# CHAPTER 2: LITERATURE REVIEW

## 2.1.0 Introduction

The problem context, and the project's goals were described in Chapter 1. This chapter guides the programmer in collecting and analysing relevant data for creating a system to recognize plant diseases. The benefits and drawbacks of various implementation strategies that can be employed to resolve a problem of this sort will also be covered in the research. This will help us decide which implementation strategy is best for the system.

## 2.2.0 Overview of Deep Convolutional Neural Networks for Image Classification

A fundamental issue in computer vision is image classification, which is the task of classifying images into one of many specified groups. It serves as the foundation for additional computer vision tasks such as segmentation, localization, and detection [1]. Classifying images into different groups based on certain characteristics may be simple for a human being but more challenging to have an automated machine doing the classifications because of the numerous issues at hand. Limitations include viewpoint-dependent object variability and the substantial in-class variability brought on by the presence of several object kinds. [2]. Traditionally, the classification issue was addressed by using a two-step method. Using feature descriptors, manually created features were initially retrieved from photos and fed into a trainable classifier. The disadvantage of this method was that the feature extraction stage's design, which was generally a complex job, had a significant impact on how well the classification task could be done [3]. Later on, deep CNN’s (DCNN’s) emerged as a result of the deep learning renaissance [4]. DCNN’s were driven mostly by use of bigger datasets, improved algorithms graphic processing units as well as better techniques such as max pooling for DCNN’s [5]. The ImageNet Large Scale Visual Recognition Challenge saw the most significant advancement, which has sparked strong interest in DCNNs, especially for image classification applications [6] when the winning entry, by A. Khrizhevsky et al, 2012 made use of a DCNN to classify approximately 1.2million images into 1000 categories, with outstanding, ground-breaking results. Since then, DCNNs have dominated ILSVRC iterations in general and its image classification component in particular [7].

## 2.2.1 Convolutional Neural Networks Basic Architecture

CNNs are cascaded networks because information only flows from their inputs to their outputs in one direction. CNNs are similarly biologically inspired to artificial neural networks. The brain's visual cortex, which is made up of layers of basic and specialized cells that alternate, [8], drives their structure. Although there are many different types of CNN designs, they all generally comprise of convolutional and pooling (or subsampling) layers that are organized into modules. These modules are followed by one or more fully linked layers, similar to a typical feedforward neural network. To create a deep model, modules are frequently stacked on top of one another. The standard CNN architecture for a task of classifying toy images is shown in Figure 2.1. Direct picture input into the network is followed by a number of convolution and pooling processes. After that, one or more fully connected layers are fed by the representations from these operations. The class label is finally output by the final fully connected layer.

Diagram

Description automatically generated

Although this is the most widely used base design in the field, numerous alterations to the architecture have been suggested recently with the aim of increasing picture classification accuracy or decreasing computation costs. While typical CNN architecture is only briefly discussed for the remainder of this part, later sections deal with a number of architectural design modifications that have led to improved picture classification performance.

## 2.2.3 Convolutional Layers & Pooling Layers

As feature extractors, the convolutional layers learn the visual features of the input images. Convolutional layers' neurons are grouped into feature maps. Each cell in a convolution layer has a receptive field, and each of these fields is connected to a group of nearby neurons in the layer above by a set of trainable weights, also known as a filter bank [9]. To construct a new feature map, inputs are superimposed with the learnt weights, and the outputs are then sent via a non - linear function. Although distinct feature maps within the same convolutional layer have varying weights so that several features can be recovered at each region, all neurons inside a feature map have weights that are mandated to be equal. [10]. In order to achieve spatial normalization to input distortions and translations, the pooling layers are used to lower the spatial resolution of the feature maps.

## 2.2.4 Fully Connected Layers

To extract more abstract feature representations while travelling through the network, various convolutional and pooling layers are typically layered on top of one another. Following these layers, there are fully connected layers that understand these feature representations and carry out high-level inference [11]. In addition, a global average-pooling layer that feeds into a straightforward linear classifier can be utilized as an alternative given that computing in the fully connected layers is frequently hampered by their compute-to-data ratio. Despite these attempts, further research is still necessary to compare the effectiveness of various classifiers built on top of DCNNs, which is an intriguing area of study.

## 2.3.0 Training

To train, ANNs and CNNs make use of a number of learning algorithms such as the back propagation algorithm to adjust their parameters such as weights, in order to obtain the optimal network result [12]. Backpropagation determines how to modify a network's parameters in order to minimize errors that have an impact on performance by computing the gradient of an objective function. Overfitting, which is poor performance on a held-out test set after the network has been trained on a small or even large training set, is a typical issue with training CNNs, and in particular DCNNs. This poses a significant barrier to DCNNs and has an impact on the model's capacity to generalize on unknown data.

## 2.4.0 Deep Learning renaissance

Although they didn't use backpropagation, the first cascaded multi-layered neural networks were trained in 1965 [13], making them possibly the earliest deep learning systems [14]. Although deep learning-like methods have existed for a long time, the term "deep learning" first became popular in 2006 when deep belief networks (DBNs) and autoencoders were used to initialize DNNs that had been trained using back propagation, [13], [14]. Prior to this, it was taught that gradient descent problems made deep multi-layered networks (including DCNNs) difficult to train and unappealing. [15]; [16]; [17]. In contrast, CNNs stood out and proved to be simpler to train than fully linked networks. [18]. Before the comeback of neural networks in 2006, medical picture segmentation was one of the successful applications that used CNNs for their image classification component in addition to the accomplishments mentioned above [19], facial recognition, detection, and verification [20]; [21]; [22], navigating off-road obstacles [23] additionally, general object categorization [24]; [25]. Ranzato, Poultney, Chopra, and LeCun in 2006 marginally improved the previous best-reported classification result by extracting sparse features using an energy-based model, which has a variety of applications including classification and segmentation [18] on the MNIST data set [10]. According to Hinton et al. (2006), their unsupervised pretrained DCNN model featured three key components and had a similar architecture to that of Le-Cun et al. (1998) but employed a far higher number of feature maps to create sparse features. An encoder analysed the input picture and generated a code vector of the imagery. A nonlinear-sparsing logistic module then converted the code vector into a sparse code vector. The sparse code vector was decoded by a decoder that created a restored version of the input picture; the result was then utilized to set the first-layer weights of the CNN. The next section demonstrates how this study, which was the first to employ DCNNs initiated by unsupervised training methods during the deep learning renaissance, inspired several additional unsupervised pretraining attempts between 2006 and 2011.

## 2.5.0 Deep Learning driven by GPU’s and Improved Algorithms

Driven by the benefits of unsupervised pretraining's speed and accuracy [13]; [26]. In order to learn hierarchical sparse features that were locally invariant to tiny shifts and distortions, Ranzatol employed a DCNN-like architecture in 2007. Their strategy, which included max pooling, produced results for the MNIST and California Institute of Technology benchmarks that were quite near to the state-of-the-art. DCNNs are still susceptible to significant shifts and distortions, despite their early success; this is still an unsolved problem. By combining nonlinear embedding algorithms with deep multi-layered architectures (including DCNNs), trained in a supervised manner, Weston, Ratle, and Collobert (2008) presented a more straightforward method to perform deep learning, arguing that the pretraining methods used by Hinton et al. (2006), Bengio (2007), and Ranzato (2007) were complicated and constrained. The Laplacian SVMs proposed by Belkin, Niyogi, and Sindhwani served as an inspiration for the resultant semi-supervised deep learning technique [27] It produced errors that were competitive with other shallow semi-supervised approaches on the MNIST data set [28] with the then-current deep learning techniques [13]; [29]; [30].

Jarrett, Kavukcuoglu, and LeCun (2009) and LeCun, Kavukcuoglu, and Farabet (2013) conducted a thorough study that examined the impact of the nonlinearities that follow convolutional filters in DCNNs, the performance of supervised, unsupervised, and randomly learned convolutional filters, and the benefits (if any) of using two stages of feature extraction as opposed to one (2010). On the MNIST (LeCun et al., 1998), CALTECH-101 (Fei-Fei et al., 2006), and NYU Object Recognition Benchmark (NORB—LeCun et al., 2004) data sets, they discovered that nonlinearities that comprise rectification and local contrast normalization were necessary for better accuracy and that better classification accuracy was obtained from two stages of feature extraction rather than one. They specifically achieved a new record on the unaltered MNIST data set, outperforming the previous best performance (Ranzato et al., 2006), by combining supervised reinforcement with unsupervised pretraining and a technique known as predicative sparse decomposition (PSD; Kavukcuoglu, Ranzato, & LeCun, 2010).

## 2.5.1 PSD – Predicative Sparse Decomposition Overview

Similar to the work by Ranzato et al. (2006), the PSD technique is based on an encoder decoder architecture that imposes sparse constraints on the feature vector by using a basic feedforward regressor that is trained to estimate a sparse solution for all the vectorized patches or their stacks in a predetermined training set. The PSD approach approximates the sparse codes, which makes it computationally cheaper and faster than other sparse coding schemes despite the fact that sparse coding techniques are often computationally expensive. The deep belief networks presented at the beginning of the deep learning renaissance made unsupervised (and partially supervised) pretraining, followed by supervised refinement, popular (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2007). Contrastive divergence techniques were employed in the most used unsupervised algorithms. [31]; [32], sparse constraints (Ranzato et al., 2006, 2007), or PSD [33]. The feature extraction filters for these algorithms are often trained so that representations at a certain stage may be reconstructed from representations at a previous stage. Although Bengio et al. (2007), Mairal, Bach, Ponce, Sapiro, and Zisserman (2008), and Ranzato and Szummer (2008) attempted to address this issue by integrating supervised criteria with unsupervised techniques, the main drawback of this approach is that the feature learning process is independent of the task. Furthermore, supervised learning has replaced unsupervised learning as the predominant paradigm for training DCNNs, despite the initial encouraging findings gained from unsupervised pretraining in recent years. Semi-supervised learning is more physiologically conceivable, though. Take children's knowledge of their surroundings, or more particularly, their ability to identify or categorize items, as an example. Their carers often offer them a few instances, which is comparable to semi- or poorly supervised learning, and they utilize this to generalize to items they haven't seen. Therefore, it is anticipated that future DCNNs will return to adopting semi-supervised techniques, such to those discussed in this section, in order to align our existing strongly supervised models closer to nature. To address the acknowledged problems with their unsupervised equivalents, these schemes will first integrate supervised criteria. Such advancements will eventually result in autonomous, unsupervised systems to handle the vast, already accessible amounts of unannotated data.

## 2.6.0 Conclusion

The implementation of image classification that is most appropriate for our use case (identifying plant diseases) will be examined in the following chapter (Chapter 3 - Methodology), and it will be determined how to best refine the result.