorithms on similar problems are to be investigated in order to identify a suitable algorithm. The term "similar" in this case means, research problems with comparable requirements e.g. in other disciplines or domains.

These three steps are further developed in section 3.1 which presents the project development.

2.2.1 Visual Inspection Task

Visual inspection is a common method of quality control, although images can be acquired from different portions of the electromagnetic spectrum (X-ray, infrared, near infrared etc.), usually the term is used only for the tasks achieved using only the visible spectrum.

Automatic Visual Inspection

The visual inspection task, usually performed by a human operator, poses a challenge for the process reliability and scaleability. This task, which usually is done for quality assurance purposes in an industrial process, is based on standards that commonly are learnt by the operator based on examples, with clear definitions of acceptable deviations. However, human operators are not able to compete in speed, repetitiveness, reliability and robustness against a machine.

On the other hand, humans are very capable at distinguishing between subtle cosmetic and functional flaws, as well appreciating variations in part appearance that may affect perceived quality, this includes the natural ability of humans to generalize and extrapolate knowledge [35]. When learning by example humans are capable of distinguishing what really matters when it comes to slight anomalies. This makes human vision the best choice, in many cases, for the qualitative interpretation of a complex, unstructured defects and unpredictable flaws.

Humans are much more accurate when dealing with deformed and hard-to-read characters, complex surfaces, and cosmetic defects. For many of these applications, machines cannot compete with humans for their appreciation of complexity [35].

There are two types of errors in visual inspection for manufacturing; missing an existing defect or incorrectly identifying a defect that does not exist (false positive). Misses can lead to loss in quality, while false positives can cause unnecessary production costs and overall wastage [36].

In order to properly set up and environment for automatic visual inspection one should consider these key stages [36]:

1. Scene Preparation
2. Image Acquisition
3. Digitalization
4. Image Enhancement
5. Feature Extraction
6. Computer vision Task
7. Final Decision
8. Results

Due to time and facilities limitations this project will only consider from feature extraction stage onwards.

Machine Vision

Machine vision is the technology and methods used to provide imaging-based automatic inspection and analysis for applications as automatic inspection, process control, and robot guidance, usually in industry [37].

Machine vision systems can inspect hundreds or thousands of elements per minute reliably and repeatedly. These systems perform reliably with consistent data. Usually, are designed via step-by-step filtering and rule-based algorithms that are more cost-effective than human inspection.

Certain traditional machine vision inspections, are difficult to program due to multiple variables that can be hard for a machine to isolate such as lighting, changes in color, curvature, and field of view [35].

Machine Vision Improvements Based on Deep Learning

New algorithms implemented in machine vision that include the utilization of advanced machine learning techniques, known as deep learning, are able to tolerate some variability in appearance due to scale, rotation, pose distortion, complex surface textures and image quality issues.

Deep learning models can help machines overcome their inherent limitations by leveraging from the self-learning of a human inspector and the speed and consistency of a computerized system. Deep learning-based image analysis is especially well-suited for cosmetic surface inspections that are complex in nature: patterns that vary in subtle but tolerable ways. This technique excels at addressing complex surface and cosmetic defects, like scratches and dents. Whether used to locate, read, inspect, or classify features of interest, deep learning-based image

analysis differs from traditional machine vision in its ability to conceptualize and generalize features [35].

2.2.2 Computer Vision Main Tasks

In order to develop a machine learning application is necessary to first recognize the task type that is going to be abstracted from the real world into a mathematical model and, furthermore an algorithm. In the computer vision world, one can identify different applications, among others there are:

- Face recognition
- Surveillance
- Biometrics
- Smart cars (Autonomous Driving)
- Image classification
- Localization
- Detection
- Image Captioning

These applications consist in abstracting real world events into a digital interpretation of the environment, that at the end is transformed in information for further analysis or decision making. These applications related to extracting information from the environment can be summarized in the following techniques [38]:

For clearer understanding a sample image is given in figure 2.1.

1. **Image Classification:** Given a set of images that are all labeled with a single category, the algorithm must predict these categories for a novel set of images. Figure 2.1a.

2. **Object Detection:** To define objects within images and outputting bounding boxes and labels for individual objects. although this task originally consist in two classes (object of in-

terest and background), it could be generalized to multiple classes to a location/classification task. Figure 2.1b.

3. **Object Tracking:** To follow a specific object of interest, or multiple objects, in a given scene. Figure 2.1c.

4. **Semantic Segmentation:** To semantically understand the role of each pixel in the image, besides from recognizing it also delineates the boundaries of each object. Unlike classification, dense pixel-wise predictions are required. Figure 2.1d.

5. **Instance Segmentation:** To segment different instances of classes, such as labelling objects from the same class with different identifiers this means to not only classify these different objects, but also identify their boundaries, differences, and relations to one another. Figure 2.1e.

Once, the main techniques of computer vision for decisions based on objects and scenes were identified, and taking into account the main purpose of the case study, which consist in a visual inspection task, the next step is to select the appropriate machine learning technique.

Traditional Computer Vision Versus Machine Learning-based Computer Vision

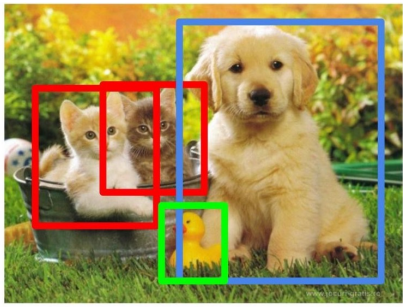
Traditionally, the ability to create computer vision (CV) solutions belonged only to those who specialized in this subfield of computer science. The approaches and techniques used were only useful for images processing and hardly could be extrapolated to other areas of computer science. This rule-based approach were successful in applications with total control of the environment and scene. It was costly in time and effort to adapt the solution to a similar task.

Images, by nature, are a source of variation (contrast, color, brightness), which in a controlled scenario could be managed and meaningful features could be extracted using a rule-based approach.

CAT

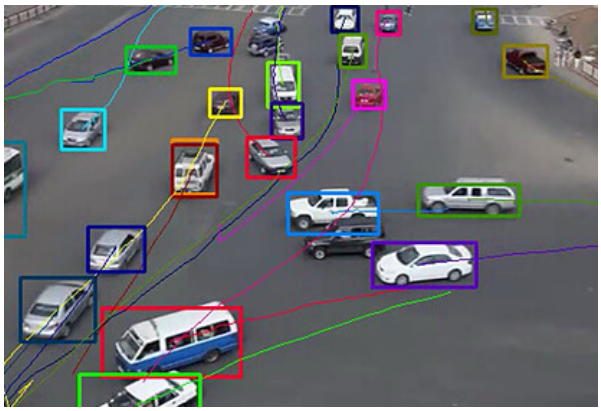
(a) Image classification. Source:[39]

CAT

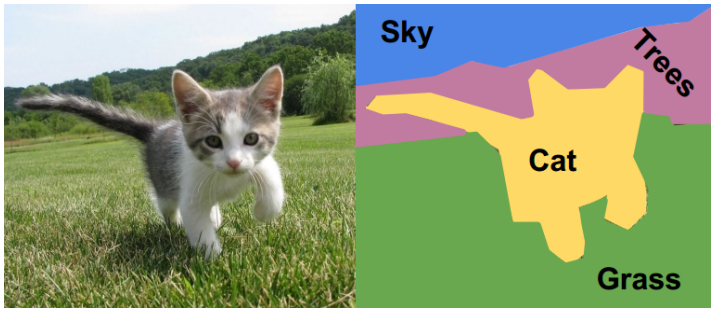


CAT, DOG, DUCK

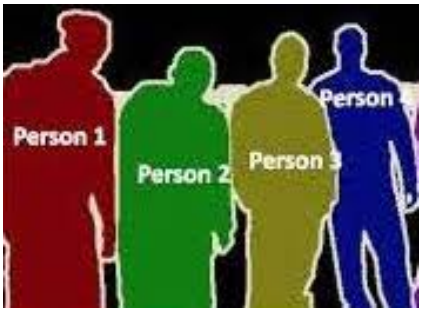
(b) Detection - Location/Classification. Source:[39]



(c) Object Tracking. Source: [40]



(d) Semantic Segmentation. Source:[41]



(e) Instance Segmentation. Source:[42]

Fig. 2.1 Computer vision for making decisions about real objects and scenes.

However, in an uncontrolled scene the challenge is directly related to the difficulty of extracting meaningful features while dealing with these variations.

Computer vision research has focused on manually defined pipelines for extracting good image features. Image feature extractors such as SIFT and HOG were the standard. (See section 2.2.3). Recent developments in deep learning (a family of techniques based on neural networks), have extended the reach of traditional machine learning models by incorporating automatic feature extraction in the base layers [43].

Industry have seized of computer vision in production lines with high process standardization, since they are usually very controlled work stations, with low variability in the environment conditions. Some applications of traditional computer vision for visual inspection in industry are:

- Gauging and measurement
- Presence/absence
- Barcode reading and identification
- Robotic guidance

Although for years CV has proven to be a reliable technology, there are other tasks that pose a true challenge. Some visual inspection tasks have complex features that hard-rule approaches are too rigid to be applied in a dynamic scene. Furthermore, some features are possible to extract by naked eye, but very hard to codify. Some of these complex tasks are:

- Complex cosmetic inspection
- Deformed and variable feature location
- Texture and material classification
- Challenging optical character recognition
- Assembly verification
- Anomaly/defect segmentation

In order to develop solutions for these complex tasks, methods based on artificial intelligence, precisely machine learning, are the to-go solutions in recent years. These models require semantically meaningful features to make semantically meaningful predictions [43].

Deep Learning in Machine Vision

Deep learning complements rule-based approaches, and it reduces the need for deep vision domain expertise. It has turned applications that previously required vision expertise into engineering challenges solvable by non-vision experts, by transferring the logical burden from an application developer, to an engineer training the system. It also opens a new range of possibilities to solve applications that have never been attempted without a human inspector [35].

2.2.3 Feature Extraction

As previously said feature extraction is key when creating a visual inspection application, although sometimes features are evident and easy identifiable (edges, lines, color, corners,brightness, shape) other are not that clear to identify or even codify. See figure 2.2's "rule-based systems" which usually rely on these primitive features.

In order to abstract more information about the scene, it is necessary to, one way or another, extract features manually or automatically.

For instance, for the object recognition task, differences between neighboring pixels are often very useful. Pixel values usually differ at the boundary of objects, when there is a shadow or on a textured surface. The difference in value between neighboring pixels is called an image gradient [43].

So one technique that leverages of this, is the Histogram of Oriented Gradients (HOG) [44]. This manual approach has been for long time key in computer vision. In figure 2.2 this method is shown as "Classic Machine Learning".

As level of abstraction increased new methods have to be created. For example, another gradient-based technique appeared; SIFT. It was originally developed for the task of object recognition. The process involves analyzing the image at a pyramid of possible scales, detecting

interest points that could indicate the presence of the object, extracting features, also called image descriptors, about the interest points, and determining the pose of the object [43].

This approach essentially replace manually defined features with manually defined models that automatically learn and extract features. The manual work is still there, just abstracted. In figure 2.2 this method is shown at the left column of "Representation Learning".

However, recent developments in neural networks and deep learning approaches, have greatly advanced the performance of state-of-the-art visual recognition systems. These trending methods are, since 2010, the focus of research. The increased and cheap computation power has opened a whole new world of possibilities in order to create powerful, complex and deep models based on "automatic learning" which allows a higher level of abstraction of underlying features. In figure 2.2 this method is described as "Deep Learning" and allows the system to learn the features by itself.

In figure 2.2 grey boxes represent level of abstractions learnt automatically by the model itself, based on an expected output.

2.2.4 Deep Learning

Deep learning is a division within machine learning that mainly consist in different methods an techniques that are applied to deep neural networks. Figure 2.3 shows that deep learning is a kind of representing learning, due to the fact that the system by itself is able to extract the features used to best represent the scene which means, to extract the most suitable information from the image an based on it make predictions.

Deep feedforward networks, also called feedforward neural networks, or multi-layer perceptrons (MLPs), are the quintessential deep learning models. These models are called feedforward because information flows through the function being evaluated from, through the intermediate

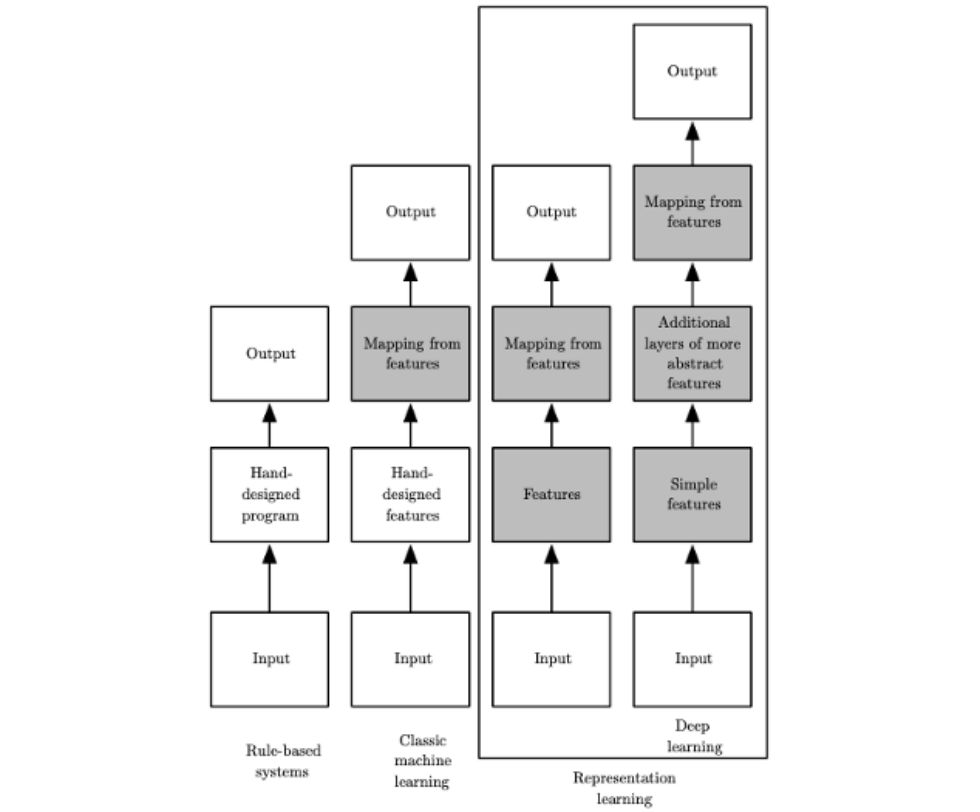


Fig. 2.2 Data learning process in different AI techniques. Source: [34]

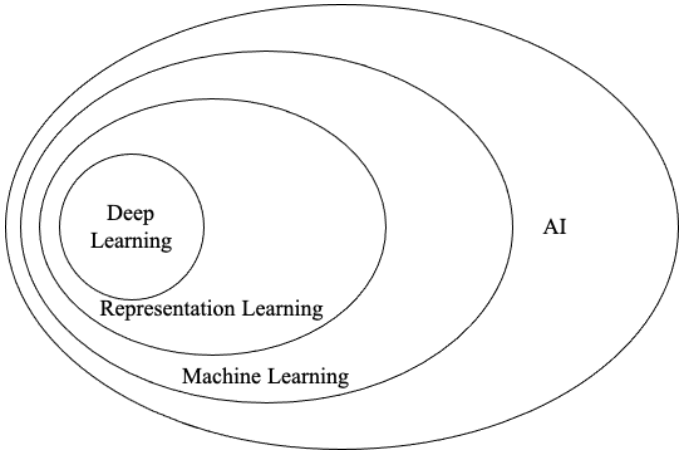


Fig. 2.3 Venn diagram of AI. Based on [34]

computations used to define the function, and finally to the output:

$$\mathbf{X} \rightarrow f() \rightarrow \mathbf{Y}$$

There are no feedback connections, as opposed to other architectures, as recurrent neural networks. The goal is to approximate some function f. For example, for a classifier, $y = f^*(x)$ maps an input x to a category y, by defining a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation [34].

Notice that these functions are usually presented in the form of layers in a network. The overall length of the layers gives the depth of the model. Thus, the terminology of deep learning.

As said before, deep learning includes different types of neural networks, each one used for different tasks, not only in computer vision but also in speech recognition, text generation, sequences analysis and generative models.

Computer vision has traditionally been one of the most active research areas for deep learning applications, because vision is a task that is effortless for humans and many animals, but challenging for computers [45].

Convolutional Neural Networks

Working with only image intensities (the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. This is why, an approach to efficient automatic feature extraction, was motivated. As a solution the Haar-like features were proposed.

A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window. Then, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image [46].

The idea of learning local pattern, as the Haar-like features, was further implemented with the introduction of the Convolutional Neural Networks (CNN). Oppose to dense layers which learn global patterns in their input feature space, convolution layers learn local patterns, consisting in small 2D windows of inputs.

The convolutional networks used for object recognition are just a specialized kind of feed-forward networks for processing data that has a known, grid-like topology. These networks use convolution in place of general matrix multiplication in at least one of their layers [34, 47].

Figure 2.4 shows the difference between a fully connected layer (bottom) and a convolution layer (top) which extract features and minimizes computational operations.

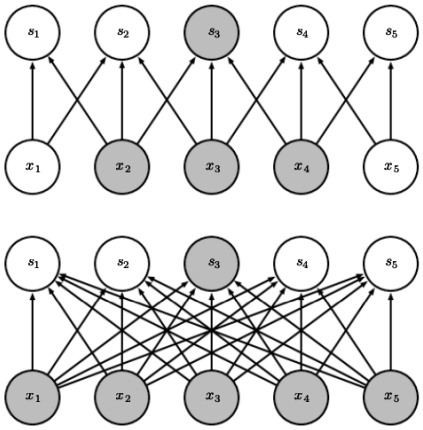


Fig. 2.4 Receptive field fully connected layer vs convolution layer. Source: [34]

Convolution leverages three important ideas that can help improve a machine learning system for problems based on images [34]:

- **Sparse Interactions**: Input images might have thousands or millions of pixels, but it is possible to detect small, meaningful features, with kernels that occupy only tens or hundreds of pixels.

- **Parameter Sharing**: In a traditional neural net, each element of the weight matrix is used exactly once when computing the output of a layer. In a CNN, each member of the kernel is

used at every position of the input. This means that rather than learning a separate set of parameters for every location, it learns only one set.

- **Equivariant representations**: This means that if the input changes, the output changes in the same way: $f(g(x)) = g(f(x))$.

One of the main advantages of CNNs is that the patterns they learn are translation invariant. After learning a certain pattern they can recognize it anywhere. This makes CNNs data efficient when processing images (because the visual world is fundamentally translation invariant) [47]. Besides this, they can learn spatial hierarchies of patterns, this means that CNNs efficiently learn increasingly complex and abstract visual concepts.

Although its advantages, convolution is not naturally equivariant to some other transformations, such as changes in the scale or rotation of an image. Because CNN learn local, translation-invariant features, they're highly data efficient on perceptual problems.

Figure 2.5 schematically presents how a CNN works and, features or image descriptors are automatic extracted from the same data.

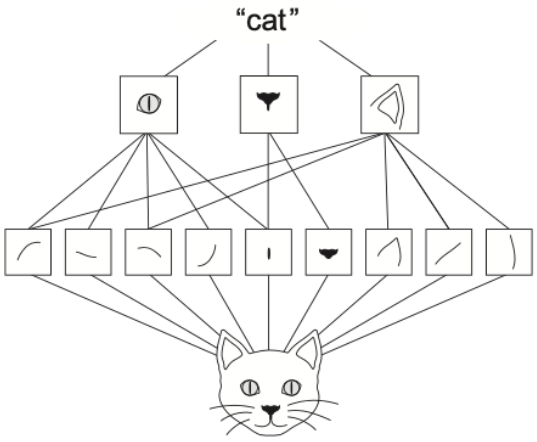


Fig. 2.5 Features detected by a CNN. Source: [47]

A typical layer of a convolutional network consists of three stages [34]:

1. The layer performs several convolutions in parallel to produce a set of linear activations.
2. Each linear activation is run through a nonlinear activation function, usually ReLu(rectified linear activation).
3. Use a pooling function to modify the output of the layer further.

A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. Often using the max function. It combines multiple inputs into a single output which means that pooling forces a local image neighborhood to produce one value instead of many. This reduces the number of outputs in the intermediate layers of the deep learning network, which effectively reduces the probability of overfitting the network to training data [47].

2.2.5 Optimization in Deep Learning

Machine learning algorithms are able to learn from data. Formally, a computer program is said to learn from experience E with respect some class task T and performance measure P if its performance at tasks in T as measured by P improves with experience E [34]. In the case of supervised learning, using a data set containing features associated to a label or target.

1. **The task**: How machine learning should process an example.
2. **Experience**: A data source with examples and their expected outputs.
3. **Performance measure**: A specific metric according to the task.

Optimization Goals & Challenges in Deep Learning

The central challenge in machine learning is that an algorithm must not only make the train error small but, also, perform well on new previously unseen inputs. This ability is known as

generalization. What separates machine learning from an optimization problem is that we want the generalization error, also called the test error, to be low as well. This is achieved by measuring its performance on a test set of examples that were collected separately from the training set [34]. Summing up, in a deep learning project one pursue two objectives:

- Make the train error small.
- Make the gap between training and test error small.

Figure 2.6 shows the behaviour of error in machine learning applications. Notice that even when the train error is constantly minimized, the test error has a "U" shape behaviour.

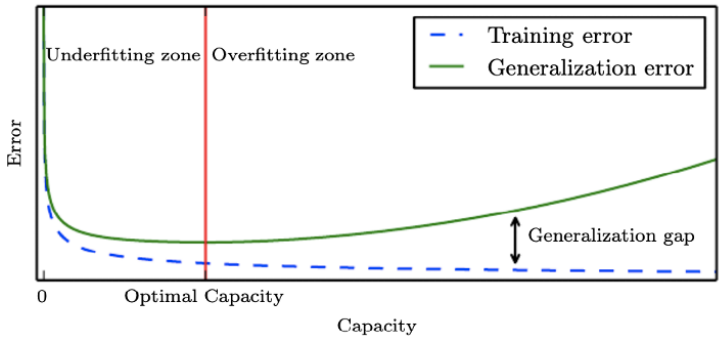


Fig. 2.6 Error behaviour in machine learning. Source: [34]

From these goals the two main challenges of deep learning appear; underfitting and overfitting. Underfitting occurs when the error value on the training set is not sufficiently low . Overfitting occurs when the gap between the training error and test error is too large.

The limited capabilities of the optimization algorithms, due to little theoretical understanding of the general non-convex optimization problem poses another challenge in order to reach the best training for a model. Since simpler functions are more likely to generalize (have a small gap between training and test error), we must still choose a sufficiently complex hypothesis to achieve low training error [34].

For the purpose of this project one can summarize a deep learning model as the result of the following recipe:

1. **A data set**: Set of samples with its target values.
2. **A cost function**: A function to be optimized during learning process.
3. **An optimization procedure**: An algorithm to reach a low value in the cost function.
4. **A model**: A neural network architecture to process inputs and generate structured outputs.

To find the parameters θ of a neural network that significantly reduce a cost function $J(\theta)$, which typically includes a performance measure, requires powerful, yet relative simple optimization strategies.

The Optimization Task

The performance measure:

In most machine learning scenarios, the performance measure P is the main goal however in some cases it is intractable or very hard to optimize using gradient-based methods. Therefore the optimization of P is carried indirectly based on a clever decision of the cost function. This means to reduce a cost function $J(\theta)$ in the hope that doing so will improve P.

This is in contrast to pure optimization, where minimizing J is a goal in and of itself [34].

The cost/loss function:

Sometimes, the loss function we actually care about (say classification error) is not one that can be optimized efficiently. In such situations, one typically optimizes a surrogate loss function instead, which acts as a proxy but has advantages [34]. Particularly, functions with tendency to reduce overfitting or hard penalization to errors. For instance, the negative log-likelihood loss usually surrogates the 0-1 loss function which causes maximum likelihood estimation.

Considerations on Optimization for Deep Learning

One of the advantages of convex optimization is that any local minimum is guaranteed to be a global minimum. With non-convex functions, such as neural networks, it is possible to have many local minima. Indeed, nearly any deep model is essentially guaranteed to have an extremely large number of local minima [34].

Optimization is difficult specially when the function is multidimensional, thus, some algorithms may fail to find a global minimum when there are multiple local minima or plateaus present. A very important difference between optimization in general and optimization as it is for training algorithms is that training algorithms do not usually halt at a local minimum. Instead, a machine learning algorithm usually minimizes a surrogate loss function but halts when a convergence criterion based on early stopping is satisfied [34]. Figure 2.7a shows this behaviour.

Optimization Algorithms

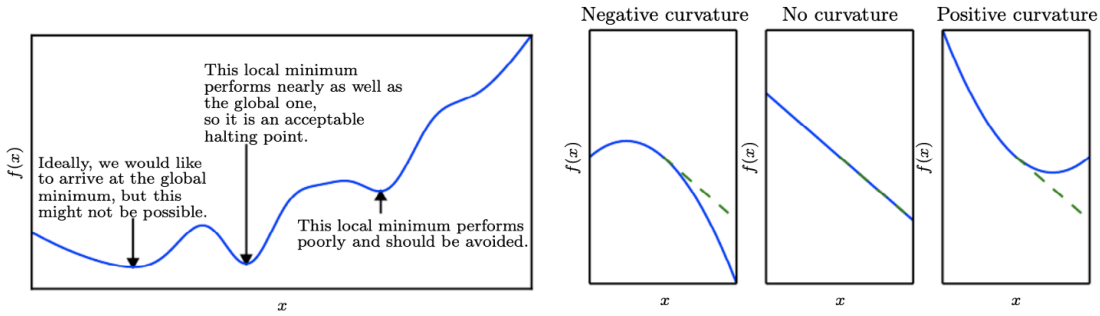
Neural networks are usually trained by using iterative, gradient-based optimizers that drive the cost function to a very low value, rather to a global minima.

Optimization algorithms that use only the gradient such as gradient descent, are called first order optimization algorithms. Optimizations algorithms that also use the Hessian matrix such as Newton's method are called second-order optimization algorithms [48].

Since gradient descent does not seize the information contained in the Hessian Matrix, it could be misleading when training, due to the expected values of the gradient would not improve the values of the function as expected.

This behaviour is shown in figure 2.7b. Notice that with negative curvature, the cost function actually decreases faster than the gradient predicts. With no curvature, the gradient predicts the

decrease correctly. With positive curvature, the function decreases more slowly than expected [34].



(a) Minimization Process (b) Second derivative in optimization

Fig. 2.7 Optimization in deep learning. Source:[34]

Stochastic gradient descent (SGD) and its variants are probably the most used optimization algorithms for machine learning in general and for deep learning in particular [34].

This *de facto* algorithm, which is the backbone of other algorithms, particularly those that are capable of using the Hessian matrix for the optimization process. Among others it is possible to find:

- AdaGrad [49]
- RMSProp [50]
- Adam [51]
- Adadelta [52]
- Adamax [51]

2.2.6 Hardware

Deep learning requires great computational performance in order to reach in short team results. Besides, in a research environment the requirement is even more demanding since different experiments must be conducted, preferable in parallel, to compare and analyse results.

At first the project was developed in a MacBook Air, 2017 with a 1.8 GHz Dual-Core Intel Core i5 processor and 8 GB 1600 MHz DDR3 of memory. Although some applications of deep learning could be developed in this machine, a problem arises when working with images. The computational time increases so much that it is not viable to continue working in this machine. This is in part due to the processor's architecture that is not optimized for parallel computing, something that is almost required when working with images. The second aspect is that memory is not enough to work with big models based on images. Although, this could be solved using batches, the main problem of computing has not alternative using this machine.

The solution used is a mix of hardware and software using Google's cloud service "Google Collab". This platform as a service allows users to develop and execute software written in Python directly from the browser. It also gives free access to GPU's which are used for deep learning parallel computing leverages from its architecture.

The software developed in Collab is executed in Google's cloud servers, so the computational burden is transferred to these processors freeing the local machine, and increasing speed.

Since cloud services use a dynamic allocation, it is not possible to know which GPU is assigned in each session. Collab offers different model as: NVIDIA K80, P100, P4, T4, and V100 [53]. However, since this is a free service for research purposes, time limits and other restrictions apply. So the capacity to perform multiple experiments, as a deep learning application require, is limited.

Besides this, it is important to mention that this is an online tool, so connectivity issues could happen. Also, it is worth to mention that this tool is in a development phase so sudden crashes occur hindering even more the multiple experimentation of different models.

Memory allocation and CPU is also dynamic and it is not possible to be defined by the user, so the real hardware used and its properties are not clear known.

It is important to mention that computation limits is the main constrain for the development of this project and limits the chances to create multiple experiments. Due to COVID-19 it was not possible to access a more powerful solution than Google Collab platform.

2.2.7 Software

The software utilized to develop this project is open source in order to use a tool with no restrictions. The language used to write the developed algorithm is Python in its version 3.6.

Besides this, available libraries designed for data wrangling and matrix manipulation where used (Pandas [54] and Numpy [55]). For images manipulation and visualization the Pillow library [56] was used.

To develop the deep learning application Keras was used. It is a deep learning API written in Python, running on top of the machine learning platform TensorFlow [57].

Tensorflow was used as backbone, it is an end-to-end, open-source machine learning platform. That efficiently executes low-level tensor operations, computes the gradient of arbitrary differentiable expressions, scales computation to many devices, and exports programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices [58].

The Segmentation Models library [59] was also used for the specific task of semantic segmentation and the implementation of the U-net architecture. See section 3.1.7

Chapter 3

Development, Results & Conclusions

3.1 Project Development

"What we want is a machine that can learn from experience". Alan Turing

Many applications require sophisticated preprocessing because the original input comes in a form that is difficult for many deep learning architectures to represent. This is why the zero step of any machine learning project is to analyze, describe and comprehend the available data.

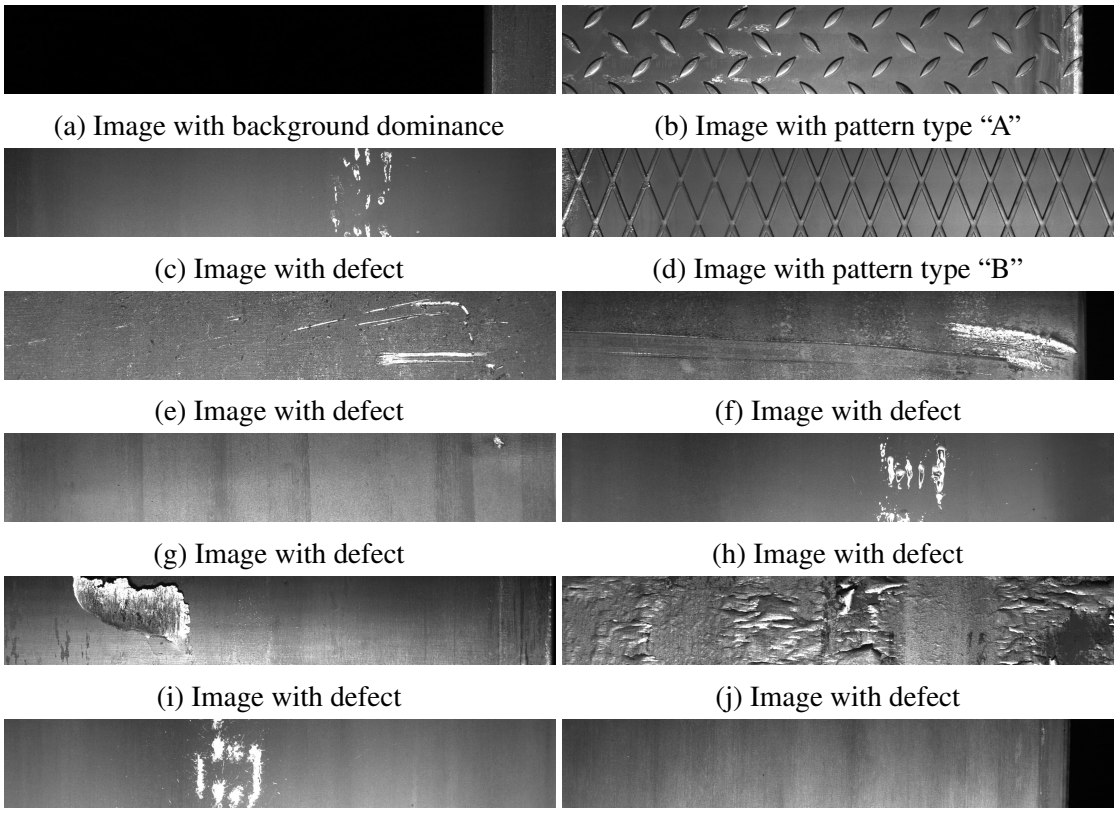
3.1.1 Data Understanding & Preparation

The dataset [33], as mentioned before in section 1.4, consists in multiple images of steel sheets with defects on their surface with the following properties:

- Each image is 256 x 1600 pixels.
- Each image has a single channel, thus images are in gray-scale.
- Each image contains one, two or none defects on the surface.

- Some images contain black zones which correspond to zones with no steel sheet.
- Some images contain patterns that no necessary are defects.
- Images are not normalized (pixel intensity is 0-255). Although, it is unknown if some transformations or adjustments were made to the images.

Figure 3.1 shows some images present in the dataset.



(a) Image with background dominance (b) Image with pattern type "A"

(c) Image with defect (d) Image with pattern type "B"

(e) Image with defect (f) Image with defect

(g) Image with defect (h) Image with defect

(i) Image with defect (j) Image with defect

(k) Image with defect (l) Image with no defect

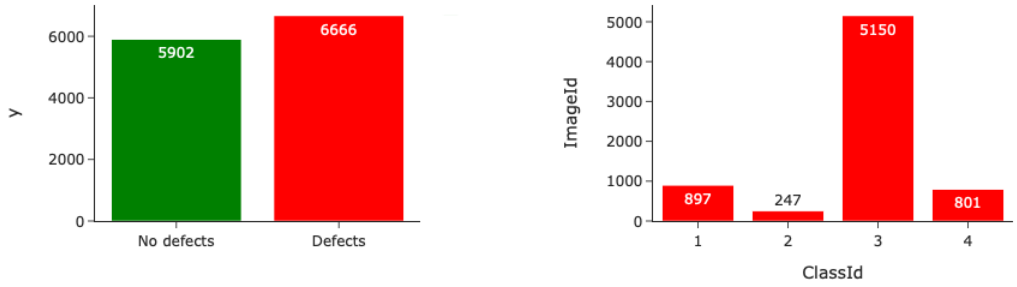
Fig. 3.1 Different kinds of surface defects in the dataset

Once the images were briefly inspected, the complete dataset were analyzed. As expected in an anomaly detection task, class imbalance is present. The whole dataset contains 12.568 images, however, only 53% of the images have defects present. Although this seems to be balanced, notice

that among the defects one must consider four types. Thus, globally, the dataset is 47% type zero and the remainder is distributed in the four classes. See figure 3.2a.

Now in the 53% of images with defects, notice that there is further class imbalance. With major class being defect number 3 with 73% of images with defects. In figure 3.2b the distribution of defect classes is summarized.

It is worth to mention that there are more defects occurrences (7.095) than images with defects (6.666), this means that in a single image can be present more than one kind of defect.



(a) Defects distribution in the dataset (b) Defect class distribution in the dataset

Fig. 3.2 Data distribution in the dataset

The dataset not only contains the images, but also contains the expected outputs or targets. Once the images were checked, the target was analyzed.

Images with no defects don't have a target since the goal is to segment defects on the surface. The images with defects have their pixels encoded in a CSV file. This structure shows the name of the image, the class of defect and pixels encoded using Run-length encoding (RLE) as seen in figure 3.3.

RLE is a form of lossless data compression in which sequences of the same data value that occur in many consecutive data elements are stored as a single data value and count. Since this

	ImageId	ClassId	EncodedPixels
0	0002cc93b.jpg	1	29102 12 29346 24 29602 24 29858 24 30114 24 3...
1	0007a71bf.jpg	3	18661 28 18863 82 19091 110 19347 110 19603 11...
2	000a4bcdd.jpg	1	37607 3 37858 8 38108 14 38359 20 38610 25 388...
3	000f6bf48.jpg	4	131973 1 132228 4 132483 6 132738 8 132993 11 ...
4	0014fce06.jpg	3	229501 11 229741 33 229981 55 230221 77 230468...

Fig. 3.3 Expected output pixel encoding

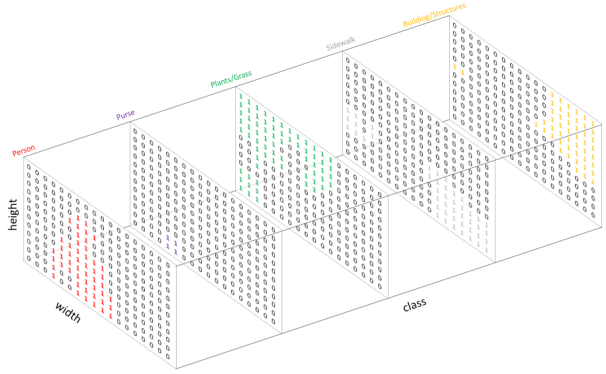
encoding is not suitable to be used in a deep learning model it is necessary to properly transform this information.

First it is necessary to decode the pixel values that are in RLE taking into account the error classes present in the image, since this is a segmentation problem five matrices were created per each image. See figure 3.4a.

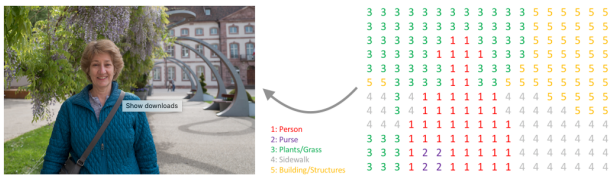
Once this data was decoded and transformed properly, using Python libraries an image mask were created by collapsing the tensor information into a single channel image. With this prepossessing the information was stored and now it is suitable for a deep learning model. Figure 3.4b shows the representation of a mask versus the input image.

For images that do not present defects, the masks created were just empty single-channel images. An algorithm was developed to decode target information and encode it in its semantic labels. In figure 3.5a the defects present on the surface are represented in the mask shown in figure 3.5b. For the development of the deep learning algorithm other image prepropressing included image resizing to 128x800 pixels and, contrast adjustment to the mask only for visualization purposes.

Since the dataset comes from an external source it is not possible to assure labels reliability. One limitation of the project is that it is supposed that all the labels are correct, although, as seen in figure 3.6 it is noticeable a surface defect (top) that has not been labeled at all, notice that the



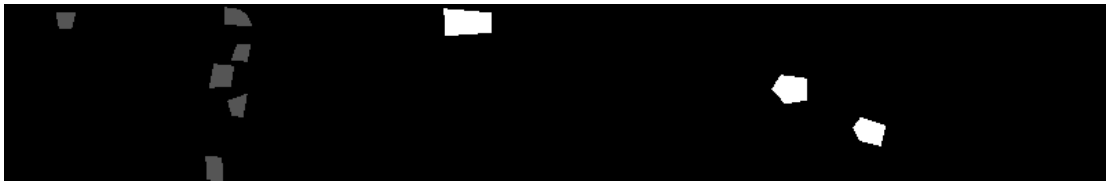
(a) Pixels decoding CSV information in a one hot encoding tensor



(b) Collapse encoding in to semantic labels

Fig. 3.4 Pixels decoding and recoding as masks. Source: [60]

(a) Grayscale image of two types of surface defects



(b) Target mask (semantic labels) created with encoded pixel data of defects.

Fig. 3.5 Original image and its semantic label target representation

mask (bottom) is completely black although it was expected to have white areas corresponding to the surface defect.

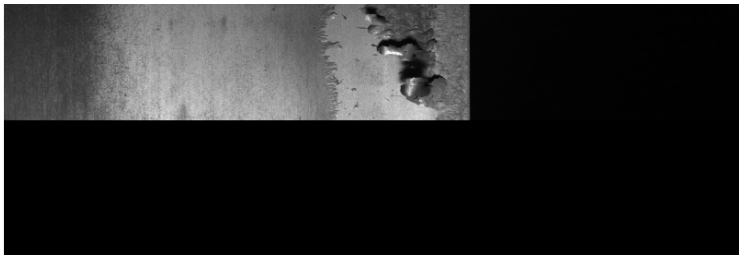


Fig. 3.6 Semantic labels errors from source

3.1.2 Data Augmentation

The best way to make a machine learning model generalize better is to train it on more data. In practice, the amount of data is limited. One way to get around this problem is to create fake data and add it to the training set [47].

Dataset augmentation may be seen as a way of preprocessing the training set only. Dataset augmentation is an excellent way to reduce the generalization error of most computer vision models [47]. Although this was considered for the project at the end it was not implemented due to:

- It has been proven useful for object detection, however since segmentation task has a different target, its limitation relays in the fact that it is necessary to apply the same transformation to the target.
- Due to the nature of the task it is not clear how the surface defects are classified, thus any augmentation that implies geometry transformations could damage the representation of the actual defect.

- Data augmentation does not necessary solves the problem of class imbalance, since data augmentation is used for reducing the chance of overfitting.

3.1.3 Metrics of Error

In order to measure the quality of the model it is necessary to define a metric of error/quality, to quantitative assess the model. It is important to mention that, as explained in section 3.1.4, these metrics are not directly optimized, instead a surrogate cost function is used.

Pixel accuracy

The to-go metric usually used in a classification task is accuracy defined by:

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Where TP, TN, FP, FN, are the primitive metrics True Positives, True Negatives, False Positives and False Negatives respectively. An extension of accuracy for segmentation models, is known as pixel accuracy it is the correct classification of a pixel within a class.

However, this metric does not behave well in presence of class imbalance. At this point it is worth to mention that class imbalance can be present within the image (class A dominates class B in the scene) and, among images (class A dominates class B in number of images). Since class imbalance is present in many real world scenarios and, in the case of defect detection even more, it is necessary to use a more robust metric.

Sørensen–Dice coefficient

This is a metric of similarity defined by:

$$DSC = \frac{2|A \cap B|}{|A| + |B|}$$

Notice that in the case of binary classification it can be expressed in terms of primitive metrics:

$$DSC = \frac{2TP}{2TP + FP + FN}$$

Since this metric evaluates the cardinality of the intersection of two sets, over the sum of their cardinalities it is more robust to class imbalance, Although this metric is intended for binary classification in recent years it has been used in computer vision to evaluate segmentation models. Notice that the Sørensen–Dice coefficient range is $[0,1]$ and it is equivalent to the well know metric F-Score when $\beta = 1$ which is the harmonic mean of precision and sensitivity:

$$F_\beta = \frac{(1+\beta^2) \cdot TP}{(1+\beta^2) \cdot TP + \beta^2 \cdot FN + FP}$$

Jaccard index or Intersection Over Union (IoU)

The Jaccard similarity coefficient is a statistic used for gauging the similarity and diversity of sample sets. It measures similarity between finite sample sets [61]. It is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Notice that the IoU metric is positively correlated with the Sørensen–Dice coefficient and often they are used interchangeable in image segmentation applications.

3.1.4 Objective Function

The optimization algorithm used for training the deep learning model uses a function that translates the inputs into a scalar in order to minimize it. The selection of this function is essential to achieve the expected goal. It is important to notice that this function is what actually is optimized, hence, is its optimization process what improves the metrics defined previously.

Usually an objective function is a distance function, that measures dissimilarity. To choose a correct objective function is necessary to consider; the model's output layer activation function, data codification, and expected output for the task.

Since a deep learning model tries to learn the data distribution of the target, the goal is that the output distribution of the model is similar as possible as the target one.

Sørensen–Dice difference function

Since the Sørensen–Dice is a measure of similarity, its complement $(1 - DSC)$ is a way to measure dissimilarity. In consideration of the optimization procedure, minimizing dissimilarity is expected to maximize similarity among data distributions. This function is given by:

$$d = 1 - \frac{2|X \cap Y|}{|X| + |Y|}$$

Notice that this function is not a formal metric since it does not fulfill the triangle inequality, thus it must be used with precaution.

Jaccard distance

Similar to the distance function described above, Jaccard distance measures dissimilarity between sample sets, it is complementary to the Jaccard coefficient and is obtained by subtracting the

Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union [61]:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Cross-entropy or log loss function

Although they are used interchargeable the origins of the log loss function and the cross-entropy function is different. The first one comes from goodness of fit of distributions, a statistical concept, while the later comes from the information theory. However, when used as loss functions, both quantify a metric of likelihood.

Cross-entropy loss function is equivalent to a maximum likelihood function under a Bernoulli or Multinoulli probability distribution. It is given by the following formula and is ubiquitously used for logistic regression models and neural networks for classification purposes:

$$H(p,q) = -\sum_{x \in \mathscr{X}} p(x) \log q(x)$$

3.1.5 Activation Function

The cost function is tightly coupled with the choice of the output unit. How to represent the output, using an activation function, determines the form of the cost function. Based on the task, the cost functions proposed and the scope of the project, the following activation functions for the output layer are considered:

Logistic function

The logistic function is a common S-shaped curve (sigmoid curve) used to introduce nonlinearity in a model or to clamp signals to within a specified interval. It is often used for classification, in neural nets is specially used when the cost function is binary cross entropy.

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1}$$

Softmax function

The softmax function, also known as softargmax or normalized exponential function is a generalization of the logistic function to multiple dimensions. It is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes. The softmax function takes as input a vector $\mathbf{z}$ of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. Each component of the output will be in the interval $(0,1)$ and the components will add up to 1, so that they can be interpreted as probabilities [62]. It is defined as:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1,\ldots,K \text{ and } \mathbf{z} = (z_1,\ldots,z_K) \in \mathbb{R}^K$$

3.1.6 Transfer Learning

One fundamental characteristic of deep learning is that it can find interesting features in the training data on its own, without any need for manual feature engineering, and this can only be achieved when lots of training examples are available. This is especially true for problems where

the input samples are very high-dimensional, like images [47]. Thus, one could say that deep learning works well when lots of data is available, although, this is partially valid.

Deep-learning models are, by nature, highly repurposable: it is possible to take, an image-classification or speech-to-text model trained on a large-scale dataset and reuse it on a significantly different problem with only minor changes. Specifically, in the case of computer vision, many pretrained models (usually trained on the ImageNet dataset [63]) can be used to bootstrap powerful vision models out of very little data [47].

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

Inductive learning

It is a popular approach in deep learning, where pretrained models are used as the starting point on computer vision and natural language processing tasks. Given the vast computational and time resources required to develop neural network models on these problems and, from the huge jumps in skill that they provide on related problems [64]. The purpose is to use what has been learned in one setting to be exploited to improve generalization in another setting.

First a base network on a base dataset is trained on a generic task, then the learned features are repurposed, or transferred, to a second target network to be trained on a target dataset and task. Figure 3.7 shows the approach of feature extraction based on a trained model.

It basically consist in using the convolutional layers of a model (encoder), as feature extractors, which are supposedly to extract key and representative information (feature maps) from the data, in order to be used as inputs for a machine learning model. The original output, (in this case a fully connected neural network), known as the top of the model, is removed since its weights are not useful for the new task.

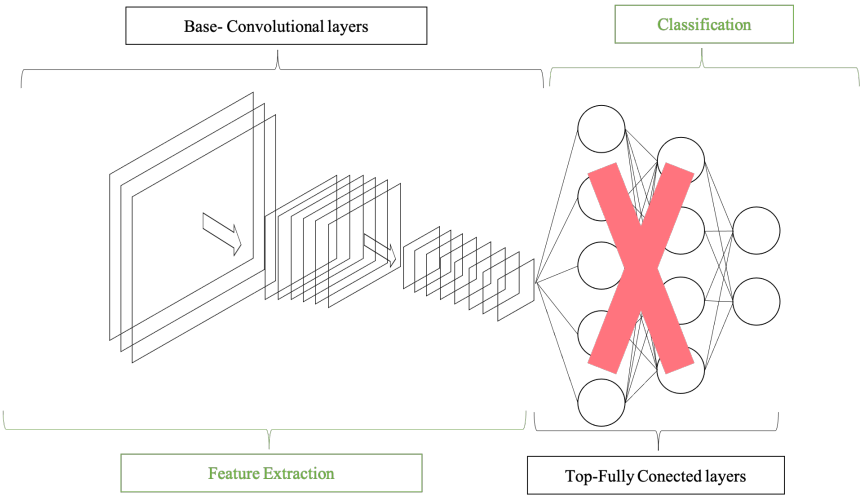


Fig. 3.7 Transfer learning in deep learning for feature extraction. *Created by author*

The features maps, which are a codification of the key information present in the data can be used to feed in any model for multiple tasks. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

It is important to take into account that the new data must be preprocessed in the same way that the original data used to train the model in the first time, otherwise the feature extraction process could be not successful as expected.

Fine tuning

Another approach to transfer learning is fine-tuning, which seeks to not only replace and retrain the classifier on top of the model on the new dataset, but to also fine-tune the weights of the pretrained network.

Fine-tuning consists of unfreezing a few of the top layers of a frozen model base used for feature extraction, and jointly training both the newly added part of the model and the top layers [47]. It is possible to fine-tune all the layers of the model, or it is possible to keep some of the earlier layers fixed and only fine-tune some higher-level portion of the network.

Notice that, as its name implies, fine-tuning is only possible when the classifier on top has already been trained. If the classifier is not already trained, then the error signal propagating through the network during training will be too large, and the representations previously learned by the layers being fine-tuned will be destroyed. As proposed by [47] the steps for fine-tuning a network are as follow:

1. Add a custom network on top of an already-trained base network.
2. Freeze the base network.
3. Train the top network.
4. Unfreeze some layers from the base.
5. Train both the base and the top.

Figure 3.7 shows the base-top discrimination in a deep learning model.

It is more useful to fine-tune the more specialized features, because these are the ones that need to be repurposed on the new problem. So usually, the layers that are closer to the top classifier are trained in the fine tuning process. Figure 3.8 shows the way base layers carry information and shows that earlier layers are prone to be reutilized while the later layers should be retrained.

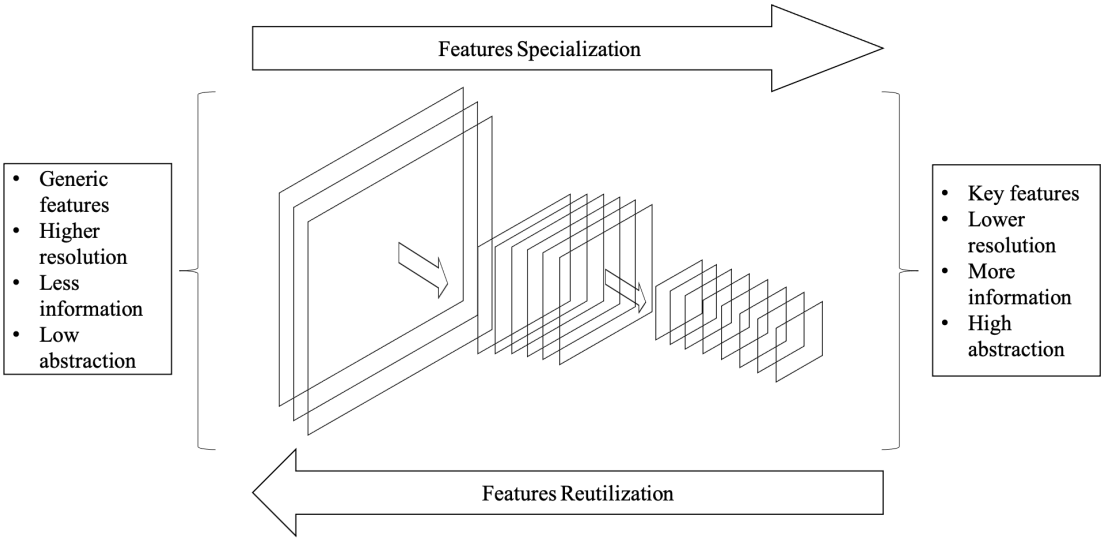


Fig. 3.8 Properties of feature encoding. *Created by author*

3.1.7 The U-net Architecture

The task (semantic segmentation) proposed in this project can be defined as a structured output. The output is a data structure containing multiple values with important relationships among different elements, in this case, the pixel-wise segmentation of images.

These kind of outputs require a different architecture, since the output must represent the relationship among pixels. Opposite to figure 3.7, which has a fully connected neural net on top, the top required for semantic segmentation must decode the input in a way that it keeps image integrity and representativeness.

The u-net is a convolutional network architecture for fast and precise segmentation of images, designed and applied in 2015 to process biomedical images. As a general convolutional neural network focuses its task on image classification, where input is an image and output is one label. However, in biomedical cases, it requires not only to distinguish whether there is a disease, but also to localise the area of abnormality. The reason it is able to localise and distinguish borders is by doing classification on every pixel, so the input and output share the same size [65, 66].

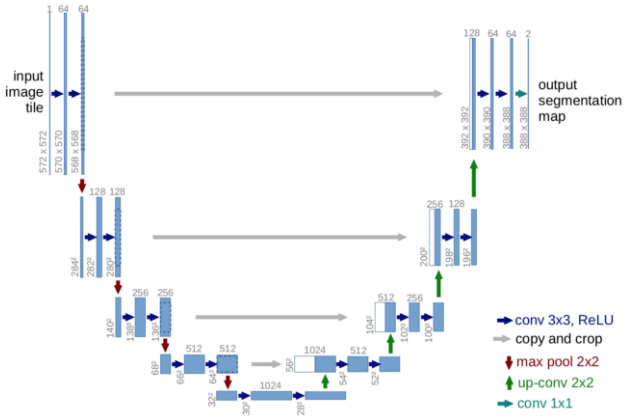


Fig. 3.9 The U-net architecture as proposed by [65]

Notice in figure 3.9 the “U” shape of the neural net due to the modification of the top layers. In this model the feature extractor or base is know as encoder, while the top layers are known as

decoders. The input and the output data preserve its structure as required for a structured output. The main difference is how the information is "translated" from the input to the output. The number of channels, originally three for a RGB image, or one for a grayscale image, are converted in multiple masks one for each class present in the model.

One of the most important characteristic of this model is the concatenate operation represented in figure 3.9 by the gray arrows. This allows to retrieve information lost in the compression process of the encoder, bypassing information from the original image directly to the decoder as a way to improve the pixel-wise segmentation in the deconvolution phase.

3.1.8 The Xception Architecture (Depthwise Separable Convolutions)

A depthwise separable convolution, commonly called "separable convolution" in deep learning frameworks consists in a depthwise convolution, this means a spatial convolution performed independently over each channel of an input, followed by a 1x1 convolution, projecting the channels output by the depthwise convolution onto a new channel space.

The Xception convolutional neural network architecture is based entirely on depthwise separable convolution layers. It makes the following hypothesis: "the mapping of cross-channels correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled". Because this hypothesis is a stronger version of the hypothesis underlying the Inception architecture [67], the name of the architecture is Xception, which stands for "Extreme Inception" [68].

The idea of using deepwise separable convolution is based on the inception architecture and the "net in net" concept represented by including a 1x1 convolution that operates as a fully connected layer among convolutional layers, this allows a more efficient use of model parameters.

3.1.9 Class Balance

For the multiple experiments conducted during the development of the project different approaches were performed to deal with data imbalance. As discussed before in section 3.1.2, data augmentation was not tested due to computation limitations, output transformation and to not have been largely used for semantic segmentation problems. Some other approaches like SMOTE or GANs for synthetic data creation were not used.

As seen in figure 3.2 error classes are highly imbalanced, the main method used to deal with this condition was the downsampling approach using as reference 900 images per class when possible. Since error classes 1, 2, 4 has fewer images, 897, 247, 801 respectively, the balance was compensated with a weighted version of the cost function. In section 3.2 the implications of this approach are explained.

3.1.10 Train, Test and Validation Split

In order to perform experiments it is necessary to split the data, the approach used was holdout method consisting in three splits; train, validation and test datasets. Cross-validation and other iterative methods were not considered due to computing limitations.

The dataset was split in: 70% for tranining purposes, 20% for validation during training and, 10% for final evaluation known as test dataset.

Roughly these values were approximated to: 12.568 images for training, 2.000 images for validation and 1.000 images for final test.

For the balanced version of the test it was done with the following splits; 3.628 images for training, 750 for validation and 375 for final test.

In order to fulfill the GPU limitations the images were loaded to the model using badges with size 32. This was done using python generators, that loaded images to memory in batches to bear

with memory limits. In some cases the limitations explained in section 2.2.6 caused exceptions due to memory overflow and many experiments had to be abandoned.

3.2 Results

"All models are wrong but some are useful". George Box

In order to find an acceptable algorithm for the segmentation task it was necessary to execute multiple experiments, however due to computation, memory and time limitations some assumptions were made:

- Unless explicitly mentioned, the optimization algorithm used for all models is Adam [51]. It was chosen because its speed, numeric stability, inclusion of second order information (momentum), its already implemented and, is *de facto* for deep learning when transfer learning.
- Unless explicitly mentioned, all the parameters of the optimization algorithm including its learning rate were not changed the default settings were used.
- Unless explicitly mentioned, the backbone used for the U-net architecture is the "Resnet-34" [69] loaded with weights trained on the ImageNet dataset. This backbone was chosen due to its speed and performance shown in papers, besides it is already implemented in the segmentation models library.
- Unless explicitly mentioned, the base (encoder or backbone) of the network was frozen in training in order to keep the feature maps learned by in the ImageNet dataset. Only the decoder or top classifier was trained.
- Unless explicitly mentioned, the models were finally evaluated using the Sørensen-Dice coefficient over the test set.
- Unless explicitly mentioned, the output mask is constructed collapsing all mask (one for each class) into a single image using the argmax fucntion.

- The number of epochs was set experimentally according the convergence of the training loss and premature stopping was used to avoid overfitting , although it was limited to 50 epochs max due to computation limitations.
- Contrast correction was performed to masks at postprocessing for visualization purposes, otherwise images will be totally black to the naked eye.

3.2.1 Testing Cost Functions

In order to train a deep learning model, once the architecture was defined (U-net) and the backbone (Renet-34) was loaded, it is necessary to set the goal of the optimization algorithm (Adam). This was done by training the full train set (12.568 images) on different cost functions and evaluate its behavior. In this phase all models were executed ten epochs due to GPU usage limitations.

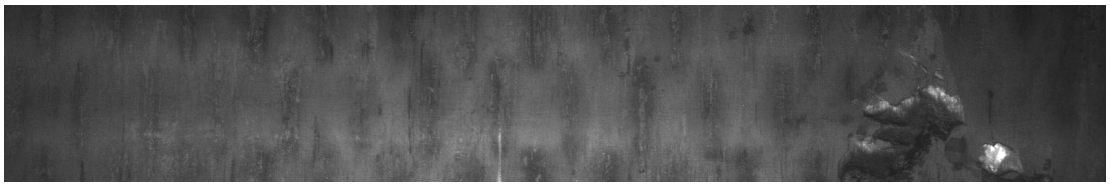
The considered losses were mentioned previously in section 3.1.4, different results were obtained according to each function.

It is important to mention that the way data is codified impacts directly in the selection of the cost function. The semantic labels described in section 3.1.1 collapse different masks in a single channel image, this implies that applying a cost function that behaves properly in a binary codification will not work well in multiple-category codification.

To face this challenge two approaches were carried out. The first one was to change directly the cost function to the sparse-categorical cross entropy loss function, which is a particular implementation in Keras of the categorical cross-entropy loss presented in section 3.1.4, this implies the modification of the activation function of the output layer as explained in section 3.2.2.

The second approach consist in changing the codification of the semantic labels and reverse the collapse of masks by recodifing the data into a one-hot encoding. With this recodification the Jaccard and dice loss were tested.

Figure 3.10 shows the results of the model with the Jaccard loss. Notice that figure 3.10c shows a grayscale behaviour when a binary behaviour is expected. This is due to the fact that the Jaccard loss is a metric of similitude, and this implies that different masks (one for each class) are going to be adjusted to be as similar as possible to the target mask. So the Jaccard loss segments well the image but is not able to classify among masks the output. Since the output is constructed using the argmax function over all masks the final image seen in figure 3.10c is the collapse of all masks with similar regions.



(a) Real image

(b) Expected output

(c) Model output

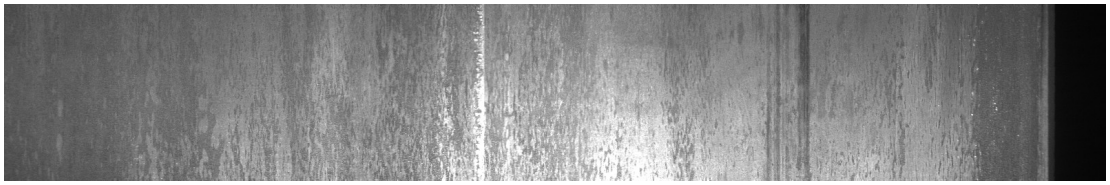
Fig. 3.10 Training with Jaccard loss

At this point the Dice coefficient is roughly close to 0.18 which is not a good performance. In order to use the Jaccard loss is necessary to change its implementation in order to distinguish each mask as class since currently it is only creating similar segments for all classes.

The other loss function considered was the Dice loss, since it operates at a pixel level the performance of the metric is quite exigent for a model. Although similar to the Jaccard loss, the

Dice loss actually outputs a binary mask which is what the structured output is expected to be. However, the classification problem still persist.

Figure 3.11 shows the result using the Dice coefficient. Notice that the final mask constructed using the argmax function is binary. However the delimitation of the regions is different from what expected. Since this is a pixel level operation notice in figure 3.11c that the regions are soft-defined opposite to figure 3.11b which is the ground truth and shows well defined regions.



(a) Real image

(b) Expected output

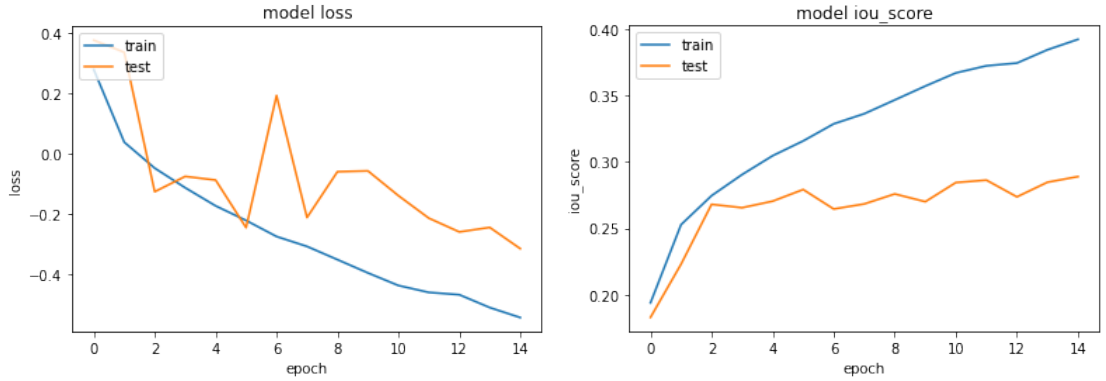
(c) Model output

Fig. 3.11 Training with Sørensen-Dice loss output

Since Dice coefficient is an exigent metric, the definition of the regions, in this case, are underfitted and will require more training with different approaches like data augmentation, parameters reduction, weighted costs or any other strategy that allows the model to extract better features for the model and improve the quality of the output.

At this point the Dice coefficient is 0.31242 on the test set which in different experiments result to be an asymptotic value for a naive classifier. Figure 3.12 shows the behaviour of the

model with 15 epochs using the Dice loss. Notice in figure 3.12b the asymptotic behaviour in the test set evaluated on the IoU coefficient, and in figure 3.12a the learning process using the Dice loss as cost function.



(a) Dice loss function (b) IoU Coefficient

Fig. 3.12 Training with Sørensen-Dice metrics

With these results it was decided to take a different direction, using as cost function the sparse-categorical cross-entropy, due to the advantage to properly codify the data and to perform classification and segmentation tasks at the same time.

It is worth to mention that, linear combination of the cost functions were carried out. Using Jaccard loss as a cost for segmentation and binary-cross entropy loss as a classification cost. However, due to time limitations results were not conclusive. This will be proposed as future work in section 3.3.2.

3.2.2 Activation Functions

As described in section 3.1.5 two activation function were considered. However, the selection of the activation function is strongly related to the cost function selection and output codification. Since the selected cost function is the sparse-categorical cross-entropy the activation function

chosen for the output layer is the softmax function, since it is able to properly distribute the probability throughout the classes that are being classified.

Figure 3.13 shows the difference between sigmoid or logistic function (figure 3.13a) and the softmax function (figure 3.13b). Notice that the result of the sigmoid function correspond to a binary classification while the softmax correspond to a multiclass single label problem.

(a) Output with sigmoid activation function

(b) Output with softmax activation function

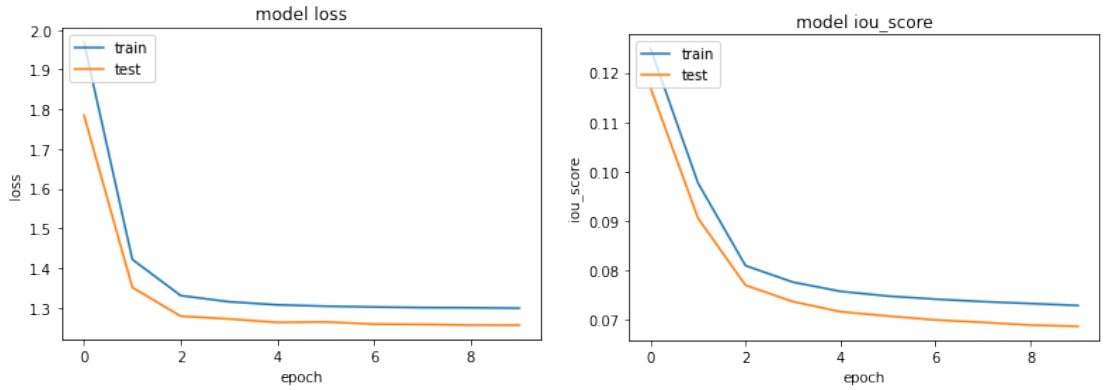
Fig. 3.13 Output mask with different activation function on the output layer

Besides this, it is worth to mention that a wrong selection of the combination of the cost function and the output layer activation function could create a complete useless model as seen in figure 3.14

3.2.3 Model Metrics

The importance of the evaluation metric is related to the way the user would like to interpret results, as mentioned in section 2.2.5 the model metrics are not actually optimized by the optimization algorithm so it is possible to train a model with no metric at all.

Since it is used only for quantitative validation of the model the metric chosen is evaluated after postprocessing the outputs. The most suitable metric for the task is the Sørensen-Dice



(a) Model loss (b) Model metric

Fig. 3.14 Results of wrong combination of cost and activation function

coefficient, due to its relationship with the F1 score and the fact that is a metric of similarity at pixel level so it is able to measure both classification and segmentation. Besides, since it is equal to the F1 score, as mentioned in section 3.1.3, it is less sensitive to class imbalance.

In order to properly measure the Dice coefficient the output masks where compared to a one-hot encoded version of the sample image, this was performed as a postprocessing task. Doing this allows the metric to properly measure similarity at pixel level per class.

A naive approach was to use accuracy at pixel level. However, as seen in figure 3.15 an accuracy near to 98% represent a very bad segmentation due to class imbalance, not only in the quantity of images per class, but also at the quantity of pixels with surface defects at all. This metric was immediately discarded.

Fig. 3.15 Useless output with high accuracy

3.2.4 The Proposed Algorithm

Once defined the cost function, the activation function, the proper metric and the best way to codify the data with some postprocessing, it is possible to propose a model. Notice that in the baseline approaches the best results achieved 0.6039 in the Dice coefficient and visually the masks were not good enough.

The proposed algorithm consist in the following steps:

1. Deal with class imbalance and key feature extraction

    (a) Load a U-net architecture with a "Resnet-34" backbone and weights trained on the ImageNet dataset.

    (b) Define the number of classes and set the activation function of the output layer to softmax.

    (c) Freeze the weights of the base or encoder part of the U-net. In order to use the backbone only as a feature extractor.

    (d) Use a categorical encoding for each target in which each pixel corresponding class is represented strictly by a different integer number.

    (e) Perform a stratified sampling until nearly balancing the classes, use a value close to the mode of classes frequency as reference for number of samples per class. Downsample classes but not oversample any data.

    (f) Define as cost function the sparse categorical cross entropy with class weights in the cost when necessary to increase model "attention" to those classes that still are undersampled.

    (g) Feed the model with batches of size 32.

(h) Train the model with the default parameter for the optimization algorithm, make sure to train it until the validation set loss presents an asymptotic behaviour (50 epochs).

2. Fine tune and refine the model to reduce overfitting.

   (a) Load all the data, and perform a holdout sampling.

   (b) Using the weights and model previously learned, set to trainable, or unfreeze, the base or encoder layers.

   (c) Reduce the learning rate of the optimizer in order to preserve the parameter already learned and also to avoid damage of the generic features on the encoder. In this case the learning rate was changed from $1e^{-3}$ to $1e^{-5}$.

   (d) Set the cost function to sparse categorical cross entropy with with no weighted classes. The activation function of the output layer must remain the same.

   (e) Train the model for 10 times less epochs than the first train, (5 epochs), in order to prevent overffing, corrupt the learned features and, to get the model affected by the class imbalance present.

Figure 3.16 presents the general recipe to develop an algorithm based on the fine tuning method with the "U"Net architecture and Resnet-34 backbone.

The results using this approach are shown in figure 3.17. Notice that figure 3.17b, which is the ground truth, versus figure 3.17c, which is the model output, is similar. Although the model overestimates the area of defects, the segments are clearly defined and also notice in figures 3.17d, 3.17e, 3.17f, 3.17g, 3.17h that the classification is accurate and the pixels are segmented in their corresponding mask.

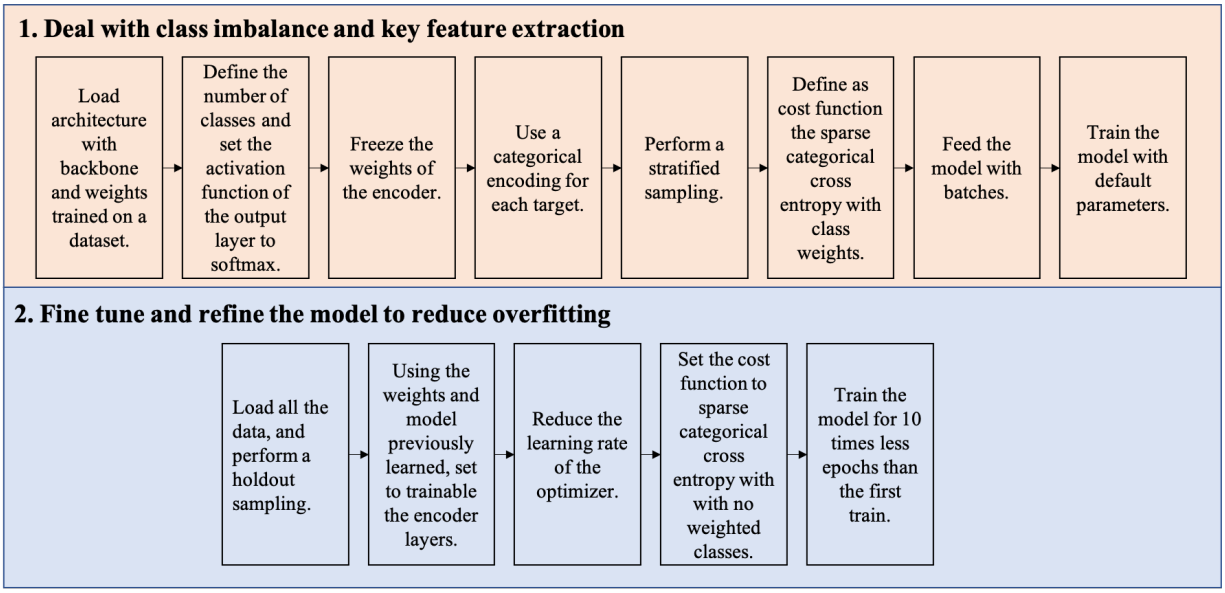


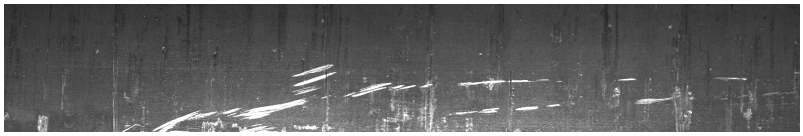
Fig. 3.16 Proposed recipe of the algorithm. *Created by autor*

Using this approach the value of the dice coefficient is **0.83536** which is a great improvement. This proposed method deals with class imbalance by downsamping the images while for the reduced number of labels it seizes the approach of transfer learning.

When unfrozen the encoder, this model trains 24.439.384 parameters which is a considerable big network which trained from zero could not be possible using a single GPU and this few quantity of images.

Although the results are satisfactory, there are some limitations of this implementation. Notice that this model is pretrained using the ImageNet dataset, however this task (defects on steel surface) is very particular and maybe the features extracted from the ImageNet dataset are not the best for this particular task. On the other hand, the huge amount of parameters to train makes impossible for the model to be trained from scratch with the available data.

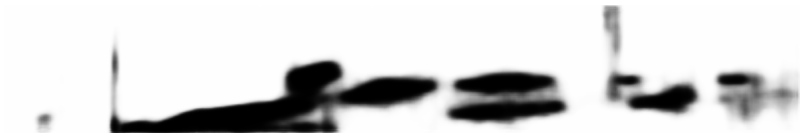
Despite this, there is a lot of techniques and strategies to improve the model including data augmentation, different definition of the cost function, or even changing the parameters of the optimization algorithms. The main limitation is the huge amount of parameters required to be



(a) Real Image

(b) Expected output

(c) Model output (argmax mask)



(d) Mask for class 0 (No defects)

(e) Mask for class 1

(f) Mask for class 2

(g) Mask for class 3

(h) Mask for class 4

Fig. 3.17 Proposed algorithm output

trained and, the way the model extracts features and seizes the information encoded in the feature maps.

Finally in the next section a different approach is used to outperform the proposed method and its limitations are discussed.

3.2.5 The Deepwise Separable Convolution Implementation

This approach uses an Xception-like backbone as proposed by [70] over a U-net architecture. The key characteristic of this model is that it utilizes the deepwise separable convolution method. This drastically reduces the amount of parameters required to encode feature maps, meaning that this method is able to extract better representations of information.

Basically the model consist in multiple blocks of 32, 64, 128 and 256 filters with the following structure:

1. An input layer.
2. For the encoder (for each in $[32, 64, 128, 256]$):
   (a) An activation layer with ReLu function.
   (b) A deepwise separable convolution with kernel size of 3x3.
   (c) A batch normalization layer.
   (d) An activation layer with ReLu function.
   (e) A deepwise separable convolution with kernel size of 3x3.
   (f) A batch normalization layer.
   (g) A maxpooling layer of size 3x3.
   (h) A residual projection for each block in order to be used in the upsampling process in the decoder (concatenation of bypassed information from inputs).

3. For the decoder (for each in $[256, 128, 64, 32]$):

   (a) An activation layer with ReLu function.

   (b) A deconvolution operation with kernel size of 3x3.

   (c) A batch normalization layer.

   (d) An activation layer with ReLu function.

   (e) A deconvolution operation with kernel size of 3x3.

   (f) A batch normalization layer.

   (g) An upsampling 2x2 operation.

   (h) A residual projection upsampling for each block (concatenation of bypassed information from inputs).

4. An output layer which is a convolutional layer where the number of filters is equal to the number of classes and a kernel size of 3x3, with a softmax activation function.

Using this model the number of trainable parameters was reduced drastically to just 2.055.781, this implies a reduction of near ten times from the previous approach. Nevertheless, the information codified in the feature maps is supposed to be at least as efficient as the previous model.

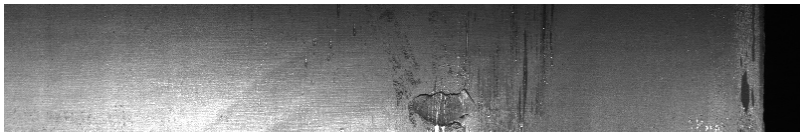
It was trained directly in the whole dataset with no class weight on the cost function. Besides this, since this network has fewer parameters it was trained completely from scratch. That is to say, that the encoder is unfrozen during the training phase. This model was trained for just 5 epochs. Also, the time required for its training is lesser. The optimization algorithm used was Adam with its default learning rate $1e^{-3}$.

Figure 3.18 presents the results obtained with this model, figure 3.18a shows the real image. Notice that figure 3.18b and figure 3.18c are almost identical which implies a high quality

segmentation. The expected output and the model output (argmax reconstruction) are in grayscale, implying that in a single image, there are two or more types of defects. In figures 3.18d, 3.18e, 3.18f, 3.18g, 3.18h the detail of masks, one for each class, is presented, here one can identify that the defects present belong to classes 3 and 4.

With this model a Dice coefficient of **0.9062** was achieved. This model improves considerably the performance while requiring less preprocessing and fewer parameter to be trained. Besides, this approach proves that it is not necessary to use transfer learning when there are not enough samples, instead a clever approach to feature extraction will encode more information that could be used to outperform bigger models.

Nevertheless, some limitations exists with this model, the fact that it reaches a good performance without class balance poses a question about the impact of this problem in the model. Additionally, the approach of class balance and retraining used previously in section 3.2.4 is not possible, since this model practically converges in the first epoch. One hypothesis is that it extracts all the information available that further iterations are not useful. Other concern with this approach is the risk of overfitting and how to deal with it, although it seems to currently not impact the model is necessary to test other techniques such data augmentation, dropout, and weight decay to further improve the model.



(a) Real image (input)

(b) Expected output

(c) Model output

(d) Mask for class 0 (no defects)

(e) Mask for defect type 1

(f) Mask for defect type 2

(g) Mask for defect type 3

(h) Mask for defect type 4

Fig. 3.18 Output using deepwise separable convolution(Xception)

3.3 Conclusions & Future Work

3.3.1 Conclusions

- Transfer learning is a powerful way to leverage of big datasets that are used to develop key features that can be latter utilized for a specific task. It reduces the hypothesis space in order to facilitate the learning process. Although it is widely used it must not be assumed to be the best option. As seen in this project, other techniques such as depthwise separate convolution are able to reach good results with a model trained from the scratch. It is important to understand their advantages and limitations and when is the best option to use each. It is possible to say that deep learning is not completely limited to big datasets as it is usually believed.

- The deep learning world is growing so fast that a state-of-the-art technique two years ago, currently is obsolete. A best practice when diving into a deep learning project is to read and investigate as much as possible, since this trending topic is generating a lot of developments, techniques and strategies that improve models day by day. The oldest model used in this project is four years old, however it has been outperformed by far with new approaches and, now models such as “resnet34”, “VGG16” or “AlexNet” are taught as classic models. This fast development in the AI world commit researches to be in a constant learning process and obliges them to constantly reevaluate their models and adapt them to the new trends.

- The Xception architecture, particularly the concept of the 1x1 convolution has proven to be efficiently for the task proposed in this project, as stated by its author it could change the path for further developments on CNN. Notice that most of the ideas used in these models part from relatively old concepts developed during the XX century, what has changed is the way these tools are implemented. Besides, deep learning has seen an exponential grow

mostly thanks to powerful computation that everyday is becoming more accessible for anyone. It is important to consider how the future of computation will impact AI.

- There is not perfect model, and the way deep learning works is similar to an art, it is important to be creative and implement different approaches or other tools out of the domain of AI in order to improve the performance of the model. Although it extracts automatically the key features, it is advantageous to clearly comprehend the real world task, so that way it is possible to "codify" reality into a model in the best way.

- Although class imbalance affects deep learning models there are a lot of other variables to control that usually the simplest approach to resolve this issue works the better at great scale. It is evidenced that a combination of simple solutions work fine to develop an acceptable model. Data augmentation is a key tool that should be further studied particularly for this task, in how it affects the identification of defects on the steel surface.

- The large amount of components involved in a deep learning project implies that a person must be prepared in optimization, calculus, programming, algebra and others. This pose a challenge particularly in the local market, since in the next few years the demand of professionals with these abilities will grow and it is possible, if the people are not prepared to face a scarcity of AI professionals.

- Convolutional Neural Networks have opened the door for a lot of vision tasks that previously were only developed by computer scientist with a specialization in computer vision. This allows a widespread opportunities for developers to create new applications, the challenge is to implement this technology in real world applications that can impact positively in the local economy and businesses. It is important to work jointly with other professionals to discover new creative ways to apply these technologies.

3.3.2 Future Work

- To further experiment with these architectures and evaluate their potential in other applications. Due to computational limitations and other factors including time it was not possible to fully analyze the behaviour of the Xception model. Also it will be important to evaluate different cost function configurations including lineal combinations, customized costs, or weighted classes to explore its behaviour and eventually improve the model.

- To Evaluate the implementation of data augmentation particularly advanced techniques based on synthetic data, such as generative models, in order to deal with class imbalance and data scarcity. Since this is a defect detection task it is difficult in a controlled process to create a representative sample for defects, then this synthetic data could be used not only for training a more powerful model but also to be used in an unsupervised environment where images are able to auto tagging themselves and classify new errors prior to be taught that.

- To utilize traditional computer vision techniques for data preprocessing in order to facilitate the feature extraction process and improve the model. This is important when the developer has previous knowledge of the task and, somehow, can assist the model to increase its performance.

- Recreate a similar case study with a local business to evaluate not only the technical aspects of the development and implementation of a deep learning model but also the social, cultural and other barriers that could hinder or impact the implementation of AI in Colombian Industries.

References

[1] Stephen C-Y. Lu. Machine learning approaches to knowledge synthesis and integration tasks for advanced engineering automation. *Computers in Industry*, 15(1):105 – 120, 1990.

[2] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: State of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.

[3] Keliang Zhou, Taigang Liu, and Lifeng Zhou. Industry 4.0: Towards future industrial opportunities and challenges. *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*, pages 2147–2152, 2016.

[4] Jim Davis, Thomas Edgar, James Porter, John Bernaden, and Michael Sarli. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers and Chemical Engineering*, 47:145–156, 2012.

[5] Marco Ardolino, Mario Rapaccini, Nicola Saccani, Paolo Gaiardelli, Giovanni Crespi, and Carlo Ruggeri. The role of digital technologies for the service transformation of industrial companies. *International Journal of Production Research*, 56(6):2116–2132, 2018.

[6] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus Dieter Thoben. Machine learning in manufacturing: Advantages, challenges, and applications. *Production and Manufacturing Research*, 4(1):23–45, 2016.

[7] Stefan Lang. *Durchgängige Mitarbeiterinformation zur Steigerung von Effizienz und Prozesssicherheit in der Produktion*. doctoralthesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2007.

[8] Ethem Alpaydin. *Introduction to machine learning*. The MIT Press, 2020.

[9] S. Doltsinis, P. Ferreira, and N. Lohse. Reinforcement learning for production ramp-up: A q-batch learning approach. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 610–615, 2012.

[10] Gülser Köksal, İnci Batmaz, and Murat Caner Testik. A review of data mining applications for quality improvement in manufacturing industry. *Expert Systems with Applications*, 38(10):13448 – 13467, 2011.

[11] D. Pham and A Afify. Machine-learning techniques and their applications in manufacturing. *Proceedings of The Institution of Mechanical Engineers Part B-journal of Engineering Manufacture - PROC INST MECH ENG B-J ENG MA*, 219:395–412, 04 2005.

References

[12] Wen Chen, Yiping Gao, Liang Gao, and Xinyu Li. A New Ensemble Approach based on Deep Convolutional Neural Networks for Steel Surface Defect classification. *Procedia CIRP*, 72:1069–1072, 2018.

[13] Yiping Gao, Liang Gao, Xinyu Li, and Xuguo Yan. A semi-supervised convolutional neural network-based method for steel surface defect recognition. *Robotics and Computer-Integrated Manufacturing*, 61(September 2018), 2020.

[14] Ruoxu Ren, Terence Hung, and Kay Chen Tan. A Generic Deep-Learning-Based Approach for Automated Surface Inspection. *IEEE Transactions on Cybernetics*, 48(3):929–940, 2018.

[15] Jiabin Jiang, Xiang Xiao, Guohua Feng, Zichen Lu, and Yongying Yang. Detection and classification of glass defects based on machine vision. 1110210(September 2019):34, 2019.

[16] Kun Liu, Heying Wang, Haiyong Chen, Erqing Qu, Ying Tian, and Hexu Sun. Steel Surface Defect Detection Using a New Haar-Weibull-Variance Model in Unsupervised Manner. *IEEE Transactions on Instrumentation and Measurement*, 66(10):2585–2596, 2017.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3(January):2672–2680, 2014.

[18] Stephen Ezell Robert Atkinson. The Manufacturing Evolution. 2019.

[19] The fourth industrial revolution what it means and how to respond, 2017. Date retrieved 22-03-2017.

[20] Bill Lydon. Industry 4.0: Intelligent and flexible production. *INTECH*, 63(3):12–17, May 2016. Date retrieved 27-03-2017.

[21] Stephen Ezell. Why Manufacturing Digitalization Matters and How Countries Are Supporting It. *Information Technology & Innovation Foundation*, (April):1–66, 2018.

[22] Ying Cheng, Ken Chen, Hemeng Sun, Yongping Zhang, and Fei Tao. Data and knowledge mining with big data towards smart production. *Journal of Industrial Information Integration*, 9:1 – 13, 2018.

[23] Alejandro Betancourt. Making AI Work with Small Data, 2020.

[24] MinTIC. CONPES de transformación digital promoverá la competitividad del país y la eficiencia del sector público, 2019.

[25] Dinero. Así son las pymes colombianas, Sep 2019.

[26] Portafolio. El país se alista ante la tarea de regular la inteligencia artificial, Jan 2020.

[27] Editorial La República S.A.S. En qué están usando las empresas locales la inteligencia artificial en la actualidad, Nov 2019.

[28] Microsoft. Con el impulso de la inteligencia artificial, colombia podría triplicar su productividad y aumentar su pib hasta un 6.8%, Nov 2019.

[29] Cámara de Comercio de Bogotá, Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia, and International Chamber of Commerce. Observatorio de la economía digital de colombia, 2018.

[30] Cámara de Comercio de Bogotá. Inteligencia artificial en la industria de alimentos, 2019.

[31] Severstal. Presentation of the company. Date retrieved 20-10-2020.

[32] Severstal. Severstal: Steel defect detection, 2019. Date retrieved 01-07-2020.

[33] Severstal. Presentation of the company, 2019. Date retrieved 01-07-2020.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2017.

[35] Cognex. Deep learning for factory automation combining artificial intelligence with machine vision, 2019. Date retrieved 06-05-2020.

[36] Jamshed Khan. Everything you need to know about visual inspection with ai, Apr 2020. Date retrieved 10-10-2020.

[37] Machine vision, Oct 2020. Date retrieved 28-11-2020.

[38] James Le. The 5 computer vision techniques that will change how you see the world, Jan 2020. Date retrieved 05-07-2020.

[39] A beginner's guide to object detection, 2017. Date retrieved 20-11-2020.

[40] Ai object tracking. Date retrieved 20-11-2020.

[41] Hasan Tariq. Semantic segmentation, May 2018. Date retrieved 20-11-2020.

[42] Intance segmentation. Date retrieved 20-11-2020.

[43] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. O'Reilly, 2018.

[44] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[45] Dana H Ballard, Geoffrey E Hinton, and Terrence J Sejnowski. Parallel visual computation. *Nature*, 306(5938):21–26, 1983.

[46] Haar-like feature, May 2020. Date retrieved 22-11-2020.

[47] François Chollet. *Deep learning with Python*. Manning Publications Co., 2018.

References

[48] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer, 2006.

[49] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[51] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[52] Matthew Zeiler. Adadelta: An adaptive learning rate method. 1212, 12 2012.

[53] Cloud gpus (graphics processing units) google cloud, 2018. Date retrieved 30-11-2020.

[54] The pandas development team. pandas-dev/pandas: Pandas, feb 2020.

[55] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[56] Alex Clark. Pillow (pil fork) documentation, 2015.

[57] François Chollet et al. Keras. https://keras.io, 2015.

[58] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[59] Pavel Yakubovskiy. Segmentation models. https://github.com/qubvel/segmentation_models, 2019.

[60] Jeremy Jordan. An overview of semantic image segmentation., Nov 2020. Date retrieved 25-11-2020.

[61] Jaccard index, Sep 2020. Date retrieved 23-11-2020.

[62] Softmax function, Nov 2020. Date retrieved 24-11-2020.

[63] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[64] Jason Brownlee. A gentle introduction to transfer learning for deep learning, Sep 2019. Date retrieved 21-11-2020.

[65] 2018. Date retrieved 02-10-2020.

[66] Jeremy Zhang. Unet line by line explanation, Oct 2019. Date retrieved 28-11-2020.

[67] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[68] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[69] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[70] Keras Team. Keras documentation: Image segmentation with a u-net-like architecture, Apr 2020. Date retrieved 22-11-2020.