Polytechnic University of Marche

Faculty of Engineering

Master's Degree
Computer Engineering and Automation

Augmentation of synthetic data using GANs for improvement of textures classification task in a real use case of Industry 4.0

Aumento di dati sintetici mediante GANs per il miglioramento del task di classificazione di textures in un caso d'uso reale di Industria 4.0

Thesis Supervisor:
Prof. Emanuele Frontoni

Co-supervisor:
PhD. Luca Romeo
MEng. Riccardo Rosati

Student:
Freshman number:1100690
Alex Giacomini

Academic Year 2021-2022

Alla mia famiglia e a tutte le persone che hanno creduto in me.

Ringraziamenti
Per cominciare, vorrei ringraziare i miei professori delle superiori, Prof. P.G. in particolare, senza i quali tutto questo non sarebbe potuto succedere, complici di un influenza e di un incoraggiamento nella scelta del percorso universitario dovuto ad una condivisa passione e visione per il mondo informatico e dell'automazione.
Ringrazio allo stesso modo la mia famiglia, tra cui Mamma,King,Rita, la pestilenza Leo, e il braccio destro del diavolo Viola, senza i quali questo percorso non sarebbe potuto compiersi, e fonte continua di incoraggiamento e supporto, sia materiale che psicologico. Altrettatanta riconoscenza e gratiduine per tutti i coinquilini universitari, Dan, Mason, Friedrich, Super M., Fonta., senza i quali il percorso universitario sarebbe stato un arduo viaggio del Purgatorio. Un riconoscimento importante va alla Casa Verde, ancora di salvezza e di abbuffate varie, ma in particolare al suo simbolo Sara P. (altresì conosciuta come 'Sarvee'), annata ritenuta da me forse come la più bella del percorso magistrale (chissà, forse perchè insediatoci come sanguisuga, ahah). Altrettanti grazie a tutti i ragazzi del percorso magistrale che hanno condiviso con me momenti di svago, ansia, ma soprattutto di condivisione che riconosco essere costruttivi tanto quanto il percorso universitario effettivo. Quindi ringrazio in particolar modo il Capitano Giorg, Matthew Cos., Sir Lore. D'All., Laura D.S., Sir G.Patton, Oliver, Bea, e il restante meraviglioso equipaggio dell'Olandese Volante. Ulteriori ringraziamenti e rispetto a tutti i numerosi ragazzi e colleghi con cui ho condiviso lo sviluppo di progetti universitari, e che mi hanno permesso di condividere un percorso di crescita ulteriore, sia personale che di capacità. Un grande grazie anche a tutti i miei amici Urbaniesi e Sant'angiolesi di sempre per tutti i momenti di svago passati, sempre di un livello meritevole non indifferente (rappresentati come la scala di misura della qualità, altresì il "SUCCO della vita"). Estrema gratitudine e riconoscenza vanno al Prof. Frontoni, e in particolare a Luca Romeo e Riccardo Rosati per l'interesse e la grande disponibilità dimostrata nel prolungato periodo di tirocinio svolto, e della opportunità Americana fornitami, persone d'oro. Infine devo ahimè ringraziare l'ago della mia bilancia, forse quello che ha gaarantito inaspettatamente più di tutti una crescita personale ma soprattuto umana, la numero uno dei "C", e la punta della piramide, Laura F., un'incantata come poche altre e un'elemento della tavola periodica anche forse più innaturale del Bismuto, ma dotata di un'amore, umanità e comprensione indiscutibili, più di ogni altra persona che abbia mai conosciuto.

Abstract

The thesis, focused in performing a Supervised Q.C. (quality control) Multi-class classification task, originated from a specific company's demand (Benelli Armi S.p.a. Sec.1.2). In particular, the aim of this work is to classify rifle stocks quality grades, in order to correctly following all the procedure to finally sell the final shotgun with different prices depending on the quality of the rifle stock itself.
It has to be underlined one of the problems acting in these kind of Deep Learning experiments, which is, in this particular case of study, the Unbalanced dataset of input samples used to train the classifier network.
The goal of the project will be the design of two Data Augmentation strategies in order to balancing the training set, and at the same time improve the prediction task of quality classes.
To achieve this end, the implementation of an Online Data Augmentation approach is implemented, based on CV techniques (ON.-D.A.-C.V.T). On the other side a GAN (Generative Adversarial Learning) architecture will be carried out to perform an offline Data Augmentation (OFF.-D.A.-GAN), in order to augment the amount of images for minority classes later used to train a VGG16 classifier to achieve the best results possible in accuracy. Both of the Data Augmentation strategies will take advantage of the training set, the same used also for the Classification task.
Results obtained on a real-use case dataset demonstrate that the online D.A. approach based on the basic CV transformations obtained greater results compared to the offline D.A. based on GAN regarding the classification predictions results. At the same time it's guaranteed a reduction of the memory resources needed to complete the training phase as well as a lower computational complexity.

Sommario

La tesi, incentrata sull'esecuzione di un compito di classificazione multiclasse di tipo Supervised Q.C. (controllo qualità), nasce dalla richiesta di una specifica azienda (Benelli Armi S.p.a. Sec.1.2). In particolare, l'obiettivo di questo lavoro è quello di classificare i gradi di qualità dei calci dei fucili, in modo da seguire correttamente tutta la procedura per arrivare a vendere il fucile finale con prezzi diversi a seconda della qualità del calcio del fucile stesso.
Va sottolineato uno dei problemi che agiscono in questo tipo di esperimenti di Deep Learning, che è, in questo particolare caso di studio, il dataset non bilanciato di campioni di input utilizzati per addestrare la rete del classificatore.
L'obiettivo del progetto sarà la progettazione di due strategie di Data Augmentation al fine di bilanciare il set di addestramento e, allo stesso tempo, migliorare il compito di predizione delle classi di qualità.
Per raggiungere questo obiettivo, viene implementato un approccio di Online Data Augmentation, basato su tecniche CV. Dall'altro lato, un'architettura GAN (Generative Adversarial Learning) sarà utilizzata per eseguire un Data Augmentation offline, al fine di aumentare la quantità di immagini delle classi minoritarie utilizzate successivamente per addestrare un classificatore VGG16 e ottenere i migliori risultati possibili in termini di accuratezza. Entrambe le strategie di Data Augmentation sfrutteranno il set di addestramento, lo stesso utilizzato anche per il compito di classificazione.
I risultati ottenuti dal dataset di un reale caso di studio dimostrano che l'approccio D.A. online basato sulle trasformazioni CV di base ha ottenuto risultati maggiori rispetto al D.A. offline basato su GAN per quanto riguarda i risultati delle previsioni di classificazione. Allo stesso tempo è stata garantita una riduzione delle risorse di memoria necessarie per completare la fase di addestramento garantendo allo stesso tempo una minore complessità computazionale.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Definitions

1.1.1 Artificial Intelligence

The Idea of the Artificial Intelligence (AI) field, is essential to develop machines endowed with intelligence and autonomous learning capacities, that help the AI system to adapt itself to problems acting like a human, as much as possible. AI concept comes from 2 different theories:

- Strong AI, according to which machines are able to develop a consciousness, meaning that are aware of the activity carried out. So, all the subjects are connected to develop systems in order to replicate human intelligence;

- Weak AI, according to which retain possible develop machines able to solve specific problems without having consciousness of the task performed. So the goal of this theory is instead not to create machines with intelligence, but to obtain systems able to perform multiple complex human functions, to help him on solving tasks;

So, AI is that branch of the Computer Science Field that studies the development of Systems, both Hardware and Software, equipped with specific capacities typically visible to humans, such as interaction with all the Environments, adapting and learning new things, and even reason and think, and so to pursuing autonomously a defined task.

1.1.2 Machine Learning

Machine Learning (ML) is an Artificial Intelligence (AI) technique that harnesses machine computation to perform complex processing useful in simulating the way a human would

learn under the same circumstances. At the programming level, a Machine Learning application should therefore not be defined by instructing the system on all the operations to be performed, as is the case with traditional software, but by the development of an algorithm capable of acquiring knowledge from the data and becoming progressively more efficient in exploring and processing the data, to formulate predictive analysis rather than, in more advanced cases, making decisions on its own is necessary.
Such a view does not mortify the role of the developer, but it does substantially vary his work. His mission becomes that of accompanying the machine in the autonomous work that artificial intelligence requires. This has triggered the emergence of various professional figures, with skills related to data science and computer science, who are asked to prepare data, monitor the work of the machine, and verify the results of the analyses, using computer tools specifically designed to carry out these activities.

Deep Learning

It's the most important Sub-Branch of ML, which contain an ensemble of techniques able to simulate processes similar to human neurons, through the usage of stratified Artificial Neural Networks (ANN), capable to analyze data and subsequently creating learning models based on multiple connected layers of neurons as shown in 1.1. These stratified layers allow elaborating complex tasks improving time after time during a training phase, passing information between all the levels of the Neural Network, automatically modifying themselves on their own.

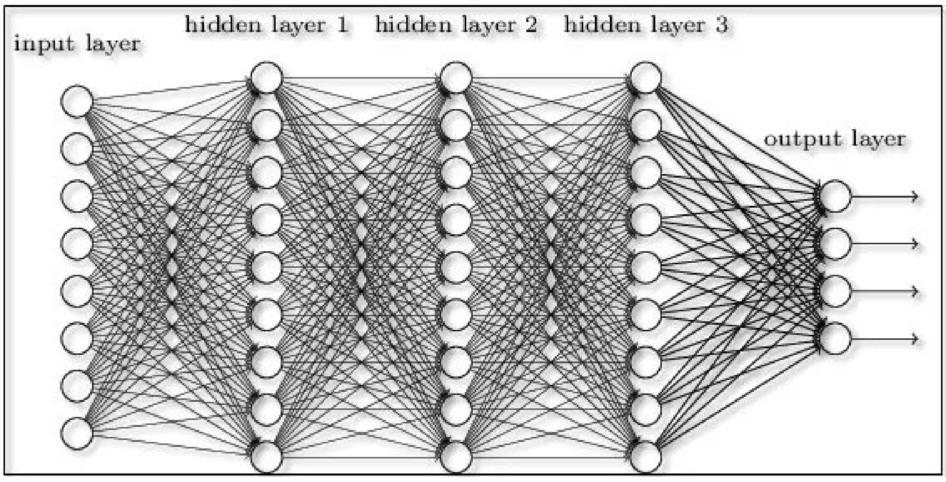


Figure 1.1: Fully Connected Layers - SOURCE

Steps of ML

1. Data collection: capturing and storing data sets in systems such as databases, data warehouses, or data lakes, capable of making them available to business analytics applications;

2. Data preparation: useful operations to clean data to increase overall data quality and facilitate subsequent operations in the analysis phase;

3. Data exploration: knowledge aimed at descriptive analysis activities, with the possibility of generating through data visualization that graphical information useful to better understand the meaning of data, through their mathematical characteristics;

4. Data Learning: constitutes the crucial phase of a Machine Learning process and includes an additional preparatory activity, which appears preprocessing, useful for reworking the data to optimize statistical properties; feature selection, to understand the most significant aspects and phenomena and transformation, which serves to convert the data into the format most functional for the algorithms that are intended to be used. In the actual learning phase, the Machine Learning model analyzes the data following the directions given to it;

5. Model validation: testing and prediction: these are steps that perform exactly the function intended by their definition until the tests produce results that are deemed reliable enough to actually enter production. In other words, when the Machine Learning system is sufficiently trained, it can be used to perform predictive analysis from new data sets.

Learning Methods alghoritms

There are 3 principal learning methods:

- Supervised Learning, applied when data is categorized, aka there is an association between the data and a specific numerical value, called label. So, data are known from the initial phase, because they are already structured in a specific way. The goal will be to find out the relation between input data used to train the Neural network and testing data used to validate the goodness of the network.
  This is the specific case of study of this thesis, where the Neural architecture has to recognize an image between 10 different quality classes of a rifle stock;

- Unsupervised Learning, differently from the previous one, works with data that isn't organized or neither structured, so the algorithm has to identify autonomously the relations and schemes starting from the input dataset. Data is usually grouped by similarity criteria, putting together those who are similar to each other according to the prediction of the neural network from hidden schemes it discovers. An example is described by clustering, which creates groups of elements with similar features as best;

- Semi-Supervised Learning, where the dataset is a mix of both structured and unstructured information. It then act, using the previous 2 ways of learning, adopting the appropriate strategy. The advantage of using this kind of learning is to refine models previously calculated with the other types;

Fields of application of ML

Nowadays there are many fields of application, reached by the increase of data availability, high computing power, and large storage capacity. On the others, there are autonomous drive, chatbots, decisional supports [11], e-commerce and on-demand services, and even video surveillance, financial trading, and image recognition systems. Moreover we can find predictive maintenance [12, 13] and quality control systems [14, 15, 16] like in this case of study.

1.1.3 Differences Between AI, ML and DL

So, Machine Learning is essentially a part of the wide technological universe represented by Artificial Intelligence, which is also one of the most relevant branches. In detail, Machine Learning and its sub-branch Deep Learning, allow dealing with both the Strong and Weak theories of AI.
Nowadays the biggest improvements regard the Weak theory, while the Strong AI is improving slowly, for reasons that will be discussed in the next subsections. For what concern the difference between Machine Learning and Deep Learning, as has been said, is a particular part of the field that works with many levels of layers passing information to Neurons that take particular decisions. In fact, they are both fed with data, but they handle it in different ways. Machine Learning needs a set of initial instructions, based on structured data and from which it develops autonomy. Instead, Deep Learning learns also from not structured data, which makes it particularly efficient for image problems, video content, or even not tabular documents where there's no organization and structure in the information.

1.2 Involved Organizations in the thesis

This thesis comes from the collaboration born in 2020 between the company Benelli Armi SpA and the Polytechnic University of Marche, through the research group VRAI (Vision Robotics Artificial Intelligence) based in the University area department. Benelli Armi SpA is an Italian manufacturer of firearms based in Urbino (PU, Italy), a leader in the production of semi-automatic rifles for hunting, sports, defense, and exports to 78 countries around the world.

Figure 1.2: Benelli Armi SpA - SOURCE

On the other side, VRAI is a research group of the Computer Science Department of the Polytechnic department which varies their actions in multiple and different tasks such as Industry 4.0, Retail and Geographic Information System. In detail, specific scopes cover Computer Vision and Pattern Recognition, images and signal processing in many heterogeneous application fields, and Earth observation and Retail environments. Goals are obviously on automatizing processes like the interpretation of images obtained by different systems, data acquisition optimized by different technologies and machines and so on.

Figure 1.3: VRAI (Vision Robotics Artificial Intelligence) - SOURCE

1.3 General aspects

In detail, the objective of the paper is to analyze rifle stock images through different approaches and techniques, adopting Computer Vision methodologies and Pytorch framework methods to manipulate images and optimize Deep Learning processes like the VGG16 Neural Network used for the classification task in this project. The Next sections will focus on each of these singular aspects and their logical connection and interaction. In any case, it is good to clarify right away that a case study like this one under consideration, is one of the pivotal ones in the scope of Industry 4.0. However, the high costs due to the resources required and needed to afford these problems are still maintaining slow the progress of the techniques and Systems. In fact, nowadays, most systems are still adopting strategies where a lower level of Physical/Virtual hardware resources is preferred at the expense of more execution time. These last aspects are closely related to the thesis task because of the high-resolution images, which contain a high number of features and Random details represented by grains and veins.
Firstly, these features would require a wide amount of neurons, and so a high number of fully connected layers, which like has been said before, is still not perfectly obtainable because of the limited capacities of the actual technologies, even if some of these systems are already able to analyze and combine more information than humans, creating connections between a number of variables that nobody in the humanity would be able to combine and develop. This will inevitably bring limited accuracy results, considering also that the Multi classification problem is composed of 10 different classes, that at the same are not even too different from each other bringing bias to the accuracy metrics. This problem will translate itself unavoidably in a Huge amount of time needed both for the classifier and GAN trainings as will be observed in the next sections.
Secondly, the problem linked to the limited amount of memory resources available will be addressed by an On the-Fly Data Loader which will load images from a specific batch when needed, and then discarded in advantage to the new batches, repeating always the same procedure, and ensuring avoiding as much as possible to going Out Of Memory (OOM).

1.4 Main Contributions

The thesis focuses on a connected branch of the Computer Science department, based on a real use case of Industry 4.0, aka Data Augmentation of synthetic images (texture images), using D.A. online/offline approaches to improve the classification task results. Thus, the main contributions of this work are summarized below:

- Design of 2 different Data Augmentation strategies to improve classification performances in a complex task such as Q.C. . In particular, both strategies follow the idea of Over-Sampling the minority classes, so that we are able to obtain a balanced dataset to train the Classifier.
  The first strategy is based on Augmenting the dataset images with an Online Data Augmentation approach, by adopting typical CV transformations during the training phase, and by employing a Data Loader in order to reduce both the computational complexity and memory resources usage.
  The second strategy is instead based on testing the efficacy of a GAN (Generative Adversarial Network) which will generate some samples for each unique quality class sub-set of data to be added to the starting unbalanced dataset.

- Test the 2 methods implemented, through the usage of a real dataset created specifically for the Q.C. task. The dataset is directly acquired in collaboration with Benelli Armi S.p.a company and shared with the University with all the acquired images in order to test DL strategies.
  These tests will allow the comparison between the 2 proposed D.A. strategies by comparing results obtained by a VGG-16 Neural Network classifier, implemented to solve the Q.C. task.

Repository Github - Project Code Link
The actual code is executable from the colab Environment, and it's in a continuous update.

Repository Google Drive - Libraries linked to the project code Link
In this Google drive folder, other useful classes can be found implemented and ready to be used, such as a Custom split-data algorithm and even a Custom EarlyStopping Class to monitor any metric wanted during the classifier training phase. Moreover, it contains some GAN evaluation metrics to validate most of the Generative adversarial learning Neural networks.

Repository Github - Dataset CSV (Rifle Stocks info) Link
This file allows to directly load the images in the current project, containing so the filenames with all the labels and their info.

Repository Google Drive - Dataset Repository (Rifle Stocks Images) Link
The current path contains all the cropped images and generated ones by the GAN architecture.

Colab Features and settings All the experiment tests were performed into the Google Colab **Pro** Environment, with TPU accelerator hardware, with a high RAM setting activated.

Chapter 2

Related Work

2.1 State of the Art - Quality Control task in Industry 4.0 domain

Quality control in Industry 4.0 has had many applications recently. Thanks to the increase and progress of systems for data acquisition, manipulation, and organization, the amount of available data is increasing more and more over time, as well as new architectures for Deep Learning and Computer Vision tasks, which have been developed to perform texture classification [17], thus helping to detect production problems and classify product quality [18, 19].
Standard DL approaches were employed in [20] to replace costly quality control procedures based on visual inspection during the welds mass production scenario with the aim to improve the defect detection accuracy. They mainly focused on the collection of a balanced database and image pre-processing.

Quality control through automation tools available on the market is the basis of the development of industry 4.0. However, the unknown of the software allows us to analyze what comes from the instrumentation and therefore lacks defining a good model and the generalization of these analyses carried out by highly-specialized technicians. In particular, if QC was immediately imposed as a discriminating factor between old industry and industry 4.0, it should be noted that one of the factors between adopting a quality control mechanism and not doing so, lies mainly in cost. If on one hand, we have a cost to train highly specialized personnel in order to solve the quality control task, on the other hand, there is a high cost also regarding the instrumentation and the computational and temporal resources that this approach can require. It should therefore be noted that if more and more implementations 4.0 related to the IoT are being born, the monitoring of health and quality of various materials/ products/ instrumentation necessarily need high computational resources to perform their work at best.

Optical Illumination platforms for image acquisition In visual inspection, excellent optical illumination platforms and suitable image acquisition hardware are the prerequisites for obtaining high-quality images [1]. Image processing and adaptation are key technologies in obtaining defect information, while deep learning is significantly impacting the field of image analysis. In this study, some latest developments in industrial defect detection are described as well as some important optical modes of illumination to obtain more quality images, such as the typical modes in Fig. 2.1:

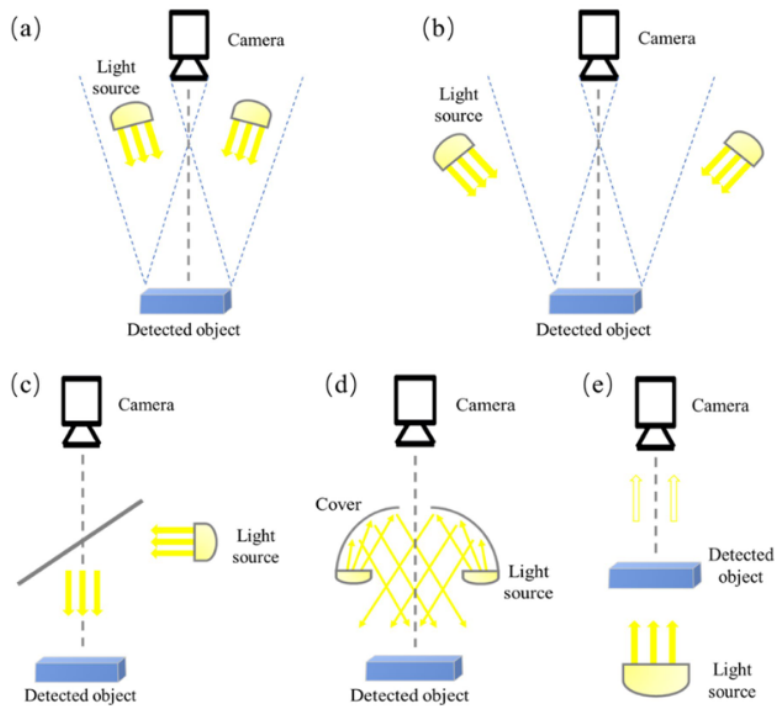


Figure 2.1: Different type of Illumination modes in the Manufacturing Industries to capture images of objects. - Copyright 2020, Zhonghe [1]

The dataset obtained for this case of study of the rifle stocks is closely similar to setting (b) named *Dark field forward lightning*, which obtains the diffuse light caused by the change of surface texture, avoiding specular light reflections and being able to obtain a good quality image for the most Manufacturing cases. More advanced types of innovative illumination is also introduced which will take on in the future in many

fields of application, such as Capturing images for Aviation components analysis, even manufacturing, and so on, bringing more efficient image datasets.
They also introduce a first CNN architecture firstly used for the image analysis 2.2, and than shifted even for more complex tasks like image segmentation, in order to proceed into image defects recognition and even objects detection and identification, which anyway goes into another direction of study.

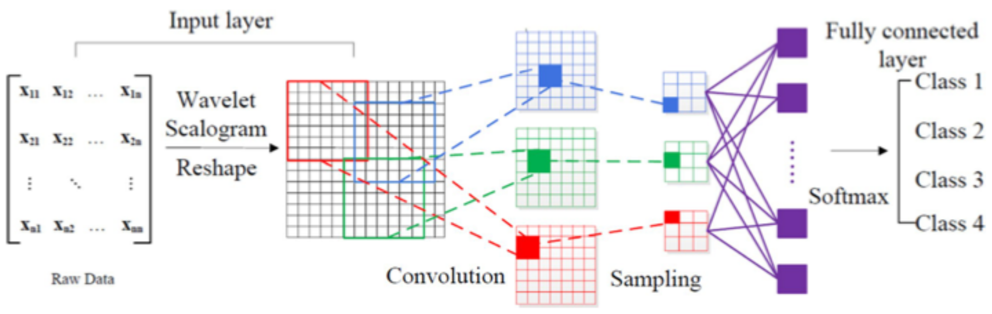


Figure 2.2: Architecture of a CNN model. - Copyright 2018, Elsevier [2]

Combination of DNN with DBN and Boltzmann machine A first approach adopted was to build an architecture composed of a Deep Neural Network (DNN), Deep Belief Network (DBN) and restricted Boltzmann machine [3] to perform a visual inspection process in the printing industry 4.0 by using as input a high-resolution optical quality control camera, to detect eventual problems (binary classification task). The figure (Fig.2.3) shows the development process structured as follows:

1. Data acquisition and pre-processing
2. Extract features via extraction task
3. Classification task, which has two possible outputs: no problems, problems detected

So, this article showed an application of how software sensor application by Deep Learning can be combined with a high-resolution optical quality control camera for precision to reduce the cost of an industrial inspection process in the printing industry 4.0 during the production process of cylinders as illustrated in (Fig.2.4). This implementation is similar to this thesis case of study, where there's an advanced camera used to obtain samples for the Deep Learning architecture, but even more difficult because composed of 10 different quality classes for the rifle stocks, which will bring a lot of uncertainty on the predictions, compared to the above paperwork.

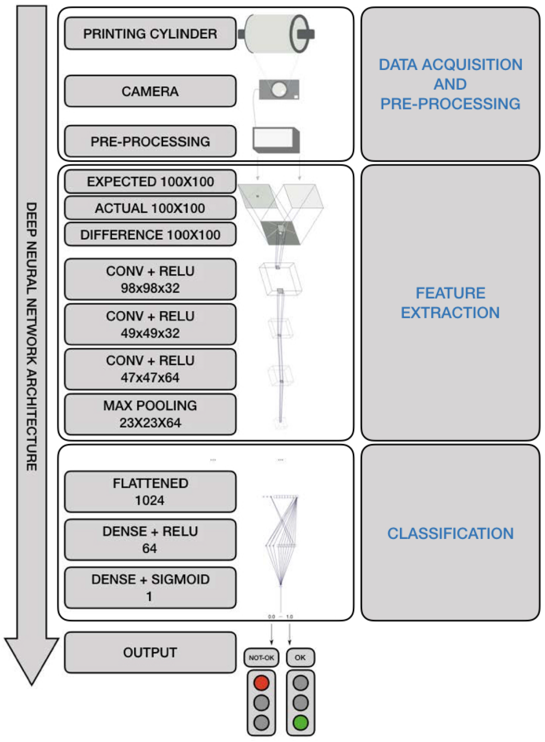


Figure 2.3: DNN and DBN architecture. - [3]

Figure 2.4: cylinders - [3]

Combination of a CNN with ANN Recently, a DL strategy was adopted for detecting geometric inaccuracy of the laser-based additive manufacturing process. They combined the output of a CNN (Convolutional Neural Network) and the output of an ANN (Artificial Neural Network) for analyzing the thermal images and include relevant process/design parameters respectively [4]. Laser-based additive manufacturing is one of those which is a key aspects of the industry in where there's the aim to use many sensors for continuous processes. A current challenge in LBAM is the geometric imprecision of fabrics. In this paper, they develop a novel Deep Learning approach that accurately predicts distortion. Taking into account the limits of LBAM tolerance, the most effective one is the local heat transfer which creates distortions in the image samples.

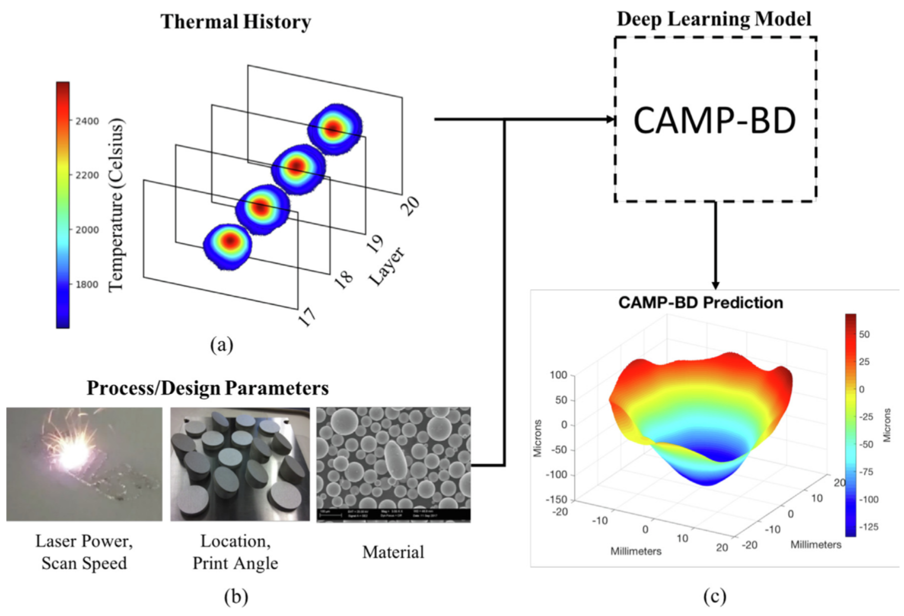


Figure 2.5: CAMP-BD Process. - [4]

This Deep Learning model is so customized for the high-dimensional, thermal history data (e.g., thermal images) and relevant process/design parameters to make point-wise distortion predictions. The developed CAMP-BD (**C**onvolutional and **A**rtificial Neural Network for Additive **M**anufacturing **P**rediction using **B**ig **D**ata) computational framework, contains a Convolutional Neural Network (CNN) for analyzing the thermal images and an Artificial Neural Network (ANN) for including relevant process/design parameters. The output of the CNN and the output of the ANN are joined within CAMP-BD, through concatenation, and then further trained to give a final pointwise distortion prediction. The point-wise distortion is scanned after the part is removed from the build plate, generating a point cloud of the pointwise distortion on each sample. To correlate the thermal images to the x, y, z coordinates in the point cloud, the building code used for fabrication is parsed to track the laser movement during fabrication. Given the location of the laser during the fabrication and the time between images from the thermal monitoring software, the x, y, z locations of each thermal image can be found. For each point in the point cloud, the nearby thermal images are stacked (Fig. 2.5(a)) and used as the input to the CNN, and the relevant process/design parameters are given as inputs to the ANN. The main benefits of CAMP-BD are twofold. First, the leveraging of large data captured in the Industry 4.0 environment by utilizing Deep Learning and HPC allows for scalability and improved predictive analysis with larger data. Second, the model can be generalized to solve other LBAM process control challenges such as porosity and residual stress with appropriate pointwise data. So this last observation, specifies that a capturing system based like in this project for the acquiring phase of the rifle stocks can be a good opportunity to improve the performances of the neural network, in which the Laser could take note better of the grains and veins of the wood piece, guaranteeing better performances in the classification task of the multiple different classes.

HUVGG16 This is the first effective case of use of the rifle stock dataset provided by the company, used as well in this thesis, but with a different approach.
The traditional data quality control (QC) process was generally limited by the high time-consuming and high coffers demand, in addition to a limit in performance substantially due to the high natural variability across different evaluators. The operation of Deep Learning (DL) strategies for working the QC task opens the realm of possibilities in order to overcome these challenges. Still, not everything would be a bed of roses for the incapability to describe bias from the collected data and the threat to reproduce bias in the outgrowth of the DL model. So, this poses a remarkable and undetermined point in the Industrial 4.0 scenario on how to handle these features which brings unwanted bias into the classifier prediction [5]. In the current work, in fact, the rifle stock dataset is composed of different series of production, which means that each of the series is composed of a different geometry that brings inevitably an unwanted contribution to the final results. So, the strategy adopted was to divide the problems into 2 different phases. The first one had the job of classifying the geometry series of each shotgun sample (12 total possibilities as will be shown in the next sections), and feeding the images from each shotgun to a unique classifier with the same structure, but based on the detection of the quality class, guaranteeing an easier classification, due to the limited range of quality grades for each shotgun predicted. This comes from the fact that each shotgun series is produced from the same raw material, bringing the quality and so the veins and the grains of the pieces of wood to be much similar, but obviously each one with a random set of details due to the Tree conformation.

In the figure (Fig.2.6) is described the previously introduced Deep Learning approach, specifically tailored for providing the aesthetic quality classification of shotguns based on the analysis of wood grains without running into an unwanted bias.
Although the proposed DL model based on VGG-16 and ordinal categorical cross-entropy loss has been proven to be reliable in solving the QC task, it is not immune to those who may be unwanted bias such as the typical characteristics of each shotgun series. This may lead to an overestimation of the DL performance, thus reflecting a more focus on

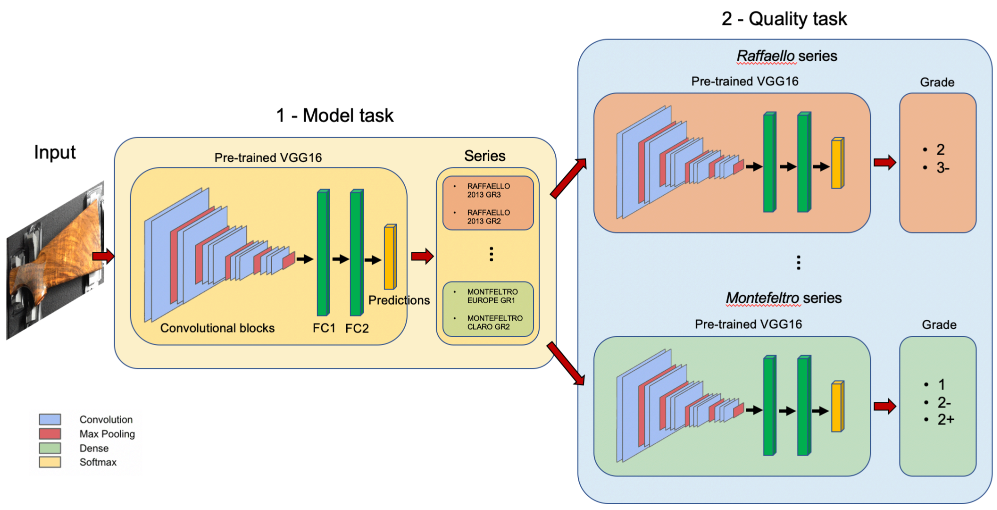


Figure 2.6: Hierarchical Unbiased VGG-16 Architecture. - [5]

the geometry than an evaluation of the wood grain. The proposed two-stage solution named Hierarchical Unbiased VGG-16 (HUVGG-16) is able to separate the shotgun series prediction (shotgun series task) from the quality class prediction (quality task) quite correctly at all, but it has anyway the problem of not generalizing sufficiently on the shotgun series. In fact, imagine getting new samples to test with a new geometry conformation, perhaps it would be classified wrongly.

2.1.1 Use case of the thesis

The main differences with this work compared to the previous [5], lie in the different application of the DL methodology (i.e. we are interested to improve the aesthetic quality classification of shotguns based on the analysis of wood grains, while as already said, reducing as much as possible the resources usage). In fact, the Q.C. classification task uses the same VGG16 architecture adopted in the previous work, with the goal to analyze the goodness of the images that will be generated with the Data Augmentation process and the Augmented set of samples obtained by the GAN implementation and then compare the results obtained with the new configuration.

2.2 State of the Art - Online Data Augmentation - Standard Computer Vision techniques

Data Augmentation was introduced to deal with a lot of study cases (i.e. NLP, Q.C., Health Diseases Discover, Image segmentation, ...) in which the dataset used to train the neural networks in many experiments was often unbalanced. So, Data augmentation involves techniques used for increasing the amount of data, based on different modifications, to expand the number of examples in the original dataset. Data augmentation not only helps to grow the dataset but also increases the diversity of the dataset. Moreover, when training machine learning models, data augmentation acts as a regularizer and helps to avoid overfitting, thanks to the higher diversity brought to the set of features fed to the neural network. Even more important is then the combination of these manipulations with an "On the Fly" approach, corresponding to Online activations of these transformations during the training phase of the Neural networks, in order to guarantee less memory usage and a lower Computational Complexity. This aspect is fundamental in high dimensional samples cases or even where the sizes of the samples of the problem would bring the System to occupy too much storage spaces, bringing the execution of going Out Of Memory (OOM), quitting the executions.

State of the Art - Image Data Augmentation in DL Deep convolutional neural networks have performed remarkably well on many Computer Vision tasks. However, these networks are heavily reliant on big data to avoid overfitting. Unfortunately, many application domains do not have access to big data, such as medical image analysis.
This survey focuses on Data Augmentation, a data-space solution to the problem of limited data. This works encompasses a suite of techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built using them. The image augmentation algorithms discussed in this paper [6] include geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, and more traditional techniques. The example in Fig. 2.7 shows the Data Augmentation logic of execution with a Crop manipulation to extract a patch from the image and later apply a horizontal flip transform to the previous image obtained.

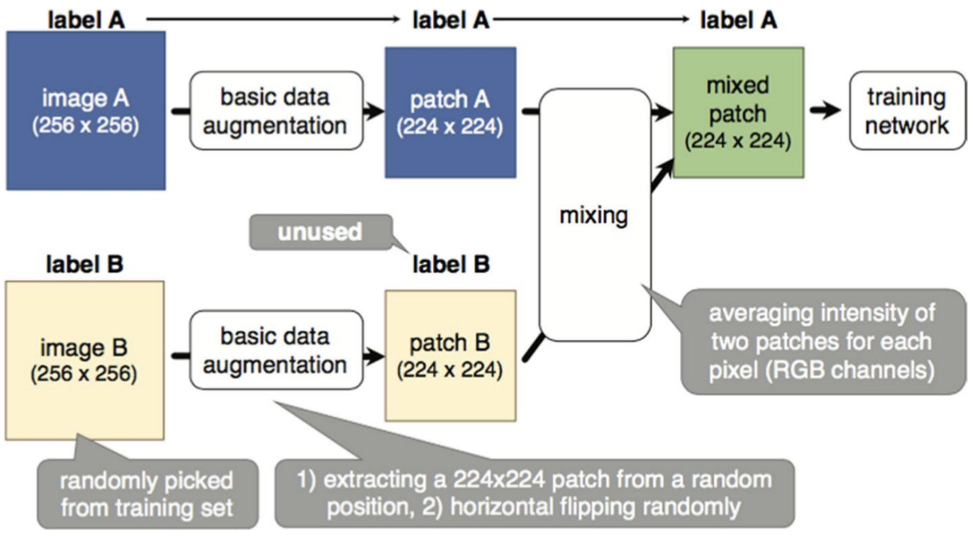


Figure 2.7: Process of Images Data Augmentation. - [6]

Similar to the example presented by this paper, even this work will focus on applying some transformations in order to increase the input data for the classifier.

Online Data Augmentation with less domain knowledge Online Data Augmentation is an important process in order to increase the number of input samples for training networks with the goal of helping the training phase by loading batches of images during the training phase in order to reduce memory resource usage both with a decrease of Computational complexity. This particular work ([7]) focuses on identifying performance and scalability issues in current data loading implementations. After that, they propose optimizations that utilize CPU resources for the data loader design. An illustration process of the data loading logic is showed in 2.8.

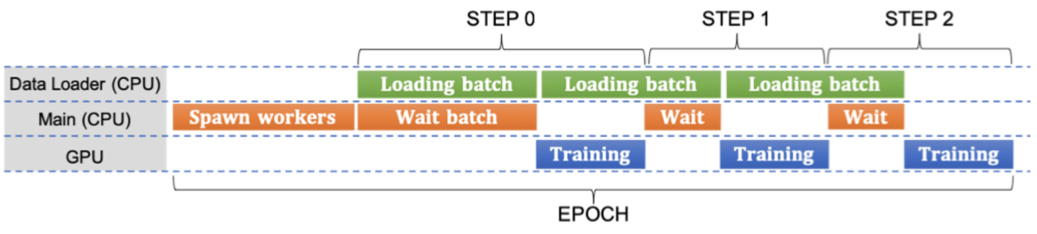


Figure 2.8: Illustrative execution timeline of a learner. - [7]

2.3 State of Art - Offline Data Augmentation - GAN

These new kind of architectures were born in 2014 like an automatic learning system with the goal to generate its own new images, basing the creation from a training set of real data, used to increase the capacity of a Generator architecture, improving epoch after epoch, and each time validated and improved more even from the action of a second parallel architecture named Discriminator. Those 2 networks compete with each other trying to increase the ability of the production of the images. Nowadays different architectures have been developed over time, but most of them still require big computational afford, so taking lots of memory resources while having at the same time a huge computation complexity.

GAN According to Goodfellow J. et Al. [21], found that Deep Learning models have not only the ability on learning features and classify them both for supervised and unsupervised tasks, but they are also able to take advantage of those features in order to create real images, starting from learning features from other samples. They proposed a strategy composed of 2 rival networks, named Generator and Discriminator.
The first one architecture has the job of creating imitations of the training samples even studying as much as possible which features are shared between all the input samples. The new image is not an exact duplication of one of the input ones, but a completely new one, with similar nature. In fact, the Generator learns the features distribution in the image space over each epoch of training and combines it time after time in different manners, to reach the creation of many different new configurations. For what concern the Discriminator network, it has instead the job to verify and check the reality of the data received from the Generator, like a forgery expert. The image will be classified as forged (falsified) not only when it's too deviated from base samples, but also when it is too perfect as it couldn't be. This last aspect means that sometimes when the input samples are homogeneous and not too different from each other, it can happen to obtain so precise images, which results clearly fake, aka created from an artificial neural network. As previously said, both the networks work against each other, where the first tries to fool the second one by creating images as real as possible, and the other has to do the "party pooper", finding out the bad operate of the Generator. Of course, as soon as the generator improves itself in creating better images, the antagonist will improve its capacity on recognizing real/unreal images, both learning features from input samples and the generated ones.
Said in technical terms, the Generator framework captures the data distribution in all

the pixels image space, organizing features and adapting them to create new items with a contribution of noise (usually denoted by Z) to complicate its job and let it become more robust, while Discriminator estimates the probability that a sample came from the training data rather than Generator. The training procedure for G is to maximize the probability of D making a mistake. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. The Figure Fig.2.9) shows the interaction of the 2 networks with the phase of backpropagations for the neuron weights in the fully connected layers of the networks at the end of the calculations of both them, each epoch passes.

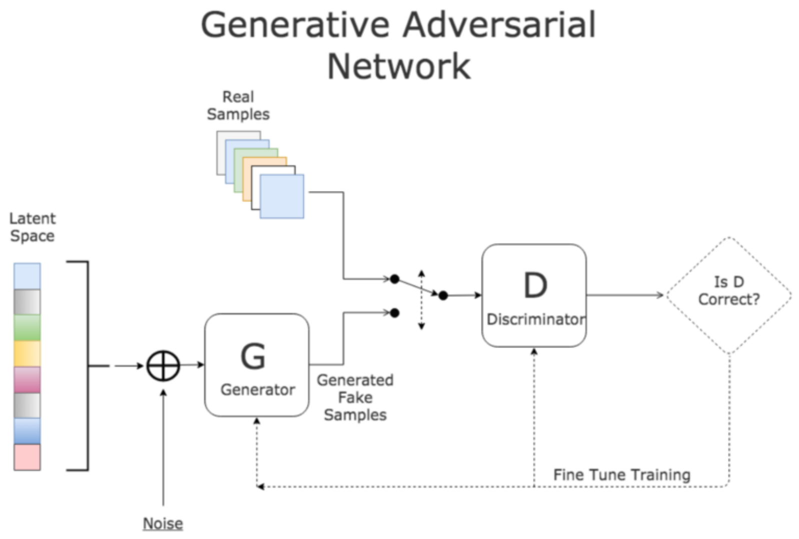


Figure 2.9: GAN Goodfellow et Al. architecture. - SOURCE

This will be the approach adopted, indeed the principal goal of the project was to implement different metrics to evaluate the goodness of the images in respect of the originals.

WGAN-GP Similar to the Manufacturing field applications, this work focus on the classification of road pavement textures, generating new samples through a WGAN-GP network architecture, and subsequently testing the accuracy of the images adopting different ML network architectures. According to Ning Chen et Al. [8] as one of the most important properties of roads, the adaption to roads with different macro-textures may significantly affect the autonomous driving technologies since road texture directly affects the skidding resistance and tire noise. Therefore, it is of great significance to detect and analyze the road macro texture with respect to different pavement types and service conditions. Fig.2.10 denote the Flow chart of the study just introduced.

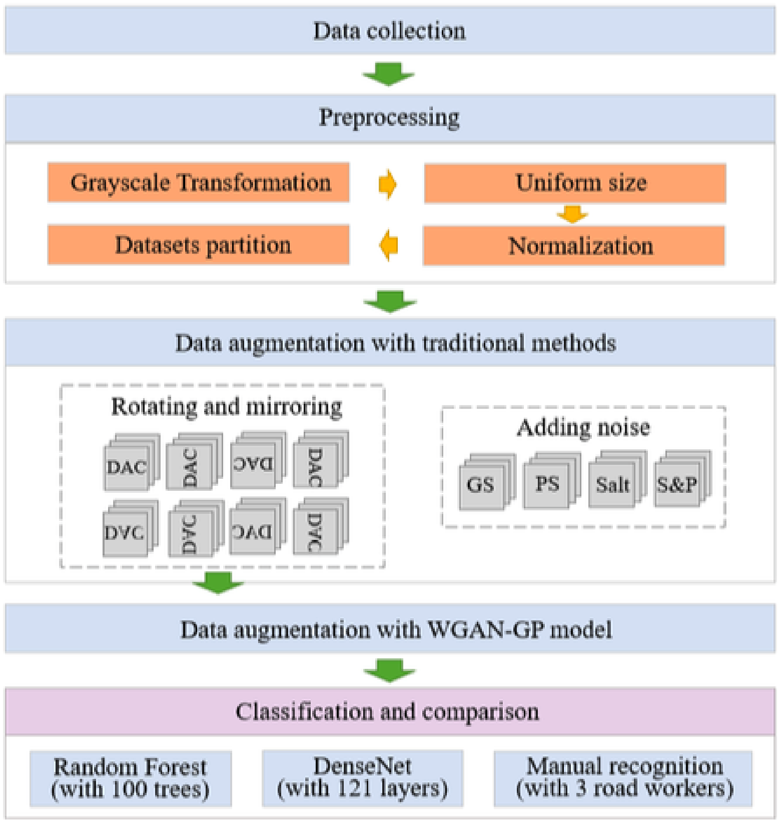

Data collection
Preprocessing
Grayscale Transformation
Uniform size
Datasets partition
Normalization
Data augmentation with traditional methods
Rotating and mirroring
DAC
DVC
Adding noise
GS
PS
Salt
S&P
Data augmentation with WGAN-GP model
Classification and comparison
Random Forest (with 100 trees)
DenseNet (with 121 layers)
Manual recognition (with 3 road workers)

Figure 2.10: Flow chart of the study of the paperwork. - [8]

The advanced WGAN-GP adopted is a network that can obtain high-quality images, but it has to be considered the limited dimension of the samples (80x80 pixels) in respect of other actual cases of study like the one in this thesis where samples have even up to 6 time the size in question. The improvement brought by this architecture is a new distance calculation for the divergence of the training domain, or, in other words, helps to avoid a mode collapse. The acronym GP stands for Gradient-Penalty, which represents a better manipulation of the layers of the network during the phase of back-propagation used to modify the weights of the Generator network. The WGAN model is shown in Fig. 2.11.

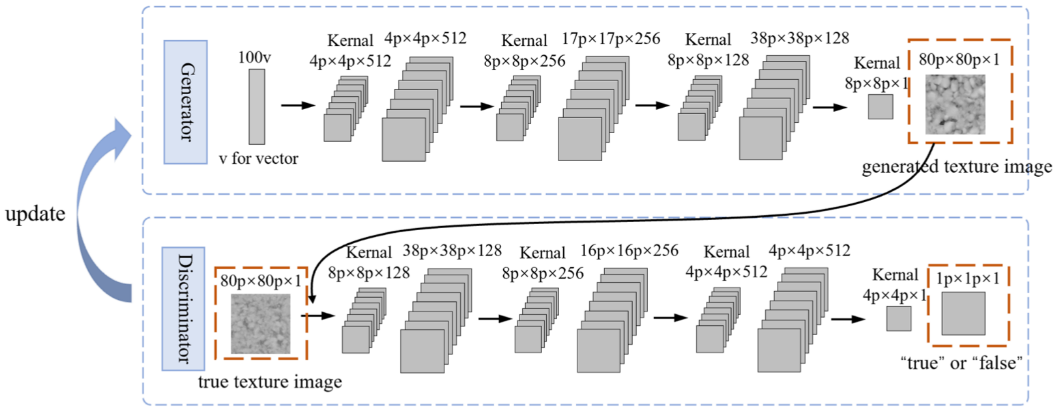


Figure 2.11: WGAN Architecture used for the work. - [8]

While for what concern the classifier adopted in Fig. 2.12 the model is showed.

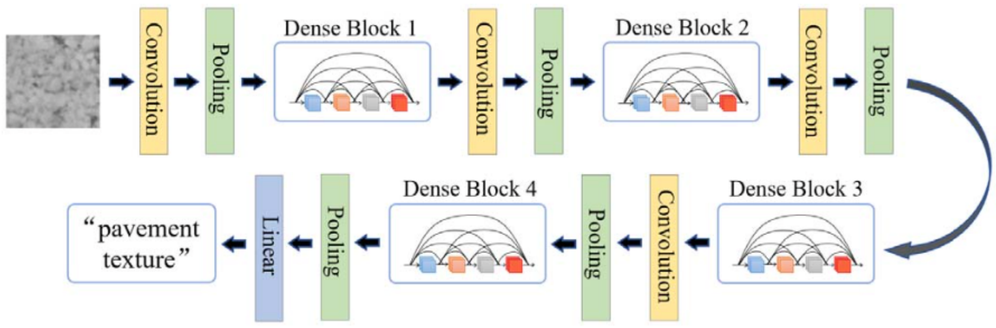


Figure 2.12: DenseNet Network used for the work. - [8]

Here's an example of created images (Fig. 2.13).

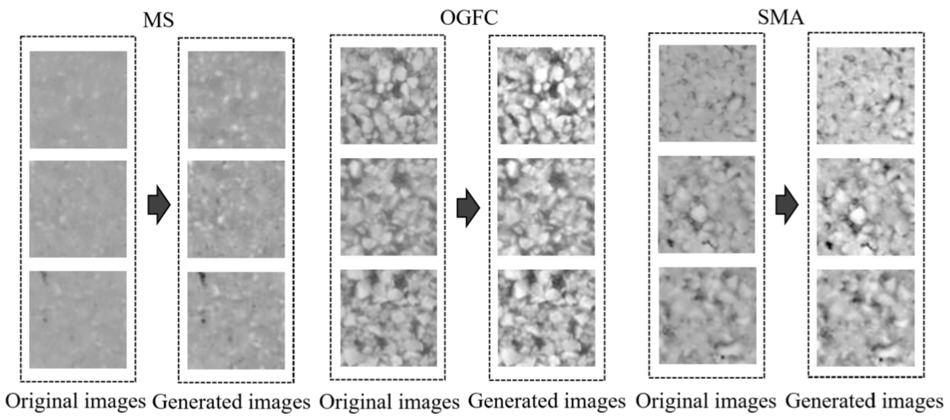


Figure 2.13: Overview of WGAN results for 3 different types of pavement textures. - [8]

STYLEGAN According to Dercilio, Gustavo Fardin et Al.[9], Microscopic wood identification plays a critical role in many economically important areas in wood science. Historically, producing and curating relevant and representative microscopic cross-section images of wood species is limited to highly experienced and trained anatomists. This manuscript demonstrates the feasibility of generating synthetic microscopic cross-sections of hardwood species. The proposed algorithm is named StyleGAN and can synthesize realistic, diverse, and meaningful high-resolution microscope cross-section images that are virtually indistinguishable from real images. Furthermore, the framework presented may be suitable for improving current deep learning models. Moreover, the StyleGAN includes the progressive increase of resolution by adding layers to the network, being able to work with images with a high dimensional space (512x512 pixels). Figure 2.14 shows the described network. Here's an example of created images (Fig. 2.15).

2.3.1 Use case of the thesis

As introduced in the initial part of this elaborate, this thesis will focus on the implementation of the first-ever introduced GAN by Goodfellow et Al. [21], focusing on the later metrics used to evaluate the quality and diversity of those images created, and in the future works, advanced architecture will be implemented to generate images even more efficiently.

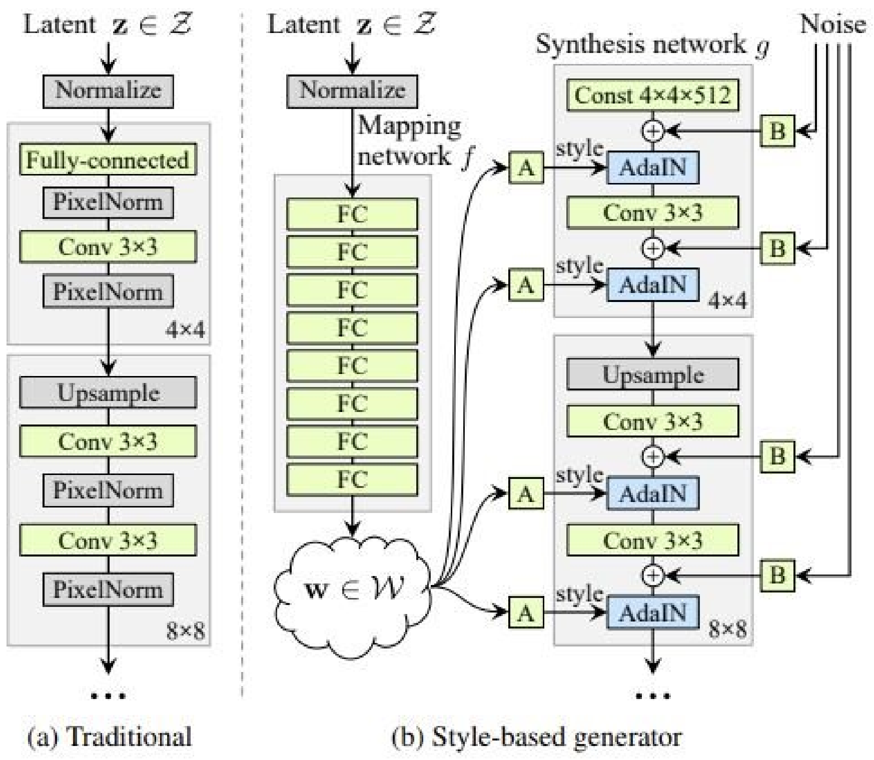


Figure 2.14: StyleGAN architecture used for the work. - [9]

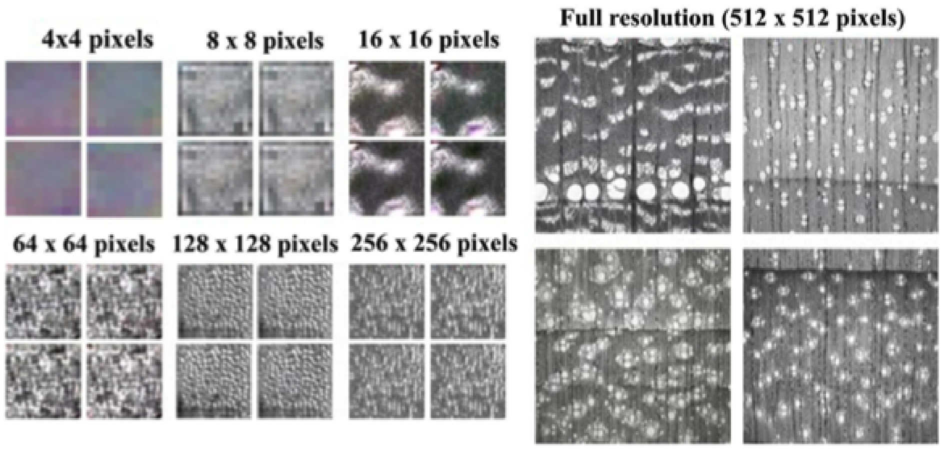


Figure 2.15: StyleGAN results using progressively increased image resolution. - [9]

Limits of RW Data Augmentation strategies One consideration must be made with respect to this particular case of study. Applications of Data Augmentation techniques involved in a complex scenario like the one of this thesis are still very limited. The gap is described by the fact that nowadays there are still too few cases of applications for complex classification tasks which require a strong Data Augmentation solution in order to obtain relatively efficacy images to improve the Classification performances.
This is due to the fact that actual methods are still too rough to reach perfect adaption for multiple cases of studies, where most of them differ so much in the image space features and details that have to be analyzed, making them difficult to be extended for different scenarios. In fact, regarding a Q.C. task like the one of this thesis, these techniques are still not very effective in order to generate a huge diversity of data samples, due to the random conformation of grains and veins and the high size of images so that it brings the network on overfitting, bringing consecutively limited classification improvements. Anyway, to justify the implementation importance of the D.A. strategies in this case study (Aesthetic quality control), there can be found several reasons:

- The dataset acquisition task is difficult and time-consuming. In fact, the annotation is a complex phase where the chosen label is a subjective choice, where each operator can interpret a specific image from a specific quality grade and so with the risk of assigning the labels badly. This brings to the easier task of Augmenting images in respect of the previous factors but will bring of course more engineering problems to define an appropriate architecture.

- The actual dataset is too variegated, where images are much different between each class and even between the samples of the same quality class sub-set of data. So, this brings the necessity of adopting D.A. strategies to obtain even more effective details.

- Resulting dataset obtained from the acquisition task is most of the times unbalanced, due to the production requests that the company receives, which would bring uncertain images from an unknown quality class that can make it even more unbalanced. Moreover, this randomness is also influenced by the raw tree used to produce those rifle stocks, which can have as well random veins and grains. Regarding the previous comment, this unbalanced dataset would bring inevitably bad results in these unbalanced cases, so the predictions are helped by the D.A. strategies which make balanced the dataset.

Chapter 3

Materials

3.1 Dataset use case

The commercial classification of wooden stocks is defined in five major categories ranging from grade 1 up to 5, where grade 1 indicates almost veinless wood and grade 5 a very twisted and variegated grain pattern. Each different type of shotgun series manufactured by the company is equipped with a stock belonging to a specific grade class. Today the Q.C. is mainly based on the evaluation of the human eye, by the analysis done by an expert operator properly trained.
The actual dataset refers to a real industrial case study of Benelli Armi Spa. The detention and conservation are regulated by an agreement between Benelli Armi Spa and Università Politecnica Delle Marche. The collected dataset is composed of both left and right side images belonging to 1060 different shotguns, for a total of 2120 images with a size of 270x470 pixels.

According to the aesthetic quality of wood, the stocks have been classified into 4 main grades (1, 2, 3, 4), (so the highest grade of quality isn't included in the actual amount of samples) and their relative minor grades ($2^-, 2^+, 3^-, 3^+, 4^-, 4^+$), resulting into 10 different classes as reported in the table 3.1. All the grades are reported in ascending order together with the number of stocks. The majority class is the grade 3^+ (343 stocks) while the minority class is the grade 4^+ (106 stocks). Figure 3.1 shows an example of stock for each of the 10 classes.

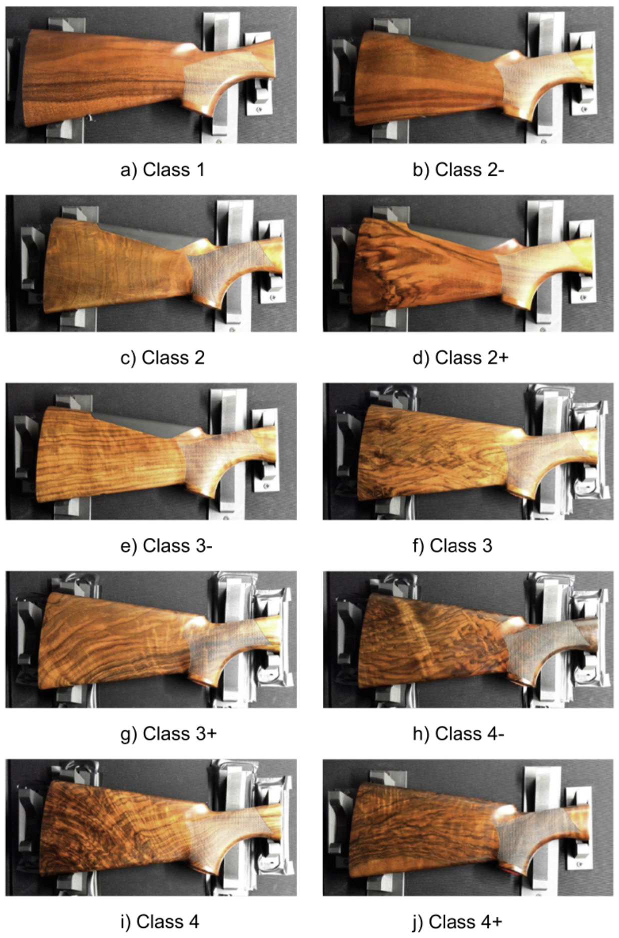


Figure 3.1: Example of different stock for each of the 10 classes.

3.2 Dataset Acquisition

The images were acquired with a high-definition RGB camera placed in the top-view configuration, and cropped into 270x470 (HxW). During the annotation procedure, a highly specialized technician accurately inspects the item and assigns the labels of the stock using a custom data annotation platform (see Figure 3.2).

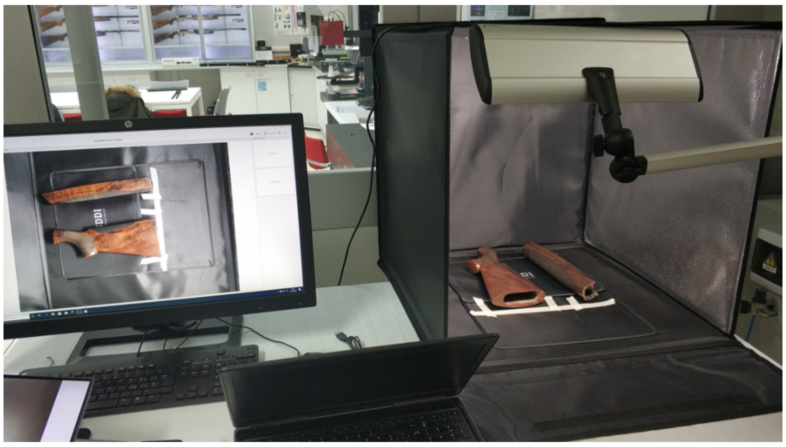


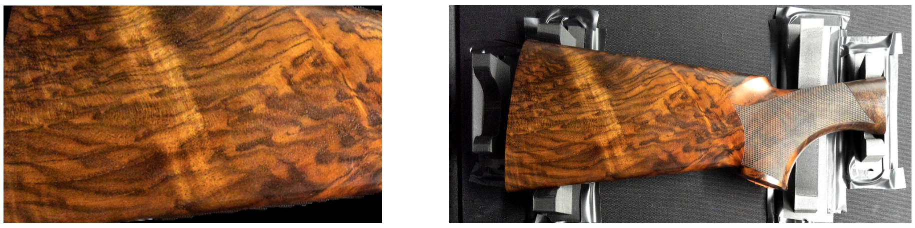
Figure 3.2: Custom data annotation platform. The stock is placed in the box where the RGB camera and an industrial lamp are mounted. The annotation software allows the operator to capture the image and record the grade.

As introduced in the Related work section, the lightning setting used to acquire the images uses a similar configuration to the one (b) of the Fig. 2.1. In table 3.1 are showed the cardinality of each Quality Class.

Label	**#Stocks**
1	165
2-	148
2	212
2+	177
3-	179
3	307
3+	343
4-	208
4	275
4+	106

Table 3.1: Cardinality of each Quality Class

This set of data will be manipulated by an Oversampling phase by the GAN described previously. Anyway, is it now important to focus on the effective configuration adopted in this thesis, in which all the dataset of images is cropped for this first thesis step, and translating the problem during future works even for the entire images. Below an example of the manipulated images is shown (Fig. 3.3)



(a) Cropped image rifle stock (b) Full image rifle stock

Figure 3.3: Configuration adopted (left) Vs Original image (right)

This choice has been done because of the huge difficulty for the GAN to generalize just only on the rifle veins and grains, avoiding the influence of the geometry features in the prediction results (considered bias as the scope of the neural network is to assign a Q.C. label due to only the conformation of the wood).

For the sake of completeness, in Table 3.2, it's shown the entire set of data available for this project, where also series membership details are illustrated, even if they're not useful considering the adoption of cropped images.

Table 3.2: Subdivision of quality classes images in respect to shotgun series production

Shotgun Series	**Cod**	**1**	**2-**	**2**	**2+**	**3-**	**3**	**3+**	**4-**	**4**	**4+**	**Tot**
ACCADEMIA-GR3+	1	0	0	0	0	0	24	85	9	2	0	120
RAFFAELLO-2013-GR3	2	0	0	0	11	130	9	0	0	0	0	150
RAFFAELLO-2013-GR2	3	0	32	149	26	14	3	0	0	0	0	224
ANNIVERSARY-50°-GR4	4	0	0	0	0	0	1	19	101	252	61	434
828-NIKEL-GR3+	5	1	0	0	0	11	176	14	0	0	0	202
828-CAL20-GR3+	6	0	0	1	0	2	35	70	12	2	0	122
MONTFELTRO-EU-GR1	7	151	19	18	4	7	1	0	0	0	0	200
MONTEFELTRO-CLARO-GR2	8	3	79	27	105	4	0	0	0	0	0	218
ARGO-E-GR4	9	0	0	0	0	0	0	0	8	2	18	28
FRANCHI-EU-GR2	10	10	18	17	31	0	0	0	0	0	0	76
ANNIVERSARY-50°-GR4	11	0	0	0	0	0	1	3	35	1	14	53
RAFFAELLO-LTD-GR4	12	0	0	0	0	0	0	7	43	11	13	74

Note that each shotgun series has a limited range of quality grade labels, because it comes from the same raw tree.

Chapter 4

Methods

To reach the goals which have been established as tasks for this thesis, 2 principal types of augmentation will be performed. One will be performed with an "Offline approach" by the GAN architecture to previously augment the number of images, while the second one, definable as "Online approach", is represented by the implementation of multiple combined image transformations provided by the CV science, which is a field that studies manipulations for the images.
In any case, both of the techniques are implemented to support the main classifier network, the scope of the thesis.

4.1 VGG16 architecture for Q.C. classification

We are interested to learn an agnostic model that classifies the 10 quality classes without being given information about the specific shotgun series. This is because, in the real-industrial case situation, the technician is engaged in the QC procedure where the information of the shotgun series is not known a priori. In addition, in this case study, the images adopted were the cropped version, so the shotgun series label was superfluous.
In this section, an inspection of the Classifier architecture used in this thesis is given. In particular, this network follows the previous work on the project to make a comparison between the 2 approaches after all the appropriate tests are completed. In figure (Fig.4.1) is illustrated the architecture now introduced.

4.1.1 Composition Layers

How it's visible, This network is composed of 13 convolutional layers that extract image features. Each convolutional block has filters with a 3×3 pixels receptive field and is

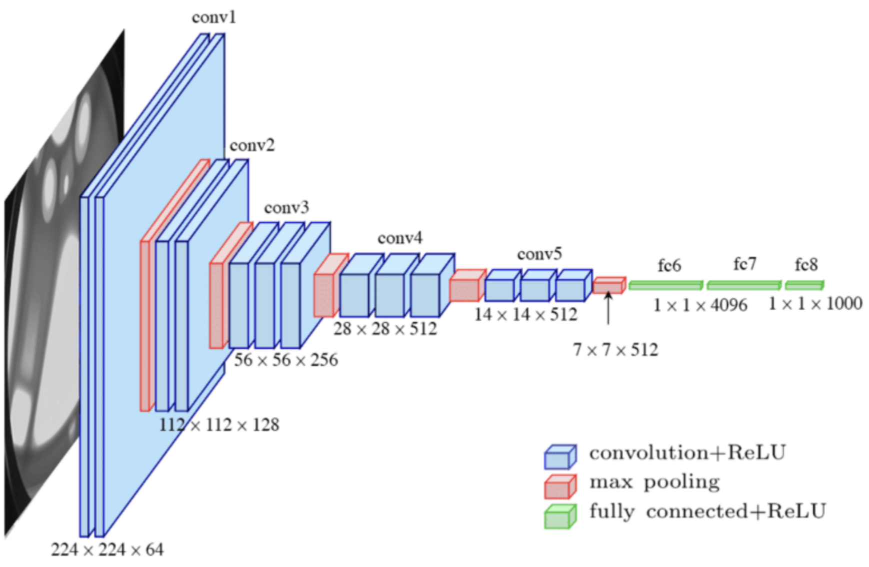


Figure 4.1: VGG 16 Architecture. - SOURCE

followed by a ReLU activation function (that will be described in the GAN architecture section 4.3). For CNN-parameters dimensionality reduction, max-pooling layers are used after 2 or 3 convolutional blocks, depending on the dept of the actual Convolutional Layer. These types of layers have the task of reducing the feature dimension for the network, or they would be too much to follow and handle. Going on, 3 fully-connected layers followed by a softmax layer are used to predict a probabilistic label map as the last blocks of the architecture. Dropout regularization layers were inserted after the first and second fully-connected layers with a rate of 0.3. As regards the fine-tuning, the first 4 convolutional blocks were frozen, meaning we prevent their weights from being modified during the training. This is done because usually, the first layers are almost similar in all the cases, (in fact, these layers represent basic CV transformations of the image to adapt the images of the network, such as crop or corners/border enhancing). So, doing this task, we let the network train faster avoiding changing weights for those layers where is useless to act the backward update procedure. Giving last details, as already sufficiently described, the fully connected layers are those layers where neurons take a decision with respect to the activation function given to each layer until obtaining the final values in the output layers, as shown in 1.1.

4.1.2 CE Loss Function

Regarding the classifier loss function, instead of adopting the BinaryCrossEntropy like the GAN architecture, the extended one is settled for this network. This is done because this work is a Multi-Class classification task, instead of a Binary-Class classification (i.e. it could be predicted if an image face is from a woman or a man).
This means that in respect of the binary version, this non-linear function should contain a factor that is able to manipulate the multi-class problem, which is afforded by a new exponential contribution, which replaces the mathematical terms that multiply the weights w_{y_n} with a single logarithmic term, but with a non-linear (e.g. exponential) contribution instead the linear arguments of the BCE logarithmic.
The mathematical formulation is 4.1:

$$l(x,y) = L = \{l_1, ..., l_{\mathrm{N}}\}^{\top}, l_{\mathrm{n}} = -w_{y_n} * log(\frac{\exp x_{n,y_n}}{\sum_{c=1}^{C} \exp x_{n,c}}) * 1(y_n \neq ignore\_index) \quad (4.1)$$

where x is the input tensor, y is the target tensor (not normalized), C is the number of classes, and N is the Batch size. Moreover, w_{y_n} are standard weights of the mathematical function applied to model the output calculation (they can be settled even by the programmer), and lastly x_n, y_n represent the 'n' element of the tensors x, y, which in turn can be both a single value or either another tensor.

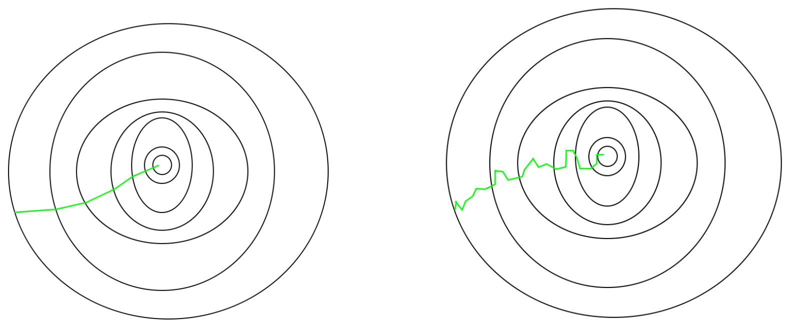
4.1.3 SGD - Optimizer

The optimizer adopted in this case is named SGD (Stochastic Gradient Descent). This one is good for this case study because its goodness comes from the replacement of the gradient calculation not from the entire dataset, but only from a small portion of it selected randomly. Moreover, in high-dimensional optimization problems like in this case, this algorithm reduces the very high computational complexity, achieving faster iterations in trade for a lower convergence rate.
Getting down to more technical terms, 'stochastic' means a system or process linked with a random probability, fetching the algorithm with random samples like said above. So this approach will use each input batch for the training to perform each iteration. The formulation is in 4.2, which represents the calculation for a single sample batch j:

$$for\_i\_in\_range(m) : \theta_j = \theta_j - \alpha(\hat{y}^i - y^i)x_j^i \quad (4.2)$$

where m is the number of batches, i, j batches counters, θ_j is the parameter to be estimated to minimize the solution, α is the learning rate, and $\hat{y}$ is an estimation derived from a modeling linear/non-linear function which fits the observed variables $x_1, ..., x_n$, in respect of the real values $y_1, ..., y_n$.

So, we find out the gradient of the cost function of a single example at each iteration instead of the sum of the gradient of the cost function of all the examples when the epoch is ended. For this reason, the path taken by the algorithm to reach the minima is usually noisier than typical Gradient Descent Algorithms, but this doesn't matter because the important aspect is to be able to reach the minima, and in this way, we explore the space with a better trend. Below a graphical interpretation is given for the typical Batch Gradient Descent algorithms and the SGD type, (Fig. 4.2).



(a) Batch Gradient Descent stock (b) Stochastic Gradient Descent

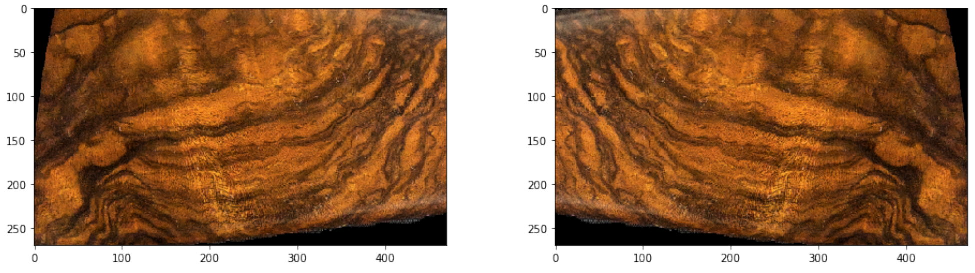
Figure 4.2: Difference from 2 of the most common types of Gradient Descent - Source

One thing to be noted is that, as SGD is noisier, it usually took a higher number of iterations to reach the minima, but it has the advantage of being more solid, consistent and much less computationally expensive.

4.2 Online Data Augmentation based on CV techniques

This type of Data augmentation is based on the creation of a Custom Dataset Class, where inside of it were recalled some transformation methods provided by the Pytorch library, which follow the CV subject. So, these transformations are essentially represented by different Kernels windows that automatically generate modification into the pixels space, obtaining the new version of the image, moving through the pixel space, and applying the internal weights applied in each cell to end up summing all the contributions. Below, the set of transformations used is given, but before taking into it, an important consideration has to be carried out. In fact, all the transformations chosen are easy transformations, because changing too much features inside the image would ruin the classification itself.

Horizontal Flip This transformation limits its action to rotating the image with respect to one of the vertical axis, as shown in Fig. 4.3.

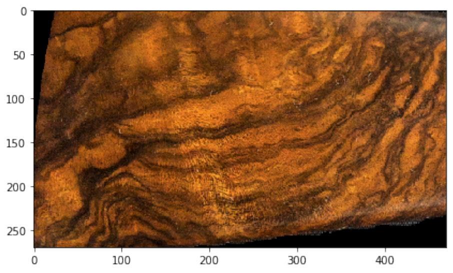


(a) Normal Image (b) Rotated Image

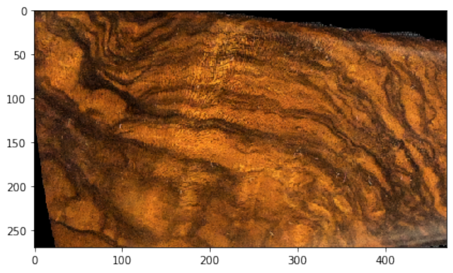
Figure 4.3: Configuration adopted (left) Vs Original image (right)

This guarantee better handling of the limited black space out of the effective geometry of the rifle stock, helping generalize better.

Vertical Flip This transformation has the same behaviour of the previous one introduced, with the difference in rotating the image respect one of the horizontal, as shown in Fig. 4.4:



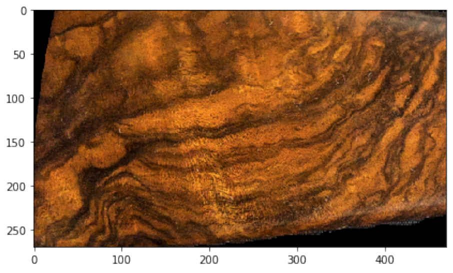
(a) Normal Image



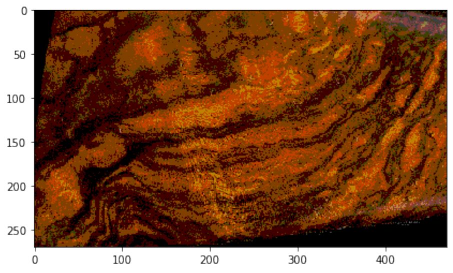
(b) Rotated Image

Figure 4.4: Configuration adopted (left) Vs Original image (right)

Posterize Differently from the other two, this one applies an effect similar to an older vintage poster. It has been limited the effect applied in order to change the colours without transforming too much the details of the normal image and guaranteeing as well the understanding as well as the others, as shown in Fig. 4.5:



(a) Normal Image



(b) Posterized Image

Figure 4.5: Configuration adopted (left) Vs Original image (right)

Brightness change The last transformation adopted is the brightness change, to act only on the saturation of the images, so we suggest the network not focus on the color feature, as is a useless factor for the final predictions. An example is illustrated in Fig. Fig. 4.5. This transformation is adopted in combination with the previous one (Posterize), to don't letting the posterize transformation change too much from the real images.

4.3 Offline Data Augmentation based on GAN

This approach is defined as an Offline approach in that its development takes place in a separate initial phase instead of acting directly during the Training phase of the classifier neural network. Essentially, it takes the same Input training set used to train the classifier, intending to generate independently from all the other executions new samples to concatenate in the same train-set before the classifier turn.

GAN architecture This architecture is basically composed of 2 principal architectures, named Generator (G) and Discriminator (D). Those 2 networks compete with each other trying to increase the ability of the production of the images. The Generator framework captures the data distribution in all the pixels image space, organizing features and adapting them to create new items with a contribution of noise (usually denoted by Z) to complicate its job and let it become more robust, while Discriminator estimates the probability that a sample came from the training data rather than Generator. The training procedure for G is to maximize the probability of D making a mistake.
In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation, which is the actual case of the use in this case, where the last layers of perceptrons are trained and upgraded each epoch of training. As a first step, the composition of the architecture of both the Generator and Discriminator is shown in Fig. 4.6:

In general terms, given a latent space z, with a prior distribution $p_{\mathrm{z}}(z)$, the Generator represents a differentiable function $G(z; \theta_{\mathrm{g}})$ which provides to output new data following a certain distribution p_{g}, where θ_{g} are the parameters of the generative model.
Discriminator represent instead a differentiable function $D(x; \theta_{\mathrm{d}})$, where θ_{d} are the parameters of the discriminative model, that produce to output the probability that x comes from the distribution of the training data p_{data}. The goal is to obtain a generator which is a good estimator of p_{data}. When this happens, discriminator is "fooled" and it is not able to distinguish anymore samples which come from p_{data} or p_{g}. To reach this situation, Discriminator is trained to maximize the probability to classify correctly samples from both training samples and generated ones. At the same time Generator is trained to minimize the following contribution $log(1 - D(G(z))$, and so maximizing the discriminator probability to consider generated data coming from p_{data}. Summarizing, training consists of optimizing a min-max game between 2 players D and G, 4.3:

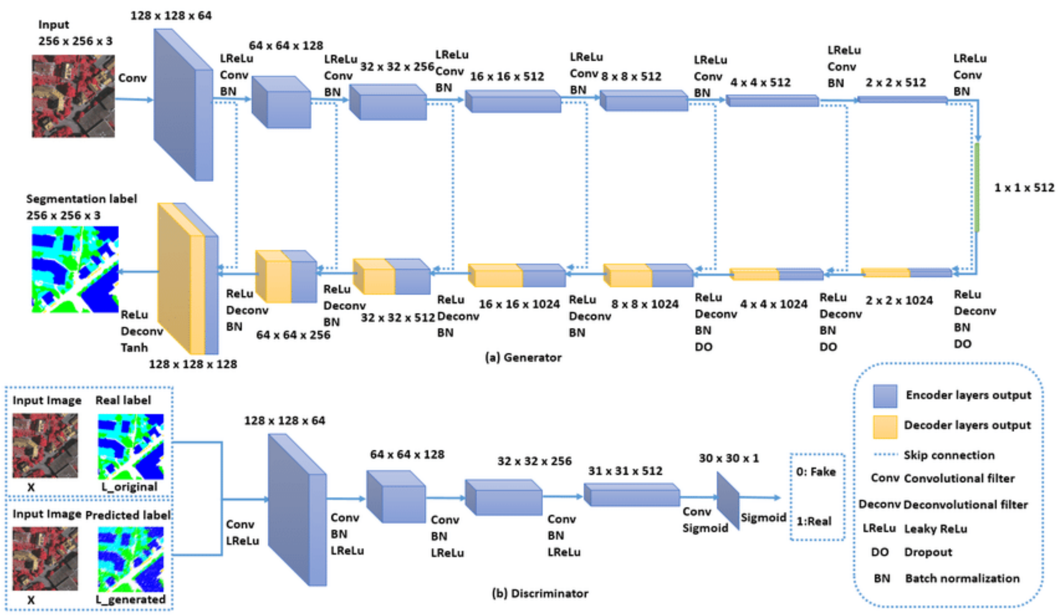


Figure 4.6: Complete GAN architecture with all the sequential layers. - [10]

$$\underset{G}{min}\underset{D}{max}\mathbb{E}_{x\sim p_{\text{data}}(x)}[log(D(x))] + \mathbb{E}_{z\sim p_{\text{z}}(z)}[log(1 - D(G(z)))] \tag{4.3}$$

where $\mathbb{E}_{x\sim p_{\text{data}}(x)}$ is the Ensemble of distributions which tend to be as the normal distribution of the real data.

4.3.1 Generator

Composition Layers As shown in the previous illustration, the Generator is composed essentially of multiple layers of Encoders and Decoders as well, which together form a structure similar to an Autoencoder network. The function of these Autoencoders is to observe the input data and subsequently to elaborate an efficient representation of those reconstructing outputs as the ones in input. To reach this result, the Autoencoders are composed of 2 parts:

- a Recognition Network, named Encoder, that converts inputs into internal representations.
- a Generative Network, named Decoder, that converts the input representation obtained by the other into an effective output.

Note that the action of the Encoder is often influenced by the Programmer adding some random noise to the input images, in order to solidify the capacity of the entire architecture. So, the principal difference is that even adopting a similar strategy with respect to the Autoencoders, the advantage of the Generator is that will reach generations of images with different details, guaranteeing the creation of unique samples. In more technical terms, it works in different phases, such is visible in the architecture illustration:

1. Forward propagation, consists of the features extraction from the input images, by a sequence of a convolutional filter, used to group the features of the input data (necessary to manipulate the entire set of data grouping into smaller kernels of information for the network, through chosen weights settled for each cell of the kernel), followed by an activation layer (i.e. LRelu), which essentially perform a non-linear mathematical function on the convoluted layers from the previous step to clean and manipulate the extracted values. These steps are repeated 7 times, where the convolutional layers halve each time the output layers, but augmenting at the same time the number of output channels to be manipulated by the Non-Linear function activator.

2. Backward propagation, consisting as well of a sequence of Decoders phases to do the reverse work, to reconstruct original image sizes but separated between each layer by the same activation function. Moreover, this phase consists also of another layer in the chain represented by the Dropout. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged. Note that the 'rate' parameter has been taken as the default one by the base architecture. Also, this procedure is repeated with 7 levels.

In particular, the Non-Linear activation function used in the Forward propagation is named LRelu, which its formulation is 4.4:

$$\text{LRelu}(x) = max(0, x) + \beta * min(0, x) \tag{4.4}$$

where

$$\begin{cases} x, & \text{if } x \geq 0 \\ \beta * x, & \text{otherwise} \end{cases} \tag{4.5}$$

The β factor controls the angle of the negative slope, an example in Fig. 4.7

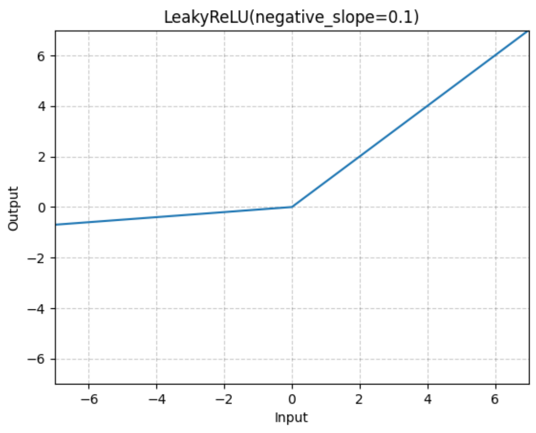


Figure 4.7: Example of LRelu with a negative slope factor $\beta = 0.1$. - SOURCE

For what concerns the Backward propagation the Relu activation function is used instead of the LRelu, where the only difference is to avoid directly negative arguments in the reconstruction image phase, avoiding the negative slope handling, which its formulation is 4.6:

$$\text{Relu}(x) = max(0, x) \tag{4.6}$$

An important note is the adoption of another activating function, for the last activation layer before obtaining back the image back (under form of a tensor) named Tanh, which follow this new formulation 4.7:

$$\text{Tanh}(x) = tanh(x) = \frac{\exp x - \exp x}{\exp x + \exp x} \tag{4.7}$$

For sake of completeness here's is the representation of the known tanh function, Fig. 4.8

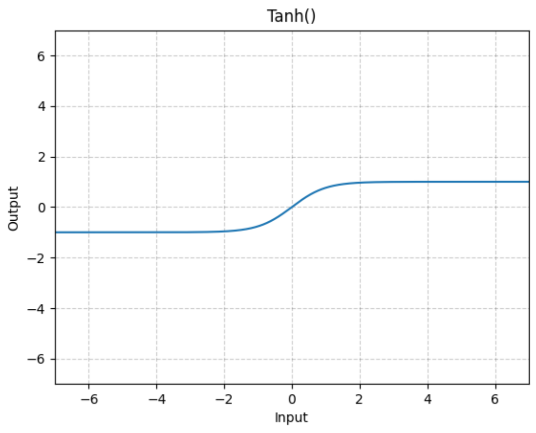


Figure 4.8: Tanh function illustration. - SOURCE

Now, it's more evident how this alternative works better for the last layer of the GAN, so it's a good improvement, with help of Normalize each factor in all of the activation layers, maintaining a value in each the direction, for both positive and negative values a normalization in the range $[0, 1]$.

On the other hand about the Convolution Layer sizes, they're always with a dimension of:

$$[(W/2) + 1] * [(H/2) + 1]. \tag{4.8}$$

Finally, the other parameters are taken as the original GAN formatting provides, as well as the weights on the Convolutional kernels.

BCE Loss Function Taking into account of the loss function, is fundamental in each Neural Network architecture because it measures the accuracy grade within a certain statistical model and describes a set of empiric data deriving from a specific phenomenon. In more technical terms, regarding the classification tasks on DL, since we obtain a probability value that a given sample belongs to a specific class (i.e. in this experiment for 1 of the 10s quality classes), we would like to increase as much as possible the accuracy of the model.

So, these loss functions have the task of maximizing these probabilities during the training phase. This concept also applies to the Generator and Discriminator network as both have to maximize their probabilities of creating real images and distinguish if they are real or not respectively. It's this function that will be used in the Back-propagation on the weights updates.

Describing the function in a mathematical way, the loss can be described as 4.9:

$$l(x,y) = L = \{l_1, ..., l_{\mathrm{N}}\}^\top, l_{\mathrm{n}} = -w_{\mathrm{n}}[y_{\mathrm{n}} * log(x_{\mathrm{n}}) + (1 - y_{\mathrm{n}}) * log(1 - x_{\mathrm{n}})] \tag{4.9}$$

where x is the input tensor, y is the target tensor, and x_n, y_n are the 'n' element of the tensor. If the reduction parameter is settled:

$$l(x,y) = \begin{cases} mean(L), & \text{if reduction = 'mean'} \\ sum(L), & \text{if reduction = 'sum'} \end{cases} \tag{4.10}$$

This is used for measuring the error of a reconstruction (i.e. for Auto-encoders like in this case).
An important note has to be given regarding the *log* contributions, in which with some particular parameters the limit would tend to infinity, 4.11:

$$\begin{cases} \lim_{x_{\mathrm{n}} \to 0} log(x_{\mathrm{n}}) = -\infty, & x_{\mathrm{n}} = 0 \\ \lim_{x_{\mathrm{n}} \to 1} log(1 - x_{\mathrm{n}}) = -\infty, & x_{\mathrm{n}} = 1 \end{cases} \tag{4.11}$$

Moreover, if we have an infinite loss value, then we would also have an infinite term in the gradient calculation of it, so this would make the BCELoss's backward method non-linear with respect to x_{n}, and using it for things like linear regression (i.e. for some particular cases of GAN applications), would not be straight-forward. The solution is that BCELoss clamps its log function outputs to have an inferior limit equal to -100, in order to always guarantee linear backward propagation.

Adam Optimizer The optimization algorithm is the main approach used today for training a machine learning model to minimize its error rate. There are 2 metrics to determine the efficacy of an optimizer:

- speed of convergence, which is the process of reaching a global optimum for gradient descent;
- generalization, which represents the model performance on new data

It contains various parameters which will be analyzed in the next chapter. These algorithms help the loss functions on finding the best weights that minimize the cost function and that maximize the probabilities performed at the same time. In this particular case of study, the best-fitted Optimizer is Adam, which maintains better performances than the others in accuracy. In technical terms, its advantages are that it stores not only an exponentially decaying average of past squared gradients v_t, but also an exponentially decaying average of past gradients m_t, similarly to momentum behavior. The formulation is the following:

$$\begin{cases} m_t = \beta_1 * m_{t-1} + (1 - \beta_1) g_t \\ v_t = \beta_2 * v_{t-1} + (1 - \beta_2) g_t^2 \end{cases} \tag{4.12}$$

where β_1, β_2 are the betas parameters of the Optimizer, v_t are the squared gradients, m_t are the past gradients and g_t are the current gradients. These contributions $(v_t, m_t)$are estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients respectively. This optimizer comes from the same family of SGD optimizers, (see the explanation in 4.1.3).

4.3.2 Discriminator

Composition Layers As shown in Fig. 4.6, as opposed to the generator, it has only a Forward propagation which functions similarly to the generator one. The only difference is the adoption of a different size of Convolutional Kernels from the 4th step of Encoded layers, where the new dimensions will be worth (2*2). Another difference is described by the usage of another kind of activation function in the last 2 Encoded layers, named Sigmoid 4.13:

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + \exp -x} \tag{4.13}$$

where the graphical representation is 4.9:

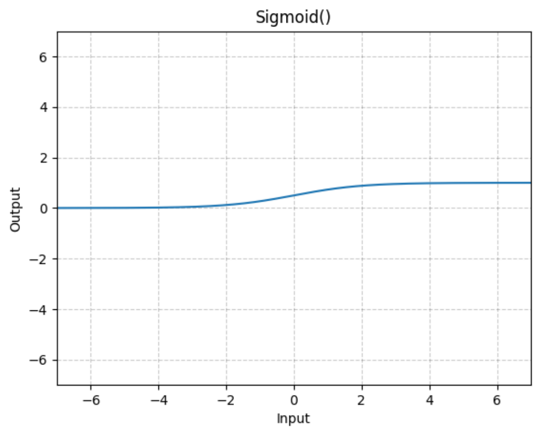


Figure 4.9: Sigmoid function illustration. - SOURCE

BCE Loss Function The same loss function is taken for this Network because the goal of the Discriminator is to maximize its binary probability to understand the reality or fiction of each of the created images.

Adam Optimizer As for what has been said regarding the loss function, the Optimizer concepts are still worth it for the Discriminator.

Chapter 5

Experimental procedure

Procedure steps A schematic representation of all the steps is shown for both the ON.-D.A.-C.V.T. in the figure (Fig. 5.1) and for OFF.-D.A.-GAN in the figure (Fig. 5.2).
It can be observed that the difference in the approaches is to apply the GAN as the first step after the pre-processing phase of the dataset in the approach OFF.-D.A.-GAN, to add a certain amount of images to the dataset, and then repeat all the same steps like the first approach.

5.1 General Training Settings

For all the experiments done, the tests performed considered images resized to 270x470 pixels (HxW). This choice was taken in order to maintain the same configuration adopted by the previous work in 2.1. Another reason that validates the choice maintaining these sizes was that original images were captured with this resolution. So, even trying to augment the sizes, would lead to a decrease in the resolution.
Moreover, for this case of study the images employed were the cropped format (3.3 - (a)), in order to focus on the generation of the grains and veins without taking care of the geometrical features that would bring bias to both the images generation and classification results. The dataset was split by a stratified hold-out procedure, i.e. using 60% of the images as training, 20% as validation, 20% as testing. Images belonging to the same rifle ID have been kept in the same set.

All the experiments were performed using Torch $1.11.0 + cu113$ and torchvision $0.12.0 + cu113$ on Google Colab with High GPU & High RAM.

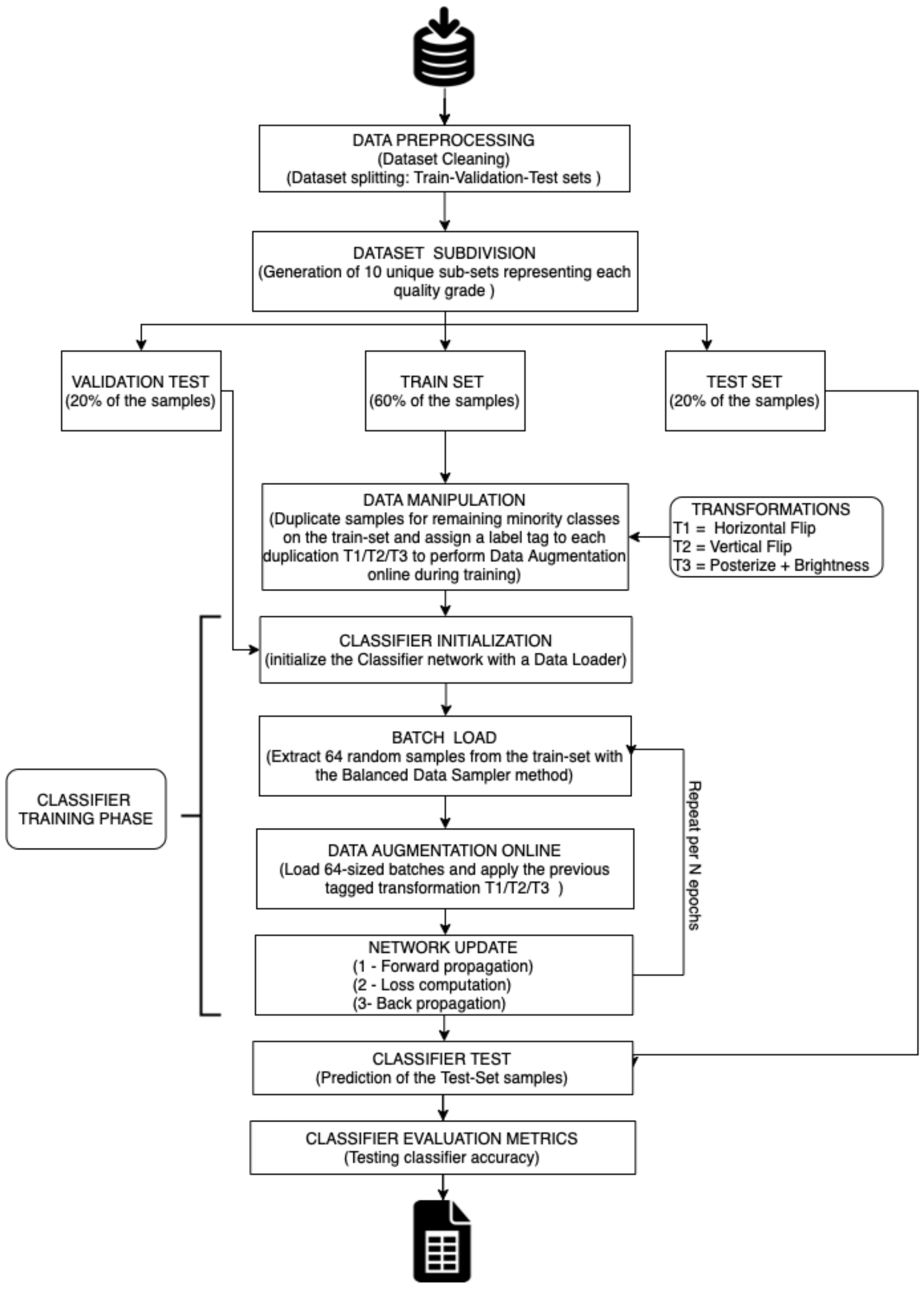

DATA PREPROCESSING
(Dataset Cleaning)
(Dataset splitting: Train-Validation-Test sets)
DATASET SUBDIVISION
(Generation of 10 unique sub-sets representing each quality grade)
VALIDATION TEST
(20% of the samples)
TRAIN SET
(60% of the samples)
TEST SET
(20% of the samples)
DATA MANIPULATION
(Duplicate samples for remaining minority classes on the train-set and assign a label tag to each duplication T1/T2/T3 to perform Data Augmentation online during training)
TRANSFORMATIONS
T1 = Horizontal Flip
T2 = Vertical Flip
T3 = Posterize + Brightness
CLASSIFIER INITIALIZATION
(initialize the Classifier network with a Data Loader)
CLASSIFIER TRAINING PHASE
BATCH LOAD
(Extract 64 random samples from the train-set with the Balanced Data Sampler method)
Repeat per N epochs
DATA AUGMENTATION ONLINE
(Load 64-sized batches and apply the previous tagged transformation T1/T2/T3)
NETWORK UPDATE
(1 - Forward propagation)
(2 - Loss computation)
(3- Back propagation)
CLASSIFIER TEST
(Prediction of the Test-Set samples)
CLASSIFIER EVALUATION METRICS
(Testing classifier accuracy)

Figure 5.1: Experimental procedure ON.-D.A.-C.V.T.

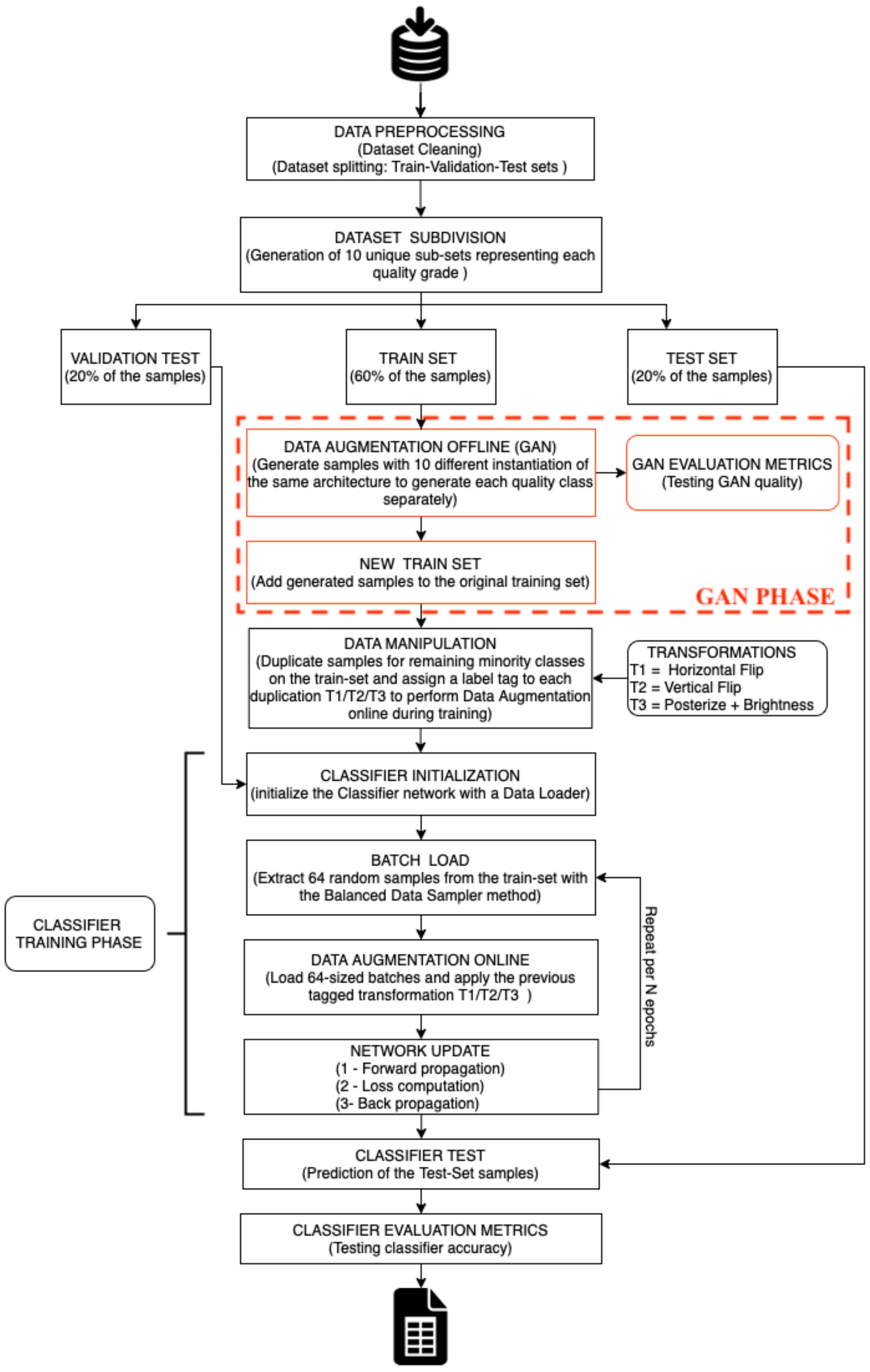

DATA PREPROCESSING
(Dataset Cleaning)
(Dataset splitting: Train-Validation-Test sets)
DATASET SUBDIVISION
(Generation of 10 unique sub-sets representing each quality grade)
VALIDATION TEST
(20% of the samples)
TRAIN SET
(60% of the samples)
TEST SET
(20% of the samples)
DATA AUGMENTATION OFFLINE (GAN)
(Generate samples with 10 different instantiation of the same architecture to generate each quality class separately)
GAN EVALUATION METRICS
(Testing GAN quality)
NEW TRAIN SET
(Add generated samples to the original training set)
GAN PHASE
DATA MANIPULATION
(Duplicate samples for remaining minority classes on the train-set and assign a label tag to each duplication T1/T2/T3 to perform Data Augmentation online during training)
TRANSFORMATIONS
T1 = Horizontal Flip
T2 = Vertical Flip
T3 = Posterize + Brightness
CLASSIFIER INITIALIZATION
(initialize the Classifier network with a Data Loader)
CLASSIFIER TRAINING PHASE
BATCH LOAD
(Extract 64 random samples from the train-set with the Balanced Data Sampler method)
DATA AUGMENTATION ONLINE
(Load 64-sized batches and apply the previous tagged transformation T1/T2/T3)
NETWORK UPDATE
(1 - Forward propagation)
(2 - Loss computation)
(3- Back propagation)
Repeat per N epochs
CLASSIFIER TEST
(Prediction of the Test-Set samples)
CLASSIFIER EVALUATION METRICS
(Testing classifier accuracy)

Figure 5.2: Experimental procedure OFF.-D.A.-GAN

5.1.1 Frameworks and Libraries

Pytorch This framework for automatized learning is open source and based on the Torch library, used for applications such as A.I., N.L.P, and M.L. . It contains many CV manipulations methods, but also optimizers, loss functions, and all the other base methods needed in this thesis.

Sklearn This framework contains other useful methods, which in this case study were represented by classification metrics for the Q.C. classification task, and confusion matrix methods.

Numpy The Framework which offers mathematical functions to manipulate vectors. In this work was necessary to face up the usage of all the tensors easily before being converted for Network usage.

5.2 D.A. online procedure

5.2.1 Custom Dataset with CV transform procedure

In order to obtain even more diversity in the training set samples, some transformations are applied to some of the input images. Considering that the GAN wasn't able to generate too many samples to reach the majority class amount, due also to the fact of a limited number of images available for the GAN, this strategy was implemented to accompany the GAN limits, so that we can complete the minority classes, reaching the needed amount for each sub-class to 206 images. Essentially, considering that Pytorch still doesn't allows augmenting data "On the fly", the "smart strategy" adopted follows a first offline phase of augmentation of data. So, we can define it as "smart" as simples subtractions were performed with respect to the maximum amount of images to be reached (i.e. $number\_images\_left_{(cl.2)} = 206 - 152 = 54$, supposing we had created with the GAN 20 images for quality grade 2) for each sub-set of classes, and then randomly duplicate the left amount previously obtained, by even applying an additional label. Also for this additional label, a random choice was executed between the 3 main types of transformations introduced in the section 4.2.
The final dataset obtained in this way will be used by the classifier during the training phase. It's important to note that, this is called an *Online Augmentation* because the effective transformations to the images will be performed during the training phase. In fact, the logic will be to extract the additional label applied to each sample of the dataset,

and depending on that specific assigned value, the corresponding transformation will be executed, giving then back the entire manipulated batch in a tensor form.
Moreover, we can define this approach as "smart", as this logic is even adaptable to other study cases, even waiting firstly for the generation of a certain amount of images from any GAN architecture.

5.3 D.A. offline - GAN training procedure

Due to the small dimension of the dataset, data augmentation was performed, to obtain a larger amount of images to feed the network classifier with more information. In order to accomplish this task, the strategy adopted was to apply the same GAN architecture to all the 10 Q.C. separately. So, the first step performed was to subtract from the entire training dataset 10 sub-sets, each one composed of all the samples within a specific label inherent to a unique quality class.
The starting training set was so composed 5.1. Even having an unbalanced situation between all the sets, in this work, tests have been performed giving to the GAN architecture only 64 images, because with this actual architecture the number of epochs to reach quite good results in terms of quality was 2500, too much for the Colab resources available to afford a problem with the image resolution of 270x470, as will be described in Chapter 6.1.

Sub-Set	**N° of images**
1	95
2-	83
2	132
2+	110
3-	111
3	179
3+	206
4-	125
4	171
4+	62

Table 5.1: Cardinality of each Sub-Set

5.3.1 Hyperparameters setting

For what concern hyperparameters, in this work were performed different combinations such as batch-size in $\{32, 64, 128\}$, latent space in $\{50, 80, 100, 200, 500\}$, epochs number in $\{200, 500, 1000, 2000, 2500, 3500, 4000\}$, beta1 in $\{0.4, 0.5\}$, beta2 in $\{0.9, 0.999\}$ and learning rate in $\{2 \cdot 10^{-4}, 2 \cdot 10^{-5}\}$. Instead, talking about optimizers, Adam, SGD and RMSprop were tested, but all of them after having found the best hyperparameters configuration.

5.4 Q.C. classification task - VGG16 training procedure

This architecture does nothing more than taking the training set modified by the Data augmentation strategies implemented before, and using it to train the classifier network. The only detail to point out refers to the usage of the Data Loader, which will load a batch of images when needed and then discard it from the memory to allow the storage of new samples time after time. To improve even more the Generalization and so the quality of the training phase, a Balanced Data Sampler was implemented, with the scope of randomly selecting in the actual batch the same random amount of images from each quality grade.
At the end of the training, classification metrics are executed in order to evaluate the complete process composed of the principal parts of this thesis.

5.4.1 Hyperparameters setting

Taking account of hyperparameters, the number of epochs was set at maximum in the range of $\{30, 50\}$, but the training was anyway managed by a Custom Early Stopping Class, which does nothing more than monitoring the validation accuracy during each epoch of training, with the patience settled to $\{10\}$. So, if the metric isn't able to improve itself for that amount of consecutive epochs, then the training is stopped, moving to the prediction phase. Then, due to the Balanced Data Sampler implementation, the batch was taken like a multiple of 10, choosing $\{60\}$.
For what concern the remaining Hyperparameters, they were fixed for all the tests performed, because the focus was to visualize, not only the best configuration for the VGG16 classifier, but to compare all the tests executed, so "leaving the stage" to the Augmentation contribution. Summing them up we have learning rate as $\{1 \cdot 10^{-3}\}$, momentum $\{0.8\}$ and weight_decay as $\{1 \cdot 10{-}5\}$.

5.4.2 Data Loader

The objective of this class is to reduce the number of Hardware resources employed to execute the model, in order to work with datasets that have a bigger quantity of images. So, working with chosen-sized batches we load the images on the fly and discard them after the epoch training is complete.
This led to the implementation of a Data Loader that allows working as described before. In detail, it has the assigned task of generating batches as an input for the training process, converting them into tensors of images. After this step is completed, data will be discarded from the memory, repeating the same steps for all the batches needed in each training epoch. In the next figure, 5.3 an example is shown:

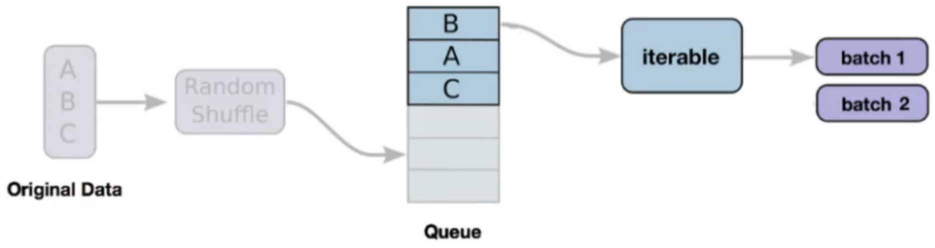


Figure 5.3: Data Loader. - SOURCE

5.4.3 Balanced Data Sampler

These Custom class essentially takes 2 inputs. The first is the tensor with all the samples ordered, and the other is their ordered tensor of quality grade labels. Subsequently, during each epoch, it randomly select $batch\_size/num\_classes = 60/10 = 6$ images from each class. It does so, creating a temporary index numpy vector of all the image indices, and then removing the 60 images chosen from it, guaranteeing to give each batch 60 different and new images.

5.5 Evaluation Metrics

In this section, all the implemented metrics of the thesis are presented, describing briefly the mathematical formula and the meaning of all of them.

5.5.1 Classification Metrics

The classification performances were principally evaluated according to accuracy 5.1, macro-precision 5.2, macro-recall 5.3 and macro-F1 score 5.4. These are the typical metrics performed on classification tasks, which use the prediction results obtained from the Confusion Matrix to return a single value:

$$Accuracy = \frac{T_{Pos} + T_{Neg}}{T_{Pos} + T_{Neg} + F_{Pos} + F_{Neg}} \tag{5.1}$$

$$Precision = \frac{T_{Pos}}{T_{Pos} + F_{Pos}} \tag{5.2}$$

$$Recall = \frac{T_{Pos}}{T_{Pos} + F_{Neg}} \tag{5.3}$$

$$Macro - F1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} \tag{5.4}$$

The principal goal will be to increase the final Accuracy score, which represents the effective skill of the program in correctly detecting the quality classes.

5.5.2 Ordinal Metrics

Other few more classification metrics were implemented, more general than those shown previously. These are based on statistical methods to analyze ordinal regression and the correlation between interesting variables, which in these cases are true labels and predicted labels.

i. **Quadratic Weight Kappa (QWK)** - Cohen's Kappa is a statistical coefficient that represents the degree of accuracy and reliability in a statistical classification; it is a concordance index that takes into account the probability of random concordance; the index calculated from the ratio of the agreement above the probability of random agreement to the maximum obtainable excess. Through the confusion matrix, you can evaluate this parameter. There are several "degrees of concordance", according to which we can define whether Cohen's Kappa is poor or excellent. Quadratic

Weighted Kappa score is a ratio that can take a value between -1 and 1. A negative QWK score implies that the model is "worse than random". A random model should give a score of close to 0. Lastly, perfect predictions will yield a score of 1. The mathematical formula is 5.5:

$$QWK = 1 - \frac{\sum_{i,j} w_{i,j} * O_{i,j}}{\sum_{i,j} w_{i,j} * E_{i,j}} \tag{5.5}$$

where w is the weighted matrix, O is the histogram matrix and E is the expected matrix.

In particular:

$$w_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{5.6}$$

where NxN is the size of the histogram matrix O (e.g. Confusion Matrix).

ii. **Minimum Sensitivity (MS)** - is a statistical measure of the performance of a binary classification test widely used: Sensitivity (True Positive Rate), measures the proportion of positives that are correctly identified (i.e. the proportion of those who have some condition (affected) who are correctly identified as having the condition). Being this a multi-class classification problem, the final result will be an average between 10 values performed considering each of the 10 times one different class, and using it against all the others. The formula is 5.7:

$$QWK = 1 - \frac{\sum_{i,j} w_{i,j} * O_{i,j}}{\sum_{i,j} w_{i,j} * E_{i,j}} \tag{5.7}$$

Note that, is the same formulation of the Recall metric, in fact, the difference is to use middle statistic operations to round better the output value.

iii. **Mean Absolute Error (MAE)** - In statistics, is a measure of errors between paired observations expressing the same phenomenon. In this thesis, the example of Y versus X represent the comparison between predictions and observed. The mathematical formulation is so represented as 5.8:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{5.8}$$

Now that all the procedures have been described, the comparison results are given in the following chapter.

5.5.3 GAN metrics

GAN metrics were implemented to test the efficacy of the architecture implemented, to obtain a reference value to store, useful even for having comparison values for future improvements or even new architectures.
For each set of created images, all of them were tested. In particular, two steps are performed, in order to get a reference value to compare with the created image results. In fact, all the metrics require input of 2 different folders of images, and then comparing their content in order to evaluate the metrics. So the approach was to first use the metrics giving the real images folder for both of the 2 parameters of the methods and then repeat the experiment passing the 2 different folders of created and real ones.
In this way, the reference value can tell us how much the generated images diverge from the real ones in terms of quality and diversity. Other metrics only take one input folder, so the strategy will be directly comparing results from real images and generated ones.

Image Quality Metrics

Image quality metrics are those used to test the effective quality, as the name suggests, with respect to a reference set of images. In this case, the generated images are compared to the reals in order to obtain a measure of the GAN efficacy.

FID FID, otherwise known as Fréchet Inception Distance, is a performance metric to evaluate the similarity (in terms of quality) between two datasets of images. It was introduced by [22]. It is shown to correlate well to the human evaluation of image quality, and it can detect intra-class mode collapse. Rather than directly comparing images pixel by pixel, FID compares the mean and standard deviation of one of the deeper layers on a CNN named Inception v3.

- A low FID score means high quality.

The mathematical representation is in 5.9:

$$FID = \| \mu - \mu_w \|_2^2 + tr(\sum + \sum_w -2(\sum^{\frac{1}{2}} * \sum_w * \sum^{\frac{1}{2}})^{\frac{1}{2}}) \tag{5.9}$$

In technical terms, FID is the squared Wasserstein metric between two multidimensional Gaussian distributions:
$\mathcal{N}(\mu, \Sigma)$, the distribution of some neural network internal representations (activations)

of the images generated by the model and $\mathcal{N}(\mu_w, \Sigma_w)$ the distribution of the same neural network activations from the real images used to train the model.

IS A previous version of the FID metric, which differently, evaluates only the distribution of generated images. As well as FID, it uses a pre-trained Inception v3 network.

- An high IS score means high-quality images.

The metric was introduced by [23], and the formulation is in 5.10

$$IS(G) = exp(\mathbb{E}_{x \sim p_\mathrm{g}} D_{KL}(p(y \mid x) \parallel p(y))) \tag{5.10}$$

where D_KL is a Divergence metric that performs the divergence between those 2 probabilities.

Image Diversity Metrics

These metrics have similar behavior to the others but are used in particular to emphasize the diversity of the input arguments.

SSIM SSIM, otherwise known as *Structural Similarity*, is a quite good diversity metric. This index provides a measure of the similarity by comparing two images based on luminance similarity, contrast similarity and structural similarity information). In more technical terms is a perception-based model that considers image degradation as a perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms.

- A low SSIM score means high diversity, but if it reaches too low values, it could mean that images have very low quality.

The formulation is in 5.11

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.11}$$

where x, y are two measures of two images windows of common size NxN, with:

- μ_x is the average of x
- μ_y is the average of y
- σ_x^2 is the variance of x

- σ_y^2 is the variance of x
- σ_{xy} is the covariance of x, y
- $c1, c2$ are two variables to stabilize the division

The entire formulation collects all the contributions of luminance, contrast and structure.

LPIPS Acronym of a metric defined *Learned Perceptual Image Patch Similarity*, is useful to judge the perceptual similarity between two images. LPIPS essentially computes the similarity between the activations of two image patches for some pre-defined networks. This measure has been shown to match human perception well.

- A low LPIPS score means that image patches are perceptually similar.

This metric is provided by the torch library.

Chapter 6

Results

All the results obtained are summarized below as follows:

1. Experimental comparisons, where Classification results are shown in table 6.2 after having given the best Hyperparameters configuration as the first step.
2. ON.-D.A.-C.V.T and OFF.-D.A.-GAN qualitative results, showing some examples of the generated images.
3. Computation Effort analysis, where observations will be made considering the Computational time needed for the experiments.

6.1 Experimental Comparisons

6.1.1 VGG16 Classifier - Best Hyperparameters Configuration

The hyperparameters settings are summarized below in table 6.1

Table 6.1: Best VGG16 hyperparameters setting

Optimizer	**Batch_size**	**Momentum**	**epochs**	**weight_decay**	**learn_rate**
SGD	60	0.8	30	$\{1 \cdot 10{-}5\}$	$2 \cdot 10^{-4}$

For what concern the final results, all the quality metrics obtained from the Confusion Matrixes of each experiment, useful to analyze the distinct qualities and benefits of each of the experiments, will be shown in a few pages.

6.1.2 Classification Metrics

In the table (Tab.6.2) are showed the results of several tests performed with Pytorch Architecture.

The test were performed with the following settings:

- X ≡ NO DATA AUG ;
- 1 ≡ DATA AUG OFFLINE ≡;
- 2 ≡ DATA AUG OFFLINE + BALANCED DATA SAMPLER:
  only a single transformation is given to each duplicated train image, that is performed before the training.
- 3 ≡ DATA AUG ONLINE + BALANCED DATA SAMPLER:
  only a single transform is given to each duplicated image, but it is performed "On The Fly".
- 4 ≡ DATA AUG ONLINE + DATA AUG OFFLINE (GAN) + BALANCED DATA SAMPLER:
  some added images from the GAN architecture, and the remaining ones duplicating the real images applying the same strategy as experiment 3;

CROP IMAGES - CLASSIFICATION METRICS											
DA	**(GEN) B.A.**	**(AVG) B.A.**	**ACC**	**PRE**	**REC**	**F1**	**QWK**	**MS**	**MAE**	**EPOs**	***t*EPO**
X	0,4971	0,5051	0,5142	0,5247	0,4971	0,4973	0,6714	0,4971	0,7099	16/100	6 min
1	0,5063	0,5109	0,5259	0,5343	0,5062	0,5058	0,6681	0,5063	0,7217	24/100	10:30 min
2	**0,5212**	0,5484	0,5377	0,5493	0,5212	0,5183	**0,7142**	0,5212	0,6911	19/100	10:30 min
3	0,4652	**0,5492**	**0,5613**	**0,5723**	**0,5441**	**0,5407**	0,7065	**0,5441**	**0,6486**	18/100	9:10 min
4	0,5078	0,5309	0,5237	0,5238	0,5078	0,5051	0,6767	0,5078	0,7005	27/100	9:10 min

Table 6.2: Results of several test performed with Pytorch Architecture

The best results obtained in terms of accuracy come from test 3, as the logic was to apply automatically a single transformation to guarantee higher diversity. Consider also that a possible future approach of combining more transformations can perhaps bring more improvements guaranteeing higher diversity, with the constraint, however, of implementing a logic in which they are not applied randomly because some of them behave better than others. But the most important aspect to note is that the actual GAN isn't able to guarantee clear improvements on the results, even for the limited diversity generated meaning that it doesn't actually bring more features to learn from.

These results come from the fact that the actual availability of the training images is too limited to guarantee wide generations of possibilities, as well as the not sufficient capacity of the basic one implemented on generating new efficacy features even if the quality is acceptable.
Regarding the classifier performances with GAN created images, is clear that there's still a gap to be bridged, as the results don't bring any advantage to the final predictions, also ruining the basic contributions from the ON.-D.A.-C.V.T. . So, this will inevitably bring on the implementation of a more valid architecture for this case of study.

6.2 Data Augmentation qualitative results

In this section some generated examples will be shown in order to visually evaluate the goodness of the images.

6.2.1 Online Data Augmentation with CV techniques comparisons

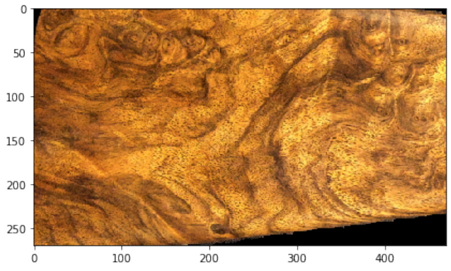
Online Data Augmentation created images comparison

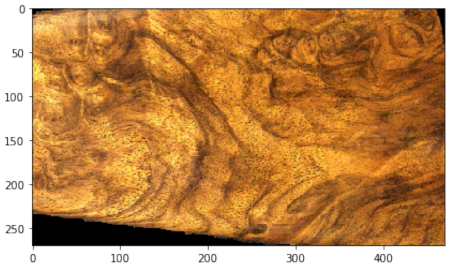
The only consideration that has to be underlined is that adopting these transformations with respect to applying a banal duplication of the starting images, brings more information to the network, permitting it to improve the classification performances.
Examples of the transformations have been already shown in 4.2, so other illustrations are avoided.

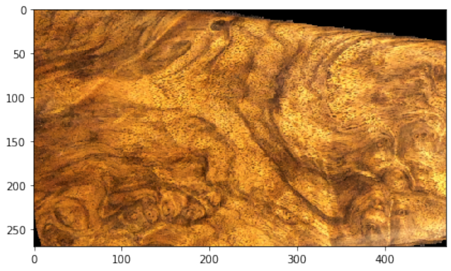
Main comparisons of these manipulations operations:

- with the horizontal/vertical and brightness transformations it's possible to maintain homogeneous images while improving a little the contrast of the colours giving more importance to the veins and grains.
- while for the other Posterize manipulation a differentiation for the colour is performed, making the images more different but without changing too much the conformations in order to not ruin the classification performances.

As already done in section 4.2, another example of all the transformations for a certain image is showed in Fig. 6.1:


(a) Normal Image


(b) Horizontal Rotated Image


(c) Vertical Rotated Image

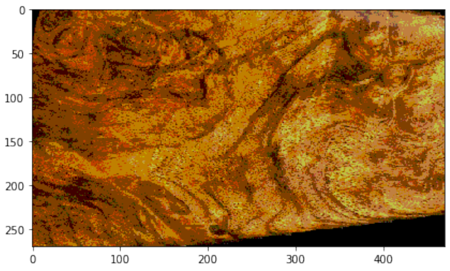

(d) Posterized and Brightened Image Image

Figure 6.1: Configuration adopted (left) Vs Original image (right)

Other CV transformations were implemented, such as Canny edge detection or even a Gaussian method combined with a Laplacian transformation, but poor results were obtained due to the mixing of the images kind, which brings too many differences to images, making them too heterogeneous.

6.2.2 Offline Data Augmentation with GAN comparisons

Best Hyperparameters setting

Several tests were carried out for all the 10 different GAN instances for all the 10 Q.C. . The strategy adopted was firstly to find out the best hyperparameters configurations for the GAN architecture to be trained with, evaluating them comparing the images saved visually, and discovering that the base settings of the architecture were the best in order to obtain the best performances. In fact, adding too much latent space would not bring much difference to the final images as the range of colors was limited, and all of them in a brown scale. All the GAN trainings were also performed by a train-set dimension of 64 images, so a perfect batch size because even adopting a train-set of 100 images wouldn't bring effectively improvements, and at the same time it caused a slowdown of the training process due to the increment of the images. Moreover, exceeding a quantity

of 100 images for each set, a disconnection from the runtime occurs losing all the training procedures.
The hyperparameters settings are summarized below in table 6.3

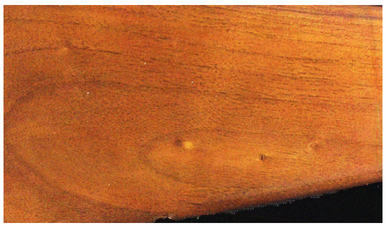
Table 6.3: Best GAN hyperparameters setting

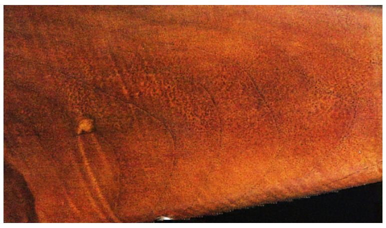
Optimizer	**Batch_size**	**Latent_space**	**epochs**	**beta1**	**beta2**	**learn_rate**
Adam	64	100	3500	0.5	0.999	$2 \cdot 10^{-4}$

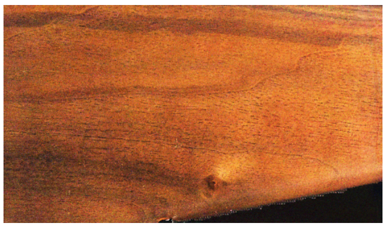
GAN created images comparison

For what concern created images (adopting the same architecture for different quality classes), it has been found that the better performances came from the high-quality classes, as those images presented more details (veins and grains from where learn from) resulting a better-pixelled image. This is because the basic GAN implemented is such a rough architecture to the point that is able to perform better with more detailed images because it considers the less detailed images too similar without generalizing much.
Now some generated examples for 3 quality grade cases will be plotted, showing also a reference batch of real images to make a visual comparison for each of these classes. In particular cases from the lowest grade will be shown, as well as the higher one and also for a middle ones.

Class 0 Here an example of a batch of the samples created for the lowest detailed Q.C. is shown in Fig. 6.2, while Fig. 6.3 shows real images:


(a) sample 1


(b) sample 2


(c) sample 3

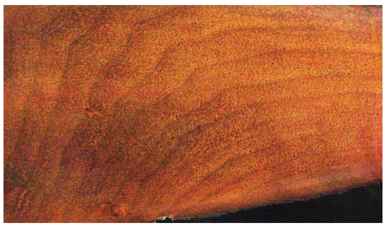
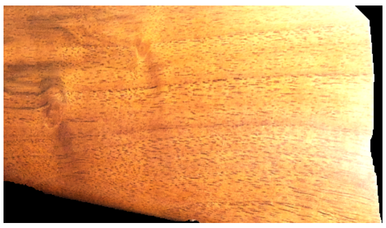
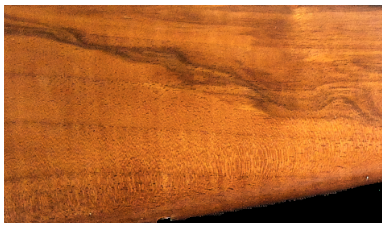
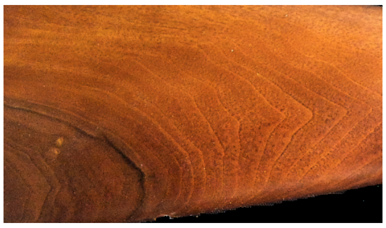

(d) sample 4

Figure 6.2: Batch of generated images (Q.C. 1)


(a) sample 1


(b) sample 2


(c) sample 3

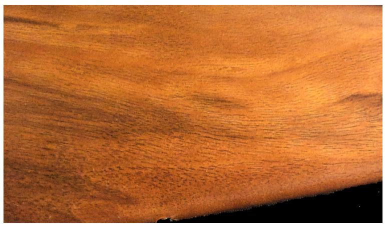
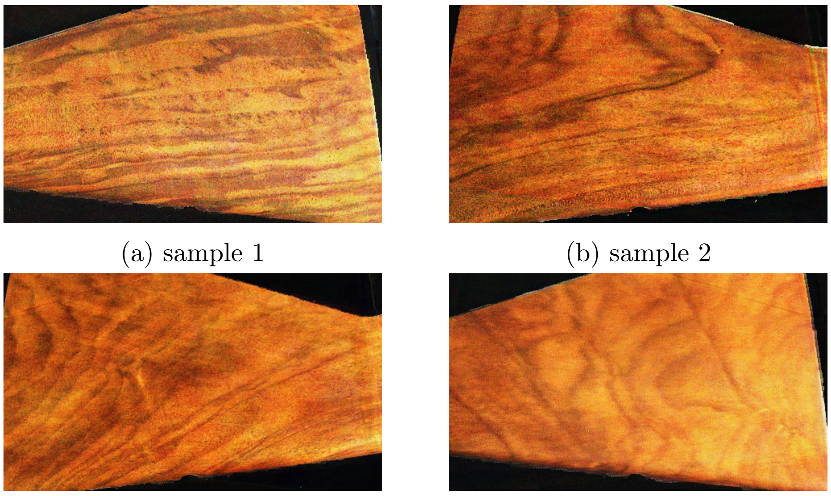

(d) sample 4

Figure 6.3: Batch of real images (Q.C. 1)

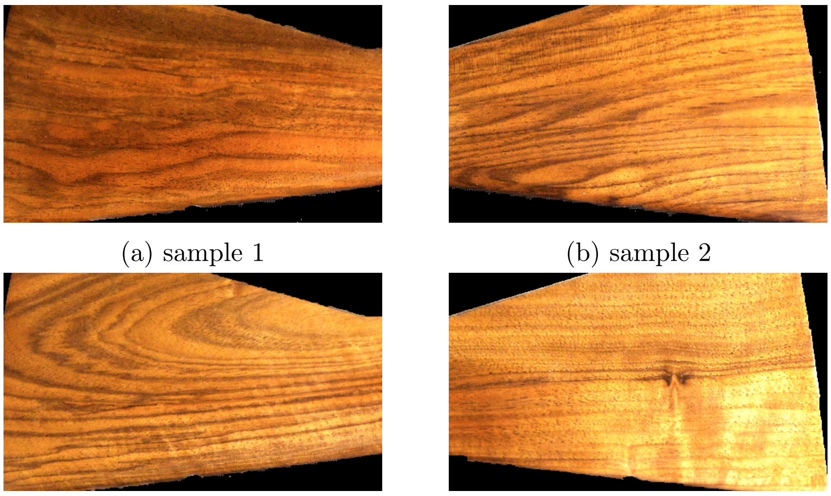
Class 3^- To obtain a better visual comparison between created images, even the batch for a middle class is illustrated in order to have a reference even for the middle detailed classes of the dataset. Results are shown in Fig. 6.4 for generated ones, and Fig : 6.5 for the real ones.



(a) sample 1 (b) sample 2

(c) sample 3 (d) sample 4

Figure 6.4: Batch of generated images (Q.C. 3^-)

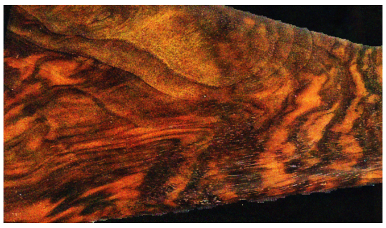


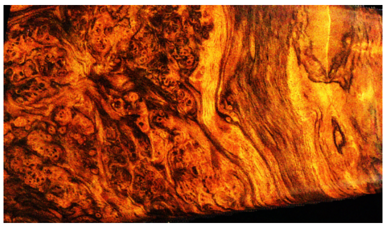
(a) sample 1 (b) sample 2

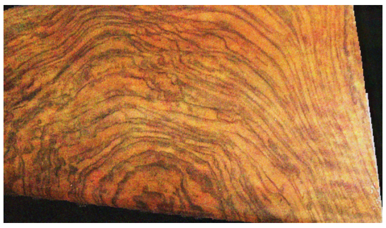
(c) sample 3 (d) sample 4

Figure 6.5: Batch of real images (Q.C. 3^-)

Class 4^+ Last but not least, the batch for the high detailed Q.C. is represented, observing the higher quality level obtained for the samples in respect of the previous 2 cases. Results in Fig. 6.6 for generated and Fig. 6.7 for reals:


(a) sample 1


(b) sample 2


(c) sample 3

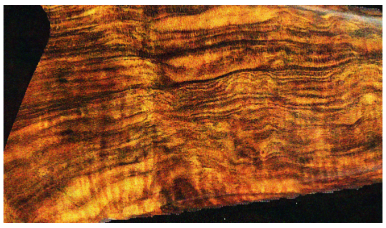
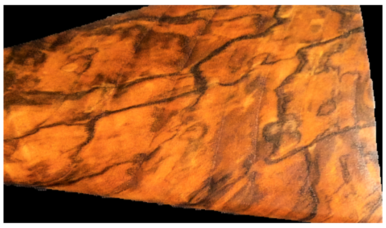
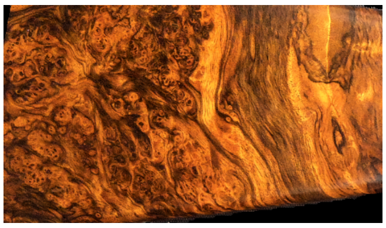
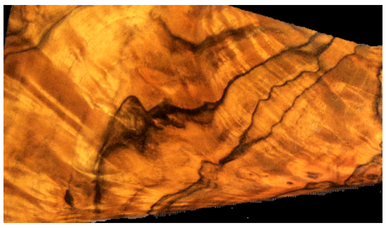

(d) sample 4

Figure 6.6: Batch of generated images (Q.C. 4^{+})


(a) sample 1


(b) sample 2


(c) sample 3

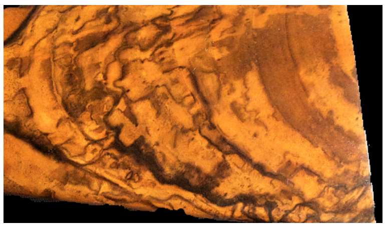

(d) sample 4

Figure 6.7: Batch of real images (Q.C. 4^{+})

An important note that has to be observed is that considering sample 2 from both the generated and real images, there's clear evidence that those images are almost the same, so the GAN architecture is only able to manipulate limited pieces of veins and grains into the image, but without changing too evidently the piece of wood's structure. This is due to the limited capacity of this rough GAN to generating too strong different features, influenced also by the very limited amount of images in the train set.

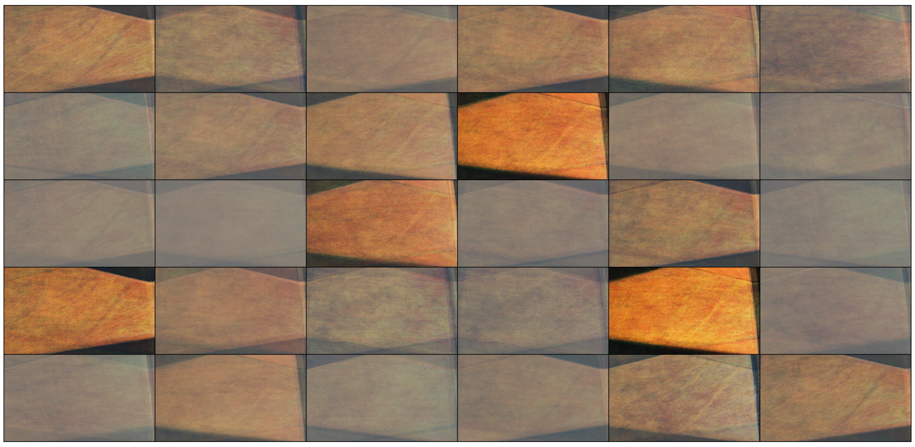
Epochs batches generated Class 3

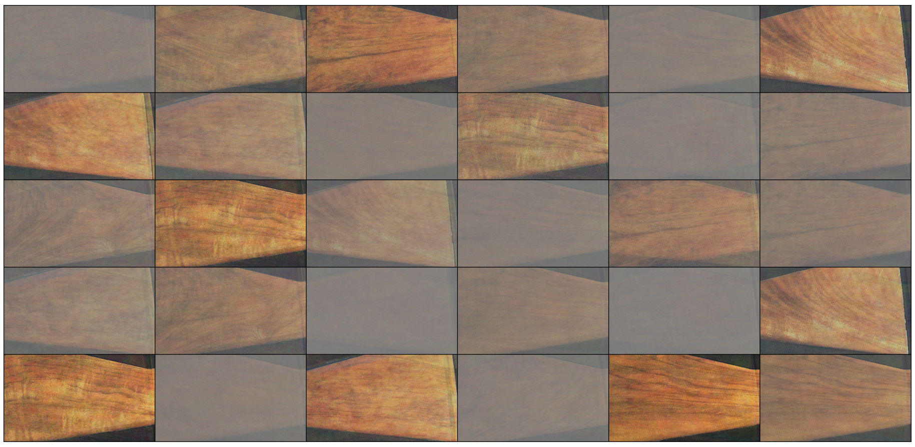
To complete the presentation of the results even some patches generated are shown to demonstrate the necessity of waiting for a number of epochs at least equal to 2500 to get quite acceptable samples, until getting the best quality possible at about 3000 epochs. After that amount of images, the GAN network often starts to totally collapse, resulting in useless to continue the training phase. This is also confirmed by the fact that it would have to start again to improve itself from low-level quality of images so that it would take too much time even to get again the same quality images. An example of batches is given till the epoch 2500 in Fig. 6.8, because after that epoch results are the same in terms of diversity, with the only difference that they can improve a little in quality and reduction of noise.

It's clear how this basic GAN architecture isn't optimized on working in such high-dimensional images, in fact, it takes really much epoch to reach good quality image creations. In fact, it takes almost about 2500 epochs to reach a quite acceptable level of quality, since batches of epoch 1500 demonstrates that has still too much noise. This aspect is obviously influenced also by the limited amount of available images for the train set, making the GAN capable only to change small features inside the image space.

6.3 Computation Effort

How it's observable from the summary table 6.2, with the DataLoader it's guaranteed not only the improvement of the results but also a reduction on computational resources required, both regarding the computational complexity and memory resources, because we load the images only when needed at the time and discarding them when the process ends.
More specifically, looking at the last column of that table, it's clear how implementing the transformations online when needed guarantees a lower time of training execution, gaining about 1.30 min for each epoch processed, which considering about 20 epochs of execution typically, brings about 30 min. This result is satisfactory in view of future developments that may require multiple epochs of executions, gaining an important amount of time back, and useful to move on quickly to the later upgrades.


(a) batch 1 - epoch 500


(b) batch 2 - epoch 1500

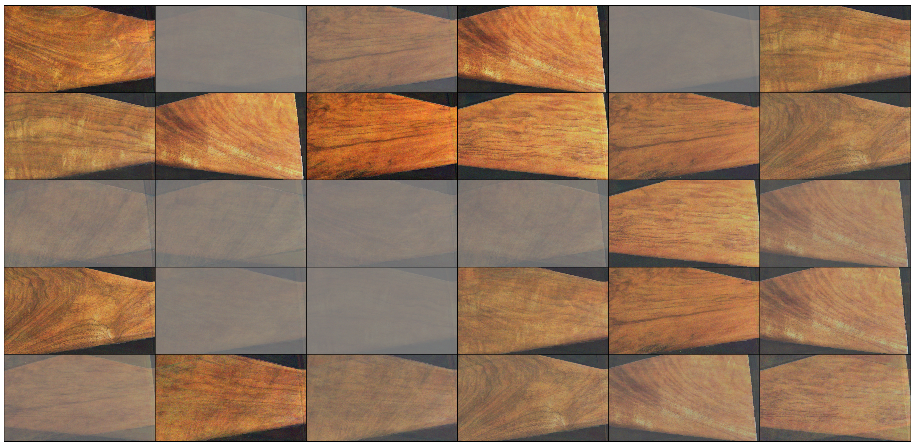

(c) batch 4 - epoch 2500

Figure 6.8: Epochs batches for Q.C. 3

Chapter 7

Discussions

7.1 Impact of the proposed approach for Q.C. classification

7.1.1 ON. - D.A. - C.V.T observations

It has been found that these kinds of transformations were the strongest contribution that brought effectively higher increments in terms of classification metrics in respect of the effectively GAN effect.
In fact, we found out that by applying uniquely these transformations even without considering the GAN created images, results had improved by about 5 percentage points, increasing the prediction accuracy **from** 0.5142 **to** 0.5613 for a specific test where D.A. was applied separately thanks to the probabilities of each transformation settled to 1, choosing the specific transformation corresponding to the label added in the dataset. In this way, we are sure that each image was surely not identical to the others.

7.1.2 Data Loader observations

For what concerns the Memory space computation aspect, as already clarified, the importance of the Data Loader is fundamental because, without its usage, we couldn't be able to finish the Classifier training without going Out Of Memory (OOM).
This constructor method provided by the Pytorch framework guarantees as shown in the results table, an effective decrease in Computational Complexity by reducing about 1.30 minutes for each epoch performed by the classifier. Anyway even if it obtains a concrete difference in temporal terms, the higher results are obtained by the improvement in memory resources management.
In fact, even if Colab doesn't directly give back (as output) memory resources employed

during the training periods (particularly when the runtime disconnects itself when goes OOM), the effective results met during the experiments were that without implementing the Data Loader, both the trainings for the GAN and the VGG16 classifier would have gone Out Of Memory in most of the cases. In particular, even adopting this helpful constructor, the GAN was still not efficient at all, as the Runtime would suddenly disconnect removing the memory resources allocated with training sets equal to or higher than 128.

7.1.3 OFF. - D.A. - GAN observations

As the results demonstrated, even obtaining appreciable quality images, the classifier network wasn't still able to perform good and acceptable improvements to the accuracy score, increasing the prediction accuracy **from** 0.5142 **to** 0.5237. This is probably due to the limited diversity of the newly generated features, meaning that the created images tended to be a lot similar to the original ones and even because the quality is still not reaching the original ones.
A more in-depth analysis will be described in the next section, 7.2.

7.2 Analysis of Offline Data Augmentation with GAN results

A detailed discussion has to be done on what concerns the aspects of the OFF. - D.A. - GAN approach. To begin with, the GAN architecture is only able to manipulate limited pieces of veins and grains into the image, but without changing too evidently the piece of wood's structure. This is due to the limited capacity of this rough GAN to generate too strong different features, firstly of the huge sized image space involved in this thesis project, and secondly, influenced by the very limited amount of images in the train set. Inevitably, it will lead to generating too few new useful features for the Classifier to learn from.
In the following subsections, some explanation examples are given to demonstrate what has been said.

7.2.1 GAN evaluation metrics

In this discussion part, the goodness of the created images is evaluated showing the metrics results performed for all the GAN instances, 7.1, and subsequently, observations will be introduced.

CROP IMAGES - GAN METRICS								
Class	**FID (ref)**	**FID**	**IS (ref)**	**IS**	**SSIM (ref)**	**SSIM**	**LPIPS (ref)**	**LPIPS**
1	121.647	139.467	2.177	1.689	0.228	0.1429	$0,539 \pm 0.0254$	$0,5353 \pm 0.0189$
2^-	118.907	108.537	2.208	1.965	0,2159	0.1329	$0,480 \pm 0.012$	$0,555 \pm 0.01$
2	93.722	136.594	1.947	1.563	0.205	0.109	$0,505 \pm 0.009$	$0,572 \pm 0.009$
2^+	115.351	132.079	2.181	1.856	0.2155	0.1199	$0,670 \pm 0.01$	$0,676 \pm 0.01$
3^-	110.552	129.417	2.289	1.582	0.233	0.127	$0,616 \pm 0.014$	$0,638 \pm 0.012$
3	91.293	135.891	2.318	1.936	0.2078	0.1081	$0,744 \pm 0.008$	$0,745 \pm 0.009$
3^+	-	-	-	-	-	-	-	-
4^-	104.707	109.904	2.005	1.827	0.193	0.119	$0,671 \pm 0.007$	$0,717 \pm 0.006$
4	88.329	123.349	2.006	2.031	0.197	0.123	$0,645 \pm 0.005$	$0,644 \pm 0.006$
4^+	135.585	114.316	1.786	1.53012	0.175	0.123	$0,645 \pm 0.012$	$0,716 \pm 0.011$

Table 7.1: Results of GAN metrics applied to each quality grade sub-set of data

Results demonstrate that the metrics results are really much next to each other, comporting even acceptable generated images. Anyway, like it has been experimented by the classifier results with the GAN images added to the dataset, it doesn't improve much in accuracy due to the limited new features created on the new images, even if the similarity is even quite different. In fact, observing the table results, in particular, the SSIM metric column, the lower the value is the higher the diversity of the created images. Anyway, getting small values like this guarantees diversity in the created images, but this is also due to lower quality images. This factor will of course bring diversity to the images but because is influenced also by the lower detailed images generated.
This observation is confirmed by the FID and IS metrics results, in which the images demonstrate to have a lower quality with respect to the real ones. Indeed respectively, a lower FID correspond higher quality, and for a higher IS corresponds a higher quality. So, observing GAN images evaluations, is clear how those have a lower definition and so pixels are more discontinuous. This, as said before, will of course bring higher diversity. For this reason, in future improvements, the expected behavior will be to obtain similar Quality Metrics evaluations (FID, IS) while trying to obtain as in this case, higher diversity getting lower levels of SSIM (Diversity Metrics).
However, the LPIPS metric results demonstrate similar results, meaning that the actual quality obtained is not even too different in respect of the real images, so it means we are not too distant in obtaining valid images. In fact, a lower level measure means more similarity, but however, this metric measures diversity in a different way (like human perceptions), so the way of interpreting this metric is that the similarity is evaluated with respect to the reality of the images, like a Discriminator network would do.
Concluding, considering the rough of the implemented GAN network, it's clear how

there's still a big growing potential on the image creation side, where future works will be followed by an advanced GAN network implementation.

7.2.2 GAN Mode Collapse

Usually, you need your GAN to produce a wide variety of outputs. In this case, for example, we would like to obtain as many different created images as possible for each quality grade instantiation of the GAN architecture. However, if a generator produces an especially plausible output, the generator may learn to produce only that output. In fact, the generator is always trying to find the one output that seems most plausible to the discriminator.
If the generator starts producing the same output (or a small set of outputs) over and over again, the discriminator's best strategy is to learn to always reject that output. But if the next generation of discriminator gets stuck in a local minimum and doesn't find the best strategy, then it's too easy for the next generator iteration to find the most plausible output for the current discriminator.
This problem is called "Mode Collapse", and may happen due to problems in training, such as the generator finds a type of data that is easily able to fool the discriminator and thus keeps generating that one type. Because there is no incentive for the generator to switch things up, the entire system will over-optimize on that one output because the discriminator never manages to learn its way out of the trap. In this particular case of study, this outcome can come from not only by the not-perfectly fitted architecture but also by the limited amount of samples used to train the architecture, which brings the GAN model on hardly train itself, learning too limited features to variegate the created images.

This is even demonstrated by the following figure Fig. 7.1 and Fig. 7.2, which shows non-constant trend for what concern the Generator loss function both for the higher Q.C. and the lower one.

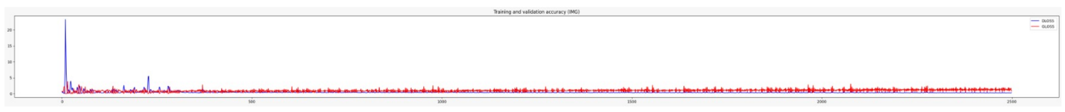


Figure 7.1: Mode Collapse class 1

The mode collapse is explained by the fluctuation of the generator loss function. Moreover, the figure Fig.7.2 shows a high variance over the generator loss trend during

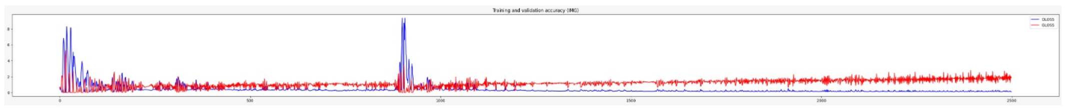


Figure 7.2: Mode Collapse class 4^+

the training, meaning that it never finds a stable space in which learns to create different samples, remaining fixed on the production of the same images. Moreover, the discriminator loss is always touching values between 0.1 and 0.4 never going higher, when instead a stable GAN training should have the training loss stabilizing between 0.5 and 0.8 typically.
In particular, this architecture guarantees limited stable spaces during the first 1000 epochs in which, however, it is not still good at generating detailed images, as already shown in Fig.6.8, where even the second image (b), at epoch 1500 is still not obtaining clear and acceptable images. This is a symptom that the actual GAN is not well fixed with the examined problem in this thesis, bringing to the implementation of a more efficient architecture in future works. To conclude, another important demonstration can be taken by observing the two generator losses behaviors in figures 7.1 and 7.2, where the second one is even obtaining a higher variance during the training phases while even increasing over time (unwanted behavior), due also to the higher features it has to learn from, confusing the generator more than the easiest classes.

Chapter 8

Conclusions & Future work

Nowadays, the quality control task on Industry4.0 is still mainly based on the evaluation of the human eye by the Company operators requiring a hard and continuous effort, but especially it can be defined as a very time-consuming task. The introduction of a DL approach has been proven to have quite good potential in being able to support the human operator. Anyway, it still does not bring desired results, in which an accuracy less than 75% is not even acceptable.

Results obtained demonstrate that the Design of 2 different Data Augmentation strategies to improve classification performances in a complex task such as Q.C. obtains different behaviors. On the first side, ON.-D.A.-C.V.T approach ensures an improvement of the classification performances, increasing the accuracy, while also reducing the memory resources needed to complete the training phase and the and also reduction of Computational Complexity bringing a lower total execution time.
Anyway, on the other side, the OFF.-D.A.-GAN approach obtains not acceptable performances compared to the basic CV transformations in order to have a concrete accuracy increment, but with high chances of improvements. In any case, the implementation of the techniques of Data Augmentation and the Data Loader as well demonstrated to be fundamental aspects in order to reach the future objective of improving the classification accuracy performances.

To sum up, we have come up against multiple limitations which have maintained the classification performances actually too low in order to guarantee an effectively Industrial implementation. Specifically, the most influential problem encountered is the really little sized dataset, that for both the task of generation and classification leaves a strong gap with possible acceptable results, making the study case difficult to apply right now. Moreover, combining this problem with the unsatisfactory effect of the original GAN implemented, it brings into the creation of limited images, which in some cases demonstrate to be almost the same, (situation described by the Mode Collapse problem).

These last considerations will obviously bring us to both the implementation of a more efficient Classifier network and a more efficient GAN which will lead to the generation of more images with higher diversity and quality, trying to reduce at the same time the Computational complexity needed avoiding hours-long trainings.
Taking into account that the DL model should focus on the wood part we are currently considering in future works the inclusion of a segmentation step prior to the Q.C. classification, in order to improve the predictions while reducing the focus on the geometries so that we can better mitigate the bias. Concurrently, a C-GAN (Conditional GAN) will be considered in order to generate samples from all the 10 classes of grades with the same instance of the GAN.

Future improvements may also be addressed to integrate a Split Cross Validation to reach better contributions from the validation set during the training phase, guaranteeing a higher feature space to be analyzed by the neural networks. Indeed, having a low dimensional set of images like in this case of study (2120 images), the random split for the creation of the training, validation and test sets can cause the overfitting problem, so that the model learns too many features from the training set, while not having a strong contribution from the validation set. This is justified by the fact that from each image, the veins can be also completely different, so it can't find useful features corresponding to the validation set images and evaluate them correctly. So, implementing a Split Cross Validation, we change each epoch the validation set in order to learn from as many images as possible, using 20% of more images from where learn from, in order to help the network generalizing better.
Anyway, the main strategy thought for the next step will be first to find out a better fitting GAN architecture (C-GAN, WGAN-GP, STYLE-GAN) for this kind of images, and then, after obtaining satisfactory results, an Adversarial Learning strategy will be implemented in combination with it. This kind of Neural Network will have the goal of reducing the geometry bias due to the geometry feature of the Non-Cropped images of the dataset, based on an adversarial ramification into the VGG16 architecture which will focus on classifying the shotgun series of the rifle stock, guaranteeing an increase of its lost function corresponding to a decrease of the bias (because trying to increase the loss means to mitigate the bias due to the geometry).

Chapter 9

Aknowledgements

This thesis was developed in collaboration with VRAI (Vision, Robotics and Artificial Intelligence, Department of Information Engineering, Polytechnic University of Marche), Benelli Armi S.p.a (`https://benelli.it/it/`) and Prof. Maggie Cheng (`https://www.iit.edu/directory/people/maggie-cheng`), Director, Center for Interdisciplinary Scientific Distribution, Illinois Institute of Technology.
Future work about this thesis will be developed within one-year visiting period at the Center Interdisciplinary Scientific Distribution, Illinois Institute of Technology (USA) supervised by Prof. Maggie Cheng.

Bibliography

[1] N. e. A. Zhonghe, Fengzhou, “State of the art in defect detection based on machine vision,” *International Journal of precision Engineering and Manufacturing-Green Technology*, vol. 9, pp. 661–691, 2021.

[2] G. e. A. Wang, Zhang, “Deep learning for smart manufacturing: Methods and applications,” *Journal of manufactoring Systems*, vol. 48, pp. 144–156, 2018.

[3] J. Villalba-Diez, D. Schmidt, R. Gevers, J. Ordieres-Meré, M. Buchwitz, and W. Wellbrock, “Deep learning for industrial computer vision quality control in the printing industry 4.0,” *Sensors*, vol. 19, no. 18, p. 3987, Sep 2019. [Online]. Available: http://dx.doi.org/10.3390/s19183987

[4] J. Francis and L. Bian, “Deep learning for distortion prediction in laser-based additive manufacturing using big data,” *Manufacturing Letters*, vol. 20, pp. 10 – 14, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221384631830172X

[5] R. Rosati, L. Romeo, G. Cecchini, F. Tonetto, L. Perugini, L. Ruggeri, P. Viti, and E. Frontoni, “Bias from the wild industry 4.0: Are we really classifying the quality or shotgun series?” in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, Eds. Cham: Springer International Publishing, 2021, pp. 637–649.

[6] T. M. K. Connor Shorten, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, 2019.

[7] G. C. Chih-Chieh Y., “Accelerating data loading in deep neural network training,” *arXiv*, 2019.

[8] Z. L. e. A. Ning Chen, Zijin Xu, “Data augmentation and intelligent recognition in pavement texture using a deep learning,” *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 2021.

[9] G. W. B. e. A. Dercilio Junior Verly Lopes, Gustavo Fardin Monti, “Creating high-resolution microscopic cross-section images of hardwood species using generative adversarial networks,” *Frontiers in Planet Science*, 2021.

[10] A. K. e. A. Bilel Benjdira, Adel Ammar, “Data-efficient domain adaptation for semantic segmentation of aerial imagery using generative adversarial networks,” *ResearchGate*, 2020.

[11] L. Romeo, J. Loncarski, M. Paolanti, G. Bocchini, A. Mancini, and E. Frontoni, “Machine learning-based design support system for the prediction of heterogeneous machine parameters in industry 4.0,” *Expert Systems with Applications*, vol. 140, p. 112869, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417419305792

[12] N. Sakib and T. Wuest, “Challenges and opportunities of condition-based predictive maintenance: A review,” *Procedia CIRP*, vol. 78, pp. 267–272, 2018.

[13] M. Calabrese, M. Cimmino, F. Fiume, M. Manfrin, L. Romeo, S. Ceccacci, M. Paolanti, G. Toscano, G. Ciandrini, A. Carrotta *et al.*, “Sophia: An event-based iot and machine learning architecture for predictive maintenance in industry 4.0,” *Information*, vol. 11, no. 4, p. 202, 2020.

[14] J. P. U. Cadavid, S. Lamouri, B. Grabot, R. Pellerin, and A. Fortin, “Machine learning applied in production planning and control: A state-of-the-art in the era of industry 4.0,” *Journal of Intelligent Manufacturing*, pp. 1–28, 2020.

[15] C. A. Escobar and R. Morales-Menendez, “Machine learning techniques for quality control in high conformance manufacturing environment,” *Advances in Mechanical Engineering*, vol. 10, no. 2, p. 1687814018755519, 2018.

[16] T. Wang, Y. Chen, M. Qiao, and H. Snoussi, “A fast and robust convolutional neural network-based defect detection model in product quality control,” *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9-12, pp. 3465–3471, 2018.

[17] Z. L. e. A. Ning Chen, Zijin Xu, "Data augmentation and intelligent recognition in pavement texture using a deep learning," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 1, p. 10, 2021.

[18] S. Basu, S. Mukhopadhyay, M. Karki, R. DiBiano, S. Ganguly, R. Nemani, and S. Gayaka, "Deep neural networks for texture classification—a theoretical analysis," *Neural Networks*, vol. 97, pp. 173–182, 2018.

[19] P. Simon and U. V, "Deep learning based feature extraction for texture classification," *Procedia Computer Science*, vol. 171, pp. 1680 – 1687, 2020, third International Conference on Computing and Network Communications (CoCoNet'19). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050920311613

[20] A. Muniategui, A. G. de la Yedra, J. A. del Barrio, M. Masenlle, X. Angulo, and R. Moreno, "Mass production quality control of welds based on image processing and deep learning in safety components industry," in *Fourteenth International Conference on Quality Control by Artificial Vision*, C. Cudel, S. Bazeille, and N. Verrier, Eds., vol. 11172, International Society for Optics and Photonics. SPIE, 2019, pp. 148 – 155.

[21] M. M. e. A. Ian J. Goodfellow, Jean Pouget-Abadie, "Generative adversarial network," *arXiv:1406.2661*, 2014.

[22] T. U. e. A. Martin Heusel, Hubert Ramsauer, "Gans trained by a two time-scale update rule convergence to a local nash equilibrium," *arXiv*, 2017.

[23] R. S. Shane Barratt, "A note on the inception score," *arXiv*, 2018.