Plant Recognition, Detection, and Counting with Deep Learning

PORNNTIWA PAWARA

COLOPHON
This thesis was typeset with LaTeX based on the *classicthesis* template.
Cover: Inspired by the xkcd.com comic, "in CS, it can be hard to explain the difference between the easy and the virtually impossible."
(Creative Commons - Attribution-NonCommercial 2.5 Generic; free to copy and share)
Cover design: Pry, Wizard, Jarukit, Stang

Printed by: Gildeprint

The research for this dissertation was conducted at the Autonomous Perceptive Systems group, Artificial Intelligence Department, Bernoulli Institute, University of Groningen, The Netherlands.

university of groningen

Plant Recognition, Detection, and Counting with Deep Learning

PhD Thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus Prof. C. Wijmenga
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Tuesday 9 February 2021 at 11.00 hours

by

Pornntiwa Pawara

born on 24 September 1975
in Nakhon Ratchasima, Thailand

Supervisor
Prof. L.R.B. Schomaker

Co-supervisor
Dr. M.A. Wiering

Assessment Committee
Prof. P. Remagnino
Prof. T.M. Heskes
Prof. D. Karastoyanova

CONTENTS

ACRONYMS

BOW	Bag of Visual Words
CNN	Convolutional Neural Network
DA	Data Augmentation
FDA	Fruit Data Augmentation
HOG	Histogram of Oriented Gradients
KNN	k-Nearest Neighbors
MAE	Mean Absolute Error
OvA	One-vs-All
OvO	One-vs-One
RCNN	Region-based Convolutional Neural Network
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine

1 INTRODUCTION

1.1 Introduction

Plant Recognition using Deep Learning

The interest in automated recognition of plant species has been significantly increasing in the research community. It has been of great importance for several purposes, including farm management, botanical research, livestock food business, and edutainment. The success of plant recognition can be applied to several areas of application, for example plant disease detection (Mohanty, Hughes, and Salathé, 2016; Ferentinos, 2018), weed detection (Santos Ferreira et al., 2017; Lottes et al., 2018), and plant species identification (Goëau et al., 2013).

Plants can be identified by looking at their most discriminating parts, such as a leaf, fruit, flower, bark, and the overall plant, taking into account such attributes as shape, size, or color. However, the identification of plant species from field observation can be complicated, time-consuming, and requires specialized expertise (Gaston and O'Neill, 2004; Wäldchen and Mäder, 2018). Computer vision and machine-learning techniques have become ubiquitous and are now invaluable to overcome problems with plant recognition in research. Although these techniques have been of great help, image-based plant recognition is still a challenge due to several obstacles, such as a very large species diversity, intra-class dissimilarity, inter-class similarity, blurred resource images, high variance in illumination of images, and the limited availability of labeled datasets.

In machine learning, image-based plant recognition is a supervised classification problem and consists of two phases: the training phase and the testing phase. The training phase can be divided into image acquisition, image preprocessing, feature extraction, and classification (Rzanny et al., 2017; Wäldchen et al., 2018). Figure 1 shows a pipeline of a plant recognition process. The data acquisition step requires the collection of image data and the names of the plants (ground-truth values) associated with them, which may require expertise for some complex plants. The image preprocessing step can include a smooth or clean background of the images or some data-augmentation techniques such as cropping or rotating. For the feature extraction and classification steps, earlier research focused on various hand-crafted feature extraction techniques combined with various classifiers. Neto et al. (2006) successfully used the Elliptic Fourier shape feature method to identify crop and weed species. Nilsback and Zisserman (2008) classified various flowers by using low-level features, including color, the Histogram of Gradient Orientations (HOG), and Scale-Invariant Feature Transform (SIFT), and combining them with a support vector machine (SVM). In the testing phase, the trained model is used to predict the class of an unseen image.

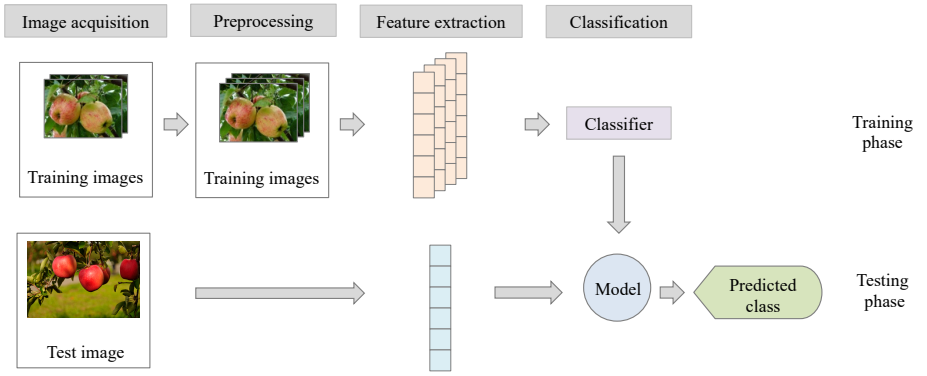


Figure 1: Image-based plant recognition pipeline.

Recently, the emerging of deep learning has changed the feature extraction process and frequently outperforms the previous classical feature

extractors in several research areas. As a consequence of high-performance computing, neural networks with millions of parameters can be trained rapidly and effectively. Several researchers trained convolutional neural networks (CNNs) for recognizing tasks on different plant species. Sun et al. (2017) developed a 26-layer deep learning model for large-scale plant classification. The work of (Dyrmann, Karstoft, and Midtiby, 2016) used CNNs on color images of weed and crop species. Several works modified the on-the-shelf deep learning architectures for the training process. The work of (Mohanty, Hughes, and Salathé, 2016) trained AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) and GoogleNet (Szegedy et al., 2015) on either sick or healthy plants from the PlantVillage dataset and achieved impressive accuracies. Another interesting crowdsourcing application is Pl@ntNet (Goëau et al., 2013), which shares and retrieves plant species by training CNNs on various parts of the plant, such as flowers, leaves, barks, or fruits.

Plant Detection and Plant Counting

In agricultural and orchard management, plant detection and counting systems are also crucial. Having reliable and accurate detection/counting systems can help organize and manage human and energy resources, resulting in benefits for sustainability, conservation, and ecology. The work of (Bargoti and Underwood, 2017) adapted transfer learning of the state-of-the-art object detection framework, Faster R-CNN (Ren et al., 2015), to detect mangoes, almonds, and apples from color and near-infrared images.

The results from the plant detection task can also be applied to the plant counting task. In general, there are two approaches to counting tasks: detection-based counting and regression-based counting. In addition to obstacles similar to those faced in plant classification, image-based plant counting has to deal with other problems, including the overlapping of plants, the occlusion of plants in the images, and the different perspective or different sizes of plants in the images. The work of (Rahnemoonfar and

Sheppard, 2017) simulated CNN for estimating the number of tomatoes in the images by creating synthetic images to obtain more training examples. We intend to improve counting accuracy by adding real plant objects to the training images.

1.2 Research Objectives and Questions

The main target of this thesis is to investigate various techniques, including data augmentation and classification, to improve plant recognition, plant detection, and plant counting. The following objectives and research questions are developed.

Objective 1: Comparing the traditional feature extractors to the deep learning techniques.
Question 1: Does deep learning outperform hand-crafted features and local descriptors in the plant domain? Can we modify the on-the-shelf CNN architectures so that they achieve better performance on plant classification? Do CNN architectures also work well on small datasets?

We start by building a baseline for plant recognition systems using deep learning architectures. We want to examine whether deep learning approaches overcome the existing hand-crafted feature extractors and some classification methods. Furthermore, we consider using a concise version of CNN architectures with a smaller number of neurons in the fully-connected layers for training. We want to examine whether they still work effectively with a better accuracy rate and less training time on small plant datasets.

Objective 2: Determine the effectiveness of the combination of data-augmentation (DA) techniques for plant classification problems.
Question 2: Does DA help to improve classification performance? If a single DA technique improves recognition accuracy, does the combination of DA techniques work more effectively?

Data augmentation helps to increase the number of training images in the training set. The earlier studies showed that applying DA on training

images often boosts classification performance. We propose several combinations of DA techniques and apply them to the training images of the plant datasets.

Objective 3: Develop a DA technique that helps to improve the fruit counting performance.
Question 3: Can DA techniques enhance the performance of the fruit counting task?

Besides the classification problem, we want to examine whether the DA technique can improve the fruit-counting system's performance. We propose a fruit-data-augmentation (FDA) technique and apply it to the training set of a novel fruit dataset. We then develop two fruit-counting approaches: regression-based counting, and detection-based counting, and train them on both the original training set and the augmented training set. For these two approaches, we evaluate the benefit of using FDA by comparing the performance of the models obtained from either the original or the augmented set.

Objective 4: Combine CNNs with One-vs-One (OvO) classification to enhance recognition accuracy.
Question 4: Do CNNs combined with the One-vs-One classification scheme outperform the traditional One-vs-All (OvA) classification scheme?

We propose using a One-vs-One classification scheme for a deep neural network for plant classification problems. We modify the neural network architecture, create a code matrix for encoding the novel labels, change the loss function, and change the classification method. We analyze the advantages of using OvO for multi-class classification problems. We further evaluate and compare the performance of the OvO and OvA classification schemes by training two CNN architectures on three plant datasets.

1.3 Thesis Overview

This thesis consists of six chapters. Each chapter introduces the main ideas which are related to the thesis objectives and research questions. The thesis is organized as follows:

Chapter 1- Introduction.

We list the motivations, research objectives, and research questions of this dissertation.

Chapter 2 - Hand-crafted features and deep learning for plant classification.

In this chapter, we compare the recognition performances of seven classification methods on three plant datasets. These classification methods include (a) a local descriptor with k-nearest neighbors (HOG with KNN), (b) a bag of visual words with the histogram of oriented gradients (HOG-BOW) combined with either a support vector machines (SVM) or multilayer perceptrons (MLP), and (c) the scratch and fine-tuned versions of the two well-known CNN architectures (AlexNet and GoogleNet). The results show that the fine-tuned CNN architectures outperform the local descriptor and the HOG-BOW techniques. We use the compact versions of AlexNet by reducing the number of neurons, resulting in excellent performance and a remarkable improvement in computing time. Moreover, the CNN architectures also perform well on a relatively small plant dataset.

Chapter 3- Data augmentation for plant classification.

This chapter describes the effects of six DA techniques (rotation, blur, contrast, scaling, illumination, and projective transformation) and several combinations of these DA methods on plant classification problems. We train two CNN architectures (AlexNet and GoogleNet), both scratch and fine-tuned versions, on three plant datasets. The results show that applying DA on the training images improves classification performance, especially when training the CNNs from scratch. Among these CNN models, the scratch AlexNet profits the most from DA. Furthermore, the combinations

of rotation and different illuminations or different contrasts help most for improving classification accuracy with the scratch CNN models.

Chapter 4 - Deep learning with data augmentation for fruit counting.

This chapter proposes a Fruit-data-augmentation technique. This method creates novel images by adding several fruits to the original training images with the same type of fruits. The FDA method helps to increase the number of images and the number of fruits in the images. FDA is applied to a training set of a fruit dataset. We evaluate the effectiveness of FDA by performing two approaches for fruit counting: a holistic regression-based approach and a detection-based approach, on the original training set and the augmented training set. We compare the performance of the models obtained from the original set to the models obtained from the augmented set. For the regression-based counting approach, ResNet50 and Inception-V3 are used. For the detection-based counting approach, Faster R-CNN and SSD-MobileNet are trained for fruit detection and afterwards, fruit counting. The results show that the regression-based approach profits from the FDA technique, whereas the detection-based counting approach does not benefit from the FDA method.

Chapter 5 - One-vs-one classification for deep neural networks.

In this chapter, we propose training deep learning architectures with a novel One-vs-One classification scheme for dealing with classification problems. We evaluate training two CNN models (ResNet50 and Inception-V3), both scratch and fine-tuned versions, with the OvO classification method and compare this approach with deep learning with a One-vs-All classification scheme. Both schemes are trained on three plant datasets and one fine-grained monkey dataset, with different train size splits (varies from 10% to 100% of the training set) and a different subset of classes taken from the datasets. The reason for performing training set subsampling is to study the effectiveness of OvO classification on the relatively small datasets. The results show that when CNN is trained from scratch, OvO classification significantly improves classification performance compared to the OvA classification scheme.

Chapter 6 - Conclusion.

This chapter concludes the dissertation and discusses the achieved objectives, provides answers to the research questions and gives directions for future research.

2

HAND-CRAFTED FEATURES AND DEEP LEARNING FOR PLANT CLASSIFICATION

The use of machine learning and computer vision methods for recognizing different plants from images has attracted lots of attention from the community. This chapter aims at comparing local feature descriptors and bags of visual words with different classifiers to deep convolutional neural networks (CNNs) on three plant datasets: AgrilPlant (Pawara et al., 2017b), LeafSnap (Kumar et al., 2012), and Folio (Munisami et al., 2015). To achieve this, we study the use of both scratch and fine-tuned versions of the GoogleNet (Szegedy et al., 2015) and the AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) architectures. We then compare them to a local feature descriptor with k-nearest neighbors and the bag of visual words with the histogram of oriented gradients combined with either support vector machines or multi-layer perceptrons. The results show that the deep CNN methods outperform the hand-crafted features. The CNN techniques also perform well on a relatively small dataset, Folio.

This chapter was published in:

Pawara, P.,Okafor, E., Surinta, O., Schomaker, L.R.B., and Wiering, M.A. (2017). *Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition*. International Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 479-486.

2.1 Introduction

The machine learning and computer vision community aims at constructing novel algorithms for object recognition and classification. Recently, different studies focused on the application of these algorithms on plant datasets. Plant classification is considered a challenging problem because of the variety and the similarity of plants in nature.

Contour based technique has been key feature for plant, and especially leaf, recognition (Guyer et al., 1986; Guyer et al., 1993; Woebbecke et al., 1995). A number of research studied the shape features for leaf edge patterns (Meyer, Hindman, and Laksmi, 1999; Du, Wang, and Zhang, 2007) and achieved good performance in classifying plant species.

The follow-up approaches to plant classification have considered using several local descriptors. The work of (Nilsback and Zisserman, 2008), used a joint learning approach of multiple kernels of local feature descriptors, including the histogram of oriented gradients (HOG) and the Scale-invariant feature transform (SIFT), a color histogram with a support vector machine (SVM) classifier for the classification of a 103 flower category dataset. The study showed that the classification performance could be improved by combining multiple features in a suitable kernel framework. An extension of the study of local feature descriptors with the use of the HOG-based approach (Xiao et al., 2010) for leaf classification showed superior performance over inner-distance shape context (IDSC) features on the Swedish leaf and ICL datasets. In (Latte et al., 2015), the authors worked on crop field recognition using the gray level co-occurrence matrix (GLCM) and various color features with artificial neural networks (ANNs). The performance was significantly increased when combining both types of features.

Other studies have focused on the use of segmentation and morphological based methods for recognizing plants using leaf datasets. For instance, Markov random field segmentation (Nilsback and Zisserman, 2010), which is optimized by using graph cut, has been used on the 13 classes of flowers. The study in (Munisami et al., 2015) combined several features of convex

hull, morphological, distance map, and color histogram with k-nearest neighbors (KNN) to classify different kinds of leaves and provided comparable accuracies with less computational time. Previous research in (Wang et al., 2014) proposed the combination of texture feature (intersected cortical model), and shape features (center distance sequence) with an SVM for classification of leaf images. Furthermore, on the use of segmentation based methods, (Zhao et al., 2015) showed that using learned shape patterns with independent inner-distance shape context (I-IDSC) features can be adopted for classification of both local and global information from leaves. The authors suggested that recognizing leaves by pattern counting approach is more effective than by matching their shape features.

Recently, attention has been shifted to the use of deep convolutional neural networks (CNNs) for plant classification. The work of (Lee et al., 2015), presented a leaf-based plant classification using CNNs to learn the discriminative features automatically. The authors in (Grinblat et al., 2016), employed a 3-layer CNN for assessing the classification performance on three different legume species, and they emphasized the relevance of vein patterns. The works in (Mohanty, Hughes, and Salathé, 2016; Sladojevic et al., 2016) used the deep CNN architectures to work on plant disease detection by focusing on leaf image classification. In (Mohanty, Hughes, and Salathé, 2016), the authors compared the performance of two CNN architectures: AlexNet and GoogleNet, with different sizes of training and test sets. The authors also worked on three image type choices: color, grayscale, and segmented leaf images. The results showed that the GoogleNet architectures steadily outperform AlexNet. Additionally, with the train-test set distribution of 80%-20%, the learning methods obtained the best results. In this study, we compare the performance of local descriptors and the bag of visual words with different classifiers to deep CNN approaches on three datasets: a novel plant dataset (AgrilPlant) and two already existing datasets.

Contributions: In this chapter, we compare seven different techniques and assess their performance for recognizing plants from images using three plant datasets; AgrilPlant, LeafSnap, and Folio. We created a novel

dataset, AgrilPlant, which consists of 10 classes of agriculture plants. For the comparison study, we make use of both scratch and fine-tuned versions of the GoogleNet and AlexNet architectures and compare them to a local descriptor (HOG) with k-nearest neighbors (KNN) and a bag of visual words with the histogram of oriented gradients (HOG-BOW) combined with either a support vector machine (SVM) and multilayer perceptrons (MLP). Using many experiments with the various techniques, we show that the CNN based methods outperform the local descriptor and the bag of visual words techniques. We also show that the reduction of the number of neurons in the AlexNet architecture outperforms the original AlexNet architecture and gives a remarkable improvement in the computing time.

Outline: The remaining parts of the chapter are organized in the following way. Section 2.2 explains the deep CNN architectures and the reduction of the number of neurons in detail. Section 2.3 entails brief discussions on the hand-crafted local descriptors. In Section 2.4, we describe the plant datasets and the experimental settings. Section 2.5 presents and discusses the performance of the various techniques. The last section concludes and recommends possible areas for future work.

2.2 Deep Convolutional Neural Networks

Deep convolutional neural networks (CNNs) were first introduced by (LeCun et al., 1989) and have become the most influential machine learning approach in the computer vision field.

A deep CNN architecture consists of several layers of various types. Generally, it starts with one or several convolutional layers, followed by one or more pooling layers, activation layers, and ends with one or a few fully-connected layers.

There are usually a certain number of kernels in each convolutional layer which can output the same number of feature maps by sliding the kernels with a specific receptive field over the feature map of the previous layer (or the input image in the case of the first convolutional layer). Each feature map that is computed is characterized by several hyper-parameters:

the size and depth of the filters, the stride between filters and the amount of zero-padding around the input feature map (Castelluccio et al., 2015).

Pooling layers can be applied in order to cope with translational variances as well as to reduce the size of feature maps (Sladojevic et al., 2016). They proceed by sliding a filter along the feature maps and outputting the maximum or average value, depending on the choices of pooling, in every sub-region.

A nonlinear layer or activation layer is conventionally applied to a feature map after each convolutional layer to introduce nonlinearity to the network. The Rectified Linear Unit (ReLU) function is a notable choice (Glorot, Bordes, and Bengio, 2011; Couchot et al., 2016) because of the computational efficiency and the alleviation of the vanishing gradient problem. The ReLU basically converts the input to its positive value or zero otherwise, i.e. $\int(x) = max(0, x)$.

The fully-connected layers typically are the last few layers of the architecture. The dropout technique can be applied to prevent overfitting because its random selection mechanism reduces the effective number of parameters in the gradient descent during training (Srivastava et al., 2014; Yoo, 2015). The final fully-connected layer in the architecture contains the same number of output neurons as the number of classes to be recognized.

2.2.1 AlexNet Architecture

The AlexNet architecture (Krizhevsky, Sutskever, and Hinton, 2012) follows the pattern of the LeNet-5 architecture (LeCun et al., 1989). The original AlexNet contains eight weight layers, which consists of five convolutional layers and three fully-connected layers.

The first two convolutional layers (conv{1,2}) are followed by a normalization and a max pooling layer. The last convolutional layer (conv5) is followed by the max pooling layer. Each of the sixth and seventh fully-connected layers (fc{6,7}) contain 4,096 neurons. The final fully-connected layer (fc8) contains 1,000 neurons because the ImageNet dataset has 1,000 classes to be classified. The ReLU activation function is applied to each of the first seven layers. A dropout ratio of 0.5 is applied

to the fc6 and fc7 layers. The output from the fc8 layer is finally fed to a softmax function.

In our study, the original AlexNet architecture is adapted by reducing the number of neurons in the fc6 and fc7 layer from 4,096 neurons to either 256, 512, and 1,024 neurons in both layers. The idea behind this is to increase the computational performance and mitigate the risk of overfitting (Xing and Qiao, 2016). We performed preliminary experiments on the AgrilPlant dataset to choose the best number of neurons. The results of this experiment are shown in Table 1. It shows that 1,024 neurons are the most efficient in terms of accuracy and it provides 34% improvement in training time compared to 4,096 neurons. Consequently, we set the number of neurons in the fc6 and fc7 layers to 1,024 for all datasets. The AlexNet architecture used in our works is shown in Figure 2.

Table 1: Accuracy comparison among different numbers of neurons and time improvement compared against 4,096 neurons in the AlexNet architecture on the AgrilPlant dataset. The results are reported with test accuracies and standard deviations using five simulations.

Number of neurons	Accuracy	Time improvement
4,096	88.30 ± 1.34	-
1,024	$\mathbf{89.53 \pm 0.61}$	34.06
512	89.13 ± 1.24	39.09
256	88.90 ± 1.35	41.08

2.2.2 GoogleNet Architecture

GoogleNet, presented in the work of (Szegedy et al., 2015), is among the first architectures that introduced the inception module that greatly dropped off the large amount of trainable parameters in the network. The inception module uses a parallel combination of 1×1, 3×3, and 5×5 convolutions along with a pooling layer. Additionally, the 1×1 convolutional filter is added to the network before the 3×3, and 5×5

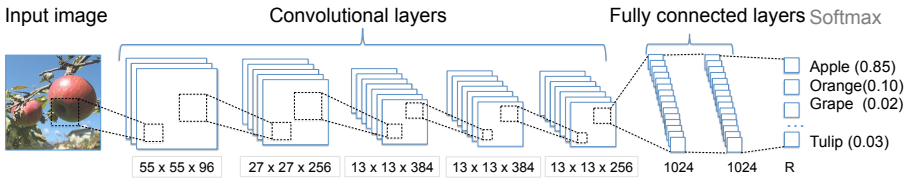


Figure 2: The AlexNet architecture used in our work. R in the fc8 layer is the number of neurons, which represents the number of classes in each dataset, which are set to 10, 184, and 36 for the AgrilPlant, the LeafSnap, and the Folio dataset, respectively.

convolutions for dimensionality reduction. This is called the "network in network" architecture (Lin, Chen, and Yan, 2013).

The GoogleNet architecture uses 9 inception modules, containing 22 layers along with four max pooling layers, and one average pooling layer. The ReLU is used in all the convolutional layers, including those inside the inception modules. To deal with the problem of vanishing gradients in the network, inspired by the theoretical work by (Arora et al., 2014), two auxiliary classifiers are added to the layers in the middle of the network during the training process (Yoo, 2015). A dropout ratio of 0.4 is applied to the softmax classifier. The illustration of the convolutional layers and the inception modules designed in GoogleNet is shown in Figure 3. A more detailed explaination along with all relevant parameters of the GoogleNet architecture can be found in the original paper (Szegedy et al., 2015).

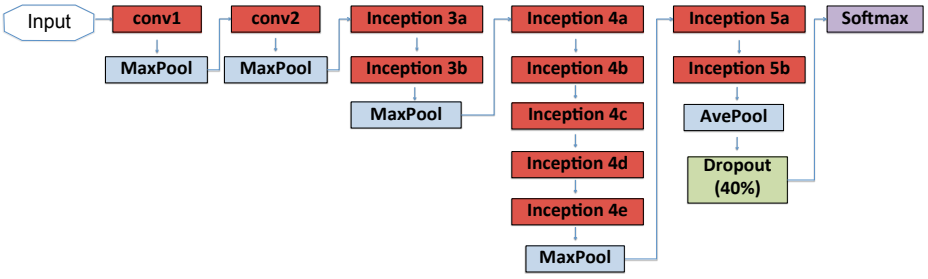


Figure 3: The illustration of the GoogleNet architecture (Szegedy et al., 2015). All convolutional layers and inception modules have a depth of two.

2.3 Classical Local Descriptors

2.3.1 Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) was initially introduced for human detection (Dalal and Triggs, 2005). The HOG feature extractor represents objects by counting occurrences of gradient intensities and orientations in localized portions of an image. Based on the work of (Bertozzi et al., 2007; Surinta et al., 2015), the HOG descriptor computes feature vectors using the following steps:

1. Split the image into small blocks of $n \times n$ cells.
2. Compute horizontal gradient H_x and vertical gradient H_y of the cells by applying the kernel [-1,0,1] as gradient detector.
3. Compute the magnitude M and the orientation θ of the gradient as:

$$M_{(x,y)} = \sqrt{H_x^2 + H_y^2} \tag{1}$$

$$\theta_{(x,y)} = arctan\frac{H_y}{H_x} \tag{2}$$

4. Form the histogram by weighing the gradient orientations of each cell into a specific orientation bin.
5. Apply L2 normalization to the bins to reduce the illumination variability and obtain the final feature vectors.

In our preliminary experiments, we use 5×5 rectangular blocks and 8 orientation bins, thus yielding a 200-dimensional feature vector. We then feed the feature vector to the KNN classifier.

2.3.2 Bags of Visual Words with Histogram of Oriented Gradients

The idea of the bag of visual words (BOW) model (Csurka et al., 2004; Tsai, 2012) in computer vision is to consider an image consisting of different visual words. The image descriptor can be obtained by clustering features of local regions in the images, which contain rich local information of the images, such as color or texture. Here, we combine BOW with the HOG feature descriptor, resulting in HOG-BOW. The construction of the HOG-BOW feature vectors involves the following steps:

1. To compute patches, the set of local region patches P is automatically extracted from the dataset of images, $P = \{p_1, p_2, ..., p_n\}$, where n is the number of patches. The size of each patch is a square of $w \times w$ pixels. Each patch is computed by using local descriptors, and then used as an input to create a codebook.

2. The codebook C is obtained by applying the K-means clustering algorithm over the extracted feature vectors of each patch based on a number of centroids.

3. Construct the BOW feature by detecting the occurrences in the image of each cluster. Each image is split into four quadrants and we compute the feature activation using sum-pooling (Wang, Wang, and Qiao, 2012).

In our experiments, based on the work of (Surinta et al., 2015), the HOG descriptor is employed as the local descriptor. The number of patches is set to 400,000, the size of each patch is 15 $\times$ 15 pixels, and the number of centroids is set to 600. As the image is split into four quadrants, the HOG-BOW generates 2,400 dimensional feature vectors.

The feature vectors are then fed to the classifiers, for which we use the L2-SVM (Suykens and Vandewalle, 1999) and a Multi-Layer Perceptron (MLP). The process of the HOG-BOW method used in our experiments is illustrated in Figure 4.

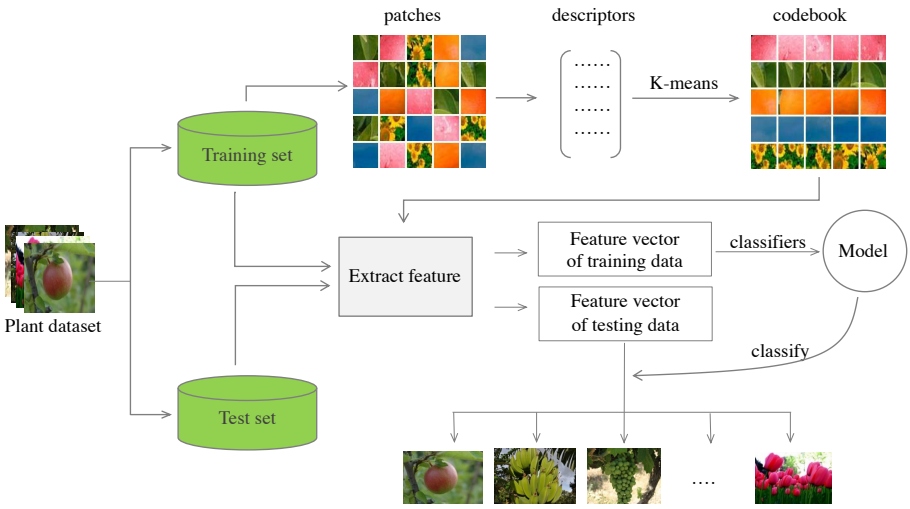


Figure 4: llustration of generating the BOW feature vectors.

2.4 Experiments

2.4.1 Plant Datasets

In our experiments, we performed experiments using three datasets: AgrilPlant, Leafsnap, and Folio.

AgrilPlant: [1] The AgrilPlant dataset consists of 3,000 agriculture images that are collected from the website *www.flickr.com*. It consists of 10 classes with the following plants: apple, banana, grape, jackfruit, orange, papaya, persimmon, pineapple, sunflower, and tulip. Each class contains exactly 300 images. The images may have been taken from five different views, i.e. entire plant, branch, flower, fruit, and leaf. A sample of the AgrilPlant dataset is shown in Figure 5a.

The challenges of classification on the AgrilPlant dataset are (a) the similarity among some classes, i.e. apple, orange and persimmon have

1 The AgrilPlant dataset has been made publicly available and can be accessed at https://www.ai.rug.nl/~p.pawara.

similar shapes and colors, (b) a diversity of plants within the same class, for example, there are green and red apples, or there are varieties of tulips, and (c) the existence of complex backgrounds or other objects such as human, car, and house on several images.

LeafSnap: The Leafsnap dataset (Kumar et al., 2012) originally contained 185 tree species and is used for leaf recognition research. The dataset consists of leaf images taken from two different sources; lab images and field images. In our experiments, we performed experiments with field images. This consists of 7,719 leaf images and has a coverage of 184 tree species (one class is missing for the field images) of the Northeastern United States. All the images were taken in outdoor environments with mobile devices and might contain some amounts of noise, blur, and shadows. The number of images in each class vary from 10 to 183 images. A sample of the LeafSnap dataset is shown in Figure 5b.

Folio: The Folio dataset, introduced in the work of (Munisami et al., 2015), consists of 32 different species of leaves which were collected from the farm at the University of Mauritius. It consists of approximately 20 images for each species. All images were taken under daylight on a white background. A sample of the Folio dataset is shown in Figure 5c.

2.4.2 Experimental Settings

We evaluate the deep CNNs architectures and the hand-crafted local descriptors combined with KNN, SVM, and MLP for plant classification. In our study, the plant datasets are split into a training set and test set with the ratio of 80:20 and 5-fold cross validation is used to evaluate the performance of the studied methods. The resolution of plant images is set to 256×256 pixels.

Most parameters for the deep CNN architectures, for both AlexNet and GoogleNet, are set to the same values for scratch and fine-tuned versions, except for max iteration and step size that are set to different values. The parameters settings are shown in Table 2.

(a) AgrilPlant (b) LeafSnap (c) Folio

Figure 5: Some example images from the three datasets. Note that we show one image per class for some classes in the datasets.

For the hand-crafted local descriptors, we combine the HOG with the KNN classifier and the HOG-BOW with MLP and SVM. We select the optimal k for the KNN classifier in the range of k = {3, 5, 7, 9}.

On each dataset, a grid search is applied to tune the C parameters for the SVM in the range of $C = \{2^1, 2^2, ..., 2^8\}$ and choose the best C parameter that gives the highest accuracy result. We then perform the 5-fold cross validation using this C parameter.

For the MLP, we use the scaled conjugate gradient (Møller, 1993) as a training algorithm. The number of neurons is set to 512 and the learning rate is set to $1.0E^{-}3$. These values resulted in the best performance using preliminary experiments.

Table 2: Summary of experimental parameters for the AlexNet and GoogleNet architectures on the three datasets.

Parameters	AgrilPlant	LeafSnap	Folio
Learning rate	$1.0E^{-3}$	$1.0E^{-3}$	$1.0E^{-3}$
Weight decay	$5.0E^{-4}$	$5.0E^{-4}$	$5.0E^{-4}$
Train batch size	20	20	20
Validation batch size	10	10	10
Max iteration (scratch)	50,000	50,000	50,000
Step size (scratch)	25,000	25,000	25,000
Max iteration (fine-tuned)	20,000	20,000	20,000
Step size (fine-tuned)	10,000	10,000	10,000
Test iterations of solver	30	77	6
Test iterations evaluation	60	154	12

2.5 Results and Discussion

We now report the test accuracies using the deep CNN methods and hand-crafted local feature descriptors with different classifiers. The experiments are carried out based on 5-fold cross validation and we report the top-1 accuracy. The results are shown in Table 3.

2.5.1 AgrilPlant Dataset Evaluation

Comparing the performance of the deep CNN methods and the hand-crafted local feature descriptors, the deep CNN methods consistently outperform the local descriptors. The fine-tuned approaches of both the GoogleNet and the AlexNet architectures obtain the best performance, reaching an accuracy of 98.33% and 96.37%, respectively. This is an improvement of approximately 5% and 6.8% over the scratch versions of each architecture. The GoogleNet fine-tuned version gives approximately 19% better performance than the HOG-BOW with SVM, which obtains

Table 3: Test Accuracy comparison among all techniques on three plant datasets.

Methods	AgrilPlant	LeafSnap	Folio
HOG with KNN	38.13 ± 0.53	58.51 ± 2.47	84.30 ± 1.62
HOG-BOW with MLP	74.63 ± 2.16	79.27 ± 3.36	92.37 ± 1.78
HOG-BOW with SVM	79.43 ± 1.68	72.63 ± 0.38	92.78 ± 2.17
AlexNet scratch	89.53 ± 0.61	76.67 ± 0.56	84.83 ± 2.85
AlexNet fine-tuned	96.37 ± 0.83	89.51 ± 0.75	$\mathbf{97.67 \pm 1.60}$
GoogleNet scratch	93.33 ± 1.24	89.62 ± 0.50	89.75 ± 1.74
GoogleNet fine-tuned	$\mathbf{98.33 \pm 0.51}$	$\mathbf{97.66 \pm 0.34}$	$\mathbf{97.63 \pm 1.84}$

the best performance among the local feature descriptors. The HOG-BOW with SVM outperforms the HOG-BOW with MLP with 4.8% difference. The HOG with KNN obtains the worst performance with an accuracy of 38.13%.

2.5.2 LeafSnap Dataset Evaluation

For the LeafSnap dataset, the GoogleNet fine-tuned and scratch versions obtain the best performance with an accuracy of 97.66%, and 89.62%, respectively. The AlexNet fine-tuned architecture follows up with an accuracy of 89.51%. The HOG-BOW with MLP, however, slightly outperforms the AlexNet scratch architecture with an accuracy of 79.27%. Comparing this to previous work on the LeafSnap dataset using curvature histograms, (Kumar et al., 2012) reported a top-5 accuracy of 96.8%. We note that GoogleNet fine-tuned significantly outperforms that method with a top-1 accuracy of 97.66%. Comparing between the local feature descriptors, The HOG-BOW with MLP gives an accuracy of approximately 6.6% and 20.7% higher than the HOG-BOW with SVM and the HOG with KNN, respectively.

2.5.3 Folio Dataset Evaluation

The work of (Munisami et al., 2015) reported an accuracy of 87.3% by using shape features and a color histogram with KNN which outperforms the AlexNet scratch version on our study with an accuracy of 84.83In our experiments, the AlexNet fine-tuned and the GoogleNet fine-tuned architectures obtain the best results with an accuracy of 97.67% and 97.63%, respectively. The next two techniques with the best performance are the HOG-BOW with SVM and the HOG-BOW with MLP classifiers, both of which yield an accuracy of 92.73% and 92.37%, respectively. The scratch version of GoogleNet still obtains acceptable results with an accuracy of 89.75%. Note that on this dataset, the HOG-BOW with either SVM and MLP classifiers gives roughly 8% better performance than the AlexNet scratch version. The HOG with KNN gives the worst result with an accuracy of 84.30%. The evaluation on the Folio dataset shows that the deep CNN architectures also perform well on a small dataset as this dataset contain only 637 images in total for 32 classes.

2.6 Conclusions

In this chapter, we have presented a comparative study of some classical feature descriptors to deep CNN approaches on three plant datasets. The HOG feature descriptor combined with KNN, and HOG-BOW combined with SVM and MLP classifiers are compared to AlexNet and GoogleNet, both trained from scratch and using the fine-tuned versions as deep CNN architectures.

We evaluated all the image recognition techniques on three plant datasets and achieved notable overall performances. The fine-tuned versions of the deep CNNs architectures persistently outperform the classical feature descriptors techniques on all datasets. The GoogleNet fine-tuned architecture obtains the best result with accuracies of 98.33% and 97.66% on the AgrilPlant dataset and the LeafSnap dataset, respectively. The AlexNet fine-tuned and the GoogleNet fine-tuned techniques also give

the best result on a relatively small dataset, Folio, with an accuracy of approximately 97.6%.

Comparing between the HOG-BOW descriptors on each of the three dataset, on the AgrilPlant dataset, the HOG-BOW combined with SVM performs 4.8% better than the HOG-BOW combined with MLP. On the other hand, the HOG-BOW combined with MLP works 6.64% better than the HOG-BOW combined with SVM. On the Folio dataset, however, both HOG-BOW descriptors give insignificantly different results with an accuracy of approximately 92%. Among all studied techniques, the HOG with KNN always yields the worst accuracy on all datasets.

In further work, we want to study the deployment of deep learning in an unmanned aerial vehicle system targeted for precision identification of plant diseases.

3 DATA AUGMENTATION FOR PLANT CLASSIFICATION

Data augmentation plays a crucial role in increasing the number of training images, which often aids to improve classification performances of deep learning techniques for computer vision problems. In this chapter, we employ the deep learning framework and determine the effects of several data-augmentation (DA) techniques for plant classification problems. We use two convolutional neural network (CNN) architectures, AlexNet, and GoogleNet trained from scratch or using pre-trained weights. These CNN models are then trained and tested on both original and data-augmented image datasets for three plant classification problems: Folio, AgrilPlant, and the Swedish leaf dataset. We evaluate the utility of six individual DA techniques (rotation, blur, contrast, scaling, illumination, and projective transformation) and several combinations of these techniques, resulting in a total of 12 data-augmentation methods. The results show that the CNN methods with particular data-augmented datasets yield the highest accuracies, which also surpass previous results on the three datasets. Furthermore, the CNN models trained from scratch profit a lot from data augmentation, whereas the fine-tuned CNN models do not profit from data augmentation. Finally, we observed that data-augmentation using combinations of rotation and different illuminations or different contrasts helped most for getting high performances with the scratch CNN models.

This chapter was published in:

Pawara, P.,Okafor, E., Schomaker, L.R.B., and Wiering, M.A. (2017). *Data augmentation for plant classification*. International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), pages 615-626.

3.1 Introduction

Plant classification using machine learning and computer vision algorithms is concerned with categorizing plant images into identifiable groups. This may help people to know, for example, the name of a tree they encounter based on a picture from a leaf of the tree. The classification problem can be challenging because of issues related to a high inter-class similarity, intra-class diversities, possible variations of complex backgrounds, and color and illumination variations within the image dataset. Previous studies have employed several supervised learning algorithms combined with hand-crafted features (Hsiao et al., 2014; Kumar et al., 2012; Nilsback and Zisserman, 2008; Wang et al., 2011) and global features (Bama et al., 2011) for investigating plant identification. An extension of the hand-crafted features' use is the combination of geometric-based features with a probabilistic neural network for classifying different classes of the Foliage dataset (Kadir et al., 2011). The recent advances in deep learning (Guo et al., 2016) have led to some big successes in several plant recognition studies (Dyrmann, Karstoft, and Midtiby, 2016; Ghazi, Yanikoglu, and Aptoula, 2017; Mohanty, Hughes, and Salathé, 2016). The authors in (Mohanty, Hughes, and Salathé, 2016) have investigated the use of the famous CNN architectures AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) and GoogleNet (Szegedy et al., 2015) for plant classification. Moreover, the research in (Ghazi, Yanikoglu, and Aptoula, 2017) considered the previous architectures and VGGNet (Simonyan and Zisserman, 2014) in their plant classification task. Generally, CNN architectures consist of many layers and have millions of parameters in the network (LeCun, Bengio, and Hinton, 2015). Therefore, they need large datasets during the learning process.

Several works (Ghazi, Yanikoglu, and Aptoula, 2017; McFee, Humphrey, and Bello, 2015; Salamon and Bello, 2017) have shown that increasing the number of images in training set with data-augmentation (DA) techniques is useful to reduce overfitting and improve the overall performance of the CNN models. The fundamental idea is that the object of interest in an image will not change its class if the image is somewhat changed

using a particular image-processing operation. Data augmentation can be performed in many ways, e.g., using translation, rotation, change in illumination, and color casting and processed in two stages: off-line and online (Sato, Nishimura, and Yokoi, 2015). Off-line augmentation involves an increase in the number of training images before the training starts, while the online stage increases the number of image appearances during the training process. The authors in (Lee et al., 2016) performed off-line augmentation by rescaling the training images into three different sizes, cropping them into smaller-sized images, and combining this with horizontal flips for creating the augmented images during training. The leaf classification system in (Sladojevic et al., 2016) employed three data-augmentation techniques: affine and perspective transformation, and rotation during the training stage. However, there has been little research to investigate the effects of many different single and combined data-augmentation methods, combining different pose and illumination variants, in order to determine if this helps the CNNs obtain significantly better performances.

Contributions: In this chapter, we examine the effects of different data-augmentation techniques using two off-the-shelf CNN techniques: AlexNet and GoogleNet, which we train from scratch or using pre-trained weights. We use three different image datasets of plants, and we evaluate the CNNs on the original datasets, the datasets obtained using a single DA technique, and the datasets obtained using several combinations of DA techniques. Note that the DA techniques are only applied to the training data. Therefore this results in 12 training set variants for the three plant recognition datasets. The results show that when the CNN methods are trained from scratch, the use of DA techniques helps obtain higher performances. Especially combinations of the rotation and illumination DA techniques or rotation and contrast are most useful for the considered datasets. For the fine-tuned CNN models, the gains of DA techniques are much smaller, although they helped to get the best results, which are also better than previous results on the three plant datasets.

Outline: Section 3.2 covers details of the three plant datasets used in this study and the different data-augmentation techniques. The CNN methods and experimental settings are described in Section 3.3. The results are shown and discussed in Section 3.4. Finally, we draw a conclusion and recommend future work in Section 3.5.

3.2 Datasets and Data Augmentation

In this section, we describe the three plant datasets and the data augmentation techniques used in the experiments. In Figure 6, we show some examples of images within the datasets.

3.2.1 Datasets

The Folio dataset (Munisami et al., 2015), a relatively small dataset, consists of 637 leaf images from 32 species. Each class contains approximately 20 images (three images are missing from the initial work of (Munisami et al., 2015)). All images were taken under daylight on a plain background. The first classification system for this dataset used shape features, and a color histogram with a k-nearest neighbor classifier (Munisami et al., 2015) and reported an accuracy of 87.3%. The most recent study in (Pawara et al., 2017b) employed CNN techniques applied to the original images. The best CNN architecture obtained a high accuracy of 97.7%. We used the same train/validation/test splits as in (Pawara et al., 2017b) with a ratio of 70:10:20.

AgrilPlant: The AgrilPlant dataset was presented in (Pawara et al., 2017b), and it consists of 3,000 plant images from 10 classes: apple, banana, grape, jackfruit, orange, papaya, persimmon, pineapple, sunflower, and tulip. Each class consists of 300 images. The AgrilPlant dataset faces some challenges due to the following reasons: 1) a dissimilarity of plants within the same class, for example, there are varieties of shape and color of tulips, or there are several colors of apples, 2) a similarity among some

classes, for example, apple, orange, and persimmon images consist of similar shapes and colors, and 3) the complex backgrounds in most of the images. We adopted the same dataset splits as previously used in (Pawara et al., 2017b) with a ratio of 70:10:20 for the train, validation, and testing sets, respectively.

Swedish: The Swedish dataset (Söderkvist, 2001) contains 1,125 plant leaf images on a plain background of 15 different Swedish tree species, with 75 images per class. The earlier research in (Söderkvist, 2001) combined simple features such as moments, area and curvature and reported an accuracy of 82%. To the best of our knowledge, the study in (VijayaLakshmi and Mohan, 2016) yielded the highest accuracy of 99.5%. This was achieved by combining shape, color, and Haralick features.

The authors in (Atabay, 2016) proposed CNN methods with horizontal flip augmentation on this dataset and obtained an accuracy of 99.1%. The challenge of classification on the Swedish dataset (Mouine, Yahiaoui, and Verroust-Blondet, 2013; Wang, Liang, and Guo, 2014; Zhang et al., 2016) is its high inter-species similarity among several classes. Our study used the same dataset splits as in (Söderkvist, 2001) with randomly selecting 25 images per class for training and the rest for testing. Additionally, the training images were further dissected in the ratio 1:4 for validation and training sets, respectively.

3.2.2 Data Augmentation

In this subsection, we describe the six different data-augmentation techniques examined to increase the number of images within the training set for each of the datasets discussed in the previous subsection. The data-augmentation techniques we studied in this chapter are:

Rotation: Our preliminary experiments were done on the AgrilPlant dataset. Using different rotational angles that exist between 8°and 90°, we observed that using a tilt of an image with angle 30°obtained good performances. This is the reason for the choice of using random image

Figure 6: Some example images from the three datasets in which we show one image per class for some classes in the datasets. From the top row to the bottom row we can see example images from the Folio, AgrilPlant, and Swedish datasets.

rotations with a rotation angle in [-30°, 30°], with empty space padded with white pixels.

Blur: The goal of the blur augmentation is to de-emphasize differences in adjacent pixel values. In this chapter, the 2D Gaussian smoothing kernel is used. The kernel size is set to $2 \times (\lceil 2\sigma \rceil) + 1$, where $\lceil . \rceil$ is a ceiling function, and σ is the standard deviation of the Gaussian distribution which is randomly set between 2 and 8.

Scaling: The training images are rescaled to larger ones with a random factor between 2 and 8 times. Hence, when feeding the images into the CNNs, we crop the images from the up-scaled images, and this corresponds to a small subpart of the image, which may contain important features of the plants.

Contrast: We first convert images from an RGB color map to an HSV color map, then multiply the S and V components of the images by a random factor between 0.8 and 2. Finally, the images are converted back to the RGB color representation.

Illumination: The training images are adjusted by adding random values between 10 and 80 to the R, G and B channels.

Projective: The projective transformation changes the projective viewpoint of the observer. After transformation, straight lines still remain straight (Sladojevic et al., 2016) but it does not preserve parallelism, length, and angle. The projective transformation requires a 3×3 transformation matrix[1].

$$(x_j, y_j, 1) = (x_i, y_i, 1) \times \begin{pmatrix} \cos(\theta) & \sin(\theta) & t_1 \\ \sin(\theta) & \cos(\theta) & t_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

where $(x_i, y_i, 1)$ represents the coordinate before the projective transformation, $(x_j, y_j, 1)$ denotes the coordinate after the transformation, θ is the rotation angle of the image, and $[t_1 \; t_2]^T$ is the projection vector which is set to $[0.001 \;\; 0.001]^T$. The angle θ is randomly chosen from the interval $[1, 30]$.

The effects of all DA techniques on some example images of the AgrilPlant dataset are shown in Figure 7. In addition to the use of these single DA methods, we also consider several combinations of the earlier discussed methods to obtain more training images. Because testing all combinations is almost infeasible, we tested only combinations in which the rotation operator is part of the combined DA technique. This results in six possible combinations of DA methods , including rotation + blur, rotation + contrast, rotation + scaling, rotation + illumination, rotation + projective, and rotation + contrast + illumination. Each single data-augmentation method adds eight adapted copies of the original images while the combination of two DA methods results in 16 different copies of the images. Lastly, the combination of three DA methods yields 24 times more training images. The total number of images present in each of the original and the DA image datasets are summarized in Table 4.

1 https://www.graphicsmill.com/docs/gm5/Transformations.htm

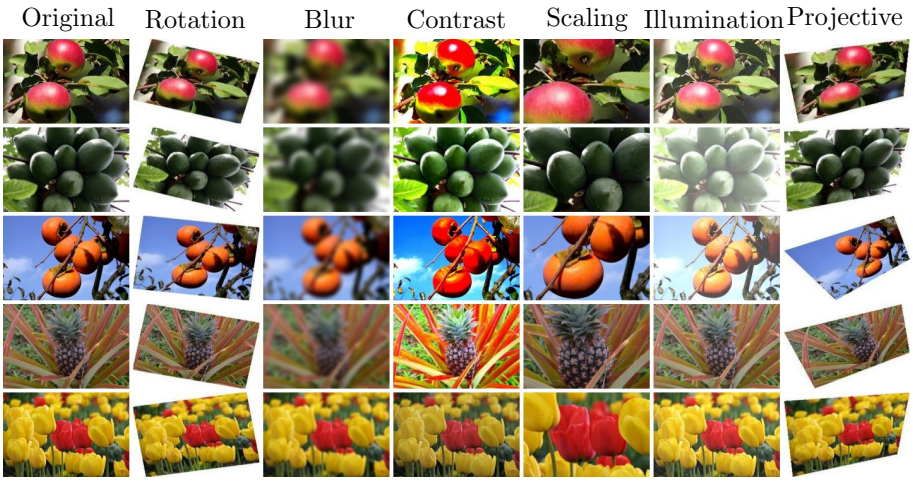


Figure 7: Effects of data augmentation on some example images of the AgrilPlant dataset.

3.3 Deep Learning Architectures

3.3.1 CNN Methods

In our study, we employ two CNN architectures: AlexNet and GoogleNet for evaluating both original and several variants of data-augmented image datasets for the three plant recognition tasks.

AlexNet: The CNN architecture AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) outperformed other computer vision methods during the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. The network consists of five convolutional layers, three max pooling layers, two dropout layers, and three fully-connected layers ending with a SoftMax classification layer. It uses the Rectified Linear Unit (ReLU) for the non-linear activation functions. In our study, we employed a customised version of AlexNet as proposed in (Pawara et al., 2017b), in which we reduced the number of hidden units in the last fully-connected layers to 1024 neurons. We also consider two instances of the AlexNet architecture: using randomly initialized weights (scratch) and using pre-trained weights (fine-

Table 4: Summary of the number of training images in the data-augmented datasets.

DA sets	Folio	AgrilPlant	Swedish
Original	445	2100	300
Individual DA	4,005	18,900	2,700
Combination of two DAs	7,565	35,700	5,100
Combination of three DAs	11,125	52,500	7,500

tuned). In the fine-tuned network, the pre-trained weights from ImageNet were used, after which we trained the whole architecture based on the errors for classifying the training images from the plant datasets.

GoogleNet: GoogleNet (Szegedy et al., 2015) is a deeper network, but has a much lower number of parameters (4 million parameters) compared to AlexNet (60 million parameters). This is a consequence of the inception module that vastly decreases the amount of trainable parameters in the network. More specifically, GoogleNet uses nine inception modules, four convolutional layers, four max-pooling layers, three average pooling layers, five fully-connected layers and three SoftMax layers for the main and auxiliary classifiers in the network. Inspired by the network-in-network approach (Lin, Chen, and Yan, 2013), the inception module uses a parallel combination of 1×1, 3×3, and 5×5 convolutions along with a pooling layer. A more detailed explanation and all relevant parameters of the GoogleNet architecture can be found in the original paper (Szegedy et al., 2015). Similarly as with AlexNet, we evaluated both scratch and fine-tuned versions of the GoogleNet architecture.

3.3.2 Experimental Setup

We evaluate the deep CNN architectures with the different data-augmentation schemes for the three plant classification tasks. In the experiments, we employed 5-fold cross validation to evaluate the performances

of the different methods. The resolution of the images is set to 256×256 pixels.

The AlexNet and GoogleNet hyper-parameters are set as follows: number of iterations: 20,000 for fine-tuned and 50,000 for the scratch version, step size: 10,000 and 25,000 for fine-tuned and scratch, respectively, train batch size: 20, validation batch size: 10, base learning: 0.001, momentum: 0.9, weight decay: 0.0005, and test interval: 10,000. Each dataset contains a different number of images, therefore we set different batch sizes for the different datasets as 7, 30 and 8 for Folio, AgrilPlant, and Swedish, respectively.

To summarize, we performed a total of 52 experiments on each dataset, which vary in the following settings: two choices of deep learning architecture (AlexNet and GoogleNet), two choices of training mechanism (fine-tuned or scratch), using the set of original images, and 12 datasets constructed with different data-augmentation techniques (as described in Section 3.2.2).

3.4 Results

In this section, we report the test accuracies using the deep learning methods on the original and augmented datasets for the different plant recognition tasks. We report the top-1 accuracy and average the results over the five folds.

3.4.1 Folio Dataset Evaluation

Table 5 shows the plant classification accuracies with different DA techniques on the Folio dataset using AlexNet and GoogleNet with both scratch and fine-tuned models. The scratch AlexNet always profits from the different DA techniques on this dataset, whereas scratch GoogleNet also profits from most DA techniques, but in a lesser degree. Scratch AlexNet profits most from the combined effects of rotation and illumination, or combined effects of rotation, contrast, and illumination which led to a performance

Table 5: Recognition results (accuracy and standard deviation) using different DA schemes for the Folio dataset.

Augmentation methods	AlexNet		GoogleNet	
	Scratch	Fine-tuned	Scratch	Fine-tuned
Original (no flip)	84.83 ± 2.85	97.67 ± 1.60	89.75 ± 1.74	97.63 ± 1.84
Original (flip)	87.50 ± 2.62	98.85 ± 0.44	93.46 ± 1.83	98.85 ± 0.77
(a) Rotation	92.69 ± 2.22	98.27 ± 0.38	93.08 ± 0.63	**99.04 ± 0.38**
(b) Blur	88.65 ± 1.31	98.65 ± 0.74	93.59 ± 1.94	98.85 ± 0.99
(c) Contrast	92.69 ± 0.44	**99.04 ± 0.38**	93.65 ± 0.74	98.65 ± 0.74
(d) Scaling	89.81 ± 0.74	**99.04 ± 0.97**	**95.00 ± 0.44**	98.65 ± 0.74
(e) Illumination	93.46 ± 2.84	98.46 ± 0.63	94.23 ± 0.99	**99.42 ± 0.38**
(f) Projective	93.08 ± 0.63	98.65 ± 0.74	93.65 ± 0.97	98.27 ± 1.31
(a) + (b)	92.50 ± 1.15	98.27 ± 0.38	93.27 ± 0.97	98.65 ± 1.15
(a) + (c)	95.00 ± 0.99	**99.04 ± 0.94**	94.81 ± 1.15	98.46 ± 0.89
(a) + (d)	92.69 ± 1.33	98.46 ± 0.63	93.65 ± 0.74	98.85 ± 1.33
(a) + (e)	**96.35 ± 0.74**	98.65 ± 1.31	94.42 ± 0.74	98.85 ± 1.33
(a) + (f)	92.69 ± 0.77	97.50 ± 0.97	93.65 ± 1.31	98.65 ± 0.74
(a) + (c) + (e)	**96.35 ± 0.97**	98.46 ± 0.63	94.23 ± 1.60	98.65 ± 0.74

improvement of around 8.8% compared to using the original images. The best single DA technique for scratch AlexNet is the illumination operator, and blur is the DA technique that helps the least in getting higher performances. For scratch GoogleNet the best DA technique uses the scaling operation and this leads to 1.5% accuracy improvement compared to training on the original images. For the fine-tuned architectures, GoogleNet with the illumination DA technique obtains the highest accuracy. Because the fine-tuned models already perform very well with the original dataset, the improvements are much smaller in this case than when using the scratch CNN architectures.

When we compare our approaches to previous CNN experiments in (Pawara et al., 2017b), which did not consider flipping of the images, these new results show a significant improvement in the recognition performance. This shows that the effect of flipping is also very important for this dataset and that the offline DA techniques can help to obtain even higher performances.

Table 6: Recognition results using different DA schemes for the AgrilPlant dataset.

Augmentation methods	AlexNet		GoogleNet	
	Scratch	Fine-tuned	Scratch	Fine-tuned
Original	89.53 ± 0.61	96.37 ± 0.83	93.33 ± 1.24	98.33 ± 0.51
(a) Rotation	90.10 ± 1.08	96.90 ± 0.69	92.53 ± 1.49	98.17 ± 0.68
(b) Blur	82.97 ± 2.26	94.43 ± 1.33	87.80 ± 1.27	97.73 ± 0.95
(c) Contrast	89.53 ± 1.26	96.27 ± 1.15	94.10 ± 0.95	98.17 ± 0.63
(d) Scaling	90.20 ± 0.95	96.93 ± 0.93	94.00 ± 1.20	98.13 ± 0.62
(e) Illumination	90.13 ± 1.06	**97.27 ± 0.38**	**95.03 ± 1.11**	98.21 ± 0.89
(f) Projective	90.87 ± 1.14	96.20 ± 0.92	93.21 ± 1.04	98.21 ± 0.76
(a) + (b)	87.70 ± 1.25	96.23 ± 0.71	90.40 ± 1.87	98.27 ± 0.62
(a) + (c)	**91.57 ± 0.96**	**97.10 ± 0.43**	**95.17 ± 1.38**	**98.60 ± 0.38**
(a) + (d)	90.40 ± 1.12	96.50 ± 0.31	92.93 ± 1.89	98.10 ± 0.82
(a) + (e)	**91.07 ± 0.49**	**97.03 ± 0.49**	94.07 ± 1.46	98.43 ± 0.60
(a) + (f)	90.50 ± 0.63	96.77 ± 0.95	92.77 ± 1.38	98.13 ± 0.92
(a) + (c) + (e)	**91.53 ± 0.78**	96.77 ± 0.71	94.73 ± 0.69	98.53 ± 0.59

3.4.2 AgrilPlant Dataset Evaluation

For the AgrilPlant dataset we also used the two CNN architectures trained from scratch or fine-tuned and evaluate them on both original and data-augmented datasets. The results are shown in Table 6. We observe that the fine-tuned GoogleNet with the combined effect of rotation and contrast yields the highest classification accuracy of 98.6%. The fine-tuned AlexNet profits most from the illumination DA technique. The performance improvements using DA on this dataset are much smaller than for the previous dataset. The reason is that there are 210 training images per class in this dataset, whereas there are only 14 training images per class in the Folio dataset. Still, for scratch AlexNet the combined DA techniques rotation+contrast and rotation+contrast+illumination result in a performance improvement of 2% compared to training from the original dataset. We also note that all CNN architectures with the blur DA technique obtain lower performances than using the original images. The reason is most probably that blurred images reduce the amount of salient features in the images from this dataset, which are still present in the test images.

Table 7: Average accuracies and standard deviation using different DA techniques on the Swedish dataset.

Augmentation methods	AlexNet		GoogleNet	
	Scratch	Fine-tuned	Scratch	Fine-tuned
Original	94.69 ± 1.18	99.65 ± 0.07	96.08 ± 1.10	99.81 ± 0.15
(a) Rotation	96.21 ± 0.80	99.52 ± 0.29	97.04 ± 0.66	99.87 ± 0.13
(b) Blur	94.75 ± 0.97	99.36 ± 0.22	96.27 ± 1.40	99.57 ± 0.58
(c) Contrast	95.09 ± 0.67	99.55 ± 0.37	96.69 ± 0.91	99.79 ± 0.18
(d) Scaling	94.88 ± 0.78	99.60 ± 0.19	96.53 ± 1.25	99.84 ± 0.15
(e) Illumination	95.23 ± 0.53	99.49 ± 0.42	96.05 ± 0.91	99.76 ± 0.17
(f) Projective	96.88 ± 0.24	99.41 ± 0.07	96.64 ± 0.65	99.73 ± 0.13
(a) + (b)	96.40 ± 1.20	99.49 ± 0.17	97.12 ± 0.33	99.81 ± 0.15
(a) + (c)	97.07 ± 0.52	99.41 ± 0.15	**98.24 ± 0.52**	99.84 ± 0.11
(a) + (d)	96.40 ± 0.72	99.65 ± 0.22	97.68 ± 0.37	**99.92 ± 0.07**
(a) + (e)	97.25 ± 0.41	99.65 ± 0.28	98.16 ± 0.57	99.81 ± 0.22
(a) + (f)	**97.81 ± 0.77**	99.41 ± 0.07	97.55 ± 0.92	99.81 ± 0.07
(a) + (c) + (e)	97.60 ± 0.57	**99.76 ± 0.20**	97.68 ± 0.77	99.73 ± 0.16

3.4.3 Swedish Dataset Evaluation

The plant classification accuracies with different DA schemes on the Swedish dataset are reported in Table 7. The results show that the scratch CNN architectures profit from almost all DA methods. The biggest performance improvement is for scratch AlexNet where the use of the combined rotation+projective DA technique leads to a performance improvement of 3.1%. For this dataset, the fine-tuned CNN models do not profit from the DA techniques, and often the results using a DA technique are even a bit lower than using the original dataset. The fine-tuned AlexNet obtained the best performance with the combined rotation+contrast+illumination DA technique and obtained an accuracy of 99.76%, while the fine-tuned GoogleNet worked best with the combined rotation+scaling DA method with an accuracy of 99.92%. Both these fine-tuned versions outperformed the previous study in (VijayaLakshmi and Mohan, 2016) which combined shape, color, and Haralick texture features and reported an accuracy of 99.5%.

3.4.4 Discussion

We have performed experiments on 3 datasets with 52 different techniques. If we look at the combined results, we can derive the following conclusions:

- The scratch version of AlexNet profits most from data augmentation. The reason is probably that it consists of most parameters to train and therefore larger datasets are very helpful.
- The fine-tuned CNN models hardly profit from data augmentation for the considered datasets. One reason is that the performances of the fine-tuned CNN methods are already very good, so there is not much room for improvement. Still, scaling helps the fine-tuned AlexNet with 0.3% average accuracy improvement and the illumination DA technique helps the fine-tuned GoogleNet a bit with an average accuracy improvement of 0.2%.
- For scratch AlexNet particular combined DA techniques lead to the biggest performance improvements. The average improvement for the three datasets using the combined DA technique rotation + contrast + illumination is 4.6%. This is followed by the combination of rotation and illumination with an average gain of 4.3%.
- For scratch AlexNet, the best single DA technique uses the projective transformation, which helps to improve the average accuracies with 3.0%.
- The scratch GoogleNet also profits most from combined DA techniques, where the combination of rotation and contrast helps to get 1.8% higher average accuracy.
- For scratch GoogleNet, the best single DA technique uses the scaling operator, which helps to improve the average accuracies with 0.9%.

3.5 Conclusion

We have investigated the usefulness of 6 different data-augmentation techniques and combinations of them using two well-known CNN architectures on three plant datasets. The results show that data-augmentation methods are important to obtain higher accuracies for CNN models trained from scratch. This shows that more training data helps a lot, which is also because some of the datasets do not contain many original training images. For the scratch AlexNet and GoogleNet architectures, especially the combined effects of rotation and illumination or rotation and contrast are very helpful. The blur operation does not help to obtain higher accuracies and sometimes even results in worse performances, despite the increase in the amount of training images. The fine-tuned AlexNet architecture profits a bit from the scaling DA technique, whereas the fine-tuned GoogleNet profits a bit from the illumination DA technique, but most other DA techniques are not helpful to obtain higher accuracies with the pre-trained CNN architectures. One reason why the fine-tuned CNN models do not really profit from data augmentation, is that they obtain very high performances on the considered datasets when trained on the original datasets. Therefore, there is very little room for improvement. The scratch CNN architectures in general need much more training examples, and therefore profit a lot from the combined DA techniques which increase the number of different training images the most.

In future work, we want to examine the effects of data augmentation on more complex datasets for which the fine-tuned CNN architectures do not perform very well using only the original images. We also want to examine the data augmentation techniques describes in (Okafor et al., 2017), where new images containing multiple version of an original image are constructed. Finally, instead of presetting the boundaries of the effects of the DA techniques, we want to focus on learning the right amounts in which images are changed with particular DA techniques using a novel adversarial learning framework.

4

DEEP LEARNING WITH DATA AUGMENTATION FOR FRUIT COUNTING

Counting the number of fruits in an image is important for orchard management, but is complicated due to different challenging problems such as overlapping fruits and the difficulty of creating large labeled datasets. In this chapter, we propose using a data-augmentation technique that creates novel images by adding a number of manually cropped fruits to original images. This helps to increase the size of a dataset with new images containing more fruits and guarantees correct label information. Furthermore, two different approaches for fruit counting are compared: a holistic regression-based approach, and a detection-based approach. The regression-based approach has the advantage that it only needs as target value the number of fruits in an image compared to the detection-based approach where bounding boxes need to be specified. We combine both approaches with different deep convolutional neural network architectures and object-detection methods. We also introduce a new dataset of 1500 images named the Five-Tropical-Fruits dataset and perform experiments to evaluate the usefulness of augmenting the dataset for the different fruit-counting approaches. The results show that the regression-based approaches profit a lot from the data-augmentation method, whereas the detection-based approaches are not aided by data augmentation. Although one detection-based approach finally still works best, this comes with the cost of much more labeling effort.

This chapter was published in:

Pawara, P., Boshchenko, A., Schomaker, L.R.B., and Wiering, M.A. (2020). *Deep Learning with Data Augmentation for Fruit Counting*. International Conference on Artificial Intelligence and Soft Computing (ICAISC), pages 203-214.

4.1 Introduction

Estimating the number of fruits in orchards is an important task for farming management. Computer vision techniques and convolutional neural networks (CNNs) (LeCun, Bengio, and Hinton, 2015; Schmidhuber, 2015) have been used for the fruit-counting task and achieved very good performances (Häni, Roy, and Isler, 2019; Liu et al., 2018; Rahnemoonfar and Sheppard, 2017). Researchers have worked on two different approaches of counting: regression-based counting (Arteta, Lempitsky, and Zisserman, 2016; Lempitsky and Zisserman, 2010; Stahl, Pintea, and Gemert, 2018) and detection-based counting (Lempitsky and Zisserman, 2010; Paul Cohen et al., 2017). Despite obtaining high accuracies using deep learning for object counting tasks (Boominathan, Kruthiventi, and Babu, 2016; Koirala et al., 2019; Onoro-Rubio and López-Sastre, 2016; Zhang et al., 2015b), fruit counting in images is still challenging due to several reasons (Liu et al., 2018; Rahnemoonfar and Sheppard, 2017), including high variances in illumination, overlapping fruits, fruit occluded by other parts of the tree, different sizes of fruits, and different degrees of ripeness of fruits leading to high variances in colors. In addition to these obstacles, there are few datasets with images of fruits, especially for the fruit-counting task, because making an annotation for the dataset can be very time consuming (Dwibedi, Misra, and Hebert, 2017).

It has been shown that increasing the number of images in the training set by using data-augmentation techniques helps to improve the performance of training CNNs (Perez and Wang, 2017; Taylor and Nitschke, 2017). Data-augmentation methods can be based on different approaches, and generally modify the image properties (Brahimi et al., 2018; Pawara et al., 2017a), such as changing color or contrast, horizontal/vertical flipping, using rotation, scaling, and translation, or synthesizing data (Rahnemoonfar and Sheppard, 2017; Tremblay et al., 2018). Several researchers successfully used generative adversarial networks (GANs) (Goodfellow et al., 2014; Antoniou, Storkey, and Edwards, 2017) or random elastic morphing (Bulacu et al., 2009) for data augmentation. The

augmentation approach described in (Dwibedi, Misra, and Hebert, 2017) combines real and synthetic images by adding collages from real objects to the images.

Contributions: To address the challenges discussed before, we extend previous research in several ways: (1) We introduce a new fruit dataset, namely the Five-Tropic-Fruits (FTF) dataset [1], which can be used for classification, detection, and counting. (2) We propose a fruit data-augmentation (FDA) technique which is useful to increase the number of images and the number of fruits in the images and apply FDA on the training set of the FTF dataset. FDA is helpful to add as many fruits as we want to the training images and can compute exact label information (total number of fruits or bounding boxes). (3) We compare two fruit-counting approaches (regression-based counting, and detection-based counting) on the original and the augmented training set of FTF and evaluate and compare the performances of two CNN architectures. For regression-based counting, the CNN architectures ResNet50 (He et al., 2016), and Inception-V3 (Szegedy et al., 2016) are used. For detection-based counting, two object-detection architectures, Faster R-CNN (Ren et al., 2015) and SSD-MobileNet (Howard et al., 2017), are trained for performing fruit detection and subsequently fruit counting.

Outline: Section 4.2 covers details of the proposed approach of our research. Section 4.3 discusses results of the experiments and Section 4.4 draws conclusions and describes possible future work.

4.2 Proposed Approach

In this section, we explain the details of our proposed approach for the fruit-counting system.

1 The FTF dataset has been made publicly available and can be accessed at https://www.ai.rug.nl/~p.pawara.

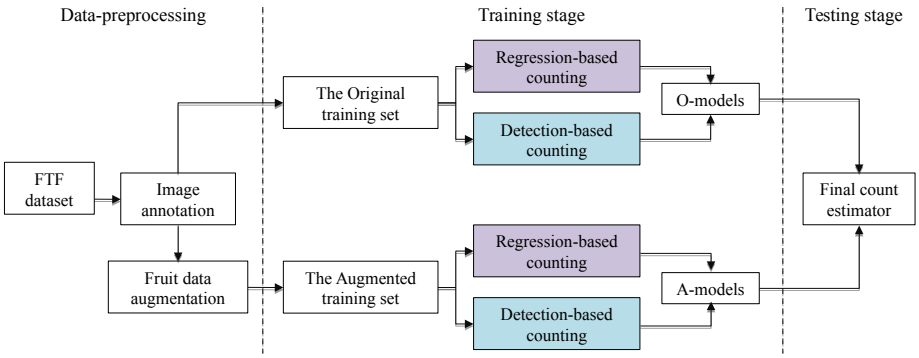


Figure 8: System pipeline

4.2.1 Overall Pipeline

Figure 8 illustrates the overall fruit-counting pipeline. The proposed approach starts with dataset pre-processing, consisting of data collection and data annotation (Section 4.2.2). Then we perform FDA on the original training set (Section 4.2.3), resulting in an augmented set. The original and the augmented training images are used as inputs for the training stage of two fruit-counting approaches (Section 4.2.4): regression-based counting and detection-based counting. Each of the counting approaches CNNs are trained on the original training set (yielding the O-models), and the augmented training set (resulting in the A-models). Finally, we evaluate and compare the performances of the O-models and the A-models in the testing stage. The details of each stage are explained in the following subsections.

4.2.2 Dataset

4.2.2.1 *Data collection*

The Five-Tropic-Fruits (FTF) dataset contains 5 classes of fruit images: apple, custard apple, lemon, pear, and persimmon. All images have been collected from the internet. Each image contains only one type of fruit and can consist of up to 15 fruits. The images can be either an indoor or

outdoor scene and can cover various objects and backgrounds. The fruits in the images may be occluded and have different levels of overlapping. Some fruits can also be truncated around the border of the images. We collected 300 images per class. The dataset is split into a training set of 80% and a test set of 20%.

4.2.2.2 *Image annotations*

We compare two counting approaches: regression-based counting and detection-based counting. These two approaches require different formats of ground truths, which we will explain in more detail in Section 4.2.4. For both approaches, we start by manually annotating each image using LabelImg [2]. Each fruit will be labeled and counted as one fruit even if it is occluded or truncated. Locations (bounding boxes) of fruits in the images and the total counts and class names are kept in XML files. Figure 9 shows some example images from the FTF dataset, their bounding-box annotations and the ground truths (total counts) for the regression-based counting task.

4.2.3 Fruit Data Augmentation (FDA)

In this subsection, we present the fruit data-augmentation method used for the counting task. The objective of FDA is to increase the number of fruits in the training images and the total number of images in the training set. The steps of FDA are:

1. **Create fruit masks:** We manually created 20 object masks of each fruit in a Portable Bit Map (PBM) format with a white object and black background.

2. **Select background images:** Background images are selected from the training set of the FTF dataset. Only images with a number between 1 and 8 fruits will be used as the background images.

2 https://github.com/tzutalin/labelIm

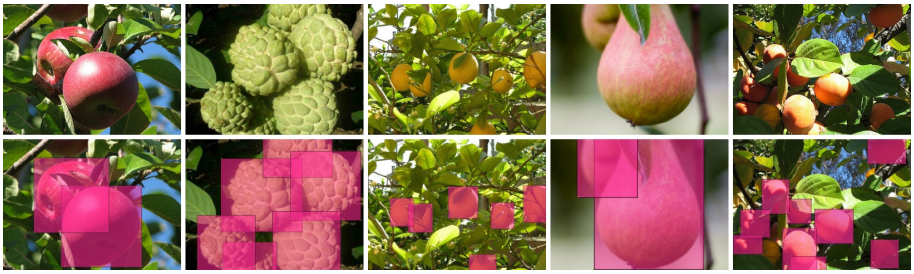


Figure 9: Example images from the FTF dataset. Some images show that some fruits are occluded or truncated. The first row shows example images from the classes: apple, custard apple, lemon, pear, and persimmon, respectively. The second and the third row show the bounding boxes and the labels (total counts) of the images used for the detection-based and regression-based counting approach, respectively.

3. **Augment the fruit masks:** We extracted the masks and perform the following augmentation methods on the masks:
   - **Rotation:** The mask objects are rotated with random angles in $[-30°, 30°]$.
   - **Scaling:** The mask objects are relatively big compared to the background images. So we rescaled the mask images to smaller sizes with a random factor in $[0.4, 0.6]$.
4. **Add fruit masks to the background images:** While placing the masks on the background images, the following cases are considered:
   - **Number of objects to be added:** The specified task is to count up to 15 fruits. As we selected background images that can already contain up to 8 fruits, we set a maximum number of objects to be placed in the background images to 7. Therefore the fruits added are of the same class and their total amount is randomly chosen between 1 and 7.
   - **Occlusion and truncation:** There are two cases of occlusion: the occlusion between the masks, and the occlusion between

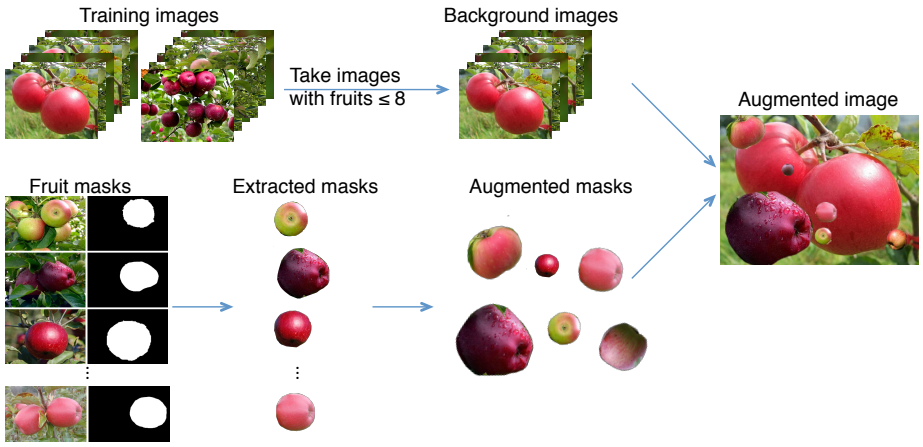


Figure 10: The fruit data-augmentation (FDA) process. The flowchart shows an example of its working on the apple class. In the example, 6 apples were added to the background (original) image.

the masks and the existing fruits in the background images. Both occlusion factors are set to a maximum Intersection-over-Union (IoU) score of 0.5, which means each object can be overlapped up to 50%. A truncation factor is set to 0.1 to ensure that at least 90% of the masks will be placed on the background images. The positions of the existing fruits in the background images can be retrieved from the corresponding XML file of the image.

- **Blending mode:** Our preliminary experiments showed that adding objects on background images with Gaussian smoothing can reduce boundary artifacts and give better performance. Hence, we place the objects on background images using Gaussian Blurring with a Gaussian kernel size set to [5,5] for which the standard deviation of the kernel is set to 2.

Figure 10 shows the overall process of FDA. We used FDA to add around 13,100 images to the original training set of 1200 images.

4.2.4 Deep Learning for Fruit Counting

In this subsection, we explain the deep learning architectures for the regression-based and the detection-based counting approaches.

4.2.4.1 *Regression-based counting*

There are two stages for regression-based counting: the training and testing stage.

1. Training stage: In our work, two state-of-the-art deep learning architectures (ResNet50 (He et al., 2016), and Inception-V3 (Szegedy et al., 2016)) are trained on the original and the augmented training set of the FTF dataset. Both CNN models are trained with two training schemes by using randomly initialized networks (scratch) or networks that were trained on ImageNet (pre-trained). In the usual regression task, a single value is used as a target value. For example, an image with 5 apples is assigned the target value 5. For the proposed regression-based counting task, we predict both the total count and the class of the possible fruits in an image. Hence, instead of using a single value, we construct a target vector with $n = 5$ values. We obtain the ground truth during the training process from the XML files described in Section 4.2.2.2, and compute the target output using the number of fruits and class. For example {5,0,0,0,0} means an image contains 5 apples, while {0,0,0,0,13} means an image contains 13 persimmons. The deep learning architectures are trained by optimizing the mean absolute error (MAE) loss function using stochastic gradient descent (SGD) with momentum. The MAE loss function L_{MAE} for a single training example is defined as:

$$L_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

Given n classes, y_i is the target value for a given class i, and $\hat{y}_i$ is the predicted value estimated by the network (for example, $\{2, 0, 3, 0, 0\}$ if the network predicted that an image has 2 apples and 3 lemons). Note that in the experiments we do not round off the continuous predicted values.

There are two layers at the end of the networks. The first fully-connected layer uses a Rectified Linear Unit (ReLU) activation function with 1024 hidden units. The final layer uses linear activation functions and n output units to compute the predicted values.

For regression-based counting, there are 8 training configurations (2 CNN architectures $\times$ 2 (pre-trained or scratch) $\times$ 2 training sets). All the configurations use the same hyper-parameters values (momentum 0.9, learning rate 0.0001, batch size 64, and 1,000 epochs), which were selected after some preliminary experiments. The models trained with the original training images are called the O-models, whereas the models trained with the augmented training images are called the A-models.

2. Evaluation metrics for counting (testing stage): We evaluate the performance of the CNN models by comparing the mean absolute value (MAE) obtained from the models trained on the original set (the O-models) and the models trained on the augmented set (the A-models). As our tasks consider both counting and classification, we report three loss values for different purposes; MAE for counting and classification purposes (MAE_{CC}), MAE for the count-only purpose (MAE_{CO}), and MAE for evaluating misclassifications (MAE_{MC}). Each of these three loss functions is explained below.

MAE_{CC}: Given n classes and m test instances, MAE_{CC} for counting and classification is calculated as:

$$MAE_{CC} = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{k=1}^{m} |y_i^k - \hat{y}_i^k| \tag{5}$$

Where y_i^k is the number of fruits in the k-th test image of class i and 0 otherwise, and $\hat{y}_i^k$ denotes the predicted values for the corresponding image.

MAE_{CO}: We also want to evaluate the models by only considering the number of fruits of the correct class (without considering the other wrongly detected fruits). MAE_{CO} is calculated as:

$$MAE_{CO} = \frac{1}{m}\sum_{k=1}^{m}|y_c^k - \hat{y}_c^k| \tag{6}$$

Where c is the class of the k-th image in the test set.

MAE_{MC}: To evaluate the performance of the model when considering only misclassified fruits, MAE_{MC} is calculated as:

$$MAE_{MC} = \frac{1}{(n-1)\cdot m}\sum_{\substack{i=1,\\ i\neq c}}^{n}\sum_{k=1}^{m}|y_i^k - \hat{y}_i^k| \tag{7}$$

Table 8 shows some examples of possible image content, the corresponding target and predicted counts and the values for the three loss functions.

Table 8: Examples of the target and predicted outputs and the values of the three loss functions for a single example in the testing stage.

Images	Count and classification			Count only			Misclassification		
	target	predicted	MAE_{CC}	target	predicted	MAE_{CO}	target	predicted	MAE_{MC}
5 apples	{5,0,0,0,0}	{0,0,0,5,0}	2.00	{5}	{0}	5.00	{0,0,0,0}	{0,0,5,0}	1.25
8 lemons	{0,0,8,0,0}	{0,4,2,0,0}	2.00	{8}	{2}	6.00	{0,0,0,0}	{0,4,0,0}	1.00
7 pears	{0,0,0,7,0}	{0,3,6,0,0}	3.20	{7}	{0}	7.00	{0,0,0,0}	{0,3,6,0}	2.25

4.2.4.2 *Detection-based counting:*

Similar to regression-based counting, there are two stages for detection-based counting: the training and the testing stage. We train two CNN object-detection architectures: Faster R-CNN (Ren et al., 2015), and SSD-MobileNet (Howard et al., 2017) for performing fruit detection and apply the CNN models to detect and count the fruits on the test images.

Faster R-CNN is a state-of-the-art object-detection architecture, which was developed based on the Region-based Convolutional Neural Network (R-CNN) (Girshick et al., 2015) and Fast R-CNN (Girshick, 2015). It consists of two networks: (1) a region proposal network (RPN) for generating region proposals or regions of interest (ROI), and (2) a convolutional network for classifying the regions of the image into objects and refining

the bounding box corresponding to each proposal. We used the pre-trained network for the MS-COCO (Lin et al., 2014) dataset to fine-tune Faster R-CNN for detection-based counting.

While Faster R-CNN uses a region proposal network to classify objects, Single Shot Multibox Detector (SSD) (Liu et al., 2016) uses VGG-16 as a backbone network to extract feature maps of different resolutions and applies convolution filters to detect objects. Each cell of the higher feature maps uses a larger area of the image for detecting large objects whereas the smaller feature maps can be used for detecting smaller objects. SSD-MobileNet (Howard et al., 2017) is an adaptive version of the original SSD architecture using the MobileNet network leading to a light-weight model and has been shown to achieve a similar detection accuracy as VGG-16 (Huang et al., 2017). For the detection-based counting task, we used fine-tuning on the network trained on the MS-COCO dataset to train SSD-MobileNet.

We train the CNN models of Faster R-CNN and SSD-MobileNet using either the original or augmented training set of the FTF dataset. Then we evaluate the models on the test set. For each test image, the models predict the bounding boxes of the detected fruits and the confidence of each detected box. We count the number of detected fruits (using a confidence threshold of 0.6) and evaluate the performances of the detection-based counting models by using the same three loss functions used for the regression-based counting approach.

4.3 Results

We train CNNs for regression-based counting and detection-based counting on the original and the augmented training set of the FTF dataset. Then we compare the performance between the models trained on the original set (the O-models) and the models trained on the augmented set (the A-models) by testing the models on the test images. This section reports the results of the regression-based counting and the detection-based counting approaches.

Table 9: Performance of different CNNs (average loss and standard deviation) for regression-based counting on the test set. All models were trained on the original and the augmented training set of the FTF dataset using five-fold cross validation. The bold numbers indicate significant differences between the loss values obtained by the two models ($p < 0.05$).

CNNs	MAE_{CC}		MAE_{CO}		MAE_{MC}	
	O-models	A-models	O-models	A-models	O-models	A-models
Resnet50-Scratch	0.72 ± 0.04	$\mathbf{0.51 \pm 0.02}$	3.10 ± 0.29	$\mathbf{1.93 \pm 0.07}$	0.12 ± 0.04	0.15 ± 0.01
Resnet50-Fine-tuned	0.38 ± 0.03	$\mathbf{0.26 \pm 0.02}$	1.28 ± 0.11	$\mathbf{1.07 \pm 0.09}$	0.15 ± 0.01	$\mathbf{0.05 \pm 0.01}$
Inception-V3-Scratch	0.60 ± 0.03	$\mathbf{0.51 \pm 0.02}$	2.40 ± 0.21	$\mathbf{1.91 \pm 0.10}$	0.15 ± 0.03	0.16 ± 0.01
Inception-V3-Fine-tuned	0.34 ± 0.03	$\mathbf{0.27 \pm 0.03}$	1.35 ± 0.11	$\mathbf{1.12 \pm 0.10}$	0.09 ± 0.01	$\mathbf{0.03 \pm 0.02}$

4.3.1 Regression-based Counting Results

Table 9 shows the performance of the O-models and the A-models for regression-based counting obtained from training ResNet50 and Inception-V3 with both scratch and fine-tuned models.

The MAE_{CC} values indicate that all A-models perform significantly better than the O-models with a $15\% - 31\%$ improvement (e.g., 0.26 vs 0.38). Scratch ResNet50 and fine-tuned ResNet50 profit most from training on the augmented set with an improvement of 29% and 31%, respectively. The highest performance (the lowest average testing error) was achieved by the A-model trained by fine-tuning ResNet50 with $MAE_{CC} = 0.26$.

When considering MAE_{CO}, all A-models also obtain significantly better performances than the O-models. Scratch ResNet50 profits the most from training on the augmented images with a 38% improvement (3.1 to 1.93). The best performing model is the A-model+ResNet50 fine-tuned with $MAE_{CO} = 1.07$.

For MAE_{MC}, which considers the misclassification predictions, the fine-tuned versions of both deep learning architectures profit a lot from training on the augmented set with a loss improvement of 67%. For the scratch versions, the A-models perform similarly to the O-models.

Figure 11 shows some example images from the test set, the ground truths, and the predicted labels obtained from training Fine-tuned-ResNet50 on the original and the augmented training set of the FTF dataset.

	(a)	(b)	(c)
[G]	{0,0,0,11,0}	{0,0,0,0,9}	{0,1,0,0,0}
[O]	{0,0,0,**6**,0}	{0,0,0,0,9}	{0,1,0,0,**1**}
[A]	{0,0,0,11,0}	{0,0,0,0,**10**}	{0,1,0,0,**1**}

Figure 11: Examples of test images, ground truths and predicted outputs obtained from regression-based counting using the Fine-tuned-ResNet50 architecture. (a) the A-model performs better, (b) the O-model performs better, and (c) Misclassified prediction. The [G], [O], and [A] symbols denote the ground truths, predicted labels from the O-model, and predicted labels from the A-model, respectively. The bold numbers in the predicted labels show wrong predicted values.

There are some wrong counts and misclassified images due to different reasons: extensive shading effects, extra objects in the images, a high similarity between fruits and leaves, the occlusion or truncation of some fruits, the similarity among different fruits and possible dissimilarity within the same fruit (i.e., green and ripe fruits or round and oval shapes of pears), which can confuse the CNN models.

4.3.2 Detection-based Counting Results

In this subsection, we report the performances of the O-models and the A-models for the detection-based counting approach obtained from training Faster R-CNN and SSD-MobileNet. Table 10 shows the detection-based counting results.

For all loss functions, there are no significant differences between the A-models and the O-models. This shows that the fruit data-augmentation method is not useful when training the object-detection algorithms Faster

R-CNN and SSD. The reason is that no new fruits (objects) with different appearances are introduced in the images, except for slight differences in scale and rotation. Therefore, the detection algorithm can either locate a specific instance of a fruit or not. The FDA method can therefore only be helpful to make more challenging images with more occlusion, but this does not seem to aid counting performance.

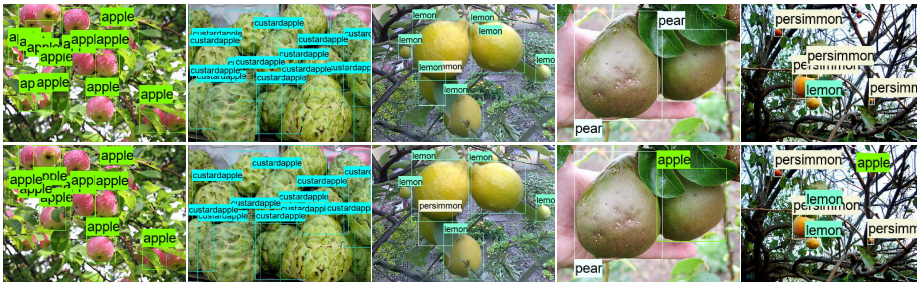
Table 10: Performances of different detection models (average loss and standard deviation) for the detection-based counting methods on the test set. All models were trained on the original and the augmented training set of the FTF dataset using five-fold cross validation. Note that there are no significant differences between the results of the O-models and the A-models ($p < 0.05$).

Object Detection	MAE_{CC}		MAE_{CO}		MAE_{MC}	
	O-models	A-models	O-models	A-models	O-models	A-models
Faster R-CNN	0.22 ± 0.03	0.21 ± 0.03	0.62 ± 0.11	0.67 ± 0.13	0.12 ± 0.02	0.10 ± 0.02
SSD-MobileNet	0.34 ± 0.07	0.35 ± 0.06	1.38 ± 0.34	1.35 ± 0.26	0.08 ± 0.02	0.10 ± 0.02

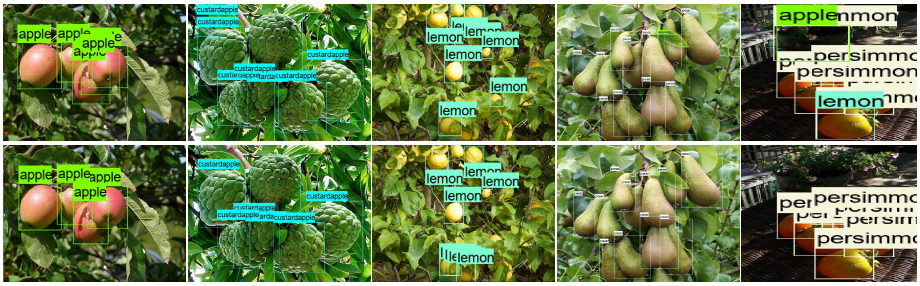
We can also observe that Faster R-CNN performs much better than SSD with MobileNet. This may be because MobileNet is a light-weight CNN, but another reason could be that for this dataset, Faster R-CNN is more accurate.

When we compare the results of the detection-based counting approaches to the regression-based counting approaches, we observe that the results of the Faster R-CNN counting approach are better than the results of the regression-based counting approaches. The only exception to this is that the loss function of the misclassified fruits is lower for the regression-based counting methods when fine-tuned models are used with the augmented dataset. On the other hand, SSD-MobileNet is significantly outperformed by the regression-based approaches when the CNNs are fine-tuned on the augmented dataset.

Figure 12 shows some examples of the test images obtained from training by Faster R-CNN on the original and the augmented set of the FTF dataset. We presented two groups of example images: (a) when the O-model performs better than the A-model, and (b) when the A-model performs better than the O-model.



(a) The O-model (top row) performs better



(b) The A-model (bottom row) performs better

Figure 12: Example of test images obtained from training by Faster R-CNN on the original and the augmented set of the FTF dataset. (a) the O-model performs better than the A-model, and (b) the A-model performs better than the O-model. Some example images show misclassified detection. For each sub-figure, the top row shows the results from the O-model, the bottom row shows the results from the A-model.

4.4 Conclusion

In this chapter, we researched the usefulness of performing data augmentation for counting fruits. The proposed method adds segmented objects (fruits) to existing images and has as advantage that novel labels (counts or bounding boxes) are exactly computed. We also created a novel dataset consisting of images consisting of between 1 and 15 fruits of five different fruit types. For the counting task, we compared two widely different

approaches: a holistic regression-based approach and a detection-based approach. For both approaches we used different convolutional neural networks or object-detection algorithms.

The results show that the fruit data-augmentation method is very helpful for the regression-based approaches. These methods directly predict the total count based on the entire image and profit from the new images which both look different and have higher total counts (which makes the problem more difficult). The performances of the detection-based algorithms did not improve by using the data-augmentation method. This can be explained because no new differently looking fruits are added and therefore locally no new variances are introduced.

Although the best detection-based counting approach (Faster R-CNN) outperforms the regression-based approaches, it requires a human to manually draw bounding boxes around each fruit in the training set. This is much more time-intensive than only labeling an image with the total count. Furthermore, SSD-MobileNet was outperformed by the regression-based counting approaches when data augmentation was used together with pre-trained CNNs.

In future work, we are planning to work on plant-disease detection and will study different data-augmentation methods to deal with the limited amount of training images for diseased plants.

5

ONE-VS-ONE CLASSIFICATION FOR DEEP NEURAL NETWORKS

For performing multi-class classification, deep neural networks almost always employ a One-vs-All (OvA) classification scheme with as many output units as there are classes in a dataset. The problem of this approach is that each output unit requires a complex decision boundary to separate examples from one class from all other examples. In this chapter, we propose a novel One-vs-One (OvO) classification scheme for deep neural networks that trains each output unit to distinguish between a specific pair of classes. This method increases the number of output units compared to the One-vs-All classification scheme but makes learning correct decision boundaries much easier. In addition to changing the neural network architecture, we changed the loss function, created a code matrix to transform the one-hot encoding to a new label encoding, and changed the method for classifying examples. To analyze the advantages of the proposed method, we compared the One-vs-One and One-vs-All classification methods on three plant recognition datasets (including a novel dataset that we created) and a dataset with images of different monkey species using two deep architectures. The two deep convolutional neural network (CNN) architectures, Inception-V3 and ResNet-50, are trained from scratch or pre-trained weights. The results show that the One-vs-One classification method outperforms the One-vs-All method on all four datasets when training the CNNs from scratch. However, when using the two classification schemes for fine-tuning pre-trained CNNs, the One-vs-All method leads to the best performances, which is presumably because the CNNs had been pre-trained using the One-vs-All scheme.

This chapter was published in:

Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L.R.B., and Wiering, M.A. (2020). *One-vs-One Classification for Deep Neural Networks*, Pattern Recognition (PR), Elsevier.

5.1 Introduction

Convolutional neural networks (CNNs) have obtained excellent results for many different pattern recognition problems (Schmidhuber, 2015; LeCun, Bengio, and Hinton, 2015). Most image recognition problems require the CNN to solve a multi-class classification problem. Whereas in the machine learning literature, different approaches have been proposed for dealing with multiple classes (Aly, 2005), in deep learning, the One-vs-All classification scheme is almost universally used. The problem of this method is that decision boundaries need to be learned that separate the examples of each class from examples of all other classes. Especially if images of different classes resemble each other quite a lot, learning such decision boundaries can be very complicated. Therefore, we propose a novel One-vs-One classification scheme for training CNNs in which each output unit only needs to learn to distinguish between examples of two different classes. This should make training the CNN easier and lead to better recognition performance.

Multi-class classification in machine learning. The best-known methods to deal with multi-class classification tasks are One-vs-All (OvA) classification and One-vs-One (OvO) classification (Alpaydin, 2014). Other approaches include One-class classification (Tax, 2001; Tao Ban and Abe, 2006), hierarchical methods (Kumar, Ghosh, and Crawford, 2002; Vural and Dy, 2004), and error-correcting output codes (Dietterich and Bakiri, 1995). One-vs-All (OvA) classification is the most commonly used method for dealing with multi-class problems. In this classification scheme, multiple binary classifiers are trained to distinguish examples from one class from all other examples. When there are K classes, the OvA scheme trains K different classifiers. An advantage of this method is that machine learning algorithms that were designed for binary classification can be easily adapted in this way to deal with multi-class classification problems. A disadvantage is that the dataset on which each classifier is trained becomes imbalanced because there are many more negative examples than positive ones for each classifier.

The One-vs-One (OvO) classification method has also regularly been used for training particular machine learning algorithms such as support vector machines (Allwein, Schapire, and Singer, 2001; Galar et al., 2015; Zhang et al., 2017) or other classifiers (Galar et al., 2011). In the OvO scheme, each binary classifier is trained to discriminate between examples of one class and examples belonging to one other class. Therefore, if there are K classes, the OvO scheme requires training and storing $K(K-1)/2$ different binary classifiers, which can be seen as a disadvantage when K is large. The authors in (Rocha and Goldenstein, 2014) described several methods to cope with a large set of base learners for OvO. Furthermore, different algorithms have been proposed to improve the OvO scheme (Liu, Bi, and Fan, 2017; Songsiri, Cherkassky, and Kijsirikul, 2018). An advantage of the OvO scheme is that the datasets of individual classifiers are balanced when the entire dataset is balanced. Comparisons between using the OvO scheme and the OvA scheme have shown that OvO is better for training support vector machines (Allwein, Schapire, and Singer, 2001; Chih-Wei Hsu and Chih-Jen Lin, 2002) and several other classifiers (Galar et al., 2011).

Multi-class classification in deep neural networks. When deep neural networks are used for multi-class classification problems, the output layer almost always uses a softmax function and one output unit for each different class. This is therefore a One-vs-All classification scheme, although the output units share the same hidden layers. Attribute learning (Farhadi et al., 2009; He and Schomaker, 2018), in which different attributes are predicted, and their combination is used to infer a class, is another promising way to deal with multi-class learning but may require substantially more labeling effort.

Contributions of this chapter. We propose a novel One-vs-One classification method for deep neural networks. The proposed architecture comprises an output layer with $K(K-1)/2$ output units and a shared feature learning part. Each output is trained to distinguish between inputs of two classes and be indifferent to examples of other classes. To construct the OvO classification scheme, we devised three steps: 1) Creating a code

matrix to transform the one-hot encoding to a new label encoding, 2) Changing the output layer and the loss function, and 3) Changing the method to classify new (test) examples.

This OvO scheme has to the best of our knowledge not been proposed before for deep neural networks. We only found one related paper that describes an OvO scheme for shallow neural networks, for which $K(K-1)/2$ different neural networks are trained and stored (Ou and Murphey, 2007). The advantages of our proposed OvO method compared to that more traditional OvO scheme are that we only need to train and store one deep neural network, and our architecture may benefit from positive knowledge transfer when training multiple output units together.

In our experiments, we use three different plant datasets (including a novel dataset called Tropic) and a dataset of different types of monkeys. Using computer vision techniques for classifying plant images plays a vital role in agriculture, monitoring the environment, and automatic plant detection systems (Wang, Liang, and Guo, 2014). Although much research has already been done on recognizing plant images, it is still a difficult and challenging task due to intra-class variations, inter-class similarities, and complex backgrounds (Guru, Sharath, and Manjunath, 2010; Fuentes et al., 2017).

We also use a different dataset consisting of types of monkeys to examine if the results on the plant recognition problems generalize to a different fine-grained species classification problem. Furthermore, we performed experiments with an imbalanced variant of the monkey dataset to study if the OvO scheme can better handle class imbalances. For classifying the image data, two deep CNNs are used, Inception-V3 (Szegedy et al., 2016) and ResNet-50 (He et al., 2016), which are trained from scratch or with fine-tuning from pre-trained weights. Finally, experiments were performed with different amounts of training images and classes from the four datasets using sub-sampling, to study the impact of smaller or larger datasets on the results obtained with the OvO and OvA schemes.

Chapter Outline. The rest of this chapter is organized as follows. Section 5.2 describes and theoretically compares the One-vs-One and One-vs-

All classification methods for deep neural networks. Section 5.3 describes the plant datasets, the monkey dataset, and the data-augmentation methods. The experimental setup is presented in Section 5.4, after which Section 5.5 presents and discusses the results. Section 5.6 concludes the chapter and describes directions for future work.

5.2 A Primer on One-vs-All and One-vs-One Classification

In this section, we explain the two classification schemes (One-vs-All and One-vs-One) for multi-class classification with deep neural networks. Then, we present a theoretical analysis of the advantages of the One-vs-One scheme.

5.2.1 One-vs-All Classification

In multi-class classification, each example belongs to precisely one class. Therefore a dataset is annotated with the correct class label using a one-hot target output vector containing zeros, except for the target class, which has a value of one. The goal is to learn a mapping between inputs and outputs so that the correct class obtains the highest activation and, preferably, is the only one that becomes activated after propagating the inputs to the outputs.

One-vs-All (OvA) classification involves training K different binary classifiers (output units), each designed to discriminate an instance of a given class relative to all other classes (Rifkin and Klautau, 2004). To do this, a softmax activation function is used in the output layer, and the weights of the deep neural network are optimized using the cross-entropy loss function and a particular optimizer.

The categorical cross-entropy loss J_{OvA} for a single training example is:

$$J_{OvA} = -\sum_{i=1}^{K} y_i \log(\hat{y}_i) \tag{8}$$

Where K denotes the number of classes, y_i is defined as the target value (0 or 1) for a given class i, and $\hat{y}_i$ denotes the probability assigned by the network that class i is the correct one. To compute these probabilities, the output values of the network are given to the softmax activation function:

$$\hat{y}_i = \frac{e^{o_i}}{\sum_{j=1}^{K} e^{o_j}} \tag{9}$$

Where o_i represents the output value for class i, which is computed by summing the weighted values passed from the final hidden layer. Note that this final summation uses a weight vector for each class and therefore the activations of the final hidden layer are linearly combined to compute the o_i values. For testing purposes on unseen examples, the predicted output class C is simply computed using:

$$C = \underset{i}{\operatorname{argmax}}\, \hat{y}_i \tag{10}$$

5.2.2 The Proposed One-vs-One Approach

In this subsection, we explain the novel One-vs-One (OvO) classification scheme for traning deep neural networks. As mentioned in the introduction, OvO classification has been used successfully for different machine learning algorithms such as support vector machines. This classification scheme has also been used for training neural networks Ou and Murphey, 2007, for which different (shallow) neural networks were trained separately for each pair of classes. Therefore, that approach leads to the necessity of training many neural networks and no possibility of sharing weights for solving multiple related pattern recognition problems. We present a novel OvO classification scheme that only requires to train a single (deep) neural network. This has as advantages that the method requires less storage space, computational time and can benefit from knowledge transfer and multi-task learning. To construct the OvO classification scheme, we devised three steps: 1) Creating a code matrix, 2) Changing the output layer and the loss function, and 3) Changing the

method to classify new (test) examples. We will explain these steps in detail below.

Creating the OvO code matrix. In OvO classification, instead of using a one-hot target vector that assigns a one to the target class and zeros to all other classes, we need to construct a method that allows for pairwise classification. Therefore, instead of using K outputs where K is the number of classes, we need to construct a target vector consisting of $L = K(K-1)/2$ values. We do this by constructing a code matrix, which converts the one-hot target vector to the target values for the L outputs. The output units in the deep neural network represent binary classifiers with outputs in the bound [-1,1]. The target values for these outputs have values -1, 0, or 1. Here, the value 0 denotes that the output should be indifferent to both classes. For example, when an output unit needs to distinguish cats from dogs, and the training image shows a zebra, the target value for that output unit would be 0. The code matrix M_c has a dimension of $K \times L$. The arrangement of the code matrix entries uses the principle of pairwise separation of classes C_i and C_j, given that $i < j$ (Alpaydin, 2014).

It is easiest to explain the code matrix using an example. Suppose we have a dataset with 5 classes, $K = 5$, so that the number of output units $L = (5 \times 4/2) = 10$. For this example, the code matrix is defined as:

$$M_c = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

When we have the one-hot target vector $\mathbf{y}$ denoting the correct class, we can multiply it with the code matrix to obtain the target outputs for the different output units. For example when $\mathbf{y^T} = (0\ 0\ 0\ 1\ 0)$, which denotes that class 4 is the correct one for a training example, then we can compute the target vector for OvO classification by: $\mathbf{y}_{ovo}^T = \mathbf{y^T} M_c = (0\ 0\ -1\ 0\ 0\ -1\ 0\ -1\ 0\ 1)$, which is simply a copy of the 4^{th} row of the code matrix. In

this example, the 3rd entry in the obtained target vector denotes that for the pairwise classification between classes 1 and 4, the target class is 4, so that the 3rd output unit should output a value of -1.

New output layer and loss function. As explained above, the OvO classification method requires more output units than OvA classification. Although this may mean the OvO scheme is complicated to use when there are a vast number of classes, many datasets do not have more than 50 classes, and in the experiments, we will focus on such (smaller) datasets. To allow the network to output pairwise classifications, we simply construct a deep model with $L = K(K-1)/2$ output units. We cannot use the softmax activation function anymore since that would assign probabilities to all output units, which add up to 1. Furthermore, the novel target output vector contains numbers between -1 and 1. Therefore, in our system, we use the hyperbolic tangent (tanh) activation function for the L output units, defined as:

$$\hat{y}_i = \frac{e^{o_i} - e^{-o_i}}{e^{o_i} + e^{-o_i}} \tag{11}$$

Although this network could be trained with the mean squared error (MSE) loss function, it is well-known that training a neural network for a classification problem can be better done with a cross-entropy loss function Goodfellow, Bengio, and Courville, 2016. Therefore, we customized the binary cross-entropy loss function, for which the target values y_i^{OvO} and output values $\hat{y}_i$ are first scaled to the range [0,1] using:

$$y_i^{OvO'} = \frac{y_i^{OvO} + 1}{2}, \quad y_i' = \frac{\hat{y}_i + 1}{2} \tag{12}$$

For dealing with numerical problems, the probability values of y' are clipped to lie in the range of $[0.00001, 0.99999]$. Now, the multi-output binary cross-entropy loss J_{OvO} for an example is computed with:

$$J_{OvO} = -\frac{1}{L}\sum_{i=1}^{L}(y_i^{OvO'} \times \log(y_i') + (1 - y_i^{OvO'}) \times \log(1 - y_i')) \tag{13}$$

Where $y_i^{OvO'}$ denotes the new target value for a given class i. Note that this loss function is also used for multi-label classification, where multiple outputs can be activated given an input pattern. The difference in our approach is that we include don't care target outputs as well, which need to be mapped to the probability 0.5 or a tanh-activation of 0 in the output layer to minimize the loss. Another choice would be to not train on such outputs at all, but that would provide less information to the network. Some preliminary experiments showed that better results were obtained by also training on target values of zero.

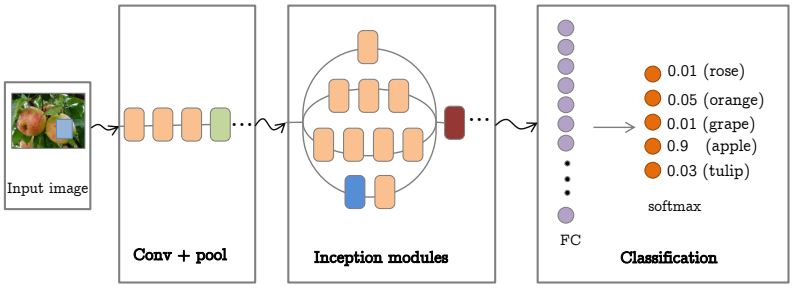
Classifying new examples. To predict the class label C for an input pattern $\mathbf{x}$, the input is first propagated to compute the L outputs $\hat{y}_i$. Then, a decoding scheme is used so that the votes of all binary OvO outputs are combined. For this, the same code matrix M_c is used to compute the summed class output vector $\mathbf{z}$ consisting of K elements:

$$\mathbf{z} = M_c \, \hat{\mathbf{y}} \tag{14}$$

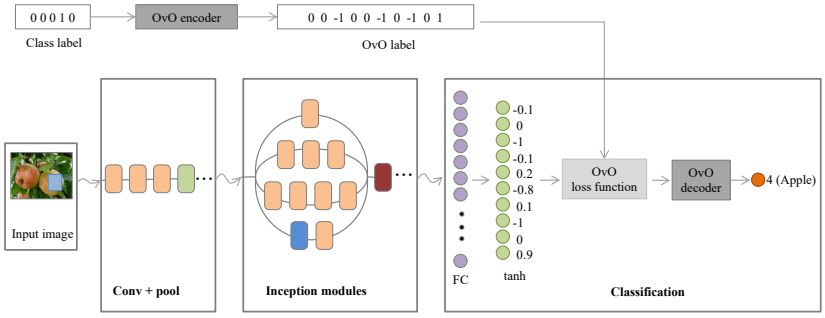
Note that this means that output vector should be similar to the corresponding values in the specific row in the code matrix, although don't care values are not important to get a large summed vote. Finally, the predicted class is selected by $C = \text{argmax}_i \, z_i$. The schematic representation for the deep neural network (Inception-V3) combined with the two classification methods is shown in Figure 13a and Figure 13b.

5.2.3 Analysis of the Advantages of One-vs-One Classification

In this subsection, we theoretically compare the One-vs-One and One-vs-All classification schemes. In our analysis, we will use simple binary classifiers for separating examples of one class from examples of one other class or examples of all other classes. Note that even in deep neural networks, the final output activations are usually computed using a weight matrix that connects the final hidden layer with each output unit. Therefore, the deep neural networks need to learn to map input patterns



(a) One-vs-All



(b) One-vs-One

Figure 13: The pipeline of the CNN showing a compact representation of Inception-V3 combined with the two classification systems; (a) One-vs-All (b) Multi-class One-vs-One. Note that the (...) represents several chains of neural network layers.

to linearly separable final hidden-layer activations. Each classifier first computes its output o_i using:

$$o_i = \mathbf{w_i^T} \cdot \mathbf{h} + b_i \tag{15}$$

Where b_i denotes the bias and $\mathbf{w_i}$ the weight vector for output i, and $\mathbf{h}$ denotes the vector containing all activations of the hidden units that are connected to the outputs. The OvA models use the softmax activation

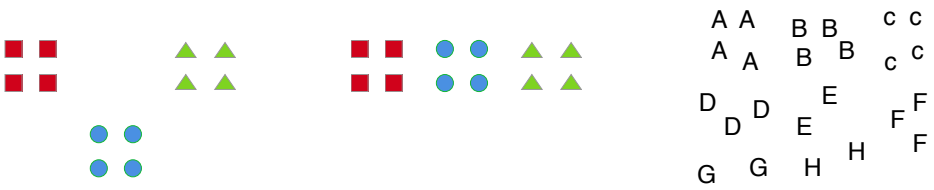
function to compute the class probabilities $\hat{y}_i = \frac{e^{o_i}}{\sum_j e^{o_j}}$ and the predicted class is given by $C = \text{argmax}_i\, \hat{y}_i$.

For simplicity reasons, in our analysis, the OvO models use a sigmoid activation function to discriminate between each pair of classes: $f_{ij} = \sigma(o_{ij})$, and we assume that $f_{ij} = 1 - f_{ji}$ for all $i \neq j$ and zero otherwise. Furthermore, we do not require these OvO models to output values close to 0.5 for different classes than the ones that are separated by the model. Note that the tanh activation function is a scaled sigmoid: $\tanh(x) = 2\sigma(2x) - 1$, so this does not impact our analysis. The predicted class for this OvO scheme on a test example is given by $C = \text{argmax}_i \sum_j f_{ij}$.

We assume a dataset $S = \{(\mathbf{x_1}, C_1), \ldots, (\mathbf{x_n}, C_n)\}$, where C_i denotes the number of the correct output class for input $\mathbf{x_i}$. First, we analyze if the OvO scheme is more powerful than the OvA scheme when separating different classes, for which we define multi-class separability for OvA and OvO.

Definition: OvA separability. A mapping $\mathbf{h} = g(\mathbf{x}, \theta)$ separates all training examples with the OvA scheme, if there exist weight vectors $\mathbf{w_i}$ and biases b_i such that $\text{argmax}_i\, \hat{y}_i = \text{argmax}_i\, \mathbf{w_i^T h} + b_i = C$ for all $(\mathbf{x}, C) \in S$.

Definition: OvO separability. A mapping $\mathbf{h} = g(\mathbf{x}, \theta)$ separates all training examples with the OvO scheme, if there exist vectors $\mathbf{w_{ij}}$ and scalars b_{ij} s.t. $\text{argmax}_i \sum_j f_{ij} = \text{argmax}_i \sum_j \sigma(\mathbf{w_{ij}^T h} + b_{ij}) = C$ for all $(\mathbf{x}, C) \in S$.

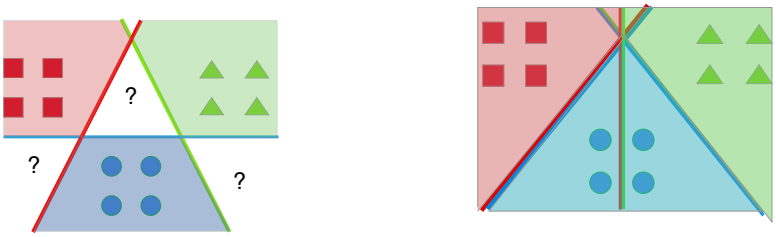


(a) Simple 3 classes. (b) Complex 3 classes. (c) Complex 8 classes.

Figure 14: Three different multi-class problems of different complexities.

We will first give an example with three linearly separable classes so that both the OvA and OvO scheme construct three decision boundaries,

see Figure 14a. It should be clear that the three classes in Figure 14a are linearly separable with OvA and OvO. The optimal decision boundaries are illustrated in Figure 15a and Figure 15b.



(a) Separation with OvA.

(b) Separation with OvO.

Figure 15: The optimal decision boundaries.

When we compare the decision boundaries for OvA and OvO, we observe several differences. First, the decision boundaries are placed in different ways. E.g., the red and green classes are separated by OvO by a vertical line in the middle. Second, with the OvO scheme, there is always one class that wins against all other classes for each input. For the OvA scheme, there are possible inputs for which there is no unique winner, such as points in the bottom left area where both the blue circle class and the red square class may have high outputs. The predicted class in such areas would depend on the exact weight vectors and bias values.

Now, let us examine the more complex problem shown in Figure 14b. The OvA scheme will have difficulties to learn to separate the blue circles from the examples of the other two classes. Although learning the correct decision boundaries is complicated for the OvA scheme, it is still possible. The blue-class model could have a higher bias value than the other models and be less sensitive to the input, and the other two classes could learn decision boundaries based on the x-axis. The OvO scheme can easily solve this problem, however, because linear divisions between each pair of classes are not hard to construct.

If we make the problem even more complex and add more classes, such as in Figure 14c, it seems impossible for the OvA scheme to separate all

classes. However, also in this case the OvA scheme can linearly separate the classes, which we will prove below. It should be noted that it is much easier for the OvO scheme to handle such a dataset.

Now, suppose we have a dataset with K classes and one input dimension h, in which each class is linearly separable from each other class using the OvO scheme. Figure 16 shows an example of such a problem with 4 classes A, B, C, and D. Note that for simplicity, we only drew a single data point for each class, but the analysis can be easily extended to multiple data points, as long as they lie close together.

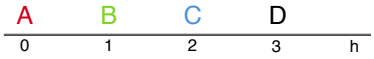


Figure 16: 1D-Problem with 4 classes.

We now make the following proposition: **Proposition 1:** If all pairs of classes are linearly separable (in one dimension), then the OvA scheme can also linearly separate all classes, but requires larger weight values to do this than the OvO scheme.

Proof of proposition 1: We assume we have K points $h_1, h_2, \ldots, h_K$ and K OvA models $f_i(h) = w_i h + b_i$. We require that each model f_i outputs the largest value on point h_i: $f_i(h_i) \geqslant f_j(h_i) + R$ for all $i, j \in \{1, 2, \ldots, K\}; i \neq j$. Here R is a positive constant that ensures the differences between model outputs are large enough so that the softmax function would output a value close to 1 for the winning class (e.g. $R = 3$).

It is not difficult to develop an algorithm that constructs the parameters w_i, b_i for all models f_i such that the above requirement holds. Let's look at the example of Figure 16 again. In this example class A belongs to point $h = 0$, B to $h = 1$, C to $h = 2$, and D to $h = 3$. We have four models $f_z(h) = w_z h + b_z$, where z is the label (A, B, C, or D). For separating A and B, we require:

$$f_A(0) = f_B(0) + R \quad \text{and} \quad f_B(1) = f_A(1) + R \tag{16}$$

There are multiple solutions, let's say we select:

$$f_A(h) = -Rh + 0.5R \quad \text{and} \quad f_B(h) = Rh - 0.5R \tag{17}$$

It is easy to verify that the previous requirement is fulfilled with these two models. Now, for class C, we require:

$$f_B(1) = f_C(1) + R \quad \text{and} \quad f_C(2) = f_B(2) + R \tag{18}$$

From which follows: $f_C(h) = 3Rh - 3.5R$. When we continue this construction process, we also derive: $f_D(h) = 5Rh - 8.5R$.

We observe that the function $\max_i f_i$ is piece-wise linear convex, which is illustrated for the models for A, B, and C in Figure 17a.

(a) Solution of OvA.

(b) Solution of OvO.

Figure 17: The solutions for the 1D problem.

It is easy to show that the algorithm can be generalized to multiple input dimensions. In the 1D case, we observed that the weights increase by 2R for each additional model, while the bias values become very negative. This finally leads to substantial weight values when there are many classes, and consequently, will decrease the generalization power. The weight-increase factor for each additional model depends on other problem-specific settings, such as the distance between examples in feature space δ (in our example $\delta = 1$), and the number of dimensions of the final hidden layer, H.

When dealing with H dimensions, the increase of the single weight can be spread over the H dimensions, so the increase of weights is $\frac{2R}{H}$ for each additional class. Therefore, projecting inputs to many hidden dimensions helps to have smaller weights, but many hidden units may also worsen generalization. When examples of different classes are closer together, the margin decreases, and the weight increase has to be multiplied with $\frac{1}{\delta}$. This also means that unbounded activation functions (e.g., ReLU) are useful for obtaining smaller weights in the final classification layer. When

we take all these factors together, the OvA scheme's largest weight would be of the order $\frac{KR}{\delta H}$. E.g., for 50 classes ($K = 50$), $\delta = 0.1$, $R = 3$, and $H = 100$, the largest weights in the final classification layer could be around 15.

Now, examine how the OvO scheme solves the above problem. In this scheme, we use models of the form $f_{ij}(h) = w_{ij}h + b_{ij}$. For the first classes A and B, we require: $f_{AB}(0) = R$ and $f_{AB}(1) = -R$ to ensure that after applying the sigmoid function, the model incurs a small loss.

It is easy to see that for $f_{AB}(h)$ the weight w_{AB} equals $-2R$, similar to the OvA scheme. However, the different models do not depend on each other, and therefore the weights do not need to increase continuously. Furthermore, models that separate examples that are farther away from each other, such as $f_{AD}(h)$, can have much smaller weight values. The solution of the OvO scheme to the one-dimensional problem is illustrated in Figure 17b.

This concludes our proof of proposition 1. Both classification schemes can be used to separate the data projected to one dimension as long as examples of different classes lie close together, but the OvA model needs much larger weights if there are many classes. Another problem with the OvA scheme is that the different outputs heavily depend on each other. When one binary OvA classifier is adapted, other outputs have to be changed as well. Furthermore, when some outputs use large weight vectors in the final layer, their errors can have a significant impact on the training process. These two factors may increase instabilities of the training process.

The learned representation can indeed make up for the problems of the OvA scheme. For example, when the final hidden layer is very large, it is easier to learn decision boundaries with OvA. However, this could lead to strange generalization effects, as has also been shown in research on adversarial examples Goodfellow, Bengio, and Courville, 2016. Furthermore, in the OvO scheme, outputs are affected by other outputs due to the shared feature-learning part, but this dependence also occurs for the

OvA models. To conclude, the OvO scheme has the following advantages compared to the OvA scheme:

- The OvO scheme can have better generalization properties than the OvA scheme because there is less need for large weight vectors or a broad final feature representation, which is connected to the classification layer.
- In the OvA scheme, each binary classifier (output) is much more dependent on the other binary classifiers than in the OvO scheme, which could increase problems with learning instabilities.
- The OvO scheme does not introduce artificial class imbalances, whereas the OvA scheme does. If the dataset is balanced, the problem for each OvO classifier is balanced as well. For the OvA scheme, the dataset for each independent classifier is imbalanced.

Finally, we want to mention that although in general the OvO scheme requires training $K(K-1)/2$ different classifiers and therefore could cost much more training time than the OvA scheme, in our proposed architecture this is not the case. In the proposed OvO method, a single deep network is used that is trained on each example in the same way as in the OvA scheme. Only when there are very many classes (like thousands), the OvO scheme would become complex to store and train.

5.3 Datasets and Data Augmentation

As mentioned in the introduction, plant image recognition systems have many applications. Convolutional neural networks (CNNs) have obtained remarkable results on different datasets for image-based plant classification (Fuentes et al., 2017; Ubbens and Stavness, 2017; Cruz et al., 2017; Ubbens et al., 2018). In Mohanty, Hughes, and Salathé, 2016, two deep learning architectures, AlexNet and GoogLeNet, were trained on the PlantVillage dataset to detect plant leaves that contain diseases. The

work described in Too et al., 2019 compared instances of Inception-V4, various instances of ResNet, and few other CNN models to classify diseases in plant images. Some works have also applied several other techniques to boost recognition performances, such as using different kinds of data augmentation (Zhang et al., 2015a; Pawara et al., 2017a) and transfer learning schemes (Douarre et al., 2018).

In this section, we briefly describe the three different plant datasets, the monkey dataset, and the data augmentation methods used in our study.

5.3.1 Datasets

In this subsection, we describe the three plant datasets and the monkey dataset used in the experiments. Figure 18 shows some example images from the plant datasets.

5.3.1.1 *AgrilPlant Dataset*

The AgrilPlant dataset was introduced in (Pawara et al., 2017b). The dataset contains 3,000 plant images with a uniformly distributed number of images per class. It contains 10 classes: Apple, Banana, Grape, Jackfruit, Orange, Papaya, Persimmon, Pineapple, Sunflower, and Tulip. Most of the images within this dataset contain variances in pose and object backgrounds. The dataset images were split in the proportion of 20% used for testing, and the remaining 80% of the images used for training.

5.3.1.2 *Tropic Dataset*

The Tropic dataset contains 20 classes of plants with a total of 5,276 images. Each of the classes contains a non-uniform distribution of images, varying from 221 to 371 images per class. The dataset contains the following plants: Acacia, Ashoka, Bamboo, Banyan, Chinese wormwood, Croton, Crown flower, Ervatamia, Golden shower, Hibiscus, Lady palm, Lime, Mango, Manila tamarind, Poinsettia, Raspberry ice Bougainvillea, Sanchezia, Umbrella tree, West Indian jasmine, and White plumeria. The images were collected by us during the day using a DSLR camera. The

Figure 18: Some example images from the three plant datasets for which we show one image per class for some classes in the datasets. The first row shows *AgrilPlant* images, the second row shows *Tropic* images, and the last row shows *Swedish* leaf images.

data was collected from diverse locations in Northeastern Thailand. All the images have similarities in illumination conditions but show different plant parts (flowers, branches, fruits, leaves, or the whole tree) and background information such as sky, houses, and soil. We randomly split the dataset in the ratio of 70% / 30% for the training and the testing set.

5.3.1.3 *Swedish Dataset*

The Swedish dataset (Söderkvist, 2001) contains 1,125 leaf images of 15 classes with 75 images per class. The leaf images were taken on a plain background. We adopted the same dataset splits as in previous studies using 25 randomly selected images per class for training and the rest of the images for testing.

5.3.1.4 *Monkey-10 Dataset*

The Monkey-10 dataset [1] contains approximately 1,400 images and 10 classes, and each class corresponds to a different species of monkeys. Each of the classes contains approximately 110 training images and 27 test images. The dataset consists of the following monkey species: Mantled

1 https://www.kaggle.com/slothkong/10-monkey-species

Figure 19: Some example images from the *Monkey-10* dataset for which we show one image per class for all classes in the dataset.

howler, Patas monkey, Bald uakari, Japanese macaque, Pygmy marmoset, White-headed capuchin, Silvery marmoset, Common squirrel monkey, Black-headed night monkey, and Nilgiri langur. Figure 19 shows some example images from the Monkey-10 dataset.

The Monkey-10 dataset was primarily used to observe if performance differences between the OvO and OvA schemes generalize to a different kind of fine-grained species dataset. Additionally, from the original Monkey-10 dataset, we randomly selected a non-uniform distribution of images from the training set, which varies from 10 to 120 images per class to create an imbalanced dataset. This dataset is called Imbalanced-Monkey-10 and serves as a purpose to study if the OvO or OvA scheme can better handle strongly imbalanced classes.

5.3.2 Data Augmentation Techniques

We applied three online data augmentation (DA) approaches during the training of the CNNs. The data-augmentation operations involve horizontal flipping, vertically shifting images up or down with random values with a maximum of 10% of the image height, and horizontally shifting images left or right with random values with a maximum of 10% of the image width (where novel pixels are filled in using nearest pixel values). These operation schemes were applied to all the training images of the datasets.

The reason for using DA is to increase the size of the training dataset when training the CNN models.

5.4 Experimental Setup

In this section, we present the different experimental setups in which we subsample the total amount of images and classes from the three plant datasets and the two monkey datasets. Afterwards we describe the experimental parameters used for training the two CNNs, Inception-V3 and ResNet-50.

5.4.1 Dataset Sampling

This subsection describes two different forms of dataset sampling to obtain more dataset subsets that will be used in the experiments:

1. Dataset subsets with fewer classes: In the AgrilPlant dataset, we additionally considered 5 randomly selected classes from the original dataset; this version of the dataset is called AgrilPlant5 while the original dataset is called AgrilPlant10. For the Tropic dataset, we considered two additional subsets from the original dataset, which involves the random selection of 5 or 10 classes from the original dataset. Hence, we name the new and original datasets (Tropic5, Tropic10) and Tropic20, respectively. Similar considerations were made on the Swedish dataset for 5 and 10 randomly selected classes. Hence, this results in the new subset variants; Swedish5 and Swedish10, while the original dataset is called Swedish15.

2. Dataset subsets in which the original training image examples (100%) were distributed into 10%, 20%, 50%, and 80% of the whole training set based on a random selection of the images. Table 11 shows the number of images per class of the datasets after sub-sampling. Note that the testing sets for the datasets were kept constant. Furthermore, we provide notations for describing the datasets using:

<dataset name><number of classes>::ts<train size>. For example, Tropic20::ts10 denotes the Tropic dataset with 20 classes containing 10% of the training data.

The reason for performing experiments with the sub-sampling dataset variations is to determine how the CNN architectures combined with either the OvO or OvA classification system can deal with recognizing images under different conditions. The primary goal is to assess the performance variations of the two different classification schemes.

Table 11: Number of training images per class after sub-sampling the datasets.

Train size (%)	Dataset				
	AgrilPlant	Tropic	Swedish	Monkey	Imbalanced Monkey
10	24	15-26	2-3	10-12	1-12
20	48	31-52	5	21-24	2-24
50	120	77-130	12-13	52-61	5-61
80	192	124-207	20	84-98	8-98
100	240	155-259	25	105-120	10-120

5.4.2 Deep CNN Training Schemes

Deep neural network architectures consist of several chains of neural network layers and operations: convolutional, normalization, non-linear activation functions, pooling, fully-connected, and the final classification layer. In this study, we perform experiments with architectures which use inception modules (Inception-V3), and residual modules (ResNet-50). We chose these deep CNN architectures, because they are well known state-of-the-art architectures, but are based on different operations (inception or residual modules).

We trained the CNN models with two training schemes using the scratch or pre-trained version based on their use of random weights or pre-trained

weights from the ImageNet dataset. Each of the training schemes employs the previously described deep convolutional neural networks (Inception-V3 and ResNet-50) combined with the OvA and OvO classification systems. The hyper-parameters were optimized using several preliminary experiments.

1. Scratch Experiments. The following experimental parameters were used: the previously described CNNs were initialized with random weights and trained for 200 epochs while optimizing the CNN loss function with the Adam optimizer, a batch size of 16, and a learning rate $l_r = 0.001$. The l_r decay uses a factor of 0.1 after every interval of 50 epochs. The scratch experiments on all the datasets were run within the computing time frame of $[10 - 130]$ minutes, depending on the given dataset/subset.

2. Fine-tuning Experiments. The following experimental parameters were used: the previously described CNNs were initialized with pre-trained weights from the ImageNet dataset. These models are trained for 100 epochs while optimizing the CNN loss function with the Adam optimizer, a batch size of 16, and a learning rate $l_r = 0.0001$. The l_r decay uses a factor of 0.1 after 50 epochs. The fine-tuning experiments on all the datasets were run within the computing time frame of $[6 - 66]$ minutes, depending on the given dataset/subset.

We used an NVIDIA V100 GPU with 28GB of memory for all experiments.

5.5 Results and Discussion

In this section, we present the classification performances of the two CNN methods (Inception-V3 and ResNet-50) combined with the two classification schemes (OvO and OvA) trained using the scratch or pre-trained instances of the CNN models on the three plant datasets, the monkey datasets, and some of the plant datasets without data augmentation on the training sets.

5.5.1 Results of Scratch-Inception-V3

We trained the scratch Inception-V3 CNN based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 12.

Table 12: Recognition performances (average accuracy and standard deviation) of Scratch-Inception-V3 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) The AgrilPlant dataset

Train size (%)	AgrilPlant5		AgrilPlant10	
	OvO	OvA	OvO	OvA
10	**77.13 ± 1.28**	71.67 ± 2.67	**77.80 ± 3.00**	73.57 ± 1.47
20	85.47 ± 2.10	83.33 ± 3.47	86.97 ± 1.69	85.87 ± 1.57
50	**92.40 ± 0.86**	89.73 ± 1.19	94.87 ± 1.00	94.57 ± 1.23
80	94.47 ± 0.90	94.33 ± 0.53	96.47 ± 0.69	96.60 ± 0.73
100	94.93 ± 0.37	94.80 ± 1.02	96.90 ± 0.65	97.40 ± 0.67

(b) The Tropic dataset

Train size (%)	Tropic5		Tropic10		Tropic20	
	OvO	OvA	OvO	OvA	OvO	OvA
10	**82.24 ± 1.91**	78.76 ± 2.09	**75.14 ± 2.73**	70.46 ± 3.22	66.51 ± 4.72	65.93 ± 3.31
20	89.06 ± 1.55	89.40 ± 1.47	**86.77 ± 1.14**	83.43 ± 2.06	81.48 ± 4.52	80.57 ± 1.35
50	**97.19 ± 0.66**	95.74 ± 1.15	95.59 ± 1.28	94.78 ± 0.34	94.62 ± 1.67	94.47 ± 0.46
80	**98.84 ± 0.53**	98.02 ± 0.47	98.38 ± 0.70	97.42 ± 0.73	**97.87 ± 0.34**	97.21 ± 0.31
100	99.13 ± 0.51	98.30 ± 1.06	98.56 ± 0.46	98.54 ± 0.22	98.18 ± 0.96	98.03 ± 0.14

(c) The Swedish dataset

Train size (%)	Swedish5		Swedish10		Swedish15	
	OvO	OvA	OvO	OvA	OvO	OvA
10	**71.60 ± 4.24**	66.08 ± 3.01	**79.52 ± 3.43**	70.96 ± 4.19	**72.91 ± 5.29**	65.41 ± 3.32
20	86.40 ± 2.61	86.96 ± 4.36	**91.84 ± 2.25**	85.60 ± 3.90	**88.73 ± 1.98**	84.99 ± 2.71
50	**98.40 ± 0.75**	95.36 ± 2.63	97.36 ± 0.86	97.36 ± 0.96	95.71 ± 1.41	94.99 ± 1.85
80	**99.36 ± 0.36**	98.56 ± 0.61	99.20 ± 0.58	98.48 ± 0.39	98.19 ± 0.49	97.41 ± 0.75
100	99.76 ± 0.36	99.44 ± 0.67	99.48 ± 0.18	99.00 ± 0.51	**98.59 ± 0.28**	97.76 ± 0.45

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 12a, we observe that training Scratch-Inception-V3 (CNN) combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 3 dataset subsets with a smaller training size. Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the AgrilPlant5::ts10 dataset with a significant difference of $\sim 5.5\%$.

2. Evaluation of the CNN on the Tropic Dataset: from Table 12b, we observe that training Scratch-Inception-V3 combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 6 dataset subsets.

3. Evaluation of the CNN on the Swedish Dataset: from Table 12c, we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA ($p < 0.05$) on 8 datasets (subsets or whole). Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the Swedish10::ts10 dataset with a significant difference of 8.5%.

5.5.2 Results of Scratch-ResNet-50

We trained the scratch ResNet-50 combined with the two classification schemes using five-fold cross-validation. The results obtained during the testing phase are reported in Table 13.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 13a, we observe that training Scratch-ResNet-50 combined with OvO significantly outperforms the CNN combined with OvA on 4 smaller subsets.

2. Evaluation of the CNN on the Tropic Dataset: from Table 13b, we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 6 subsets of this dataset. Another observation is that the CNN combined with OvO surpasses the CNN combined with OvA on the Tropic10::ts{10,20} subsets with a significant difference of $\sim 5\%$.

3. Evaluation of the CNN on the Swedish Dataset: from Table 13c, we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 4 subsets of this dataset. Furthermore, the CNN combined with OvO surpasses the CNN combined with OvA on the Swedish10::ts10 dataset with a difference of $\sim 10\%$.

Table 13: Recognition performances (average accuracy and standard deviation) of Scratch-ResNet-50 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) The AgrilPlant dataset

Train size (%)	AgrilPlant5		AgrilPlant10	
	OvO	OvA	OvO	OvA
10	**77.53 ± 0.96**	72.93 ± 3.85	**76.23 ± 2.06**	72.93 ± 2.04
20	**85.40 ± 0.64**	82.73 ± 2.29	86.03 ± 1.29	84.20 ± 1.91
50	**91.47 ± 0.90**	89.87 ± 0.77	93.13 ± 0.46	93.20 ± 0.83
80	93.53 ± 1.22	93.73 ± 1.50	96.00 ± 0.53	95.03 ± 1.19
100	94.33 ± 0.94	93.87 ± 2.06	96.10 ± 0.38	96.23 ± 0.85

(b) The Tropic dataset

Train size (%)	Tropic5		Tropic10		Tropic20	
	OvO	OvA	OvO	OvA	OvO	OvA
10	**77.31 ± 1.05**	73.59 ± 2.63	**67.57 ± 3.44**	62.38 ± 1.42	59.78 ± 2.05	59.59 ± 2.27
20	87.41 ± 3.72	83.35 ± 3.45	**82.57 ± 1.75**	77.85 ± 2.10	**79.79 ± 0.72**	76.61 ± 1.31
50	93.47 ± 2.48	91.19 ± 2.40	93.45 ± 1.20	93.09 ± 0.76	93.31 ± 0.61	93.11 ± 1.02
80	97.29 ± 1.35	96.23 ± 0.89	96.45 ± 1.20	96.43 ± 0.88	96.49 ± 0.48	95.70 ± 0.70
100	**98.64 ± 0.82**	97.48 ± 0.44	97.44 ± 0.42	97.10 ± 0.57	**97.59 ± 0.23**	96.80 ± 0.43

(c) The Swedish dataset

Train size (%)	Swedish5		Swedish10		Swedish15	
	OvO	OvA	OvO	OvA	OvO	OvA
10	**75.20 ± 1.96**	71.76 ± 1.95	**73.52 ± 3.57**	63.44 ± 1.99	66.11 ± 4.18	66.83 ± 2.49
20	86.80 ± 3.26	83.53 ± 1.61	82.32 ± 4.81	83.60 ± 2.53	84.05 ± 4.12	82.21 ± 1.81
50	96.08 ± 0.95	96.48 ± 1.34	95.56 ± 0.83	95.68 ± 0.99	93.31 ± 0.90	93.15 ± 1.20
80	98.24 ± 0.83	97.92 ± 0.91	**98.00 ± 0.40**	97.12 ± 0.46	96.19 ± 1.00	96.03 ± 0.61
100	98.96 ± 0.46	98.72 ± 0.52	98.40 ± 0.37	98.32 ± 0.23	**97.28 ± 0.35**	96.24 ± 0.94

5.5.3 Results of Fine-tuned Inception-V3

We trained the pre-trained Inception-V3 based on five-fold cross-validation. The results obtained during the testing phase are shown in Table 14.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 14a, the results show that there are 3 subsets of this dataset where training the Fine-tuned-Inception-V3 combined with OvA significantly outperforms the CNN combined with OvO.

2. Evaluation of the CNN on the Tropic Dataset: from Table 14b, we observe that the CNN combined with OvA significantly outperforms

Table 14: Recognition performances (average accuracy and standard deviation) of Fine-tuned-Inception-V3 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) The AgrilPlant dataset

Train size (%)	AgrilPlant5		AgrilPlant10	
	OvO	OvA	OvO	OvA
10	88.67 ± 2.13	90.40 ± 2.42	92.13 ± 1.52	**94.87 ± 0.88**
20	92.27 ± 2.09	92.07 ± 1.86	94.47 ± 1.77	**96.67 ± 0.59**
50	96.20 ± 1.66	96.27 ± 1.14	97.13 ± 1.02	98.03 ± 0.77
80	96.27 ± 1.16	97.53 ± 0.69	97.93 ± 0.51	**98.77 ± 0.57**
100	97.00 ± 1.18	97.07 ± 1.23	98.07 ± 0.56	98.83 ± 0.53

(b) The Tropic dataset

Train size (%)	Tropic5		Tropic10		Tropic20	
	OvO	OvA	OvO	OvA	OvO	OvA
10	97.15 ± 1.72	96.61 ± 2.50	92.93 ± 1.21	94.60 ± 1.52	90.42 ± 2.88	**93.60 ± 0.94**
20	97.39 ± 1.22	98.74 ± 0.99	96.01 ± 0.98	**98.25 ± 0.57**	95.70 ± 0.36	**96.67 ± 0.52**
50	99.32 ± 0.32	99.47 ± 0.56	98.75 ± 0.27	**99.53 ± 0.41**	98.43 ± 0.21	**99.20 ± 0.10**
80	99.66 ± 0.13	99.61 ± 0.22	99.32 ± 0.23	**99.79 ± 0.15**	99.05 ± 0.35	99.46 ± 0.23
100	99.76 ± 0.24	99.81 ± 0.32	99.56 ± 0.22	**99.87 ± 0.16**	99.33 ± 0.09	**99.68 ± 0.12**

(c) The Swedish dataset

Train size (%)	Swedish5		Swedish10		Swedish15	
	OvO	OvA	OvO	OvA	OvO	OvA
10	94.88 ± 4.10	92.48 ± 4.23	84.56 ± 2.56	**91.72 ± 4.44**	87.52 ± 4.78	86.11 ± 2.04
20	97.44 ± 3.26	97.52 ± 3.06	97.68 ± 1.40	98.96 ± 0.71	95.55 ± 2.34	94.48 ± 3.33
50	99.68 ± 0.18	**99.98 ± 0.04**	99.72 ± 0.11	99.84 ± 0.17	99.23 ± 0.40	99.20 ± 0.21
80	99.92 ± 0.18	99.92 ± 0.18	99.76 ± 0.17	99.88 ± 0.11	99.60 ± 0.27	99.81 ± 0.20
100	99.92 ± 0.18	99.92 ± 0.18	99.92 ± 0.11	99.92 ± 0.18	99.79 ± 0.15	**99.97 ± 0.06**

the CNN combined with OvO on 8 subsets of the Tropic10 and Tropic20 datasets.

3. Evaluation of the CNN on the Swedish Dataset: from Table 14c, we observe that training the CNN combined with OvA significantly outperforms the CNN combined with OvO on 3 subsets of this dataset. Another observation is that the CNN combined with OvA surpasses the CNN combined with OvO on the Swedish10::ts10 dataset with a significant difference of ~ 7%.

Table 15: Recognition performances (average accuracy and standard deviation) of Fine-tuned ResNet-50 combined with the two classification methods. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) The AgrilPlant dataset

Train size (%)	AgrilPlant5		AgrilPlant10	
	OvO	OvA	OvO	OvA
10	91.13 ± 1.39	89.47 ± 3.03	93.13 ± 1.57	93.17 ± 0.31
20	93.93 ± 2.47	92.40 ± 1.16	95.83 ± 1.87	96.17 ± 0.87
50	96.33 ± 1.62	96.07 ± 0.64	97.73 ± 1.11	97.67 ± 0.94
80	97.27 ± 0.86	97.07 ± 1.34	98.40 ± 0.48	98.47 ± 0.40
100	97.60 ± 1.44	97.33 ± 1.33	98.47 ± 0.70	98.63 ± 0.70

(b) The Tropic dataset

Train size (%)	Tropic5		Tropic10		Tropic20	
	OvO	OvA	OvO	OvA	OvO	OvA
10	96.80 ± 1.45	96.61 ± 1.20	92.54 ± 1.91	91.96 ± 1.20	90.54 ± 1.09	90.76 ± 1.40
20	98.16 ± 0.88	97.87 ± 1.09	95.80 ± 0.89	**97.70 ± 0.30**	93.96 ± 0.49	**96.27 ± 0.42**
50	99.52 ± 0.38	99.22 ± 0.47	98.72 ± 0.29	**99.19 ± 0.17**	98.17 ± 0.63	**99.05 ± 0.10**
80	99.66 ± 0.37	99.56 ± 0.32	99.24 ± 0.28	**99.71 ± 0.25**	98.80 ± 0.21	**99.38 ± 0.15**
100	99.66 ± 0.28	99.76 ± 0.24	99.58 ± 0.11	99.71 ± 0.17	99.23 ± 0.18	**99.49 ± 0.16**

(c) The Swedish dataset

Train size (%)	Swedish5		Swedish10		Swedish15	
	OvO	OvA	OvO	OvA	OvO	OvA
10	90.48 ± 4.79	89.68 ± 6.14	90.40 ± 2.37	87.88 ± 1.88	84.32 ± 4.39	85.47 ± 3.22
20	97.44 ± 1.85	98.08 ± 2.14	98.76 ± 0.96	96.80 ± 2.04	97.47 ± 2.54	94.32 ± 3.62
50	99.76 ± 0.36	99.60 ± 0.28	99.60 ± 0.20	99.72 ± 0.23	99.47 ± 0.27	99.49 ± 0.33
80	99.76 ± 0.36	99.92 ± 0.18	99.92 ± 0.18	99.68 ± 0.39	99.71 ± 0.17	99.79 ± 0.24
100	99.92 ± 0.18	99.92 ± 0.18	99.92 ± 0.11	99.92 ± 0.18	99.65 ± 0.49	99.68 ± 0.20

5.5.4 Results of Fine-tuned ResNet-50

We trained the pre-trained ResNet-50 combined with the two classification methods based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 15.

1. Evaluation of the CNN on the AgrilPlant Dataset: from Table 15a, we observe that training the CNN combined with OvO results in similar performance levels to the CNN combined with OvA on this dataset.

2. Evaluation of the CNN on the Tropic Dataset: from Table 15b, we observe that training the CNN combined with OvA significantly out-

performs the CNN combined with OvO on 7 subsets of the datasets with more classes.

3. Evaluation of the CNN on the Swedish Dataset: from Table 15c, the results show that there is no significant difference between training the CNN with the two classification methods on all subsets of this dataset.

5.5.5 Results on the Monkey Datasets

We trained the two CNNs from scratch or using pre-trained weights using the two classification methods on the two monkey datasets, Monkey-10 and Imbalanced-Monkey-10, based on five-fold cross-validation. The results obtained during the testing phase are reported in Table 16.

1. Evaluation of Scratch Inception-V3 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 16a, we observe that training the CNN combined with OvO significantly outperforms the CNN combined with OvA on 5 (smaller) subsets of the Monkey-10 datasets with several times significant differences of $\sim 7\%$.

2. Evaluation of Scratch Resnet-50 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 16b, we observe that training the CNN combined with OvO on Monkey-10 results in one case in a significantly better performance (Monkey10:ts10) with a significant difference of 5%.

3. Evaluation of Fine-tuned Inception-V3 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 16c, we observe that training the CNN combined with OvA significantly outperforms the CNN combined with OvO on one data subset of Monkey-10 and Imbalanced-Monkey-10.

4. Evaluation of Fine-tuned Resnet-50 on the Monkey-10 and Imbalanced-Monkey-10 datasets: from Table 16d, the results show that there is no significant difference between training the CNN with the two

Table 16: Recognition performances (average accuracy and standard deviation) of the studied CNNs combined with the two classification methods applied on the Monkey-10 datasets. The bold numbers indicate significant differences between the classification methods ($p < 0.05$).

(a) Scratch Inception-V3

Train size (%)	Monkey10		Imbalanced-Monkey10	
	OvO	OvA	OvO	OvA
10	**55.91 ± 1.12**	48.68 ± 5.35	38.11 ± 3.38	35.04 ± 3.49
20	**68.91 ± 2.45**	61.47 ± 3.70	**48.24 ± 4.90**	41.17 ± 4.78
50	**86.28 ± 0.63**	84.10 ± 1.95	**66.79 ± 1.99**	61.97 ± 2.63
80	93.00 ± 1.73	90.94 ± 1.94	75.33 ± 1.67	72.04 ± 3.31
100	94.16 ± 1.70	92.69 ± 1.19	78.25 ± 1.78	75.99 ± 2.34

(b) Scratch Resnet-50

Train size (%)	Monkey10		Imbalanced-Monkey10	
	OvO	OvA	OvO	OvA
10	**54.52 ± 2.49**	49.49 ± 0.98	36.43 ± 4.20	34.39 ± 2.41
20	67.66 ± 3.48	62.91 ± 3.27	42.57 ± 5.79	40.64 ± 3.43
50	80.81 ± 2.83	81.46 ± 1.19	63.64 ± 3.00	59.55 ± 3.10
80	89.56 ± 2.07	89.64 ± 0.71	70.22 ± 3.89	68.32 ± 2.77
100	92.33 ± 1.41	90.73 ± 1.30	74.53 ± 2.47	72.47 ± 3.39

(c) Fine-tuned Inception-V3

Train size (%)	Monkey10		Imbalanced-Monkey10	
	OvO	OvA	OvO	OvA
10	95.69 ± 1.42	96.86 ± 1.32	78.85 ± 6.24	75.11 ± 2.67
20	97.44 ± 1.07	97.15 ± 2.03	84.32 ± 3.27	84.46 ± 4.81
50	97.52 ± 0.73	98.17 ± 0.94	93.22 ± 2.61	94.66 ± 2.07
80	97.67 ± 1.15	**99.13 ± 0.41**	93.86 ± 1.88	**96.57 ± 1.39**
100	98.76 ± 0.66	99.27 ± 0.52	94.66 ± 2.49	96.42 ± 1.61

(d) Fine-tuned Resnet-50

Train size (%)	Monkey10		Imbalanced-Monkey10	
	OvO	OvA	OvO	OvA
10	92.40 ± 1.75	91.61 ± 1.35	64.15 ± 2.95	63.93 ± 2.96
20	94.53 ± 1.53	94.37 ± 2.24	79.85 ± 1.68	74.17 ± 5.89
50	95.77 ± 0.97	96.79 ± 1.65	89.70 ± 2.44	85.41 ± 4.48
80	97.37 ± 0.64	97.37 ± 1.40	92.55 ± 2.06	91.61 ± 2.67
100	97.66 ± 1.36	97.96 ± 0.48	93.86 ± 1.79	91.69 ± 1.73

classification methods on both the Monkey-10 and the Imbalanced-Monkey-10 dataset.

5.5.6 Results of Training CNNs without Data Augmentation

We trained the two CNNs from scratch and using pre-trained weights combined with the two classification methods on the Agril5::ts100 and Tropic10::ts100 datasets without data augmentation on the training data (again based on five-fold cross-validation). The results obtained during the testing phase are reported in Table 17.

Table 17: Recognition performances (average accuracy and standard deviation) of the studied CNNs combined with the two classification methods applied on the Agril5::ts100 and Tropic10::ts100 datasets. The bold number indicates a significant difference between the classification methods ($p < 0.05$).

Models	AgrilPlant5::ts100		Tropic10::ts100	
	OvO	OvA	OvO	OvA
Scratch-Inception-V3	91.47 ± 1.73	89.33 ± 4.43	94.15 ± 4.28	91.84 ± 5.51
Scratch-Resnet50	$\mathbf{87.60 \pm 1.57}$	83.53 ± 1.80	84.89 ± 0.87	84.40 ± 1.82
Fine-tuned-Inception-V3	93.40 ± 1.64	92.53 ± 2.60	96.50 ± 0.88	95.20 ± 5.04
Fine-tuned-Resnet50	92.53 ± 0.61	91.80 ± 1.79	93.74 ± 1.18	93.53 ± 1.31

The results show that training Scratch-ResNet-50 combined with OvO significantly outperforms the CNN with OvA on the AgrilPlant5::ts100 dataset with a significant difference of $\sim 4\%$. Another observation is that the CNNs combined with OvO always perform a bit better than the CNNs combined with OvA on these two datasets. When we compare these results to the results when data augmentation is used, we can observe that data augmentation leads to performance improvements between 3% and 13%. We also note that especially Scratch-ResNet-50 profits a lot from data augmentation.

5.5.7 Discussion

We now summarize all obtained results when data augmentation is used:

- When training the two CNNs from scratch, the OvO classification method performs significantly better in 37 out of the 100 experiments. In this case, the OvA method never significantly outperforms the OvO method.

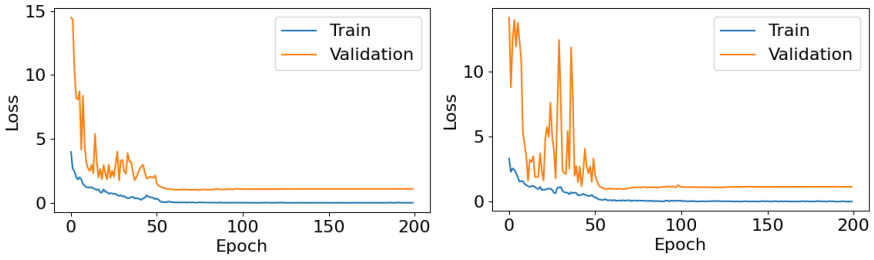
- When training the two pre-trained CNNs by fine-tuning them on the four datasets, the OvA method performs significantly better in 23 out of the 100 experiments. In this case, the OvO method never significantly outperforms the OvA method.
- The improvements of OvO when the CNNs are trained from scratch are larger for smaller datasets. When we examine dataset subsets of 10%, 20%, and 50%, then the OvO scheme performs significantly better in 29 out of 60 experiments. This agrees with the theory stating that the OvO scheme generalizes better than the OvA scheme.

We also observed that the training process is in general more stable with the OvO method compared to the OvA scheme. In Figure 20, we show two train and test loss curves on a small dataset when training ResNet-50 from scratch. The plots clearly show a more stable learning process for OvO, which agrees with the theory that it is beneficial to have output units which are not heavily dependent on each other.

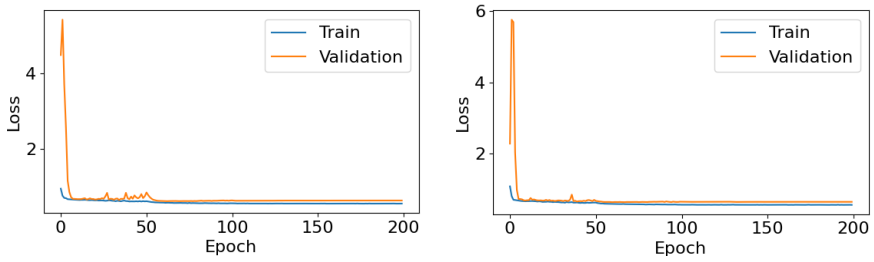
We finally want to mention several last points, which we noticed by analyzing all results. First, the results of using pre-trained weights are in general much better than the results of training the architectures from scratch. This holds for both classification methods, but the differences are much larger for the OvA scheme. Second, the performances of Inception-V3 are overall a bit better than the results of ResNet-50. The best results on the original datasets are excellent and were obtained with the pre-trained Inception-V3 architecture combined with the OvA scheme. The best performance on the AgrilPlant10 dataset is 98.8% (see Table 14a). The best performance on the Tropic20 dataset is 99.7% (see Table 14b). The best result on the Swedish15 dataset is 99.97% (see Table 14c). The best result on the Monkey-10 dataset is 99.3% (see Table 16c).

5.6 Conclusion

We described a novel technique for training deep neural networks based on the One-vs-One classification scheme. Two convolutional neural network



(a) One-vs-All



(b) One-vs-One

Figure 20: Two loss curves when training Scratch-ResNet-50 combined with the classification methods on the AgrilPlant10::ts10 dataset; (a) One-vs-All, and (b) One-vs-One.

architectures were trained using the One-vs-One scheme and the standard One-vs-All scheme on four image datasets with different amounts of examples and classes. The results show that when the deep neural networks are trained from scratch, the proposed method significantly outperforms the conventional One-vs-All training scheme in 37 out of 100 experiments. The results also show that this is not the case when the architectures were fine-tuned, for which the One-vs-All scheme wins in 21 out of 100 experiments. A possible reason why the OvA training scheme performs better with fine-tuning is that the architectures were pre-trained using the One-vs-All scheme on ImageNet. It would be interesting to train One-vs-One architectures on ImageNet and study if this would improve the transfer learning results.

Future work. There are several directions that we want to explore further. First, instead of using the One-vs-One scheme, it would be interesting to generalize our method to the use of error-correcting output codes (Dietterich and Bakiri, 1995). The proposed architecture can also be extended by connecting the One-vs-One outputs to an additional One-vs-All output layer.

Second, although transfer learning is very useful for solving a different image recognition problem, there are also quite different applications involving fMRI images, 3D medical scans, or hyperspectral camera-images. For such pattern recognition problems, almost no pre-trained architectures exist. We would therefore like to research the benefits of using One-vs-One classification for such problems.

Third, we want to study the benefits of using One-vs-One classification when combined with other deep neural networks, such as recurrent neural networks (RNNs). The training process of recurrent neural networks is usually much less stable than when training convolutional neural networks, and it would be interesting to study if the One-vs-One scheme is beneficial for training RNNs.

6

DISCUSSION

During the work in this dissertation, a considerable number of experiments were carried out using deep learning architectures. Three existing plant datasets (LeafSnap, Folio, and Swedish) and three novel plant datasets (AgrilPlant, Five-Tropic-Fruits, and Tropic) were used to deal with different tasks, including plant detection, plant counting, and plant classification. We formulated several objectives and research questions stated in Chapter 1. This chapter provides answers to the research questions and indicates possible directions for future work.

6.1 Answers to the Research Questions

Objective 1: Comparing the traditional feature extractors to the deep learning techniques.
Question 1: Does deep learning outperform hand-crafted features and local descriptors in the plant domain? Can we modify the on-the-shelf CNN architectures so that they achieve better performance on plant classification? Do CNN architectures also work well on small datasets?

To answer these questions, in Chapter 2, we compared the local feature descriptor (HOG with KNN), and a bag of visual words with the histogram of oriented descriptor (HOG-BOW) combined with two classifiers (MLP and SVM) to two CNNs (GoogleNet and AlexNet) on three plant datasets. We trained the two CNN architectures from scratch or pre-trained weights. Additionally, the preliminary experiments suggested that reducing the

number of neurons in the last two final fully-connected layers of AlexNet led to less computational time and better accuracy. Therefore, we modified and used a concise version of AlexNet. The results showed that among the local descriptors, HOG-BOW combined with either MLP or SVM yielded significantly better performance than HOG combined with KNN. When comparing the local feature descriptors to the CNN architectures, the fine-tuned versions of the CNN architectures significantly outperformed the local feature descriptors on all three datasets. On the relatively small dataset (Folio), both the fine-tuned AlexNet and the fine-tuned GoogleNet obtained the best results.

Objective 2: Determine the effectiveness of the combination of data-augmentation techniques for plant classification problems.
Question 2: Does DA help to improve classification performance? If a single DA technique improves recognition accuracy, does the combination of DA techniques work more effectively?

In Chapter 3, we firstly explored and compared the effectiveness of six single DA techniques (rotation, blur, contrast, scaling, illumination, and projective transformation) on three plant datasets using two CNN architectures: GoogleNet and AlexNet. Furthermore, we evaluated the utility of several combinations of these DA techniques, i.e., rotation+blur, rotation+contrast, and rotation+contrast+illumination.

The results suggested that almost all single DA techniques improve classification performance on all studied plant datasets. However, this is not the case for the blur operation on the AgrilPlant dataset, which resulted in worse performance. When considering the combination of the DA techniques, combining DA methods can be of great help to improve performances, especially when training CNNs from scratch. On the other hand, training CNNs with pre-trained weights hardly profits from DA. When training the scratch CNN models, the combinations of rotation and illuminations or rotation and contrasts were most promising to obtain better performances.

Objective 3: Develop a DA technique that helps to improve the fruit counting performance.
Question 3: Can DA techniques enhance the performance of the fruit counting task?

We researched this question in Chapter 4. Data-augmentation techniques generally are applied to the training set to increase the number of training images and modify the image properties by for example rotating, flipping, or scaling images. Previous works have shown that in most cases, DA techniques help to improve classification accuracy.

We proposed a different type of DA technique, Fruit data-augmentation (FDA), and applied it to the fruit dataset. This technique is able to add new fruit masks to the original images and ensures the correct fruits and number of fruits in the images. The FDA technique increases the size of the training set and adds new images containing more fruits to the training set. One of the advantages of FDA is that we do not need to create bounding boxes manually.

We performed two different approaches for fruit counting: a holistic regression-based approach, and a detection-based approach. The performances of these two fruit counting approaches are evaluated by comparing three mean absolute error (MAE) values obtained from the models trained on either the original training set (the O-models), or the augmented training set (the A-models). The three loss-values are considered for different purposes: MAE for counting and classification (MAE_{CC}), MAE for the count-only (MAE_{CO}), and MAE for evaluating misclassifications (MAE_{MC}). The two CNN architectures, ResNet50 and Inception-V3, are used for the regression-based counting approach and the two object-detection architectures, Faster R-CNN and SSD-MobileNet, are used for performing fruit detection of the detection-based counting approach.

The results suggested that the FDA technique has been of great help for the regression-based counting approach. All three loss-values indicated that the A-models performed significantly better than the O-models. However, this is not the case for the detection-based counting approach. The results showed no significant difference between the A-models and

the O-models. This means the FDA method does not aid in improving performance for the detection-based counting approach. This is because the fruit masks which were added to the training images do not have a new appearance; therefore, there is no new information for the detector-based approach introduced to the images. The fruit data-augmentation technique is helpful for the regression-based counting approach.

Objective 4: Combine CNNs with One-vs-One classification to enhance recognition accuracy.
Question 4: Do CNNs combined with the One-vs-One classification scheme outperform the traditional One-vs-All classification scheme?

CNNs are almost always combined with a One-vs-All classification for dealing with object classification problems. In Chapter 5, we described the development of a novel One-vs-One classification scheme for CNNs. For this, we changed the neural network architecture, created a code matrix to convert a one-hot encoding to an OvO label encoding, changed the loss function and output layer, and changed the method to classify the unseen examples. The advantages of the OvO classification scheme compared to the OvA classification scheme are: (a) the OvO scheme generalizes better than the OvA scheme, (b) the OvO scheme could reduce problems with learning instabilities, and (c) the OvO scheme does not create artificial class imbalances.

To evaluate the performance of the proposed method, we compared the OvO and OvA classification schemes by training two CNN architectures (ResNet50 and Inception-V3) on three plant datasets and one fine-grained dataset containing different monkey species. The two CNNs are trained from scratch or pre-trained weights. We performed experiments on different training sizes of images or classes to study the impact of the proposed method on smaller training sets.

The results suggested that when training the CNNs from scratch, the OvO classification scheme outperforms the OvA scheme on all studied datasets. Moreover, the improvements of OvO when training the CNNs from scratch are larger for smaller datasets, which supports that the OvO

scheme has better generalization properties than the OvA scheme. On the other hand, when training CNNs with pre-trained weights, the OvA scheme performed better than the OvO scheme in several cases. The possible reason is that the weights of the fine-tuned models were trained using the OvA classification on the ImageNet dataset. One observation is that training CNNs with the OvO scheme is more stable than the OvA scheme. Training CNNs combined with OvO classification can be beneficial, especially when training from scratch. This is helpful for the application domains where there are no pre-trained models available.

6.2 Future Work

We observed that deep convolutional neural networks have performed remarkably well on plant recognition, plant detection, and counting tasks. This dissertation has also shown that applying various data-augmentation algorithms to the training set helps to improve recognition and counting performances. In addition to augmentation methods, we proposed using neural networks with a novel One-vs-One classification scheme to improve plant classification accuracy.

One of the remaining challenging tasks in plant recognition research is the appearance variation and morphological variation issues. Plants generally are changed by different seasons of the year. They also vary in their forms and structures, which can be seen in the leaves, stems, flowers, or fruits. These issues could be further studied by integrating prior botanical knowledge such as spatial location, plant characteristics, and climate conditions with conditional generative adversarial networks (GANs) (Goodfellow et al., 2014; Mirza and Osindero, 2014).

Moreover, in terms of application aspect, it would be interesting to integrate neural networks with current technologies, such as unmanned aerial vehicles (UAVs), or the internet of things (IoT), and apply them to improve smart farming systems and agriculture science research. We want to examine the following systems further:

- **Disease detection:** We are interested in developing a plant disease detection system. Open field cultivation is exposed to danger with variegated leaves or fruit diseases, such as canker, mosaic, or black or white mold. We intend to improve the Fruit Data Augmentation (FDA) technique, which was discussed in Chapter 4, and apply it to the plant disease detection system. The FDA technique helps to increase the fruit counting performance. One drawback of FDA is that the fruit masks are added freely to the original images, which can create unrealistic objects (apples can be put in the sky). We can enhance the technique by verifying the leaf or fruit in the images and place the disease masks on the correct position. Furthermore, instead of counting the number of objects, we will determine the ratio of sick and healthy plants and decide whether the crops need attention.

- **Autonomous weed control system:** Weed detection remains a challenging problem due to a substantial similarity between weeds and crops. We aim to develop an autonomous weed control system by integrating UAVs to obtain entire field images. In Chapter 5, we discussed that the One-vs-One scheme works more efficiently when fewer classes are classified, and excellent performances with the fine-grained classification are achieved. In most plantations, there are usually a few types of plants, hence we expect that the OvO scheme should work well for weed detection. We can identify the weeds and draw the attention of farmers to problem areas.

- **Crop monitoring system with IoT technology:** We have shown that deep learning algorithms produce an excellent performance for plant classification. We aim to develop a precision farming system by integrating the classification results using deep learning algorithms with vital data gathering from the IoT devices, such as real-time soil moisture, temperature, or light exposure. The system should be able to monitor whether plants are sick, wither, or ready for harvest.

BIBLIOGRAPHY

Allwein, Erin L., Robert E. Schapire, and Yoram Singer (Sept. 2001). "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers." In: *Journal of Machine Learning Research* 1, 113–141.

Alpaydin, Ethem (2014). *Introduction to Machine Learning*. The MIT Press. ISBN: 0262028182, 9780262028189.

Aly, M. (2005). "Survey on multiclass classification methods." In: *Neural networks*, pp. 1–9.

Antoniou, Antreas, Amos Storkey, and Harrison Edwards (2017). "Data augmentation generative adversarial networks." In: *arXiv:1711.04340*.

Arora, Sanjeev, Aditya Bhaskara, Rong Ge, and Tengyu Ma (2014). "Provable bounds for learning some deep representations." In: *International Conference on Machine Learning*, pp. 584–592.

Arteta, Carlos, Victor Lempitsky, and Andrew Zisserman (2016). "Counting in the wild." In: *European conference on computer vision*. Springer, pp. 483–498.

Atabay, Habibollah Agh (2016). "A convolutional neural network with a new architecture applied on leaf classification." In: *The IIOAB Journal* 7, pp. 226–231.

Bama, B Sathya, S Mohana Valli, S Raju, and V Abhai Kumar (2011). "Content based leaf image retrieval (CBLIR) using shape, color and texture features." In: *Indian Journal of Computer Science and Engineering* 2.2, pp. 202–211.

Bargoti, Suchet and James Underwood (2017). "Deep fruit detection in orchards." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3626–3633.

Bertozzi, Massimo, Alberto Broggi, Mike Del Rose, Mirko Felisa, Alain Rakotomamonjy, and Frédéric Suard (2007). "A pedestrian detector using histograms of oriented gradients and a support vector machine classifier." In: *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, pp. 143–148.

Boominathan, Lokesh, Srinivas SS Kruthiventi, and R Venkatesh Babu (2016). "Crowdnet: A deep convolutional network for dense crowd counting." In: *Proceedings of the 24th ACM international conference on Multimedia*. ACM, pp. 640–644.

Brahimi, Mohammed, Marko Arsenovic, Sohaib Laraba, Srdjan Sladojevic, Kamel Boukhalfa, and Abdelouhab Moussaoui (2018). "Deep learning for plant diseases: detection and saliency map visualisation." In: *Human and Machine Learning*. Springer, pp. 93–117.

Bulacu, Marius, Axel Brink, Tijn van der Zant, and Lambert Schomaker (2009). "Recognition of handwritten numerical fields in a large single-writer historical collection." In: *2009 10th International Conference on Document Analysis and Recognition*. IEEE, pp. 808–812.

Castelluccio, Marco, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva (2015). "Land use classification in remote sensing images by convolutional neural networks." In: *arXiv:1508.00092*.

Chih-Wei Hsu and Chih-Jen Lin (2002). "A comparison of methods for multiclass support vector machines." In: *IEEE Transactions on Neural Networks* 13.2, pp. 415–425.

Couchot, Jean-François, Raphaël Couturier, Christophe Guyeux, and Michel Salomon (2016). "Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key." In: *arXiv:1605.07946*.

Cruz, Albert C, Andrea Luvisi, Luigi De Bellis, and Yiannis Ampatzidis (2017). "X-FIDO: an effective application for detecting Olive Quick Decline Syndrome with deep learning and data fusion." In: *Frontiers in plant science* 8, p. 1741.

Csurka, Gabriella, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray (2004). "Visual categorization with bags of keypoints." In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. Prague, pp. 1–2.

Dalal, Navneet and Bill Triggs (2005). "Histograms of oriented gradients for human detection." In: *Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1, pp. 886–893.

Dietterich, Thomas G. and Ghulum Bakiri (1995). "Solving Multiclass Learning Problems via Error-correcting Output Codes." In: *Journal of Artificial Intelligence Research* 2.1, pp. 263–286.

Douarre, Clément, Richard Schielein, Carole Frindel, Stefan Gerth, and David Rousseau (2018). "Transfer Learning from Synthetic Data Applied to Soil–Root Segmentation in X-Ray Tomography Images." In: *Journal of Imaging* 4.5, p. 65.

Du, Ji-Xiang, Xiao-Feng Wang, and Guo-Jun Zhang (2007). "Leaf shape based plant species recognition." In: *Applied mathematics and computation* 185.2, pp. 883–893.

Dwibedi, Debidatta, Ishan Misra, and Martial Hebert (2017). "Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection." In: *The*

IEEE International Conference on Computer Vision (ICCV), pp. 1301–1310.

Dyrmann, Mads, Henrik Karstoft, and Henrik Skov Midtiby (2016). "Plant species classification using deep convolutional neural network." In: *Biosystems Engineering* 151, pp. 72–80.

Farhadi, Ali, Ian Endres, Derek Hoiem, and David Forsyth (2009). "Describing objects by their attributes." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1778–1785.

Ferentinos, Konstantinos P (2018). "Deep learning models for plant disease detection and diagnosis." In: *Computers and Electronics in Agriculture* 145, pp. 311–318.

Fuentes, Alvaro, Sook Yoon, Sang Cheol Kim, and Dong Sun Park (2017). "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition." In: *Sensors* 17.9, p. 2022.

Galar, Mikel, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera (2011). "An Overview of Ensemble Methods for Binary Classifiers in Multi-Class Problems: Experimental Study on One-vs-One and One-vs-All Schemes." In: *Pattern Recognition* 44.8, 1761–1776.

Galar, Mikel, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera (2015). "DRCW-OVO: distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems." In: *Pattern recognition* 48.1, pp. 28–42.

Gaston, Kevin J and Mark A O'Neill (2004). "Automated species identification: why not?" In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 359.1444, pp. 655–667.

Ghazi, Mostafa Mehdipour, Berrin Yanikoglu, and Erchan Aptoula (2017). "Plant Identification Using Deep Neural Networks via Optimization of Transfer Learning Parameters." In: *Neurocomputing*.

Girshick, Ross (2015). "Fast R-CNN." In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2015). "Region-based convolutional networks for accurate object detection and segmentation." In: *IEEE transactions on pattern analysis and machine intelligence* 38.1, pp. 142–158.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier neural networks." In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.

Goëau, Hervé et al. (2013). "Pl@ ntnet mobile app." In: *Proceedings of the 21st ACM international conference on Multimedia*, pp. 423–424.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT press.

Goodfellow, Ian et al. (2014). "Generative adversarial nets." In: *Advances in neural information processing systems*, pp. 2672–2680.

Grinblat, Guillermo L, Lucas C Uzal, Mónica G Larese, and Pablo M Granitto (2016). "Deep learning for plant identification using vein morphological patterns." In: *Computers and Electronics in Agriculture* 127, pp. 418–424.

Guo, Yanming, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew (2016). "Deep learning for visual understanding: A review." In: *Neurocomputing* 187, pp. 27–48.

Guru, DS, YH Sharath, and S Manjunath (2010). "Texture features and KNN in classification of flower images." In: *IJCA, Special Issue on RTIPPR (1)*, pp. 21–29.

Guyer, D Eetal, GE Miles, MM Schreiber, OR Mitchell, and VC Vanderbilt (1986). "Machine vision and image processing for plant identification." In: *Transactions of the ASAE* 29.6, pp. 1500–1507.

Guyer, DE, GE Miles, LD Gaultney, and MM Schreiber (1993). "Application of machine vision to shape analysis in leaf and plant identification." In: *Transactions of the ASAE (USA)*.

Häni, Nicolai, Pravakar Roy, and Volkan Isler (2019). "A comparative study of fruit detection and counting methods for yield mapping in apple orchards." In: *Journal of Field Robotics*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Sheng and Lambert Schomaker (2018). "Open Set Chinese Character Recognition using Multi-typed Attributes." In: *arXiv:1808.08993*.

Howard, Andrew G et al. (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." In: *arXiv:1704.04861*.

Hsiao, Jou-Ken, Li-Wei Kang, Ching-Long Chang, and Chih-Yang Lin (2014). "Comparative study of leaf image recognition with a novel learning-based approach." In: *Science and Information Conference (SAI), 2014*. IEEE, pp. 389–393.

Huang, Jonathan et al. (2017). "Speed/accuracy trade-offs for modern convolutional object detectors." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7310–7311.

Kadir, Abdul, Lukito Edi Nugroho, Adhi Susanto, and Paulus Insap Santosa (2011). "Neural network application on foliage plant identification." In: *International Journal of Computer Applications* 29.9, pp. 15–22.

Koirala, A, KB Walsh, Z Wang, and C McCarthy (2019). "Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'." In: *Precision Agriculture*, pp. 1–29.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*, pp. 1097–1105.

Kumar, Neeraj et al. (2012). "Leafsnap: A computer vision system for automatic plant species identification." In: *European Conference on Computer Vision*. Springer, pp. 502–516.

Kumar, Shailesh, Joydeep Ghosh, and Melba M. Crawford (2002). "Hierarchical Fusion of Multiple Classifiers for Hyperspectral Data Analysis." In: *Pattern Analysis and Applications* 5, pp. 210–220.

Latte, M, Sushila Shidnal, B Anami, and V Kuligod (2015). "A combined color and texture features based methodology for recognition of crop field image." In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 8.2, pp. 287–302.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning." In: *Nature* 521.7553, pp. 436–444.

LeCun, Yann et al. (1989). "Backpropagation applied to handwritten zip code recognition." In: *Neural computation* 1.4, pp. 541–551.

Lee, Sue Han, Chee Seng Chan, Paul Wilkin, and Paolo Remagnino (2015). "Deep-plant: Plant identification with convolutional neural networks."

In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 452–456.

Lee, Sue Han, Yang Loong Chang, Chee Seng Chan, and Paolo Remagnino (2016). "Plant identification system based on a convolutional neural network for the lifeclef 2016 plant classification task." In: *Working notes of CLEF 2016 conference*.

Lempitsky, Victor and Andrew Zisserman (2010). "Learning to count objects in images." In: *Advances in neural information processing systems*, pp. 1324–1332.

Lin, Min, Qiang Chen, and Shuicheng Yan (2013). "Network in network." In: *arXiv:1312.4400*.

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common objects in context." In: *European conference on computer vision*. Springer, pp. 740–755.

Liu, Wei et al. (2016). "SSD: Single shot multibox detector." In: *European conference on computer vision*. Springer, pp. 21–37.

Liu, Xu et al. (2018). "Robust fruit counting: Combining deep learning, tracking, and structure from motion." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1045–1052.

Liu, Yang, Jian-Wu Bi, and Zhi-Ping Fan (2017). "A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm." In: *Information Sciences* 394, pp. 38–52.

Lottes, Philipp, Jens Behley, Andres Milioto, and Cyrill Stachniss (2018). "Fully convolutional networks with sequential information for robust

crop and weed detection in precision farming." In: *IEEE Robotics and Automation Letters* 3.4, pp. 2870–2877.

McFee, Brian, Eric J Humphrey, and Juan Pablo Bello (2015). "A Software Framework for Musical Data Augmentation." In: *ISMIR*, pp. 248–254.

Meyer, George E, Timothy W Hindman, and Koppolu Laksmi (1999). "Machine vision detection parameters for plant species identification." In: *Precision agriculture and biological quality*. Vol. 3543. International Society for Optics and Photonics, pp. 327–335.

Mirza, Mehdi and Simon Osindero (2014). "Conditional generative adversarial nets." In: *arXiv preprint arXiv:1411.1784*.

Mohanty, Sharada P, David P Hughes, and Marcel Salathé (2016). "Using deep learning for image-based plant disease detection." In: *Frontiers in plant science* 7, p. 1419.

Møller, Martin Fodslette (1993). "A scaled conjugate gradient algorithm for fast supervised learning." In: *Neural networks* 6.4, pp. 525–533.

Mouine, Sofiene, Itheri Yahiaoui, and Anne Verroust-Blondet (2013). "A shape-based approach for leaf classification using multiscale-triangular representation." In: *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*. ACM, pp. 127–134.

Munisami, Trishen, Mahess Ramsurn, Somveer Kishnah, and Sameerchand Pudaruth (2015). "Plant leaf recognition using shape features and colour histogram with K-nearest neighbour classifiers." In: *Procedia Computer Science* 58, pp. 740–747.

Neto, João Camargo, George E Meyer, David D Jones, and Ashok K Samal (2006). "Plant species identification using Elliptic Fourier leaf shape

analysis." In: *Computers and electronics in agriculture* 50.2, pp. 121–134.

Nilsback, Maria-Elena and Andrew Zisserman (2008). "Automated flower classification over a large number of classes." In: *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE, pp. 722–729.

— (2010). "Delving deeper into the whorl of flower segmentation." In: *Image and Vision Computing* 28.6, pp. 1049–1062.

Okafor, E., R. Smit, L.R.B. Schomaker, and M.A. Wiering (2017). "Operational Data Augmentation in Classifying Single Aerial Images of Animals." In: *Proceedings of the IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*.

Onoro-Rubio, Daniel and Roberto J López-Sastre (2016). "Towards perspective-free object counting with deep learning." In: *European Conference on Computer Vision*. Springer, pp. 615–629.

Ou, Guobin and Yi Lu Murphey (2007). "Multi-Class Pattern Classification Using Neural Networks." In: *Pattern Recognition* 40.1, 4–18.

Paul Cohen, Joseph, Genevieve Boucher, Craig A Glastonbury, Henry Z Lo, and Yoshua Bengio (2017). "Count-ception: Counting by fully convolutional redundant counting." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 18–26.

Pawara, Pornntiwa, Emmanuel Okafor, Lambert Schomaker, and Marco Wiering (2017a). "Data augmentation for plant classification." In: *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, pp. 615–626.

Pawara, Pornntiwa, Emmanuel Okafor, Olarik Surinta, Lambert Schomaker, and Marco Wiering (2017b). "Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition." In: *ICPRAM*, pp. 479–486.

Perez, Luis and Jason Wang (2017). "The effectiveness of data augmentation in image classification using deep learning." In: *arXiv:1712.04621*.

Rahnemoonfar, Maryam and Clay Sheppard (2017). "Deep count: fruit counting based on deep simulated learning." In: *Sensors* 17.4, p. 905.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks." In: *Advances in neural information processing systems*, pp. 91–99.

Rifkin, Ryan and Aldebaro Klautau (2004). "In defense of one-vs-all classification." In: *Journal of machine learning research* 5.Jan, pp. 101–141.

Rocha, Anderson and Siome Klein Goldenstein (2014). "Multiclass from binary: Expanding one-versus-all, one-versus-one and ECOC-based approaches." In: *IEEE Transactions on Neural Networks and Learning Systems* 25.2, pp. 289–302.

Rzanny, Michael, Marco Seeland, Jana Wäldchen, and Patrick Mäder (2017). "Acquiring and preprocessing leaf images for automated plant identification: understanding the tradeoff between effort and information gain." In: *Plant methods* 13.1, pp. 1–11.

Salamon, Justin and Juan Pablo Bello (2017). "Deep convolutional neural networks and data augmentation for environmental sound classification." In: *IEEE Signal Processing Letters* 24.3, pp. 279–283.

Santos Ferreira, Alessandro dos, Daniel Matte Freitas, Gercina Gonçalves da Silva, Hemerson Pistori, and Marcelo Theophilo Folhes (2017). "Weed detection in soybean crops using ConvNets." In: *Computers and Electronics in Agriculture* 143, pp. 314–324.

Sato, Ikuro, Hiroki Nishimura, and Kensuke Yokoi (2015). "Apac: Augmented pattern classification with neural networks." In: *arXiv:1505.03229*.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview." In: *Neural networks* 61, pp. 85–117.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition." In: *arXiv:1409.1556*.

Sladojevic, Srdjan, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic (2016). "Deep neural networks based recognition of plant diseases by leaf image classification." In: *Computational intelligence and neuroscience* 2016.

Söderkvist, Oskar (2001). *Computer vision classification of leaves from swedish trees*.

Songsiri, Patoomsiri, Vladimir Cherkassky, and Boonserm Kijsirikul (2018). "Universum Selection for Boosting the Performance of Multiclass Support Vector Machines Based on One-versus-One Strategy." In: *Knowledge-Based Systems* 159, pp. 9–19.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting." In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Stahl, Tobias, Silvia L Pintea, and Jan C van Gemert (2018). "Divide and count: Generic object counting by image divisions." In: *IEEE Transactions on Image Processing* 28.2, pp. 1035–1044.

Sun, Yu, Yuan Liu, Guan Wang, and Haiyan Zhang (2017). "Deep learning for plant identification in natural environment." In: *Computational intelligence and neuroscience* 2017.

Surinta, Olarik, Mahir F Karaaba, Tusar K Mishra, Lambert RB Schomaker, and Marco A Wiering (2015). "Recognizing handwritten characters with local descriptors and bags of visual words." In: *International Conference on Engineering Applications of Neural Networks*. Springer, pp. 255–264.

Suykens, Johan AK and Joos Vandewalle (1999). "Least squares support vector machine classifiers." In: *Neural processing letters* 9.3, pp. 293–300.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016). "Rethinking the inception architecture for computer vision." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Szegedy, Christian et al. (2015). "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Tao Ban and S. Abe (2006). "Implementing Multi-class Classifiers by One-class Classification Methods." In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 327–332.

Tax, David Martinus Johannes (2001). "One-class classification: Concept learning in the absence of counter-examples." PhD thesis. Technische Universiteit Delft.

Taylor, Luke and Geoff Nitschke (2017). "Improving deep learning using generic data augmentation." In: *arXiv:1708.06020*.

Too, Edna Chebet, Li Yujian, Sam Njuki, and Liu Yingchun (2019). "A comparative study of fine-tuning deep learning models for plant disease identification." In: *Computers and Electronics in Agriculture* 161, pp. 272–279.

Tremblay, Jonathan et al. (2018). "Training deep networks with synthetic data: Bridging the reality gap by domain randomization." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977.

Tsai, Chih-Fong (2012). "Bag-of-words representation in image annotation: A review." In: *ISRN Artificial Intelligence* 2012.

Ubbens, Jordan R and Ian Stavness (2017). "Deep plant phenomics: a deep learning platform for complex plant phenotyping tasks." In: *Frontiers in plant science* 8, p. 1190.

Ubbens, Jordan, Mikolaj Cieslak, Przemyslaw Prusinkiewicz, and Ian Stavness (2018). "The use of plant models in deep learning: an application to leaf counting in Rosette plants." In: *Plant methods* 14.1, p. 6.

VijayaLakshmi, Balasubramanian and Vasudev Mohan (2016). "Kernel-based PSO and FRVM: An automatic plant leaf type detection using texture, shape, and color features." In: *Computers and Electronics in Agriculture* 125, pp. 99–112.

Vural, Volkan and Jennifer G. Dy (2004). "A Hierarchical Method for Multi-Class Support Vector Machines." In: *Proceedings of the Twenty-First International Conference on Machine Learning*, pp. 105–113.

Wäldchen, Jana and Patrick Mäder (2018). "Plant species identification using computer vision techniques: A systematic literature review." In: *Archives of Computational Methods in Engineering* 25.2, pp. 507–543.

Wäldchen, Jana, Michael Rzanny, Marco Seeland, and Patrick Mäder (2018). "Automated plant species identification—Trends and future directions." In: *PLoS computational biology* 14.4.

Wang, Xingxing, LiMin Wang, and Yu Qiao (2012). "A comparative study of encoding, pooling and normalization methods for action recognition." In: *Asian Conference on Computer Vision*. Springer, pp. 572–585.

Wang, Xuan, Junhua Liang, and Fangxia Guo (2014). "Feature extraction algorithm based on dual-scale decomposition and local binary descriptors for plant leaf recognition." In: *Digital Signal Processing* 34, pp. 101–107.

Wang, Zhaobin et al. (2014). "Plant recognition based on intersecting cortical model." In: *2014 International joint conference on neural networks (IJCNN)*. IEEE, pp. 975–980.

Wang, Zhiyong, Bin Lu, Zheru Chi, and Dagan Feng (2011). "Leaf image classification with shape context and sift descriptors." In: *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*. IEEE, pp. 650–654.

Woebbecke, DM, GE Meyer, K Von Bargen, and DA Mortensen (1995). "Shape features for identifying young weeds using image analysis." In: *Transactions of the ASAE* 38.1, pp. 271–281.

Xiao, Xue-Yang, Rongxiang Hu, Shan-Wen Zhang, and Xiao-Feng Wang (2010). "HOG-based approach for leaf classification." In: *International Conference on Intelligent Computing*. Springer, pp. 149–155.

Xing, Linjie and Yu Qiao (2016). "Deepwriter: A multi-stream deep CNN for text-independent writer identification." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, pp. 584–589.

Yoo, Hyeon-Joong (2015). "Deep convolution neural networks in computer vision: a review." In: *IEIE Transactions on Smart Processing & Computing* 4.1, pp. 35–43.

Zhang, Chaoyun, Pan Zhou, Chenghua Li, and Lijun Liu (2015a). "A convolutional neural network for leaves recognition using data augmentation." In: *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference*, pp. 2143–2150.

Zhang, Cong, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang (2015b). "Cross-scene crowd counting via deep convolutional neural networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 833–841.

Zhang, Shanwen, Yingke Lei, Chuanlei Zhang, and Yihua Hu (2016). "Semi-supervised orthogonal discriminant projection for plant leaf classification." In: *Pattern Analysis and Applications* 19.4, pp. 953–961.

Zhang, Zhong-Liang, Xing-Gang Luo, Salvador García, Jia-Fu Tang, and Francisco Herrera (2017). "Exploring the effectiveness of dynamic ensemble selection in the one-versus-one scheme." In: *Knowledge-Based Systems* 125, pp. 53–63.

Zhao, Cong, Sharon SF Chan, Wai-Kuen Cham, and LM Chu (2015). "Plant identification using leaf shapes—A pattern counting approach." In: *Pattern Recognition* 48.10, pp. 3203–3215.

SUMMARY

This dissertation emphasizes using deep convolutional neural networks to deal with three problems: plant recognition, detection, and counting. The dissertation can be summarized into six chapters, as follows:

Chapter 1 introduces a brief motivation for plant recognition, detection, and counting research. Furthermore, the objective and research questions are provided.

Chapter 2 aims to develop a baseline of the plant recognition systems. We compare the scratch and fine-tuned versions of two deep convolutional neural network architectures (AlexNet and GoogleNet) to several local feature descriptors or bags of visual words combined with different classifiers: a histogram of oriented gradients (HOG) with k-nearest neighbors (KNN), a bag of visual words with the histogram of oriented gradients (HOG-BOW) with multilayer perceptrons (MLP), and HOG-BOW with a support vector machine (SVM). Additionally, AlexNet is customized by reducing the number of neurons to 1,024 neurons to reduce computational time. In total, seven methods are used for performing plant classification on three plant datasets, in which one of these is a relatively small dataset. The results suggest that among the classical feature descriptors, HOG-BOW combined with either MLP or SVM performs better than HOG with KNN. When comparing the deep convolutional neural networks (CNNs) to the local feature descriptor methods, the fine-tuned CNNs achieve notable performance and significantly outperform the classical feature descriptors. These methods also work best on the relatively small dataset.

Chapter 3 examines the benefits of data augmentation techniques for plant classification. We evaluate six data augmentation techniques (rotation, blur, contrast, scaling, illumination, and projective transformation) and six

combinations of these techniques, resulting in a total of 12 data augmentation schemes. We train two CNN architectures (AlexNet and GoogleNet) from scratch or pre-trained weights, on three plant datasets. The results show that five out of six individual data augmentation techniques help to improve classification performance. Among these techniques, only the blur operation does not help and results in worse accuracy on one of the studied datasets. The combination of rotation+illumination or rotation+contrast helps most to get the best performance. Additionally, when training the CNNs on the augmented sets, the scratch-AlexNet architecture profits most from data augmentation with an average of 4.6% improvement compared to training on the original image sets. On the other hand, the fine-tuned CNN models hardly profit from data augmentation. The possible reason is that the fine-tuned CNN models already obtained great performances, so performing data augmentation on the training sets does not benefit much.

Chapter 4 proposes a Fruit-data-augmentation (FDA) technique for the fruit counting problem. This technique is beneficial for adding the number of fruits to the images and increasing the number of images to the training set. To evaluate the benefit of the FDA method, we train two approaches for fruit counting, a holistic regression-based approach, and a detection-based approach. For the regression-based approach, ResNet50 and Inception-V3 are used; for the detection-based approach, Faster R-CNN and SSD-MobileNet are trained for the fruit detection, afterwards the counting task. For each of these two approaches, we evaluate the counting performance by comparing three loss-values obtained from the models trained on the original training set (the O-models) and the models trained on the augmented training set (the A-models). The results suggest that the FDA method has been of great help for all the regression-based approaches, whereas the detection-based approaches do not profit from the FDA method.

Chapter 5 introduces a novel One-vs-One (OvO) classification scheme for CNNs. CNNs almost always employ a One-vs-All (OvA) classification

scheme for performing multi-class classification. The proposed OvO approach changes the neural network architecture and trains each output unit to classify between a specific pair of classes. Additionally, this scheme changes the loss function, uses a code matrix to generate an OvO label encoding for each input class, and changes the method for classifying images. We analyze the advantages of using the OvO scheme for CNNs by training two CNN architectures (ResNet50 and Inception-V3) combined with either the OvO or OvA classification schemes. The two CNN architectures are trained from scratch or pre-trained weights on three plant datasets and one fine-grained dataset of ten monkey species. In addition to the experiments on the whole training sets, we perform experiments on the subsets of all datasets to analyze the benefits of the proposed method on the smaller training sets. The results show that when CNNs are trained from scratch, the OvO scheme significantly outperforms the OvA scheme on all datasets. Moreover, the improvement of the proposed method is larger when training the CNNs on the smaller datasets. However, this is not the case when training CNNs with pre-trained weights. This can be explained by the fact that the pre-trained models were obtained from training with the OvA scheme on the ImageNet dataset.

Chapter 6 provides the answers to the research questions, and suggests several directions for future work.

SAMENVATTING

Dit proefschrift legt de nadruk op het gebruik van diep-convolutionaire neurale netwerken om drie problemen aan te pakken: het herkennen, detecteren en tellen van planten. Het proefschrift wordt beschreven in zes hoofdstukken, te weten:

Hoofdstuk 1 introduceert een korte motivatie voor onderzoeken naar het herkennen, detecteren en tellen van planten. Daarnaast worden de doelen en onderzoeksvragen uiteengezet.

Hoofdstuk 2 heeft als doel om een maatstaf voor plantenherkenningssystemen te ontwikkelen. We vergelijken de standaard versies en de geoptimaliseerde versies van twee architecturen voor diep-convolutionele neurale netwerken (AlexNet en GoogleNet) met de lokale-kenmerk beschrijvingen of de zakken-met-visuele-woorden gecombineerd met verschillende klassificatiemethodes (HOG met KNN, HOG-BOW met MLP, en HOG-BOW met SVM). Om de rekentijd te verminderen is AlexNet bovendien aangepast door het aantal neuronen terug te brengen tot 1.024 neuronen. In totaal worden er zeven methoden gebruikt om planten van drie datasets te klassificeren, waarvan één een relatief kleine dataset is. De resultaten laten zien dat van de klassieke kenmerkbeschrijvingen, HOG-BOW in combinatie met MLP of SVM beter presteert dan HOG met KNN. Wanneer we de CNN-architecturen vergelijken met de lokale-kenmerkbeschrijvingen, presteren de geoptimaliseerde CNN opmerkelijk goed, en aanzienlijk beter dan de klassieke kenmerkbeschrijvingen. Deze methoden werken ook het beste op de relatief kleine dataset.

Hoofdstuk 3 onderzoekt de voordelen van verschillende methodes voor databehandeling voor de klassificatie van planten. We hebben zes data behandelingstechnieken (rotatie, vervaging, contrast, schaling, belichting en

projectieve transformatie) en zes combinaties van deze technieken geëvalueerd, wat heeft geleid tot een totaal van 12 varianten van databehandeling. Hiervoor hebben we twee CNN-architecturen (AlexNet en GoogleNet) getraind, vanaf niets of vanaf voor-getrainde gewichten, op drie plantdatasets. De resultaten laten zien dat vijf van de zes databehandelingstechnieken de klassificatieprestaties verbeteren. Van deze technieken helpt alleen vervaging weinig, en leidt tot een verminderde nauwkeurigheid op één van de datasets. De combinatie rotatie en belichting, en de combinatie rotatie en contrast helpen het meest om de prestaties te verbeteren. Bovendien profiteert de AlexNet architectuur, getraind vanuit niets, het meest van trainen met de aangepaste datasets, met een verbetering van gemiddeld 4,6% ten opzichte van trainen met de originele dataset. De geoptimaliseerde CNN-modellen profiteren daarentegen nauwelijks van de databehandeling. Een mogelijke reden hiervoor is dat de geoptimaliseerde CNN-modellen al hoge prestaties bereiken, zodat het uitvoeren van databehandeling op de trainingssets weinig oplevert.

Hoofdstuk 4 presenteert een FDA-techniek (Fruit-data-behandeling) voor het probleem van het tellen van fruit. Deze techniek is gunstig voor het toevoegen van het aantal vruchten aan het beeldmateriaal en het verhogen van het aantal beelden in de trainings set. Om de toegevoegde waarde van de FDA-methode te evalueren, trainen we twee benaderingen voor het tellen van fruit: een holistische aanpak gebaseerd op regressie, en een aanpak gebaseerd op detectie. Voor de aanpak gebaseerd op regressie worden ResNet50 en Inception-V3 gebruikt; voor de aanpak gebaseerd op de detectie worden Faster R-CNN en SSD-MobileNet getraind om fruit te detecteren ten behoeve van de telling. Voor beide benaderingen evalueren we de telprestaties door drie verlieswaarden te vergelijken, die zijn verkregen door de modellen te trainen op de oorspronkelijke trainingsdata (de O-modellen) en door de modellen te trainen met de aangepaste trainingsdata (de A-modellen). De resultaten laten zien dat de benaderingen gebaseerd op regressie voordeel behalen uit de FDA-methode, terwijl de benaderingen gebaseerd op detectie niet profiteren van de FDA-methode.

Hoofdstuk 5 introduceert een nieuwe One-vs-One (OvO) klassificatiemethode voor CNN's. Over het algemeen maken CNN's gebruik van One-vs-All (OvA) klassificatie voor het klassificeren in drie of meer klassen. De voorgestelde OvO-methode verandert de neurale-netwerkarchitectuur door elke uitgangspunt te trainen om onderscheid tussen twee specifieke klassen. Daarnaast verandert dit schema de verliesfunctie, maakt het gebruik van een codematrix om een OvO-label te genereren voor elke inputklasse, en wijzigt het de methode voor het klassificeren van beelden. We analyseren de voordelen van het gebruik van het OvO-schema voor CNN's door twee CNN-architecturen (ResNet50 en Inception-V3) te trainen met de OvO- en de OvA-klassificatiemethode. De twee CNN-architecturen worden vanuit het niets getraind, of maken gebruik van vooraf getrainde gewichten gebaseerd op drie plantdatasets en een uitgebreide dataset van tien apensoorten. Naast de experimenten op de gehele trainingssets, hebben we ook experimenten uitgevoerd op deelverzamelingen van alle bestudeerde datasets om de voordelen van de voorgestelde methode te analyseren op kleinere trainingsets. De resultaten tonen aan dat wanneer CNN's vanaf niets worden getraind, de OvO-methode op alle bestudeerde datasets significant beter presteert dan de OvA-methode. Bovendien is het voordeel van de voorgestelde methode groter wanneer de CNN's op kleinere datasets worden getraind. Dit is echter niet het geval bij het trainen van CNN's met vooraf getrainde gewichten. Dit kan worden verklaard door het feit dat de vooraf getrainde modellen zijn getraind met de OvA-methode op de ImageNet-dataset.

Hoofdstuk 6 geeft de antwoorden op de onderzoeksvragen, en presenteert een discussie en mogelijk toekomstig onderzoek.

Thesis Publications

Pawara, P.,Okafor, E., Surinta, O., Schomaker, L.R.B., and Wiering, M.A. (2017). *Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition*. International Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 479-486.

Pawara, P.,Okafor, E., Schomaker, L.R.B., and Wiering, M.A. (2017). *Data augmentation for plant classification*. International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), pages 615-626.

Pawara, P., Boshchenko, A., Schomaker, L.R.B., and Wiering, M.A. (2020). *Deep Learning with Data Augmentation for Fruit Counting*. International Conference on Artificial Intelligence and Soft Computing (ICAISC).

Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L.R.B., and Wiering, M.A. (2020). *One-vs-One Classification for Deep Neural Networks*, Pattern Recognition (PR), Elsevier.

Other Publication

Okafor, E., **Pawara, P.**, Karaaba. F, Surinta, O., Codreanu. V, Schomaker, L.R.B., and Wiering, M.A. (2016). *Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition*. IEEE Symposium Series on Computational Intelligence (SSCI), pages 1-8.

ACKNOWLEDGEMENTS

Throughout my PhD, I have met many people and received a great deal of support and assistance.

I would like to acknowledge the Royal Thai government for funding through the National Science and Technology Development Agency (NSTDA) that supported my PhD.

I would like to express my sincere gratitude to my supervisors Dr. Marco Wiering and Prof. Lambert Schomaker, whose expertise are invaluable in formulating the research questions and methodology. Marco, thank for your encouragement, motivation, and insightful feedback that pushed me to a higher level. When I was uncertain, you told me that we are not trying to build a rocket, that truly guided my way. I would also like to extend my greatest gratitude to Lambert for your excellent guidance, support, and immense knowledge. I am grateful for all the supports you have given me.

I would like to thank my assessment committee: Prof. P. Remagnino, Prof. T.M. Heskes, and Prof. D. Karastoyanova for their valuable time in assessing my thesis.

My dearest friend, Artem, thanks for the destiny that we started the first step of our PhD at the same time and that we have the same sense of humor. Doing PhD could be tough but spending time with you throughout all these years made it so joyful (and yet, full of tears). Our love lasts to Pluto and back!

I should also like to express my appreciation to Sheng, Emmanuel, and Joe for sharing the idea and constructive discussion. Additionally, Sheng, I miss playing Gomoku with you. Thank Maruf for a lovely friendship. You always care and make things much easier. I enjoy discussing with you any topics and feel relieved to have you around.

I thank Anja, Pieter, Lorenzo, Stipe, Sha, Aniket, and Vishal for the rides, sports, and activities. For Yifei, we are friends till the end! Cheers to Burcu, Anna, Trudy, and Harmanes for friendship since the beginning. Special

thanks to Harmanes for translating the Dutch summary for this thesis. To all my friends at AI, I am happy to have met you all.

I have much appreciated P' Jane, Jacob, and Jason for a warm welcome to your family. Thank you for Thai food and make me feel like home. I am delighted to meet with Duenpen, Naim, and Thai friends in Groningen. Thank Duenpen and Pat for taking me to the hospital in the very first month in Groningen. Deeply thank Ve-Run, aka VARP, for always caring and supporting me in any circumstance.

I would sincerely like to thank all the secretaries of the Bernoulli Office for being warm and supportive, especially Elina, Sarah, and Jan for their endless assistance. You all always had kind talk every time I dropped by. Thank Remco for fixing stuff when I was the first one with specific problems. I do appreciate all your support.