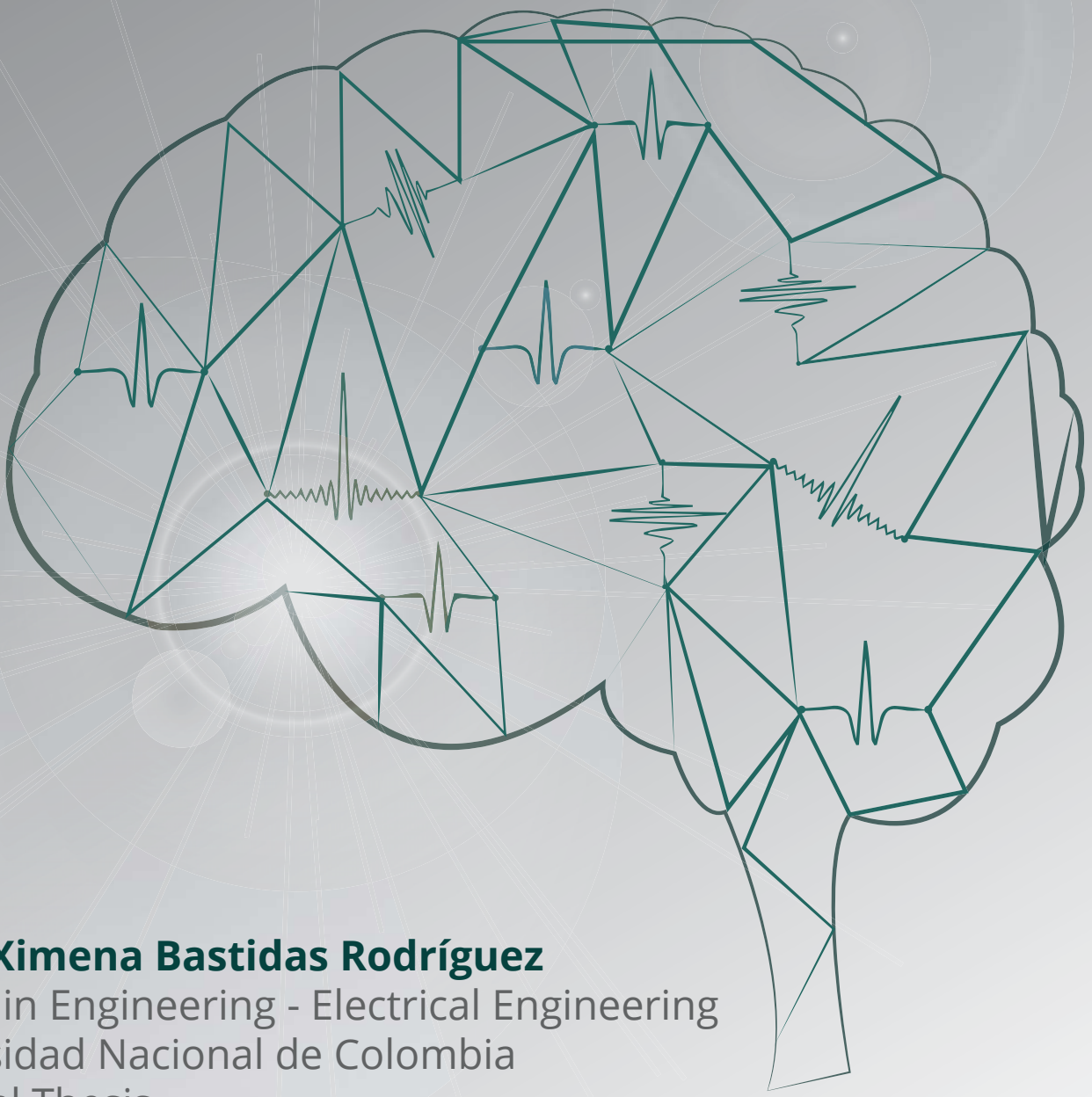
A textural deep neural network architecture for mechanical failure analysis.



Maria Ximena Bastidas Rodríguez
PhD(c). in Engineering - Electrical Engineering
Universidad Nacional de Colombia
Doctoral Thesis

Advisor:
Ph.D., Flavio Augusto Prieto Ortiz
Co-Advisor:
Ph.D., Luisa Fernanda Polanía Cabrera

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ

A textural deep neural network architecture for mechanical failure analysis

Maria Ximena Bastidas Rodríguez

Universidad Nacional de Colombia
Faculty of Engineering, Department of Electrical and Electronic Engineering
Bogotá, Colombia
2019

A textural deep neural network architecture for mechanical failure analysis

Maria Ximena Bastidas Rodríguez

A Thesis submitted as partial fulfilment of the requirement for the degree of:
PhD. in Engineering - Electrical Engineering

Advisor:
Ph.D., Flavio Augusto Prieto Ortiz

Co-Advisor:
Ph.D., Luisa Fernanda Polanía Cabrera

Research Line:
Digital Image Processing
Research Group:
GAUNAL

Universidad Nacional de Colombia
Faculty of Engineering, Department of Electrical and Electronic Engineering
Bogotá, Colombia
2019

To my parents Hugo y Amparo, whose unconditional love has taught me invaluable lessons, and drives me to be the best version of myself.

Acknowledgements

To my thesis advisors, Ph.D. Flavio Prieto Ortiz and Ph.D. Luisa Polanía Cabrera. Your ideas, orientation, companion, lessons and patience were fundamental not only in the development of this work but also, in my academic research and human growth during these four years.

To the CEIBA Foundation for providing me with a scholarship, which financial resources allowed me to present the results here presented.

To Professor, Ph,D. Toshiya Hachisuka, who provided me with his knowledge, advice, time and physical and human resources to develop this research, and for allowing me to live one of the most gratifying experiences of my life. And to Adrien Gruson, thank you for being my mentor and friend; definitely, this research would not be the same without you.

To Professor M.Sc., Edgar Espejo Mora, for serving as the failure analysis expert in this research project and helping me to obtain the failure databases used in this work. And to M.Sc., Johnny Obando and M.sc., Juan Carlos Martinez, which kind collaboration was essential to validate the results here presented.

To the Universidad Nacional de Colombia, I hope someday I will give back everything that this institution has provided me with in my academic and personal training.

Finally, to my parents, sisters and loved ones, sincerely thank you because with your life and example, you have taught me that with dedication and effort I can achieve all my life goals. And, no mater what happens in our lives, we will always have each other. I love you all, thank you for being part of my life.

Abstract

Nowadays, many classification problems are approached with deep learning architectures, and the results are outstanding compared to the ones obtained with traditional computer vision approaches. However, when it comes to texture, deep learning analysis has not had the same success as for other tasks. The texture is an inherent characteristic of objects, and it is the main descriptor for many applications in the computer vision field, however due to its stochastic appearance, it is difficult to obtain a mathematical model for it. According to the state of the art, deep learning techniques have some limitations when it comes to learning textural features; and, to classify texture using deep neural networks, it is essential to integrate them with handcrafted features or develop an architecture that resembles these features. By solving this problem, it would be possible to contribute in different applications, such as fractographic analysis.

To achieve the best performance in any industry, it is important that the companies have a failure analysis, able to show the flaws' causes, offer applications and solutions and generate alternatives that allow the customers to obtain more efficient components and productions. The failure of an industrial element has consequences such as significant economic losses, and in some cases, even human losses. With this analysis it is possible to examine the background of the damaged piece in order to find how and why it fails, and to help prevent future failures, in order to implement safer conditions. The visual inspection is the basis for the generation of every fractographic process in failure analysis and it is the main tool for fracture classification. This process is usually done by non-expert personnel on the topic, and normally they do not have the knowledge or experience required for the job, which, without question, increases the possibilities of generating a wrong classification and negatives results in the whole process.

This research focuses on the development of a visual computer system that implements a textural deep learning architecture. Several approaches were taken into account, including combining deep learning techniques with traditional handcrafted features, and the development of a new architecture based on the wavelet transform and the multiresolution analysis. The algorithm was test on textural benchmark datasets and on the classification of mechanical fractures with particular texture and marks on surfaces of crystalline materials.

Key words: Machine Learning, Computer Vision, Deep Learning, Texture Analysis, Failure Analysis.

List of Figures

List of Tables

Content

1. Introduction

Convolutional neural networks (CNNs) have become the dominant approach for different machine learning applications. Since AlexNet [68] performed much better than other models in the ImageNet challenge [27], several deep learning architectures have been developed. Based on backpropagation, CNNs can leverage correlation and structure inside datasets by directly tuning the network trainable parameters for a given task. However, when it comes to texture, deep learning has not had the same success as in other classification tasks [13], which becomes a limitation of deep learning given that texture is an inherent feature of objects and is the primary descriptor for many applications in computer vision [21].

The trend in CNNs is to increase the number of layers to be able to model more complicated mathematical functions, to the point that recent architectures surpass 100 layers [52, 54]. There is, however, no guarantee that increasing the number of layers is always advantageous. Zagoruyko *et al.* [120] indeed showed that decreasing the number of layers and increasing the width of each layer leads to better performances than their commonly used thin and very deep counterpart, while reducing training time. Their results also support our general observation that current CNNs are not necessarily designed systematically, but usually through a manual process based on trial-and-error [32].

A limitation of such networks is the lack of interpretability, which is usually referred to as the Achilles heel of CNNs. Convolutional neural networks are frequently treated as black-box function approximators which map a given input to a classification output [29]. As deep learning becomes more ubiquitous in domains where transparency and reliability are priorities, such as healthcare, autonomous driving and finance, the need for interpretability becomes imperative [19]. Interpretability enables users to understand the strengths and weaknesses of a model and conveys an understanding of how to diagnose and correct potential problems [29]. Interpretable models are also considered less susceptible to adversarial attacks [99].

Furthermore, when it comes to texture, deep learning has not had the same success as in other classification tasks [13], which becomes a limitation of deep learning given that texture is an inherent feature of objects and is the primary descriptor for many applications in computer vision [21].

This research focuses on developing a deep learning approach able to classify texture. We

choose failure analysis as an application because it plays an important role in many industries, such as the automotive industry [86]. The failure of an industrial element may lead to significant economic losses, and in some cases, even human losses. Visual inspection is the base for fracture classification and is typically performed by non-expert personnel on the topic who may not have the knowledge or experience that the job requires [7]. Fractographic classification can be performed as a texture analysis problem [12]. It relates to the visual analysis of mechanical fracture surfaces to find propagation patterns and the origin of the fracture [69]. Through characterization of the fracture surface, it is possible to study the history before the failure happened, and ultimately, the causes of the failure.

To achieve this, we first proposed a modification of traditional CNNs consisting of integrating deep models with handcrafted features. More precisely, three different sets of handcrafted features, namely, Haralick, fractal dimension and LBP, extracted from the features maps generated by the convolutional layers of the VGG-19 deep learning model. We tested our model in a textural benchmark dataset denominated KTH-TIPS [18]; which demonstrated that blending deep learning models with traditional approaches leads to better results compared to those obtained with deep learning techniques only. This motivated us to design an architecture that captures texture features while performing an end-to-end training.

Theoretical properties of traditional signal processing approaches, such as multiresolution analysis using wavelets, are well studied, which makes such approaches more intepretable than CNNs. Inspired by Wavelet-CNNs studies [39, 89, 117], this work proposes an approach able to perform multiresolution analysis within the network architecture by using the lifting scheme [107] to obtain a data-driven wavelet transform. The lifting scheme offers many advantages compared to the first-generation wavelets, such as adaptivity, data-drivenness, non-linearity, faster and easier implementation, fully in-place calculation, and reversible integer-to-integer transform [121].

Unlike previous works which combine CNNs and wavelets, the model learns all the filters from data in an end-to-end framework. Due to the connection with multiresolution analysis, the number of layers in our network is determined mathematically. In addition, the need for hyper-parameter tuning is significantly less compared to state-of-the-art architectures because it is based on stacking modules of the same topology. The combination of end-to-end training and multiresolution analysis via the lifting scheme allows us to efficiently capture the essential information from the input for image classification. The use of multiresolution analysis generates a relevant visual representation at each decomposition level, which contributes to the interpretability of the network.

The evaluation of the proposed network was performed on three competitive benchmarks for image classification tasks, namely, *CIFAR-10*, *CIFAR-100* and *ILSRVC-2012* (ImageNet).

The proposed model gets comparable results to those obtained by state-of-the-art classification networks. This work is the first to propose trainable wavelet filters in the context of CNNs. It was also tested on KTH-TIPS-2b [18] dataset achieving state-of-the-art results with a small number of parameters.

Finally, both approaches were tested on two fracture datasets: *i)* a real-scale fracture dataset, with three types of fractures, namely, ductile, brittle and fatigue; and *ii)* a Scanning Electron Microscope (SEM) dataset, with four kind of fractures, the three before mention plus a corrosion fatigue mode. And the results for the real-scale fracture dataset were compared to those gotten by two experts on the topic, in order to validate them. It can be concluded that the proposed architectures have competitive performances to the ones obtained with handcrafted features, while performing an end-to-end training.

As an appendix to this work we present an approach considered as a possible data augmentation application, motivated by the small amount of images obtained for the real-scale and SEM datasets. This approach uses a Progressive Growing GAN, in order to obtain new generated synthetic data. However, this approach does not represent a high improvement over the final classification, reason why it is not included in the main text.

1.1. Objectives

1.1.1. General Objective

To design a visual computer system by implementing a textural deep neural network architecture, capable of classifying texture patterns in mechanical fractures surfaces as support in failure analysis inside the department of Nariño - Colombia.

1.1.2. Specific Objectives

1. To propose a deep convolutional network architecture applied to texture databases.

2. To develop a system capable of recognizing texture by combining deep neural architectures and handcrafted features.

3. To develop an automatic classification algorithm for images from mechanical fracture surfaces.

4. To perform the systematic validation of the final system of mechanical fracture classification, with statistical significance, by the support of human experts on the topic.

1.2. Contributions

Deep learning Combined with Handcrafted Features

- The proposed model extracts three sets of handcrafted features, namely fractal dimension, Haralick features and LBP, from the output of different convolutional layers of the VGG-19 model pre-trained with the ImageNet database
- Experimental results suggested that it is beneficial to integrate handcrafted features with deep neural networks for the problem of texture classification.

Note: Achieves specific objective No. 2

Deep Adaptive Wavelet Network

- The network is interpretable since approximation and detail coefficients, which have a relevant visual representation, are generated by the multiresolution analysis, using the lifting scheme at each decomposition level.
- The architecture is designed in a systematic fashion, without a trial-and-error process, and with significantly less hyper-parameter tuning than state-of-the-art CNN models for image classification.
- The network extracts features using a multiresolution analysis approach to capture essential information for classification. The loss function used to train the network ensures that the captured information is relevant to the classification task, while keeping the properties of the wavelet transform. Furthermore, it offers competitive accuracy in image classification tasks.

Note: Achieves specific objective No. 1

Deep Learning and Fracture Analysis

- Two Failure datasets are analysed: *i)* Real-scale fracture database, with three kind of fractures (ductile, brittle, fatigue) and *ii)* Scanning Electron Microscope with four kind of fractures to analyze (ductile, brittle, fatigue and corrosion fatigue).
- The results of Deep Learning plus handcrafted features method show that this approach can improve those of the VGG-19 features obtained from a pre-trained architecture.
- DAWN architecture proves to be suitable for textural analysis in real world problems.
- Human expert validation is presented showing competitive results on the real-scale fracture database for the DAWN architecture.

Note: Achieves specific objectives No. 3 and 4

1.3. Publications

1. C.A. Rivera-Morales, M.X. Bastidas-Rodríguez, F.A. Prieto-Ortiz *Densenet model combined with Haralick's handcrafted features for texture classification*, IEEE Latin American Conference on Computational Intelligence (LA-CCI), 2018.

2. M.X. Bastidas-Rodriguez, F.A. Prieto-Ortiz, L. Polania *A textural deep neural network with handcrafted features for mechanical failure classification*, IEEE International Conference on Industrial Technology (ICIT), 2019.

3. MX Bastidas-Rodríguez, A Gruson, L Polanía, S Fujida, FA Prieto-Ortiz, K Takayama, T Hachisuka *Deep Adaptive Wavelet Network*, Winter Conf. on Applications of Computer Vision (WACV), 2020. Accepted (arXiv:1912.05035).

4. MX Bastidas-Rodríguez, L Polanía, A Gruson, FA Prieto-Ortiz, T Hachisuka *Deep Learning for fractographic classification in metallic materials*, Engineering Failure Analysis, Accepted.

5. MX Bastidas-Rodríguez, E Espejo, L Polanía, FA Prieto-Ortiz, *Database on mechanical fracture surfaces of metallic materials*, Data in Brief, On going.

2. Chapter 1: State of the Art

Many real world image analysis uses texture as the primary descriptor; including automated inspection, image retrieval, and medical image analysis [81]. Even though the texture could be described as a variation of intensities or colors that create certain patterns in a determined region, it is still a difficult concept [114]. Proof of this are all the different attempts made by computer vision researchers trying to explain it [113]. In consequence, a lot of methods have been proposed in the computer vision field trying to generate the texture classification or segmentation in an image.

As a particular application of texture recognition, this work will focus on fractographic analysis, which is a visual analysis on the surface of mechanical fractures, looking for specific features like propagation patterns and the fracture origin. The characterization of the fracture surface, makes it possible to show the history before the flaw, which allows us to find the causes of the failure. It is important to highlight that without the necessary information about how the fracture was generated it is hard to find the causes that produced it.

The next subsections present the state of the art for this research, which includes some textural analysis used in computer vision in the last years and a new approach in computer vision techniques called deep learning with its influence on texture analysis. Finally, a background about failure analysis and fractography.

2.1. Textural Analysis in Computer Vision

In computer vision, many different methods have been proposed to analyze texture. Most of them used the same four steps: pre-processing the images, features extraction, feature selection and machine learning task(s) (classification, segmentation, pattern recognition, among others) [81]. However, these methods are manly focused on the feature extraction step, trying to improve the results on a particular task by finding different features. These features could be divided into four main groups: *i)* Statistical features, *ii)* Structural features, *iii)* Probability models and finally, *iv)* Filter models. The first group, uses statistical features to describe texture by modeling the human visual system [81]. The most common method is the one proposed by Haralick [97], which uses the gray level co-occurrence matrix of an image, calculated in 4 directions, and extracts 26 features from them. This method presents favorable results in different researches such as in the one presented by Pham [93]. However,

as summarized by Sayadi *et al.* [101] to improve these results many other approaches have been proposed. For example, the gray level run length matrices for texture analysis [40] calculates the number of consecutive pixels that have the same gray level value, and puts it on matrices calculated in 4 directions with the run length vs. the gray level value. The gray level aura matrices [31], based their concept in the aura set and the aura measure to generate the matrix. Finally, the Local binary patterns [109] are defined as a gray-scale and rotation invariant approach for texture classification; just to name a few.

The second group uses structural features by decomposing the texture into elements denominated as primitives [81]. For example, the morphological operations [94], which uses different morphological transformations like granulometry, orientation map surface area, watersheds, etc., to discriminate textures. Fractal dimension is also used in this type of operations and contains information about the geometric structure of the object [90]. However, according to Maani [81], these methods work fine with macro-structural textures but not with micro-textural ones.

The third group is probability models. For instance, Markov Random Fields [24] are proposed as texture models, proving that the Markov random field controls the strength and direction of the clusters inside the images. The problem with this kind of method is that it is necessary to choose the correct model for a given texture and then map it to a selected probability model [81].

The final group of texture analysis applied filters on the image to get different features, this is the case of Gabor Filters defined in [56] as basically sine and cosine functions modulated by a Gaussian window to develop a local Fourier analysis.

In the past few years, researchers have been modifying these methods by adding different approaches to them, but continuing with the four basic steps mentioned above. This is the case of the approach proposed by Maani *et al.* [81], where the researchers used the magnitude of the coefficients of the 1D Fourier transform to define a local frequency descriptors. They tested their descriptor on the Outex, CUReT, and KTH-TIPS databases. Pham [30] used the Kolmogorov-Sinai entropy, measuring the amount of chaos in an image. Hadida [45] and Tang [110] applied the LBP method to two particular classification problems: the first one used 13 variants of local binary pattern and applied them to gender recognition; while the second one used LBP to classify tea leaves. Ahmadvand [28] developed two techniques to generate local parameter histograms based on Gaussian-Markov random fields descriptors applied to Brodatz and CUReT databases. Zhao *et al.* [122] proposed a new method denominated Completed Robust Local Binary Pattern (CRLBP), where the authors modified the original LBP by replacing each central pixel in a 3×3 local area by its average local gray level, obtaining a robust descriptor to noise and illumination. In [49] the concept of aura

matrices was extended having into account the imprecise nature of images and used them as a texture descriptor. Furthermore, many of these research test their textural descriptors in data sets as the Brodatz Image Database obtaining great results, but when these are tested in textures with minimal different patterns between them, these results tend to decrease. Purchasing a network able to characterize textures with a significant percentage of accuracy will allow us to solve various problems in computer vision.

2.2. Deep Learning Technique

The aim of deep architectures is to learn features hierarchies by obtaining high-level features from the composition of lower level hierarchies. This allows a system to learn features directly from the input image without depending exclusively on handcrafted features [15]. These architectures are based on the mammal brain, which appears to obtain information in different stages such as detection of edges, primitive shapes and moving up to more complex shapes [15]. Although this research field is not new, many researchers have tried to train deep multi-layer neural network. Only until 2006 Hinton at University of Toronto reported positive experimental results by using deep belief networks [53]. Since then, deep networks have been successfully applied in many different applications in machine learning [15]. Among the reasons that motivate the use of deep networks according to Bengio [15] are:

- They learn with little human intervention and with a big amount of abstraction.
- They allow representing complicated function needed in artificial intelligence applications.
- They allow learning from a big amount of data.
- They have the ability to make unsupervised learning, obtaining structure from the data.

2.2.1. Convolutional Neural Networks (CNN)

Convolutional Neural Networks are one of the first truly successful deep learning approach. In the computer vision field, many architectures have the images as input, and therefore, its neurons are arranged in a 3-dimensional space with width, height, and depth. Contrary to conventional neural nets, the neurons in a layer will only be connected to a small portion of the layer before, instead to be all neurons fully-connected [75]. In Fig. **2-1** a comparison among a regular neural network and a CNN is shown.

Convolutional nets architecture contains three main types of layers: *i)* Convolutional layer, *ii)* pooling layer and *iii)* fully-connected layer (the same of regular neural networks). The

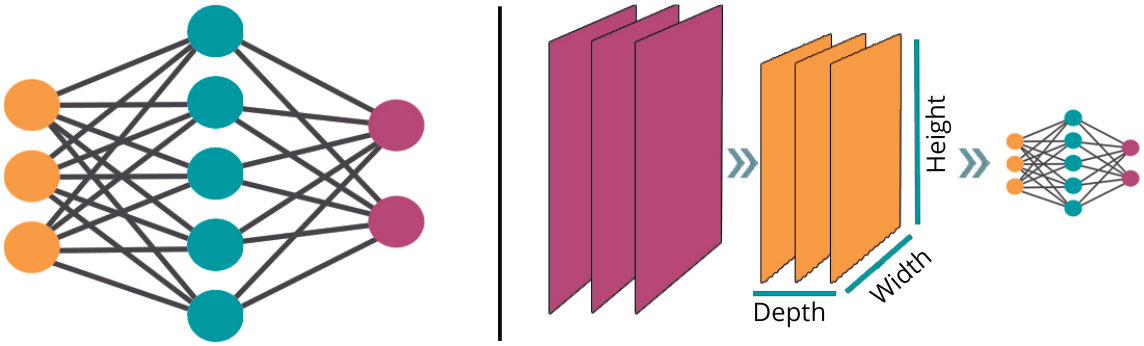


Figure 2-1.: Left: a regular three-layer neural network. Right: CNN. Every layer transforms the 3D input volume into a 3D output volume of neurons activations. The red input layer holds the image, so its width and height are the dimensions of the image, and the depth would be three due to the image channels (Image inpired by [75]).

Convolutional layer consists of a set of filters with a small 2D dimension but extended through the full depth of the input volume. In the first layer, the depth of the filter will be the number of channels in the image [75]. For each 3D filter, a 2D dimensional activation map will be produced and the depth of the output volume will be produced by stocking the entire set of activation maps in each convolutional layer [75]. Each neuron on the convolutional layer is also connected only to a local 2D region of the input layer, but fully connected along the entire depth. Fig. **2-2** presents an example of a convolutional layer. It also shows that each neuron acts like a regular neural network by performing a dot product of their weights with the input followed by a nonlinearity [75]. The convolutional layer obtains an

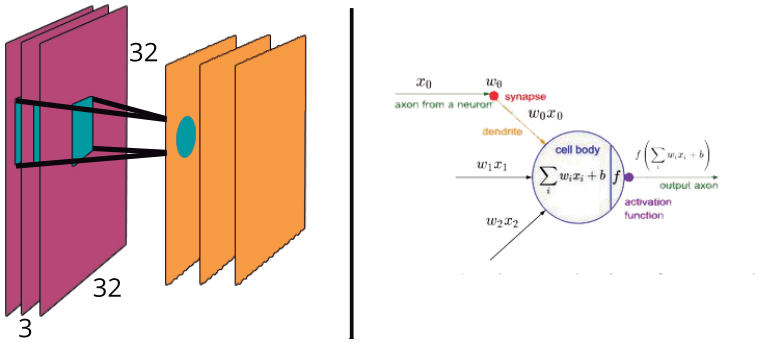


Figure 2-2.: Example of a convolutional layer. At the left: the input volume of $32 \times 32 \times 3$ is in red and an example of neurons in the first convolutional layer, each neuron is connected to a local 2D region of the input volume, but to the full depth. At the right: The neurons act like a regular neural network, but restricted in its connectivity (Image inpired by [75]).

output volume, whose depth corresponds to the number of filters used in the layer. Each

filter looks for a different feature on the input image. Its stride corresponds to the number of pixels a filter slide through the image; this means that when the stride is equal to 1, the filter will move one pixel at a time and so on. Sometimes, it is convenient to add some zeros around the input border, to control the spatial size of the output. This process is called zero-padding [75]. The pooling layer is commonly used between convolutional layers and its function consist in progressively reducing the spatial size of the representation to reduce the amount of parameters and computation in the network. These layers apply a max operation in every depth slice of the input and re-size it in the 2D space [75]. The most common pooling layer consists in filters of size 2×2 with a stride of 2 as it is shown in Fig. **2-3** where an example of pooling layer is presented. *iii)* Finally, the fully-connected layer has neurons connected to all activations in the previous layer as occurs in a regular neural network [75].

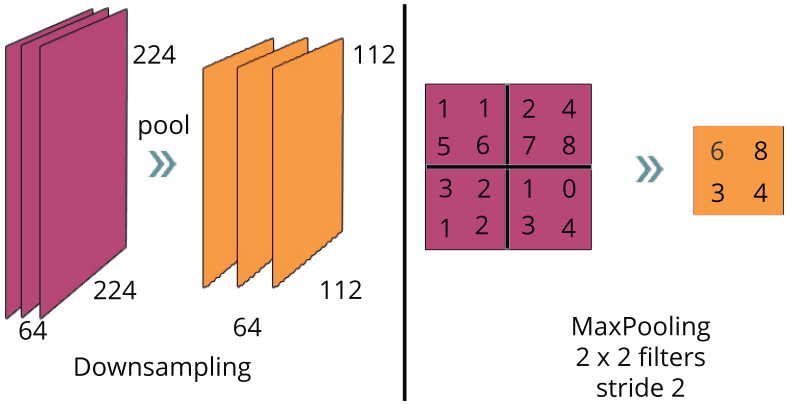


Figure 2-3.: At the left: An example of downsampling the input volume through a pooling layer. At the right: The most common downsampling operation the max pooling (Image inpired by [75]).

2.2.2. Transfer Learning and Data Augmentation

To use deep neural networks it is usually necessary to input a large amount of data in the net, this way, the network will have enough information to accomplish the classification task. However, in many applications, there is not enough data to train a network from scratch. For this reason, two methods are commonly used in order to solve this problem: *i)* Transfer Learning, which uses a pre-trained network with a big enough dataset and transfers some parameters to a new system where the input data is limited. And *ii)* Data Augmentation, which makes different transformations on the input data and, by doing this, it obtains some new and different information. These two methods are explained below in details.

Transfer learning According to Torrey and Shavlik [111], transfer learning improves a learning task by transferring the knowledge already learned from a related work. The study

of this method is motivated by the fact that humans can use previously acquired knowledge to solve new problems faster and more efficiently [91]. In image analysis, this method is used because there is not the necessary amount of data to train a deep neural network from scratch. The two most common approaches are using the convolutional network as a fixed feature extractor or fine-tuning the convolutional network [75]. In the first case, the convolutional network of the pre-trained source is taken, and the last fully-connected layer is removed; finally, this convolutional network is used as a fixed feature extractor from the dataset [75]. The second method fine-tunes the weights acquired in the pre-trained source into the target dataset; this approach takes advantage of the fact that early features of the convolutional network extract more generic features than the final layers [75].

Data augmentation In many applications the available dataset is not enough to train a classifier. For this reason, some research transform the available samples into new samples, preserving the labels. This process is called Data Augmentation (DA) [37]. Even though Data Augmentation is usually made to train very large deep networks, looking to improve the generalization error, it is not a simple process due to it involving some manual choices [37]. In practice, this process is made by hand, applying a small set of transformations for which the classifier will believe to be invariant [50]. Some DA methods could be brightness augmentation, which changes the intensity by simulating day and night conditions; horizontal and vertical shifts, which simulates the effect of different position; rotation of the image, among others.

2.2.3. Deep learning in Texture Applications

The most recent researches in deep learning have proved that it is possible to identify objects by learning features directly from the data. However, when it comes to classify textures it is not that simple. Geirhos *et al.* [41], developed several test to evaluate if CNNs are biased towards shape or texture. In their experiments, CNNs encounter more difficulties recognizing objects when they lose textural information. The authors created a stylized dataset using a style-transfer technique and they prove that when an object is stylized the network will probably classify the texture rather than the object itself. However, in their method they are not exactly classifying texture, and furthermore, they do not evaluate inner class variation on textural datasets. On the other hand, Basu *et al.* [13], argument that CNNs has had a limited success when it comes to learning textural features, because there are some limitations with the existing neural networks architectures. By using the Vapnik and Chervonenk dimension, the authors showed that handcrafted features (such as Haralick features) create a low-dimensional representation of texture databases that helps to reduce the overall error rate. Furthermore, they calculated the intrinsic dimension of some popular object recognition databases by using the maximum likelihood algorithm, and compared them with textural databases (results in Table **2-1**); concluding that texture

datasets have an inherently higher dimensional manifold compared to object recognition datasets. Therefore, they are more difficult to classify by neural networks. Finally, they point out that, to classify texture using deep neural networks, it is necessary to integrate them with handcrafted features or create a novel neural architecture that allows learning features from the input datasets resembling these handcrafted texture features.

Table 2-1.: Intrinsic dimension of some object recognition and texture datasets [13]

Object Datasets			
Dataset	MNIST	CIFAR10	DET [1]
Intrinsic Dim	9.96	15.9	17.01
Texture Datasets			
Dataset	Brodatz	KTH	KTH-2
Intrinsic Dim	34.87	43.69	54.19
Intrinsic Dim (Haralick features)	4.03	3.73	3.93

In recent years, there are a few researches on textural computer vision problems, using deep learning techniques. Hafemann *et al.* [46] classify forest species by using CNN addressing it as a texture classification problem. They used two forest species datasets, one with macroscopic images of 41 classes containing over 50 high-resolution images for each class, getting 95.77 % of accuracy against a 97.77 % achieved in the state of the art. And one with microscopic images of 112 species containing 20 images for each class, with an accuracy of 97.32 % improving on the-state-of-the-art by 5 %. The architecture used in this research consists of one input layer; two combinations of convolutional and pooling layers with 64 filters of 5×5, and the pooling layer with windows of 3×3 and stride 1; one locally-connected layer with 32 filters of 3×3 and stride 1; and finally, one fully-connected output layer.

Cimpoi *et al.* [21] proposed a describable texture dataset, which consists of a collection of 5640 real-world texture images labeled with adjectives taken from a vocabulary of 47 English words. Then, they performed two feature analysis: first they tested different handcrafted features, and secondly they used transfer learning with ImageNet in a VGG 19 architecture to extract features from the last pooling layer of the model. Then two classification methods are tested: using those features as input to an SVM classifier and inputting them into a fisher vector transformation before the SVM classifier. Finally, they concluded that CNN features is a good texture descriptor to achieve texture segmentation and classification. However, according to Andrearczyk and Whelan [6], this method is computationally expensive, so they developed a simple network architecture called Texture-CNN (T-CNN), which obtains an energy measure from the last convolutional layer before connecting it to a fully connected layer. It was successfully applied in the biomedical texture image analysis performed by the same researches [5] where they test this approach by using three datasets of liver tissues

images.

Finally, Hafemann *et al.* [47] presented a transfer learning method, however, their transfer method does not transfer the weights from the source, as previous works have done, but by performing a forward propagation of the target dataset, obtaining a new representation for each sample, and performing the final classification by using a support vector machine. The source dataset contains 41 classes, and around 65 GB images with resolution 3264×2448; the target dataset is the Brodatz dataset highly used in texture analysis.

2.2.4. Wavelet Analysis and Convolutional Neural Networks

There are some works that incorporate wavelet representations into CNNs. Oyallon *et al.* [89] proposed a hybrid network which replaces the first layers of ResNet by a wavelet scattering network. This modified ResNet resulted in comparable performance to that of the original ResNet but has a smaller number of trainable parameters. Williams *et al.* [116] took the wavelet sub-bands of the input images as a new input and processed them with CNNs. In a different work [117], they showed a wavelet pooling algorithm, which uses a second-level wavelet decomposition to subsample features. Lu *et al.* [80] addressed the organ tissue segmentation problem by using a dual-tree wavelet transform on top of a CNN. Cotter and Kingsbury [23] also used a dual-tree wavelet transform to learn filters by taking activation layers into the wavelet space.

Recently, Fujieda *et al.* [39] proposed a wavelet CNNs (WCNNs), which were built upon the resemblance between multiresolution analysis and the convolutional filtering and pooling operations in CNNs. They proposed a CNN similar to DenseNet, but the Haar wavelets (which are commonly used in multiresolution analysis) where used as convolution and pooling layers. These wavelet layers were concatenated with the feature maps produced by the succeeding convolutional blocks. This model is more interpretable than CNNs since the wavelet layers generate the wavelet transform of the input. However, the use of a fixed wavelet (Haar) is likely suboptimal as it restricts the adaptability and cannot leverage data-driven learning.

2.3. Failure Analysis

The visual observation of the surface of a fracture gives information about the characterization of the rupture of the element, like the crack propagation mechanism or the fracture's origin, the tenacity of the material, the tension configuration, distribution, and magnitude and the chemical environment. This observation allows us to make a qualitative estimation of the acting tension magnitude. There are some patterns that show if the acting forces of the failure were high or low. The determination of the fracture's origin is one of the main

contributions because it is created inside the piece, for instance, in some structural imperfection, it is required to get better materials, but if a fissure is placed on the component surface, it is necessary to increase the superficial resistance of the element [57].

Fractography is a discipline of materials science, and it studies the topographical features of each failure mode on the surface of the fracture [34]. This study is made by a visual inspection, the first steps of this analysis could be performed by a well-trained human eye and, in case of doubt, magnifiers could be used to help, such as a stereoscopic or a SEM [4]. The use of an experienced person's eye could extract plenty of information such as the presence of the crack, corrosion marks, deformations, among others [102]. The initial general procedure is to photograph the entire fractured part. Here, it is important to capture high-resolution and clear contrast images. On a macroscopic scale, fractures can generally be classified as ductile, brittle, fatigue fractures and fractures resulting from both stress and environmental conditions [102]. Microscopical analysis allow to observe finer details in the fracture's surface [102]. This analysis also takes into account the kind of force that the element experiences, the work environment, and the operation temperature, among other variables.
The mechanical fractures occur when a mechanical element splits into two or more fragments

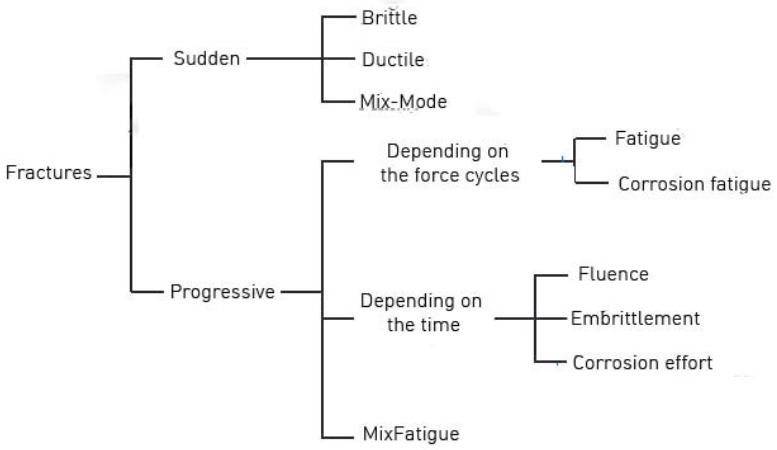


Figure 2-4.: General classification of more common fracture failure modes in metallic materials, according to its sudden or progressive nature [34]. The fractures to work with are framed in a square.

following three steps: *i)* nucleation of one or several cracks, *ii)* propagation of the crack(s) and *iii)* fracture of the element [34]. Depending on the speed at which these phases take place, fractures can be classified into sudden and progressive. They are sudden if the crack propagation occurs between 0,2 to 0,4 times the speed of sound in the material and they are progressive if the crack propagation is slow for instance 1 mm/day. It is possible to match up the majority of failure modes in metallic materials through the basic classification between

sudden and progressive. The first ones are classified depending on their behavior: ductile, brittle or mixed mode; while the second ones are classified depending on what produces the nucleation and the crack propagation, whether it is the force cycles (generating behaviors of fatigue and corrosion fatigue), the time (generating slow fluency, embrittlement and corrosion effort) or mixed modes [34]. These fracture types can also be classified in mechanical fractures, fractures assisted by the environment or fractures assisted by the temperature. The classification of the fracture modes can be seen in Fig **2-4**.

The distinctive marks and textures on the fracture's surface are presented at following [34]:

Distinctive Marks (Figure **2-5**):

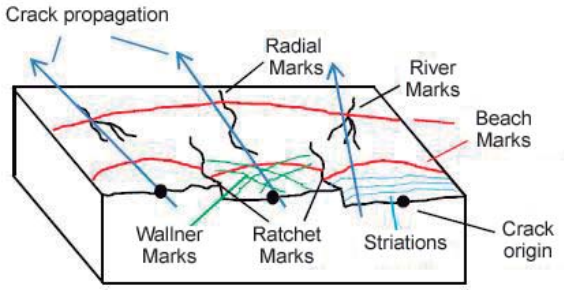


Figure 2-5.: Common marks that can be find on fracture surface of mechanical elements[34].

- The direction of propagation marks. These marks are aligned with the local direction of the crack growth on the fracture's surface. Through them, two or more crack surfaces are split and located at different height levels. Some of those marks are: *i* Saw-tooth or Ratched marks, which are generated at the origin of the fracture and it splits each mark between two adjacent independent cracks, creating the saw-tooth. *ii* River marks, which look like a riverbed and are caused by the fact that during the crack propagation, several planes are split up over the surface of the fracture. And finally, *iii* Radial marks, which differ from the river marks in that they only split the surface into two planes [34].
- Front crack position marks. They show the location where, at some point, the crack front was. They are perpendicular to the local growth direction and the direction of propagation marks. These marks are: *i* Beach marks, generated by the speed growth of the crack, by changes in the effort states or by environmental actions on the front of the crack, among others. And *ii* Striations, which are formed with every crack growth cycle and, for this reason, are separated by spaces of some micrometers [34].
- Wallner marks or interaction with deformation of waves. These are formed due to the interference between the crack front (during an unstable crack propagation) and the

deformation waves produced by the cracking phenomenon (when they are reflected by free surfaces, they turn back into the crack zone or produced by applying forces at high speed). These waves create small oscillations on each side of the main crack propagation, leaving marks very similar to beach marks [34].

- Material change marks. They are formed when the crack propagation inside a piece go through a change from a high toughness material to one with little toughness or vice versa. Or when there is an effective change in the material composition [34].

Distinctive Textures:

- Granular texture. Shows a similar aspect to sand grains and tends to be shiny on metals. It is formed from polycrystalline materials [34].

- Fibrous texture. It is more opaque and rough than the granular texture [34].

- Smooth texture. It is not very rough and in metals can be shiny [34].

- Flat texture. Under visual inspection, this texture is close to the flatness concept [34].

Becker [14] presented the Table **2-4** showing some general macroscopic and microscopic fractographic features. He pointed out that it is possible to misinterpret the fracture surface's features, and, for this reason, the failure analysis process should use all the information available to reach a conclusion as to the cause of the failure. Using just a single method i.e. Macroscale examination without microscale examination, would lead to an incorrect conclusion.

The fractographic analysis starts with a visual observation of the macroscopic features on the fracture's surface, followed by the stereoscopic analysis, which can be used to confirm previous observations [57]. Developing an automatic algorithm for segmentation and classification of fractures would contribute to this analysis and, for this reason, would prevent the generation of future failures that would lead to economic and human losses in industries.

2.3.1. Computer vision techniques used in fractographic applications

An expert system is a computer program that emulates the behavior of a human expert and consists in one of the most practical applications of artificial intelligence [76]. One of the main reasons to use this kind of systems is that it is sometimes not possible to find an available expert or that it may be too expensive to hire one. Optimizing time, improving productivity and facilitating workers training are also very important reasons for such developments [76].

Computer techniques are essential to process fractographic digital images, since it allows it to obtain more information in less time and achieving better results than traditional ways [17]. These techniques have been used in failure analysis and materials science for different applications. For example, to estimate the crack growth rate (CGR) through textures of surfaces acquired by SEM, obtaining a fracture surface reconstruction through sub-images previously selected process called auto-shape analysis [72]. In many cases, the images of some specific material are analyzed, as in the work of Camargo *et al.* [17], whom evaluate the API5L-X52 steel, looking for the characterization of micro-holes morphology. Laushmann [73] and Kosarevych [66] explained fractographic texture methods with the aim of reconstructing the history of fatigue fracture, using the relationship between textural features on the fracture's surface and the CGR. For 3D analysis, Kolednik *et al.* [63] used the shape reconstruction method through stereo images, applying an algorithm able to find homologous points between two images. The stereo algorithm recovers the shape through multiple images from the same scene, using a fixed viewing direction and different light source [95]. The use of the information given by three-dimensional images allows it to determine relevant quantitative features, such as the local roughness and leads to a deeper comprehension of the failure mechanisms [62]. The reconstruction process is applied with the final purpose of obtaining the 3D mesh for each region of interest (ROI) [62].

For the recognition or classification of fracture's surfaces, Komenda *et al.* [58] worked with three kinds of sintered steel looking to recognise the morphology, measures and apparent porosity on the fracture's surface. They got their images with SEM, and they analyzed the fatigue fracture mode. Among the obtained features are the morphological gradient on gray level images, and they took into account the nearest local neighbors to do the classification. In [64], they create a program able to classify between six different morphologies of fracture's surfaces, using as descriptors, the Fourier spectrum, the co-occurrence matrix (where they extract three features: contrast, variance, and correlation), and 3D topology. The data set acquisition is made using SEM, and the cluster method used the K-means algorithm. However, the classification percentage does not get over the 80 % on the majority of morphologies.

2.3.2. Deep Learning methods used in fractographic applications

Deep learning have become a highly immersive field in different applications and topics; some studies show fractographic analysis using deep learning approaches. Most of them focus on one kind of failure mode or type of material, such as the approach presented by Mahto [82], who analyzed the fracture mode generated on a ductile iron material by a tension test using various heat. Konovalenko *et al.* [65] used CNNs to analyze the rupture surface of a titanium alloy, detecting dimples on SEM images, and they arrived to the conclusion that CNNs perform similarly to stat-of-the-art handcrafted features in this application. Tsopanidis *et*

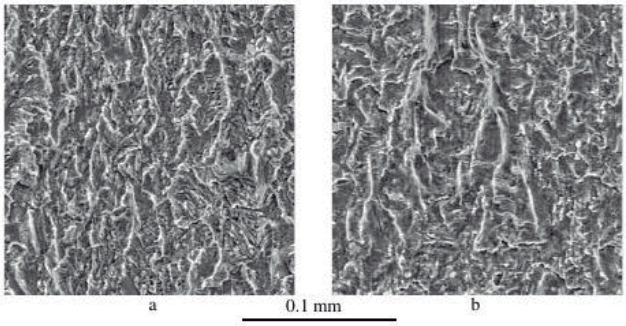


Figure 2-6.: Samples of typical texture on a fatigue fracture surface. Low (a) and High (b) CGR. AISI304L stainless steel [72].

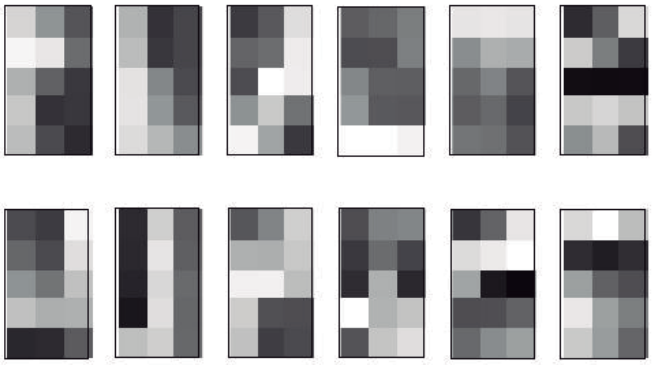


Figure 2-7.: Basic subimages selected through training images [72].

al. [112] analyzed transgranular and intragranular brittle fractures of two ceramimc materials by using CNNs and a semantic segmentation algorithm in SEM images. They concluded that the proposed method had a high performance, even when the model is only trained in one of the two ceramic materials. Wang *et al.* [108] used a DenseNet model pre-trained with ImageNet and fine-tuned features to localize the fracture's origin in fatigue failure mode of SEM images; they recognized that the proposed method do not overcome other object detector approaches, though they highlighted that results are promising and proposed several future work modifications. Haider [48] also analyzed fatigue failure modes. He used Monte Carlo simulation output as input to the deep learning model of six hidden layers to determine the time associated to the crack growth increments. Feng *et al.* [38] studied the solidification cracking susceptibility, and by using deep learning pre-training and fine-tuning methods in a small dataset (478 datapoints), he obtained better results than by using an SVM and a shallow neural network in the same data.

In opposition to state-of-the-art studies, the approach presented here does not focus on one type of material or failure mode, and presents a generalized model. Furthermore, most of

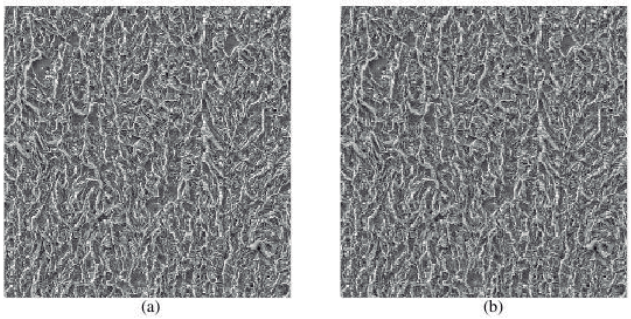


Figure 2-8.: Comparison between the original image (a) and a reconstruct image (b) (256 x 256 pixels sections) [72].

the researches focuses only on SEM images, but in this work we focus on two databases, a real-scale one and one taken with SEM.

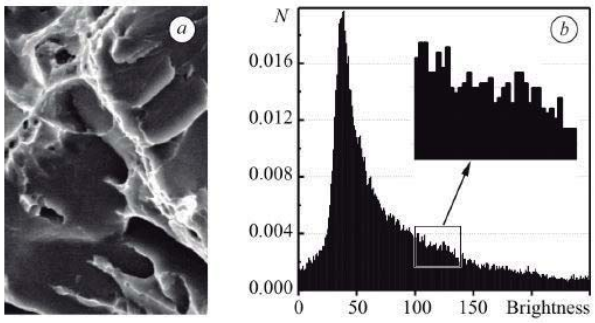


Figure 2-9.: (a) Fractogram of a typical ductile fracture of bridges between fragments of brittle cleavage fracture in neighboring areas of the fracture (b) and brightness histogram of this image. [66].

2.3.3. Texture analysis and histogram of a surface fracture image

With the texture methods, it is possible to extract information from the whole image, to make a statistical estimation, or to obtain a parameters model through the gray-scale image [73]. Currently, some of them have been developed to obtain the history of the fracture's reconstruction [66]. They use histograms or the intensity pixel information to reconstruct an image or to segment it. For this method, it is appropriate to use SEM with a zoom between 30× and 500× [72].

Among the qualitative fractography of fatigue fracture, the traditional source of information about the crack growth are the striations. In the Figure **2-6** are shown two samples of the typical textures for this mode of fracture, and it is possible to observed a high and low crack growth. To calculate this CGR, the authors made an auto-shape analysis, which consists

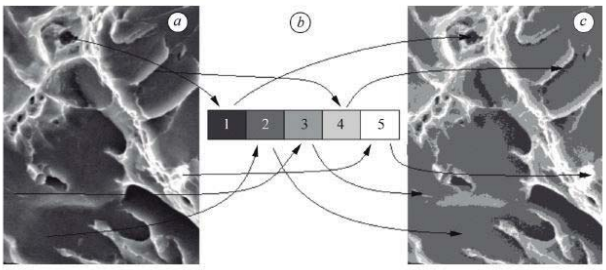


Figure 2-10.: Scheme of a fractographic image conversion (a) using shine levels as labels (b) Inside a segmented image (c) [66].

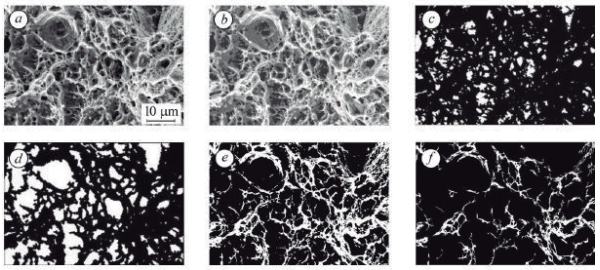


Figure 2-11.: Successive transformation of an input fractographic image (a), Segmented images as result of pixel fragmentation inside five subsets (b), with isolation of typical elements (c), dimples formed (d), bridges formed between them (e), and edge dimples conglomeration (f) [66].

in decomposing the testing images set through special small sub-images (Figura **2-7**). This way, every image is adjusted to a linear combination of those. In Figure **2-8** it is possible to see the comparison between the original image and a reconstructed image. However, They do not mention the existing error between the images. The auto-shape analysis method is simple, easy to implement and computationally fast, obtaining results that can be compared with other textural methods used in fractography and fatigue features, some of them more complicated and with more time consumption [72].

In Figure **2-9** (a) the elements of a transgranular brittle fracture are shown in gray-scale images. Due to the fact that the fracture relief is almost the same, with a small shiny gradient, the histogram for this image is unimodal [66]. The essential step in the image segmentation method using the histogram is to determine a finite set of local extremes. As a rule, during the isolation of the histogram set, the existence of a maximum local and a minimum local is necessary [66]. In Figure **2-10**, the segmentation of a fracture surface using the image histogram can be observed. After choosing, in the image, a point that in luminosity belongs to a subset, a binary image was formed in which these points are part of the object, and the rest of the points are part of the background. With these new images, it is easy to calculate

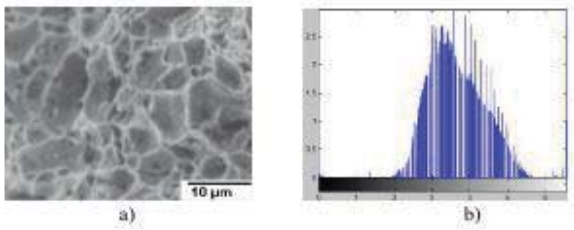


Figure 2-12.: Fractograpgy of a metal welding (a) and the image histogram (b) [17].

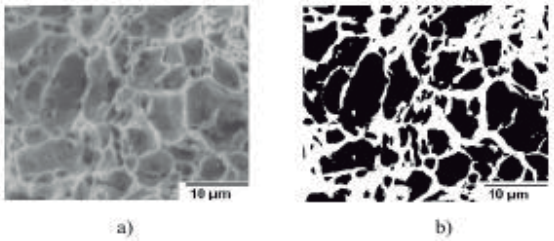


Figure 2-13.: Fractography of a metal welding; Original image (a) and segmented image (b) [17].

the qualitative features of the objects, for instance: area, perimeter and orientation [66]. In Figure **2-11** a segmentation sequence of the image is shown, and this illustrates the successive transform of an input image in the segmented images.

In Figure **2-12**, a fractographic grey-scale image of a ductile fracture with micro-holes morphology is shown. This image was segmented with the Otsu method (Figura **2-13**) and its histogram. This segmentation was made in order to observed the growth of micro-holes density and the reduction of their diameter as consequence of age in a welding micro-alloyed steel [17].

In Figure **2-14**, a fiber extraction is shown through a fatigue fracture image. This extraction was made with an image normalization looking for similar textures. The procedure was made by detecting the edges convolving the original image with the following mask [73]:

$$U = \begin{bmatrix} -1 & -3 & 9 & -3 & -1 \\ -1 & -3 & 9 & -3 & -1 \end{bmatrix}.$$

2.3.4. Fractographic classification in metallic materials by using computer vision

In our previous work [98], it was decided to create a classification system with the support of my adviser and co-adviser. This system allow us to classify fracture surfaces with three failure

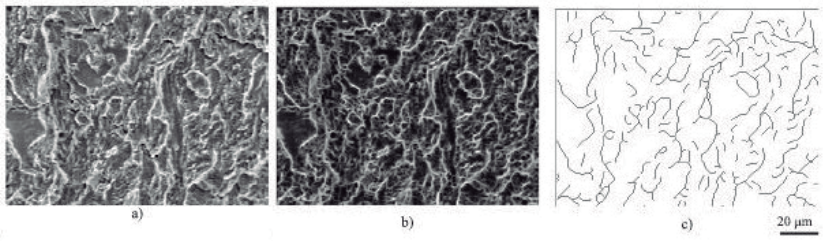


Figure 2-14.: Fiber extraction in a fatigue fracture surface image; (a) normalized image (400 X 300 pixels), (b) fiber detection, (c) significant fiber traces [73].

Table 2-2.: Results with the classification percentage of accuracy (ACC), sensibility (TPR) and specificity (SPC) for each mode of fracture according to ANN classifier by using 2D images.

Fracture	ACC	TPR	SPC	F1-score
Ductile	80.64 %	63.88 %	91.23 %	0.719
Fatigue	84.95 %	91.66 %	81.74 %	0.797
Brittle	82.8 %	68.51 %	88.64 %	0.698
Average	82.79 %	74.7 %	87.2	0.74

modes: *i)* ductile, *ii)* brittle and *iii)* fatigue. As computer vision methods, we use an approach obtaining 2D images in real scale [12]. The texture was the main descriptor and the features that we used were Haralick's features [97, 93] (obtained from the gray level co-occurrence matrix in 2D [59]). The texture energy laws [2], calculated from the convolution between the images with energy masks [55]; and, finally, the fractal descriptor [90], through the fractal dimension [77, 26], which define the object in terms of occupation and self-similarity [11]. Furthermore, two classifiers were analyzed: artificial neural networks (ANN) [92, 3] and a support vector machine (SVM) [1, 106, 105].

In the 2D analysis, the obtained results are very favorable [12], achieving the aim of classifying the fractures with an accuracy percentage over the 80 % in the three fracture modes studied and a macro f1-score of 0.74 (Table **2-2**). We worked with two classification algorithms commonly used in computer vision, artificial neural networks (ANN) and a support vector machine (SVM). Their performances were evaluated through the ROC space [36] and the ICSI and F metrics [70], where we concluded that the ANN classifier presents better results than the SVM. Additionally, the input that generates each feature group was analyzed individually and by pairs, where we can see that the SVM is more efficient than ANN in particular cases of fracture modes. However, the ANN classifier presents a better global performance than SVM for the three studied fracture modes (Table **2-3**).

Finally, the algorithm with 2D images achieves a classification of 77.4 % of all the pieces and is comparable with the results obtained by a human expert on the topic, which classi-

Table 2-3.: Performance evaluation according to the fracture mode and the classifier (GLCM: Haralick, LE: energy lows y D: Fractal Dimension).

Fracture mode	Classifier	Parameter	ICSI	F
Brittle	ANN	**GLCM+LE+D**	**0.397**	**0.698**
		GLCM	0,279	0,6171
		D	-0,14	0,4516
	SVM	**GLCM+LE+D**	**0.119**	**0.535**
		D	0.7721	0.86
		LE	-0.085	0.447
Fatigue	ANN	**GLCM+LE+D**	**0.621**	**0.797**
		GLCM+LE	0,5	0,74
		D	-0,0238	0,439
	SVM	**GLCM+LE+D**	**0.283**	**0,515**
		GLCM	0,81	0,90
		LE+D	-0,026	0,48
Ductile	ANN	**GLCM+LE+D**	**0.460**	**0.719**
		GLCM+LE	0,46	0,071
		LE	0,086	0,51
	SVM	**GLCM+LE+D**	**0.247**	**0.614**
		GLCM+D	0,32	0,65
		D	-0,017	0,4406

fication percentage is between the 70 % and 87 %. This was achieved by the expert without an access to the real pieces, just by looking at the images. The pieces that were not correctly classified showed similar features to another kind of fracture or the surface was too small to be analyzed.

Table 2-4.: Microscale fractography features [14]

Mark/Indication	Implication
Visible distortion	Plastic deformation exceeded yield strength and may indicate instability (necking, buckling) or post-failure damage.
Visible nicks or gouges	Possible crack initiation site.
Fracture surface orientation relative to component geometry and loading conditions.	Helps to separate loading modes I, II, III Identifies macroscale ductile and brittle fracture.
Both flat fracture and shear lips present on fracture surface	Crack propagation direction parallel to shear lips, Mixed-mode fracture (incomplete constraint)
Tightly closed crack on surface	Possible cyclic loading, Possible processing imperfection, e.g., from shot peening, quench cracks
Radial marks and chevrons (v-shape)	Point toward crack initiation site, Show crack propagation direction.

Mark/Indication	Implication
Crack arrest lines (cyclic loading)(beach marks, conchoidal marks)	Indicates cyclic loading, Propagation from center of radius of curvature, Curvature may reverse on cylindrical sections as crack propagates
Ratchet marks	More likely in cyclic loading, Indicates initiation site(s)
Adjacent surface and or fracture surface discoloration	May indicate corrosive environment, May indicate elevated temperature
Oxidized fingernail on fracture surface	Possible crack initiation site
Fracture surface reflectivity	Matte: ductile fracture or cyclic loading, Shiny: cleavage likely, Faceted (bumpy) and shiny; intergranular fracture in large grain size.
Fracture surface roughness	Increase in surface roughness in direction of crack growth (may be affected in bending by compressive stressed region when crack moves into this region), smooth region plus rough region in direction of growth—cyclic loading, rough matte fractures are ductile, may indicate transition from fatigue crack growth to overload.
Rubbing (general)	may indicate vibration, may show final direction of separation, swirl pattern indicates torsion.
Rubbing (localized)	may indicate crack closure in cyclic loading, May obliterate beach marks.
Deformed draw marks, rolling scratches	If twisted, indicates torsion loading.
Machining marks (normal to axis of component)	Not distorted in torsion loading.
Variable roughness of fracture edge	In brittle bending, rough side is tension side.
Dimpled fracture surface	Ductile overload fracture at this location.
Faceted fracture surface	Brittle cleavage fracture, possible SCC fracture, may be low $\triangle K$ fatigue.
Intergranular with smooth grain boundaries	Likely either improper thermal processing or environmental assisted fracture (high temperature, corrosive environment).
Intergranular with dimpled grain boundaries	Decohesive rupture—fracture at high fraction of melting point, Possible improper processing creating denuded zone adjacent to grain boundary.

Mark/Indication	Implication
River pattern or fan pattern	Cleavage fracture; crack runs down river; fan rays point to initiation site within a grain.
Tongues	Twinning deformation during rapid crack propagation.
Flutes on transgranular fracture surface	Indicates corrosive environment and ductile fracture, Crack propagates parallel to flutes.
Striated or ridged fracture	Cyclic loading fatigue striations; Constant spacing, constant stress, amplitude; variable spacing, variable stress amplitude or block loading, Striated surface caused by second phases in microstructure.
Grooves or flutes	SCC, TGF
Artifacts (mud cracks)	Dried liquid on surface. May indicate incomplete cleaning of surface. If in the as-received condition, may indicate fluids from service and may indicate SCC conditions. Material should be analyzed.
Artifacts (tire tracks)	Common in cyclic loading, due to entrapped particulate matter.

3. Chapter 2: Deep learning Combined with Handcrafted Features

Summary

Deep Learning is a machine learning technique that learns features directly from input data and has achieved outstanding results in object classification. However, when it comes to texture classification, the results of deep learning are not as good as in other classification tasks. In this chapter, a deep learning method combined with handcrafted features is presented. This method uses the calculation of three common handcrafted features, namely, *i)* Haralick's features, *ii)* Fractal Dimension and *iii)* and Local Binary Patterns (LBP), taking as input different levels of a common pre-trained deep learning architecture denominated as VGG.

3.1. Handcrafted features

Three sets of handcrafted features widely used in texture recognition were selected: *i)* Haralick features [97], *ii)* fractal dimension [11] and *iii)* LBP [123]. The first two sets of features were already used in failure mode classification in [12] and the third set was already used for texture classification [123].

3.1.1. Haralick' features

One of the most popular methods to find features in texture analysis, it is called the Haralick's approach [97]. This method uses the grey level co-occurrence matrix of the image to obtain several features trough the relationship among the pixels. The co-occurrence matrix is the spatial relationship among two pixels located at certain distance d and an angle θ form the reference pixel. This matrix is not symmetric, because it does not take into account the relationship given by $-\theta$; To solve this issue, it is necessarily to sum its transpose matrix [59]. This concept is illustrated in Fig **3-1**. Finally, a probabilistic matrix is obtained by $p_{ij} = V_{ij} / \sum_{i,j=0}^{N-1} V_{ij}$, where i is the number of rows, j is the number of columns, V_{ij} is the value of the cell at position (i, j) and N is the total number of rows or colums in the matrix [59]. An example of the calculation of this matrix it is shown below, taking $d = 1$

and $\theta = 0°$

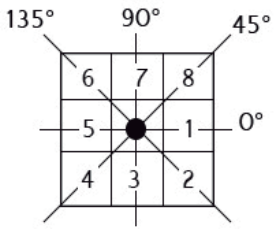


Figure 3-1.: Nearest neighbor pixels to a reference pixel in direction 0°, 45°, 90° and 135°, pixels 1,8,7 and 6 respectively, and direction $-\theta$, pixels 5, 4, 3 and 2.

For example, given an input image:

$$Test\_Matrix = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{pmatrix}.$$

The possible pixel combinations in the image are:

$$Cooccurrence\_Matrix = \begin{pmatrix} (0,0) & (0,1) & (0,2) & (0,3) \\ (1,0) & (1,1) & (1,2) & (1,3) \\ (2,0) & (2,1) & (2,2) & (2,2) \\ (3,0) & (3,1) & (3,2) & (3,3) \end{pmatrix} = \begin{pmatrix} 2 & 2 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The previous matrix is not symmetric, so it does not take into account the $-\theta$ directions. To turn it into a symmetric matrix, its transpose matrix is sum it. In this particular example the directions $\theta = 0°$ and $-\theta = 180°$ are obtained. Finally, the probabilistic matrix is generated.

$$p(i,j) = \begin{pmatrix} 0,166 & 0,166 & 0,083 & 0 \\ 0 & 0,166 & 0 & 0 \\ 0 & 0 & 0,25 & 0,083 \\ 0 & 0 & 0 & 0,083 \end{pmatrix}.$$

After generating the co-occurrence matrix, the 11 features suggested by Haralick [97] are obtained. Where: N_g is the total number of grey levels in the image, $p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1_{i+j=k}}^{N_g} p(i,j)$,

$k = 2, 3, ..., 2N_g$ and $p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1_{|i-j|=k}}^{N_g} p(i,j)$, $k = 0, 1, ..., N_g - 1$.

- Angular second moment: It is also known as uniformity of energy. The greater the similarity between the pixels, the greater this moment [93], $f_1 = \sum_i \sum_j p(i,j)^2$.
- Contrast: It Measures the variation among the intensity of grey levels [93], $f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{Ng} p(i,j) \right\}_{|i-j|=n}$.
- Correlation: It shows the linear dependency among the values in the co-occurrence matrix. Given a reference pixel, it is obtained 1 if they are perfectly correlated and 0 if they aren't correlated at all [93], $f_3 = \sum_i \sum_j ((ij)p(i,j) - \mu_x \mu_y)/(\sigma_x \sigma_y)$, where μ_x, μ_y, σ_x and σ_y are the mean and the standard deviation of p_x y p_y.
- Variance or sum of squares: It is the dispersion measure among the neighbor pixels and the mean [93], $f_4 = \sum_i \sum_j (i - \mu)^2 p(i,j)$.
- Inverse difference moment: It measures the homgeneity on the image [93], $f_5 = \sum_i \sum_j p(i,j)/(1 + (i-j)^2)$.
- Sum average: $f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i)$.
- Sum Variance: $f_7 = \sum_{i=2}^{2N_g} (i - f_6)^2 p_{x+y}(i)$.
- Sum Entropy: $f_8 = -\sum_{i=1}^{2N_g} p_{x+y}(i) \log p_{x+y}(i)$.
- Entropy: It is defined as the quantity of chaos or disorder [93], $f_9 = -\sum_i \sum_j p(i,j) \log p(i,j)$.
- Difference variance: $f_{10} = \sum_{i=0}^{N_g-1} (i - f'_{10})^2 p_{x-y}(i)$, where $p_{x-y}(k) = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} p_{d,\theta}(i,j)$, $k = |i-j| = 0, 1, 2..., (N_g - 1)$ y $f'_{10} = \sum_{i=0}^{N_g-1} i p_{x-y}(i)$.
- Difference entropy: $f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) \log p_{x-y}(i)$.

Following the recomendation of Haralick [97], in this work, the co-occurrence matrix was calculated with 256 grey levels and in four possible directions, $\theta = 0°$, $45°$, $90°$ y $135°$, with a distance $d = 1$. The final features used are the average and the range for the four directions in each feature, as it is shown in Eq. 3-1 and 3-2.

$$Average(d) = \frac{1}{N_\theta} \sum_\theta T(d,\theta). \tag{3-1}$$

$$Range(d) = \max_\theta [T(d,\theta)] - \min_\theta [T(d,\theta)]. \tag{3-2}$$

Where $T(d,\theta)$ is the feature value and N_θ is the total number of directions.

3.1.2. Fractal Dimension

Features obtained from the fractal dimension *(D)* were used, by computing Eq. 3-3. This feature extracts important information about the geometric structure of the image [90]. While the topological dimension is given by an integer value, that describes the number of dimensions that an object is immersed (1D, 2D , 3D), the fractal dimension uses real values to described an object in terms of space occupancy and self-similarity [11]. One of the most popular methods to obtain it is called *Differential box counting (DBC)*, which assumes the grey levels on the image as a 3D space, having the dimensions x and y as rows and columns of the image, respectively; and the dimension z as the grey level on the pixel [77].

$$D = \frac{\log(N_r)}{\log(1/r)}, \tag{3-3}$$

Differential box counting method Given an image of size $M \times M$ pixels, it is divided in a grid of size $s \times s$ pixels and a radius given by $r = s/M < 1$. If G is the total number of grey levels, then $s' = G \cdot s/M$ corresponds to the grey level units in the z direction for each box. Then each box has a size of $s \times s \times s'$. If the minimum and maximum grey level in the (i, j) cell falls in the boxes number k and l, respectively, then, the contribution of the (i, j) cell to N_r is given by $n_r = l - k + 1$. Finally, $N_r = \sum_{i,j} n_r(i, j)$ [77]. Fig. **3-2** shows an example of the calculation of n_r with $s' = s = 3$. Supposing that the maximum grey level is equal to 8 and the minimum grey level is equal to 2 in the cell, then those level falls into the $3rd$ and $1st$ boxes, respectively, and $n_r = 3 - 1 + 1$.

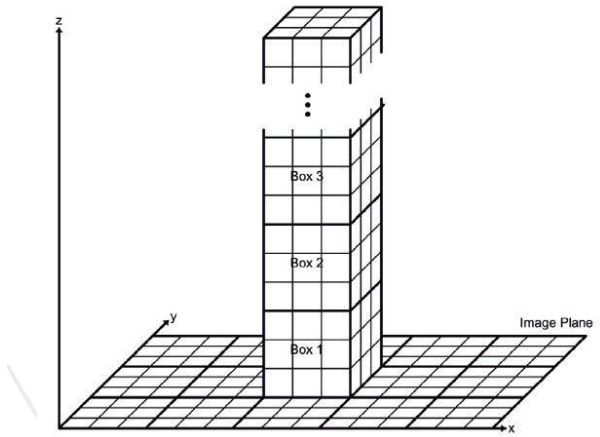


Figure 3-2.: Example of the calculation of DBC method for one of the boxes with size $s \times s \times s'$ con $s' = s = 3$ [77].

Fractal Dimension on modifications of the original image Additionally to the calculation of the fractal dimension in the original image, D is obtained through two modifications of the

original image, namely, high grey level and low grey level value images, defined by Eq. 3-4 and 3-5; and one image obtained with the average of four neighboring pixels to a reference pixel, Eq 3-6 [26]. If two images I_1 and J_1, has the same value of D, then its high grey level values images I_2 and J_2, and low level value images I_3 and J_3 do not has the same roughness and then, D will have different values in these modify images [26].

$$I_2(i,j) = \begin{cases} I_1(i,j) - L_1, & \text{If } I_1(i,j) > L1 \\ 0, & \text{otherwise} \end{cases}. \tag{3-4}$$

$$I_3(i,j) = \begin{cases} 255 - L2, & \text{If } I_1(i,j) > (255 - L_2) \\ I_1(i,j), & \text{otherwise.} \end{cases}, \tag{3-5}$$

$$I_4(i,j) = \frac{1}{4}\sum_{i=1}^{2}\sum_{j=1}^{2} I_1(i,j), \tag{3-6}$$

where, $L_1 = g_{min} + av/2$ and $L_2 = g_{max} - g_ave/2$, with g_{max}, g_{min} and g_ave being the maximum, minimum and average values on the original image.

3.1.3. Local Binary Pattern (LBP)

Since they were proposed in 1990 by Ojala *et al.* [109], these features are highly used in texture analysis problem. They are based on the binary relationship among the pixels [109]. Taking the center pixel as reference and using it as a threshold for the pixel neighbors; if the intensity of the center pixel is greater or equal to its neighbor, then the neighbor assumes a value of 1, in the other case it assumes a value of 0. After that it is necessary to calculate the local binary pattern for the center pixel, starting for any neighbor pixel (being consistent for the next pixel LBP calculation and for all the images in the dataset). Fig. **3-3** presents an example of LBP calculation.

Mathematically, the LBP code is computed by comparing a given pixel with its neighbors [123], presented in Eq. 3-7, where g_c corresponds to the grey value of the center pixel and g_p is the value of its neighbor. P is the total number of pixels and R is the radius of the neighborhood

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p,\ s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3-7}$$

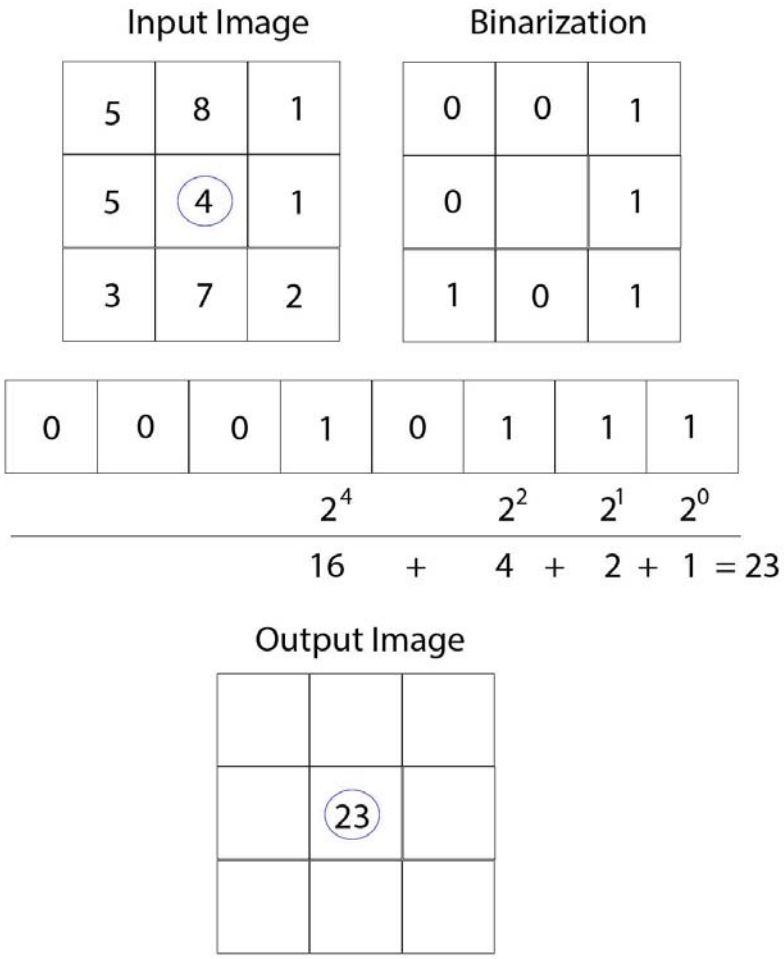


Figure 3-3.: LBP calculation example. The central pixel is the reference pixel, that acts as threshold for the neighboring pixels.

The last step consists in obtaining an histogram with 256 possible values, given by Eq. 3-8, where the maximal LBP pattern value is given by K [123].

$$H(k) = \sum_{i=1}^{I}\sum_{j=1}^{J} f(LBP_{P,R}(i,j),k),\ k \in [0,K]$$
$$f(x,y) = \begin{cases} 1, & x = y \\ 0, & Otherwise \end{cases}. \tag{3-8}$$

3.2. Very Deep Convolutional Network for Large-Scale Image Recognition (VGG)

After AlexNet [68] won the ImageNet [27] challenge in 2012, which consists on classifying among one thousand classes, several architectures have been constructed. In 2014, the VGG [103] architecture improved the results achieved by its predecessors and served as a basis to develop other architectures such as ResNet and DenseNet. Furthermore, there are only two variations of the architecture tested on ImageNet and the one that achieves better results on this database is the VGG with 19th layers, which makes it an easy test architecture for the model presented in further sections. This architecture, presented in the Fig. **3-4** consists in five sequential convolutional blocks, separated by a maxpooling layer. The first two blocks contain two convolutional layers with the same amount of filters, 64 and 128, respectively. The three next blocks contain three or four (depending on whether it is a 16th or 19th model) convolutional layers with the same amount of filters, 256, 512, 512 for each block, respectively. The kernel size for each convolution layer is 3×3. After the convolutional blocks a set of three fully-connected layers of 4096, 4096 and 1000 neurons is applied.

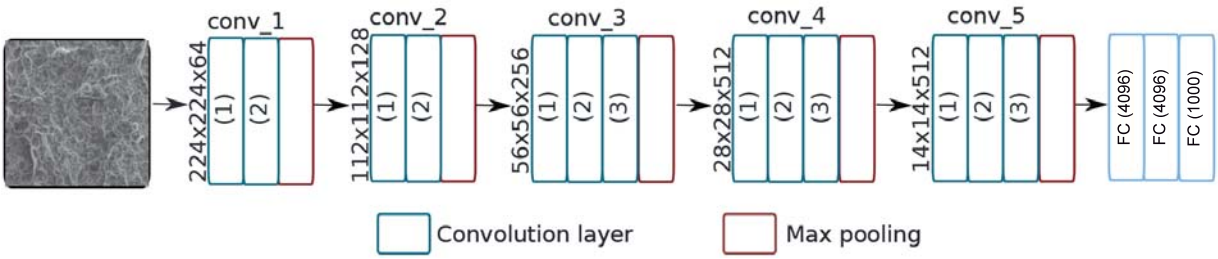


Figure 3-4.: VGG 19 original model

Table **3-1** presents the performance of VGG models trained on ImageNet, The VGG 19 model will be use in the next section in its pre-trained form.

Table 3-1.: VGG performance on ImageNet model [103] for the top-1 and top-5 error rate

model	params	Top-1 %	Top-5 %
VGG 16	138 M	24.7 %	7.5 %
VGG 19	143 M	23.7 %	6.8 %

3.3. Deep Learning architecture combined with handcrafted features

Traditional deep learning architectures have not achieved good results when it comes to texture classification. As stated by [13], this is because texture datasets lie in a much higher dimensional manifold than object recognition datasets. In [13], the need to create new deep learning architectures that integrate handcrafted features was emphasized. This paper addresses this need by proposing a network architecture that extracts handcrafted texture features from the output of selected convolutional layers.

One of the most common problems in deep learning is the lack of sufficiently large datasets

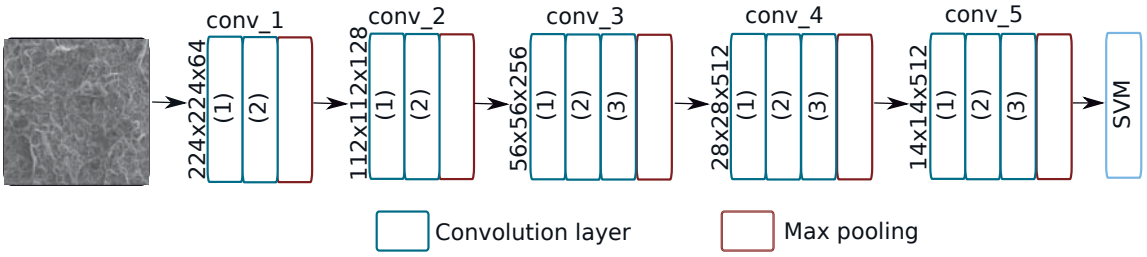


Figure 3-5.: VGG-19 architecture

for training. Even with methods [88], such as data augmentation and transfer learning, networks still overfit. Unfortunately, to the best of our knowledge, large databases for texture classification that would allow efficient training with deep neural networks are not available. To alleviate this problem, in the context of textural classification, we propose to enhance VGG-19 features by incorporating handcrafted texture features in the feature extraction process. These features not only improve the classification performance of the network but also mitigate the translation and rotation invariance problems of CNNs.

The method starts with the VGG-19 model, shown in Fig. **3-5**, pre-trained on the ImageNet database. Then, three sets of handcrafted features, namely, fractal dimension, Haralick and LBP are extracted from the outputs of selected convolutional layers. Classification using the extracted features is performed with an SVM. Fig. **3-6** illustrates the process with an input RGB image of size $224 \times 224 \times 3$ corresponding to a fracture surface. Fig. **3-6** a) illustrates the process when the handcrafted features are extracted from the output of the first convolutional layer, which contains 64 filters, and therefore, its output corresponds to a set of feature maps of size $224 \times 224 \times 64$. Fig. **3-6** b) illustrates a similar process when the handcrafted features are extracted from the output of the second convolutional layer, which contains 128 filters, and therefore, its output corresponds to a set of feature maps of size $112 \times 112 \times 128$.

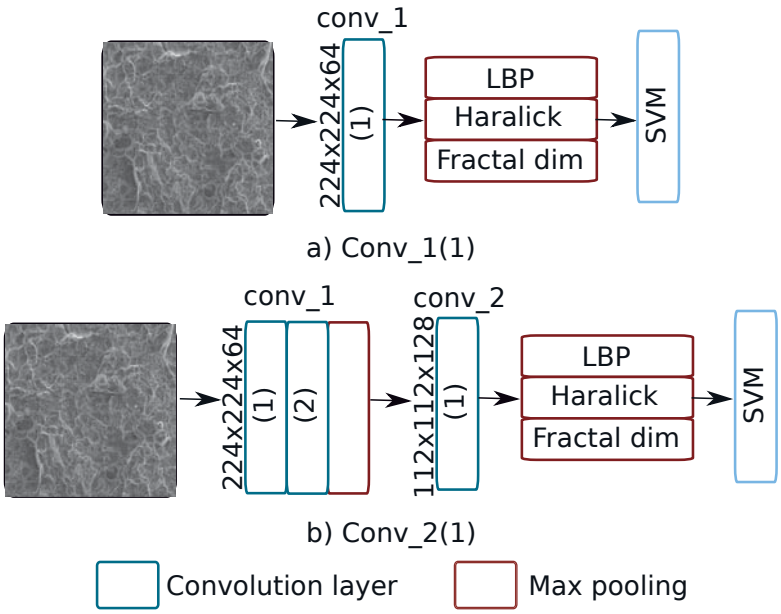


Figure 3-6.: Feature extraction process. (a) Feature extraction from the first convolutional layer Conv_1(1) and b) Feature extraction from the second convolutional layer Conv_2(1).

3.4. Experiments and Results on the KTH-TIPS and KTH-TIPS2-B Texture Databases

The KTH-TIPS Texture Database was developed by the Computational Vision and Active Perception Laboratory (CVAP) at the KTH Royal Institute of Technology in Stockholm [18]. There are three versions of this dataset: KTH-TIPS, KTH-TIPS2-A and KTH-TIPS2-B. In this study, we work with the first and the third versions since they are the most widely used versions in the literature [39]. The KTH-TIPS dataset has nine different classes with 81 images per class with a size of 200×200 pixels. In this work, 70 images are used for training and 11 images are used for testing. The KTH-TIPS2-B dataset has 11 classes with four folders per class called samples, each sample has 108 images. So, as shown by Fujieda [39], one of the samples of each class was used for training and another sample was used for testing. In this paper, both databases are used as a benchmark to demonstrate that incorporating handcrafted features in deep networks leads to improvement in performance. Fig. **3-7** shows samples of six classes of the KTH-TIPS2-B Texture Database.

Table **3-2** shows the most representative results obtained by classifying the testing sets of the KTH-TIPS and KTH-TIPS2-B with the proposed approach using different handcrafted features and convolutional layers. The results of the VGG-19 model without handcrafted features for the KTH-TIPS2-B and fracture databases present a low performance, having a

low F1-score of 0.64 and 0.55, respectively. Haralick features for the fourth group of convolutional layers and fractal dimension features from the early convolutional layers were not calculated due to the high computational cost. For the KTH-TIPS and KTH-TIPS2-B databases, the results of the VGG-19 model were outperformed by our approach using LBP and Haralick features. An F1-score of 1.0 was achieved for the KTH-TIPS dataset by extracting LBP and Haralick features from the first feature maps of the third convolutional layer group. For the KTH-TIPS2-B dataset, the best results, corresponding to an F1-score of 0.77, were obtained by extracting LBP features from the second feature maps of the fifth convolutional layer group.

Figure 3-7.: Examples for 6 classes of the KTH-TIPS2-B texture database.

Table **3-3** shows the best accuracy results obtained with the proposed approach for the KTH-TIPS and KTH-TIPS2-B databases. These results are comparable with those of the state of the art for the two databases. For the KTH-TIPS dataset, an accuracy of 100 % is obtained with the proposed method against a 99.8 % accuracy in the state-of-the-art, while an accuracy of 79.8 % is obtained with the proposed method for the KTH-TIPS2-B dataset against a 81.8 % accuracy obtained with a VGG model in combination with the Fisher vector proposed by Cimpioi [21] and against a 74.2 % accuracy obtained with the wavelet CNN proposed by Fujieda [39]. Furthermore, LBP features are computationally less expensive than the Fisher vector representation.

Table 3-2.: Results of the proposed approach and comparison with the VGG-19 model for texture benchmark datasets

Texture benchmark databases				
KTH-TIPS				
Feature set	VGG Layer	Precision	Recall	F1-score
VGG-19 model	conv_5(4)	99 %	99 %	0.99
Fractal dim.	conv_3(1)	78 %	77 %	0.77
Haralick	conv_3(1)	**100 %**	**100 %**	**1.0**
LBP	conv_3(1)	**100 %**	**100 %**	**1.0**
KTH-TIPS2B				
Feature set	VGG Layer	Precision	Recall	F1-score
VGG-19 model	conv_5(4)	67 %	64 %	0.64
Fractal dim.	conv_3(1)	54 %	55 %	0.52
Haralick	conv_5(1)	**80 %**	**77 %**	**0.74**
	conv_5(2)	75 %	74 %	0.71
LBP	conv_3(1)	64 %	71 %	0.66
	conv_4(1)	62 %	70 %	0.64
	conv_5(1)	**84 %**	**78 %**	**0.77**
	conv_5(2)	80 %	80 %	0.77

Table 3-3.: Accuracy of the best results obtained with the proposed method for the KTH-TIPS and the KTH-TIPS2-B textural databases.

Database	VGG Layer	Feature set	Accuracy
KTH-TIPS	Conv_3(1)	Haralick	100 %
		LBP	100 %
KTH-TIPS2-B	Conv_5(2)	LBP	79.8 %

3.5. Conclusions of the Chapter

The proposed model extracts three sets of handcrafted features, namely fractal dimension, Haralick features and LBP, from the output of different convolutional layers of the VGG-19 model pre-trained with the ImageNet database. To test the model, two benchmark textural databases were employed: KTH-TIPS and KTH-TIPS2-B. Experimental results showed that the proposed approach outperforms the F1-score results of using just the VGG-19 model by 13 % for the KTH-TIPS2-B database when using LBP features. Therefore, experimental results suggest that it is beneficial to integrate handcrafted features with deep neural networks for the problem of texture classification. This assumption, lead us to think that by building an architecture able to obtain texture representation in an end-to-end manner would help in the texture recognition problem.

4. Chapter 3: Deep Adaptive Wavelet Network

Summary

Even though convolutional neural networks have become the method of choice in many fields of computer vision, they still lack interpretability and are usually designed manually in a cumbersome trial-and-error process. This chapter aims at overcoming those limitations by proposing a deep neural network, which is designed in a systematic fashion and is interpretable, by integrating multiresolution analysis at the core of the deep neural network design. By using the lifting scheme, it is possible to generate a wavelet representation and design a network capable of learning wavelet coefficients in an end-to-end form. Compared to state-of-the-art architectures, the proposed model requires less hyper-parameter tuning and achieves competitive accuracy in image classification tasks.

4.1. Wavelet Transform in Texture Analysis

A wavelet is a function $\Psi \in L^2(\mathbb{R})$ with zero average Eq. 4-1. The wavelet transform decomposes the signal into translated and dilated wavelets, called wavelet basis. These are constructed by dilating (2^j) and translating ($2^j n$) a single function Ψ (Eq. 4-2). The wavelet basis revels the signal behavior thorough the amplitude of its coefficients, it defines a sparse representation of the signal, and in images, large wavelet coefficients are located in edges and irregular textures [84].

$$\int_{+\infty}^{-\infty} \Psi(t)\, dx = 0. \tag{4-1}$$

$$\{\Psi_j, n(x) = \frac{1}{2^j}\Psi\frac{(x - 2^j n)}{2^j}\}_{j\in\mathbb{Z}, n\in\mathbb{Z}^{\nvDash}}. \tag{4-2}$$

Texture points out intrinsic properties of the surfaces that includes characteristics as roughness, granularity and similarity [79]. However, there is not an appropriate mathematical model for textural images. The texture is generally defined by the human visual perception [84]. Wavelet transform have been widely used in texture recognition tasks, this due to

its ability to perform a scale analysis. This process is similar to the one performed by the human visual system [79]. Some feature extraction techniques applied to each sub-band (or some of them depending on the application) on the wavelet transform, include obtaining a measure of energy [20, 71, 104], statistical features such as Haralick's [118] and histogram analysis [96], among others. The good performance obtained by these feature extraction process in texture applications such as image classification [51, 9], image segmentation [33, 10] and texture syntheses [35], makes wavelet analysis an interesting case of study.

4.2. CNNs as Multiresolution Analysis

Convolutional neural networks proposed by LeCun in 1989 [74] contain filtering and downsampling steps. In order to have a better understanding of CNNs, we propose to interpret convolution and pooling operations in CNNs as operations in multiresolution analysis [83]. In the following, only one-dimensional input signals are considered for simplicity, but the analysis can be easily extended to higher dimensional signals.

Given an input vector $x = (x[0], x[1], ..., x[N-1]) \in \mathbb{R}^N$, and a weighting function ω, referred to as kernel, the convolution layer output (or feature map) $y = (y[0], y[1], ..., y[N-1]) \in \mathbb{R}^N$ can be defined as

$$y[n] = (x * \omega)[n] = \sum_{j \in K} x[n+j]\omega[j] \tag{4-3}$$

where K is the set of kernel indices.

The role of the pooling layers is to output a summary statistic of the input [43]. It is normally used to reduce the complexity and to simplify information. Most common pooling layers consist of convolution and downsampling in signal processing. Using the standard downsampling symbol $\downarrow$, the output vector o from a pooling layer can be written as

$$o = (b * \mathbf{p}) \downarrow p, \tag{4-4}$$

where $\mathbf{p} = (1/p, .., 1/p) \in \mathbb{R}^p$ is the pooling filter.

We can now interpret convolution and pooling layers as operations in multiresolution analysis. In this analysis, the resolution of a signal (measure of the amount of detail in a signal) is changed by a filtering operation, and the scale of a signal is changed by a downsampling operation [85]. The wavelet transform, for example, repeatedly decomposes a signal into spectrum sub-bands by using low-pass k_l and high-pass k_h filters and applies downsampling by a factor of 2.

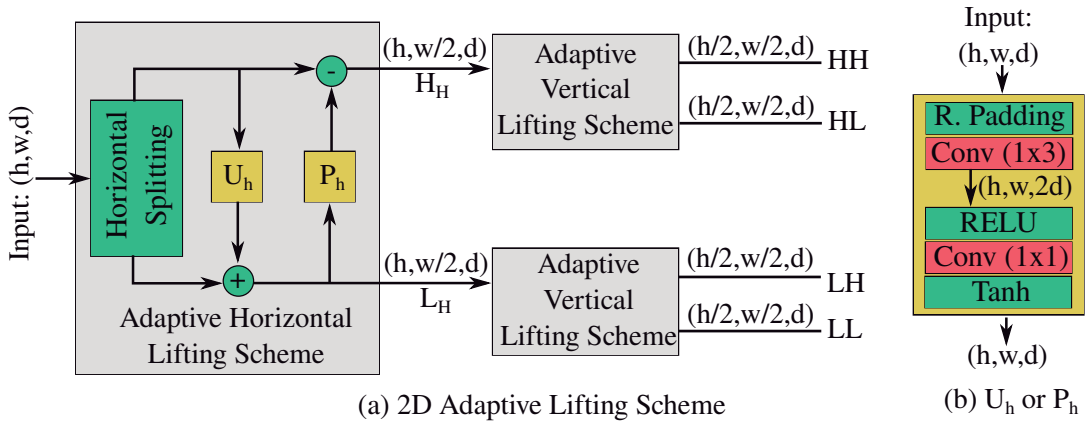


Figure 4-1.: (a) The 2D Adaptive Lifting Scheme consists of successively applying horizontal and vertical lifting steps where each of them have their own predictor and updater. (b) The predictors and updaters are based on operations, such as paddings, convolutions, and non-linear activation functions, which can be either trainable (red boxes) or fixed (green boxes).

Then, to perform a multiresolution analysis, a new signal decomposition is obtained by taking as input the low-pass filtered sub-band c_l. Each of these decompositions are referred to as levels, and generate a hierarchical decomposition of the signal into $c_{l,t}$ and $d_{h,t}$ each time. Let $k_{l,t}$ and $k_{h,t}$ denote the low-pass and high-pass filters at step t, respectively. Such transformation is thus represented as a sequence of convolution and pooling operations,

$$\begin{aligned} c_{l,t+1} &= (c_{l,t} * k_{l,t}) \downarrow 2 \\ d_{h,t+1} &= (d_{h,t} * k_{h,t}) \downarrow 2, \end{aligned} \tag{4-5}$$

where $c_{l,t+1}$ and $d_{h,t+1}$ denote the approximation and detail coefficients generated at step t, respectively, $c_{l,0} = x$ and $d_{h,0} = x$. Based on this level decomposition-based construction, it is possible to compare CNNs structures with multiresolution analysis, as Eqns. 4-4 and 4-5 are quite similar, with the difference that in CNNs the filters are randomly selected and their output does not have a meaningful interpretation.

4.3. Lifting Scheme

The first-generation wavelets are mathematical functions that allow for efficient representations of data using only a small set of coefficients by exploiting space and frequency correlation [85]. The main idea behind the wavelet transform is to build a sparse approximation of natural signals through the correlation structure present on them. This correlation is normally local in space and frequency, meaning that there is a stronger correlation among the

neighboring samples on the signal. The construction of mother wavelets is traditionally performed by using the Fourier transform, however, this can also be constructed in the spatial domain [25].

The lifting scheme, which is also known as second-generation wavelets [107], is a simple and powerful approach to define wavelets that has the same properties as the first-generation wavelets [25]. The lifting scheme takes as input a signal x and generates as outputs the approximation c and the details d sub-bands of the wavelet transform. Designing such lifting scheme consists of three stages [22] as follows.

Splitting the signal. This step consists of splitting the input signal into two non-overlapping partitions. The simplest possible partition is chosen; *i.e.* the input signal x is divided into even and odd components denoted as x_e and x_o, respectively, and defined as $x_e[n] = x[2n]$ and $x_o[n] = x[2n+1]$.

Updater. This stage will take care of the separation in the frequency domain, looking that the approximation has the same running average as the input signal [25]. To achieve this, the approximation c should be a function of the even part x_e of the signal plus an update operator U.

Let $x_o^{L_U}[n] = x_o[n - L_U], x_o[n - L_U + 1], \ldots, x_o[n + L_U - 1], x_o[n + L_U]$ denote the sequence of $2L_U + 1$ neighboring odd polyphase samples of $x_e[n]$. The even polyphase samples are updated using $x_o^{L_u}[n]$, and the result forms the approximation c, as described in Eqn. 4-6, where $U(\cdot)$ is the update operator.

$$c[n] = x_e[n] + U(x_o^{L_U}[n]). \tag{4-6}$$

Predictor. The splitting partitions of the signals are, typically, closely correlated. Thus, given one of them, it is possible to build a good predictor P for the other set, by tracking the difference (or details) d among them [25]. As the even part of the signal $x[n]$ corresponds to the approximation $c[n]$ (Eqn. 4-6), then it is possible to define P as a function of $c[n]$.

Let $c^{L_P}[n] = c[n - L_P], c[n - L_P + 1], \ldots, c[n + L_P - 1], c[n + L_P]$ denote a sequence of $2L_P + 1$ approximation coefficients. In the prediction step, the odd polyphase samples are predicted from $c^{L_P}[n]$. The resulting prediction residuals, or high sub-band coefficients d, are computed by Eqn. 4-7, where $P(\cdot)$ is the prediction operator.

$$d[n] = x_o[n] - P(c^{L_P}[n]). \tag{4-7}$$

4.4. Lifting Scheme Via Neural Networks

Yi *et al.* [119] proposed a method to replace the prediction and update operators with non-linear functions using trainable neural networks, which leads to operators that adapt to the input signal. To define the loss function needed to train the networks, Yi *et al.* proposed to use the following two assumptions.

Assumption on $c[n]$. The signal c must be the coarse approximation of the input signal x. According to Eqn. 4-6, c is the modified version of x_e, and therefore, close to x_e by definition. Thus, there is only need to minimize the distance between c and x_o. In other words, the loss function of the update network needs to be defined as

$$\text{loss}(U) = \sum_n (c[n] - x_o[n])^2, \tag{4-8}$$

and by substituting Eqn. 4-6 into Eqn. 4-8, the loss function becomes

$$\text{loss}(U) = \sum_n [U(x_o^{L_U}[n]) - (x_o[n] - x_e[n])]^2. \tag{4-9}$$

In the adaptive lifting scheme in [119], the updater operator is designed as a neural network with inputs $x_o^{L_U}[n]$, output $x_o[n] - x_e[n]$, and trained through backpropagation using the loss function in Eqn. 4-9.

Assumption on $d[n]$. Better approximations are obtained with low-energy high-pass coefficients. From (4-7), it becomes evident that having low-energy high-pass coefficients implies that the output of the predictor network should be close to $x_o(n)$, which leads to the loss function:

$$\text{loss}(P) = \sum_n (P(c[n]) - x_o[n])^2. \tag{4-10}$$

In the adaptive lifting scheme [119], the predictor operator is designed as a neural network with inputs $x_o[n]$, output $c[n]$, and trained through backpropagation using the loss function in Eqn. 4-10.

4.5. Deep Adaptive Wavelet Network (DAWN)

We propose a new network architecture, Deep Adaptive Wavelet Network (DAWN), which uses the lifting scheme to capture essential information from the input data for image classification. The adaptive lifting scheme presented by Yi *et al.* [119] showed that neural networks trained through backpropagation can be used to implement the lifting scheme for

one-dimensional (1D) signals. The DAWN architecture extends this idea to address a classification task, and integrates multiresolution analysis into neural networks. The proposed model performs multiresolution analysis at the core of the classification network by training the parameters of two-dimensional (2D) lifting schemes in an end-to-end fashion. None of the previous wavelet-based CNN approaches have performed this end-to-end training while learning the wavelet parameters.

4.5.1. 2D Adaptive Lifting Scheme

We first explain the proposed *2D Adaptive Lifting Scheme*, and then present the integration of the 2D lifting scheme into the proposed classification architecture.

The *2D Adaptive Lifting Scheme* consists of a horizontal lifting step followed by two independent vertical lifting steps that generate the four sub-bands of the wavelet transform. These sub-bands are denoted as LL, LH, HL, and HH, where L and H represent low and high frequency information, respectively, and the first and second positions refer to the horizontal and the vertical directions, respectively. Note that the 2D lifting scheme, illustrated in Figure **4-1** (a), performs spatial pooling, as the spatial size of the outputs are reduced by half with respect to the input.

The *Adaptive Horizontal Lifting Scheme* performs horizontal analysis by splitting the 2D signal into two non-overlapping partitions. We chose to partition the 2D signal into the even ($x_e[n] = x[2n]$) and odd ($x_o[n] = x[2n+1]$) horizontal components. Then a horizontal updater (U_h) and a horizontal predictor (P_h) operators are applied in the same way as described in Section 4.3. The vertical lifting step has a similar structure as the horizontal lifting step, but in this case, the splitting is performed in the vertical component of the 2D signal, followed by the processing, performed by the vertical updater U_v and the vertical predictor P_v operators.

Predictor and Updater. The internal structure of the updater and the predictor is the same for both the vertical and horizontal directions. Figure **4-1** (b) shows the structure of the horizontal predictor (or horizontal updater). At the beginning, reflection padding is applied instead of zero padding to prevent harmful border effects caused by the convolution operation. Then, a 2D convolutional layer, where the kernel size, depending on the direction of analysis ($(1, 3)$ if horizontal while $(3, 1)$ if vertical), is applied. The output depth of the first convolutional layer is set to be twice the number of channels of the input. Then, a second convolutional layer with kernels of size (1,1) is applied. The output depth of this layer is set the same as the initial input depth of the predictor/updater. The stride for all the convolutions is set to $(1, 1)$.

The first convolutional layer is followed by a *relu* activation function, and we can benefit from its properties of sparsity and a reduced likelihood of vanishing gradient. The last convolutional layer is followed by a *tanh* activation function as we do not want to discard negative values in this stage.

Design Choices. We arbitrarily chose to perform the horizontal analysis before the vertical analysis. However, there are no performance variations by computing the vertical analysis first. The number of convolutional layers and the kernel size used in the predictor/updater will be discussed during the hyperparameter study (Section 4.6). The main concern while choosing the depth was to maintain a relevant visual representation of the approximation and details sub-bands, while not considerably increasing the number of network parameters.

4.5.2. DAWN Architecture

The DAWN architecture is based on stacking multiple *2D Adaptive Lifting Schemes* to perform multiresolution analysis (see Figure **4-2**). The architecture starts with two convolutional layers followed by a multiresolution analysis of M levels. Each level consists of a *2D Adaptive Lifting Scheme*, which generates as output the four wavelet transform sub-bands LL, LH, HL and HH, and the input correspond to the low level sub-band (LL) from the previous level. The details sub-bands from each level (LH, HL, HH) are concatenated and followed by a global average pooling layer [78], used to reduce overfitting and to perform dimensionality reduction. In the last level, the global average pooling of the outputs at each level are concatenated before the final fully-connected layer and a log-softmax to perform the classification task.

Number of levels. The minimum size of feature maps at the end of the network for this architecture is set to 4×4 as it is the minimum possible size that still maintains the 2D signal structure. Assuming that the input images are square, the number of levels M, is given by $M = \lfloor \log_2(i_s) - \log_2(4) \rfloor$, where i_s is the input image dimension. For example, for input images of size 224×224, $i_s = 224$ and $M = 5$. Note that this number of layers is automatically given since our network is based on multiresolution analysis. The effect of choosing different levels, than the ones given by M is analyzed during the hyperparameter study (Section 4.6).

Initial convolutional layers. As in every classification task, the proposed approach needs a discriminative representation of the data before the classification takes place. To obtain a discriminative feature set before the first downsampling of the signal, the architecture starts by extracting descriptors with two sequences of Conv-BN-ReLU, where Conv and BN stand for Convolution and Batch Normalization respectively, with kernel size 3×3 and with the same depth. The depth in these initial convolutional layers is one of the few hyper-parameters

of DAWN. By fixing the depth and determining the number of decomposition levels, one can automatically obtain the depth of features maps of the last 2D lifting scheme for a given input image size.

Loss function and constraints. End-to-end training is performed using the cross-entropy loss function, in combination with some regularization terms to enforce a wavelet decomposition structure during training. The loss function takes the form of Eqn. 4-11, where P denotes the number of classes, y_i and p_i are the binary ground-truth and the predicted probability for belonging to class i, respectively. The regularization parameters λ_1 and λ_2 tune the strength of the regularization terms. Also, m_l^I and m_l^C denote the mean of the input signal to the lifting scheme at level l and the mean of the approximation sub-band at level l, respectively. And, $\mathbb{D}_l$ is the concatenation of the vectorized detail sub-bands at level l.

$$\begin{aligned} \textbf{Loss} = &- \sum_{i=1}^{P} y_i \log(p_i) \\ &+ \lambda_1 \sum_{l=1}^{M} \mathrm{H}(\mathbb{D}_l) + \lambda_2 \sum_{l=1}^{M} \|m_l^I - m_l^C\|_2^2. \end{aligned} \tag{4-11}$$

To promote low-magnitude detail coefficients [42], the first regularization term in Eqn. 4-11

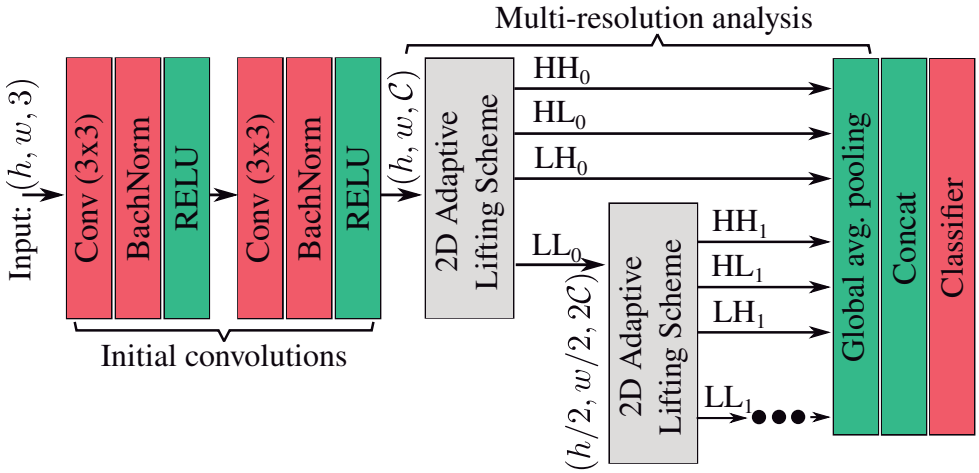


Figure 4-2.: The proposed architecture is composed by three modules: *i)* Initial convolutional layers to increase the input depth, *ii)* M levels of multiresolution analysis, where 2D lifting scheme is applied on the approximation output of the previous level, and *iii)* a large concat of details from the different levels and the approximation, followed by a global average pooling and a dense layer. The operations in the architecture can be classified as either trainable (red boxes) or fixed (green boxes).

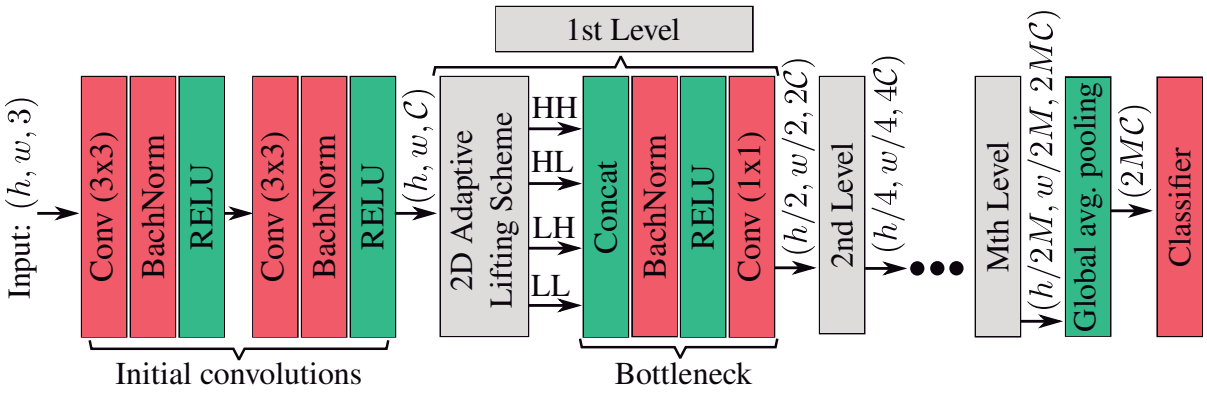


Figure 4-3.: The proposed architecture for the Moified DAWN is composed by three modules: *i)* Initial convolutional layers to increase the input depth, *ii)* M levels of multiresolution analysis where each of them contains a 2D lifting scheme, and *iii)* a global average pooling and a dense layer. The operations in the architecture can be classified as either trainable (red boxes) or fixed (green boxes).

minimizes the sum of the Huber norm of $\mathbb{D}_l$ across all the decomposition levels. The choice of a Huber norm compared to ℓ_1 is motivated by training stability. The second regularization term minimizes the sum of the ℓ_2 norm of the difference between m_l^I and m_l^C across all the decomposition levels in order to preserve the mean of the input signal to form a proper wavelet decomposition [42].

4.5.3. Modified Deep Adaptive Wavelet Network

Multiresolution analysis based on wavelet transform have been widely used in texture recognition task, however, in some applications a deeper representation is needed. For those scenarios, a variation of the DAWN architecture is proposed. This variation is based on the wavelet tree structure [20], which not only calculates the wavelet transform for the next level form the approximation, but also from some details output of the previous level. It has the same initial convolution layers, followed by M levels of *2D Adaptive Lifting Scheme*, but this time, the four sub-bands of the previous level are concatenated and a bottleneck layer is applied to compress the number of components generated. This variation of the architecture is presented in Fig. **4-3** and will be referred as *M-DAWN* in further sections. It is important to note that for this model the regularization term is not used as different from multiresolution analysis the four sub-bands of the 2D adaptive lifting scheme are concatenated and used as input for the next level.

Bottleneck layer. In traditional CNNs, it is possible to choose the number of filters and the position of the pooling layer arbitrarily. However, for a given 2D lifting scheme, the output depth is four times larger than the input depth and the pooling step is mandatory. This incremental factor is problematic because it increases the number of trainable parameters significantly through the architecture. To alleviate this issue, we propose to use a convolutional layer with kernels of size 1×1 to only increase the number of channels by two for each level. Batch normalization and a non-linear activation function are added to reduce training time and to increase the non-linearity of the network.

4.5.4. Visual Representation Results

The decomposition generated by the lifting scheme has a relevant visual representation as it is composed of approximation and details sub-bands of an input signal. Figure **4-4** shows the visualization of the multiresolution analysis for different number of decomposition levels. To generate the visualizations presented in Figure **4-4**, the network was run without the initial convolutional layers on KTH database (Section 3.4).

Many decomposition levels are very similar to traditional wavelet decomposition where the approximation sub-band captures the low-frequency information of the image while the detail sub-bands tend to capture high-frequency information. However, some sub-bands are slightly different as the loss function also minimize the cross-entropy loss function to ensure good classification performance (Section 4.5).

4.6. Experiments and Results on Benchmarks Datasets

The evaluation of the DAWN model was analyzed on one texture dataset, KTH-TIPS2-b and three benchmarks datasets for object recognition task, namely, CIFAR-10, CIFAR-100, and ImageNet. The obtained results are compared against different models commonly used for classification: ResNet [52]; DenseNet [54] with growing factor of 12; a variant of VGG [103], which adds batch normalization, global average pooling, and dropout. The proposed architecture is also compared with previous networks using some multi-resolution analysis component: wavelet CNN (WCNN) [39], and Scattering network [89]. For this later one, we show the results of the handcrafted representation and the hybrid network that combines scattering transform on top of a Wide-Resnet. For KTH-TIPS2-b, T-CNN [6] results is shown as this architecture specifically tailored to texture analysis. The training was done on multiple NVIDIA V100 Pascal GPUs with 12Gb of memory.

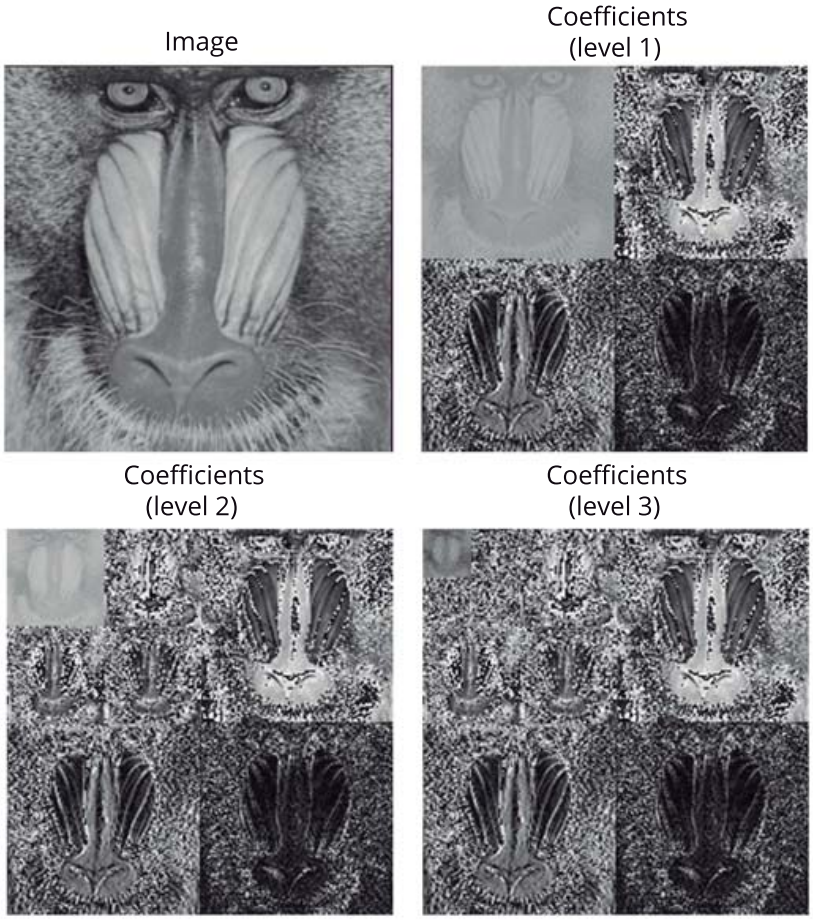


Figure 4-4.: Results of extracting the coefficients for 3 decomposition levels of the 2D Adaptive Lifting Scheme in the DAWN architecture. The loss function applied is the same as in Eqn. 4-11. For visualization purposes, the LH, HL and HH sub-bands were multiplied by a factor of 10.

Implementation An SGD optimizer with a momentum of 0,9 is used for training. The initial learning rate is set to 0,03 for all the databases. The batch size is set to 256, 64 and 16 for ImageNet, the CIFAR databases and KTH-TIPS2-b, respectively. A learning rate decay of 0,1 is applied on epochs 30 and 60 for ImageNet and KTH-TIPS2-b; and on epochs 150 and 255 for CIFAR. The number of epochs is set to 90 and 300 for ImageNet and KTH-TIPS2-b, and the CIFAR databases, respectively. The regularization parameters λ_1 and λ_2 are set to 0,1 for all the experiments. For the Scattering networks [89] on the CIFAR databases, the original training setup has been used, as it achieves higher accuracy than the one obtained with the configuration proposed in this paper for the other architectures.

Object Recognition Datasets

CIFAR CIFAR-10 [67] contains 60000 colour images of size 32×32 belonging to 10 classes. The same partition used to train and test DenseNet [54] is used in this paper, *i.e.* 50000 images for training and 10000 images for testing. CIFAR-100 [67] has 100 classes with 500 images per class. The data augmentation consists in applying random cropping with a padding of 4 pixels and horizontal mirroring operations.

Table **4-1** shows the best results of each architecture on these two databases. There are different DenseNet configurations available with a default growth value of 12. The configuration chosen for the comparison was the one with the closest number of parameters to that of the proposed model. The 18-layer ResNet architecture, after replacing the initial convolutional layers with a convolutional layer with stride 1 and kernel size 3×3, is used for comparison. Those layers were removed because they are normally used to reduce the size of the image at the beginning of the network, which is not required for the small images of the CIFAR datasets. For WCNN, an experiment on varying the number of levels was conducted and the result of the best variant is reported in Table **4-1**. Scattering transform network configurations are the same used the original paper [89] for these datasets.

For the CIFAR databases, the proposed network uses three levels of lifting scheme, as the input image size is 32×32. Table **4-1** shows that increasing the number of initial convolutional filters tends to improve the accuracy performance. Therefore, it is up to the user to balance between a more compact network, in terms of number of parameters, and a network with better classification performance. DAWN architecture outperforms WCNN for both datasets even when the proposed architecture has significantly less number of parameters. The scattering network with handcrafted representation (Scattering+MLP) achieves less accuracy than DAWN architecture as the wavelets are not learned.

It also has a competitive accuracy for CIFAR-10 compared to VGG and ResNet architectures; furthermore, DAWN with a depth of 256 for the initial convolution layers, outperforms the results in both architectures for CIFAR-100 dataset. The scattering hybrid representation (Scattering+WRN) has a considerable higher number of parameters than the other architectures, and its performance is similar to VGG and ResNet for both datasets. In this application, the DenseNet architecture exhibits good performance due to its ability to retain relevant features through the entire network.

Hybrid network As an additional experiment, the proposed multiresolution analysis can be combined with other network architecture. This hybrid network (DAWNN+WRN) consists in replacing the scattering transform by the 2D lifting schemes (Figure **4-2**) inside the Scattering+WRN architecture. This proposed hybrid architecture has similar number of trainable parameters than Scattering+WRN. On CIFAR databases, this architecture gets 93,76 % and 74,88 % of accuracy for CIFAR-10 and CIFAR-100, respectively, which is slightly

Table 4-1.: Comparison of accuracy results on the CIFAR-10 and CIFAR-100 databases. The number of trainable parameter are shown for CIFAR-100 database.

Architecture	# param.	CIFAR-10	CIFAR-100
VGG (variation)	15.0 M	94.00 %	72.61 %
ResNet 18	11.2 M	94.25 %	73.30 %
DenseNet 40	1.10 M	94.73 %	75.25 %
DenseNet 100	7.19 M	95.90 %	79.8 %
WCNN L3	2.28 M	89.85 %	65.17 %
Scattering+WRN	45.5 M	92.31 %	72.26 %
Scattering+MLP	17.0 M	81.90 %	49.84 %
DAWN (16 init.)	59.3 K	86.04 %	56.7 %
DAWN (32 init.)	0.21 M	90.41 %	65.06 %
DAWN (64 init.)	0.73 M	92.69 %	70.57 %
DAWN (128 init.)	2.79 M	**93.34** %	72.47 %
DAWN (256 init.)	10.9 M	92.02 %	**74.04** %

higher compared to the one obtained by Scattering+WRN.

Hyperparameter Tuning DAWN network uses a few number of hyperparameters inside the architecture. Besides the initial convolution depth, the other hyperparameters are the kernel size and the number of convolutional layers inside the updater and predictor of the lifting scheme. This section presents an analysis of the effect of these hyperparameters on the final architecture results. For simplicity, the experiments are performed on CIFAR datasets using the DAWN architecture with 64 initial filters.

Kernel size and number of convolutions Both of these hyperparameters affect the lifting scheme module, whose role is to generate a mathematical function for the wavelet representation. The update operator U needs to represent the frequency structure of the input signal, while the predictor P needs to represent the spatial structure of the input signal. These hyperparameters also affect the final number of trainable parameters for the whole architecture. Table **4-2** shows the effect when changing these hyperparameters: *i)* the kernel size experiments were obtained with the U/P structure described in Figure **4-1** *ii)* the number of hidden layers inside the module is generated by the repetition of the first convolutional layer of the U/P module. It is noticed that the performance results do not have a high variance for combinations of hyperparameters with similar number of trainable parameters.

Number of multiresolution analysis levels Table **4-2** shows how the number of trainable parameters depends on the number of levels of the 2D adaptive lifting scheme. This table

Table 4-2.: Results of tunning the DAWN architecture with 64 initial convolutions. The first table entry is the network configuration used to generate the results in Table **4-1**. The hyperparameters tested are kernel size (k), the number of hidden convolutional layers (h), and the number of levels (l). The number of trainable parameter are shown for CIFAR-100 database.

Configuration	CIFAR-10	CIFAR-100	# param.
(k=3, h=1, l=3)	92.69 %	70.57 %	734'628
(k=1, h=1, l=3)	88.09 %	64.30 %	439'716
(k=2, h=1, l=3)	92.27 %	68.01 %	587'172
(k=4, h=1, l=3)	92.69 %	70.96 %	882'084
(k=3, h=2, l=3)	92.58 %	70.51 %	918'564
(k=3, h=3, l=3)	92.46 %	68.85 %	1'140'900
(k=3, h=4, l=3)	92.35 %	68.19 %	1'363'236
(k=3, h=1, l=0)	75.49 %	44.12 %	45'348
(k=3, h=1, l=1)	90.53 %	66.71 %	275'108
(k=3, h=1, l=2)	92.17 %	70.42 %	504'868

illustrates how the performance varies from not using any lifting scheme level (only initial convolutions), which results in poor performance, to using the maximum number of possible levels (according to Section 4.5). As shown in Table **4-2**, it is usually beneficial to use the maximum number of levels as it leads to higher accuracy values for both datasets. Note that in the CIFAR database, the input size is 32×32, which makes makes the maximum number of possible levels equal to 3.

ILSRVC-2012 (ImageNet) The ILSRVC-2012 dataset consists of 1.2 million images for training and 50000 images for testing, from 1000 different classes. For this database, the M-DAWN architecture contains 5 decomposition levels. The proposed model is compared to commonly used models for image classification using the top-1 error rate. As shown in Table **4-3**, the proposed model obtains better results than those attained by AlexNet [68], Scattering + Rwsnet 10 [89] and WCNN [39]. Even though ResNet [52], DenseNet [54] and VGG [103] attain smaller top-1 errors rates, our proposed model is less complex, requires less hyper-parameter tuning and is designed in a systematic fashion without a cumbersome trial-and-error process.

KTH-TIPS2-B Texture Databases Similar to Section 3.4, KTH-TIPS2-B was used as texture benchmark. As in other works [39], one of the samples of each class was used for training and the rest sample folders were used for testing. The data augmentation consists in applying random cropping and mirroring operations. Table **4-4** contains the average and

Table 4-3.: Comparison of top-1 and top-5 error rate results on the ILSRVC-2012 Database.

Architecture	# param.	top-1 %	top-5 %
AlexNet [68]	62.3 M	36.7	15.4
VGG-E 19 [103]	143 M	27.3	9.0
ResNet-50[120]	25.6 M	24.01	7.02
DenseNet-121 [54]	8.0 M	25.02	7.71
DenseNet-201 [54]	20.2 M	22.58	6.34
Scattering + Resnet-10[89]	12.8 M	31.3	11.4
WCNN (4 levels) [39]	11.6 M	34.75	–
M-DAWN (16 init.)	6.46 M	32.08	11.89
M-DAWN (32 init.)	24.8 M	28.95	9.768

standard deviation across different training sessions.

In this database, WCNN [39] with 4 levels achieves better accuracy compared to T-CNN with a smaller number of trainable parameters. The proposed architecture with a depth of 16 for the initial convolutional layers, achieves the same accuracy as WCNN but with a much smaller number of parameters. Note that the initial convolutional layers are essential for extracting meaningful feature representations, and without them the performance of the model drops significantly.

Scattering network with the handcrafted representation (Scatter+FC) consist of using a scattering transform of spatial scale five followed by a global average pooling and ending with a fully connected layer. This network configuration is very similar to the proposed network structure used for this database (Figure **4-2**). This network configuration achieves similar performance to the proposed approach with sightly less trainable parameters as the wavelets are not trainable. This result indicates that our architecture is able to learn representations that are similar to the scattering transform.

The proposed architecture performs better than DenseNet 13 and 22 BC with similar number of parameters. Note that for DenseNet, the number indicates the total number of layers used inside the network and BC meaning the use of the bottleneck compression approach [54]. Scattering network with hybrid configuration (Scatter+WRN) increases significantly the number of trainable parameter compared to the handcrafted representation network. This hybrid configuration perform poorly as it overfit the dataset, and it has a highly dependence on the CNN architecture and the setup of hyperparameters.

Table 4-4.: Comparison of accuracy results on the KTH-TIPS2-b database where all the network are trained from scratch without pre-trained information.

Architecture	# param.	Avg.	Std.
T-CNN	19'938'059	63.80 %	1.68
WCNN L4	10'211'811	68.83 %	0.73
Scatter+WRN	10'934'283	60.33 %	2.19
Scatter+FC	22'484	68.57 %	2.86
DenseNet 22 BC	74'684	65.71 %	1.35
DenseNet 13	89'711	66.16 %	1.52
DAWN (no init.)	2'894	58.60 %	4.10
DAWN (16 init.)	71'227	68.88 %	2.14

4.7. Conclusions of the Chapter

We presented the DAWN architecture, which combines the lifting scheme and CNNs to learn features using multiresolution analysis. In contrast to the black-box nature of CNNs, the DAWN architecture is designed to extract a wavelet representation of the input at each decomposition level. Unlike traditional wavelets, the proposed model is data-driven so that it adapts to the input images. It is also trainable end-to-end and achieve state-of-the-art performance for texture classification with very limited number of trainable parameter. Furthermore, the decomposition generated by the lifting scheme has a relevant visual representation as it is composed by approximation and detail sub-bands of an input signal. Interpreting convolution and pooling operations in CNNs as operations in multiresolution analysis helped us to systematically design a novel network architecture. The performance of DAWN is comparable to that of state-of-the-art classification networks when tested on the CIFAR-10 and CIFAR-100 datasets. For ImageNet, the DAWN model outperforms AlexNet, Scatter + Resnet-10 and WCNN, which demonstrates its ability to perform well in image classification tasks while being designed in a systematic fashion with little hyper-parameter tuning.

5. Chapter 4: Deep Learning and Fracture Analysis (Experiments and results)

Summary

Periodic fractographic analysis of fracture surfaces helps to improve the performance of mechanical pieces and avoids economical and security problems in many industries, such as the automotive industry. Classifying a fracture into a failure mode is necessary to determine the causes that generated the fracture in the first place. Experts in fracture classification of metallic materials usually use texture and surface marks to determine the type of fracture. In this chapter two datasets for failure modes are presented; the first one is the same used in previous handcrafted features studies [12], and consists on a real-scale fracture dataset; The second one was obtained for this study and consist in different scale images obtained with a Scanning Electron Microscopy (SEM). Furthermore, the performance on these datasets of the approaches presented in previous chapters: DL + Handcrafted features, Chapter 3 and DAWN architecture, Chapter 4, proposed for texture analysis is also presented. Finally a Human-Expert based comparison is done, in order to evaluate the final performance.

5.1. Fracture Databases

As it was presented in the state-of-the-art, Chapter 2, each failure mode has its own features represented with different marks and textures. An expert on the topic searches for these features when performing fracture mode classification. First, the expert observes the fracture surface at real-scale. Since this step is typically not enough to determine the fracture mode, the next step is to search for the features by using lenses and microscopes [57]. Table **5-1** illustrates the main features that an expert on the topic typically uses for fracture surface classification. In terms of texture, ductile fracture shows a fibrous texture, while brittle fracture presents a granular texture and a fatigue fracture presents a smooth and plane texture.

Table 5-1.: Main visual features analyzed by an expert on failure classification.

Fractures	Marks	Texture
Ductile	Microvoids and deformation of the piece	Fibrous
Brittle	River and radial	Granular
Fatigue	Ratchet, beach and striations	Plane, smooth

Real-Scale For the real-scale database, to test the performance of the proposed method, the same database used in [12] with three fracture modes, namely ductile, brittle and fatigue, is employed. The dataset consists of real scale images, acquired with a camera without augmentation. An contains a total of 300 images (100 per class) for the training step and 186 images for the testing step. Fig. **5-1** exhibits two samples of real-scale images per each class. All images were resized to a scale of 224×224 pixels. The training and testing datasets consist of 100 and 186 images per class, respectively.

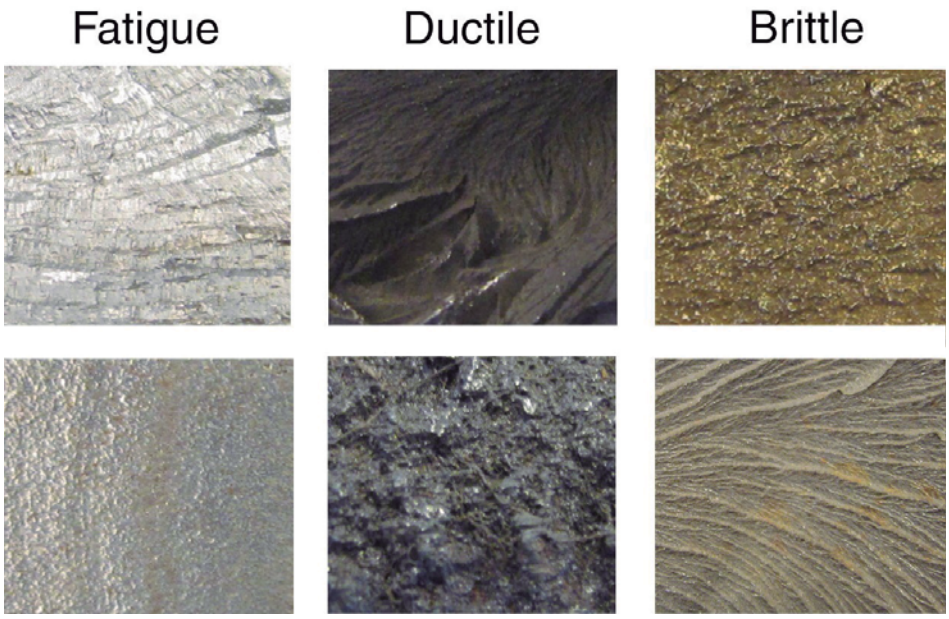


Figure 5-1.: Examples of real-scale images from the fracture database used in this paper.

SEM The SEM fracture database consists on images divided among four different classes of fractures, namely, ductile, brittle, fatigue and corrosion fatigue. The scale is not fixed and it varies depending on the piece. Therefore, all images are resized to 224×224 pixels. Figure **5-2** shows two samples per class of the SEM fracture database. It is worth noting that images belonging to the same class also exhibit a remarkable variation, which makes fracture classification a difficult task for a non-expert eye. A total of 120 images per class

was used for training, while 31 images per class were used for testing.

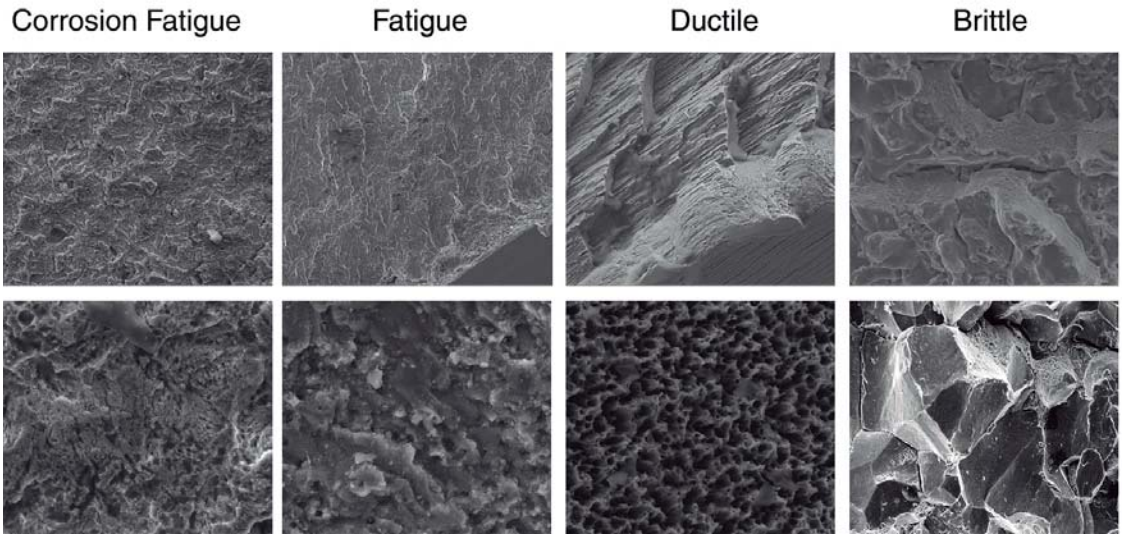


Figure 5-2.: Samples of SEM images from the fracture database used in this paper.

5.2. Traditional Deep Learning Models

We obtained the accuracy of training traditional deep learning models, pre-trained with ImageNet, over the two fracture database. Three traditional deep learning models were analyzed, namely, VGG-16, VGG-19 and ResNet 18. For the VGG models the torchvision version with batch normalization were used. The results presented in Table **5-2** shows the tested models, the number of last tuned layers and the accuracy achieved in both databases. For tuning the parameters a learning rate of $1e^{-3}$ was used, and a SGD optimizer with 0.9 momentum and weight decay of $1e^{-4}$ was applied. As general analysis, it can be observed that deep learning techniques show interesting results in the fractographic analysis and it is possible to train end-to-end architectures on these kind of problem. However, to improve its results some modifications have to be done; specially for the analysis on the real-scale database. Moreover it represents and attractive case of study. A further analysis per dataset is presented in the below paragraphs.

Real-scale fracture dataset As it can be observe in Table **5-2**, the results varies in a small range. Furthermore, it is not possible to achieve similar performance to the ones got with object datasets when using pre-trained models, where the accuracy normally achieves higher results than the ones obtained by the ones achieved with handcrafted features. However,

best result for this dataset was achieved by using ResNet-18 architecture with an F1-score of 0,737; similar to the one achieved by using handcrafted features. (Table **2-2**).

SEM dataset it presents an acceptable accuracy, having into account the small amount of data and the differences presented among the images with different scales; the best result for this database is achieved by using ResNet-18 and fine-tuning the whole architecture layers with an accuracy of 85,48 %. SEM images contain more details in terms of shapes than the ones perceived by the real-scale image; for instance, in this images, it is possible to distinguish the microvoids of the ductile fracture or the transgranular or intergranular crack of the brittle fracture mode. This characteristics allows the system to classify the fracture modes better in SEM than in real-scale images, by using pre-trained and fine-tuning models.

Table 5-2.: Comparison among traditional deep learning architectures pre-trained with ImageNet and fine-tuned at some layers

model	layers	Real-scale		SEM	
		acc.	F1-score	acc.	F1-score
VGG-16	4	67.74 %	0.642	61.29 %	0.591
	8	71.51 %	0.696	70.16 %	0.573
	10	72.04 %	0.71	76.61 %	0.748
	all	71.51 %	0.699	84.68 %	0.827
VGG-19	4	68.28 %	0.678	61.29 %	0.613
	8	70.43 %	0.673	75.81 %	0.733
	10	68.82 %	0.661	76.61 %	0.762
	all	68.28 %	0.652	83.87 %	0.830
ResNet-18	4	70.97 %	0.688	70.10 %	0.683
	8	69.89 %	0.678	79.03 %	0.768
	10	70.43 %	0.684	79.03 %	0.773
	all	**74.73 %**	**0.737**	**85.48 %**	**0.848**

5.3. Deep Learning Combined with Handcrafted Features

The same approach used in the Chapter 3, where a VGG architecture pre-trained with ImageNet was intervened to extract textural handcrafted features, more precisely, Haralick's features, fractal dimension, and LBP features, was applied for the real-scale and SEM fracture databases, by using an SVM classifier. The results of the proposed approach for both databases are exhibited in Table **5-3**. In the general case, it can be conclude that adding textural handcrafted features helps to improve the results obtained by only using VGG-19 model feature extraction, and thus creating a model capable of learning this textural featu-

res directly from the input data becomes an interesting research topic. Further analysis per dataset is given in the below paragraphs.

Table 5-3.: Results of the proposed approach and comparison with the VGG-19 model for fracture datasets.

Fracture databases				
Real				
Feature set	VGG Layer	Precision	Recall	F1-score
VGG-19 model	conv_5(4)	56 %	55 %	0.55
Fractal dim.	conv_1(1)	**73 %**	**73 %**	**0.72**
	conv_2(1)	63 %	58 %	0.57
Haralick	conv_1(1)	63 %	63 %	0.63
	conv_2(1)	64 %	64 %	0.64
	conv_3(1)	**67 %**	**68 %**	**0.68**
	conv_5(1)	63 %	62 %	0.62
LBP	conv_1(1)	56 %	57 %	0.56
	conv_2(1)	**66 %**	**65 %**	**0.65**
	conv_3(1)	66 %	65 %	0.65
	conv_4(1)	65 %	64 %	0.64
SEM				
Feature set	VGG Layer	Precision	Recall	F1-score
VGG-19 model	conv_5(4)	48 %	47 %	0.47
Fractal dim.	conv_1(1)	37 %	31 %	0.33
	conv_2(1)	**35 %**	**33 %**	**0.33**
Haralick	conv_3(1)	64 %	63 %	0.64
	conv_5(1)	66 %	65 %	0.65
	conv_5(2)	**68 %**	**68 %**	**0.68**
LBP	conv_2(1)	58 %	58 %	0.58
	conv_3(1)	72 %	72 %	0.72
	conv_4(1)	**73 %**	**72 %**	**0.73**
	conv_5(1)	69 %	69 %	0.69

Real-Scale Fracture Dataset For the real-scale fracture database, the best results, corresponding to an F1-score of 0.72, were obtained by extracting fractal dimension features from the first convolutional layer (Conv_1). By comparing with the results in Table **2-2**, it is noticed that this F1-score is similar to the one obtained by the state-of-the-art algorithm for fracture mode classification using handcrafted features. For the other two sets of features, Haralick and LBP, their metrics are better than those achieved with the VGG-19 model; however, they have a larger error rate than the one obtained with the fractal dimension.

SEM Dataset Opposite to the results obtained for the real-scale fracture dataset, for the SEM fracture database the best results are generated with the LBP feature sets extracted from the first convolutional filter maps of the fourth layer, obtaining a F1-score of 0.73. This shows that the different scale composition of the SEM database are better represented by the local binary patterns code rather than the fractal dimension, similar to the outcome presented for the KTH-TIPS database **3-2**. It is worth notice that for these experiments a total of 404 images for training and 167 images for testing were used. These results corroborate the ones presented in the Chapter 3 were a common deep learning architecture as VGG, obtains better results when it is intervened by adding textural handcrafted features.

5.4. Deep Adaptive Wavelet Neural Network (DAWN)

Table 5-4.: Comparison of accuracy and F1-score results on the real-scale Fracture database and SEM Fracture Database for Deep Learning Architectures.

Architecture	#param.	Real-scale		SEM	
		acc.	F1-score	acc.	F1-score
VGG (variation)	15 M	62.37 %	0.604	35.48 %	0.242
ResNet 18	11 M	75.27 %	0.721	57.26 %	0.561
DenseNet 40	1 M	70.43 %	0.661	58.82 %	0.580
DenseNet 13	88 K	72.04 %	0.704	57.65 %	0.563
DenseNet 22 BC	88 K	68.82 %	0.67	54.03 %	0.514
T-CNN	19 M	64.52 %	0.625	35.29 %	0.281
WCNN	10 M	70.97 %	0.677	52,42 %	0.506
Scater + Resnet10	15 M	66.13 %	0.634	55.65 %	0.532
scatter + FC	3 M	66.13 %	0.634	**62.10 %**	**0.609**
DAWN (no init.)	2 k	72.58 %	0.718	33.06 %	0.245
DAWN (16 init)	70 k	**75.27 %**	**0.747**	38.71 %	0.310
DAWN (32 init)	277 k	71.51 %	0.69	44.35 %	0.381
M-DAWN (no init)	174 k	74.73 %	0.734	58.06 %	0.538
M-DAWN (16 init)	5 M	69.89 %	0.669	62.10 %	0.569
M-DAWN (32 init)	19 M	70.43 %	0.666	**63,71 %**	**0,622**

DAWN and DAWN_c architectures were thought as deep architectures focused on texture analysis. As it was shown on Chapter 4 the architecture achieves state-of-the-art performance for a texture benchmark dataset, with a considerable small number of parameters. We tested the architecture with the fractures datasets. Results are compared with different deep learning architectures, including texture based architectures. For the real-scale dataset, for

ResNet [52], DenseNet [54], VGG-19 [103] (Batchnorm version) and WCNN [39] the experiments were run over 120 epochs, with a learning rate of 0.1 and a weight decay of a factor of 10 in epochs 30, 60 and 90. While for DAWN, T-CNN [6] and Scattering [89] and all architectures for SEM dataset, the tests we used a training over 90 epochs, with a learning rate of 0.03 or 0.01 (choosing the best performance for each), and a weight decay of factor of 10 at epochs 30 and 60. Also, for DAWN architecture a regularization term of 0.2 for approximation and details was also used. For all the experiments, a batch size of 16 and a data augmentation of random crops and horizontal and vertical flips were applied. Table **5-4** present the results obtained.

Real-scale Dataset For the real-scale images, the same as previous analysis data split for training and testing was used. The results show that DAWN architecture obtain a better performance than traditional and texture based deep architectures for this particular problem. A **F1-score of 0.747** was obtained, similar to the one got with handcrafted features alone (Table **2-2**), deep learning plus handcrafted features (Table **5-3**), and the pre-trained models (Table **5-2**). Even though ResNet obtains similar results with a F1-score of 0.721, the number of parameters is considerably higher than the one used by DAWN. Figure **5-3** shows the confusion matrix for the proposed DAWN architecture. It can be observed that the ductile and fatigue fractures have a high classification performance, while more problems are observed for brittle fractures. It is also worth notice that different than ResNet, the common deep learning architectures or the texture focus architectures do not have a better performance than the smallest DAWN, with no initial convolution applied.

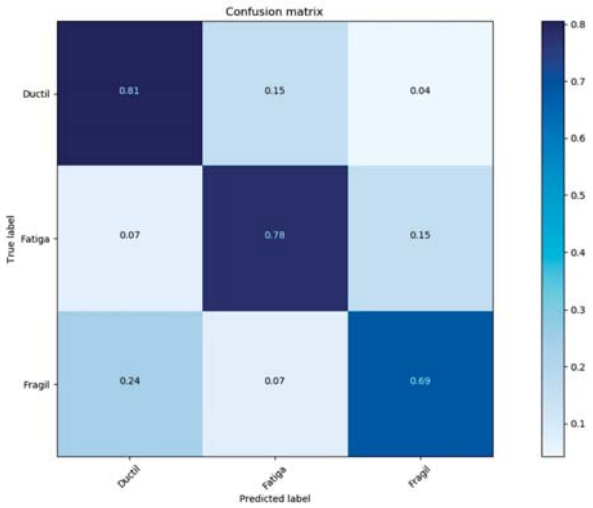


Figure 5-3.: Confusion matrix of the real-scale fracture database.

SEM Dataset For SEM architecture a 120 images per class were used for training and 31 images per class were used for testing (which represents the 20 % of the samples). In this dataset all architectures seem to have more complications than in the previous databases analyzed; this can be explain as the different scale obtained in the images, generates different textures on them. Furthermore, in this scale of analysis, the ductile's dumping, brittle intragranular or transgranular fractures, among other characteristics are more visible than in the real scale dataset; for this reason, the shapes take more importance as features, and the architectures behave similar to object databases. Best F1-score of 0,622 was achieved with the M-DAWN (Modified DAWN), though it is very similar to the one achieved by the scattering network plus a fully connected layer. Among the architectures without handcrafted features. It can be observed that by using deep learning in this kind of images it is possible to achieve interesting results. Fig. **5-4** presents the confusion matrix for the SEM database.

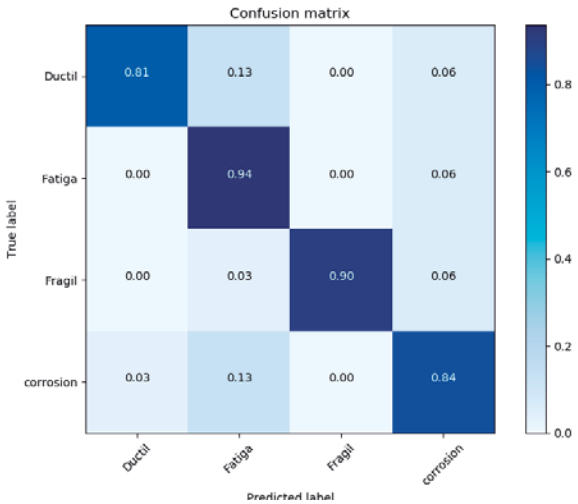


Figure 5-4.: Confusion matrix of the SEM fracture database.

As final analysis, in Table **5-5** the results of two of the strongest deep learning architectures in the state-of-the-art as ResNet-18 and DenseNet-121, compered to DAWN_c architecture and evaluated on the SEM dataset. All methods were pre-trained with ImageNet weights and leaving all the layers trainable. As it can be observed, the proposed network achieves a good performance in this kind of dataset, and it is even a bit better than the other two architectures. These results allow us to conclude that adaptive wavelet based architectures allow to obtain more interpretable models, with less hyperparameters, and that them adapt better to texture based images. Fig. **5-4** shows the confusion matrix for the SEM fracture dataset.

Table 5-5.: Comparison of accuracy and F1-score for pre-training deep learning architectures with ImageNet weights and keeping all the layers on the models trainable, evaluated on the SEM dataset.

model	SEM	
	acc.	F1-score
ResNet-18	85.48 %	0.848
DenseNet-121	81,45 %	0,802
DAWN$_c$(32*init.*)	87.10 %	0.864

5.5. Comparison of Fracture Classification Approaches Among Handcrafted Features, Deep Learning and Human Experts on the Topic

Table 5-6.: Comparison of the fracture classification analysis among the proposed algorithms and two experts on the topic.

Expert	Correct pieces	Wrong pieces	Not defined	% class.
Expert 1	28	4	0	90.32 %
Expert 2	22	9	0	70.96 %
Handcrafted features [12]	24	3	4	77.4 %
DAWN	24	4	3	77,4 %

To check the behaviour of the algorithm against human experts, we decided to validate the results of the real scale dataset, as it is the hardest fracture dataset to classify for an expert on the topic, without access to the real piece. It is important to note that both of the experts consulted are magisters on materials an process at the National University of Colombia, and non of them had access to the real piece, only had access to the image of the whole piece. The ground-truth was obtain by a different expert that had access to the real piece and could perform microscopic analysis in case of doubt. The evaluation methodology for the algorithms consisted in take six ROI images per piece and classify them individually. The final labeled of the piece, then it is obtained when more than three images are classified in the same category. An example of the classification method is shown in Fig. **5-5**. Results of this comparison are shown in table **5-6**. It can be observed that DAWN architecture performs similar to handcrafted features and achieves the same percentage of correct pieces classified. Furthermore, it is worth notice that fracture classification is a non trivial problem where even experts on the topic can encounter some difficulties when analysing real-scale images.

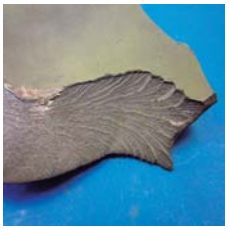
Piece	ROI					
						
Handcrafted DAWN	Ductile Ductile	Ductile Ductile	Ductile Ductile	Ductile Ductile	Ductile Ductile	Ductile Ductile
	Final class. Handcrafted: Final DAWN: Final class. expert:			Ductile Ductile Ductile		
						
Handcrafted DAWN	Fatigue Fatigue	Brittle Ductile	Brittle Brittle	Brittle Fatigue	Fatigue Fatigue	Fatigue Ductile
	Final class. Handcrafted: Final DAWN: Final class. expert:			Not defined Fatigue Brittle		

Figure 5-5.: Comparison of three test pieces among the proposed algorithms and the expert that generated the ground-truth labels

5.6. Conclusions of the Chapter

This work addressed the real-world problem of mechanical failure classification by using a database consisting of 484 real-scale images corresponding to three fracture modes: *i)* ductile, *ii)* brittle and *iii)* fatigue; and a SEM fracture database with 640 images of four failure modes, same three as before plus one additional mode, named corrosion fatigue.
Three different methods were analysed, the first method is a traditional approach, were three different and common deep learning architectures were pre-trained with ImageNet training weights and then a fine tuning process was performed. This method allowed us to conclude that deep learning architectures have a good performance on the SEM images database, and,

for the real-scale dataset the performance it is the same as the one achieved with handcrafted features. For the second method, the Deep learning plus handcrafted features approach, experimental results shows that it outperforms the F1-score results of using just the VGG-19 model by 17 % for the real-scale fracture database when using the fractal dimension, by 26 % for the SEM fracture database when using the LBP feature set. The Final approach consisted in testing the DAWN and M-DAWN architecture on the two fracture datasets; results shows that DAWN architecture performs similar to the handcrafted feature method and ResNet pre-trained with ImageNet, but with less trainable parameters and hyperparameters to tune. For the case of the SEM dataset best results are achieved by pre-training the DAWN_c architecture and keeping all the layer trainable, having an F1-score of 0.864.

6. Conclusions and Future Work

6.1. Conclusions

The research in texture analysis by using computer vision has been based on finding different methods to extract handcrafted features from the data. The most common of these methods are the gray level co-occurrence matrix, the Markov random fields, the local binary patterns and different filters as Gabor or wavelet. Due to the importance in different applications, texture is an active field of investigation. However, it is not an easy problem to approach. Though the texture is a property of the objects easily identified by the human visual system, it does not have a precise definition. Recent researches focus on learning the features directly from the data by using an approach denominated Deep Neural Networks which is based on the mammal brain, and it extracts features trough many layers and a big amount of input data. When the application has a limited amount of data, it is necessary to implement a transfer learning or data augmentation algorithm. Even though these methods have been successful in applications such as object recognition, when it comes to real world texture it is not so intuitive.

As a specific application of the textural architecture, this work is centered on the classification of mechanical fractures modes to be a resource in failure analysis. More specifically, it focuses on the fractographic analysis, which is a visual analysis of the surface of mechanical fractures, looking for particular features like propagation patterns and the fracture's origin. Through the characterization of the fracture surface, it is possible to show the history before the flaw, which allows us to find the causes of the failure. Among the possible causes for the failure of the piece, for example, is a wrong selection of materials (E.g. it does not perform correctly in the environmental conditions), a bad design of the element (E.g. bad decision with the effort or corrosive substances considerations), and installation or maintenance problems.

To tackle this issues, we presented two different approaches focused on texture recognition, *i)* Deep learning combined with handrafted features and *ii)* Deep Adaptive Wavelet Network:

Deep learning combined with handrafted features The first approach uses a common pre-trained CNN architecture, namely VGG-19, and extracts feature maps from different

hidden layers. Then three different sets of handcrafted features, called Haralick's features, Fractal dimension and Local Binary Patterns, commonly used in textural problems were obtained from these feature maps. For KTH-TIPS2-b dataset a F1-score of 0.77 was obtained compared against 0.64 obtained by VGG; For the fractographic problem, a F1-score of 0.72 and 0.73 for real-scale and SEM architecture were obtained, respectively, compared against a F1-score of 0.55 and 0.47 of VGG trained on the same datasets. The experimental results allowed us to conclude that by combining these two approaches, it is possible to improve the results obtained by training VGG in an end-to-end manner, without transfer learning. This assumption lead us to propose an architecture able to extract textural features by learning them from the input data.

Deep Adaptive Wavelet Network The second approach, proposed an architecture by analysing the relationship between multiresolution analysis and deep learning. Furthermore, wavelet approaches had been widely used in texture analysis problems. The designed architecture uses the second wavelet construction denominated lifting scheme. This approach allowed us to construct wavelets with the same properties of the first generation wavelet but in a spatial analysis. Also, we proposed an architecture to obtain the lifting scheme coefficients by using backpropagation. By doing so, it was possible to learn the wavelet coefficients and to adapt them to the input datasets. The proposed architecture was built by using the adaptive lifting scheme and the multiresolution analysis in a classification problem and has the following properties:

- Different to other Wavlelet-CNN methods, the wavelet coefficients are learned during the backpropagation process.
- The hyperparameter tunning is smaller than other CNN based architectures, as the number of layers is given by the number of levels on the multiresolution analysis.
- The loss function constrains proposed regularization methods to maintain the wavelet properties during training.
- The proposed variation of DAWN for complex models, DAWN_c, allowed us to train the architecture in more complex datasets such as ImageNet.

Finally, the results obtained with this architecture showed that it is possible to have similar performances to the state-of-the-art for CIFAR 10 and CIFAR 100 datasets. And for ImageNet, the proposed model outperforms AlexNet, Scatter + Resnet 10 and WCNN architecture. In terms of texture classification, for KTH-TIPS2-b dataset it is possible to obtain state-of-the-art results with a considerable small number of parameters; and for the fracture datasets, DAWN architecture obtains better results than other traditional and textural based architectures, and performs similarly to handcrafted features with a considerably smaller number of parameters; finally, $DAWN_c$ pre-trained with ImageNet weights, got a F1-score of 0.864, and it is better than the one obtained with traditional deep learning architectures.

6.2. Future Work

Data recollection It is well known that for, deep neural networks to work well, a big amount of data it is necessary. Even though in this work we did a big effort to obtain fracture real-scale images and SEM images, we acknowledge that it is important to get more data to obtain better results. In the case of SEM images it is important to obtain images at different scales as the visual perception of the texture change depending on the scale of the image.

Multiresolution analysis as a deep learning architecture Similarly to the DAWN architecture, Bruna and Mallat [16] used a multiresolution analysis based on wavelet transform as a backbone of their architecture. Both this work and ours focus on the wavelet extraction as an operation invariant to deformation. In Bruna's work, the modulus is obtained from each wavelet coefficient at different levels. In DAWN architecture, the details coefficients per level of the wavelet transform are carried out to the end of the network. One big difference between DAWN and the Scattering handcrafted representation is the ability of DAWN to learn the wavelet configuration. It is this ability that allows it to adapt to the data and perform equivalently across different datasets, as it was shown in Tables **4-4** and **4-1**.

Combining Multiresolution analysis with more traditional CNNs architectures The hybrid network with the proposed 2D lifting scheme shows the potential of improving the accuracy or reducing the number of trainable parameters for other networks. How to combine or incorporate more CNN features in the proposed network and keeping performances across the different datasets is an interesting work avenue.

Initial convolutions At the moment, the architecture uses initial convolutional layers to increase the number of channels from the input image, which is a simple approach. Research using more advanced architectures for this part of the proposed network is left as future work. Moreover, multiresolution analysis is usually applied on an image instead on a CNN output. Changing the order of the initial convolutions and the different lifting scheme might conduct to some exciting new architectures.

A. Appendix: Generative Adversarial Networks and Data Augmentation

Summary

Deep learning approaches have had an outstanding success in many task, including image classification, however it is undeniable that this could not be possible without the existence of big enough datasets, that allow the networks to learn different features from the data. There are some strategies on the state-of-the-art to overcome the problem when the available data is not enough to improve the performance. This is the case of texture datasets, where the available data is not enough to obtain or improve the performance generated by hand-crafted features. In this work, we use the Progressive Growing GANs approach to perform Data Augmentation into generate new samples for texture classification task. As particular case of this analysis we work on surface fracture recognition. In the recent years, an interest on Generative Adversarial Networks (GANs) has been increasing. This approach, introduced in 2014 by Ian Godfellow [44], is able to produce new content from a starting training dataset. Usually, the GAN is form from two networks, The *generator* that will generate the new samples form some random noise, trying to fit some training distribution; and the *discriminator* which will discriminate among the new and old samples. These two networks encounter themselves in an adversarial training, where the winning of one of them, will represent the losing of the other. The final aim will be to generate samples undistinguished from the training samples.

Generative Adversarial Networks Deep Learning, has become in a widely used approach for image classification, since AlexNet [68] beat the ImageNet Challenge [27] in 2012 by a big margin. However its need for a huge training datasets to perform well it is a limitation in several real world problems, such as texture analysis, where the datasets do not contain many samples per class. Many GANs studies have been proposed for low image resolution generation [61], however, with the aim of using this approach in data augmentation for real world problems, it is important to obtain some high quality samples. Among all the GANs used to tackled this problem we can highlight the Cycle-GAN [60] which uses a pair of generators and discriminators in order to produce image to image translation; it was later used by Wang and Perez [115] where they compere several techniques to treat

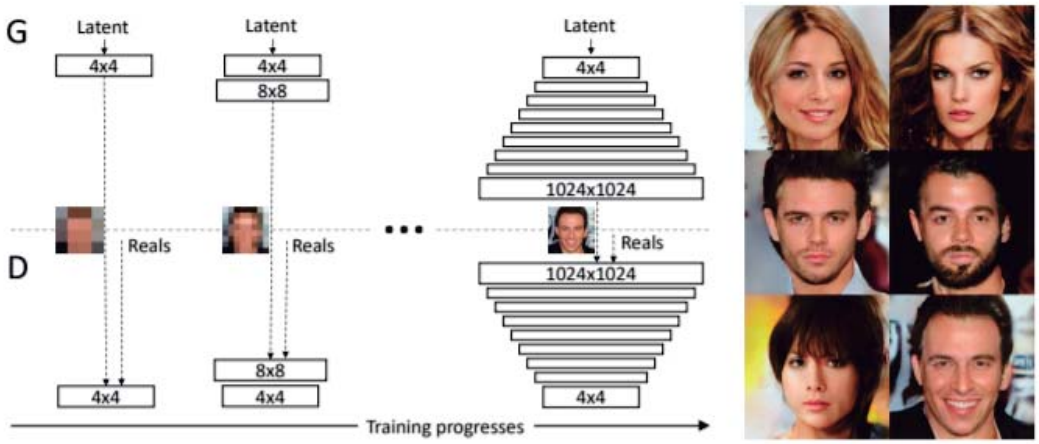


Figure A-1.: Progressive Growing GAN architecture. Both, the generator (G) and the discriminator (D) start with a resolution of 4×4 and progressively grows until achieve the spatial resolution needed for the new samples [61].

the problem before mentioned. ACGAN [87] which, different to traditional GAN, uses an auxiliary classifier giving to each generated sample a class label and the discriminator predicts not only between real or fake samples but also among the classification labels. DAGAN [8] in this approach the generator uses an autoencoder to obtain new samples, here first the training samples pass trough an encoder, then some noise is added and finally the new images are obtained through the decoder. More recently, the Progressive GAN [61] proposed by NVIDIA researchers, has shown very good results in generating high resolution images. This approach is the one use in this study, in order to augment data for textural benchmark datasets and Fracture surfaces dataset.

Progressive Growing GAN In this approach proposed by Karras *et al.* [61] the generator and discriminator start by a obtaining some low resolution images (of 4×4 size pixels), and then adding layers to the network, to progressively increase the resolution of the images. By doing so, the network learns features in a multi-scale analysis, by obtaining low level features in the first steps, and finally obtaining finer details as the network grows. In this approach the discriminator and generator are a mirror of each other and always grow in the same amount of layers and filters. Fig. **A-1** illustrates the process before mentioned. In the training process all layers remain trainable, and to avoid some issues with the already trained weights, a transition between scale changes is performed. This process is shown in Fig. **A-2**. It consists in perform a nearest neighbor interpolation to project the previous scale layer to the new layer added with the new spatial resolution. The projected layer is multiplied by a factor of $1 - \alpha$ and concatenated with the new output layer multiplied by an α factor, to form the new image with the new spatial resolution.

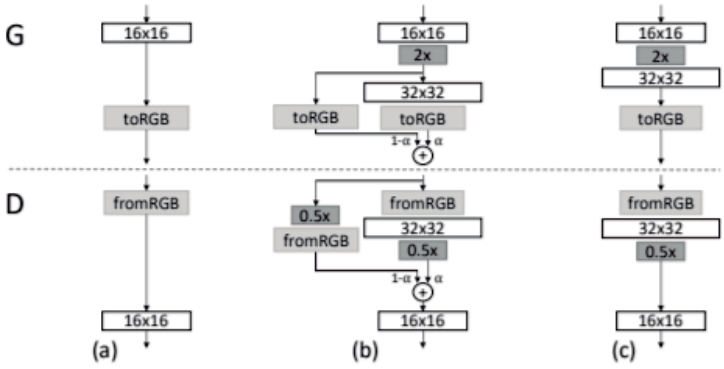


Figure A-2.: A transition among spatial resolution of 16×16 (a) to 32×32 (c) pixels is presented. [61].

To avoid repetition and similarities in the generated samples, and to increase variation on them, researchers use the approach proposed by Salimans *et al.* [100], where the use of *minibatch discrimination* is applied. At the end of the discriminator a layer with this purpose is added. Its main function is to discriminate among minibatches rather than discriminate among one sample. This layer projects the input vector into a tensor of the standard deviation of each feature in each spatial location over the minibatch, and then the average of these features is obtain to get one single value, obtaining an non-trainable feature map.

Setup Configuration One of the problems of GANs is that they need heavily and complex computation, thus it needs a huge amounts of memory to obtain high resolution images. In this work we obtain images of size 128×128 pixels due to memory constrains. The architecture configuration for the generator and discriminator is shown in Table **A-1**.

The images obtained in this process are presented in Fig. **A-3** for the real-scale dataset. It can be observed that even though the image presents a high-quality visualization, they look to much alike to the training images, thous when training the classification system adding this images, it tends to overfit easier than without this data augmentation. Then, some modifications have to be done to use this kind of networks as data augmentation system for texture analysis, and we leave it as future work. This research stage it is left then as future work.

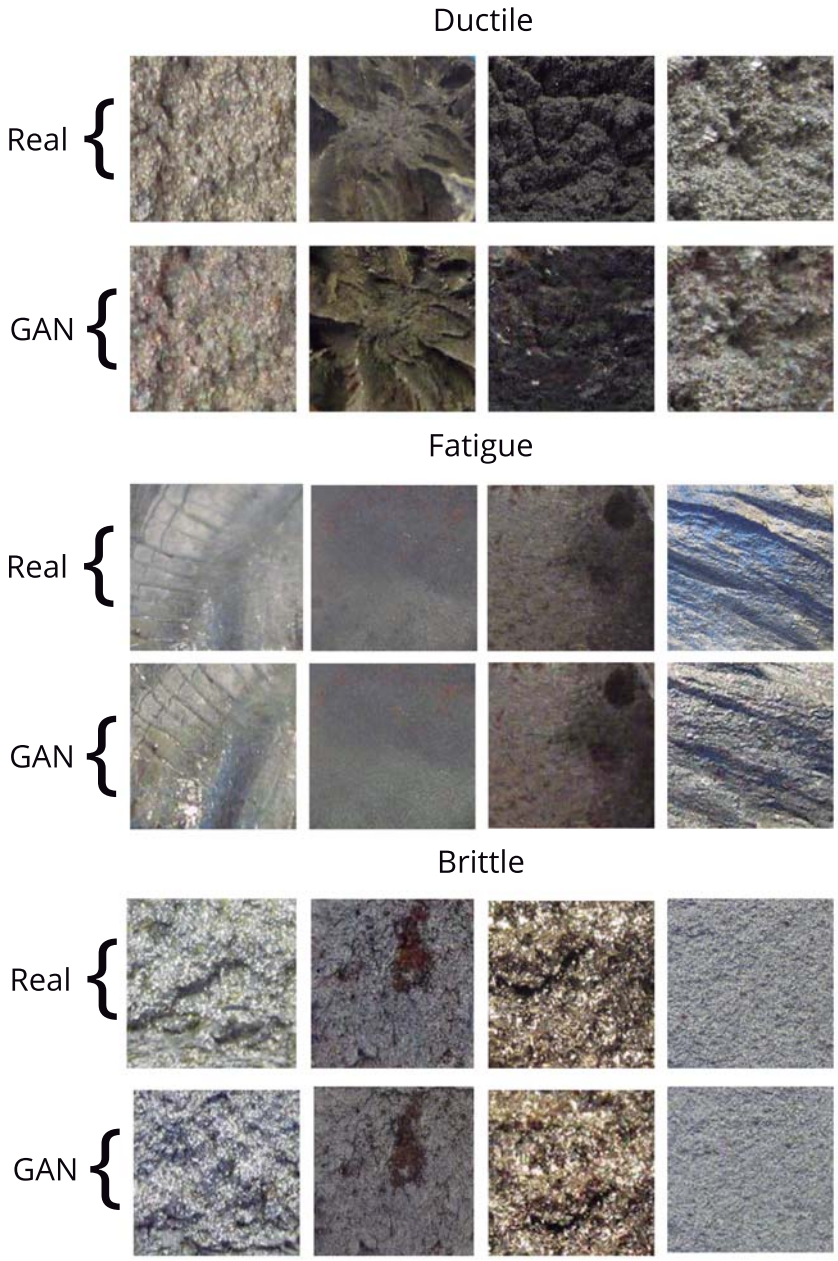


Figure A-3.: Results of Pregressive GAN applied to the real fracture dataset

Table A-1.: Generator and discriminator used for the Real-scale Fracture Dataset to generate images with 128×128 pixels

Generator	Output Shape
Latent vector	$512 \times 1 \times 1$
Conv 4×4	$512 \times 4 \times 4$
Upsample	$512 \times 8 \times 8$
Conv 3×3	$256 \times 8 \times 8$
Conv 3×3	$256 \times 8 \times 8$
Upsampling	$256 \times 16 \times 16$
Conv 3×3	$128 \times 16 \times 16$
Conv 3×3	$128 \times 16 \times 16$
Upsampling	$128 \times 32 \times 32$
Conv 3×3	$64 \times 32 \times 32$
Conv 3×3	$64 \times 32 \times 32$
Upsampling	$64 \times 16 \times 16$
Conv 3×3	$32 \times 64 \times 64$
Conv 3×3	$32 \times 64 \times 64$
Upsampling	$32 \times 128 \times 128$
Conv 3×3	$16 \times 128 \times 128$
Conv 3×3	$16 \times 128 \times 128$
Conv 1×1	$3 \times 128 \times 128$

Discriminator	Output Shape
Input Image	$3 \times 128 \times 128$
Conv 1×1	$16 \times 128 \times 128$
Conv 3×3	$16 \times 128 \times 128$
Conv 3×3	$32 \times 128 \times 128$
Dawnsample	$32 \times 64 \times 64$
Conv 3×3	$32 \times 64 \times 64$
Conv 3×3	$64 \times 64 \times 64$
Dawnsample	$64 \times 32 \times 32$
Conv 3×3	$64 \times 32 \times 32$
Conv 3×3	$128 \times 32 \times 32$
Dawnsample	$256 \times 16 \times 16$
Conv 3×3	$256 \times 16 \times 16$
Conv 3×3	$512 \times 16 \times 16$
Dawnsample	$512 \times 8 \times 8$
Conv 3×3	$512 \times 8 \times 8$
Conv 3×3	$512 \times 8 \times 8$
Dawnsample	$512 \times 4 \times 4$
Minibatch stddv	$512 \times 4 \times 4$
Conv 3×3	$512 \times 4 \times 4$
Conv 4×4	$512 \times 1 \times 1$
Fully-connected	$1 \times 1 \times 1$

Bibliography

[1] Introduction to Support Vector Machines. , p. on–line: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[2] Computer Vision. En: *Department of Computer Science and Engineering, University of Washington* (2000), p. on–line: https://courses.cs.washington.edu/courses/cse576/book.

[3] Redes Neuronales: de la teoría a la práctica. En: *on-line: https://www.mql5.com/es/articles/497* (2014)

[4] Material Failures – The Role of Fractography in Determining the cause. (2017)

[5] Andrearczyk, V. ; Whelan, Paul F.: Deep Learning for Biomedical Texture Image Analysis. En: *Proceedings of the 18th Irish Machine Vision and Image Processing conference* (2016)

[6] Andrearczyk, V. ; Whelan, Paul F.: Using Filter Banks in Convolutional Neural Networks for Texture Classification. En: *Pattern Recognition Letters, Vol. 84* (2016), p. 63–69

[7] Anselm, Martin K.: Manager, Analytical & Failure Analysis Services, Universal Instruments. En: *On-line: https://www.smta.org/.* (2018)

[8] Antreas Antoniou, Harrison E.: Data Augmentation Generative Adversarial Networks. En: *arXiv:1711.04340* (2017)

[9] Ardakani, Ali A. ; Gharbali, Akbar ; Mohammadi, Afshin: Classification of Benign and Malignant Thyroid Nodules Using Wavelet Texture Analysis of Sonograms. En: *Journal of Ultrasound in Medicin* (2015)

[10] Arivazhagan, S. ; Ganesan, L.: Texture segmentation using wavelet transform. En: *Pattern Recognition Letters* 24 (2003)

[11] Backes, A.R. ; Bruno, O.M.: Plant Leaf Identification Using Multi-scale Fractal Dimension. En: *ICIAP* (2009), p. 143–150

[12] Bastidas-Rodriguez, M.X. ; Prieto-Ortiz, F.A. ; Espejo, Edgar: Fractographic classification in metallic materials by using computer vision. En: *Engineering Failure Analysis* 59 (2016), p. 237–252

[13] Basu, S. ; Mukhopadhyay, S. ; Karki, M. ; DiBiano, R. ; Ganguly, S. ; Nemani, R. ; Gayaka, S.: Deep neural networks for texture classification - A theoretical analysis. En: *Neural Networks* 97 (2018), p. 173 – 182

[14] Becker, W.T.: Fracture Appearance and Mechanisms of Deformation and Fracture. En: *University of Tennessee, Emeritus; S. Lampman, ASM International* (2010)

[15] Bengio, Yoshua: Learning Deep Architectures for AI. En: *Foundations and Trends in Machine Learning, Vol. 2* (2009)

[16] Bruna, Joan ; Mallat, Stephane: Invariant Scattering Convolution Networks. En: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 35 (2013), Nr. 8

[17] Camargo, O.M. ; Arista, B.V ; Camargo, J.M: Digital Processing of Fractographic Images for Welded Joints on Microalloy Steel API5L-X52 Aged. En: *IEEE Latin America, Transactions* (2013), p. Vol. 11, NO 1

[18] Caputo, B. ; Hayman, E. ; Mallikarjuna, P.: Class-specific material categorisation. En: *IEEE International Conference on Computer Vision* (2005), p. 1597–1604

[19] Chakraborty, S. [u. a.]: Interpretability of deep learning models: A survey of results. En: *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, 2017, p. 1–6

[20] Chang, T. ; Kuo, C. . J.: Texture analysis and classification with tree-structured wavelet transform. En: *IEEE Transactions on Image Processing* 2 (1993), Nr. 4, p. 429–441

[21] Cimpoi, M. ; Maji, S. ; Kokkinos, I. ; Vedaldi, A.: Deep filter banks for texture recognition, description, and segmentation. En: *International Journal of Computer Vision* 118 (2016), Nr. 1, p. 65–94

[22] Claypoole, R. L. ; Davis, G. M. ; Sweldens, W. ; Baraniuk, R. G.: Nonlinear wavelet transforms for image coding via lifting. En: *IEEE Transactions on Image Processing* 12 (2003), Nr. 12, p. 1449–1459

[23] Cotter, F. ; Kingsbury, N.: Deep Learning in the Wavelet Domain. En: *arXiv preprint arXiv:1811.06115* (2018)

[24] Cross, George R. ; Jain, Anil K.: Markov Random Field Texture Models. En: *IEEE Transactions on pattern analysis and machine intelligence* (1983)

[25] Daubechies, I. ; Sweldens, W.: Factoring wavelet transforms into lifting steps. En: *The Journal of Fourier Analysis and Applications* 4 (1998), Nr. 3, p. 247–269

[26] Deepa Sankar, Tessamma T.: Fractal Features based on Differential Box Counting Method for the Categorization of Digital Mammograms. En: *Internationa Journal of Computer Information Systems and Industrial Management Applications* (2010)

[27] Deng, J. ; Dong, W. ; Socher, R. ; Li, L. ; Li, Kai ; Fei-Fei, Li: ImageNet: A large-scale hierarchical image database. En: *IEEE Conference on Computer Vision and Pattern Recognition* (2009), p. 248–255

[28] Dharmagunawardhana, Chathurika ; Mahmoodia, Sasan ; Bennettb, Michael ; Niranjan, Mahesan: Rotation invariant texture descriptors based on Gaussian Markov random fields for classification. En: *Pattern Recognition Letters 69* (2016), p. 15–21

[29] Dong, Y. ; Su, H. ; Zhu, J. ; Zhang, B.: Improving interpretability of deep neural networks with semantic information. En: *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 4306–4314

[30] D.Pham, Tuan: The Kolmogorov-Sinai entropy in the setting of fuzzy sets for image texture analysis and classification. En: *Pattern Recognition 53* (2016), p. 229–237

[31] Elfadel, Ibrahim M. ; Picard, Rosalind W.: Gibbs Random Fields, Cooccurrences, and Texture Modeling. En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1994)

[32] Elsken, T. ; Metzen, J. ; Hutter, F.: Simple and Efficient Architecture Search for CNNs. (2017)

[33] E.Salari ; Z.Ling: Texture segmentation using hierarchical wavelet decomposition. En: *Pattern Recognition* 28 (1995), Nr. 12

[34] Espejo, E.: Fallas por Fractura y Fractografía. En: *Notas de clase curso Análisis de Falla Universidad Nacional de Colombia - Sede Bogotá* (2013)

[35] Fan, Guoliang ; Xia, Xiang-Gen: Wavelet-based texture analysis and synthesis using hidden Markov models. En: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 50 (2003), Nr. 1, p. 106–120

[36] Fawcett, Tom: An Introduction to ROC Analysis. En: *Pattern Recognition Letters 27* (2006)

[37] Fawzi, Alhussein ; Samulowitzy, Horst ; Turagay, Deepak ; Frossard, Pascal: Adaptive Data Augmentation for Image Classification. En: *Image Processing (ICIP), 2016 IEEE International Conference* (2016)

[38] Feng, Shuo ; Zhou, Huiyu ; Dong, Hongbiao: Using deep neural network with small dataset to predict material defects. En: *Materials & Design* 162 (2019), p. 300–310

[39] Fujieda, S. ; Takayama, K. ; Hachisuka, T: Wavelet Convolutional Neural Networks. En: *arXiv:1805.08620* (2018)

[40] Galloway, Mary M.: Texture Analysis Using Gray Level Run Lengths. En: *Computer Graphics and Image Processing* (1975), p. 172–179

[41] Geirhos, Robert ; Rubisch, Patricia ; Michaelis, Claudio ; Bethge, Matthias ; Wichmann, Felix A. ; Brendel, Wieland: ImageNet-Trained CNNS are biased towards texture; increasing shape bias improves accuracy and robustness. En: *ICLR* (2019)

[42] Gholipour, M. ; Noubari, H. A.: Hardware implementation of lifting based wavelet transform. En: *International Conference on Signal Processing Systems* Vol. 1, 2010, p. V1–215–V1–219

[43] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning.* MIT Press, 2016. – `http://www.deeplearningbook.org`

[44] Goodfellow, Ian J. ; Pouget-Abadie, Jean ; Mirza, Mehdi ; Xu, Bing ; Warde-Farley, David ; Ozair, Sherjil ; Courville, Aaron ; Bengio, Yoshua: Generative Adversarial Networks. En: *arXiv:1406.2661* (2014)

[45] Hadida, Abdenour ; Ylioinas, Juha ; Bengherabi, Messaoud ; Ghahramani, Mohammad ; Taleb-Ahmed, Abdelmalik: Gender and texture classification: A comparative analysis using 13 variants of local binary patterns. En: *Pattern Recognition Letters 68* (2015), p. 231–238

[46] Hafemann, Luiz G. ; Oliveira, Luiz S. ; Cavalin, Paulo: Forest Species Recognition using Deep Convolutional Neural Networks. En: *22nd International Conference on Pattern Recognition* (2014)

[47] Hafemann, Luiz G. ; Oliveira, Luiz S. ; Cavalin, Paulo R. ; Sabourin, Robert: Transfer Learning between Texture Classification Tasks using Convolutional Neural Networks. En: *2015 International Joint Conference on Neural Networks (IJCNN)* (2015)

[48] Haider, Ali: Accelerated Fatigue Reliability Analysis of Stiffened Sections Using Deep Learning. En: *Graduate College of the Oklahoma State University* (2018)

[49] Hammouche, Kamal ; Losson, Olivier ; Macaire, Ludovic: Fuzzy aura matrices for texture classification. En: *Pattern Recognition* (2016), p. 212–228

[50] Hauberg, Søren ; Freifeld, Ore ; Boesen Lindbo Larsen, Anders ; Fisher, III ; Hansen, Lars K.: Dreaming More Data: Class-dependent Distributions over Diffeomorphisms for Learned Data Augmentation. En: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (2016), p. 342–350

[51] Hazra, Dipankar: Texture Recognition with combined GLCM, Wavelet and Rotated Wavelet Features. En: *International Journal of Computer and Electrical Engineering* 13 (2011), Nr. 1, p. 1793–8163

[52] He, K. ; Zhang, X. ; Ren, S. ; Sun, J. Jian S.: Deep Residual Learning for Image Recognition. En: *Conference on Computer Vision and Pattern Recognition* (2016)

[53] Hinton, G.E. ; Osindero, S. ; Teh, Y.: A fast learning algorithm for deep belief nets. En: *Neural Computation* (2006)

[54] Huang, G. ; Liu, Z. ; Van der Maaten, L.: Densely Connected Convolutional Networks. En: *Conference on Computer Vision and Pattern Recognition* (2018)

[55] I, Laws K.: Textured Image Segmentation. En: *University of Southern California* (1980)

[56] Idrissa, Mahamadou ; Acheroy, Marc: Texture classification using Gabor filters. En: *Pattern Recognition Letters* (2002), p. 1095–1102

[57] Ipohorski, M.: Fractografía, Aplicaciones al Análisis de Fallas. En: *CNEA 490* (1988), p. Comisión Nacional de Energía Atómica, Argentina: Buenos Aires

[58] Jacek, Komenda ; Barbara, Maroli ; Lars, Höglund: Recognition of patterns on fracture surfaces by automatic image analysis. En: *Image Anal Stereol* (2002)

[59] Jiménez Guerrero, María Del P.: Extracción de características de textura basada en la Transformada Wavelet Discreta, Capitulo 5. En: *Escuela Superior de Ingenieros, Universidad de Sevilla* , p. `http://bibing.us.es/proyectos/abreproy/11494/fichero/PROYECTO\%252FCapitulo+5.pdf`

[60] Jun-Yan Zhu, Phillip Isola Alexei A. E.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. En: *ICCV* (2017)

[61] KarrasNVIDIA, Tero ; Aila, Timo ; Laine, Samuli ; Lehtinen, Jaakko: Progressive Growing of GANs for Improved Quality, Stability, and Variation. En: *ICLR* (2018)

[62] Khokhlov, M. ; Fischer, A. ; Rittel, D.: Multi-Scale Stereo-Photogrammetry System for Fractographic Analysis Using Scanning Electron Microscopy. En: *Experimental Mechanics* (2012)

[63] Kolednik, O. ; Scherer, S. ; Schwarzböck, P. ; Werth, P.: Quantitative Fractography by Means of a New Digital Image Analysis System. En: *Proc. of ECF13, Paper 1U.60. Elsevier, Amsterdam* (2000)

[64] Komai, Kenjiro ; Minoshima, Kohji ; Ishii, Shoich: Recognition of Different Fracture Surface Morphologies using Computer Image Processing Technique. En: *SME Intentationat Jettma Series A* (1993)

[65] Konovalenko, Ihor ; Maruschak, Pavlo ; Prentkovskis, Olegas ; Junevičius, Raimundas: Automated Method for Fractographic Analysis of Shape and Size of Dimples on Fracture Surface of High-Strength Titanium Alloys. En: *Materials* 11 (2018)

[66] Kosarevych, R.Ya. ; Svirs'ka, O.Z. Student L. ; Rusyn, B.P ; Nykyforchyn, H.M.: Computer Analysis of Characteristic Elements of Fractographic Images. En: *Materials Science* (2013), p. Vol.48, No. 4

[67] Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images. En: *Tech Report* (2009)

[68] Krizhevsky, A. ; Sutskever, I. ; Hinton, G.E.: Imagenet classification with deep convolutional neural networks. En: *Advances in Neural Information Processing Systems* 25 (2012), p. 1097–1105

[69] Kuna, M.: Finite Elements in Fracture Mechanics, Solid Mechanics and Its Applications. En: *Springer Science+Business Media Dordrecht* (2013)

[70] Labatut, Vincent ; Cherifi, Hocine: Accuracy Measures for the Comparison of Classifiers. En: *Cornell University, arXiv:1207.3790* (2012)

[71] Laine, A. ; Fan, J.: Texture classification by wavelet packet signatures. En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), Nr. 11, p. 1186–1191

[72] Lauschmann, H. ; Nedbal, I.: Auto-Shape Analysis of Image Textures in Fractography. En: *Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering, Dept. of Materials, República Checa: Trojanova* (2002)

[73] Laushmann, H. ; Rácek, O. ; Túma, M. ; Nedbal, I.: Textural Fractography. En: *Image Annual Stereol* (2002)

[74] LeCun, Y.: Generalization and network design strategies. En: *Connectionism in Perspective* (1989)

[75] Li, Fei-Fei: Class notes: CS231n: Convolutional Neural Networks. En: *Standford University, available on: http://cs231n.github.io/convolutional-networks/*

[76] Liebowitz, Jay: Expert Systems: A Short Introduction. En: *Engineering Fracture Mechanics* (1995), p. Vol. 50, No. 5/6, pp. 601–607

[77] Lin, Hsuan T.: Texture Classification using Fractal-based Features and Support Vector Machines. En: *Department of Computer Science and Information Engineering, National Taiwan University* , p. `http://www.work.caltech.edu/~htlin/course/doc/FractalFinal.pdf`

[78] Lin, M. ; Chen, Q. ; Yan, S: Network In Network. En: *International Conference on Learning Representations* (2014)

[79] Livens, S ; Scheunders, P ; van de Wouwer, G ; Dyc, D V.: Wavelet for texture analysis, an overview. En: *6th International Conference on Image Processing and Its Applications* (1997)

[80] Lu, H. ; Wang, H. ; Zhang, Q. ; Won, D. ; Yoon, S. W.: A Dual-Tree Complex Wavelet Transform Based Convolutional Neural Network for Human Thyroid Medical Image Segmentation. En: *IEEE International Conference on Healthcare Informatics* (2018), p. 191–198

[81] Maani, Rouzbeh ; Kalra, Sanjay ; Yang, Yee-Hong: Noise robust rotation invariant features for texture classification. En: *Pattern Recognition, vol. 46* (2013), p. 2103–2116

[82] Mahto, Bishnu P.: Characterization of Ductile Iron Through Fractograpic Study. En: *Master thesis, National Institute of Technology* (2014)

[83] Mallat, S.G.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2 (1989), Nr. 7, p. 674–693

[84] Mallat, Stéphane: Wavelet Tour of Signal Processing. En: *Academic Press, 3rd edition* (2009), p. Chapter 1: Sparse Representations

[85] Meurant, G.: *Wavelets: a tutorial in theory and applications*. Vol. 2. Academic press, 2012

[86] Moreno, C.J. ; Espejo, E.: A performance evaluation of three inference engines as expert systems for failure mode identification in shafts. En: *Engineering Failure Analysis* 53 (2015), p. 24–35

[87] Odena, Augustus ; Olah, Christopher ; Shlens, Jonathon: Conditional Image Synthesis with Auxiliary Classifier GANs. En: *arXiv:1610.09585* (2017)

[88] O'Gara, Sarah ; McGuinness, Kevin: Comparing Data Augmentation Strategies for Deep Image Classification. En: *IMVIP 2019: Irish Machine Vision Image Processing, Technological University Dublin, Dublin, Ireland* (2019)

[89] Oyallon, E. ; Belilovsky, E. ; Zagoruyko, S.: Scaling the scattering transform: Deep hybrid networks. En: *International Conference on Computer Vision* (2017)

[90] P. Shanmugavadivu, V. S.: Fractal Dimension Based Texture Analysis of Digital Images. En: *Procedia Engineering* (2012), p. 2981–2986

[91] Pan, Sinno J. ; Fellow, Qiang Y.: A Survey on Transfer Learning. En: *IEEE Transactions on Knowledge and Data Engineering* (2010), p. 1345–1359

[92] Pei, J. ; Wright, J. ; Smyth, A.: Mapping polynomial fitting into feedforward neural networks for modeling nonlinear dynamic systems and beyond. En: *Science Direct, Computer methods in applied mechanics and engineering* (2005)

[93] Pham, Tuan A.: Optimization of Texture Extraction Algorithm. En: *Computer Engineering, Department of Electrical Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University Techbology, Netherlands* (2010)

[94] Pierre, Soille: Morphological Texture Analysis: An Introduction. En: *Morphology of Condensed Matter: Physics and Geometry of Spatially Complex Systems, Springer Berlin Heidelberg* (2002), p. 215–237

[95] Pradhan, M.P. ; Pradhan, R. ; Ghose, M.K: Shape Reconstruction of Fracture Surface for HSLA Materials using Photometric-Stereo Images. En: *International Symposium on Devices MEMS, Intelligent Systems and Communication (ISDMISC)* (2011)

[96] P.S. Hiremath, S. S.: Wavelet based co-occurrence histogram features for texture classification with an application to script identification in a document image. En: *Pattern Recognition Letters* 29 (2008), p. 1182–1189

[97] R. Haralick, K. S. ; Dinstein, I.: Textural features for image classification. (1973)

[98] Rodríguez, Maria Ximena B.: Clasificación fractográfica en materiales cristalinos empleando visión por computador. En: *Universidad Nacional de Colombia, Tesis de Maestría* (2015)

[99] Ross, A.S. ; Doshi-Velez, F.: Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. En: *AAAI Conference on Artificial Intelligence*, 2018

[100] Salimans, Tim ; Goodfellow, Ian J. ; Zaremba, Wojciech ; Cheung, Vicki ; Radford, Alec ; Chen, Xi: Improved techniques for training GANs. En: *NIPS* (2016)

[101] Sayadi, Mounir ; Sakrani, Samir ; Fnaiech, Farhat ; Cheriet, Mohamed: Gray-level Texture Characterization Based on a New Adaptive Nonlinear Auto-Regressive Filter. En: *Ecole de technologie supérieure, Notre-Dame , Montreal, Quebec, Electronic Letters on Computer Vision and Image Analysis* (2008)

[102] Silva, J.M. ; Infante, V. ; de Freitas, M. ; Reis, L.: Microscopy analysis of damaged aeronautical components. En: *Microscopy: Science, Technology, Applications and Education, A. Méndez-Vilas and J. Díaz (Eds.)* (2010)

[103] Simonyan, K. ; Zisserman, A.: VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. En: *International Conference on Learning Representations* (2015)

[104] Smith, J.R. ; Chang, S.F.: Transform features for texture classification and discrimination in large image databases. En: *IEEE International conference on Image Processing* 3 (1994), p. 407–411

[105] Staelin, Carl: Parameter Selection for Support Vector Machines. En: *HPL-2002-354, HP Laboratories Israel* (2002)

[106] Suykens, J.A.K ; Vandewalle, J.: Least Squares Support Vector Machine Classifiers. En: *Neural Processing Letters* (1999)

[107] Sweldens, W.: The lifting scheme: A construction of second generation wavelets. En: *Society for Industrial and Applied Mathematics* 29 (1998), Nr. 2, p. 511–546

[108] S.Y.Wang ; P.Z.Zhang ; S.Y.Zhou ; D.B.Wei ; F.Ding ; F.K.Li: A computer vision based machine learning approach for fatigue crack initiation sites recognition. En: *Computational Materials Science* 171 (2020)

[109] T. Ojala, M. P. ; Harwood, D.: Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. En: *Proceedings of 12th International Conference on Pattern Recognition* (1994)

[110] Tang, Zhe ; Su, Yuancheng ; Er, Meng J. ; Qi, Fang ; Zhang, Li ; Zhou, Jianyong: A local binary pattern based texture descriptors for classification of tea leaves. En: *Neurocomputing 168* (2015), p. 1011–1023

[111] Torrey, Lisa ; Shavlik, Jude: Transfer Learning.

[112] Tsopanidis, Stylianos ; Moreno, Ral H. ; Osovski, Shmuel: Toward quantitative fractography using convolutional neural networks. En: *arXiv:1908.02242* (2019)

[113] Tuceryan, M. ; Jain, A.K. ; Chen, C. H. (Ed.) ; Pau, L. F. (Ed.): *The Handbook of Pattern Recognition and Computer Vision*. 2nd. World Scientific Publishing Co., 1998. – 207–248 p.

[114] Tuceryan, Mihran ; AK, Jain: Texture Analysis. En: *The Handbook of Pattern Recognition and Computer Vision, Edition: 2nd Edition, Chapter: Texture Analysis, Publisher: World Scientific Publishing Co., Editors: C. H. Chen and L. F. Pau* (1998), p. 207–248

[115] Wang, Jason ; Perez, Luis: The Effectiveness of Data Augmentation in Image Classification using DeepLearning. En: *arXiv:1712.04621* (2017)

[116] Williams, T. ; Li, R.: Advanced Image Classification Using Wavelets and Convolutional Neural Networks. En: *IEEE International Conference on Machine Learning and Applications* (2016), p. 233–239

[117] Williams, T. ; Li, R.: Wavelet Pooling for Convolutional Neural Networks. En: *International Conference on Learning Representations* (2018)

[118] de Wouwer, G. V. ; Sheunders, P. ; Dyck, D. V.: Statistical texture characterization from discrete wavelet representations. En: *IEEE Transactions on Image Processing* 8 (1999), Nr. 4, p. 592–598

[119] Yi, Z. ; Wang, R. ; Li, J.: Nonlinear wavelets and BP neural networks adaptive lifting scheme. En: *International Conference on Apperceiving Computing and Intelligence Analysis Proceeding* (2010), p. 316–319

[120] Zagoruyko, S. ; Komodakis, N.: Wide Residual Networks. En: *CoRR, abs/1605.07146* (2017)

[121] Zhang, X. ; Wang, W. ; Yoshikawa, T. ; Takei, Y.: Design of IIR orthogonal wavelet filter banks using lifting scheme. En: *IEEE Transactions on Signal Processing* 54 (2006), Nr. 7, p. 2616–2624

[122] Zhao, Yang ; Jia, Wei ; Hu, Rong-Xiang ; Min, Hai: Completed robust local binary pattern for texture classification. En: *Neurocomputing 106* (2013), p. 68–76

[123] Zhenhua Guo, Lei Z. ; Zhang, David: A Completed Modeling of Local Binary Pattern Operator for Texture Classification. En: *IEEE Transactions on Image Processing* (2010), p. 1657 – 1663