Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

Recognition of Plant Diseases Using Convolutional Neural Networks

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRA EN SISTEMAS COMPUTACIONALES

Presenta: **CYNTIA DENISE LEÓN LEMUS**

Asesor **DR. JOSÉ FRANCISCO CERVANTES ALVAREZ**

Tlaquepaque, Jalisco. Febrero de 2020.

Acknowledgments

I would like to acknowledge:

My thesis advisor Francisco Cervantes, who not only advised me on this project but became a mentor who shared his knowledge with me, and who guided me in the best way that made it possible to conclude this project.

My friend Pedro Romero for all those afternoons and nights of study and for sharing his knowledge in mathematics that facilitated my understanding of machine learning.

My friend Leobardo Ruiz for his professional advice and recommendations and for his support in difficult times that made this journey more bearable.

Intel for all the support in resources that made the realization of this project possible.

Agradecimientos

Me gustaría agradecer a:

Mi asesor de Tesis Francisco Cervantes, que no sólo me asesoró en este proyecto, sino que se convirtió en un mentor que compartió conmigo su conocimiento y que me orientó de la mejor forma que hizo posible concluir este proyecto.

Mi amigo Pedro Romero por todas esas tardes y noches de estudio y por compartirme de su conocimiento en matemáticas que facilitó mi entendimiento en el tema de aprendizaje automatizado.

Mi amigo Leobardo Ruiz por sus consejos y recomendaciones profesionales y por su apoyo en momentos difíciles que hicieron más llevadora esta travesía.

Intel por todo el apoyo en recursos que hicieron posible la realización de este proyecto.

Dedication

I would like to dedicate this project to:

God, for giving me health, perseverance, and patience to reach this point. He more than anyone knows how many were my tears shed for their support. I thank him from the bottom of my heart for all the love he has given me, all the opportunities he has presented me and according to his ways, clear my ways to achieve this goal.

My Mother Blanca Lemus and my Father Jose Luis León, for all their support, for all those words of encouragement I needed when I felt that I could no longer, for always listening to me when my heart needed to let off steam, for being my personal motivators who never doubt that I can to achieve it, I thank them with all my heart, it is not enough for me to give them back so much that they have done it for me.

My sisters Karina León and Blanca León who are always there when I need them, who believe in me, who make me laugh and with their joy motivates me to keep fighting.

My friend Brenda Cruz who always knows how to translate what my heart wants, that inspires me to follow my dreams and that she always has the right words to boost me.

Dedicatoria

Me gustaría dedicar este proyecto a:

Dios, por haberme dado salud, perseverancia y paciencia para llegar hasta este punto. Él más que nadie sabe cuántas fueron mis lágrimas derramadas que pidieron por su apoyo. Le agradezco de todo corazón por todo el amor que me ha brindado, todas las oportunidades que me ha presentado y a deacuerdo a sus formas, limpiarme los caminos para poder lograr este objetivo.

Mi Madre Blanca Lemus y mi Padre Jose Luis León, por todo su apoyo, por todas esas palabras de aliento que necesitaba cuando sentía que no podía más, por siempre escucharme cuando mi corazón necesitaba desahogarse, por ser mis motivadores personales que nunca dudan que podré lograrlo, les agradezco de todo corazón, no me alcanza la vida para devolverles por tanto que han hecho por mi.

Mis hermanas Karina León y Blanca León que siempre están ahí cuando las necesito, que creen en mi, que me hacen reir y que con su alegría me motiva a seguir luchando.

Mi amiga Brenda Cruz que siempre sabe traducir lo que mi corazón quiere decir, que me inspira a seguir mis sueños y que siempre tiene las palabras correctas para impulsarme.

Abstract

Early detection of diseases in crops for human consumption is crucial for agricultural production in the world. Nowadays, Deep learning algorithms are used for the analysis and classification of images such as the convolutional neural network used to detect diseases in plants. Unfortunately, these algorithms still need a lot of computation to obtain results that could take hours, days, or maybe weeks for a prediction. This paper aims to provide an analysis of how the Wavelet transform in distributed platforms can provide competitive results with a fraction of the resources used comparatively in time and memory used to detect diseases in tomato leaves.

Resumen

La detección temprana de enfermedades en los cultivos para consumo humano es crucial para la producción agrícola en el mundo. Hoy en día, los algoritmos de aprendizaje profundo se utilizan para el análisis y clasificación de imágenes, como la red neuronal convolucional utilizada para detectar enfermedades en las plantas. Desafortunadamente, estos algoritmos aún necesitan grandes cantitades de cómputo para obtener resultados que pueden llevar horas, días o quizás semanas para una predicción. Este trabajo de obtención de grado tiene como objetivo proporcionar un análisis de cómo la transformada Wavelet en plataformas distribuidas puede proporcionar resultados competitivos con una fracción de los recursos en relación al tiempo y memoria utilizados para detectar enfermedades en las hojas de jitomate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivation

Early detection of diseases in crops for human consumption is crucial for agricultural production in the world. Monitoring crops in large agricultural regions to detect plant diseases early can be an expensive and slow activity. This is due to the use of classical methods based on observations made with the naked eye by experts. Additionally, crop loss can impact the economy and devastate natural ecosystems.

Mexico is the main tomato supplier in the international market according to SAGARPA, Secretaría de Agricultura y Desarrollo Rural [3], with a participation of 25.11% of the international exportation. In 2017, tomato exports arrived at 2.14B US dollars [4]. For this reason, it is crucial for the Mexican economy an early detection of diseases in plants.

This project will present a wavelet convolutional neural network to identify disease in tomato leaves through images with the motivation of presenting options to get results in less time by feature optimization.

As the population grew, the need to generate more food also increased and with it the importance of optimizing resources. The tools used to keep a crop healthy are no longer enough today, which is why it was necessary to rely on technology to use every resource efficiently. Detecting diseases on time can help to save a whole crop from a bacterium or fungus. Human resources are no longer sufficient to expedite this process, so the automation of this visual analysis can make a total difference in the usage of resources. Here is where technology advances can help

to this noble cause by identifying diseases through images. Machine learning has evolved through years with the motivation to provide systems that can learn by accessing data in order to find patterns and make decisions. In 1943, Warren McCulloch and Walter Pitts presented a paper about a simple binary neural network model, where they described how a machine would learn inspired by how a human or animal brain works. Nowadays, Convolutional Neural Networks are very popular as part of deep learning for image analysis and classification. Those algorithms have more and more applications on the daily use.

Currently, Pathologists rely on plant symptoms to identify their ailment. The identification is based on the observation ability and the response of various questions related to crop care and environmental factors. The diagnosis, using this methodology, can take weeks, and this could be too late on preventing loss crops. These conventional methods are not suitable because according to the Food and Agriculture Organization of the United Nations, one-third of the agricultural production is lost annually by plant diseases and all because it was not detected on time [5]. According to CONAFOR, diseases and pests are the main factors of disturbances that cause: deformations, loss of growth, landslide, and loss of crops [6]. Compared with conventional methodologies, a plant disease detection through image can be an optimal solution for farmers because diagnosis could be detected on time for its fast-responding.

1.2. Problem description

Configure a Wavelet Convolutional Neural Network to identify diseases in tomato leaves by taking advantage of the available computational resources to help farmers on crop maintenance, and in this way, avoid loose and waste of materials.

1.3. Question

Can the application of the Wavelet transform on an image before it gets into the Convolutional Neural Network, improves the accuracy of the diagnosis of diseases in tomato leaves using half of the computational resources?

1.4. Objective

To explore the use of techniques to reduce the computational complexity of convolutional neural networks and experimentally analyze their impact on performance to detect diseases on tomatoes leaves through images.

1.4.1. Specific objectives

The specific objectives of this research are:

- To integrate the Wavelet transform into the structured one of a convolutional neural network to reduce the size of the data.
- To evaluate experimentally the outcome of using the Wavelet transform to reduce the input data in a convolutional neural network.
- To apply a parallel computing approach for the training process of convolutional neural networks to reduce the training time.
- Experimentally to evaluate the outcome of using a parallel computing approach for the training process of a convolutional neural network.

1.5. Outline

The rest of this document is organized as follows. Chapter 2 contains literature related to solve the plant disease diagnosis through images giving an overview of what authors did to solve this issue and an analysis of their approaches that can contribute to the solution proposed on this project. Chapter 3 describes tools, methodologies, API's and resources to develop a solution for the plant disease detection through images that consider the computational complexity in the algorithm. Chapter 4 contains the proposal of a Wavelet Convolutional Neural Network to detect diseases in tomato leaves. Chapter 5 contains results and discussion for Wavelet Convolutional Neural Network experimentation. Finally, Chapter 6 contains the conclusions of this project.

Chapter 2

Review of related work

This chapter presents relevant works found in the literature that addresses the challenge of the plant disease diagnosis through images. We describe how the authors have boarded this issue, their approaches, and their results. The chapter concludes with a discussion about the results and open challenges related to the problem addressed in this grade work.

2.1. Plant disease diagnosis

In the literature, there are several techniques that authors have used to address the problem of plant disease diagnosis. We have grouped the techniques into two main approaches:

- Approaches based on digital image processing and machine learning.
- Approaches based on deep learning.

2.1.1. Approaches based on digital image processing and machine learning

The usage of simple techniques for diagnosing plant diseases is appropriate when the application is required to run on hardware with few resources and are connected to the Internet. One of these techniques was proposed by Nikos Petrellis [1]. His technique is simple, and it can be implemented in mobile applications. The author developed a framework called Spot Recognition System (SRS) based on 3 parameters: the number of spots, the area, and the gray level, those can

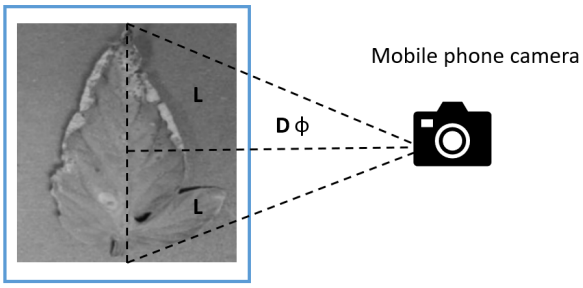

Figure 2.1: Leaf dimensions estimation

appear on the stem, the leaf, or the fruit plant. The main idea of this technique is to ignore the background of the image and only analyze the dots (Figure 2.1).

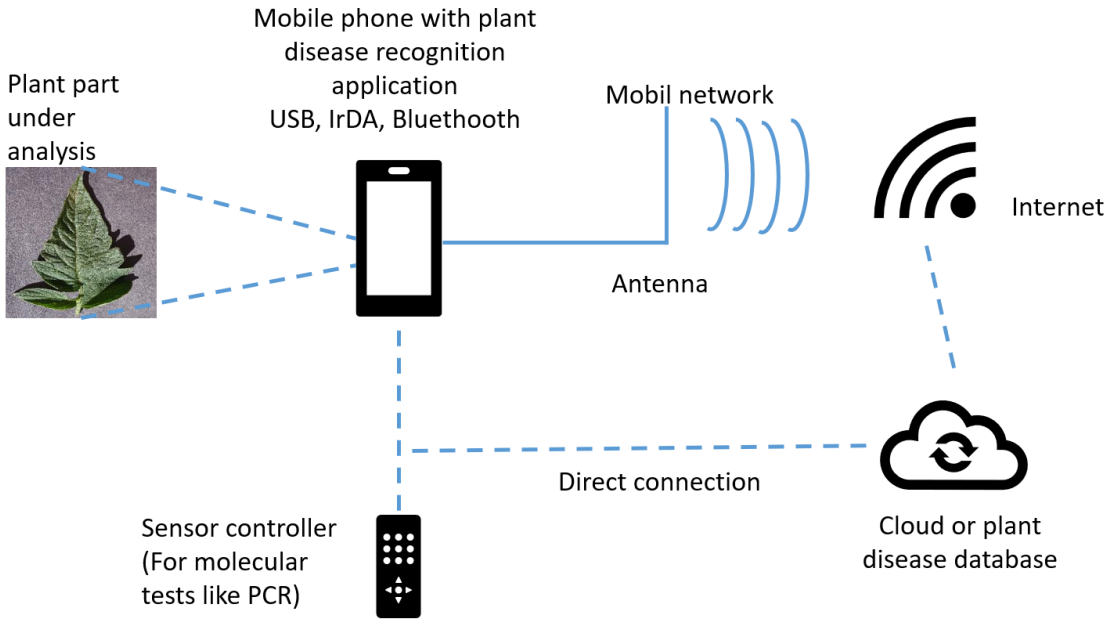

Figure 2.2: The plant disease recognition framework under development [1]

Figure 2.2 shows the SRS framework proposed in this project where a mobile with a considerable resolution takes a photo from a part of the plant, the output of the SRS plus temperature and humidity conditions are sent to a database or to the cloud through internet to be analyzed and obtain the disease diagnosis of the plant, this framework can use other recognition techniques as neural networks to increase the accuracy.

To tests this technique, Petrellis used tangerine tree leaves with lesions to diagnostic sooty

mold (fungal growth), citrus canker or scab. The project only shows two different results from the same image using different thresholds using an inverted gray image to pixels with high brightness. One equals 115 and other to 220, according to results, threshold reduces or increases the estimated spot area so MinArea=2 and low contrast=115 throws better results with fewer spot errors than 220 thresholds.

An alternative for the plant disease detection approach is the usage of image segmentation and soft computing techniques proposed by Vijai Singh et al. [7]. Vijai presents a technique based on separating or grouping an image into different parts. This proposal applies genetic algorithms for color image segmentation which generates meaningful solutions for complicated problems.

Figure 2.3: Input and output image of tomato bacterial spot leaf disease

Figure 2.3 shows the input of an image provided by a digital camera and the output that is improved by increasing contrast to mask all green pixels so they can be removed, and the rest of the cells are sent to be segmented by this genetic algorithm, then using color co-occurrence methodology extracts contrast, energy, local homogeneity and entropy features to be sent to SVM (Support Vector Machine) to done the classification.

To validate their results, leaves images were used for training and testing the model. Results show a significant improvement implementing a genetic algorithm for image segmentation and SVM for classification than using Minimum distance Criterion (MDC) with K-Means Clustering: 86.54% to 95.71%.

Many other works in literature have proposed to use techniques of machine learning as neural networks for diagnostic of plant diseases. The table 2.1 summarizes the most relevant proposals.

We can found that the precision of classical neural networks have a very high precision.

Table 2.1: List of contributions according to different NNs types for plant disease detection [2]

	Project	NNs type	Species	Disease/pets	Accuracy
Moshou D. et al [8]	Automatic detection of "yellow rust" in wheat using reflectance measurements and neural networks	Multi-Layer Perceptron and Self-Organizing Map	Wheat (Triticum sp.)	Yellow rust	99%
Lawrence GW. et al [9]	Remote sensing and precision nematicide applications for Rotylenchulus reniformis management in cotton	Self-Organizing Map-Neural Network	Cotton (Gossypium sp.)	Reniform nematode	97%
Li B. et al [10]	Hyperspectral identification of rice diseases and pests based on principal component analysis and probabilistic neural network	Probabilistic Neural Network	Rice (Oryza sativa L.)	Aphelenchoides besseyi, Rice leaf roller	95%
Abdulridha J. et al [11]	Detection and differentiation between Laurel wilt disease, phytophthora disease, and salinity damage using a hyperspectral sensing technique	Radial-Basis Function and Multi-Layer Perceptron	Avocado (Persea americana)	Laurel wilt (Lw) disease	98%
Mohanty SP. et al [12]	Using deep learning for image based plant disease detection	ConvNet	varied species	26 crop diseases	99.30%
Sladojevic S. et al [13]	Deep neural networks based recognition of plant diseases by leaf image classification	ConvNet	varied species	13 crop diseases	96.30%

However, it is necessary to emphasize that these works only perform the classification of one plant, while CNN has many classes and its performance is quite precise.

2.1.2. Approaches based on deep learning

Although previous techniques showed good results, they still have an improvement field that comes with deep learning techniques that have emerged recently. These have improved training results by a large amount of data. One of the most important algorithms that works with images is the convolutional neural network that specializes in those. In this section we describe the main related work based on the deep learning approach.

Sharada et al. in [12] proposed a plant disease detection based on convolutional neural network. In [12], authors used different deep learning architectures for plant disease detection and different representation of images with the goal to find out which combination has a higher accuracy to solve this issue. It was used a database of approximately 54,306 images of leaves of plants, which contains 38 class labels, each label is a crop-disease pair (healthy and diseased), the image size was standardized to 256 x 256 pixels, this dataset was provided by 'plantvillage.psu.edu' which is a research and development unit of the Pennsylvania State University that encourages farmers to use accessible technology to help to maintain their crops.

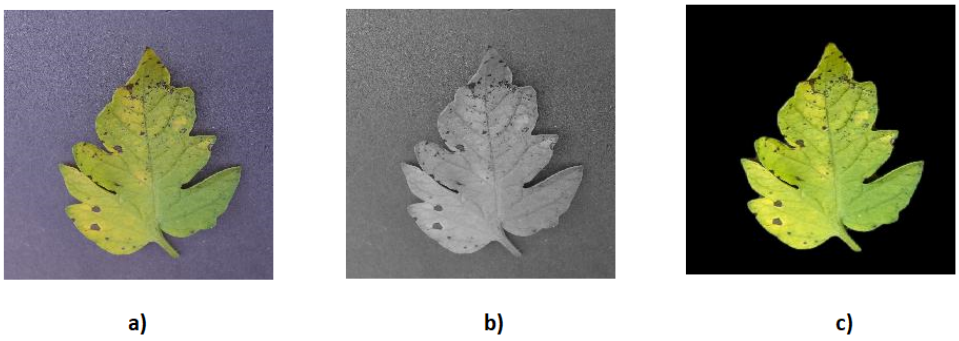

Figure 2.4: Sample images: a) leaf color b) leaf gray-scale c) leaf segmented

It was used 3 different versions of the database; color, grey-scale and a new database generated by image segmentation and removing the extra background information that does not add anything to the training but it only generates noise to it. Figure 2.4 shows samples of these 3 different versions of databases.

After they evaluated the approaches of each different architecture, AlexNet and GoogLeNet were selected for their design in large scale visual recognition. AlexNet has 5 convolution layers followed by 3 fully connected layers ending with a softMax layer. GoogLeNet architecture is deeper

and wider with 22 layers, but even when it has more layers and seems to be deeper than AlexNet, GoogLeNet has much fewer parameters due to its parallel convolution 3x3 and 5x5.

The setup consisted in 60 different model combinations, using different architectures (GoogleNet, AlexNet), training mechanisms (Transfer training, training from scratch), Data-set type ((Color, gray-scale, leaf segmented), and training-testing set distribution (80-20 %, 60-40%, 50-50%, 40-60%, 20-80%)

Training from scratch is as its name says, training the neurons without a previous training result, this one is different from transfer training that is a model who has learned previously. The training-testing distribution are batches of different size of the images database that helps to not over-fitting the model.

Results showed that GoogLeNet has a better accuracy than AlexNet in each of the 60 experimental configurations. And in each batch GoogLeNet showed to have better results using images as they were provided by PlantVillage this mean by images without modifications.

As shown in the previous work, Convolutional Neural Networks has a great accuracy on images RGB due to its handling with huge amount of data. However, it is possible to use hyperspectral images in order to improve their resultas. Kamlesh et al. in [2] analyses and compares results from different Neural Network algorithms for plant disease detection using hyperspectral data to improves their results.

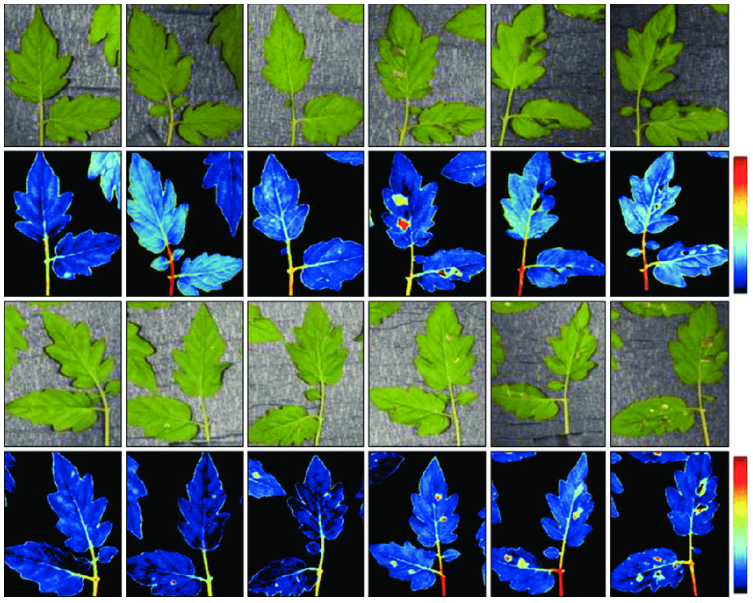

Figure 2.5: Hyperspectral image of a tomato leaves infected with Pseudomonas cichorii JBC1

Images are taken through a hyperspectral sensor that captures the data through the Near Infra-Ref (NIR) range of electromagnetic spectrum, this technique captures spectral information from plants. Then reflectance data is sent to different neural network models for disease detection. Hyperspectral data reveals more detailed information from plants than a regular image because it is more sensitive to leaf pigments like chlorophyll a, chlorophyll b, violaxanthin, b-carotene, neoxanthin, and carotenoids as shown in Figure 2.5 and as a result of the usage of this huge amount of information are the improvements on diagnosis accuracy.

According to authors plant diseases are manifested in 3 categories: pre-symptomatic, symptomatic and asymptomatic, because of this, it is necessary to provide as much information as required for an accurate classification, hyperspectral imaging provides this kind of information that improves disease diagnostic through neural networks.

2.2. Discussion

In this chapter, we reviewed the different techniques used by authors to solve the plant disease recognition through images issue, we can determinate that projects where used convolutional neural networks showed really good results but the computing capacity is enormous. Here is where it comes the necessity to develop improvements that reduce the computational complexity in the algorithms as it is the main objective of project.

Chapter 3

Convolutional Neural Networks

This chapter describes the methodologies, techniques, and resources used to accomplish the goal of building a deep neural network to identify diseases in plants. Some of the topics here are the components of a convolutional neural network, the wavelet transformation, the different platforms used to train it and more.

3.1. Convolutional neural network

The convolutional neural network is a class of deep neural network algorithms commonly used on image recognition and classification due to its strength on feature extraction by the application of several filters and highlights image borders. The CNN works directly with each image of the dataset to extract all possible information where filters slide over the image to create new matrices that will get into a fully connected layer for the classification face. A neural convolutional network can be configured as sequences of several kind of layers. The common layers used to build a convolutional neural network are:

- Convolutional layer
- Pooling layer
- ReLu Layer
- Dropout layer

- Full connected layer

Next we explain the main idea behind each kind of layer and how we can use them to build a convolutional neural network.

3.1.1. Layers

CNN consists of multiple layers that transform the image to get its classification through a sequence of different layers.

Convolutional

The convolution layer is the core building block of a CNN [14], its main purpose is to extract features from each image such as edges, color, and orientation. The convolution preserves the relationship between pixels by performing a dot product of two matrices, one is the filter, the learn-able parameter also called, kernel or feature detector, and the other one is the input, in this case, the image as a matrix as shown in figure 3.1

During convolution, the filter slides cross over the image horizontally and vertically with a given stride “the sliding size” giving as a result, a matrix that represents the activation map of the image. The kernel learns on the backpropagation from the model with each iteration.

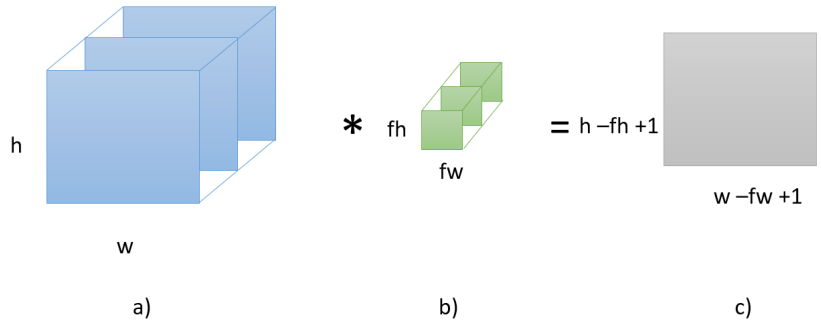

Figure 3.1: a) Input, RGB image matrix b) Filter, same deep than image c) Output, convolution result

Convolution operation consists of a dot product between 2 matrices, the kernel and the related size of the image, then add the multiplication results to get the integer that will be part of the

activation map, and depending of the stride value, kernel will continue sliding over the image to get next features until it gets through all image to get a new matrix as shown in Figure 3.2.

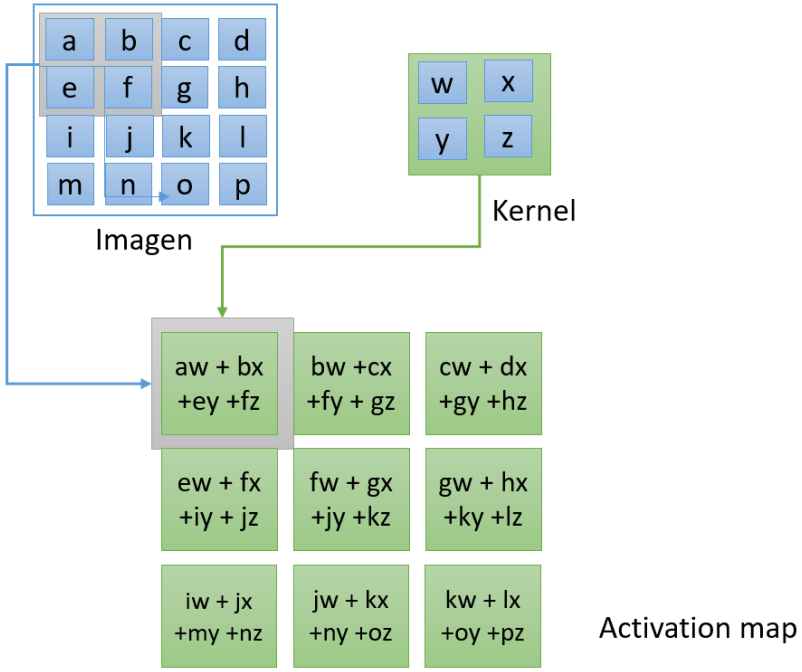

Figure 3.2: Convolution operation

ReLu

Rectified Linear Unit is applied to each pixel and replace all negative values on the activated map to zero. 3.3 shows ReLU operation and it is usually used after convolution layer to introduce non-linearity in the CNN and it "accelerates the convergence by six times." [14]

Pooling

Also called subsampling or down-sampling layer, it reduces the activated map by selecting the most relevant data in a given quadrant in the matrix. Figure 3.4 shows that the maximum value is the most important number within that window, Max Pooling has shown better results but also an average, min or sum of all elements are some operations used to down-sampling the input. Figure 3.4 illustrates this layer.

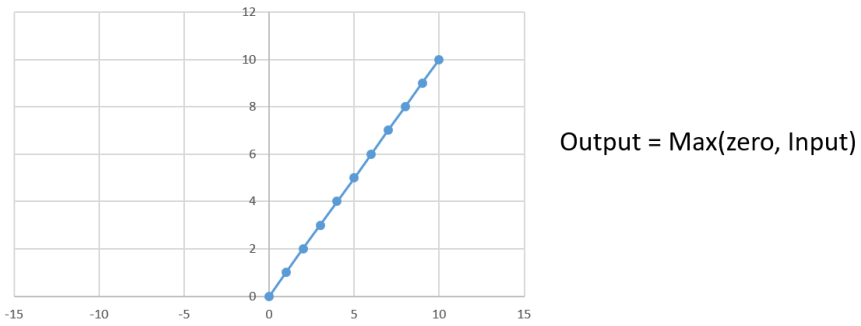

Figure 3.3: ReLU Operation

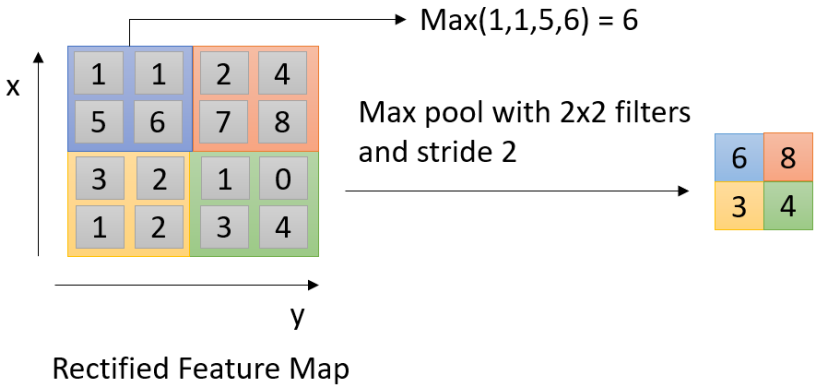

Figure 3.4: Max pooling illustration

Full connected

The previous layers are used for feature extraction, but when input gets into the fully connected layer, the goal is the classification. "The Fully Connected layer is a traditional Multi-Layer Perceptron" [15] where it uses softmax as the classifier algorithm in the output layer.

The activated map gotten in previous layers are flattened to get a vector. They feed the fully connected layer having fully connections with each neuron, they are computed as a matrix multiplication followed by a bias effect. The output layer will have the same dimension as the number of classes. Figure 3.5 shows how images were transformed after going through each layer to be latter flattened to get into the fully connected layer getting the classification for each one on the output layer that has the same neurons number as classes of the model.

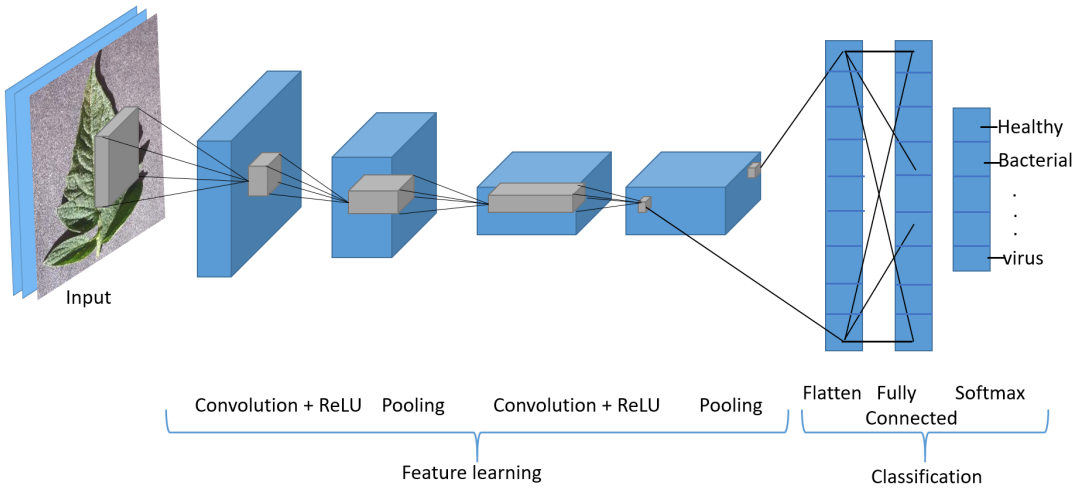

Figure 3.5: Convolutional Neural Network diagram

Dropout

Dropout layer is used to prevent over-fitting by deactivating neurons on the fully connected layer, this action improves the neural network by learning the same concept but using different neurons. Within some dropout increment helps the model to increase accuracy and decrease loss but at some point, it starts to going down. To get the optimal dropout value, it will be necessary to experiment with it.

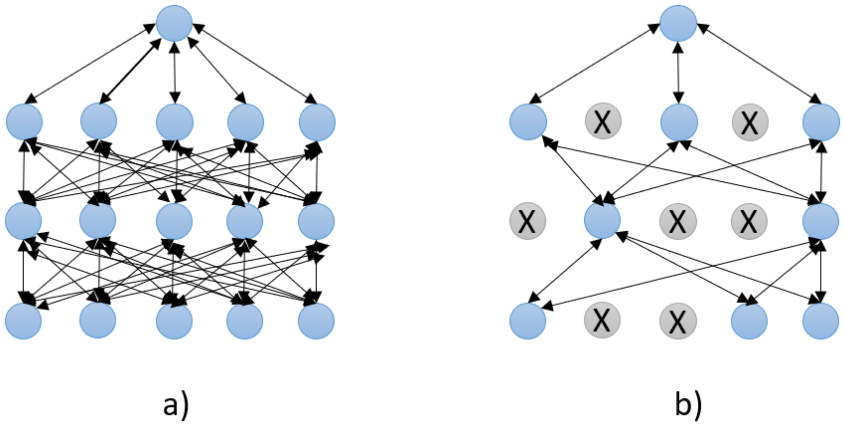

Figure 3.6: a)Standard Neural Network b)Neural Network after applying dropout

3.2. Convolutional Neural Network Computational complexity

The previous section describes the basic layers that constitute a CNN. A CNN architecture consists of several layers and it can be configured using different sequences of layers; some of the most relevant architectures in literature are: AlexNet [], VGG19[] and ZFNEt[]. This section, address their computational complexity.

The computational complexity is the calculation of the resources needed for the execution of an algorithm. Those resources can be expressed in terms of time and space, which varies on the input characteristics.

The time considers the elementary operations needed while the algorithm's computation, the actual time can change from a computer to another but generally speaking, the computational complexity calculates the number of steps needed for running it, and the space refers to the computer memory used for the execution of the algorithm.

The convolution computational complexity is calculated using the next equation:

$$Y_{r,p} = \sum_{q=1}^{Q} W_{r,q} * X_{q,p} \tag{3.1}$$

Where X is the input feature map, Y is the output feature map, W is the kernel, and P is the batch size. The formula above is applied on each convolution layer in the CNN architecture. In summary, the computational complexity of a CNN architecture depends on the number of convolutional layers, the number and size of the kernels used, and the number and size of the images.

3.3. Wavelet transform

Section above explains what is the computational complexity for a CNN and how heavy it can get while the architecture gets more complex. This section will define what a wavelet transformation is and how it can improve a conventional Convolutional Neural Network.

The Convolutional neural networks have demonstrated an excellent precision for image classification, however, one of the main problems they present is the data lost in some layers during

training that reduces accuracy and the big amount of memory needed to train due the large amount of features created during forward propagation. For this reason, some authors have analyzed and proposed spectral analysis techniques to improve the convolutional neural network. The following works describe this innovative technique.

The wavelet transform is a mathematical tool that focus on spectral aspects of the image, it is a decomposition hierarchically organized by next equations:

$$W_{\rho}(a_0,p,q) = \frac{1}{\sqrt{PQ}} \sum_{x=0}^{p-1} \sum_{y=0}^{Q-1} f(x,y)\rho_{a0,p,q}(x,y) \tag{3.2}$$

$$W_{\psi}^{b}(a,p,q) = \frac{1}{\sqrt{PQ}} \sum_{x=0}^{p-1} \sum_{y=0}^{Q-1} f(x,y)\psi_{a,p,q}(x,y) \tag{3.3}$$

This transformation is applied to each image f(x,y), P and Q are the image dimensions, W is the approximation coefficients and W the detail coefficients, a is the resolution level. The resolution is applied horizontally, vertically and diagonal directions. The sub-bands are LHa, HLa, and HHa, a = 1, 2,.., are the resolution level as shown in figure 3.7.

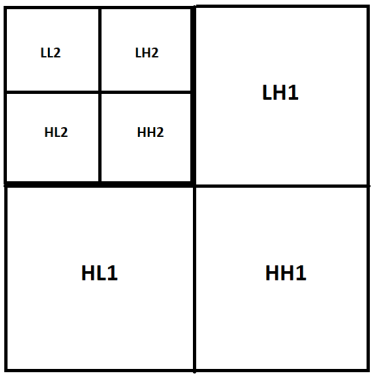

Figure 3.7: Level 2 of wavelet decomposition

This decomposition extracts the spatial and frequency characteristics of the image by reducing it almost at the half of its size but storing the main information and reducing computing capacity.

3.3.1. Use of the wavelet transform in image classification

This project written by Chiraz Ben Chaabane et al. in [16] proposes the usage of wavelet layers instead of pooling layers in a Convolutional neural network with the objective to improve predictions results of the handwritten digits database from the National Mixed Institute of Standards and Technology (MNIST).

The basic architecture of a convolutional neural network contains convolutional, pooling and fully connected layers and its main two steps: feature extraction "Convolution and Pooling" and classification "fully connected" work together with spatial structure data to efficiently learn and throw great prediction results. But even with such accomplishment, some structures in features can be lost because it takes aimlessly the maximum or mean value in the window of the image as output for next layers affecting accuracy.

Wavelet instead offers spatial and frequency characteristics overview of an image by a hierarchically organized decomposition by low-pass and high-pass filtering using down sampling and up sampling as shown in Figure 3.8. It is more precise due to its deterministic and not random characteristics selection.

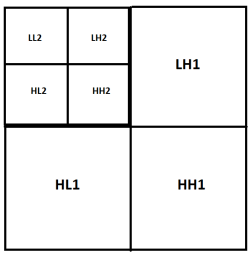

Figure 3.8: Wavelet decomposition

The Convolutional neural network architecture proposed by authors consists of pooling layer replacement by wavelet layer. It has two blocks of convolution followed by a wavelet decomposition and ending with a fully connected layer as shown in Figure 3.9. The First convolution has 32 filters and second has 64. Each filter is 5 x 5 in both convolution layers. Each image is 28 x 28 pixels. The database consists of 60,000 images for training and 10,000 for testing. The accuracy results for a basic CNN was 99.20% vs Wavelet CNN with a 99.40% demonstrating that by the elimination of randomness increases accuracy.

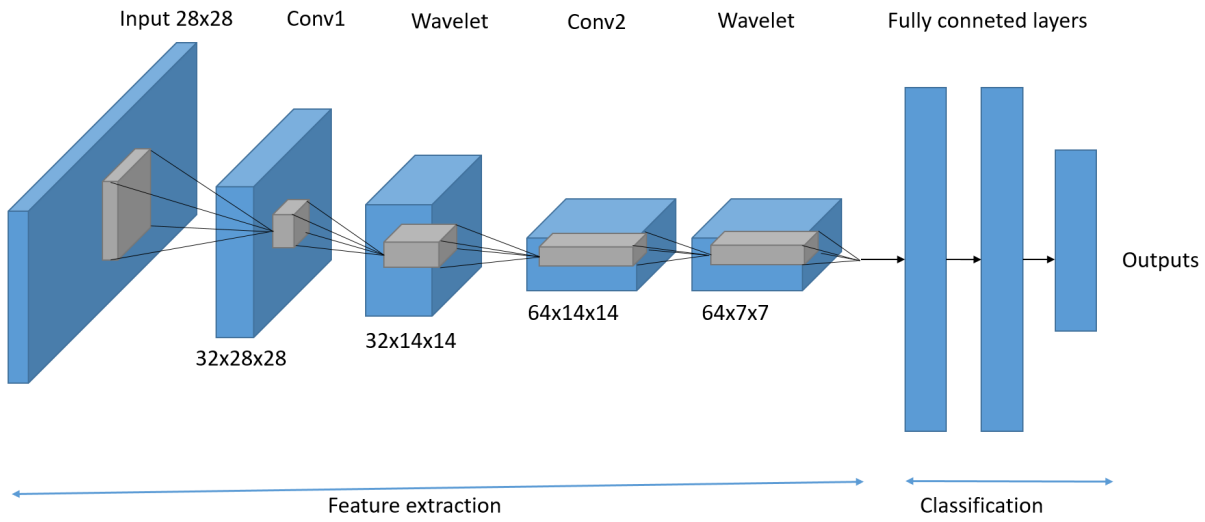

Figure 3.9: Wavelet convolutional neural network architecture

3.3.2. Wavelet Convolutional Neural Network

The previous work emphasizes on lost information using pooling layer in CNN so it suggests the usage of wavelet layers to fix the mentioned issue, next project focuses on how Wavelet Convolutional Neural Network achieves better results with much less memory needed in training making it easy to train and less prone to overfitting.

This project written by Shin Fujieda et al. in [17] improves texture classification through images by concatenating decomposed images by wavelet transformation. This classification is a huge challenge due texture is independent of the shape of the object in the image and it varies due to viewpoint, light and scale. This reformulated convolutional neural network architecture shown in Figure 3.10 reduces the size of the feature map by incrementing the stride to 2 in convolutional layers instead of using pooling layers. Additionally, it uses an average pooling layer that is more accurate to extract texture feature allowing to know to the neural network the connection with multiresolution preventing over fit and for each convolution a wavelet decomposition layer is incorporated to the algorithm.

They used two public dataset kth-tips2-b with 11 classes of 432 textures and DTD with 47

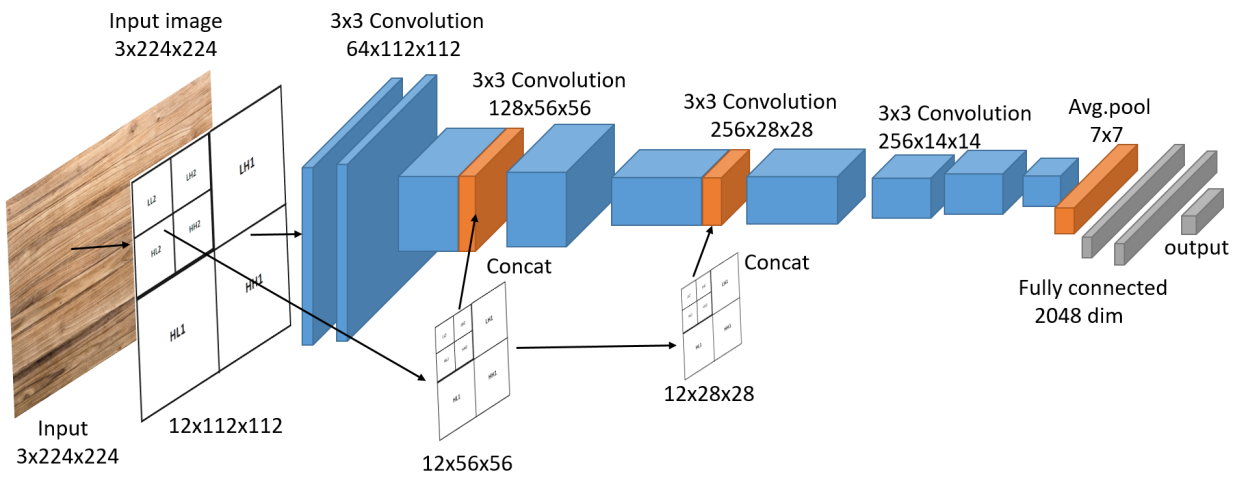

Figure 3.10: Wavelet convolutional neural network architecture

classes of 120 images in the wild to train the model and compare their architecture against AlexNet and T-CNN models from scratch. Results show more precision on Wavelet CNN by an accuracy of 60% than AlexNet with 48.3% and T-CNN with 49.6%. For fine-tuning the model, they compared it with Shearlet (62.3%), VGG-16 (70.7%), T-CNN (74.2%) and FC+ FV-CNN (VGG-16) (73.9%) models against Wavelet CNN (74.2%).

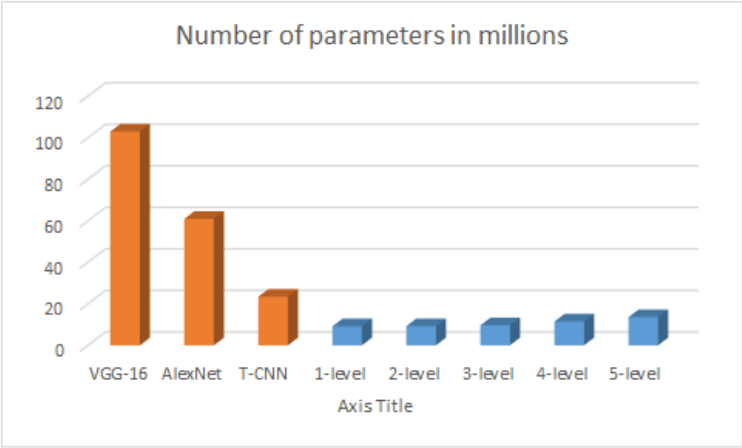

Figure 3.11: The number of trainable parameters in millions

But accuracy is not the only improvement in this novel architecture. The number of

parameters are significantly reduced. Even comparing the most complex model proposed in this paper (5-level) against AlextNet or VGG-M model reflects how much they decreased. Figure 3.11 shows the number of parameters in millions per model and those are the input that will get into the fully connected layer and the more parameters the more memory is needed for training.

3.4. Deep learning frameworks

The chapter above describes how wavelet transformation improves a Convolutional Neural Network by providing more precise images information, but it is also necessary to define tools that will help to build it. This section will explain the most common used frameworks to develop and train machine learning models.

3.4.1. Tensorflow and Keras

"Tensorflow is an open-source software library for numerical computation" [18]. It was developed by Google as a machine learning framework. It runs in GPUs and CPUs and it is implemented in C++ but it has python and C++ frontends. As Figure 3.12 shows, Tensorflow engine is the one that communicates with the supported devices, but on the top of it, run Keras and Estimator as higher-level APIs.

Keras is an API written in python and runs over TensorFlow, CNTK, or Theano. One of the advantages of it, it makes very simple to develop a deep learning model by reducing the number of actions required to start training it. For example, Keras saves you time creating the weights and biases used in each layer. Unlike Tensorflow that forces you to think about the different shapes of them, but Keras will save you on this step. Simply stated, "Keras prioritizes developer experience". [19]

Keras is a very popular framework used by Netflix, NASA, Uber, Yelp, Instacart, Zocdoc, Square, and many others, supports Convolutional and recurrent neural networks, runs on CPU and GPU and it supports multiple backend engines.

For its compatibility with Horovod, the distributed training framework, and its simplicity to develop, Keras is the framework selected to develop the Wavelet Convolutional Neural Network model for this project.

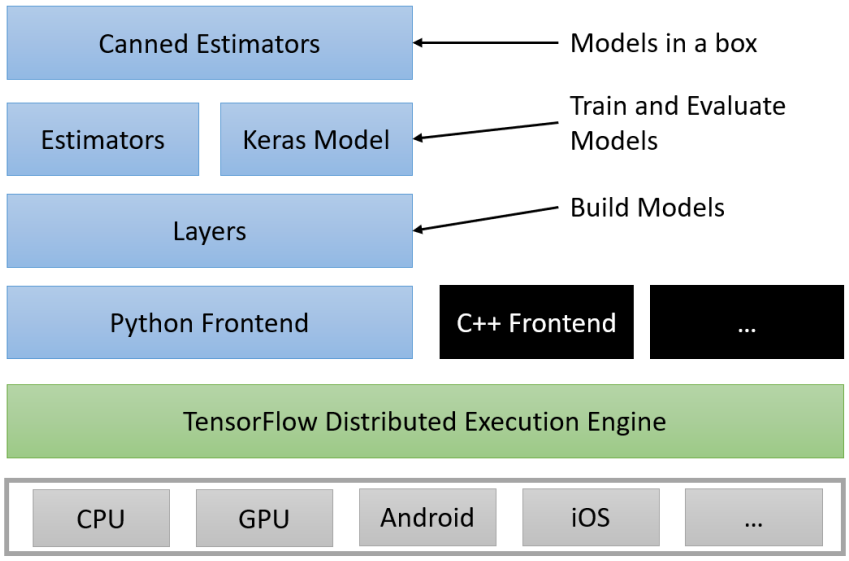

Figure 3.12: Tensorflow and Keras diagram

3.5. Distributed platforms

The frameworks explained in the section above make easier to build a deep learning model by a matter of just calling and using their APIs. But what it is not told, is the time and resources needed to rain a neural network. Depending on the database and images size used, it could take weeks or maybe months to have results from a single machine. This section will explain a distributed training framework used to optimize the execution time by the usage of all the available resources.

3.5.1. Horovod

Horovod is a distributed training framework with the purpose of speed up deep learning projects with TensorFlow. It was developed by Uber with the necessity of training deep learning models that use a lot of data in short times. Uber applies deep learning to improve user experience in fields like self-driving research to trip forecasting and fraud prevention. For this reason, it is Uber's interest to speed up training results.

Horovod combines data parallelism principles and ring-allreduce algorithm to distribute the training job. This consists on splitting the data on multiple nodes, each node will calculate the gradients by separately and later they will be averaged to update the model. This process will be repeated N times as number of epochs needed for the training as shown in figure 3.13.

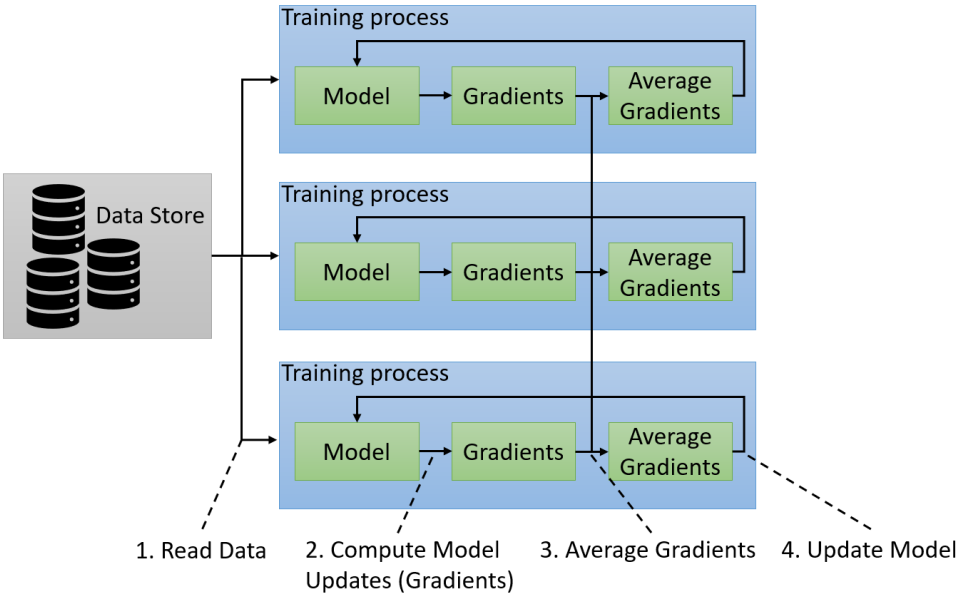

Figure 3.13: Distributed training model

Also Uber implemented the ring-allreduce algorithm to avoid the computational bottleneck on Horovod for the risk of using only one parameter server. This algorithm consists of the communication of N nodes with two peers 2*(N-1) times. Each node sends and receives data from a data buffer as shown in figure 3.14. With the ring-allreduce algorithm, worker nodes can average and disperse the gradients to all nodes without the usage of a parameter server. "Users utilize a Message Passing Interface (MPI) implementation such as Open MPI to launch all copies of the TensorFlow program" [20]. The next section will explain how MPI setups the infrastructure for the distributed jobs.

3.5.2. Message passing interface (MPI)

MPI is a specification of how a message passing library should be. Its standard is based on an MPI forum that consists of over 40 organizations, vendors, researchers, software library developers, and users and it focuses on the message passing parallel programming model. The goal of this standard is to provide a portable, practical, efficient, and flexible message passing interface.

MPI was designed for high performance on clusters, parallel computers and heterogeneous networks. In its beginnings it was designed for distributed memory but now it supports shared

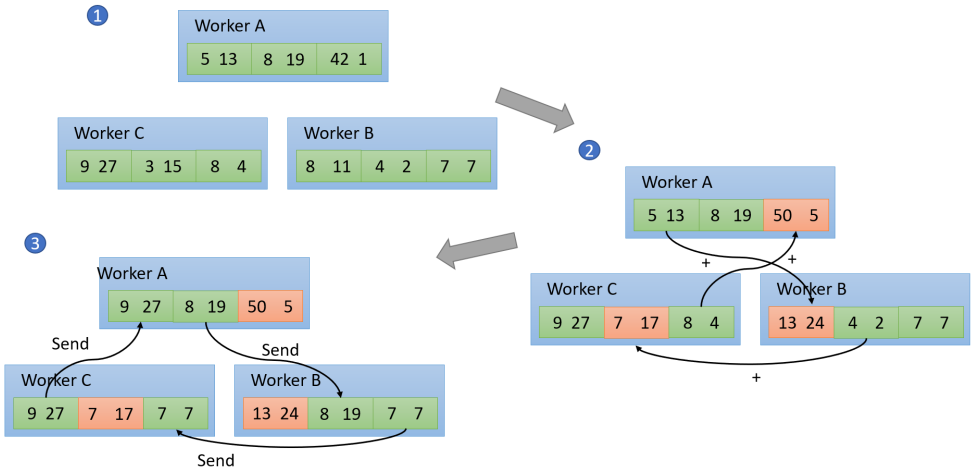

Figure 3.14: The ring-allreduce algorithm

memory and hybrid hardware platforms.

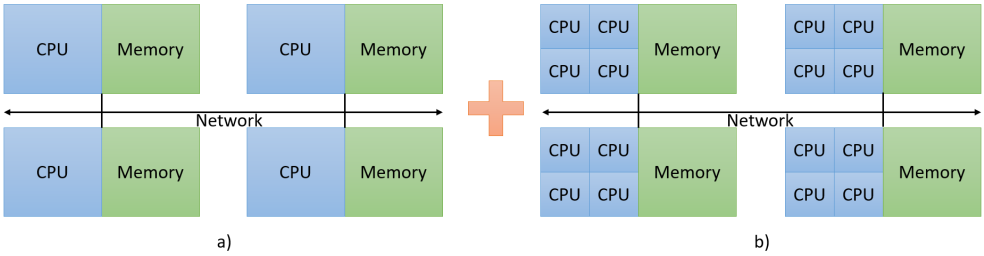

Figure 3.15: a) Distributed memory b) Shared memory

A Shared memory system shares the same memory with all the CPU-cores so for computer science, shared memory is memory that will be simultaneously accessed by multiple processes so they will share the same location in memory.

The distributed memory system is an architecture that multicore computers are connected through a network, and they can communicate with each other. For computer science, the processes will access the physical memory through a logical address space so it requires explicit commands to transfer data from a process to another.

MPI library has multiple versions due to the different compilers they are built for on Linux, BG/Q and Coral Early Access and Sierra clusters and MPI executables are lunched using SLURM. The next section will explain what this resource manager is about.

3.5.3. Simple Linux Utility for Resource Management (SLURM)

SLURM is a resource manager for any size of Linux clusters that is fault-tolerant and is highly scalable. Its three main functions are:

- Allocate exclusive and/or non-exclusive access to compute nodes to users during a period of time
- Provides a framework for start, execute and monitor work usually parallel job on a set of compute nodes
- Orchestrate resources by managing a queue of pending jobs.

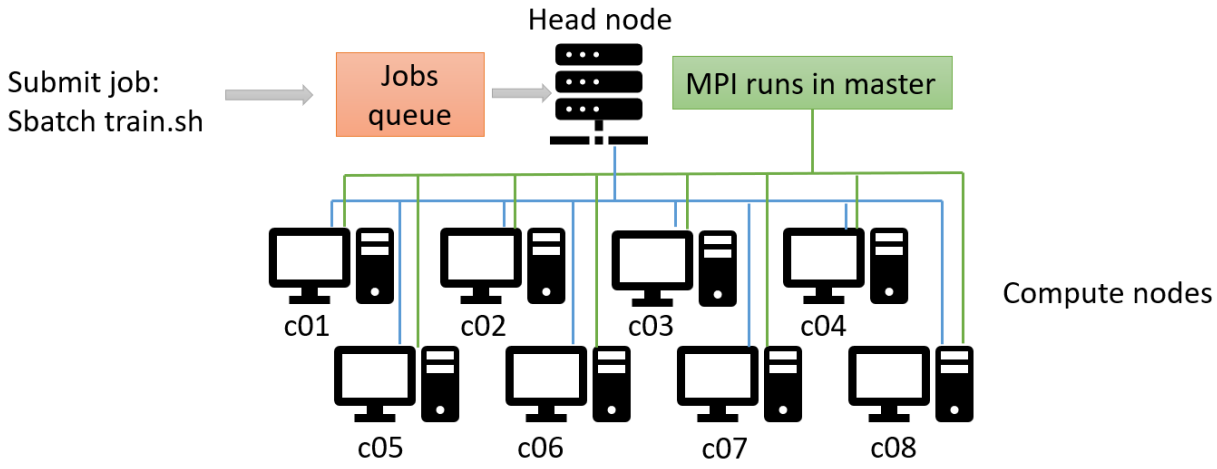

Figure 3.16: Distributed job diagram

Figure 3.16 shows SLURM and MPI tools working together to distribute a job. First, it is necessary to submit a job "train.sh" using SLURM, this script calls the CNN train script which calls Horovod to create Tensoflow copies that will be distributed on the compute nodes. This submission waits in a queue for resources. When resources are available SLURM enables the head node to execute the job "train.sh" that launches all the MPI processes that distribute the training trough the compute nodes.

In this chapter, we have addressed some of the main deep learning techniques, the Convolutional layers characteristics, parallelization tools, resources management tool and how we will apply it in our project to accomplish our goal.

Chapter 4

Disease recognition in leaves using Wavelet ConvNets

This chapter describes the proposal of a Wavelet Convolutional Neural Network to detect diseases in tomato leaves, its architecture, the dataset used for training.

4.1. Problem analysis

The convolutional neural network has shown great accuracy in image analysis. A conventional neural network cannot compete on results against it. Their architectures have shown significant improvement year by year, and an example of these is the Wavelet transformation implemented among the model.

This project proposes to analyze important Convolutional Neural Network architectures like AlexNet and ZFNet but giving as input, images that are transformed by the wavelet algorithm before entering the model. It will be analyzed the cost-benefit of using this transformation in deep learning algorithms to identify diseases in tomato leaves through images and at the same time analyzing the approach of a distributed vs local training.

4.2. Data description

The dataset images used was gotten from spMohanty's GitHub repo, it has 3 different representations of the same image: color (Original RGB images), grayscale, and segmented (RGB images with just the leaf segmented and color corrected).

Table 4.1: Tomato images dataset description

Class	Tomato Name	Images in database	With rotation	Train 70	Test 30
0	Late blight	1,909	11,454	8,018	3,436
1	Septoria leaf spot	1,771	10,626	7,839	3,187
2	Spider mites Two spotted spider mite	1,676	10,056	7,040	3,016
3	Target Spot	1,404	8,424	5,897	2,527
4	healthy	1,591	9,546	6,683	2,863
5	Bacterial spot	2,127	12,762	8,934	3,828
6	Early blight	1,000	6,000	4,200	1,800
7	Leaf Mold	9,52	5,712	3,999	1,713
8	Tomato mosaic virus	3,73	2,238	1,567	671
	TOTAL	12,803	76,818	53,773	23,045

This project focuses on Tomato images on color. The model uses the 3 RGB channels on training. Table 4.1 describes the total of tomato dataset from the repo of 12,803 images. How that number is not sufficient for a deep learning work, images where rotated on 90, 180 and 270 degrees to increase the samples on the database, also each image was rotated on mirror horizontally and vertically giving as a result 5 more samples per image with a total of 76,818. It is also important to mention that it was used the 70% of the images for training and 30 for testing. Figure 4.1 shows an example of a rotated image.

4.3. Convolutional Neural Network Architectures

AlexNet and WGG19 CNN architectures were selected at the beginning due to their accuracy reputation. Figure 4.2 shows the AlexNet architecture formed by 5 convolution layers followed by 3 fully connected layers ending up with a softmax activation. Table 4.2 shows the convolutional layers characteristics.

Figure 4.3 shows the VGG19 architecture that is formed by 16 convolutional layers, followed

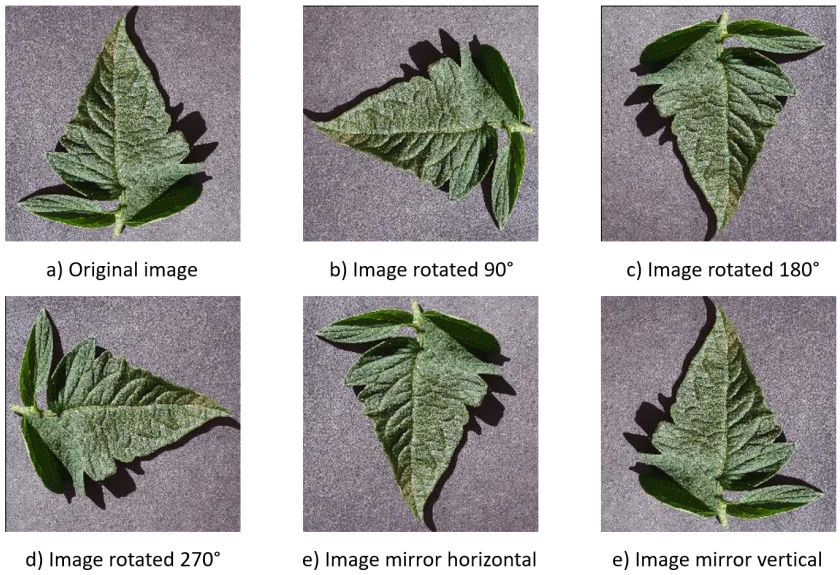

Figure 4.1: Tomato leave image rotation in database

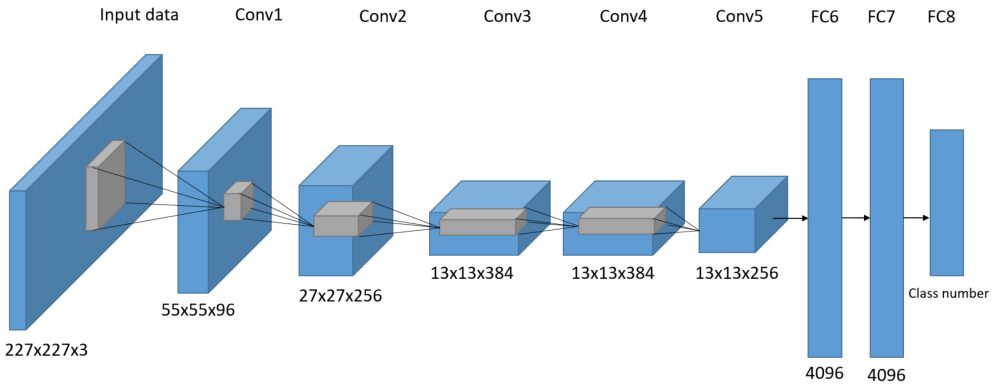

Figure 4.2: AlexNet architecture

Table 4.2: AlexNet Convolutional layers characteristics

Layer	Num. of filters	Kernel size	Stride
1st Convolution	96	11,11	4,4
2nd Convolution	256	11,11	1,1
3rd Convolution	384	3,3	1,1
4th Convolution	384	3,3	1,1
5th Convolution	256	3,3	1,1

by 3 fully connected layers. Table 4.3 shows each convolutional layer characteristics. As we can see, VGG19 has significantly more convolutional layers than AlexNet and additionally, they have

more filters with a small kernel size, becoming it a heavy algorithm to compute. The next section will show some results but an estimation showed that it could take months to finish all the test suite. Since there was not enough time to train and validate VGG19, it was necessary to replace it by ZFNet.

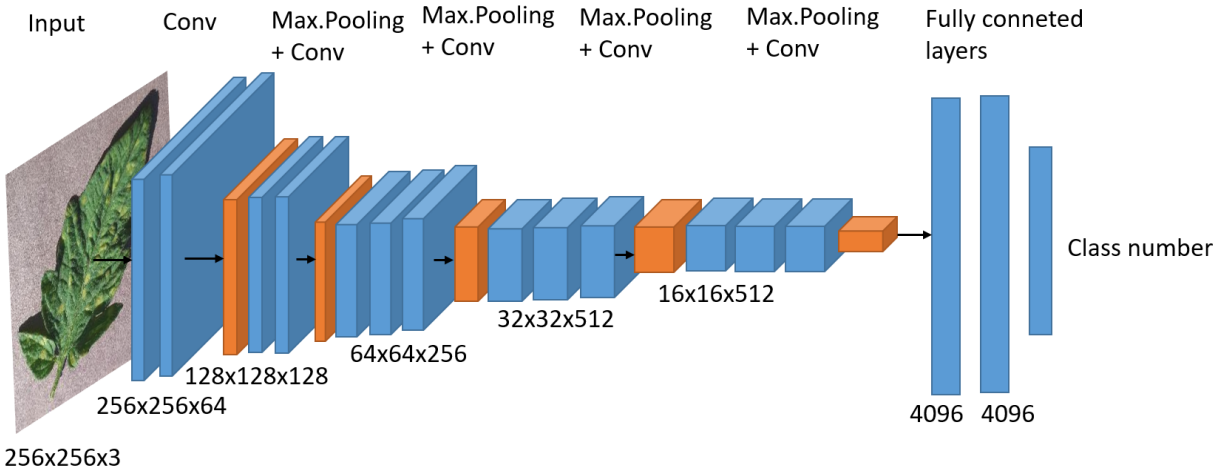

Figure 4.3: VGG19 architecture

Table 4.3: VGG19 Convolutional layers characteristics

Layer	Num. of filters	Kernel size	Stride
1st Convolution	64	3,3	1,1
2nd Convolution	64	3,3	1,1
3rd Convolution	128	3,3	1,1
4th Convolution	128	3,3	1,1
5th Convolution	256	3,3	1,1
6th Convolution	256	3,3	1,1
7th Convolution	256	3,3	1,1
8th Convolution	256	3,3	1,1
9th Convolution	512	3,3	1,1
10th Convolution	512	3,3	1,1
11th Convolution	512	3,3	1,1
12th Convolution	512	3,3	1,1
13th Convolution	512	3,3	1,1
14th Convolution	512	3,3	1,1
15th Convolution	512	3,3	1,1
16th Convolution	512	3,3	1,1

ZFNet architecture is formed by 5 convolutional layers and 3 fully connected layers. It

seems to be alike AlexNet but its differentiation is the convolution characteristics shown in Table 4.4.And Figure 4.4 also describes this architecture.

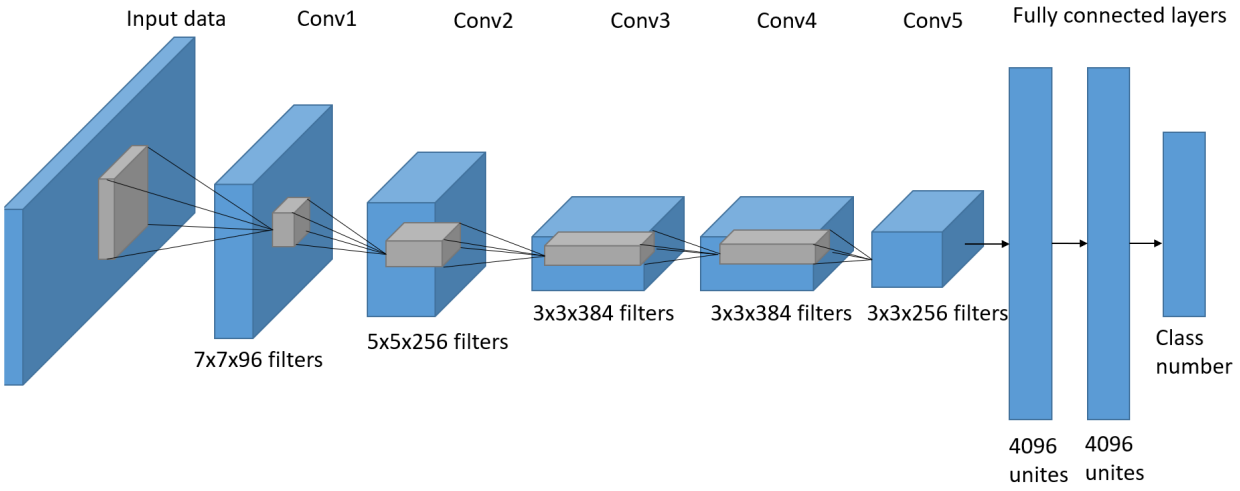

Figure 4.4: ZFNet architecture

Table 4.4: ZFNet Convolutional layers characteristics

Layer	Num. of filters	Kernel size	Stride
1st Convolution	96	7,7	2,2
2nd Convolution	256	5,5	2,2
3rd Convolution	384	3,3	1,1
4th Convolution	384	3,3	1,1
5th Convolution	256	3,3	1,1

Besides AlexNet and ZFNet architectures used for tests, each test suite is replicated but transforming each image with the Wavelet rbio1.3 algorithm as shown in Figure 4.5. In this way, there are 4 different architectures to test; AlexNet, Wavelet AlexNet, ZFNet and Wavelet ZFNet.

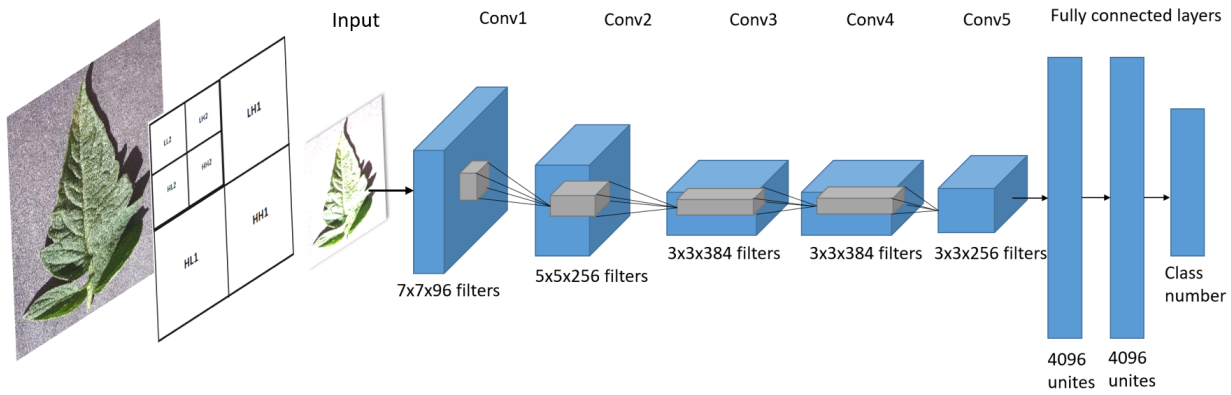

Figure 4.5: Wavelet rbio1.3 ZFNet architecture

Chapter 5

Results and Discussion

This chapter contains the results for the CNN and Wavelet CNN architectures experimentation, a comparison between them, and a discussion of the results.

5.1. Results

Se realizaron un total de 102 pruebas. Cada conjunto de pruebas constaba de 24 ejemplos en los que cada arquitectura fue entrenada con diferentes números de épocas en un número diferente de distribución de nodos. Algunos de los elementos de entrenamiento se mantuvieron persistentes en cada prueba, como la tasa de aprendizaje igual a 0.1, un tamaño de lote de 128 y un abandono de 0.4. La función de pérdida aplicada fue la crossentropía categórica y el optimizador utilizado para el entrenamiento fue Adadelta. [21].

Figures 5.1 and 5.2 show the accuracy behavior of each model; AlexNet and ZFNet, with and without the Wavelet transform on the test data during the training with sveral epochs (10, 20, 30, 40, 50, and 100), and using 1, 2, 4 and 8 nodes.

Table 5.1: AlexNet accuracy results

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	79.5%	91.2%	91.4%	96.9%	84%	87.4%
2	76.3%	78.3%	87.6%	96.5%	82.0%	80.2%
4	59.8%	56.6%	68.1%	93.2%	81.2%	66.5%
8	43.9%	39.3%	58.5%	43%	50.1%	41.7%

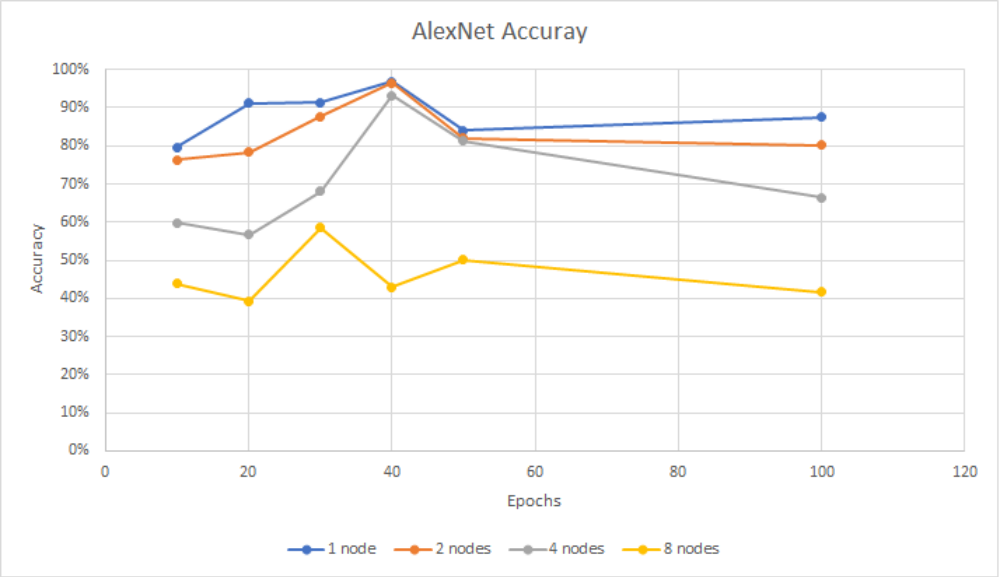

Figure 5.1: AlexNet Accuracy graph

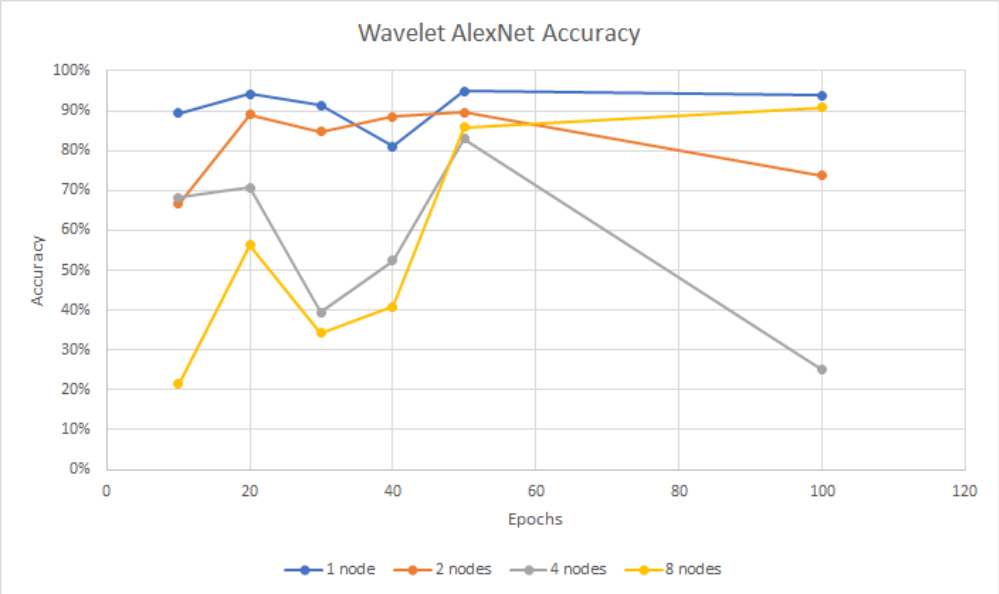

Figure 5.2: Wavelet AlexNet Accuracy graph

Table 5.1 shows accuracy results for the 24 AlexNet architecture model combinations with different number of epochs and the number of nodes distribution. Its highest accuracy was 96.9% using 1 node and 40 epochs. But using 2 nodes and 40 epochs; accuracy was 96.5 %, but with the benefit of reducing the time by half. Time expend on each execution is shown later.

Table 5.2 shows Wavelet AlexNet model with its 24 different combination results. Its highest accuracy was 94.9% using 1 node and 50 epochs, but an approximate accuracy was 90.8% using

Table 5.2: Wavelet AlexNet accuracy results

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	89.4%	94.2%	91.3%	81.1%	94.9%	93.8%
2	66.7%	89%	84.8%	88.5%	89.5%	73.7%
4	68.1%	70.7%	39.3%	52.3%	83%	25%
8	21.3%	56.2%	34.3%	40.9%	85.7%	90.8%

8 nodes and 100 epochs. The benefit of job distribution was that the time was reduced by 68% in these results and 50% in results above.

Table 5.1 and table 5.2 show the accuracy results for AlexNet and Wavelet AlexNet architectures. The accuracy of the Wavelet AlexNet got a little bit affected. But while accuracy was affected, the time was significantly reduced. Graphics from figure 5.3 and figure 5.4 show the amount of training and test time that took each architecture.

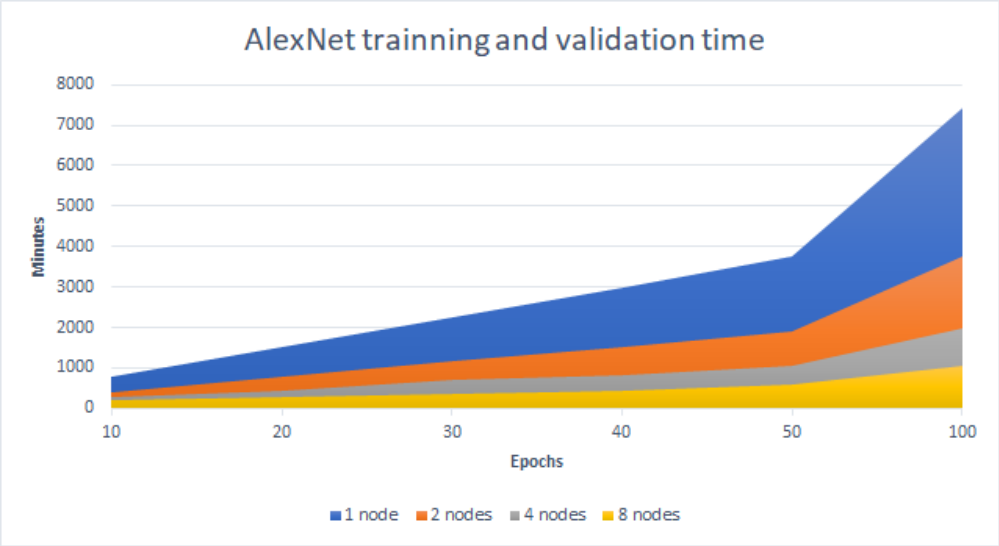

Figure 5.3: AlexNet time graph

Figure 5.3 shows that AlexNet took 7,418 minutes to train and validate using one node and 100 epochs while figure 5.4 shows that Wavelet AlexNet took 2,667. That means that Wavelet architecture used the 35% of the total time that AlexNet took.

Tables 5.3 and 5.4 show the amount of time in minutes for AlexNet and Wavelet AlexNet architectures train and test 24 combination executions. Time increases as the number of epochs, but it decreases as the number of nodes also increases.

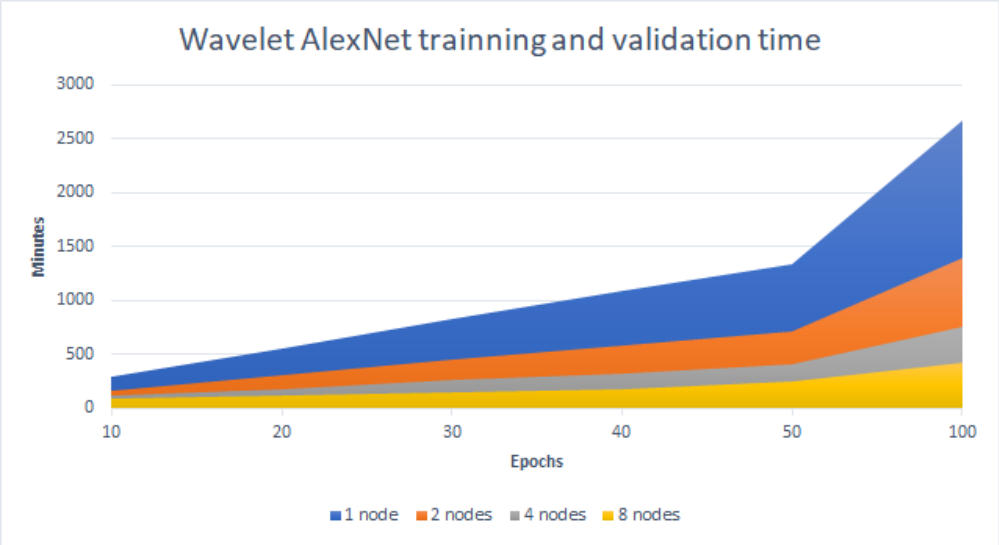

Figure 5.4: Wavelet AlexNet time graph

Table 5.3: AlexNet train and test time in minutes

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	768 min	1504 min	2268 min	2989 min	3746 min	7418 min
2	409 min	787 min	1160 min	1532 min	1907 min	3760 min
4	270 min	421 min	688 min	817 min	1043 min	1975 min
8	201 min	272 min	357 min	434 min	588 min	1061 min

Table 5.4: Wavelet AlexNet train and test time in minutes

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	296 min	559 min	826 min	1084 min	1339 min	2667 min
2	170 min	305 min	448 min	581 min	715 min	1390 min
4	125 min	180 min	268 min	324 min	403 min	761 min
8	91 min	121 min	151 min	182 min	243 min	425 min

The highest precision for the AlexNet was 96.91 % distributed in 1 node and 40 epochs. While for the Wavelet AlexNet was 94.21 % on the 50 epoch distributed in 1 node. Even when AlexNet needed less epochs, the time that Wavelet AlexNet took on epoch 50 is much less. Graphics from figures 5.5 and 5.6 represent better this behavior.

The Wavelet transformation is definitely an option to reduce time and resources for AlexNet architecture. In next section is presented the ZFNet and Wavelet ZFNet results.

Table 5.5 shows ZFNet accuracy results where its highest score was 98.5% distributed in 8

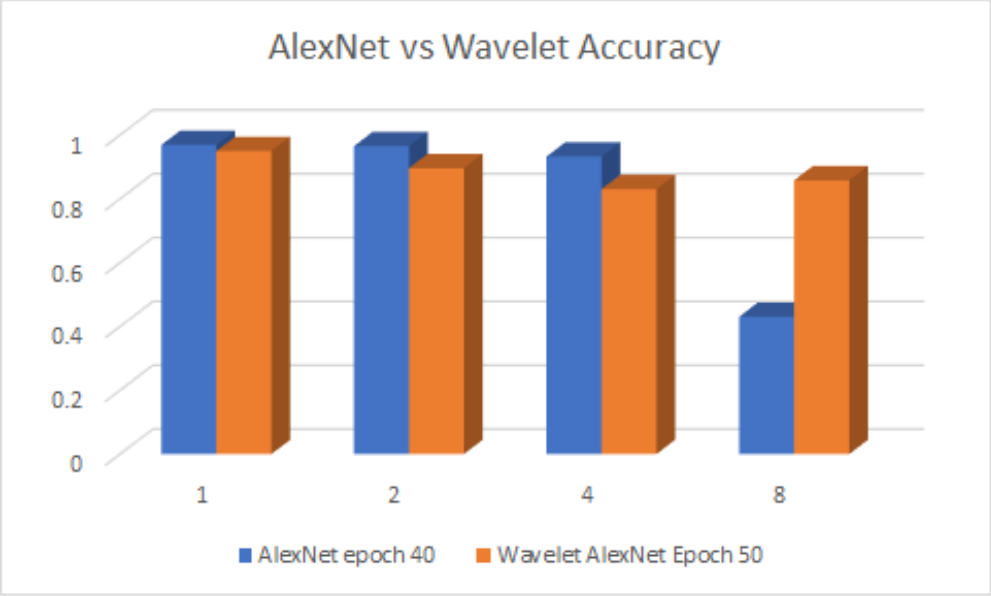

Figure 5.5: Accuracy AlexNet vs WaveletNet

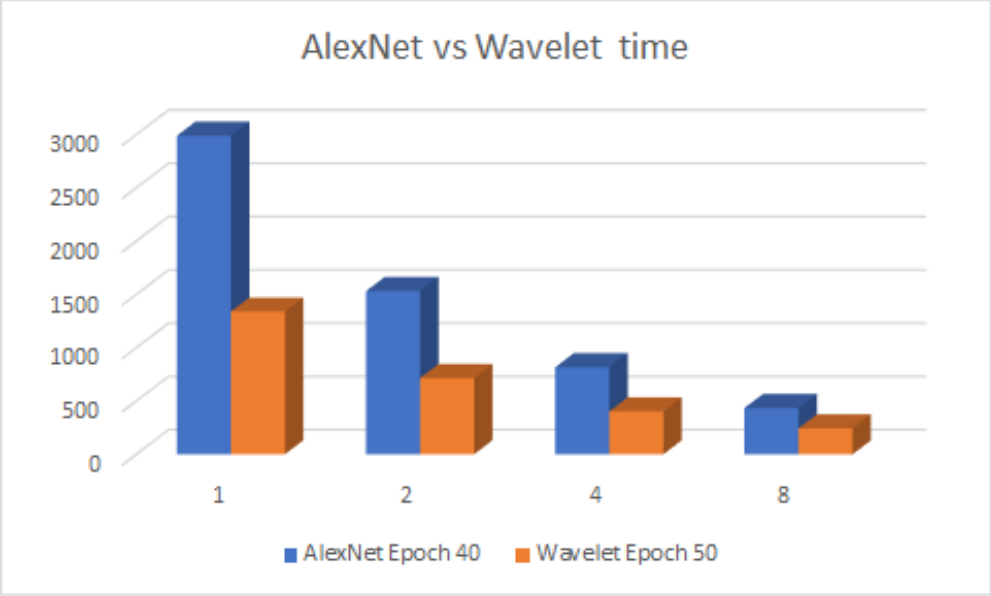

Figure 5.6: Trainning and test time AlexNet vs Wavelet AlexNet

nodes using 100 epochs, besides of reducing the time execution at 85% than executed in one node, ZFNet showed better results.

Table 5.6 shows Wavelet ZFNet accuracy results. Its highest score was 96.9% using 40 epochs and 2 nodes. But using 8 nodes with 100 epochs accuracy was 94% but with a reduction of time by 26.88%.

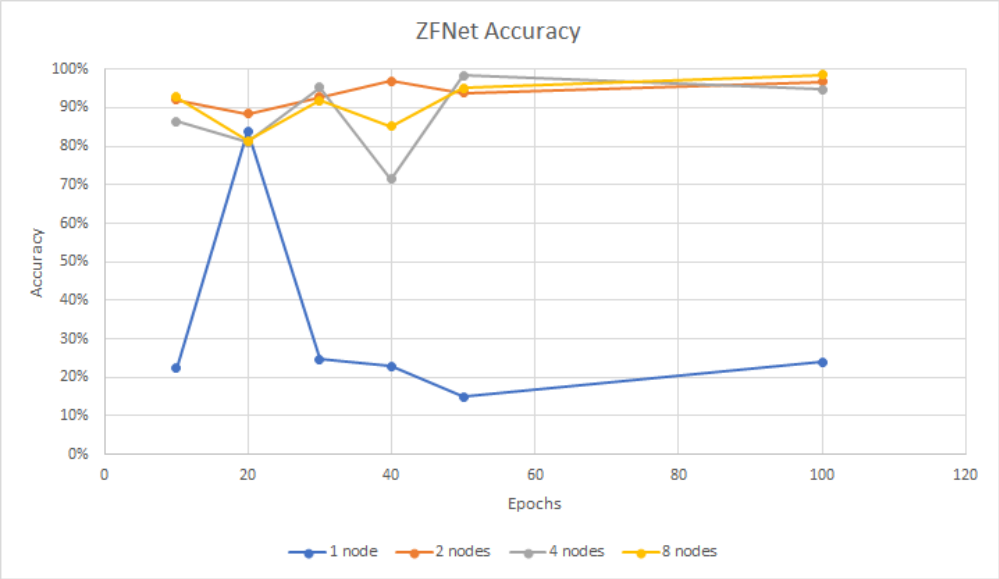

Figure 5.7: ZFNet Accuracy graph

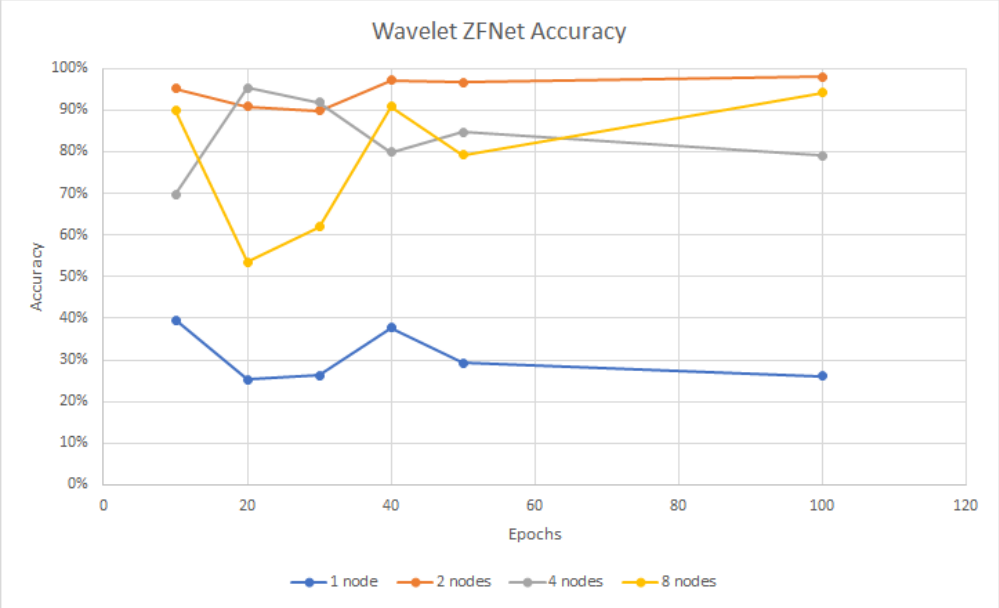

Figure 5.8: Wavelet ZFNet Accuracy graph

Results for ZFNet had a drastic change compare it with AlexNet. The highest accuracy for ZFNet was 98.51% in epoch number 100 distributed in 8 nodes. While Wavelet ZFNet accuracy got a 94.09% with same characteristics. Graphics in figures 5.7 and 5.8 show the time that each architecture took for training and test.

ZFNet took 5,612 minutes to train and test using 100 epochs distributed in 1 node while Wavelet ZFNet took 1,907 minutes. Wavelet ZFNet was reduced by 66% of the time that ZFNet

Table 5.5: ZFNet accuracy results

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	22.2%	83.9%	24.6%	22.9%	14.9%	24%
2	91.9%	88.3%	92.6%	96.9%	93.8%	96.8%
4	86.5%	81%	95.3%	71.3%	98.3%	94.8%
8	92.7%	81.3%	91.9%	85.2%	95%	98.5%

Table 5.6: Wavelet ZFNet accuracy results

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	39.4%	25.3%	26.2%	37.7%	29.2%	26.0%
2	95.1%	90.8%	89.8%	97%	96.6%	97.8%
4	69.7%	95.2%	91.8%	79.7%	84.6%	79%
8	89.8%	53.4%	61.8%	90.8%	79.1%	94%

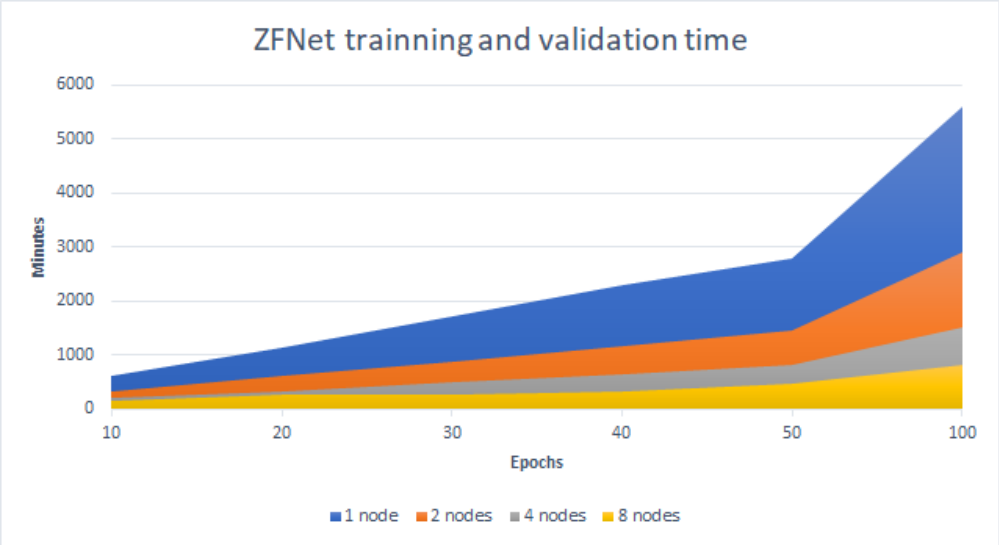

Figure 5.9: ZFNet and Wavelet ZFNet Accuracy results

Table 5.7: ZFNet train and test time in minutes

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	602 min	1139 min	1717 min	2288 min	2793 min	5612 min
2	320 min	609 min	891 min	1169 min	1468 min	2891 min
4	208 min	330 min	508 min	631 min	822 min	1524 min
8	154 min	261 min	276 min	338 min	463 min	833 min

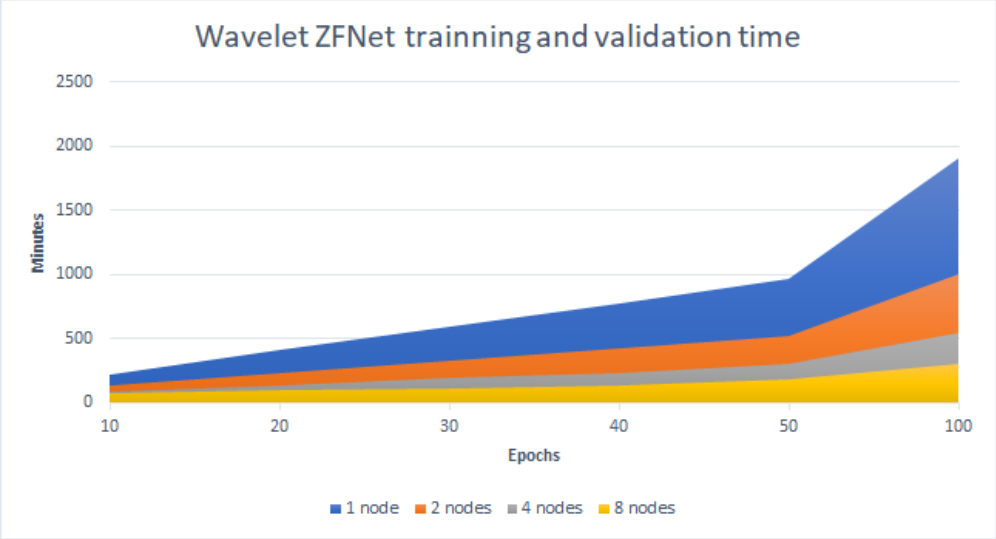

Figure 5.10: ZFNet and Wavelet ZFNet Accuracy results

Table 5.8: Wavelet ZFNet train and test time in minutes

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	219 min	411 min	598 min	781 min	971 min	1907 min
2	131 min	228 min	325 min	424 min	516 min	1009 min
4	91 min	133 min	195 min	237 min	299 min	546 min
8	73 min	94 min	116 min	137 min	180 min	310 min

took but the accuracy was not reduced in the same way. There are also some points where Wavelet ZFNet had higher accuracy than ZFNet with same epoch and distribution characteristics. Graphic 5.11 and 5.12 show this behavior for ZFNet and Wavelet ZFNet at epoch number with different distributions.

It was commented in chapter 4.3 that VGG19 architecture had to be stopped because it was not enough time to deploy all of the test suite. Table 5.9 shows some results obtained from it and table 5.10 shows the time that executions took. It took 7,991 minutes = 5.54 days for 10 epochs in 1 node distributed. It would possibly take more than 55 days for 100, almost two months for just to test only VGG19 architecture. Training and test execution for all of test suite would take months, more time that we have to present this project.

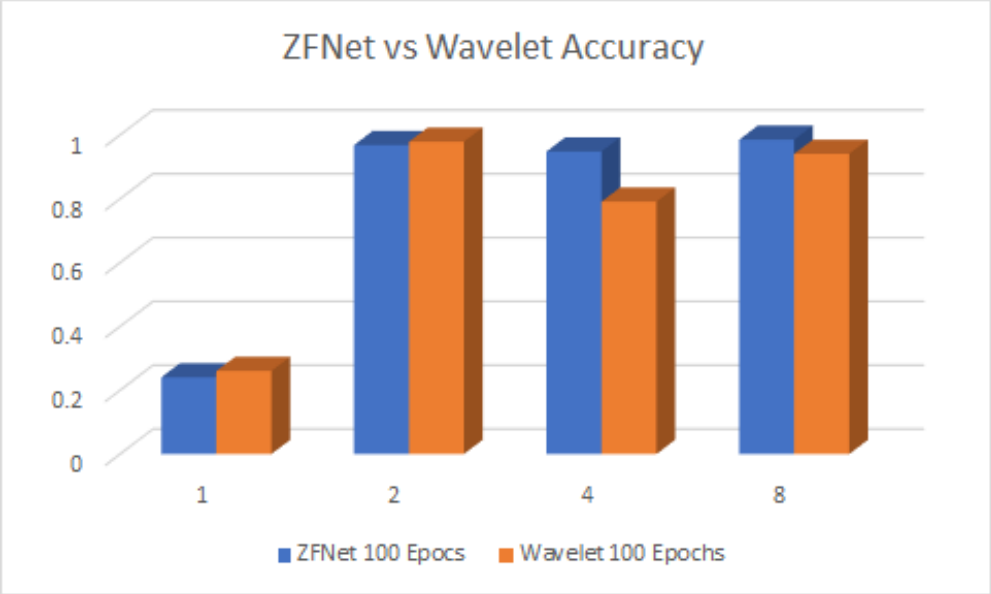

Figure 5.11: Accuracy ZFNet vs Wavelet ZFNet

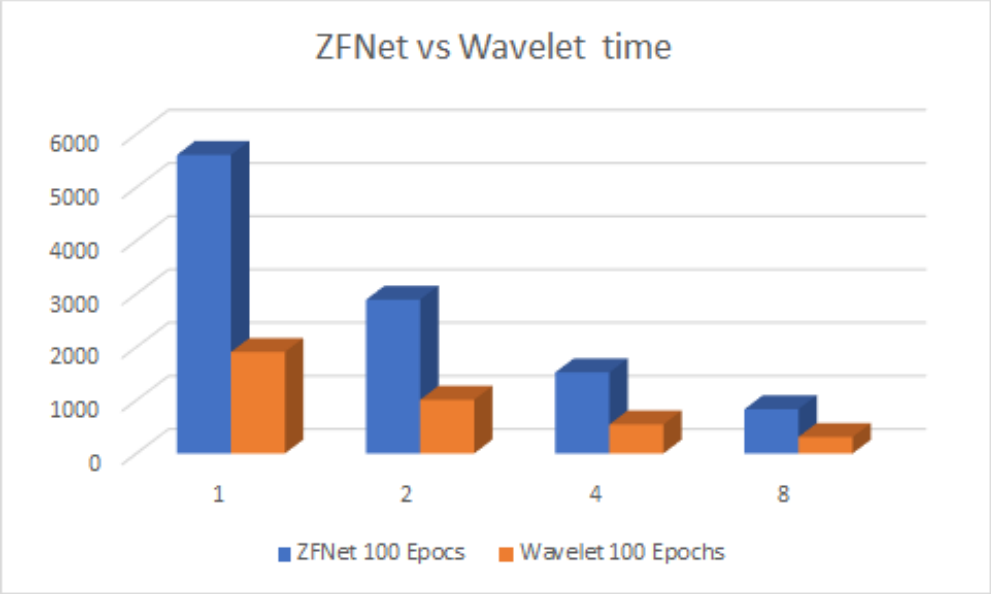

Figure 5.12: Trainning and test time ZFNet vs Wavelet ZFNet

5.2. Discussion

The Wavelet transformation is definitely an option to reduce resources and time for AlexNet and ZFNet architectures. At some point, accuracy was affected, but resources used decreased significantly, comparing it with accuracy results vs time expended, Wavelet transformation becomes an option to improve the computational complexity with very similar results but saving 65% of the

Table 5.9: VGG19 accuracy results

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	0.620942545	0.859920153	N/A	N/A	N/A	N/A
2	0.638604409	0.769571255	0.87016143	0.9 06439854	N/A	N/A
4	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A

Table 5.10: VGG19 train and test time in minutes

Nodes	10 epochs	20 epochs	30 epochs	40 epochs	50 epochs	100 epochs
1	7991	15911	N/A	N/A	N/A	N/A
2	4086	8107	12103	16134	N/A	N/A
4	N/A	N/A	N/A	N/A	N/A	N/A
8	N/A	N/A	N/A	N/A	N/A	N/A

time used.

The job parallelization using a different number of nodes by horovod also helped to decrease considerably the time expended on training and test execution. But resources were not the only thing improved with it, the accuracy results for ZFNet were also improved demonstrating that horovod is a useful API to improve time in CNN training and test.

Chapter 6

Conclusions

6.1. Conclusions

Nowadays, there are solutions that help us to identify diseases in plants through images in order to reduce the cost in time and money for possible crop losses due to such diseases.

Those solutions have implied the usage of artificial neural network, and they have been improved through time by incorporating the convolutional layers on their architectures. However, such solutions can be computationally complex and they can take several hours, days or in some cases even weeks to train and get results.

This project proposed the Wavelet transform to reduce computational complexity without loosing accuracy but only applying it on the images before getting into the Convolutional Neural Network and training them in distributed platforms.

Results in chapter 5 showed that in some cases accuracy was a little bit affected but computational complexity and execution time was significantly reduce.

The contribution of these findings is that the Wavelet transform improves computational complexity for any deep learning model without affecting the quality of information sent to the artificial neural network but taking care of one of the most important resources in life, 'time'.

6.2. Future work

This work accomplished the objective to compare the behavior of some existent Convolutional Neural Networks architectures with the incorporation of the Wavelet transform algorithm on the images database. But if we want to find out the best model to identify diseases in the tomato leave by images, it would be necessary to extend the tests by changing the learning rate, or analyze the behavior by eliminating the dropout or changing its value, increment the database, use different combination of number of nodes distribution, change the type of wavelet algorithm used, change the gradient or even change it for other CNN architecture. The key is to have enough time to play with parameters.

Bibliography

[1] N. Petrellis, "Plant disease diagnosisbased on image processing,appropriate for mobile phone implementation," in *Proceedings of the 7th International Conference on Information and Communication Technologies in Agriculture, Food and Environment*, Z. Andreopoulou and D. Bochtis, Eds., vol. 1498, 2015.

[2] K. Golhani, S. K. Balasundram, G. Vadamalai, and B. Pradhan, "A review of neural networks in plant disease detection using hyperspectral data," *Information Processing in Agriculture*, vol. 5, no. 3, pp. 354 – 371, 2018.

[3] S. C. Ivette Saldaña. México el principal exportador de jitomate en el mundo. [Online]. Available: https://www.eluniversal.com.mx/cartera/negocios/mexico-el-principal-exportador-de-jitomate-en-el-mundo

[4] R. Morales. México ingresa al top 10 de exportadores. [Online]. Available: https://www.eleconomista.com.mx/empresas/Mexico-ingresa-al-top-10-de-exportadores-agroalimentarios-20190805-0122.html

[5] O. de las Naciones Unidas para la Alimentación y la Agricultura. [Online]. Available: http://www.fao.org/news/story/es/item/469315/icode/

[6] G. d. l. L. Degante. [Online]. Available: https://www.inforural.com.mx/afectadas-mas-33-mil-hectareas-forestales-plagas-enfermedades/

[7] V. Singh and A. Misra, "Detection of plant leaf diseases using image segmentation and soft computing techniques," *Information Processing in Agriculture*, vol. 4, no. 1, pp. 41 – 49, 2017.

[8] D. Moshou, C. Bravo, J. West, S. Wahlen, A. McCartney, and H. Ramon, "Automatic detection of 'yellow rust' in wheat using reflectance measurements and neural networks," *Computers and Electronics in Agriculture*, vol. 44, no. 3, pp. 173 – 188, 2004.

[9] C. Hillnhütter, A. Schweizer, V. Kühnhold, and R. A. Sikora, *Remote Sensing for the Detection of Soil-Borne Plant Parasitic Nematodes and Fungal Pathogens*. Dordrecht: Springer Netherlands, 2010, pp. 151–165.

[10] L. Z. H. J. Z. L. Z. W. Bo, L. and S. Jingjing, "Hyperspectral identification of rice diseases and pests based on principal component analysis and probabilistic neural network," *Transactions of the Chinese Society of Agricultural Engineering*, 2009.

[11] J. Abdulridha, R. Ehsani, and A. De Castro, "Detection and differentiation between laurel wilt disease, phytophthora disease, and salinity damage using a hyperspectral sensing technique," *Agriculture*, vol. 6, no. 4, p. 56, 2016.

[12] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *CoRR*, vol. abs/1604.03169, 2016.

[13] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Comput Intell Neurosci*, pp.

1–11, 2016.

[14] M. Mishra. [Online]. Available: https://www.datascience.com/blog/convolutional-neural-network

[15] [Online]. Available: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[16] C. Ben Chaabane, D. Mellouli, T. M. Hamdani, A. M. Alimi, and A. Abraham, "Wavelet convolutional neural networks for handwritten digits recognition," in *Hybrid Intelligent Systems*, A. Abraham, P. K. Muhuri, A. K. Muda, and N. Gandhi, Eds. Cham: Springer International Publishing, 2018, pp. 305–310.

[17] S. Fujieda, K. Takayama, and T. Hachisuka, "Wavelet convolutional neural networks," 2018.

[18] [Online]. Available: https://opensource.com/article/17/11/intro-tensorflow

[19] [Online]. Available: https://keras.io/why-use-keras/#keras-prioritizes-developer-experience

[20] [Online]. Available: https://eng.uber.com/horovod/

[21] [Online]. Available: https://keras.io/optimizers/