An Intelligent System for Coffee Grading and Disease Identification

Serawork Wallelign

THESE DE DOCTORAT DE

L'ECOLE NATIONALE D'INGENIEURS DE BREST

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Serawork Amsalu WALLELIGN

An Intelligent System for Coffee Grading and Disease Identification

Thèse présentée et soutenue à Plouzané le 04 Février 2020
Unité de recherche : Lab-STICC, UMR CNRS 6285
Thèse N° : 4

Rapporteurs avant soutenance :

Jean-Michel LOUBES	Professeur, Université Paul Sabatier,Toulouse
Dominique VAUFREYDAZ	Maître de conférences HDR, Université Grenoble Alpes/INRIA, Montbonnot

Composition du Jury :

Président :	Patrick MEYER	Professeur, IMT Atlantique, Plouzané
Examinateurs :	Jean-Michel LOUBES	Professeur, Université Paul Sabatier,Toulouse
	Dominique VAUFREYDAZ	Maître de conférences HDR, Université Grenoble Alpes/INRIA, Montbonnot
	Pascal REDOU	Maître de conférence HDR, ENIB Plouzané
	Mihai POLCEANU	Maître de conférence, University Greenwich (U.K.)
Dir. de thèse :	Cédric BUCHE	Professeur, ENIB Plouzané.

ABSTRACT

One of the key factors in the success of deep learning models is their ability to learn important representations from the input data automatically, without the need for human experts designing task specific features. However, to learn these representations, deep learning methods usually require large amounts of data, which are expensive to obtain especially because of the efforts required for gathering and labeling data. This thesis investigates the applicability of deep learning for real world situations where data exist in small amounts, collected at different locations (labs), using different acquisition techniques and with minimally controlled conditions. As a use case, Ethiopia's coffee plant disease identification and grading is automated using deep learning techniques. Part of the dataset for disease identification is captured on the farm while some of it is downloaded from the internet. By first training the model using a similar plant dataset and then applying transfer learning, it was possible to design a model that classifies coffee plant diseases with test accuracy of 90.18% using only 562 images of coffee plant.

The dataset for coffee beans grading is collected at the Jimma branch of Ethiopia's Commodity Exchange (ECX) in two rounds. To solve the dataset shift that occurred in the two sets because of illumination and camera differences, image preprocessing and augmentation techniques were proposed. However, the proposed techniques did not improve the performance of the model when compared to the commonly used preprocessing methods. By careful designing of the architecture, applying machine learning techniques like data augmentation and ensemble learning, it was possible to obtain a good classifier (89.1% accuracy on the test dataset) that grades coffee beans, performing better than classical machine learning approaches and off-the-shelf deep learning models. The coffee beans dataset and the models will be made publicly available to support further research on these important topics.

Key Words: Deep Learning, Convolutional neural network, Ensemble learning, Small datasets, Coffee

RÉSUMÉ

L'un des facteurs clés du succès des modèles d'apprentissage profond est leur capacité d'apprendre automatiquement des représentations importantes à partir des données d'entrée, sans qu'il soit nécessaire que des experts humains conçoivent des caractéristiques spécifiques aux tâches. Cependant, pour apprendre ces représentations, les méthodes d'apprentissage en profondeur nécessitent généralement de grandes quantités de données, qui sont coûteuses à obtenir, surtout en raison des efforts requis pour recueillir et étiqueter les données. Cette thèse examine l'applicabilité de l'apprentissage en profondeur à des situations du monde réel où les données existent en petites quantités, recueillies à différents endroits (laboratoires), en utilisant différentes techniques d'acquisition et dans des conditions minimales contrôlées. Il y a deux contributions principales. Tout d'abord, nous nous intéressons au problème de la détection des maladies du caféier, qui jusqu'à présent n'était pas abordé dans la littérature, en utilisant un ensemble de données contenant des images de maladies du caféier téléchargées sur Internet et que nous avons capturées en milieu naturel, et une approche transfert - apprentissage. Il a été possible de concevoir un modèle qui classifie les maladies du caféier avec une précision de 90,18 % en utilisant seulement 562 images du caféier.

Deuxièmement, nous nous attaquons au problème du classement des grains de café. Un ensemble de données pour le classement des grains de café est créé en capturant des images de grains de café à la branche Jimma du Commodity Exchange (ECX) de l'Éthiopie en deux séries. Afin de tenter de résoudre le décalage de l'ensemble de données qui s'est produit dans les deux ensembles en raison des différences d'éclairage et de caméra, des algorithmes de correction des couleurs ont été ajoutés au pipeline existant de traitement des images et d'augmentation des données. Toutefois, ces techniques n'ont pas amélioré les performances du modèle par rapport aux méthodes de prétraitement couramment utilisées. Cela indique que la différence dans les techniques d'acquisition d'images n'était pas la seule raison du décalage de l'ensemble de données. Nous avons proposé et évalué une architecture de réseau qui, combinée à des techniques d'augmentation des données et d'apprentissage d'ensemble, a mené à un classificateur amélioré (précision de

89,1 % sur l'ensemble des données d'essai) qui évalue les grains de café et qui est plus performant que les méthodes classiques d'apprentissage machine (amélioration de 25,47 %) et que les modèles prêts-à-servir (18 %). L'ensemble de données sur les grains de café et les modèles seront mis à la disposition du public pour appuyer la poursuite des recherches sur ces sujets importants.

Mots clés : Apprentissage profond, Réseau neurones convolutionnels, Apprentissage d'ensemble, Ensemble de données, Café.

ACKNOWLEDGEMENTS

The journey of this PhD has been a life changing experience for me and it would not have been possible without the support and guidance of so many people and organizations.

First I would like to express my deepest gratitude for my sponsors the French government and the Ethiopian Ministry of Science and Higher Education (MoSHE) who funded the PhD through higher education capacity building program. I extend my thanks to the Ethiopian commodity exchange office, employees of Jimma branch of ECX and coffee research institute of Jimma University and the farmers for their willingness and support during data collection.

I would like to use this opportunity to express my sincere appreciation to my supervisor, Professor Cédric Buche for accepting me as his student and opening the door for this great opportunity. His mentorship and encouragement allowed me to grow as a research scientist. I am also grateful for the support I got from Dr. Mihai Polceanu. It was the many discussions I have had with him that majorly defined the path of the work. His advice on the research and my carrier have been invaluable.

I would also like to thank my committee members Professor Patrick Meyer, Dr. Pascal Redou and Dr. Towfik Ali for their comments and suggestions. I am thankful for all current and former members of CERV especially Andreea-Oana Petac, Dr. Cindy Even and Dr. Mihai Polceanu for the support you have given me since I first arrived in France in 2016. You made the adventure less frustrating and fun, thank you.

Finally, I am forever grateful for my families, friends and relatives who believed in my potential and encouraged me to work hard and pursue my dream. It is your love and support that helped me to stay strong and pass those frustrating times. A special thanks goes to my daughter, Eliana, without her permission I would not have come to France and started the PhD. Thank you for being brave and patient for the times I was absent both physically and emotionally.

CONTENTS

List of Figures

List of Tables

ACRONYMS

Adam: Adaptive Moment Estimation 19, 58, 82
AI: Artificial Intelligence 10
ANN: Artificial Neural Network xi, 9, 16–18, 45, 47, 52

CART: Classification And Regression Tree 13
CBD: Coffee Berry Disease 43
CLR: Coffee Leaf Rust 43
CNN: Convolutional Neural Network xii, 2, 3, 5, 9, 17, 19, 20, 22, 37, 45, 46, 48, 64, 65, 75, 79, 81–83, 89, 100, 101, 103
CWD: Coffee Wilt Disease 43

DL: Deep Learning 3, 5, 10, 37, 43, 45–48, 52, 55, 83, 99, 100
DT: Decision Tree xi, 9, 13, 16, 17, 45, 52, 78

ECX: Ethiopia Commodity Exchange 3, 67

GAN: Generative Adversarial Network 3, 29, 31, 76, 101
GDP: Gross Domestic Product 1, 41

ICO: International Coffee Organization xi, 41, 42

KNN: K-Nearest Neighbors 11–13, 45, 78, 79

LDA: Linear Discriminant Analysis 78
LR: Logistic Regression 9–11, 47, 78

NB: Naive Bayes 78

PCA: Principal Component Analysis 44, 47, 69

RBF: Radial Basis Function 16
RCNN: Recurrent Convolutional Neural Network 101
ReLU: Rectified Linear Unit 21, 58, 59, 84
RF: Random Forest 16, 45–47, 78

SVM: Support Vector Machines 9, 14–16, 44–47, 52, 78, 79

t-SNE: t-distributed Stochastic Neighbor Embedding 69

CHAPTER 1

1. INTRODUCTION

Machine learning methods have been applied to solve social and economical problems of the society. With the advancement of algorithms and computing resources, their application area has become very diverse ranging from autonomous driving to medical devices which previously seemed impossible to tackle (Greenspan et al., 2016; Krizhevsky, Sutskever, et al., 2012; Bojarski et al., 2016). Agriculture is among such areas where researchers are applying computing solutions. This thesis proposes automating the coffee plant disease identification and grading processes using image processing and machine learning approaches.

The datasets for the study were collected in Ethiopia, a country in East Africa, which is among the top producers and exporters of coffee in the world. The country's economy is mainly agricultural, with more than 80% of its population employed in the sector. Currently, agricultural activities represent 45% of the Gross Domestic Product (GDP) and 90% of the foreign exchange earnings. GDP is the monetary value of all the finished goods and services produced within a country's borders in a specific time period usually a year. Among the agricultural products, coffee represents the majority of the economy of the nation. The coffee production plays a vital role in the country's social, economical, cultural and environmental factors (Stellmacher, 2007). The sector accounts for 24% of total foreign exchange earnings and provides a livelihood for more than a quarter of the country's population (Mitiku et al., 2017).

Although Ethiopia is the origin of Coffee Arabica and still among the top exporters of coffee in Africa, the traditional cultivation and knowledge gap hinders the farmers from making enough earnings from their farm. One of the major causes of low yield or production in many parts of Ethiopia

is due to coffee diseases, where a number of cases have been reported (Abera et al., 2011; Alemu et al., 2016; Demelash, Kifle, et al., 2018). Detecting these diseases at early stage plays a vital role in controlling the spread and treating them in a timely manner. This process requires an expert to identify the disease, understand how it expands among the coffee trees, and describe the methods of treatment and protection. Unfortunately, most of the time, there are no experts in the area to give a data based analysis and advice to the farmers. Therefore, during the first year, we designed a machine learning based automatic coffee disease identification model to detect and classify three main coffee plant diseases using images of the plant or parts of the plant. Among the machine learning techniques Convolutional Neural Network (CNN) is used to design the model. CNNs are a specialized kind of neural network for processing data that has known, grid-like topology such as time-series data and image data (Goodfellow, Bengio, et al., 2016).

Coffee grading is the second area that we automated in this work. Grading is a process that all the produced coffee beans pass before introduced into the market. It is a quality assessment of categorizing coffee beans on the basis of various criteria such as size of the bean, where and at what altitude it was grown, how it was prepared and picked, and how good it tastes. The grade value given determines the market price of the beans. This important process is performed by trained experts who are qualified to describe their sensory perception into coffee quality traits exposing the quality assessment to inconsistent results and subjectivity. As a result, supporting the human operator systems with a consistent, non-destructive and cost effective automated system for coffee quality classification is necessary for such commercial products that generate a huge amount of income. During the second year and half of third year, I worked on designing a model to automate the manual coffee grading.

One of the challenges in using machine learning approaches to automate real world situations is getting sufficient amount of data to design the models. After exploring the top sources of datasets (Kaggle (Kaggle Inc., 2019), Amazon (Amazon, 2019),...) and finding no datasets suitable for our task, we collected the images for the datasets ourselves. For the coffee disease model, we created a dataset by downloading images from trusted internet sources and also capturing images ourselves on coffee farms using mobile devices in Ethiopia. The images then are labeled

by researchers at coffee research institute of Jimma University, Ethiopia. 562 images of coffee plant (healthy plus infected) were collected.

The performance of machine learning models is dependent on the amount and quality of data used. The coffee plant disease dataset that we prepared is small compared to the amount required to train CNN models which lead to overfitting; a condition where a model performs good for the training data but fails to generalize when tested with unseen data. To mitigate this problem, in addition to careful designing of the model architecture, we used data augmentation and transfer learning (Torrey and Shavlik, 2010) to design an accurate model. Data augmentation is generating new samples by applying task-specific transformations to the existing data whereas transfer learning is applying knowledge gained in one task to solve a problem in another but related domain. We first trained the model using a plant disease dataset exhibiting similar disease symptoms, then transferred the learned features to classify coffee plant diseases.

I collected the dataset for coffee beans grading at the Jimma branch of Ethiopia Commodity Exchange (ECX), a company responsible for grading of coffee beans, in two rounds nine months apart. In the two rounds, 2109 images of coffee beans were collected using two different mobile devices. In addition to the small dataset issue, the variation in data collection methodology introduced a dataset shift which creates additional challenge in designing the model. We first experimented with classical machine learning approaches and off-the-shelf Deep Learning (DL) models applying transfer learning. The results from these models were not satisfactory enough which lead us to explore further the modern trends in machine learning. We designed a new CNN architecture combined with ensemble learning (Dietterich, 2000; H. Chen et al., 2017) that achieved better than both the classical models as well as the already trained DL models.

Finally, in an attempt to improve the performance further and mimic the hierarchical nature of the manual grading process, a cascaded CNN model is designed. However, the model is too complex for the existing dataset; it overfits even when data augmentation and optimization techniques are used. We have also experimented with Generative Adversarial Networks (GANs) (Goodfellow, Pouget-Abadie, et al., 2014) to generate new training samples from the existing images. But when down scaled images are used as an input, besides failing to improve the

model's performance the generated images appeared blurry and unnatural. It was possible to generate realistic images when the input image is cropped into several pieces before resizing, however, these images cannot be used to train the classifier model because the grading decision is made based on a collection of beans and this approach will result in significant loss of information.

1.1 Contributions

This thesis aims at designing an intelligent system for coffee grading and disease identification. It presents models designed using deep learning techniques that assist the decision making of farmers and coffee grading experts.

— First, a model that detects coffee plant diseases using images of the plant or part of the plant is designed using convolutional neural network and small images of coffee plant (captured by ourselves and some downloaded from the internet).

— The second contribution consists in collecting and creating a new dataset for coffee beans grading.

— Third, using the previous dataset, a model that classifies raw coffee beans into twelve quality grades is designed. Even though the dataset is small compared to the amount of data required to train deep learning models, it was possible to obtain a good model (89.1%) accuracy on the test dataset that performs better than the classical machine learning approaches as well as off-the-shelf deep learning architectures. Since this is, to the best of our knowledge, the first work that attempts to solve these two important problems using deep learning techniques, the coffee beans dataset as well as the models will be made publicly available to be used as a baseline for future research.

— Finally, the designed models are deployed in a mobile application so that the farmers can use the application to get first hand information about the presence and type of disease and take appropriate action.

1.2 Organization

Part I is dedicated to the state-of-the-art and contains chapters 2 and 3.

Chapter 2 presents techniques and algorithms that we use to design the models starting from classical machine learning, fully connected neural networks followed by CNNs. Then we investigate the effect of dataset in representation learning focusing on DL models followed by a discussion of possible solutions for identified dataset issues.

In chapter 3, a brief review of application of machine learning in agriculture is presented. We start by discussing the current practices and the gap in plant disease identification together with an investigation of the existing proposed solutions. The next section presents the economical and social impact of coffee on the country followed by a thorough discussion of the manual coffee grading in Ethiopia's context. We will finalize by presenting some of the publicly available plant datasets.

Part II is dedicated to contributions and contains chapters 4 to 7.

Chapter 4 presents the design of coffee plant disease identification using CNN. One of the challenges, i.e having a small dataset, stated in chapter 2, is addressed by designing a model using transfer learning and data augmentation with only 562 images of coffee plant. We discuss the approaches used to design the model, experimental and evaluation results.

In chapter 5, we describe the data collection and preparation process of the coffee grading dataset. The properties of the dataset are investigated and presented. The possible application areas focusing on machine learning research are presented here.

Chapter 6 presents the design of coffee grading model using CNN and the coffee beans dataset discussed in chapter 5. Though the dataset is small, highly imbalanced and with high data variability, it was possible to design a good classifier using CNN combined with ensemble learning. All the design steps, experimental results and model evaluation is discussed in detail.

Chapter 7 presents the mobile application implementation of the models. Both the disease identification and grading models are implemented in android application so that coffee farmers and stakeholders can easily access them.

In the last chapter, we conclude by discussing the contributions and limitations of the work and indicating directions for future improvements.

Part I

State of the art

CHAPTER 2

2. IMAGE CLASSIFICATION

In this chapter, we will introduce the image classification problem, which is a task of assigning an input image a label from a set of predefined categories. Classification includes image sensors, image preprocessing, feature extraction and image classification. Many classification techniques have been developed for image classification. We will start by first giving a brief review of classical machine learning algorithms such as Logistic Regression (LR), Decision Trees (DTs), Support Vector Machine (SVM) etc., followed by Artificial Neural Networks (ANNs) focusing on CNN which is a recent development of neural networks. CNN will be used in designing of our model architectures in the coming chapters.

We will continue our discussion by giving a brief overview of the challenges of data on feature (representation) learning. The performance of machine learning methods is heavily dependent on the choice of data representation on which they are applied. For that reason, much of the actual effort in deploying machine learning algorithms goes into the design of preprocessing pipelines and data transformations that result in a representation of the data that can support effective machine learning (Bengio et al., 2013). Therefore, we will discuss the problems associated with data and data collection techniques on representation learning and the proposed techniques and algorithms to deal with the problems, focusing on the computer vision domain.

2.1 Classical machine learning

Machine learning is among the most important fields of Artificial Intelligence (AI). It refers to the process of building algorithms that can learn from existing observations, and leverage that learning to predict new observations, or determine the output of new input. Machine learning algorithms can be grouped into several broad categories like supervised and unsupervised learning. Image classification lies under supervised learning where a set of input data, usually called training data, is used to model the underlined input-output relationship in the data. DL is currently leading the race of machine learning powered by better algorithms, computation power and large data. But still classical machine learning algorithms have their strong position in the field. Here, we will discuss the logic behind some classical machine learning algorithms, their application areas, advantages and disadvantages.

2.1.1 Logistic regression

Logistic regression (LR) is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome (Hosmer Jr et al., 2013). The output of the logistic regression will be a probability ($0 \leq x \leq 1$), and can be used to predict the binary output (YES/NO, SPAM/NOT SPAM, Benign/Malignant, ...). For the cases where the outcome is more than two, multinomial logistic regression can be used. As the working principles are the same, we discuss here LR for binary outcomes.

Basic theory

LR allocates weight parameter, θ, for each of the training features. The predicted output ($z(\theta)$) will be a linear function of features and θ coefficients (eqn. 2.1).

$$z(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... \tag{2.1}$$

Where θs are trainable weight parameters and x is input data. The LR model uses the logistic function, a sigmoid function (eqn 2.2), to squeeze the output of eqn. 2.1 between 0 and 1.

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

When we insert z into the sigmoid function, we got an output value, h between 0 and 1. $h(\theta)$ represents $P(y = 1|x)$, the probability of the output

to be one given input x.

$$h = g(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

Cross entropy loss function

For categorical outputs, we use cross entropy loss function given by:

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)), & \text{if } y = 1 \\ -log(1 - h_\theta(x)), & \text{if } y = 0\ . \end{cases} \tag{2.4}$$

The above equation can be rewritten for multiple training samples m,

$$-\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))\right] \tag{2.5}$$

The gradient of the loss function multiplied by the learning rate is used to update the weights after each iteration. Learning rate is a small scalar number used during training that determines how fast or slow the training is progressing.

LR is easy and fast method which doesn't require too many computational resources. It is mostly used for linear classification problems. To get an accurate LR model, we have to filter out attributes that are not related to the output variable as well as attributes that are correlated with each other. Therefore, feature engineering plays a big role in the performance of the model. We cannot also use LR for nonlinear classification problems.

2.1.2 K-nearest neighbor

K-nearest neighbors (KNN) algorithm is a type of supervised machine learning algorithm which can be used for both classification as well as regression problems. KNN works on the bases of an assumption that objects of similar nature exist in close proximity.

Basic theory

In case of KNN classification, a majority voting is applied over the k nearest neighbors to predict the output of a new dataset. It is a lazy learning model (Ripley, 2007) where the computations happens only at

run-time which makes testing phase slow and costly in terms of computational resources. It uses distance functions like euclidean distance to determine the k-nearest points (see Algorithm 1). The performance of the KNN algorithm is mainly dependent on the choice of the number of nearest point. Suppose that we want to classify the test data shown by a puzzle symbol to Class 1 or Class 2 in Fig. 2.1. When k=3 is selected, we have two Class 1 (blue squares) and one Class 2 (red circle) nearest neighbors. Therefore, the test data will be of Class 1. But if k=5 is selected, we will have two Class 1 and three Class 2 datasets in the proximity resulting in Class 2 prediction.

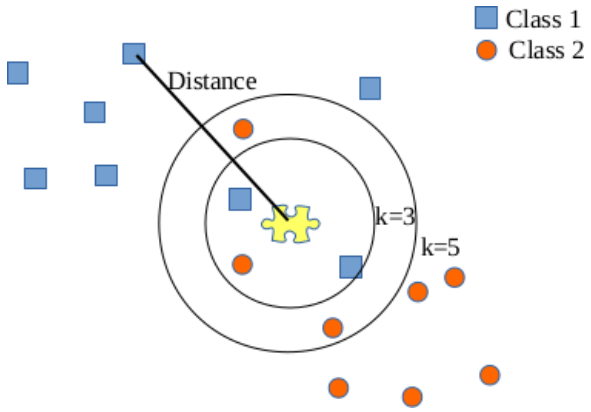


Figure 2.1 – Example of choice of k on prediction output. The test dataset shown by a puzzle symbol will be grouped to Class 1 (blue square shapes) when k=3 but will be grouped to Class 2 (red circles) when k=5.

Algorithm 1: KNN Algorithm

Input: training and test dataset
Output: prediction output

1 Load the dataset
2 Choose the value of k
3 **for** *every sample in test set* **do**
4 (I) Calculate the distance between the test and each training data
5 (II) Based on the distance value, sort the training data in ascending order
6 (III) Choose the top k training data
7 (IV) Assign a class to the test point based on most frequent class of these rows
8 **end**

KNN is simple and easy to implement. New data points can also be added to the train data set at any time since model training is not required. The disadvantages of KNN are it does not work well with high dimensional data and feature scaling is necessary. It is also computationally expensive.

2.1.3 Decision trees

DTs are decision support tools that use a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility (Quinlan, 1986). When training a dataset to classify a variable, the idea of the DT is to divide the data into smaller datasets based on a certain feature value until the target variables all fall under one category.

Basic theory

It is a flowchart-like structure in which each node having a condition over a feature. The nodes decides which node to navigate next based on the condition. Once the leaf node is reached, an output is predicted (Fig. 2.2). The right sequence of conditions makes the tree efficient.

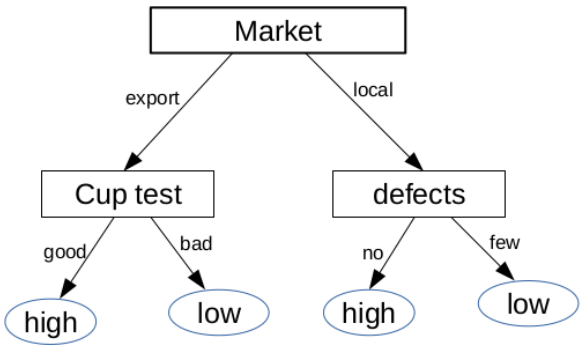


Figure 2.2 – A DT showing a classification into high/low. Nodes shown in square show attributes; links contain the decision conditions while the leaf nodes (oval) contain prediction outputs

Algorithms to select conditions

Entropy/Information gain and gini index are used as the criteria to select the conditions in nodes. For CART (classification and regression

trees) that we will use later in the paper, use gini index as the classification metric. It is a metric to calculate how well the data points are mixed together, determined by equation 2.1.1.

$$gini\_index = 1 - \sum P_t^2 \tag{2.6}$$

where P_t^2 is probability of target. A gini_index gives an idea of how good a split is by how mixed the classes are in the two groups created by the split. A perfect separation results in a gini_index of 0, whereas maximum value of gini_index could be when all target values are equally distributed.

Overfitting is one of the key practical challenges faced while modeling decision trees. If there is no limit set of a decision tree, it will give you 100% accuracy on training set because in the worst case, it will end up making 1 leaf for each observation.

2.1.4 Support vector machines

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. The hyperplane will be N-1 dimension if we have N features. In two dimensional space this hyperplane is a line dividing a plane in two parts where each class lay in either side.

Basic theory

SVM has two variants to support linear and non-linear classifications. We will start our discussion with linear SVM where the classes are linearly separable. Assuming having two classes, SVM tries to find the optimal hyperplane that separates the two classes. As shown in Fig. 2.3a, there are many possible hyperplanes that could be chosen to separate the two classes of data points. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between closest data points of both classes (Fig. 2.3b). Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

If the two classes are non-separable we can still look for the hyperplane that maximizes the margin and that minimizes a quantity proportional to the number of misclassification errors. The trade off between margin and misclassification error is controlled by a positive constant C that has to be chosen beforehand (Osuna et al., 1997). With a normalized or standardized dataset, these hyperplanes can be described by the

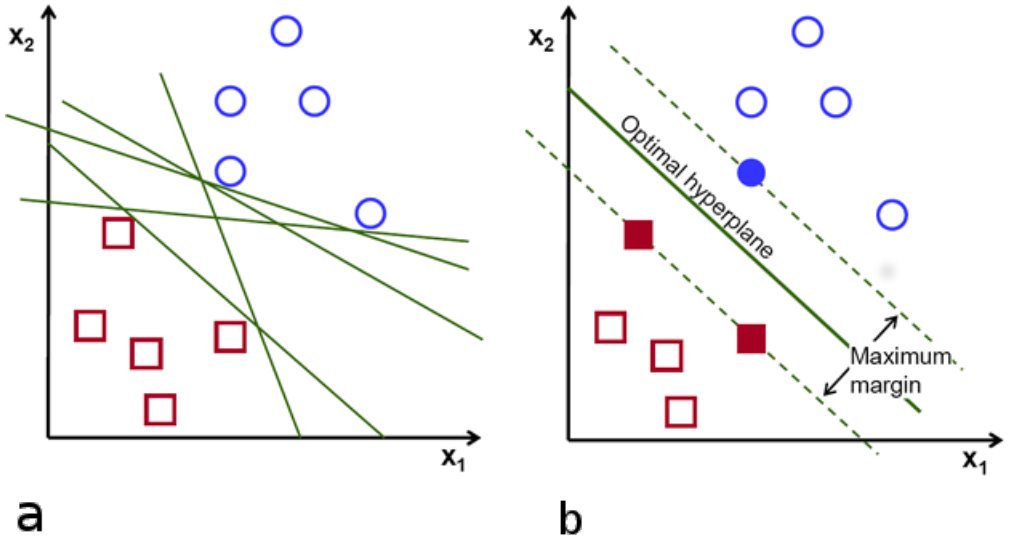


Figure 2.3 – Separating hyperplanes a. Possible hyperplanes b. Optimal hyperplane

equations

$$\vec{\omega}.\vec{x} - b = 1 \tag{2.7}$$

and

$$\vec{\omega}.\vec{x} - b = -1 \tag{2.8}$$

where $\vec{\omega}$ is the normal vector to the hyperplane

Geometrically, the distance between the two hyperplanes is $\frac{2}{\|\vec{\omega}\|}$ so to maximize the distance between the planes we want to minimize $\|\vec{\omega}\|$.

Cost function

Minimizing $\|\vec{\omega}\|$ works well for fully separable classes. To include outliers the second term in eqn. 2.9 is included. Now, the loss function becomes,

$$\|\vec{\omega}\|^2 + C\sum_{i=1}^{m} max(0, 1 - y_i(\vec{\omega}.\vec{x}_i - b)) \tag{2.9}$$

Therefore during training, the objective is to find a set of weight parameters that minimizes the above equation. The value of C determines the penalty over the outliers; a small value of C ignores the outliers and maximizes the margin. A large value of C is sensitive to the outliers.

It is unlikely that most real world problems to be solved by a linear classifier. Hence, it is necessary to extend linear SVM to accommodate

the non-linear decision surfaces. This is simply achieved by transforming the higher dimensional feature space into a lower dimension using kernel functions. The distribution of labels in transformed feature space will be such that training data will be linearly separable. Sigmoid, hyperbolic tangent and Radial Basis Function (RBF) are some of the kernel functions used in SVM.

Some of the advantages of SVM are through the kernel trick, SVM becomes suitable for complex problems. It also handles outliers using the C value. But it takes longer time for large training data. Also the accuracy is highly dependent on the choice of hyperparameters, C and kernel function values.

2.1.5 Random forest

Random Forests (RFs) are an ensemble learning method for classification and regression that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees (Liaw, Wiener, et al., 2002). Ensemble learning works by running a base learning algorithm several times and forming a vote out of the resulting hypothesis (Dietterich et al., 2002). The derived model will be more robust, accurate and handles overfitting better than constituent models.

Basic theory

RF first constructs a set of DTs then combines them to produce a strong classifier . Individually, predictions made by decision trees may not be accurate, but combined together, the predictions will be closer to the actual value. For simple elaboration refer to Fig. 2.4.

It uses the same cost functions as DT algorithm (refer to section 2.1.3). RF is accurate and powerful model but is slower and complex when the forest becomes large.

2.2 Artificial Neural Networks

ANNs are one of the main tools used in machine learning. They are brain-inspired systems which are intended to replicate the way that we humans learn (Jain et al., 1996). An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Through each connection neurons receive,

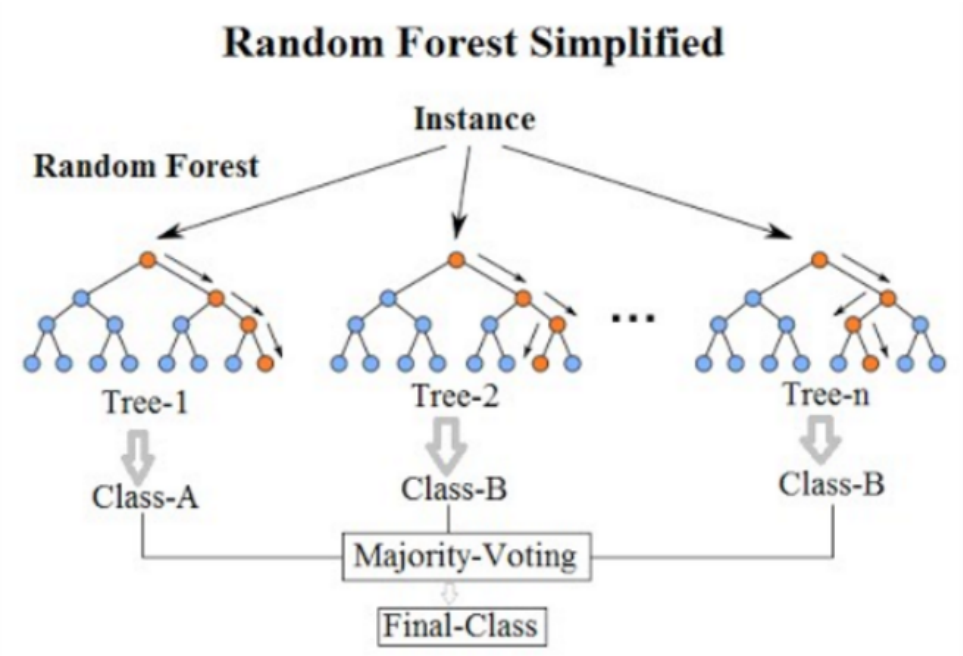


Figure 2.4 – A RF constructed by ensemble of n DTs. credits:(Koehrsen, 2017)

performs some mathematical calculation and transmit synapses represented by weights. The ANNs ability to learn non-linear complex relationship automatically from data made them a model of choice in many application areas. In addition, advances in computing power and techniques for building and training ANNs have allowed these models to achieve state-of-the-art results in image classification (Krizhevsky, Sutskever, et al., 2012; K. He et al., 2016; Szegedy et al., 2015). This breakthrough in image classification is attributed to a special type of ANN called CNN. In this section, we will discuss the basics of fully connected neural network and CNN.

2.2.1 Fully connected neural network

Fully connected neural networks are the simplest kind of ANN where each node in a layer is connected to all the nodes in the next layer. The neurons can be configured in one or more layers as shown in Fig. 2.5. The depth (the number of layers) and width (the number of neurons) depends on the amount of data and computational resources.

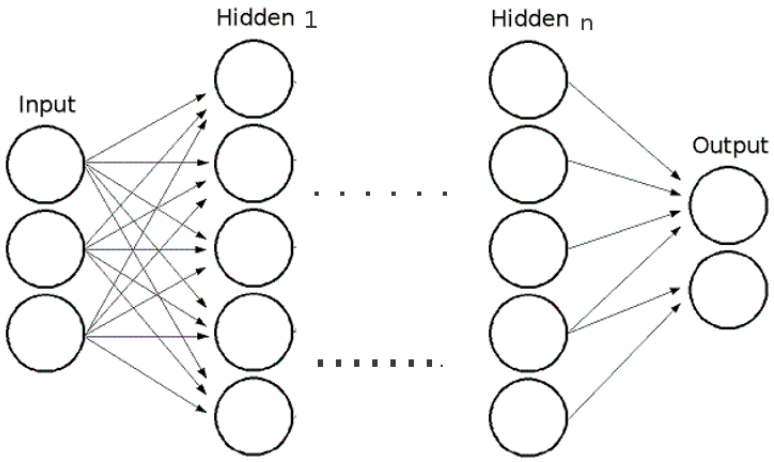


Figure 2.5 – ANN architecture with n hidden layers

As the input propagates through each hidden layer, it is transformed by the weights and the functions used in the nodes and finally produces an output. Learning happens when the produced output is compared with the actual value and the error propagates back to adjust the weights. This is called the back propagation algorithm. The process is repeated for a specified number of iterations (epochs) or until the error is reduced below the targeted threshold.

Mathematically, if the output neurons in layer l is stored in a column-vector a^l, where the superscript index denote the layer. The connections from the neurons in layer $l-1$ to the layer l are stored in a weight matrix W^l, and the biases for each neuron is stored in a bias column-vector b^l,

During the forward pass,

$$a^l = \phi(W^l a^{l-1} + b^l) \tag{2.10}$$

where ϕ is a nonlinear activation function. After the result of last layer (output layer) is calculated, it is compared with the actual value and the error is propagated back using gradient decent optimization algorithm. The error is determined by the cost function like mean squared error (MSE) or cross entropy cost function. During back propagation, if C is the cost function each weight is updated by

$$W := W - \alpha \frac{\partial C}{\partial W} \tag{2.11}$$

$$b := b - \alpha \frac{\partial C}{\partial b} \tag{2.12}$$

where α is the learning rate

There are various optimization algorithms that we can choose to train the network including but not limited to SGD, AdaGrad, RMSProp and Adam. Refer to (Kingma and Ba, 2014; Ruder, 2016) for the full list and principles of each optimization technique.

2.2.2 Convolutional neural network

CNNs are a specialized kind of neural network for processing data that has known, grid-like topology such as time-series data and image data (Goodfellow, Bengio, et al., 2016). It has been widely applied to a variety of pattern recognition problems, such as computer vision, speech recognition, etc. The CNN is inspired from the human visual system (Hubel and Wiesel, 1962) and continually implemented by many researchers (Fukushima, 1980; LeCun, Bengio, et al., 1995). The basic idea of CNN is to build invariance properties into neural networks by creating models that are invariant to certain inputs transformation. This idea originates from a problem that often occurs in the feed forward neural networks, especially fully connected networks. Fully connected layers connect all available inputs to a set of defined output where a single weight per each connection is used without weight sharing. Using a weight per connection in fully connected networks may lead to overfitting and also make them computationally expensive.

CNNs combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling (LeCun, Bengio, et al., 1995). Following, we describe each idea in detail.

Local receptive fields: When dealing with high dimensional inputs, it is infeasible to connect neurons to every neuron of the next layer. Instead, the input of CNN only makes the connection with only a local region of the input volume. Each neuron in a hidden layer will be connected to a small field of the previous layer, which is called a local receptive field. With local receptive fields, neurons can extract elementary visual features such as oriented edges, end-points, corners (or similar features). These features are then combined by the higher layers.

Shared weights: In the convolutional layer, the neurons are organized into multiple parallel hidden layers, which are called feature maps. Each neuron in a feature map is connected to a local receptive field. For every feature map, all neurons share the same weight parameter that is known

as filter or kernel.
Pooling: Pooling usually follows the convolutional layers and used to reduce the number of features. In addition to reducing the features, a pooling layer detects translation invariant features; once a feature is found its exact location is not as important as a rough location with respect to other features.

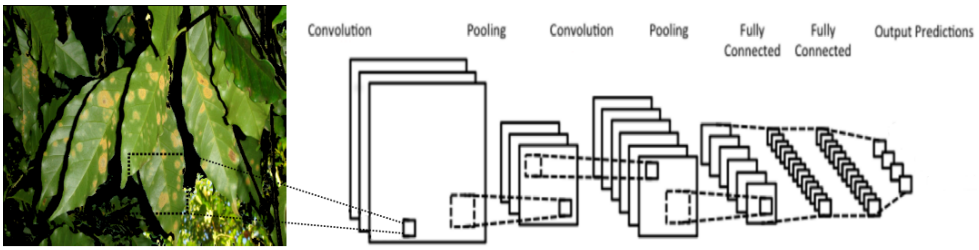


Figure 2.6 – Architecture of CNN.(LeCun, Bottou, et al., 1998)

Figure 2.6 shows a CNN architecture based on the LeNet-5. The LeNet-5 architecture is the first CNN introduced by LeCun, Bottou, et al., 1998 to recognize hand written digits. It consists of two convolutional layers and two sub-sampling layers followed by fully connected layers. The convolutional and pooling layers acts as feature extractors while the fully connected layers serves as a classifier. Below we will see each layer in brief.

Convolutional layer

The convolutional layer is the core building block of CNN. It performs a mathematical convolution operator that preserves the relationship between pixels by learning image features using small squares of input data. It takes inputs such as filter, kernel size, stride and padding. The filter is filled with array of numbers; weight parameters. Szegedy et al., 2015 demonstrated using small filters (3X3) improves the performance for ImageNet dataset (J. Deng et al., 2009). The kernel size specifies how many of such filters we are using. The stride is a parameter that determines how many pixels the filter is moving along (sliding) during convolution. Some times filters does not perfectly fit along the input volume. Padding tells whether to add zeros along the boarder of the input (zero padding) or drop the part of the image where the filter did not fit. All these are hyper parameters that need to be tuned during model design and are mainly dependent of the availability of resources.

Activation layer

The convolution layer is often followed by nonlinear activation function similar to the fully connected neural networks that we have discussed earlier. Rectifier Linear Unit (ReLU) is the most widely used activation function because researches (Krizhevsky, Sutskever, et al., 2012) has proven that ReLU result in faster training. Its output is given by

$$\phi(x) = \max(0, x) \tag{2.13}$$

Among the years, many variations are proposed extending the function of ReLU (Leaky Relu, parametric ReLU (Xu et al., 2015), Exponential ReLU (Clevert et al., 2015)). We will use ReLU and Leaky ReLU later during our model design after each convolutional layer.

Leaky ReLU adds a small gradient to when the input is less than zero as shown in the equation below.

$$\phi(x) = \begin{cases} x & \text{for } x \geq 0 \\ \frac{x}{a} & \text{otherwise} \end{cases} \tag{2.14}$$

where a is a number between (1, $+\infty$).

For the last layer softmax activation function is often used. It is an activation function that turns numbers into probabilities that sum to one. It outputs a vector that represents the probability distributions of a list of potential outcomes using a formula

$$\phi(x)_i = \frac{e^{x_i}}{\sum_j^k e^{x_j}} \tag{2.15}$$

where i is in $[1, 2, ..., k]$ and k is the dimensional vector of $\phi(x)$.

Pooling

As stated above, the pooling layer is used to reduce the number of features. Depending on the type of pooling, it performs some operation (averaging, maximum, ...) sliding over a given area of the input. It takes two parameters the filter size and strides. Maxpooling, which takes the largest number over the area rectified by the filter, is the most commonly used pooling method.

Fully connected layer

The output of the last pooling layer is flattened and fed to one of more fully connected layers that serve as a classifier.

The training of CNN is similar to that of fully connected neural networks using back propagation algorithm. The same cost functions and optimization algorithms that we have discussed for fully connected layers can be used for CNNs also. Fig. 2.7 shows the performance comparison of these optimization techniques when training CNN using CIFAR10 (Krizhevsky, Hinton, et al., 2009) dataset.

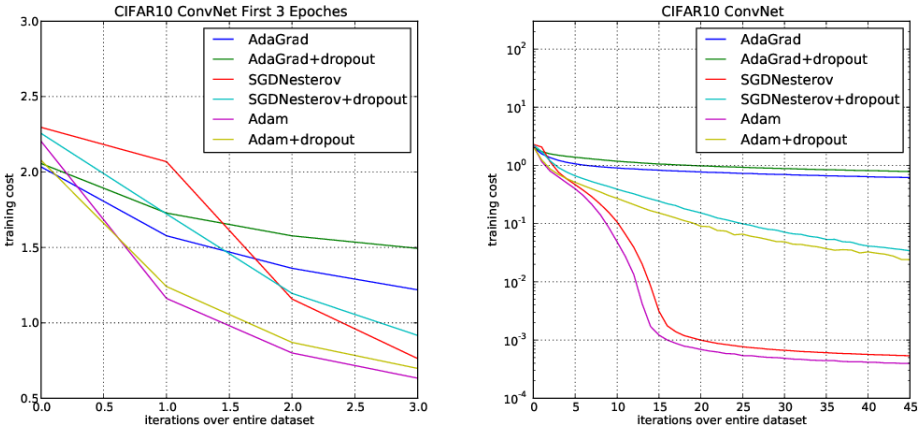


Figure 2.7 – Performance comparison of optimization algorithms (Kingma and Ba, 2014)

2.3 Data and representation learning

The performance that one can achieve from a model is a factor of the model of choice, the data and the features prepared. One can argue, however, if good features are prepared and selected, a 'bad model' would learn and generate good (not optimal) results giving more emphasis on feature engineering. Feature engineering involves the selection of a subset of informative features and/or the combination of distinct features into new features in order to obtain a representation that enables classification. The traditional approach for image classification has been based on hand-engineered features. In contemporary machine learning, this is shifting from hand-engineering features to automatic learning of representation from raw data (LeCun, Bengio, and Hinton, 2015). In this

section, we will discuss about data and representation learning focusing on computer vision domain.

2.3.1 Feature extraction in image classification

Feature extraction can be viewed as finding a set of vectors that represent an observation while reducing the dimensionality. In image classification, it is desirable to extract features that are focused on discriminating between classes. Although a reduction in dimensionality is desirable, the error increment due to the reduction in dimension has to be without sacrificing the discriminative power of the classifiers (Benediktsson et al., 2003). Depending on the application, features can be extracted globally or locally for image based modeling. Global features are extracted from the content of the whole image while local features are extracted from local regions of an image.

In the vision domain color, edge, texture are often used as descriptors for feature extraction. There are several techniques in each category like histogram analysis, Hu-moments (M.-K. Hu, 1962), Haralick Texture (Haralick, Shanmugam, et al., 1973), SIFT (Lowe, 2004), SURF (Bay et al., 2008) and so on. The choice of the algorithms depends on the problem at hand and needs domain specific knowledge. Therefore, the quality of the extracted features is mainly dependent on the experience and the expertize of the person designing them. This leads the performance of machine learning models to highly depend on the extracted features. However, a recent trend in machine learning has demonstrated that learned representations are more effective and efficient (LeCun, Bottou, et al., 1998; Krizhevsky, Sutskever, et al., 2012). The main advantage of representation learning is that algorithms automatically analyze large collections of images and identify features that can categorize images with minimum error. However, for representation learning to be successful and extract relevant features, large amount of data is required. In most applications, it is often difficult to acquire enough data and the data usually have highly skewed distributions. Also, for the representation learning to be called successful, these features should be invariant to rotation, translation and illumination conditions.

2.3.2 Small datasets

To understand the problem of small data for machine learning, we must first discuss bias and variance.

Bias measures the deviation of the average prediction of a model from the actual value. High bias occurs when the model has little flexibility to learn the underlying features in the data. This always results in high error on the training as well as test dataset.

Variance occurs when the model is too complex fitting irrelevant (noise) features together with the underlying features in the data. For example, a model may consider wearing caps as a feature to the identification of gender although it has no any discriminating ability. Models with high variance fits the training set very well while performing poorly on the test dataset. Fig. 2.8 shows an example of bias vs variance using bull's eye diagram. If a model is simple with too few parameters, it will have high

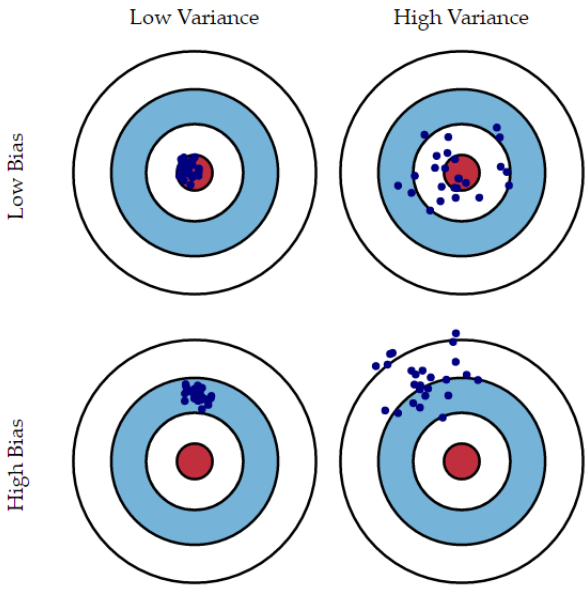


Figure 2.8 – Example of bias vs variance (James et al., 2013)

bias and low variance. But if it is complex with too many parameters, it will have low bias and high variance. Therefore, there is a trade-off between bias and variance and the goal is to find an optimal model that is complex enough to capture the pattern present in the data but not too complex, to prevent overfitting.

Deep learning models show extremely high performance in several domains. However, a large number of data is required to achieve good performance without the model suffering from overfitting. Fig. 2.9 shows the performance of deep learning and classical learning as a function of

training data set. It shows that the performance of deep learning models increases with training data, however we should keep in mind the bias-variance problem while designing the model architecture. In real world applications, collecting and labeling data is often expensive. Small data set conditions exist in many fields, such as disease diagnosis, fault diagnosis or deficiency detection in mechanics, aviation and navigation, etc (Mao et al., 2006).

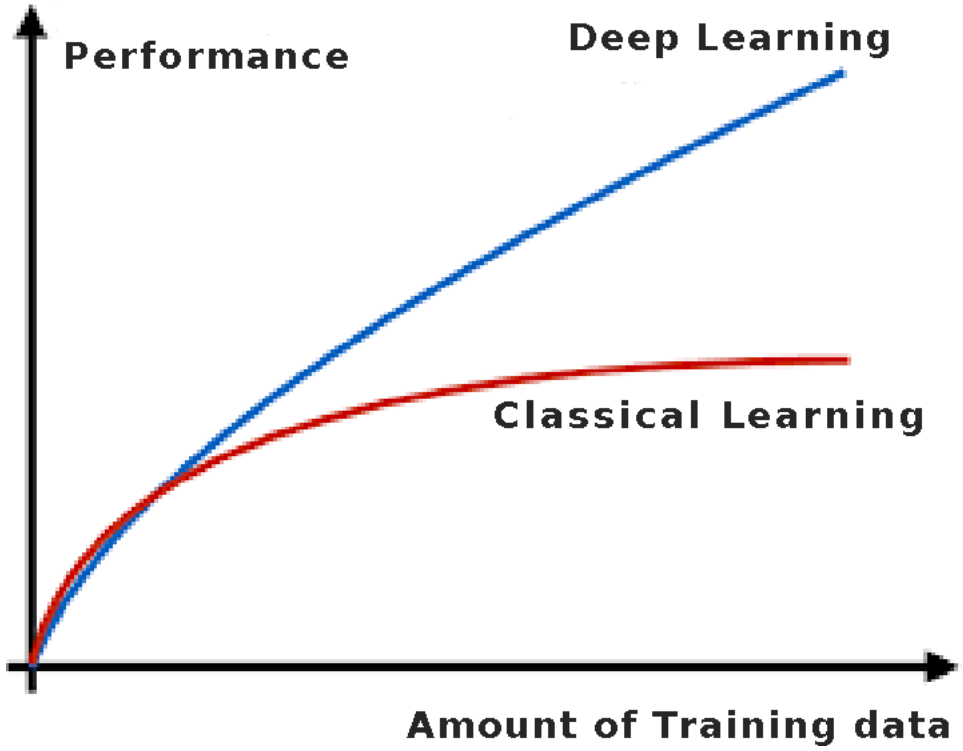


Figure 2.9 – Performance of deep learning and classical learning as a function of amount of training data (Zappone et al., 2019)

2.3.3 Imbalanced dataset

Most of publicly available datasets for predictive problems contain data equally distributed among classes. But in real world applications, it is often difficult to obtain equal amount of instances for each class. This condition is called the imbalanced data problem. Without handling the data imbalance issue, conventional methods tend to be biased towards the majority class with poor accuracy for the minority class (H. He and Garcia, 2008). This is especially a problem when prediction of the

rare class is a priority in cases like fraud detection, emergency prediction and diagnosis of rare medical conditions. As an example, in the case of fraud detection, if there are 100,000 genuine instances and 100 fraud instances in the dataset. Suppose we have two models where,

a. 60 frauds are classified as genuine and 100 genuine are classified as fraud

b. 10 frauds are classified as genuine and 200 genuine are classified as fraud

If number of correct classification is the metrics used, then model **a** is a good classifier (160 wrong classifications vs 210 wrong classifications). But in this case it is better to have high false positives than false negatives. Therefore, model **b** is better at detecting fraud than model **a** (with 10/100 = 0.1 error vs 60/100= 0.6 error). However, imbalanced data can also occur in areas that do not have an inherent imbalance problem. In this case, the imbalance is caused by data collection techniques and limitations in obtaining a representative dataset.

Learning from imbalanced data is more problematic when there is no enough data. In the previous section, we have seen that in a given classification task, the size of the dataset has an important role in designing a good classifier. Lack of examples therefore, makes learning regularities difficult for rare classes. In fact, it has been shown that as the size of training set increases, the error rate caused by imbalanced training data decreases (Japkowicz and Stephen, 2002). The machine learning community has addressed the issue of the data imbalance in two ways: in the data level and in the algorithm level. The data level focuses to balance the data by resampling, either by over-sampling the minority class and/or under sampling the majority class (Japkowicz, 2000; Chawla et al., 2002; H. He, Bai, et al., 2008). The algorithm level assigns distinct costs to the training samples (Elkan, 2001).

A cost sensitive learning takes costs, such as misclassfication cost, into consideration during model construction and produces a classifier that has the lower cost (G. H. Nguyen et al., 2009). Different approaches were devised to convert the existing cost insensitive learning algorithms into cost sensitive learners (C. H. Nguyen and Ho, 2005; Alejo et al., 2007; Zhou and X.-Y. Liu, 2006; Tao et al., 2005), including sampling, weighting, thresholding and ensemble learning.

Methods that use weighting assign higher weight for the samples from the minority class and lower weight for samples from the majority class forcing the classifier to be more sensitive to the misclassification of the minority class. For example, considering a binary classification problem, where cross entropy is a common choice of cost function given by,

$$CrossEntropy = -y * log(p) - (1-y) * log(1-p) \tag{2.16}$$

where p is the predicted probability and y is the binary indicator (0 or 1). For the weighted learning the loss function now becomes,

$$CrossEntropy\_weighted = -w_{class1} y * log(p) - w_{class0}(1-y) * log(1-p) \tag{2.17}$$

where w_{class0} and w_{class1} are the weight of each class.

Different strategies are proposed to determine the weight values. C. H. Nguyen and Ho, 2005 proposed to weight samples based on the local data distribution, while Tao et al., 2005 recommended posterior probability to weight training samples. Like any other solutions, cost sensitive learning techniques also have some drawbacks. They assume misclassification costs are known, however in practice, specific cost information is unavailable because costs often depend on a number of factors that are not easily compared (G. H. Nguyen et al., 2009).

2.3.4 Domain shift

In life, the meaning of numbers can change. Inflation reduces the value of money. Lighting changes can affect the appearance of a particular color or the meaning of a position can change depending on the current frame of reference. Furthermore, there is often the possibility of changes in measurement units. All of these can cause dataset shift. We call this particular form of dataset shift "domain shift" (Quionero-Candela et al., 2009). Deep networks can learn generic representation when trained using large-scale datasets. But due to domain shift, these learned representations do not generalize well when tested on different datasets. For example, Fig. 2.10 shows two famous digit recognition datasets, MNIST (L. Deng, 2012) and SVHN (Netzer et al., 2011). Unlike humans, a model trained on the MNIST dataset fails to recognize digits on the SVHN dataset and vice versa even though both datasets contain digits.

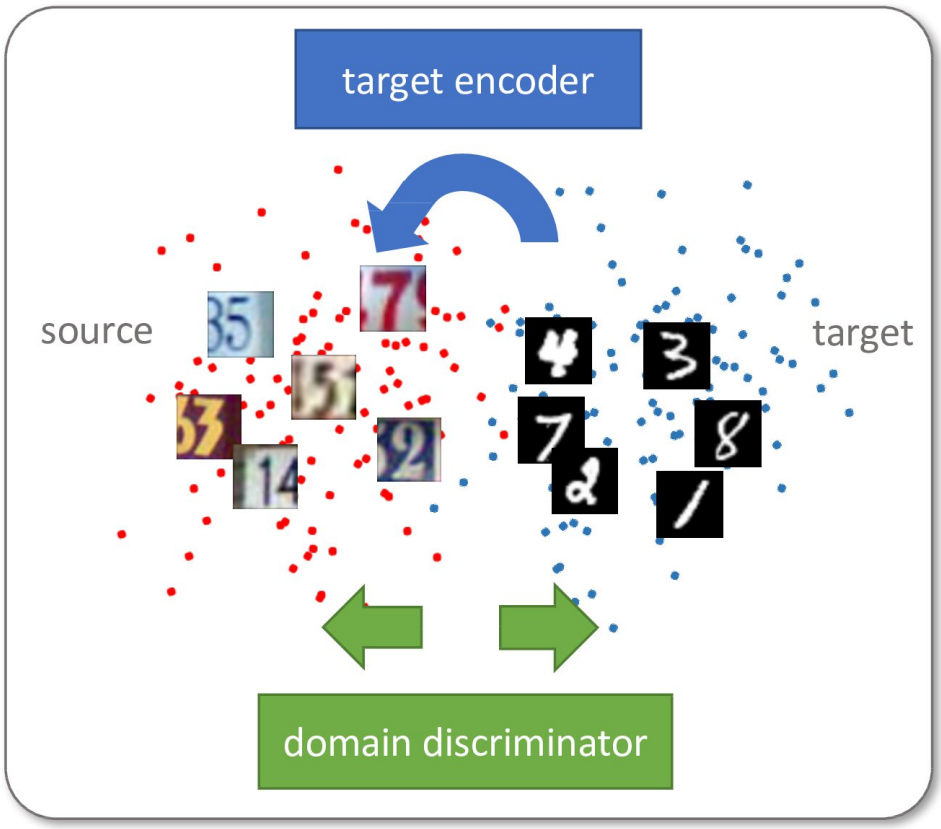


Figure 2.10 – Example of domain shift (Tzeng et al., 2017)

To better understand the problem of domain shift, we consider a potential use case. Suppose a startup company wants to deploy a mobile-based disease identification system for a particular plant. The idea is to design an application that can detect and identify plant diseases taking the images captured by the farmers using their mobile phones. During the design phase the startup couldn't get enough images of the plant showing visual symptoms and decided to train the model using images collected from research labs. The images obtained from the research labs contain a single leaf image taken in a laboratory environment, where the lighting, the camera and the background were adjusted. A model trained using this dataset that achieves performance beyond the requirement would fail when moved to the real environment. The images submitted by the farmers are of different distribution from the training images due to lighting, background, resolution, scale and camera quality. There-

fore, in addition to solving the initial problem, the model needs to adapt to the domain shift. Domain adaptation is an open research area where learned knowledge obtained from one domain is transferred to another domain.

2.4 Learning with small datasets

The problem of leaning from limited data has been approached from various directions. Here we will discuss some of the approaches that we will experiment with during designing our models.

2.4.1 Synthesize data

Synthesizing data, also known as data augmentation, is generating new samples by applying task-specific transformations to the existing data. It has been used as a regularizer in preventing overfitting by increasing the number of training sample (Simard et al., 2003; Cireşan et al., 2010) and to balance classes in the data by applying augmentation only on under-represented classes (Chawla et al., 2002). A recent development in neural networks, GAN, has also been used as a data generation technique by synthesizing data from random noise to improve the performance of classifiers (Zheng et al., 2017; Goodfellow, Pouget-Abadie, et al., 2014; Frid-Adar et al., 2018; Odena, 2016). Before discussing about each technique, let us see a simple example how data augmentation increase the robustness of neural network.

Data augmentation increases the robustness of a neural network by preventing it from learning task irrelevant patterns, essentially boosting the overall performance. As an example consider samples from a dataset that will be used in chapter 4. Suppose we are designing a neural network to classify plants into diseased and healthy using leaf images. If all healthy leaf images in the training dataset have constant background and show full image of a single leaf as shown in Figure 2.11a, but the diseased leaf samples show only portions of the leaf as shown in Figure 2.11b, a network trained and achieving good performance on this data will fail dramatically if it is tested with an image like the one in Figure 2.12. Adding a random cropping augmentation function, however would solve the problem.

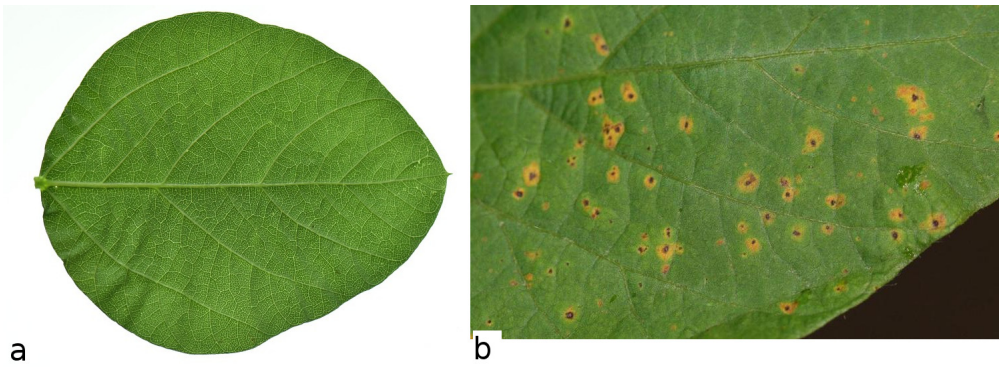


Figure 2.11 – Example of training images: a) healthy leaf image b) diseased leaf image

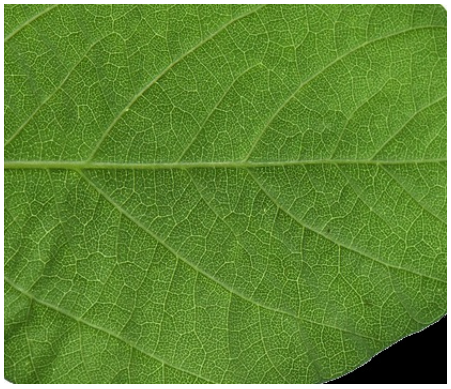


Figure 2.12 – Example of test image: healthy leaf labeled as diseased

Geometric transformations

In this section, we will see some basic geometric transformation applied to images to increase the training dataset. These are not by far the only techniques available, refer to (Shorten and Khoshgoftaar, 2019) for the complete list. An essential concept of when applying these transformation is that the transformations applied to labeled data do not change the semantic meaning of the label. For example, taking a case in computer vision, the scaled, rotated, flipped, translated image of a dog would be still be an image of a dog, and thus it is possible to apply these transformations to produce additional training data while preserving the semantic meaning of the label.

Flipping

Images are added by flipping the original image vertically or horizontally. Depending on the dataset, this may not be a label-preserving transfor-

mation e.g., datasets involving text recognition such as MNIST or SVHN.

Rotation

Rotation augmentations are done by rotating the image right or left on an axis between 1^o and 359^o. The safety of rotation augmentations is heavily determined by the rotation degree parameter and problem type.

Scale

The image can be scaled outward or inward. While scaling outward, the final image size will be larger than the original image size. Most image frameworks cut out a section from the new image, with size equal to the original image.

Translation

Translation just involves moving the image along the horizontal or vertical direction (or both). This can be very useful to avoid positional bias in the data. For example, if all the images in a dataset are centered, this would require the model to be tested on perfectly centered images as well. Performing random translation on the training dataset improves generalization ability of the model without imposing this constraint.

Adding noise

Over-fitting usually happens when your neural network tries to learn high frequency features (patterns that occur a lot) that may not be useful. Adding just the right amount of noise can enhance the learning capability.

Generative adversarial networks for data augmentation

In this section, we will see the GAN approaches of data augmentation. GANs were first introduced by Goodfellow, Pouget-Abadie, et al., 2014. It is a type of generative model, that means it produces new data based on the training image. After first introduction, several researches have been done and fascinating results were achieved from images to video generation.

Generator and discriminator

A GAN is made of two neural networks that compete against each other, the generator and the discriminator (see Fig. 2.13). The generator, as the name implies, generates data similar to the training data while the discriminator checks the validity of the created data, i.e tries to identify whether the data is from the generator or from the real data. In other words, the generator tries to fool the discriminator, while the discriminator tries to prevent this from happening.

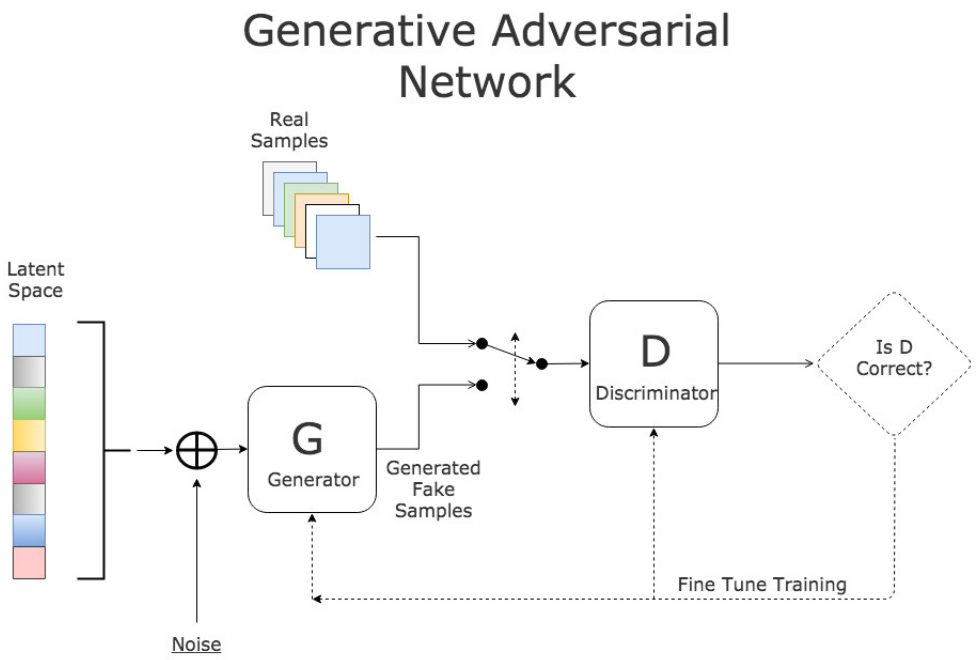


Figure 2.13 – Concept of GAN

The generator $G(z,\theta_1)$ learns to map a latent space, z, to the distribution of the data, x, it aims to reproduce, so that when fed with a noise vector from the latent space, it predicts a sample from the estimated distribution. Whereas the discriminator $D(x,\theta_2)$, outputs the probability that the data came from the real dataset. In both cases, θ_i represents the weights or parameters that define each neural network. This is essentially a minimax game played by the two networks. The discriminator wants to maximize the probability of the real data being identified as "real" and the generated data being identified as "fake" while the generator wants to minimize the probability that the discriminator identifies its generated data as "fake". Mathematically, the value function is given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[logD(x)\right] + \mathbb{E}_{z\sim p(z)}\left[log(1-D(G(z)))\right] \quad (2.18)$$

where

— $D(x)$ is the probability that x is "real" according to the discriminator.

— $G(z)$ is a sample generated by the generator given a latent vector (z).

As the training progresses and ideally when equilibrium is reached, the generator synthesizes data which looks real and the discriminator is confused maximally unable to differentiate between real and fake.

2.4.2 Transfer learning

Transfer learning is applying knowledge gained in one task to solve a problem in another similar domain (Pan and Yang, 2009). Many machine learning methods work well only under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the assumption does not hold, classification methods might perform worse. However, in practice, this assumption may not always hold true. For example, in medical image classification the training data collected in one lab and used to model the classifier may appear different from the test data collected in another lab under different conditions. Collecting labeled data which represents all possible conditions is expensive and infeasible. Therefore, classifying the test data correctly using only the training data becomes an interesting problem.

Transfer learning allows the domains, tasks, and distributions used in training and testing to be different. In transfer learning, we first train a base network on a base dataset and task, and then we re-purpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task (Yosinski et al., 2014). To use transfer learning for predictive modeling, there are two common approaches to follow: develop model approach and pre-trained model approach. Both methods follow the same steps and reuse the model trained on another task except in the first approach a new model is designed using similar task where there is abundant data to map input-output relationships while the pre-trained model approaches uses one of the available trained models (VGG (Simonyan and Zisserman, 2014), GoogLeNet/Inception (Szegedy et al., 2015), ResNet (K. He et al., 2016)) released by different research organizations. The approaches followed in both cases is summarized in algorithm 2.

Pre-trained models

Now we have discussed how transfer learning can be used to improve the performance of a model, let us see some of the pre-trained models according to their chronological order of development.

LeNet (LeCun, Bottou, et al., 1998)
LeNet, a pioneering 5 layer convolutional network by LeCun, Bottou, et al., 1998, that classifies digits, was applied by several banks to recognize

Algorithm 2: Steps in transfer learning

Step 1: **Select the source task/model:** we must select the source task in case of model development approach. To get the best out of this approach, there should be some similarity between the source and the task of interest. If we are using pre-trained models, this step involves choosing a pre-trained model.

Step 2: **Build source model:** this step is required in develop model approach. Once the source task is selected, a model must be designed and trained to learn features in this task.

Step 3: **Reuse the model:** the model on the source task or pre-trained model can then be used as a starting point for the training on the task of interest. Here we can train the whole model or parts of the model. Usually the convolutional layers are frozen and only the classifier part is trained. Whether to train the whole layer or not and how many layers to train depends on the similarity of the tasks and the amount of data available for the task of interest. The more similar the tasks, the less layers retrained.

Step 4: **Fine-tuning:** this is an optional step involving adapting the model to the task of interest by tuning parameters of the model.

hand-written numbers on checks digitized in 32x32 pixel greyscale input images.

AlexNet (Krizhevsky, Sutskever, et al., 2012)
AlexNet is the first model developed by Krizhevsky, Sutskever, et al., 2012. It had produced breakthrough results in ImageNet dataset (J. Deng et al., 2009) achieving a top-1 error rate of 37.5% which was better than previous start of art methods which achieved 47.1%. The network consists of 8 layers out of which 5 are convolutional layers and other 3 are fully connected layers. AlexNet is more deeper than LeNet with more filters and convolutions. It was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs.

GoogLeNet/Inception (Szegedy et al., 2015)
GoogleNet is developed by Google in 2014 and achieved a top-5 accuracy of 6.67% on ImageNet dataset. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but re-

duced the number of parameters from 60 million (AlexNet) to 4 million.

VGG (Simonyan and Zisserman, 2014)
VGG is developed by Visual Geometry Group at University of Oxford. It has two varieties the VGG-16 and VGG-19 with 16 and 19 layers, respectively. Simonyan and Zisserman, 2014 investigated the effect of the depth of the network on the performance of image recognition model using the ImageNet dataset. They used small filter (3x3) instead of large filters. It improved the top-5 error to 7.3% from 15.3% of that of AlexNet. VGG-16 will be used as a pre-trained model in our experiments because it is easy to implement and the trained weights parameters are available in the deep learning frame work we use to implement the models.

ResNet (K. He et al., 2016)
Residual Neural Network (ResNet) by K. He et al., 2016 is another break through witnessed by deep learning community after AlexNet. They introduced a novel architecture by using skipped connections called residual modules. Residual modules are proposed to solve the vanishing gradient problem exhibited by deeper networks. The vanishing gradient problem is that the signal required to change the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. This makes the learning of the earlier layers almost negligible. ResNet-X has several varieties where X represents the number of layers. ResNet-152 achieves a top-5 error rate of 3.57% which beats human-level performance on the ImageNet dataset.

2.4.3 Ensemble learning

An ensemble of classifiers is a set of discriminative models whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples. One of the most active areas of research in machine learning community has been to study the method of constructing good ensemble of classifiers. Studies proved that it is possible to get good classifier from ensemble of several accurate and uncorrelated classifiers. An accurate classifier is one that outperforms random guessing, and uncorrelated classifiers are those that commit independent errors. There are a few different methods for ensembling, but the two most common are:

— Bagging (Breiman, 1996) trains a set of classifiers and creates in-

dividuals for its ensemble by training each classifier on a random redistribution of the training set. The output of each classifier is then combined together to smooth out the output. Breiman, 1996 showed that Bagging is effective on unstable learning algorithms where small changes in the training set result in large changes in predictions.

— Boosting (Freund, Schapire, et al., 1996) trains a large number of weak classifier sequentially. Each classifier in the series focuses on learning the samples that were predicted incorrectly by the previous classifiers.

2.4.4 Other methods

In this section, we will provide a brief overview of other approaches that are developed to train deep learning models using small dataset.

Dropout

Dropout is a technique developed by Srivastava et al., 2014 to address the problem of overfitting. Dropout randomly cancels neurons (along with their connections) during the training. It regularizes the network by forcing it to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

L1 and L2 regularization

In L1 and L2 regularization (Krogh and Hertz, 1992; Ng, 2004) a regularization term is added to the cost function to stop it from perfectly fitting the training data. The difference between L1 and L2 regularization varies only on how the regularization term is computed. In case of L1, the new cost function becomes:

$$Cost\ function = loss + \lambda \sum_{i=1}^{k} \|\omega_i\| \tag{2.19}$$

whereas for L2,

$$Cost\ function = loss + \lambda \sum_{i=1}^{k} {\omega_i}^2 \tag{2.20}$$

where λ is the regularization parameter, ω represents the weights and loss is the cost function (MSE, cross entropy, ...)

Batch normalization

Batch normalization is a regularization method developed by Ioffe and Szegedy, 2015. It normalizes each input in a batch by subtracting the batch's mean and dividing it by the batch's standard deviation. It also enables fast and stable training for DL models.

N-shot learning

N-shot learning are machine learning algorithms which build a model from very limited data. N represents the number of training samples allowed to be seen before the model starts predicting. Zero-shot learning is the extreme where the model predicts a sample without using any training example from that class. When N is one, we have one shot learning and few shot learning when it more than one. An approach to these algorithms is the use of networks that learn a distance function such that image classification is possible even if the network has only been trained on one or a few instances.

2.5 Conclusion

This chapter presented a brief overview of image classification. We started by presenting the basics of the core models used in classification starting from classical machine learning models to CNN. The performance of a classifier model is mainly dependent on the features used during training. As a result of that extracting feature has shifted from hand-engineering to automatic feature learning from raw data which is dependent on the amount and distribution of data. This is especially a problem in modeling real world situations where obtaining large amount of labeled data is expensive. Therefore, investigating the dataset to identify the problems in the dataset should be the first step before designing a reliable model so that it would be possible to implement appropriate techniques. We continued our discussion by giving an in depth analysis of how data affects representation learning and the models performance in general. We presented the problems associated with data and data collection techniques such as small dataset, imbalanced dataset and domain shift on representation learning. Finally, a survey of algorithms proposed to solve the issues when learning using small datasets is presented.

CHAPTER 3

3. Machine Learning in Agriculture

3.1 Introduction

Agriculture is among the top factors that are considered as basis of human life. In addition to being an important source of energy, it has become a fundamental piece in the puzzle to solve the problem of global warming. It also plays a vital role in the economy of a country creating employments and foreign exchange earnings. According to the World bank data, in 2018 among the world's population 28% is employed in the sector from which the majority are located in sub-Saharan African countries and Asia. Therefore, the growth of agriculture is essential for the growth of the economy of a country. Smart agriculture has been deployed to assist the sector and enhance yield generating more output from the same amount of input in the past few years. It encompasses many different modern technologies that can be used individually, or together, to increase the efficiency of agricultural operations. Machine learning is among such smart farming techniques. It has been applied in several agricultural applications such as weed and disease identification (Bah et al., 2018; Milioto et al., 2018; Hung et al., 2014), land cover classification (Tuia et al., 2015; Kussul et al., 2017), fruit and food grading (Son et al., 2018; Semary et al., 2015; El-Bendary et al., 2015).

In this chapter, we will provide a survey of machine learning approaches focusing on plant disease identification and crop grading. We will start by a brief discussion of the effect of plant diseases in crop yield. Then we will see the current disease detection techniques and some potential so-

lutions. The current practice of coffee grading is presented in detail in the context of Ethiopia. Finally, we will present some of the publicly available plant datasets.

3.2 Plant diseases and identification

As agriculture struggles to support the ever growing global population, plant disease causes major production and economic loses. Roughly, yield losses caused by pathogens, animals, and weeds, are responsible for losses ranging between 20 and 40 % of global agricultural productivity (Savary et al., 2012). This has substantial effect on the income of a farmer, food security and economy of a country. For example, there are incidents recorded in history where plant diseases compromised food security of countries sometimes leading to devastating famines (Padmanabhan, 1973; Ullstrup, 1972). Hence, prevention and control of plant diseases is vital.

Farmers spend a lot of money on plant disease control, however, their effort is often wasted because it is not assisted by information and technology. This is especially amplified in poor countries where there is a significant lack of experts nearby and infrastructure to obtain timely information. This delays early detection and taking prevention measures on time which exacerbates the damage. For example, soybean rust (a fungal disease in soybeans) has caused a significant economic loss and just by removing 20% of the infection, the farmers may benefit with an approximately 11 million-dollar profit (Sankaran et al., 2010). Therefore, early detection and identification of plant diseases plays the utmost important role in taking timely measures.

There are several ways to detect plant pathologies. Some diseases do not have any visible symptoms associated, or those appear only when it is too late to act. In these cases, it is necessary to perform sophisticated analysis, usually by means of powerful microscopes. In some cases, the signs can only be detected in parts of the electromagnetic spectrum that are not visible to humans (Barbedo, 2013). Most diseases, however, generate some kind of manifestation in the visible spectrum. The diseases may exhibit symptoms on different parts of the plant, i.e. leaves, stem, fruits/seeds etc. In this case, disease detection is performed by visual inspection.

Manual detection using visual perception has some drawbacks. First, it results in subjectivity and unreliability because experts may tire and lose concentration. It also requires trained experts to inspect the farm periodically, which incurs a huge cost (Bock et al., 2010; Barbedo, 2013). Unfortunately, most of the time there are no experts in the area especially in developing countries like Ethiopia where farmers have to walk long distances to get advice. Therefore, looking for a fast, automatic, less expensive and accurate method to detect plant diseases is of great importance. Image processing and machine learning techniques have been applied to plant disease detection since the 1970's (Al Bashish et al., 2011). We will present a review of these methods in section 3.2.2.

3.2.1 Coffee in Ethiopia

Coffee is one of the most important globally traded agricultural commodities, with consumption occurring mostly in developed countries and production in developing ones. According to the International Coffee Organization (ICO), the total production by all exporting countries was 10.81 million tons of green coffee in the coffee harvest year 2017/2018 (ICO, 2019). Coffee is one of the most important agricultural activities to earn foreign exchange in many tropical and subtropical countries (Faridah et al., 2011; Ayitenfsu, 2014). Measuring the value of coffee in terms of foreign exchange; however, often limits the social and economical relevance of the crop within these producing countries. A large fraction of the world population has some relationship to the coffee sector, from farmers and farm laborers, processors, exporters, importers, roasters, and retailers to consumers. Approximately 125 million people worldwide are estimated to depend on this commodity for their incomes in 2002 (Trade et al., 2003). Any event affecting coffee production and/or coffee prices could potentially impact the livelihoods of millions of people, as witnessed by the coffee crisis exhibited in the early 2000's (Osorio, 2004). This has had devastating effect on the economic and social social life of many producers in developing countries causing in widespread job losses, migration to cities and abroad abandoning farms and less money available for health care and education.

Ethiopia's economy is not any different from the world coffee economics. The country's economy is mainly agricultural, with more than 80% of its population employed in this sector. Currently agricultural activities represent 45% of the GDP and 90% of the foreign exchange earnings. Among which, coffee represents the majority of the economy of the nation

with 24% of total foreign exchange earnings (Mitiku et al., 2017). In the year 2018/19, according to the global agriculture report network, about 435,000 metric tons of coffee were produced in the country. Fig. 3.1 shows the exportable production of coffee in the country. The sector has pivotal role on the social, economical, cultural and environmental factors providing a livelihood for more than a quarter of the country's population.

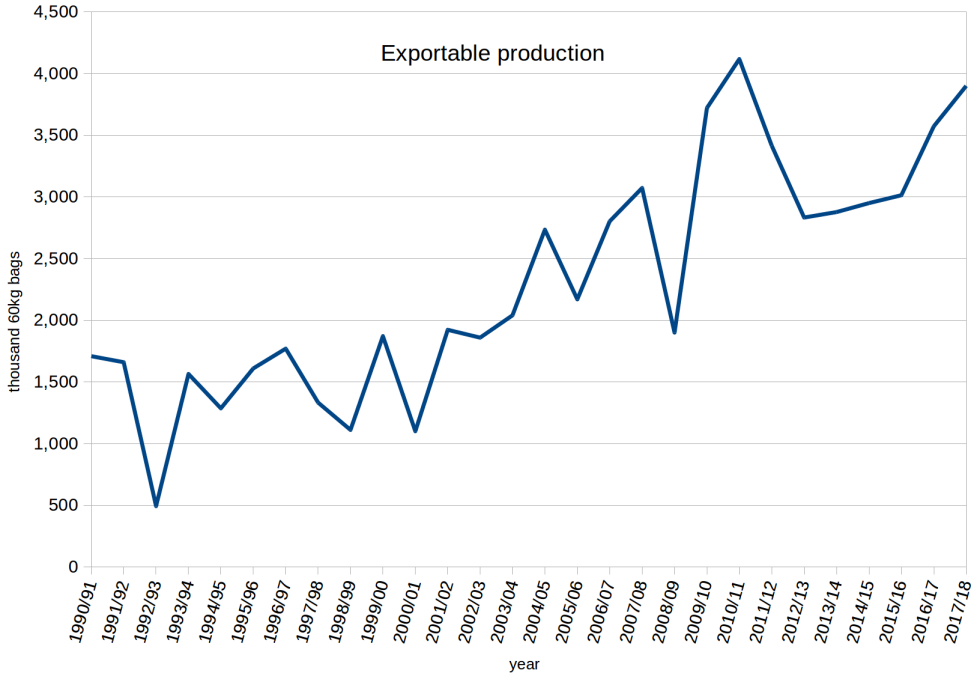


Figure 3.1 – Exportable production of coffee in Ethiopia. ICO (ICO, 2019) is the source of data used to generate the graph

Ethiopia is believed to be the origin of coffee Arabica. In 2018, it was the fifth largest coffee producer country in the world following Brazil, Vietnam, Colombia and Indonesia according to ICO data. Ethiopia mainly exports green coffee beans taking a 3% share among the total export in the world. Although its market share in the world is low, Ethiopia plays an important role in the global coffee economics because of the fine quality and unique taste of its coffee (Daviron and Ponte, 2005). Despite of being the producer of best quality coffee beans, the country has not yet fully exploited the benefits from the crop. Traditional manual farming and

coffee plant diseases are some of the factors that contribute to low yield and production.

Coffee plant diseases cause significant yield and quality losses (Abera et al., 2011; Alemu et al., 2016; Demelash, Kifle, et al., 2018). Major coffee plant diseases in Ethiopia are Coffee Leaf Rust (CLR), Coffee Berry Disease (CBD) and Coffee Wilt Disease (CWD) (Zeru et al., 2012). Zeru et al., 2012 investigated the occurrence of these diseases in four main coffee growing regions of Ethiopia. As demonstrated in Fig. 3.2 CLR has the highest prevalence. The current practice for disease prevention is through variety selection, constant monitoring and selective pesticide application. Monitoring diseases at field level helps to prevent large outbreaks and minimize chemical control.

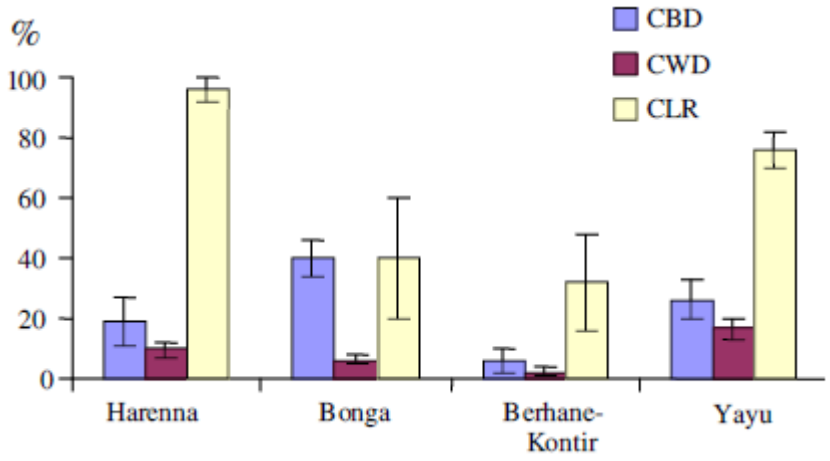


Figure 3.2 – Occurence of the three major diseases in four coffee growing regions of Ethiopia (Zeru et al., 2012)

3.2.2 Machine learning approaches for plant disease identification

Image processing and machine learning techniques have been used for plant disease detection. In this section, we present a comprehensive review of application of machine learning for disease detection. We approached the review grouping the papers in two: researches that applied classical machine learning and researches that used DL methods. Most of the papers that followed traditional approach for image classification start from image acquisition in a controlled environment usually in

a lab settings (Chung et al., 2016; Tian et al., 2012; Waghmare et al., 2016). Then image preprocessing like resizing and background reduction is performed to prepare the image for the next steps. The next step is usually image segmentation to isolate the area of interest (diseased part in this case) followed by feature extraction and selection. Finally, the extracted features are used as an input for the classifier (Waghmare et al., 2016; Camargo and Smith, 2009; Arivazhagan et al., 2013; Al-Hiary et al., 2011; Sannakki et al., 2013). Fig. 3.3 shows the steps in traditional image classification methods.

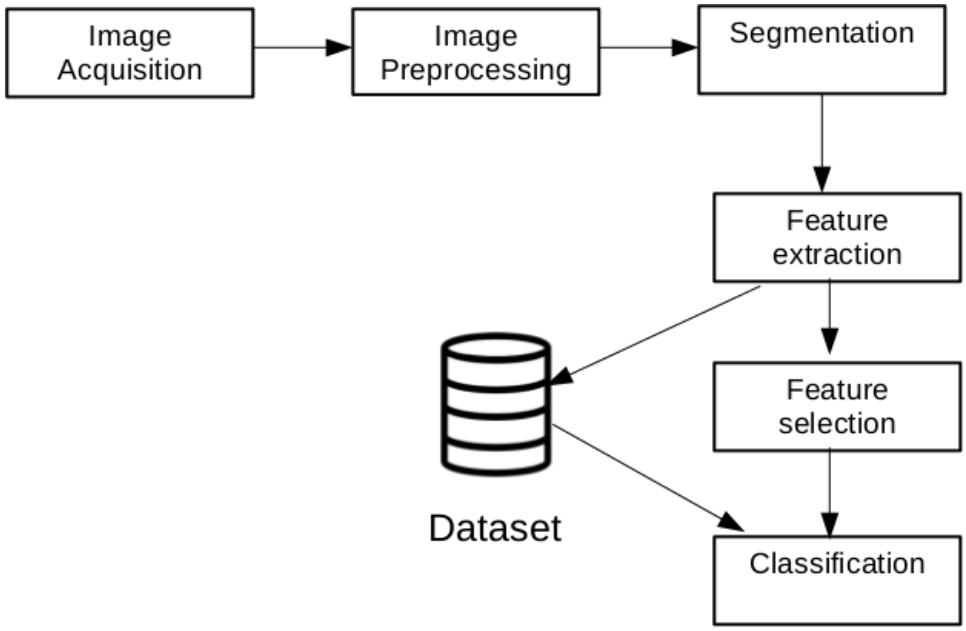


Figure 3.3 – Image processing steps in traditional image classification

Chung et al., 2016 used SVM classifiers to distinguish between healthy and infected rice seedlings. They used genetic algorithm to select essential traits for identification and optimize the SVM parameters. Tian et al., 2012 also used genetic algorithm to optimize SVM classifier to identify apple plant diseases and Principal Component Analysis (PCA) for feature selection. SVM is also used to classify cotton (Camargo and Smith, 2009), beet (Neumann et al., 2014) and grape (Padol and Yadav, 2016; Waghmare et al., 2016) plant diseases. All these papers followed image preprocessing followed by some form of segmentation. Then they extracted one or a combination of color, texture and shape features. Some of them applied feature selection before training the SVM classifier. Ari-

vazhagan et al., 2013 also followed a similar approach to classify common diseases of 30 plant species.

Several researches followed the same steps to detect and classify plant diseases except replacing the classifier by ANN. Awate et al., 2015 used ANN to classify diseases that exist in three fruits, grape, apple and pomegranate. ANN are also used to classify grape plant disease in Sannakki et al., 2013 and Wang et al., 2012, wheat diseases in Wang et al., 2012 and groundnut disease in Ramakrishnan et al., 2015. DTs and RF are also applied as a classifier in few occasions. Sabrol and Satish, 2016 designed a classifier based on DTs to detect and classify tomato plant diseases after the image processing and feature extraction methods. Qin et al., 2016 compared three classifiers i.e., RF, SVM and KNN for identification of alfalfa leaf disease.

Use of image processing and feature extraction requires expert knowledge in the problem area. Moreover, the extracted features varies based on the type of the methods used leading the overall performance of the classification to depend heavily on the underlying predefined features. However, recently, DL is achieving exciting results minimizing the need of preprocessing and feature extraction and also gaining the attention of more researchers (Reyes et al., 2015). Although DL is developing rapidly with the advancement of computational resources and algorithms, its introduction to agricultural applications has only begun in the last few years, and also to a limited extent.

Almost all of the research that uses deep learning for plant disease diagnosis applied CNN for plant disease diagnosis (Ferentinos, 2018; Fujita et al., 2016; Ma et al., 2018; Picon et al., 2018; Mohanty et al., 2016). Mohanty et al., 2016 used 54,306 images of diseased and healthy plant leave images from a publicly available dataset and applied transfer learning on off-the-shelf trained models AlexNet and GoogleNet to 14 crop species and 26 diseases (or absence there-of). Their trained model achieved 99.35% accuracy on the held-out test set however when tested on a set of images taken under conditions different from the images used for training, the model's accuracy is reduced substantially, to just above 31%. This is a well known problem in deep learning where a model's generalization ability is highly affected when tested using data even with a slight variation from the one used for training.

To mitigate the effect of overfitting due to limited dataset and also to achieve good performance at the same time, Ferentinos, 2018, Sladojevic et al., 2016 and Durmuş et al., 2017 followed transfer learning approach on the already trained models like AlexNet, GoogleNet, VGG and ResNet. B. Liu et al., 2018, Ma et al., 2018 and Fujita et al., 2016 designed a new CNN architecture and use intensive data augmentation to classify apple and cucumber diseases respectively. Ma et al., 2018 compared the performance of their designed model with traditional methods (RF and SVM) and with AlexNet for cucumber plant disease classification. The AlexNet model out performed both approaches.

Despite the capability of CNNs to extract features from the input image without the need for extra image processing requirements, Picon et al., 2018; Ma et al., 2018 and Lu et al., 2017 applied some form of segmentation on input images. Picon et al., 2018 applied image segmentation instead of down sampling the input images as done by many DL models to reduce the number of network parameters thus reducing the number of training images. The justification for the need to apply image segmentation is early symptoms of disease (usually small in size) may disappear while the image is reduced in size, making it difficult to detect the disease. Their approach improves the performance of the model better than the classical methods of machine learning as well as when DL is trained with full images (unsegmented images).

Ma et al., 2018, on the other hand, segmented only the diseased part of the leaf before feeding it to CNN to classify cucumber plant disease. Applying image segmentation algorithms, the symptom images were segmented from the leaf and the clutter background. Their model achieved a performance accuracy of 93.4% which performed better than the conventional classifiers. But there was no performance comparison made with a model that uses full images, to see the effect of the segmentation.

One of the limitations of using DL for agricultural applications is the lack of publicly available datasets for researchers to work with. Even for the existing datasets for plant disease detection, most of the images are collected in controlled environments which limits the reliability of models designed using these datasets for real world situations. Due to this, researchers are forced to collect the data by themselves which sometimes can be expensive especially when experts are needed to label the data. From the papers we surveyed, only three papers used images down-

loaded the publicly available database (Mohanty et al., 2016; Ferentinos, 2018; Durmuş et al., 2017). For the detailed review of application of DL in agriculture, refer Kamilaris and Prenafeta-Boldú, 2018.

3.3 Crop quality grading

Crop quality grading is another area of agriculture where machine learning is applied. Quality grading is a post-harvest processes that consists of classifying crops into categories based on quality. Color, shape, size and texture are often used as attributes for grading. In this section, we will present related works on crop grading. We will also present the manual grading of raw coffee beans.

3.3.1 Grading using machine learning

Similar to disease identification, most of the researches in crop grading used the traditional image classification approach (Aran et al., 2016; Kumar et al., 2017; Pandey et al., 2014). They used image preprocessing and segmentation followed by feature extraction. Color and texture features are widely used in the papers. After feature extraction, Semary et al., 2015 applied PCA for dimensionality reduction to design SVM based tomato fruit grading. The authors compared the model's performance with the neural network model and the SVM performed better. Olaniyi et al., 2017 proposed a model to grade banana fruit using texture based features. They designed two classifiers using SVM and ANN and compared the result. Here also the SVM resulted in better classification performance. Kumar et al., 2017 and Aran et al., 2016 proposed a machine learning model to grade pomegranate and cashew fruits respectively. Aran et al., 2016 compared the performance various models (RF, LR and ANN) for cashew fruit grading.

Oliveira et al., 2016 and Faridah et al., 2013 proposed image processing and ANN based models to grade raw coffee beans. Oliveira et al., 2016 extracted only the color features for classification. While color is one of the attributes for manual grading, as we will explain in the next section, it is not the only deciding factor. Both the authors prepared their own dataset capturing the images of the beans in a controlled environment. As shown in Fig. 3.4, camera distance, illumination and size of images were adjusted during image acquisition. While this approach may provide high training accuracy, it will not be applicable for real situations.

Research also used image processing and machine learning to classify beans according to their geographic origin (Turi et al., 2013) and detect defective coffee beans (Ayitenfsu, 2014; Pinto et al., 2017). All the three papers took the image of a single bean in a controlled environment for classification. This approach may be applied for sorting beans before grading but does not provide full information to grade beans.

Figure 3.4 – Image acquisition devices used by a. Faridah et al., 2013 and b. Oliveira et al., 2016

We have noticed there is very few work in crop grading using machine learning especially DL. From all the papers we found during our survey only one used CNN (Pinto et al., 2017). One of the reasons for the lack of research in the area is the availability of enough dataset since labeling/-grading requires experts specifically trained for a particular crop, it makes collecting annotated data expensive.

3.3.2 Manual grading process

Although coffee is traded globally, there is no a single agreed method of coffee grading among the top producing countries of the world. For instance countries like Indonesia base grading on the amount and type of defects present in the beans (Faridah et al., 2011) while other countries like Ethiopia use the physical characteristics (size, color, shape and uniformity of the beans), defects, presence of foreign objects as well as cup taste. We will see only the manual grading process in Ethiopia.

The current practice of coffee grading is conducted manually based on a classification into four basic groups by type and market: washed beans for the domestic and international markets, and unwashed beans for each of these markets (Figure 3.5). Washed and unwashed are processing types. Processing refers primarily to the method of removing the skin,

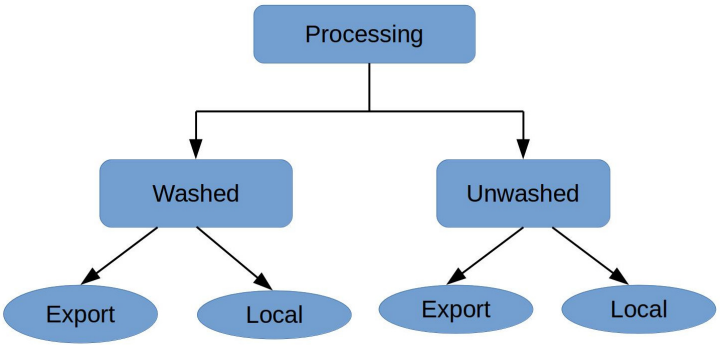


Figure 3.5 – Grading categories based on the way of processing and market. Experts use different criteria for each of the four categories

pulp, and parchment from the outer layers of the coffee cherry, to reveal the green coffee bean underneath. In the washed processing water is used to transport the beans in the processing machine as well as during fermentation stage (refer to Fig. 3.6a). The process starts immediately after the berries are picked from the tree. The beans go into fermentation process to remove the remaining sugary and sticky cover. Finally, the coffee is taken to dry in sun. This operation is the key difference between the unwashed and the washed methods, since in the washed method the pulp of the fruit is separated from the beans before the drying stage. The parchment remains attached to the coffee beans as shown in Fig. 3.6b. The coffee beans produced this way results in a coffee that is cleaner and brighter.

Figure 3.6 – Washed coffee processing a. farmers washing the beans to remove the pulp b. washed and dried beans

Dry process, also known as unwashed or natural coffee, is the oldest method of processing coffee. The entire cherry after harvest is first cleaned and then sun dried by placing on tables or thin layers. The beans

are dried with all of their layers intact (refer to Fig. 3.7a) including the coffee cherry and mucilage. As the cherries dry, they are raked or turned by hand to ensure even drying (refer to Fig. 3.7b). Once the process is completed, sacks of dried cherries are taken to a hulling station for the removal of the outer cherry. Unwashed processed green coffee beans often have a yellowish or orange-like tinge to them.

Figure 3.7 – Unwashed coffee processing a. cherries dry with all covers intact b. farmers shuffling the beans for even drying

Since the type of processing employed affects the physical appearance and quality of the beans, different criteria is used for each group to assess the grade of the beans. For example, the presence of beans covered by husk (refer to Figure 3.8) is a criterion for unwashed categories but not for the washed categories. The experts (cuppers) assign scores for the raw coffee beans through one or a combination of two methods: Raw Quality and Cup Value (Cup Test). The raw quality is evaluated by examining the physical properties of the raw coffee beans. It entails characteristics like defects, shape and make, color, and odor. The cup value is determined by roasting and tasting sample coffee for acidity, cleanness and flavor. The final score is the sum of the raw value and cup value. The grade is then decided based on discretizing the score using intervals for each grade.

From each coffee shipment, 3kg representative sample coffee is taken from a 10-15 ton delivery. The sample is then coded and tested for screen size and moisture tests. If it passes these tests, out of the 3kg, 300g is used for raw evaluation. The remaining sample is divided for roasting, future reference and client's display. The sample for raw evaluation is sorted manually by hand picking to separate each defect. A score is then

Figure 3.8 – Some defects that affect the grade of coffee a) broken beans b) beans damaged by insects c) beans covered by husk d) unmatured beans e) foreign objects f) unmatured and broken

given based on the ratio of defects to the entire sample or using the number of defective beans present. Some of the defects are shown in figure 3.8. This accounts for 20% of the total score. Then, each cupper works individually on each sample and record his opinion for every criteria. After every expert finished examining the sample, the final value of each attribute is decided by discussion among the experts. If there is a difference on the points, they will convince one another and elaborate their reasons to let their respective points come up to uniform final decision. The final score will be given by an overall agreement of all the experts. The score is converted to the grade value based on in which range it lies. For example coffee beans whose score is in the range 85 and above will be classified in grade-1 while beans with a score in the range 75 and 84 will be classified as Grade-2 coffee. Usually at least five experts should exist in the lab and the number of experts must be odd to make it easier to break the tie in case if they were not able to convince each other. This makes the experts idle for hours even days when unforeseen circumstances happen forcing even one of them to be absent from work, increasing the waiting time of the arrival beans to be graded.

3.4 Publicly available plant datasets

In this section, we present the most commonly used publicly available datasets.

PlantVillage (University, 2019) is a research and development unit of Penn State University. It is built on the premise that all knowledge that helps people grow food should be openly accessible to anyone on the planet. It contains images of many crops and their disease labels.

LifeCLEF contains informations of the identity, geographic distributions and uses of the plant. The dataset exists since 2014 and evolving every year with data added from voluntaries worldwide (LifeCLEF, 2019).

MalayaKew dataset Contains single leaf images of 44 plant species for plant identification tasks (Lee et al., 2015).

Flavia dataset is another plant identification dataset with images of a single leaf from 32 plant species (Wu et al., 2007).

3.5 Conclusion

This chapter presented the state-of-the-art in plant disease identification and crop grading. Most of the models proposed for both tasks use ANN, SVM, DTs, and etc., on top of preprocessing and feature extraction. There is also a shift towards DL techniques because of its high accuracy and its ability to extract feature automatically. But the shift is not as fast as in the other areas of application due to lack of publicly available data. We have also seen the importance of coffee on the world economy especially on producing countries. Despite its importance there are only few attempts to assist the production and processing steps with modern technologies. The manual grading of coffee beans in Ethiopia is also presented.

Part II

Model design for coffee grading and disease identification

CHAPTER 4

4. COFFEE DISEASE DETECTION FROM IMAGES OF THE PLANT

Part of this chapter was published in the 31st International Florida Artificial Intelligence Research Society Conference (Wallelign, Polceanu, and Buche, 2018)

FLAIRS-31

In this chapter, we present a design of a CNN model to classify coffee plant diseases using only few images of the plant or parts of the plant. We use 562 images of coffee plant of four classes (the three common diseases and healthy). In chapter 2, we have seen how proper and large dataset is important to design an accurate classifier and the algorithms and techniques that are used to solve these issues. Because the amount of data that we have is small compared to the amount required to train DL models, we will implement some of the methods like data augmentation and transfer learning to design the model. First the model will be designed using another plant dataset, then we will use transfer learning to transfer the learned features to classify coffee plant disease.

4.1 The coffee plant disease dataset

Some of the images in the coffee disease dataset are downloaded from trusted internet sources. But these are small in number to train any model due to that we captured additional images by ourselves on coffee

farms of Ethiopia using mobile devices. We did not incur any conditions when capturing the images. Then the images are labeled by researchers at coffee research institute of Jimma University. This way a total of 562 images were collected. But collecting additional data was not possible due to the seasonal nature of the diseases and we could not find archived data in research institutes in the country. We can see from Fig. 4.1, the images are representative of the ones present in the natural environment.

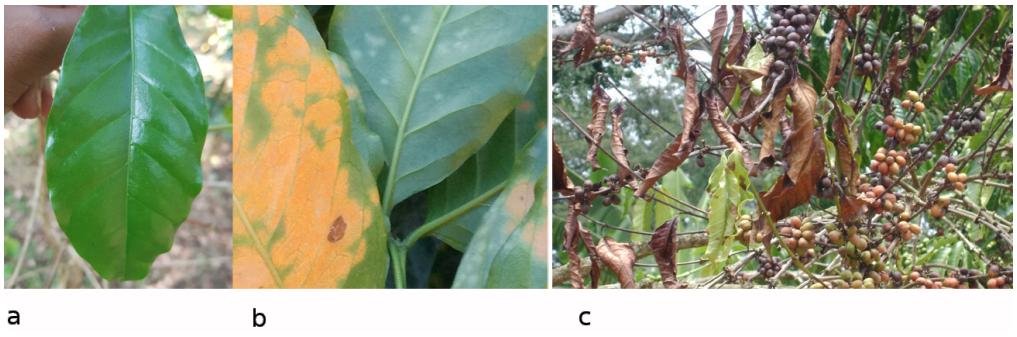


Figure 4.1 – Images from the coffee disease dataset. a. Healthy b. Coffee rust c. Coffee wilt

4.2 Transfer learning

As discussed in chapter 2, transfer learning is often used when there is no enough data to train the model from scratch. Among the two transfer learning approaches, we follow the develop model approach and design a model using another plant disease dataset and apply transfer learning. We selected this approach because it is easier to fine-tune the learned features on a similar task than generic features. After a thorough investigation of the publicly available datasets, we found a close similarity between the shape of the leaves and the appearance of disease symptoms between coffee and soybean plants. Therefore, soybean plant will be used to design the source model. The next section discusses the soybean plant dataset and the techniques used to design the model.

4.2.1 The soybean disease dataset

The dataset for the experiment is downloaded from the PlantVillage database which contains different plant leaf images and their labels. It

contains a collection of images taken both in the natural and controlled environment. A dataset containing 12,673 leaf images of four classes including healthy leaves is downloaded. The number of samples per class of the dataset is summarized in Table 4.1.

No.	Type of Disease	Number
1	Healthy Leaf	6234
2	Septorial leaf blight	3565
3	Frogeye leaf spot	2023
4	Downy Mildew	851
Total		12673

Table 4.1 – Number of samples per class of the soybean disease dataset

Fig. 4.2 shows few samples from the database. Since the images were taken in the uncontrolled environment the different lighting condition and background in the training images may bias the neural network. To test this hypothesis, the experiment was also performed using the grayscale and the segmented version of the database. Sample images of the gray and segmented leaf images are shown in Fig. 4.2 and Fig. 4.3 respectively.

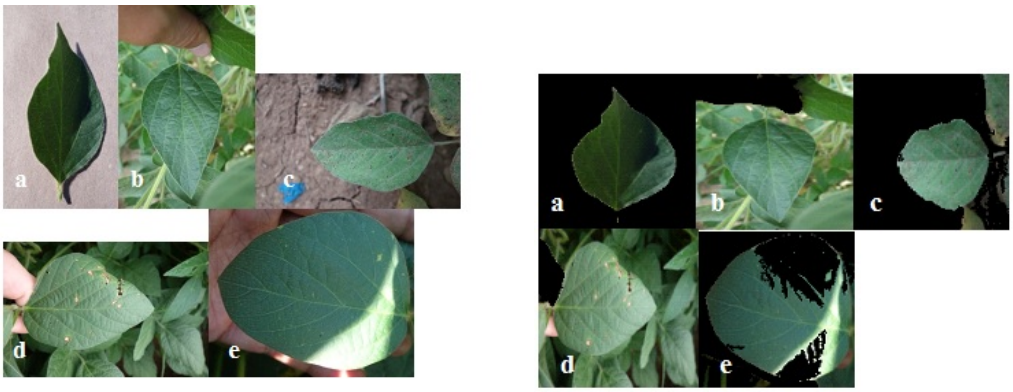


Figure 4.2 – Sample images from the database (Left) and their segmented version (Right) a) healthy leaf image taken under a constant background b) healthy leaf image taken under uncontrolled environment [c-e] leaf images from a plant affected by: c) septorial leaf blight d) frogeye leaf spot e) downy mildew

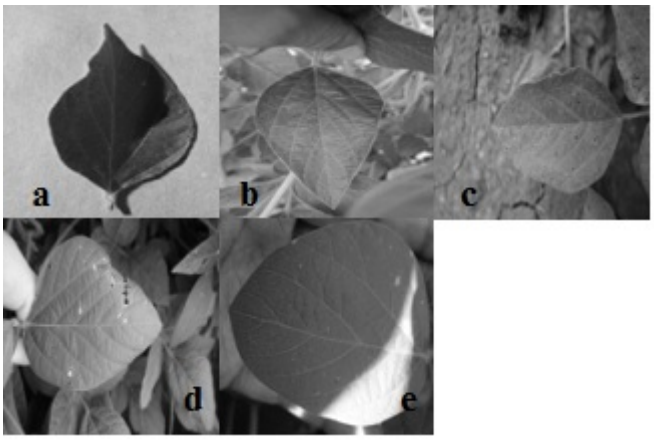


Figure 4.3 – Sample grayscale images size 128x128 pixels a) healthy leaf image taken under a constant background b) healthy leaf image taken in uncontrolled environment [c-e] leaf images from a plant affected by: c) septorial leaf blight d) frogeye leaf spot e) downy mildew

4.3 Experimental results

All the models are implemented in Keras deep learning framework (Chollet et al., 2015) on a single GTX 1070 GPU. The dataset for the training are prepared by resizing the images originally at different resolution to 128x128 pixels and scaling intensity values between [0,1]. The dataset is then divided 70% for the training, 10% for validation and 20% for testing. The segmented and grey-scale versions of soybean dataset as well as the coffee disease dataset are prepared in a similar fashion.

4.3.1 Soybean plant disease model

CNN architectures vary with the type of the problem at hand. We designed the model by trial and error by training several models of varying architectures and network parameters like learning rate, kernel size, filter size. We used ReLU activation function after every convolutional layer, since it is proven to result in faster training (Krizhevsky, Sutskever, et al., 2012). The models are trained using Adam optimizer with batch size of 100 for 1000 epochs. The results obtained for the three versions of the dataset (color, grey-scale and segmented) is summarized as shown in Table 4.2 below.

	Architecture	Validation accuracy	Test accuracy
Gray scale	**[3X3, 4X4]**	**77.60%**	**78.74%**
	[5X5, 5X5]	70.20%	70.07%
	[3X3, 4X4]	77.20%	78.67%
	[3X3, 2X2]	77.60%	77.87%
Color	**[3X3, 4X4, 1X1]**	**89.30%**	**88.20%***
	[3X3, 2X2, 2X2]	89.50%	86.90%
	[3X3, 4X4, 3X3]	89.90%	88.00%
	[3X3, 4X4]	88.00%	85.50%
	[3X3, 2X2]	87.30%	85.30%
Segmented	[5X5, 3X3]	87.40%	86.00%
	[3X3, 4X4]	87.60%	85.90%
	[3X3, 2X2]	87.00%	85.50%
	[3X3, 4X4, 3X3]	**88.30%**	**87.50%**

* Base model

Table 4.2 – Classification result from different models

As we can see from the result, the classification accuracy of the color images is better than the gray scale and the segmented images. This shows that color is an important feature for classification of plant diseases. The model that provides good classification accuracy has three convolutional layers. We used this model as a base model for further optimizations. The base model consists of three convolutional layers each followed by ReLU and maxpooling layers. Finally two fully connected layers are added for classification.

The first convolutional layer filters the input image with 32 kernels of size 3x3. After maxpooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is given to softmax function which produces a probability distribution of the four output classes. The architecture of the proposed model is shown in Table 4.3.

Layer	Type	Filter Size	Stride	Output size
L1	Conv	3x3	1	128x128x32
	Pool	2x2	2	64x64x32
L2	Conv	4x4	1	61x61x64
	Pool	2x2	2	30x30x64
L3	Conv	1x1	1	30x30x128
	Pool	2x2	2	15x15x128
L4	MLP with 512 nodes		-	-

Table 4.3 – Architecture of the proposed model

The graphs of the training accuracy versus validation accuracy of the base model, in Fig. 4.4, show that the model is overfitting. To overcome overfitting, we run the experiment again adding data augmentations together with regularization techniques like dropout and L2 regularization. Since the dataset is too small when compared to the total number of trainable parameters of the model, the first experiment we did is to increase the training data by rotating, flipping, rescaling of the images. The result obtained when using data augmentation is shown in Fig. 4.5

The result shows that data augmentation alone solves overfitting significantly. It also increases the validation accuracy to 98.82%. This explains the original dataset was too small compared to the total number of trainable parameters of the model. To investigate the effect of regularization method used on the model, dropout with probability of 0.5 and L2 regularization were separately included to model. The result, summarized in Table 4.4, shows both methods slightly improved the performance of the model and dropout regularization resulting in a higher performance than L2 regularization. Therefore, the model with dropout will be used as a

Model	Validation accuracy	Test accuracy
Base model with aug. and dropout	**99.21%**	99.32%
Base model with aug. and L2 regularization	98.62%	98.73%

Table 4.4 – Effect of dropout and regularization

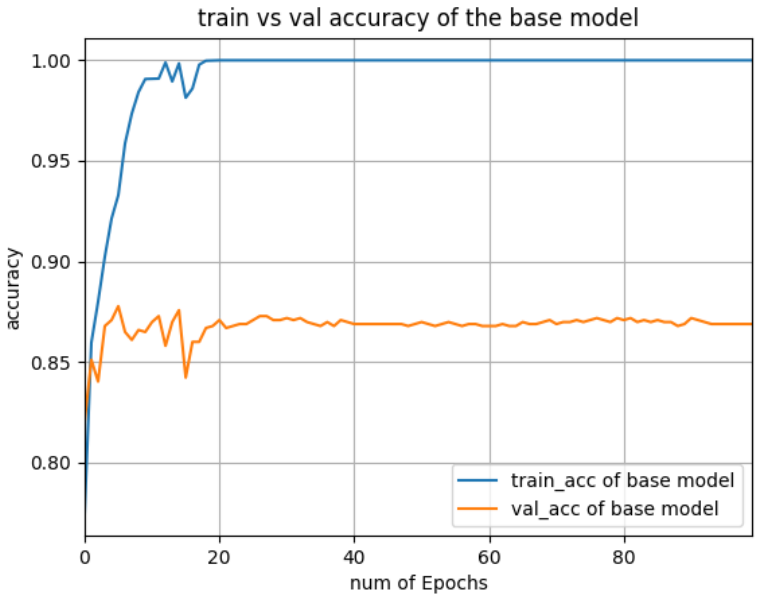


Figure 4.4 – Training vs validation accuracy of the base model

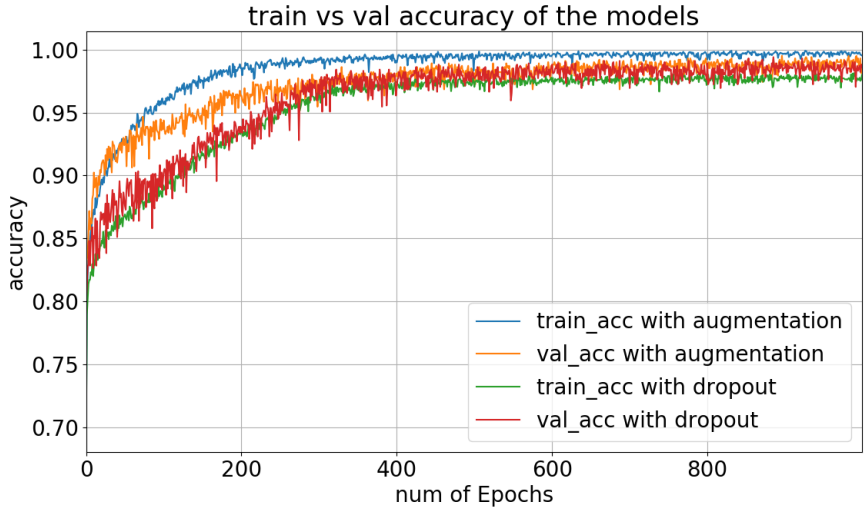


Figure 4.5 – Effect of using data augmentation and dropout on the performance of the base model.

source model to apply transfer learning. The classification metrics of the proposed model is summarized in Table 4.5.

	Precision	**Recall**	**f1-score**	**Support**
Healthy	1.0	1.0	1.0	1228
Septorial	0.99	1.0	1.0	718
Frogeye	0.99	0.97	0.98	407
Downy Mildew	0.98	1.0	0.99	182
Avg/total	0.99	0.99	0.99	2535

Table 4.5 – Performance Metrics of the best model

In addition to checking the training and validation accuracies to see how well the model is performing, we have also visualised the output of each layer in response to a given input. Figure 4.6b. shows the visualization of the filters in the first activation layer in response to the image in 4.6a during the forward pass. If we look at the filter outputs, we can see that not all filters are activated for this input image. Also the filters responded differently; some acted as edge detectors, some responded only for the particular region of the image like the discolored parts of the leaf.

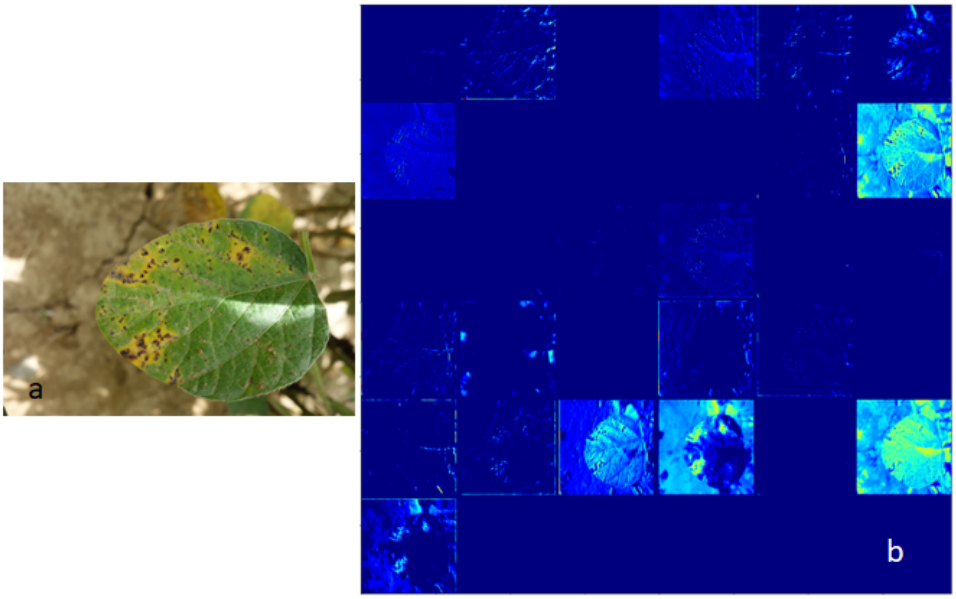


Figure 4.6 – Visualization of the filters in the first activation layer

4.3.2 Coffee plant disease model

Once the source model is designed for a similar task, the next step is to transfer the learned features either using them as a starting weights and retraining the whole model using the current data or keeping the features and tuning only the classifier. Retraining the whole model is ideal when the extracted features are generic and it is required to make them specific for the current task. This approach works only if there is sufficient data for the current task. We experimented with both approaches and compare the results.

Model design

Since both datasets have equal number of outputs, there is no need to change the architecture of the source model. A second architecture is designed to reduce the number of trainable parameters by replacing the last fully connected layers with 1x1 convolutional layer with number of kernels equal to the number of nodes in the fully connected layer. This approach serves as regularization technique by significantly reducing the complexity of the model (Springenberg et al., 2014). The modified architecture, referred as allConv, has a structure as shown in Table 4.6. After L5 Global average pooling is used.

Layer	Type	Filter Size	Stride	Output size
L1	Conv	3x3	1	128x128x32
	Pool	2x2	2	64x64x32
L2	Conv	4x4	1	61x61x64
	Pool	2x2	2	30x30x64
L3	Conv	1x1	1	30x30x128
	Pool	2x2	2	15x15x128
L4	Conv	1x1	1	15x15x512
	Pool	2x2	2	7x7x512
L5	Conv	1x1	1	7x7x4

Table 4.6 – Modified architecture (allCnn) replacing fully connected layers by convolutional layers

Training

From the 562 images of coffee plant, 77.2% of the images are from the healthy class. Cost sensitive learning is used during training to prioritize the inputs from the under represented classes using the weights calculated by the sample distribution in the dataset. Higher weights are used for classes with few samples. Using the weights from the source model as initial weight, both the source model and allCnn model are trained in two ways:

1. Retraining only the last layer
2. Retraining the whole model

Table 4.7 shows the experimental results of the models for each training approach. Retraining only the last layers resulted in better performance for both models. And the performance of 1x1 convolution is better no matter the training approach. One of the factors for the gain in performance is because it resulted in a simpler model (no. of parameters < 10% of the source model). We have also observed the fine-tuned base

Model	Layer retrained	Test acc.	F1 score
Source model	All	83.92%	81%
Source model	MLP	86.61%	85%
allConv	All	88.39%	88%
allConv	L4 & L5	90.18%	90%

Table 4.7 – Comparison of transfer learning approaches for the coffee disease dataset

model always predicts between the two dominant classes though class balancing was used during the training.

4.4 Conclusion

In this chapter, convolutional neural network is implemented to detect and classify coffee and soybean plant diseases. We have shown how transfer learning can be used when working with small datasets. It was possible to design a model with very little data when compared to the amount usually required to train CNNs. The model was trained using the images taken in the natural environment and achieved 99.32% classification ability for the soybean dataset. This shows the promising ability of CNN to extract important features in the natural environment which is

required for plant disease classification. As far as our knowledge, this is the first attempt which use the images taken in the wild environment and achieved remarkable performance. We have also demonstrated that applying data augmentation on the training set improves the performance of the network when the dataset is very small. The effect of dropout and regularization to overcome overfitting also validated. Using the same model and performing transfer learning, it was possible to design a CNN model using only very limited coffee plant disease dataset.

CHAPTER 5

5. Coffee Beans Grading Image Dataset

A proper dataset is crucial to train an accurate and efficient machine learning model. Therefore, data preparation is the first step in designing any machine learning model. This chapter presents the data collection and preparation of the coffee beans grading dataset that we will use to design the model that classifies raw coffee coffee beans into their quality grades. The first thing we did is to search for publicly available datasets that could be used for coffee beans grading. Unfortunately, to the best of our knowledge, there is no such dataset currently available. This led us to collect and prepare the dataset ourselves and offer it to the scientific community. Understanding the properties of a dataset is another important step since it helps us to choose the type of model to implement. This chapter also discusses the properties of the coffee beans grading dataset.

5.1 Data collection and preparation

The images of coffee bean samples were collected at Jimma grading center in Ethiopia. Jimma grading center is one of the nine branches of ECX located north west of Ethiopia near a place where coffee is believed to be originated. We collected the data in two rounds, from December 18, 2017 to January 25, 2018 and from October 22 to November 13, 2018. The 300 gram beans reserved for raw evaluation were used to prepare the dataset. A small portion is taken from the sample, dispersed evenly on an A4 white paper; a picture is captured, then the beans are placed in a separate container. The process is repeated until the picture of all

the beans in the sample is taken. Capturing the images this way prevents the beans from overlapping and results in a sample image that is representative of the whole container. We did not subject any restriction while capturing the images except the whole beans on A4 paper should be captured. All images were captured inside the laboratory, with fluorescent and natural lighting that came through the glass window.

The images were collected in two rounds nine months apart using different capturing devices. The first set of images; Dataset1, was collected during the harvest season resulting in a dataset dominated by samples from the washed category. Whereas, the second set (Dataset2) was collected when the beans from the previous year's harvest were graded which results in more samples from the unwashed category. 1266 and 843 images of coffee beans 12 grades were collected in the first and second round respectively. Table 5.1 shows the number of samples per grade collected for the two datasets.

Grade	**Dataset1**	**Dataset2**	**Total**
Grade-1	115	0	115
Grade-2	614	75	689
Grade-3	84	29	113
Grade-6	56	318	374
Grade-7	116	323	439
Grade-8	43	13	56
Local-1	14	16	30
Local-2	156	19	175
Local-3	30	11	41
Local-4	15	14	29
Local-5A	16	10	26
Local-5C	7	15	22
Total	**1266**	**843**	**2109**

Table 5.1 – The number of samples per class in each dataset. Almost half of Dataset1 is from the washed catagory while about 76% Dataset2 is from unwashed category.

The images in Dataset1 were captured using Samsung s7 edge camera with resolution of 2268x4032 pixels. This set is mainly populated by samples from washed category (G1, G2 and G3) which is characterized by clean and bright beans. Whereas the images in Dataset2 were

captured by iPhone 7 plus camera with resolution of 3024x4032 pixels dominated by samples from unwashed category (G6, G7) with yellowish beans. When trying to fit all the beans on the A4 paper, we were indirectly controlling the distance from the device to the beans. This created a scale difference between the two sets since the cameras have different resolution. Since the collected images have high resolution, after experimenting with different window size, the images are cropped into two along the longer side to increase the number of samples without losing important information.

5.2 Dataset analysis

As discussed in the previous section, the way the data is collected resulted in two sets of data that differ in scale, color and number of sample per class. Sample images from the Dataset1 and Dataset2 are shown in figure 5.1. The effect of these variations is thoroughly investigated to take appropriate mitigating actions during model design.

Physical appearance

During the on-site survey at the grading organization, the experts stated that the physical appearance of the beans has an effect on the cup value since the presence of defects and foreign bodies affect the taste. To test the hypothesis, the correlation between the raw value and cup value of the sample data is calculated. The plot of the raw value versus the cup value is shown in figure 5.2. The correlation coefficient between the raw value and the cup value is 0.87 which indicates the existence of a strong relationship between the two values. Therefore, only considering the physical properties of the beans should be sufficient to obtain a model of the coffee grading process.

Dataset shift

To see if the image collection method introduced any dataset shift, the datasets were visualized using t-Distributed Stochastic Neighbor Embedding (t-SNE). t-SNE is a non-linear dimensionality reduction technique developed by Maaten and Hinton, 2008 and well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. After mixing the datasets and labeling images Dataset-1 and Dataset-2, we performed PCA to reduce the dimension-

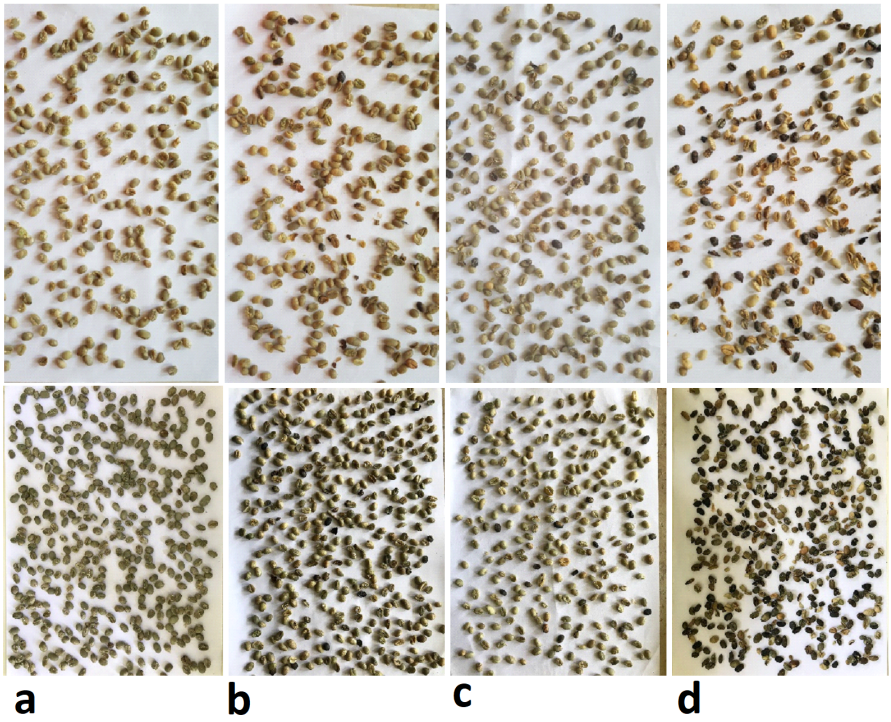


Figure 5.1 – Sample images from the dataset. Top row: images from Dataset1. Bottom row: images from Dataset2. a) Grade-2 b) Grade-7 c) Local-1 d) Local-5A

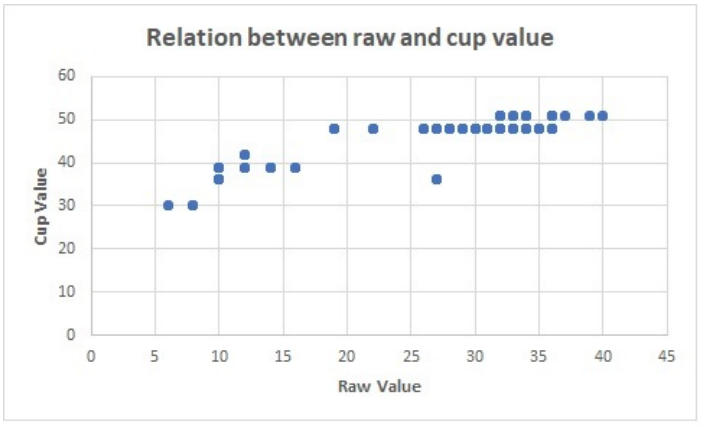


Figure 5.2 – Correlation between the raw value and cup value. Because of the high correlation, we can use only raw value to model the system without loss of much information.

ality of the data before feeding it to the t-SNE. This way we get a clear visualization of the dataset as shown in figure 5.3. As we can see from figure 5.3, there clearly exists a mismatch (dataset shift) in the dataset.

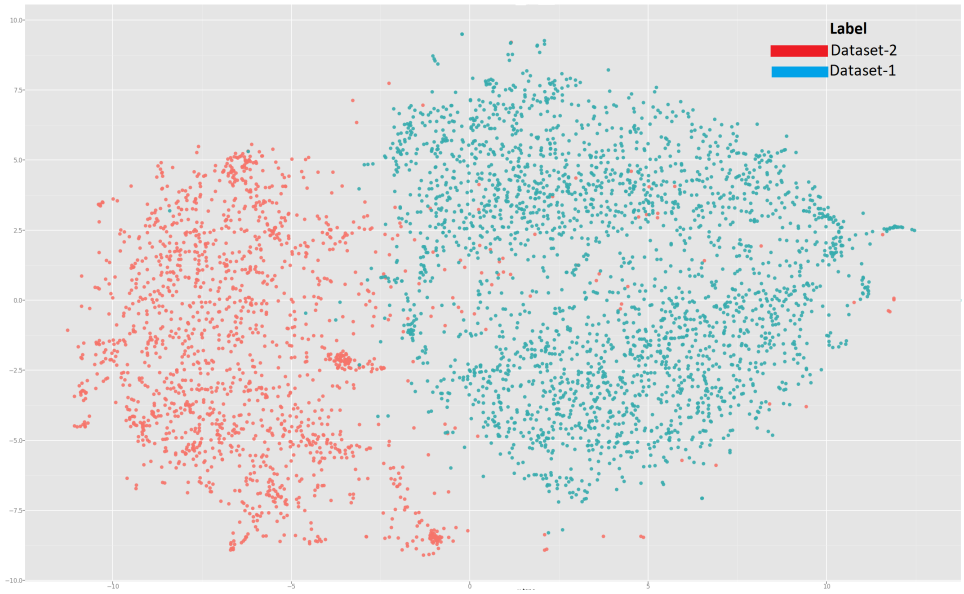


Figure 5.3 – t-SNE plot of the distribution of the images in the datasets after feature reduction is performed using PCA. blue-Dataset1, red-Dataset2

Sample selection bias

The term sample selection bias refers to a systematic flaw in the process of data collection or labeling which causes training examples to be selected non-uniformly from the population to be modeled (Moreno-Torres et al., 2012). In the previous section, we stated that the two datasets were collected during different seasons. Dataset1 was collected during the harvest season which resulted in most of the samples coming from the washed category (G1, G2 and G3). While Dataset2 was collected nine months after, where the beans in the unwashed category (G6, G7) dominated the lab. This introduced a sample selection bias between the two datasets.

Imbalanced dataset

In addition to the sample selection bias between the two sets, looking at Table 5.1, it is easy to see the sample imbalance in the two sets as

well as the total dataset. Grade-2 coffee represents 32.67% of the total dataset while Local-5A and Local-5C each represents about 1% of the dataset. As discussed in chapter 2, this makes designing a reliable model harder favoring the classes with large number of samples. The class imbalance problem is more significant when there is noise in the data. Noise is introduced in our dataset from the image acquiring technique and the labeling error caused by human experts.

Changing environments

The use of two different devices having different protocols and resolutions create domain shift as we discussed in Chapter 2. Color of the beans is one of the attributes used for coffee beans grading. But the camera differences and the post processing performed by cameras makes the same object appear different when captured using different devices. Also, even when using the same device, an object may appear different under varying illuminations. For the human eye, it is easy to identify an object even under such variations; however, for the computer it is a difficult task. What makes this dataset challenging is that there is high inter-set (between Dataset1 and Dataset2) variation while not much intra-set (variations within Dataset1 or Dataset2) variations.

5.3 Possible application areas

The coffee beans grading dataset can be used for computer vision applications such as classification and domain adaptation.

Image Classification: In addition to the grading of coffee beans, the dataset can be used for fine-grained image recognition. The difference between fine-grained image recognition and generic image recognition is: in generic image recognition the target objects are visually quite different (chairs, computers, shelfs,...) but in fine-grained image recognition images are quite similar. For accurate recognition, it is desirable to distinguish them by capturing slight and subtle differences (Zhao et al., 2017).

Domain adaptation: The dataset shift between the two sets makes it suitable dataset for domain adaptation. Domain adaptation is a branch of machine learning which deals with scenarios in which a model trained on a source distribution is used in the context of a different (but related) target distribution.

5.4 Conclusions

The images were collected in two rounds, using different acquisition techniques to investigate the effect of data collection methodology on the performance of a model. We have also investigated the possible issues that exist in the dataset so that it will be possible to address them during designing of the classifier. Due to the variation in the data collection methodology, issues like imbalanced dataset and dataset shift are introduced. The new dataset will be made publicly available for the scientific community following the formats of common vision datasets (L. Deng, 2012; J. Deng et al., 2009). In chapter 6, this coffee beans dataset will be used for designing a model to classify the beans into their quality grades.

CHAPTER 6

6. AUTOMATIC COFFEE BEANS GRADING

Part of this chapter was published in the 27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (Wallelign, Polceanu, Jemal, et al., 2019)

WSCG-2019

6.1 Introduction

This chapter discusses the design of a robust CNN model that classifies raw coffee beans into their 12 quality grades using small datasets which have high data variability. We used the coffee beans grading dataset presented in chapter 5. It contains images of raw coffee beans acquired in two sets using different acquisition technique under varying illuminations which poses a complex challenge to designing a robust model. To design the CNN model, preprocessing techniques were applied to the input in order to reduce task irrelevant features. But adding the preprocessing techniques did not improve the performance of the model for our dataset. We have also used ensemble methods to solve the high variance that exists in networks when working with small datasets. Finally, we were able to design a model that classifies the beans into their quality grades with an accuracy of **89.1%** on the test dataset. The

model's performance is compared with that of classical machine learning models.

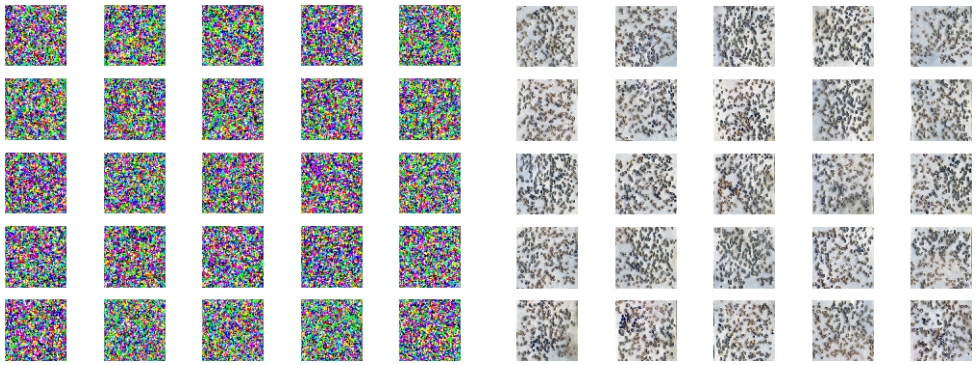
6.2 Preprocessing and data augmentation

To increase the number of training data, in addition to the geometrical transformation for data augmentation, we used GAN to generate additional data. we have also proposed a data preprocessing and augmentation technique to minimize the dataset mismatch that occurred due to the variation in data collection methodology. In this section, we will discuss the implementation of these techniques.

6.2.1 Data generation using GAN

GANs has been used for supervised and unsupervised applications. In chapter 2 we have seen that in generic GAN the discriminator is trained to predict whether the image is from real dataset or from the generator. We implemented semi-supervised GAN where the discriminator learns to predict classes in addition to fake/real (Odena, 2016). The discriminator has now multiple outputs, one binary output for fake/real and a second output to classify between the grades. The second output of the discriminator has thirteen labels i.e, twelve grade classes plus one for real or from generator. All the images from the generator will be categorized in the thirteenth label. We can take the discriminator as a multi-task classifier where one task is to discriminate fake images while the second task is to classify images into their grade labels, having a shared layers for feature detection and separate layers for each task.

We used a sequence of convolutional, dropout and LeakyReLU layers to design the generator and discriminator's shared layers. For the outputs of the discriminator a fully-connected layer with one and thirteen nodes is added for identification of fake/real and classification of grades. The input images are resized to 224x224 pixels. During one iteration of training, half real and half generated images are given to the discriminator. The model is trained for 20000 iterations. The output images from the generator are shown in Fig. 6.1. However, adding these images to the dataset reduced the performance to classify between grades.



a. Initial noise b. Generated images

Figure 6.1 – Images generated a. Image generated at first iteration b. Image generated at the 10000 iteration

6.2.2 Augmentation for color correction

The colors that are present in images are determined by the intrinsic properties of objects and surfaces as well as the illuminant of the scene (Gijsenij et al., 2011). For a robust color-based system, these effects of the light source should be filtered out. This effect, i.e the ability to account for the color of illuminants, is known as color constancy. Many computer vision problems in both still images and videos can make use of color constancy processing as a preprocessing step to make sure that the recorded color of the objects in the scene does not change under different illumination conditions. But most often in deep learning models only intensity normalization is performed on the input images (VGG (Simonyan and Zisserman, 2014), Inception (Szegedy et al., 2015)). This may be enough when working with large datasets with millions of images but not for small datasets which lack in representation. Therefore, we applied a white balancing technique to preprocess the input images for color constancy and compare the result with intensity normalization. Gray-World (Buchsbaum, 1980) color constancy technique is used. It is based on the assumption that the color in each sensor channel averages to gray over the entire image. We chose this algorithm because of its simplicity and fast computation time.

Gamma correction is another operation that digital cameras perform in order to match the human perception of an image. In digital cameras the received signal from the sensors is linearly proportional to the light source that hits the sensor. To account for the non linearity of this process in

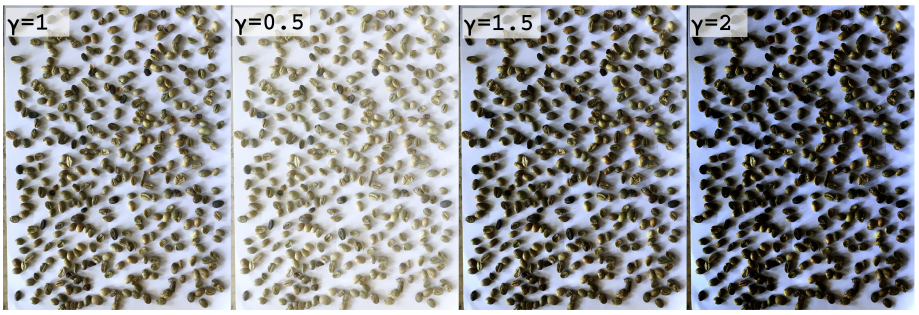


Figure 6.2 – Sample gamma corrected images. Images appear darker as we increase the value of gamma. From left to right gamma = 1 (no change), gamma = 0.5, gamma = 1.5 and gamma = 2.

humans, a non linear transformation function is applied. Sample gamma transformed images are shown in Fig. 6.2. Images appear darker as the value for higher values of gamma. However, this transformation is dependent on characteristics of the display and on the imaging device manufacturer. Therefore, we added a gamma correction augmentation technique during the training of the model.

6.3 Experimental results

6.3.1 Classical machine learning models

First we experimented with classical machine learning algorithms since they are still the go to methods when working with small datasets. Among the classical machine learning algorithms, we experimented with Logistic Regression (LR), Naive Bayes (NB), Linear Discriminant Analysis (LDA), KNN, DTs, Random Forest (RF) and SVM classifiers. We have seen in chapter 2 that for the successful training of these methods careful designing of features is very crucial. Color, shape and texture are selected as the primary cues to quantify the global features of coffee bean images because as stated in chapter 3, the raw value of the manual classification is characterized by these entities. The features are extracted using color histogram analysis in hsv color-space, Hu-moments (M.-K. Hu, 1962) and Haralick Texture (Haralick, Shanmugam, et al., 1973) for color, shape and texture features respectively. Then these features are concatenated to form a single global feature.

Kernel	Gamma	C	mean score	std
Linear	-	1	0.421	0.050
Linear	-	10	0.479	0.049
RBF	0.001	1	0.122	0.196
RBF	0.0001	1	0.125	0.203
RBF	0.001	10	0.237	0.048
RBF	0.0001	10	0.201	0.307

Table 6.1 – Performance score of SVM for different parameters

The optimal parameters for each classifier is selected by performing a grid search from several possible values, then the best parameters obtained for every classifier is used to compare the performance of the models. For example Table 6.1 shows the performance of SVM for different values of kernel, gamma and C. After the parameters for the rest of the classifiers are optimized in a similar fashion, the models are trained using K-fold cross_validation using the optimal parameters for each classifier. As it is shown in the boxplot of the results from the classifiers (Fig. 6.3), KNN out performs all the other algorithms for the coffee grading dataset with mean score of 66.924% F1 score on the training data and 63.626% on the test dataset.

6.3.2 Cascaded convolutional neural network

To mimic the manual grading of coffee beans, a CNN model with a hierarchical structure is designed based on Yan et al., 2015. In cascaded CNN the beans are first classified into their broad category then based on the result the next level classifier predict their grade levels. This arrangement is similar to the way manual grading is performed (see chapter 3). The first part (referred as Category branch in the remaining of the text) learns the common features of the beans and their processing categories; i.e Export washed, Export unwashed, Local washed and Local Unwashed followed by four branches which predicts the particular grade for each category. The block diagram of the proposed model is shown in Fig. 6.4 below. Because the convolutional layers at the beginning of the network serves as feature extractors, the first convolutional layers are shared with all the branches as well as the Category branch. For each Branch 1 - 4 additional convolutional layer is added following the shared layers to extract features specific to its category. Finally for the classification, two fully connected layers are added at the end of every branch.

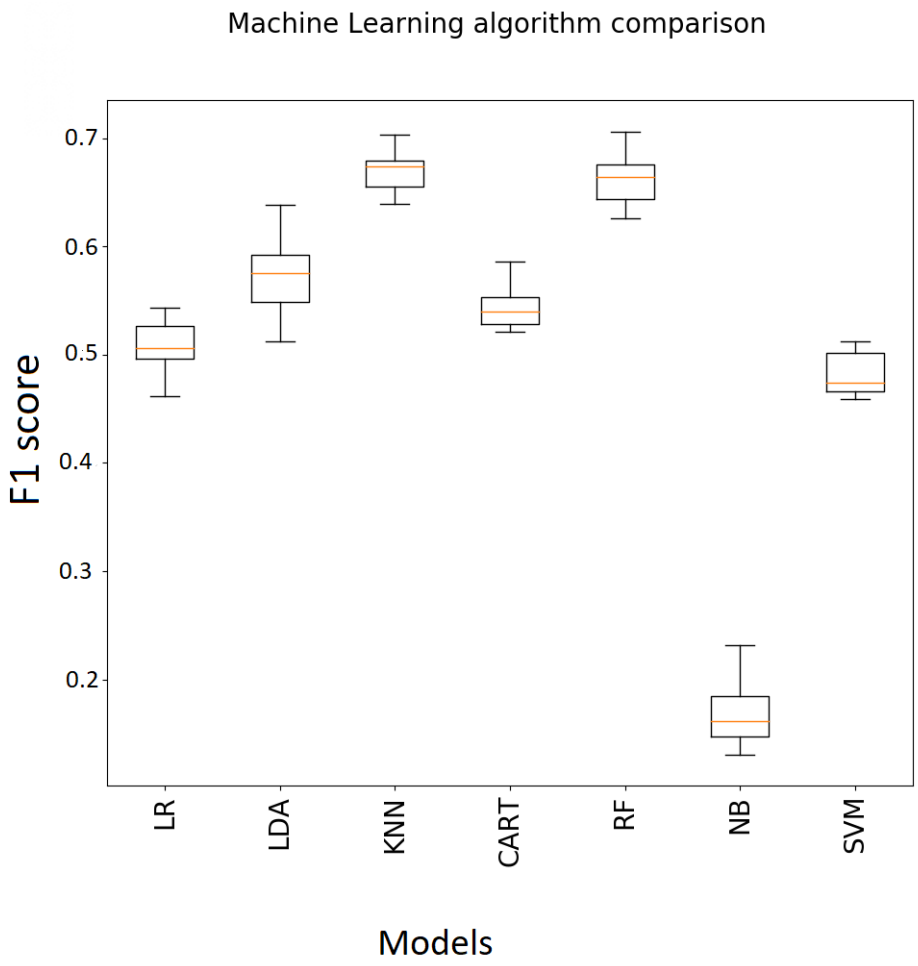


Figure 6.3 – Box plot of the results of the classical machine learning algorithms on coffee grading dataset

The output of the Category branch is the probability distribution of an input being in the four categories. Branches 1 - 4 take the same input image and returns the probability distribution of the image with respect to the 12 grades. The outputs from the four branches are combined and weighted using the argmax of the output of the Category branch to get the final prediction.

Model architecture

The architecture of the model is adapted from the VGG-16 model. Table 6.2 shows the layers in the category classifier. The branches has the same structure except one convolutional layer with 32 3x3 filters is

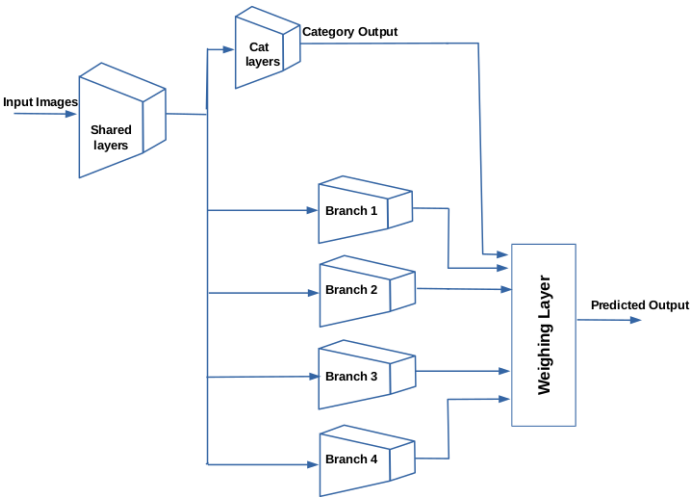


Figure 6.4 – Block diagram of cascaded CNN

added to extract feature for grade classification followed by two dense layers with 512 and 12 nodes respectively. We have also designed the model replacing the fully connected layers by 1x1 convolutions to reduce the network parameters as explained in chapter 4.

Layer	Filters	Size, stride	Remark
Input		224x224	RGB
Conv1	96	7x7,3	Shared
Conv2	96	1x1,1	»
Conv3	192	3x3,1	»
Conv4	192	1x1,1	»
MaxPooling	192	3x3,2	»
Conv5	384	3x3,1	»
Conv6	384	1x1,1	»
MaxPooling	384	3x3,2	»
Dense	256	0.5 dropout	cat.
Dense	4		»
Softmax			

Table 6.2 – Architecture of Category classifier

The network has two outputs corresponding to the category and grade predictions. The training of the model is the same as traditional CNN except the loss now becomes:

$$loss = w\_category * cat\_loss + w\_grade * grade\_loss$$

where $w\_category$ and w_grade are parameters between [0, 1] and cat_loss and $grade\_loss$ are the cross entropy losses for category and grade networks respectively. For our experiment to give emphasis for the loss from the grade output 0.1 and 0.9 are used respectively. The model is trained for 1000 epochs using Adam optimization and learning rate $lr = 0.0001$. We trained the model for 1000 epochs with a stopping criteria of 100.

Results

Using the argmax of the category output as an input for weighting layer for branch selection may affect the performance since the misclassification from the category branch will propagate to the wrong branch. To account for that we have also trained the models using the probability distributions from the category branch as an input for the weighting layer. The results of the experiment are shown in Table 6.3. The model can classify the coarse labels (categories) with good classification performance but it resulted in lower performance than a traditional CNN model with similar network parameters.

Model	Cat. test acc(%)	grade test acc(%)	Remark
proposed	96.54	81.31	argmax
allCNN	95.59	79.16	argmax
proposed	87.86	81.07	probability
allCNN	95.71	71.30	probability

Table 6.3 – Performance output of cascaded CNN models

6.3.3 Ensemble learning

Neural networks are sensitive to initial conditions, both in terms of the initial random weights and in terms of the statistical noise in the training dataset. This stochastic nature of the learning algorithm means that each time a neural network model is trained, it may learn a slightly different version of the mapping function from inputs to outputs, that in turn will have different performance on the training and holdout datasets. A solution

for high variance in networks, while also improving accuracy, is using ensemble learning (Dietterich, 2000). Ensemble learning is training several models varying the initial conditions and then combining the predictions from each model to make one final prediction. We applied two types of ensemble methods, Checkpoint Smoothers (CS) and Checkpoint Ensemble (CE) (H. Chen et al., 2017), to compare the performance between the models as well as to choose the final model. Training DL models takes hours even days depending on the dataset and model complexity. Due to this reason building ensembles by running the models multiple times is not feasible. However, we can work around that by taking the models around minimum validation loss to form the ensemble instead. Therefore, during training each model is saved after every iteration. Then we can select the models using the minimum validation loss and apply either CS or CE. CS averages the weight of few models around the model with minimum validation loss while CE averages the predictions of the models.

Model architecture

Since we have a small dataset, the main focus during the design of the CNN architecture was to obtain a simple model with few trainable parameters. To achieve this goal, we followed two methods. First, to use transfer learning on existing trained model. Therefore, we applied transfer learning on the pre-trained VGG model by removing the last fully connected layers. After removing the fully-connected layers of the VGG, two layers with 1024 nodes and 50% dropout were added followed by an output layer with 12 nodes equivalent to the number of grades. Then we trained only the new fully connected layers by keeping the weights in the other CNN layers the same as the one learned using the ImageNet dataset.

Second, to design a new architecture and compare the result with the one obtained from transfer learning. The architecture of the new CNN model is adapted from the same model that we used for transfer learning except two modifications. Instead of using fully connected layer for the classifier, we used a 1x1 conv layer followed by global average pooling. It is stated that if the image area covered by units in the top most convolutional layer covers a portion of the image large enough to recognize its content then fully connected layers can also be replaced by simple 1-by-1 convolutions (Springenberg et al., 2014). This has a great advantage when working with small datasets by regularizing the network further since it results in a smaller number of model parameters than the

fully connected layer. The second modification is, although using filters smaller than 5x5 pixels is recommended for use in the first convolutional layer, we found through experiment that a 7x7 filter with stride=3 worked better for our dataset.

For the sub-sampling layer we used overlapping maxpooling with stride = 2 and kernel size = 3. For activation layers, using LeakyRelu (alpha = 1/a = 0.001) resulted in relatively faster training and a more stable network than with the ReLU activation function. Therefore, LeakyRelu is used after every convolutional layer except the last one where softmax activation is used to generate the probability of a sample being in each grade. The architecture of the designed model (referred to hereafter as CNN_1) is summarized in Table 6.4.

Layer	Filters	Size, stride	Remark
Input		224x224	RGB
Conv1	96	7x7,3	LeakyReLU
Conv2	96	1x1,1	LeakyReLU
Conv3	192	3x3,1	LeakyReLU
Conv4	192	1x1,1	LeakyReLU
MaxPooling	192	3x3,2	
Conv5	384	3x3,1	LeakyReLU
Conv6	384	1x1,1	LeakyReLU
MaxPooling	384	3x3,2	0.5 dropout
Conv7	950	1x1,1	LeakyReLU
Conv8	12	1x1,1	12 outputs
GlobalAverage			
Softmax			

Table 6.4 – Architecture of the model

Training

The models were implemented in Keras deep learning framework on a single GTX 1070 GPU. The dataset is divided into three sets for training (70%), validation (10%) and testing (20%). All images were resized to 224x224 pixels, intensity values were scaled between 0 and 1. To compare the preprocessing techniques, two version of datasets were prepared. In one, let's call it DS1, images were normalized by subtracting the mean of the training set from each image. The second, DS2, was

prepared using the white balancing color constancy technique. The experiment was conducted on both datasets DS1 and DS2. Each model is trained for 1000 epochs with early stopping, Adam optimization (learning rate = 0.0001) is used with categorical cross-entropy loss function. The performances of the models on DS1 and DS2 for different runs is

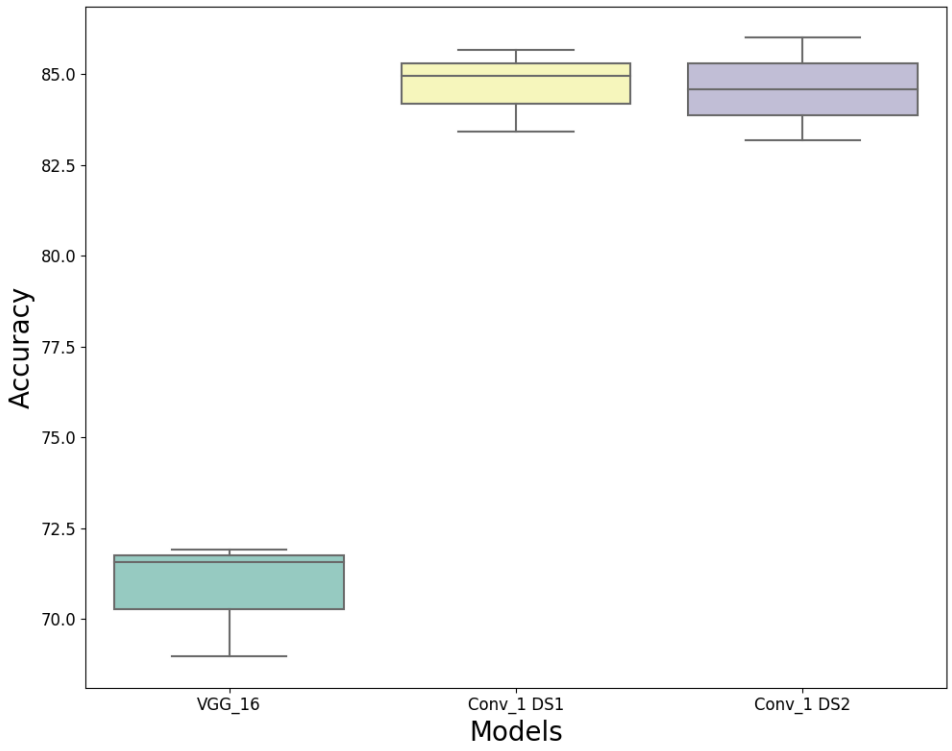


Figure 6.5 – Performance of models for DS1 and DS2. There is high variance on test performance of the models when trained using different initial conditions

shown in Figure 6.5. It can be observed from the plot that training the models starting from different initial conditions resulted on varying prediction performance making optimization and choosing the final model very challenging. The plot also shows when transfer learning is applied on VGG-16 resulted in lower performance than CNN_1 for this dataset. We have also experimented retraining the last two convolutional layers but the performance did not show much improvement compared to the

CNN_1 model.

The models for ensembling are picked based on their minimum validation loss (see Table 6.5 and Fig. 6.5). Hence, models found at run-3 for DS1 and at run-2 for DS2 are used to create group of models. Then, we applied CS and CE on n models around the one found by minimum validation loss (MV) varying the number of models, n, participating in ensemble formation. The test accuracy of the models for $n = 10$, $n = 20$ and $n = 50$ is summarized in Table 6.5.

	CNN_1 (DS1) %	**CNN_1 (DS2) %**	Remark
run-1	84.95	84.25	MV
run-2	85.66	**86.02**	MV
run-3	**85.78**	85.07	MV
$n = 10$	88.40	87.68	CE
	88.39	87.80	CS
$n = 20$	87.80	87.20	CE
	88.51	87.68	CS
$n = 50$	88.51	87.80	CE
	89.1	**88.34**	CS

Table 6.5 – Summary of the experimental results. The model's prediction performance seems better when the dataset is preproccessed using only mean normalization.

We can see from the table, the performance of the model when only intensity normalization is used is higher than the model with color correction preprocessing. But looking at the values, it is difficult to conclude the model is actually better or the difference is simply a mere statistical chance. Statistical significance tests are used to address this problem and quantify the level of confidence or significance in the difference between models. Dietterich, 1998 reviews five statistical tests and recommends McNemar's statistical hypothesis test in cases where there is limited amount of data and when the models can only run once. The McNemar's statistical test is suitable for deep learning models which takes hours even days to train them and re-running is computationally expensive.

The McNemar's test (Everitt, 1992) uses contingency table formed by how the models classify each sample on the test dataset. After the models are trained, each model will classify every sample in the test set and record if the sample is classified correctly or not. The contingency table is determined by filling each cell as shown in Table 6.6.

	CNN_1(DS2) correct	CNN_(DS2) wrong
CNN_1(DS1) correct	a	b
CNN_1(DS1) wrong	c	d

Table 6.6 – Contingency table

The null hypothesis of marginal homogeneity states that the two marginal probabilities for each outcome are the same, i.e. $pa + pb = pa + pc$ and $pc + pd = pb + pd$. The test calculates a p-value based on the null hypothesis, i.e the disagreement of the models is equal. If this value is less than a specific amount (usually 0.05), then the difference is significant. Otherwise, it fails to reject the null hypothesis.

Following the above steps, the contingency table for CNN_1(DS1) and CNN_1(DS2) is calculated as shown in Table 6.7 and the resulting p-value is 0.61 which is higher than the threshold indicating same proportions of errors by the models which fails to reject the null hypothesis. Therefore, the difference between the models' performances is not significant.

	CNN_1(DS2) correct	CNN_(DS2) wrong
CNN_1(DS1)	701	51
CNN_1(DS1)	45	47

Table 6.7 – Contingency table prepared based on models' predictions on the test dataset

As the results indicate, applying color correction preprocessing and augmentation did not bring any improvement on the performance of the model. Also using ensemble methods results in better prediction performance than the model selected based on minimum validation loss. Both the CE and CS models have comparable results. Because CS model performs only one prediction during test time compared to n test time predictions by CE model, it takes less prediction time and computational

resources than CE with equal number of models. Therefore, we chose the model designed using CS ensemble when $n = 50$ trained using DS1 as a final model.

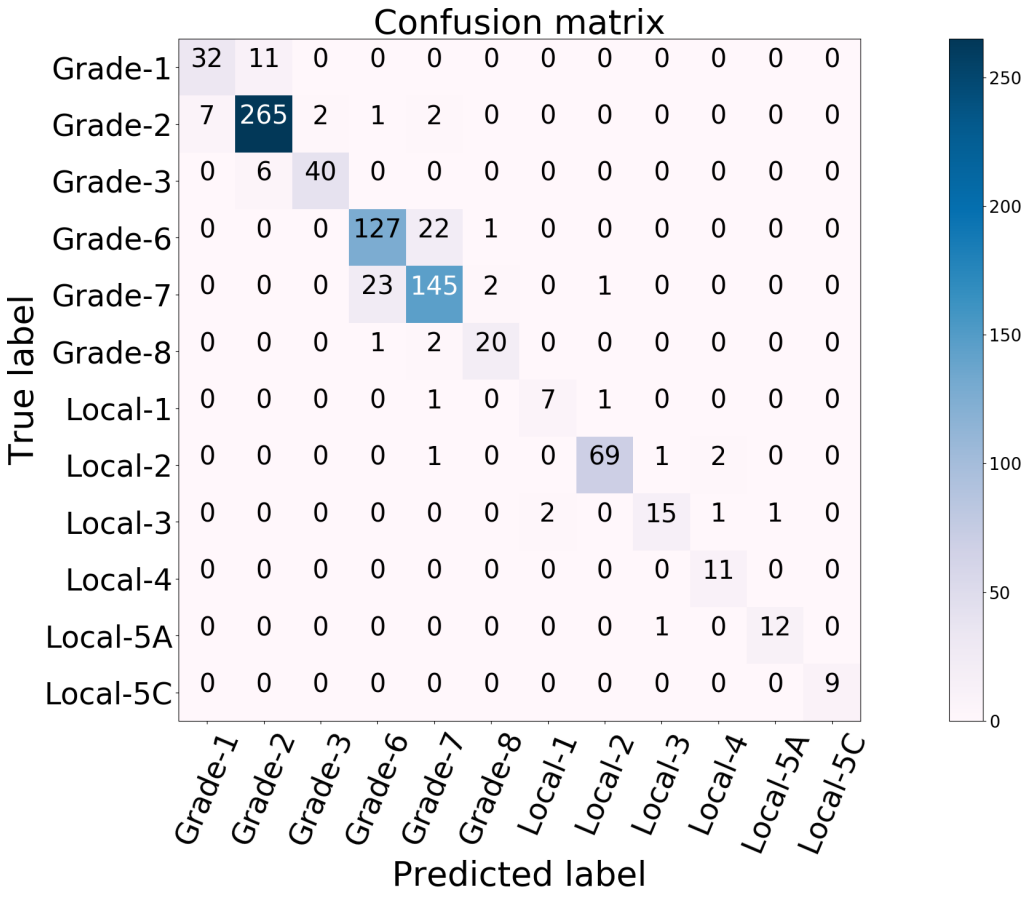


Figure 6.6 – Confusion matrix of the model

The Top-2 accuracy of the selected model is 98.22%. Also the confusion matrix (Fig 6.6) of this model on the test dataset shows most of the misclassifications occur between two neighboring grade values. This maybe due to the fact that in the manual classification the grade value is inferred based on which range the final score lies, i.e for example if the score lies in the range 75 to 84, it will be a Grade-2 coffee and if it lies between 63 and 74 then its grade will be Grade-3. Therefore, two coffee beans with a total score of 75 and 74 will be classified in Grade-2 and Grade-3 respectively even though there is no significant difference in their visual appearances.

6.4 Conclusion

We designed a CNN model to classify raw coffee beans into 12 quality grades and achieved 89.1% on the test dataset. It was also shown that the CNN model can extract features and classify the input images with minimal preprocessing. White balancing the images and adding color correction to the augmentation function did not improve the performance of the model. We have also implemented ensemble methods to design a strong model from several weak-classifiers. In the error analysis, we have seen that most misclassifications occur between two neighbor classes. In the future we will investigate the effect that treating the output as a classification problem rather than a regression problem had on the performance of the model.

During our stay in the grading organization, we were informed that the experts estimate their repeatability between 80 - 85%. Since the model was inferred based on the labels given by the experts, its prediction performance is affected by this error. We did not use any method to address wrong labels due to human errors.

CHAPTER 7

7. Deployment in Mobile Application

Machine learning models are developed and used to simplify tasks in several application areas. For the developed models to be useful and reach many users, one of the most practical approaches is to deploy them in mobile applications. Machine learning is extensively used in mobile applications analyzing targeted user behavior patterns and searching requests to make suggestions as well as recommendations. E-commerce, map, on-line video and audio playing applications are few areas where machine learning is applied. Their increasing availability and computation performance made mobile devices a suitable candidate to deploy the models designed in chapters 4 and 6. This chapter discusses the designing and implementation of a mobile application that we named 'InfoBuna' for coffee plant disease identification and raw beans grading. Buna is an Amharic (Ethiopian language) term for coffee.

7.1 Requirements of mobile application

The target users of the 'InfoBuna' app are coffee farmers of Ethiopia who mostly live in rural parts of the country with no or little education. 'InfoBuna' should have the following features:

Android application:
Despite Ethiopia being among the countries with the lowest number of smart phone users in the world, according to Ethio telecom's (the only service provider in the country) annual report of 2018/19, there are 49 million devices in the country among which 93% are android and 2.7%

are ios devices. Therefore, to reach a large number of users, the models are implemented in android mobile app.

Native application:

Before designing the App, it must be decided whether to perform the prediction on the server or on the device. Due to the limited availability of the internet and data transfer overhead, it is better to load the trained model and perform the prediction on the device. While this has the advantage of minimizing data transfer cost, it doesn't allow to gather important data that could be used to improve the models.

Simple user interface:

The app should have a simple user interface with few interaction with the user to minimize cognitive load of the users. It needs to include functionalities to capture, open and save an image plus a window to display prediction results as shown in a use case diagram of figure 7.1.

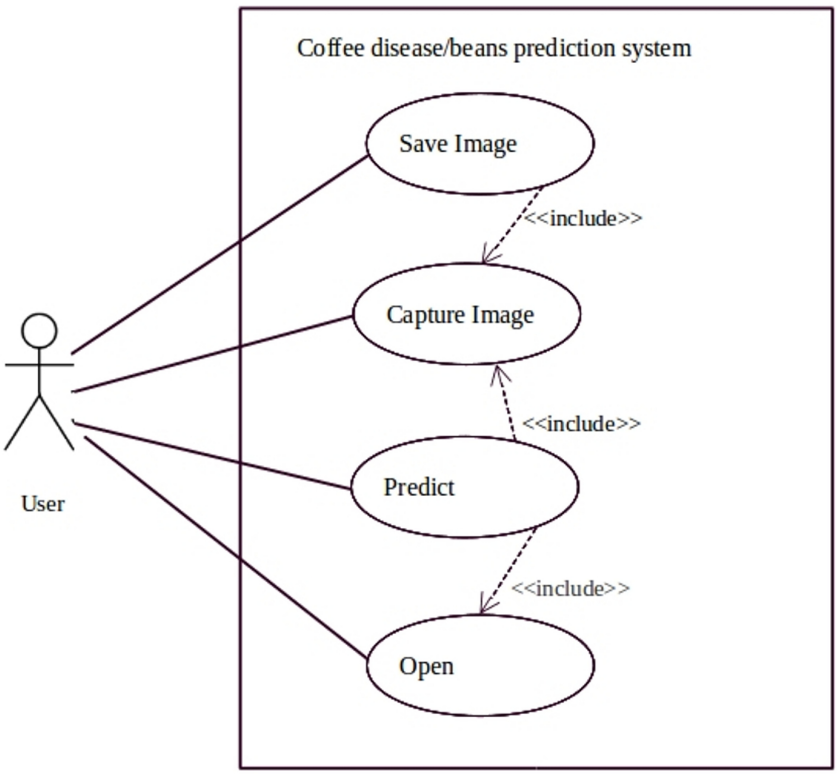


Figure 7.1 – Use case diagram of the App

7.2 Application implementation

Android studio (Studio, 2017) is used to design the App. It is implemented using java programming language. The app supports mobile devices running android 7 and above. The layout is designed to support devices with different screen sizes and resolution. Both the models are implemented in a single application. Tensorflow-android is added as a dependency since the predictions are going to be performed on the device. The app is 59.95MB in size and has logo as shown in figure 7.2 below.

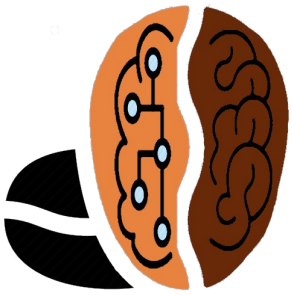


Figure 7.2 – Logo of InfoBuna

7.2.1 User interface

"InfoBuna" has a simple user interface for choosing the task, image handling and displaying the prediction result. The first user interface prompts the user to choose which classification to perform as shown in figure 7.3. Depending on the user's choice, the next page has functionalities to load an image from the device's storage or capture an image, classify the image and save the captured image into device's memory.

Using the "OPEN" button the user can load the input image that can be used as an input for the model. If the user wishes to give an image captured on real time, the "CAMERA" button can be used to open the device's camera. Once an input image is loaded, the "PREDICT" button can be used to load the trained model and pass the image as an input and print the prediction result as shown in *Prediction results* layout for coffee beans classification. The state machine diagram in figure 7.4 shows the steps required to make classification.

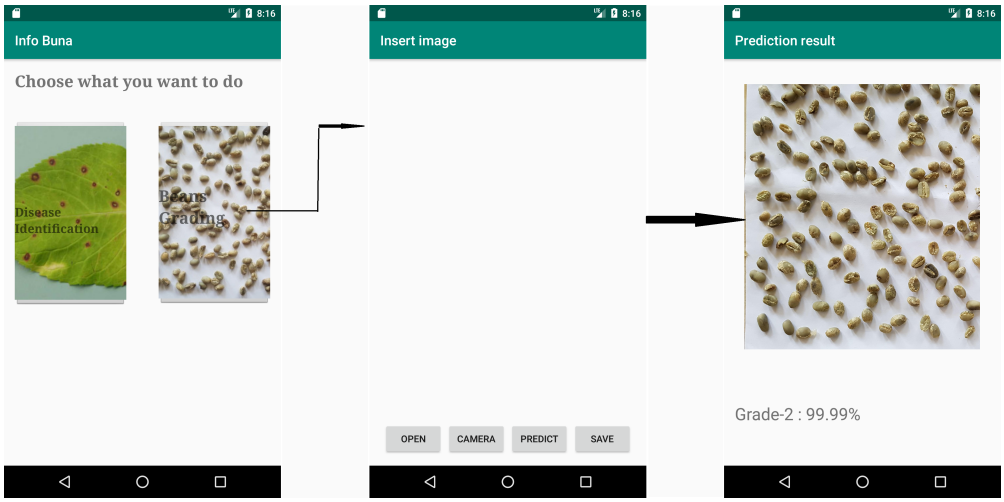


Figure 7.3 – User interface of the App

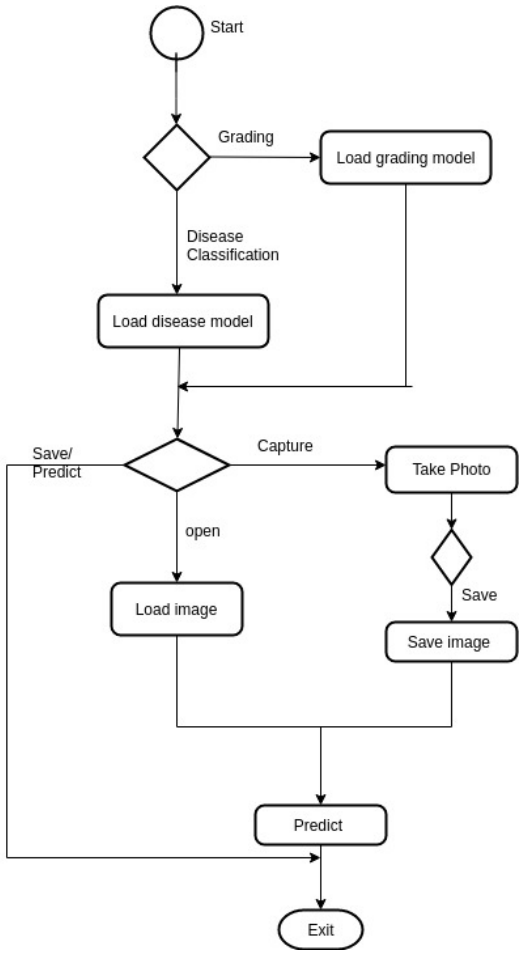


Figure 7.4 – State machine diagram to perform classification

7.2.2 Loading the trained model

```
def keras_to_tensorflow(keras_model, output_dir,
    model_name,out_prefix="output_", log_tensorboard=True):

    if os.path.exists(output_dir) == False:
        os.mkdir(output_dir)
    out_nodes = []

    for i in range(len(keras_model.outputs)):
        out_nodes.append(out_prefix + str(i + 1))
        tf.identity(keras_model.output[i], out_prefix + str(i + 1))

    sess = K.get_session()
    from tensorflow.python.framework import graph_util, graph_io
    init_graph = sess.graph.as_graph_def()
    main_graph = graph_util.convert_variables_to_constants(sess,
        init_graph, out_nodes)
    graph_io.write_graph(main_graph, output_dir, name=model_name,
        as_text=False)

    if log_tensorboard:
        from tensorflow.python.tools import import_pb_to_tensorboard

        import_pb_to_tensorboard.import_to_tensorboard(
            os.path.join(output_dir, model_name),
            output_dir)

keras_model=load_model("model_Norm224_av25.hdf5")
output_dir = os.path.join(os.getcwd(),"checkpoint")
keras_to_tensorflow(keras_model,output_dir=output_dir,
    model_name="coffee_grading.pb")
print("MODEL SAVED")
```

Figure 7.5 – Code used to convert keras model into tensorflow

As we have discussed in Chapters 4 and 6, both of the models are designed using keras. Currently, there is no interface to deploy keras models on mobile devices. However, Google has released Tensorflow Lite (Abadi et al., 2016) to make it easy to perform machine learning on

device. Therefore to deploy the models into android, we have to first convert the trained keras models into the tensorflow version. This is achieved using a simple code shown in 7.5 that takes the trained model and convert it into the tensorflow .pb frozen version. In the example shown the trained model, *"model_Norm224_av25.hdf5"*, for coffee beans grading is converted into the tensorflow equivalent *"coffee_grading.pb"*. The same is performed for the disease classification model.

7.2.3 Preprocessing the input image

During designing of the models, the input images used for training were preprocessed, i.e resized and normalized before they were fed into the network. To get the correct prediction, the same steps should be performed to the images that are going to be predicted. For example, before performing coffee beans classification, the input image is resized to 224x224x3, in BGR order, pixel values should be between [0,1] and normalized using the same mean and std values used during training. These operations can be executed using the code fragment shown in figure 7.6.

```
float[] output = new float[size * size * 3];

int[] intValues = new int[source.getHeight() * source.getWidth()];

source.getPixels(intValues, 0, source.getWidth(), 0, 0,
    source.getWidth(), source.getHeight());
for (int i = 0; i < intValues.length; ++i) {
   final int val = intValues[i];

   output[i*3] = (Color.blue(val)/255f - 0.60487276f)/0.27177873f;
   output[i * 3 + 1] = (Color.green(val)/255f -
       0.6519608f)/0.21680197f;
   output[i * 3 + 2] = (Color.red(val)/255f -
       0.6750148f)/0.19056307f;
```

Figure 7.6 – Code fragment to preprocess images using the same mean and standard deviation values used during training

Finally, after the image is resized and preprocessed, it is passed to the pre-trained model to get prediction. The prediction output is displayed by

printing the class label with highest probability and the confidence of the image to be in that class (refer to Fig. 7.3). Fig. 7.7 shows a coffee farmer and an ECX sorter using the app on real time.

Figure 7.7 – InfoBuna used by a farmer on coffee farm [Top] and by an ECX employee [Bottom]

7.3 Conclusion

We have seen how to deploy a trained model into an android mobile device. Users can perform coffee plant disease classification and coffee beans grading from images captured using their phone and get first hand information about the crop. The app is simple with user friendly interfaces which makes it easier to use. Currently, the app lacks features like multiple language support, user identification and server side component to store statistics and data which can be used to further improve the app. After performing a user experience survey and adding the miss-

ing features, "InfoBuna" will be uploaded in Google play store so that the farmers can download and use it.

CHAPTER 8

8. CONCLUSION

The aim of this work was to design a coffee plant disease identification and grading system using machine learning and image processing techniques. Since coffee is among one of agricultural products that earn the country its foreign currency, assisting the production and processing of coffee with recent technologies is of utmost importance for increasing productivity as well as boosting the economy of the country. Therefore, two models were designed and evaluated using DL; one that classifies coffee plant diseases using images of the plant or parts of the plant, another that grades coffee raw beans into their quality categories using images of the beans captured by mobile devices. Finally, the designed models are deployed on an easy to use android app so that the farmers can use to get accurate information about their plant that helps them to take appropriate measure.

We first made a literature survey on the application of deep learning in agricultural domain. Our survey indicated that only few researches used DL even though it shows promising results in other areas of application. One of the challenges of using DL for real world applications is the need of very large amount of data to successfully train the model. We further investigated the effects of data and data collection techniques on representation learning and the proposed approaches to address these issues. Our investigation indicated that small imbalanced dataset and dataset shift are the main issues that hinders model performance and reliability.

To design both models we used small datasets compared to the amount required to train DL models effectively. The coffee disease classification

model was designed using images downloaded from the internet plus images we captured ourselves on coffee farms. Despite the limited amount of data (562 images) of varying resolution and scale using transfer learning approach, a model is designed that detects the existence of disease and classifies according to the disease type. A model was first designed using a similar dataset containing 12673 leaf images of soybean plant, then it is fine-tuned using the coffee plant disease dataset to transfer the learned features. We have also shown that the number of layers retrained influences the performance of the model.

The second model is used to classify coffee beans into their quality grades using images of raw coffee beans. The dataset was collected at one of the grading centers in Ethiopia, Jimma grading center, in two rounds. While the manual grading uses both the physical properties as well as the cup taste of the beans, the dataset was prepared only based on the physical properties of the beans because of the strong correlations between the two values. In total 2109 images of coffee beans were collected in two sets using two mobile devices. In addition to the small number of data, the change in collection methodology in the two sets resulted in a data distribution mismatch between them. Our focus during designing the network architecture was to make the number of parameters as small as possible to effectively train the model and reduce overfitting. The new CNN architecture designed and trained from scratch performed better than both classical machine learning and off-the-shelf DL models. In an attempt to solve the dataset mismatch, we introduced color correction preprocessing and augmentation techniques to the existing data pipeline without succeeding to improve the performance of the model any further.

Though the model performed better than the other approaches, it suffered from high variance. We have also shown that using ensemble learning can reduce the variance as well as increase the performance of the model. Finally, by careful designing of the architecture, using data augmentation, preprocessing and ensemble techniques, a classifier is designed that grades raw coffee beans with a Top-1 accuracy of 89.1% and Top-2 accuracy of 98.22%.

An accurate model is nothing unless it reaches to its intended users. One of the ways a model can be accessed by many users is to implement it in a mobile based application. With this intension, an android

mobile app is designed and both the models were deployed. The app has simple user interface with functionalities to capture images, open images saved on the device, classify the type of disease or the grade of the beans depending on the model used and save the prediction output if required. The models, designed in Keras, were converted to tensorflow version and the prediction is performed on the device. While this has an advantage minimizing the communication overhead, it is not possible to collect statistics that would be useful to improve the models further.

8.1 Future work

We have seen that the use of transfer learning approach improved the classification performance of the coffee plant disease identification model with very little training images. In addition, the model detects only a single disease in an image and fails when multiple diseases exist in a single leaf. The performance of the model can be further improved by enriching the dataset including more representative samples as well as additional labeling to detect multiple diseases in a single image. Moreover, using a two stage classification i.e putting an object detection before the classifier to identify the diseased part of the plant, may improve the detection of small symptoms which would have disappeared when the original image is resized to the input size of the model. This can be achieved using recurrent convolutional neural networks (RCNN) (Liang and X. Hu, 2015) which beats state-of-the-art CNN based models on most object recognition benchmark datasets.

The coffee beans dataset was collected at only Jimma grading center which grades coffee beans produced in that region but currently there are more than nine grading centers in the country located at main coffee growing areas. The dataset does not include images of beans from these regions which have different physical characteristics depending on the area where they were grown (Turi et al., 2013). The model is trained and tasted only using the beans from one area. In chapter 2, we have discussed how a slight difference between the training and test data affects a classifier. Therefore, for the coffee beans model to grade the beans from the other regions, it should be optimized and tested before deploying in other grading centers. We have attempted to generate images using GAN model and were able to get promising results. The application of GAN for grading should also investigated in the future.

Instead of collecting large amount of data at each center and retraining the model, it would be more feasible to use either supervised or semi-supervised domain adaption. Research has shown the ability of domain adaptation to learn a mapping between domains in the situation when the target domain data are either fully unlabeled (unsupervised domain adaptation) or have few labeled samples (semi-supervised domain adaptation) (Ganin et al., 2016). Domain adaptation can also be used to solve the data mismatch problem that exists between Dataset1 and Dataset2 by learning features that are discriminative for the main learning task (coffee beans grading) and indiscriminate with respect to the shift between the two sets.

Another limitation in the dataset is that there are missing grade categories (Grade-4 and Grade-5) because these samples were not available in the center during data collection. In the future samples beans from these grades should be collected and the model should be retrained using the new dataset which includes these missing classes. A different and interesting approach would be to use zero/few-shot learning (Socher et al., 2013; Gidaris and Komodakis, 2018; Xian et al., 2018) to classify the missing classes during test time. These approaches show promising results for coarse classes with very little/no training sample for some classes. However, it would be difficult to model for fine-grained classes like our task.

Though the mobile app supports important functionalities such as capturing the input image, performing predictions and displaying the output, it lacks features like screening inputs and language support. Currently, the app is available only in English. Since the target users are coffee farmers who may not have sufficient education to read and write in English, it would be much easier to use the app if it is available in the language the user understands. In the future, it should include at least the languages spoken in main coffee growing areas. At this point, every image is accepted as an input whether it is an image of a leaf, beans or a person. It is up to the user to enter the right image to the model. In the future, a first level filter should be added to detect if the right kind of image is submitted. For instance, for the coffee beans grading model if the user attempts to enter images other than coffee beans, the app should prompt the user to enter only images of coffee beans. This could be done by training a binary classifier to distinguish between beans and NOT beans.

8.2 Publications

During the thesis, the following papers were published in international conferences with peer-reviewing:

1. **Wallelign S**, Polceanu M, Jemal T, Buche C, "Coffee Grading with Convolutional Neural Networks using Small Datasets with High Variance", the 27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG-2019), pp (113-120), 2019. This paper describes the coffee beans grading model design. Part of the content of this paper is contained in chapter 6 of the thesis. The conference is ranked B according to CORE2018 and 29.9% acceptance ratio.
2. **Wallelign S**, Polceanu M, Buche C, "Soybean Plant Disease Identification Using Convolutional Neural Network", the 31st International Florida Artificial Intelligence Research Society Conference (FLAIRS-31), pp (146-151), 2018. This paper describes the first part of disease identification using CNN. Part of the content of this paper is included in chapter 4 of this thesis. The conference has rank C (CORE2018).

BIBLIOGRAPHY

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). « Tensorflow: A system for large-scale machine learning ». In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283 (cit. on p. 95).

Abera, A., Lemessa, F., and Muleta, D. (2011). « The antifungal activity of some medicinal plants against coffee berry disease caused by Colletotrichum kahawae ». In: *Int. J. Agric. Res* 6.3, pp. 268–279 (cit. on pp. 2, 43).

Al Bashish, D., Braik, M., and Bani-Ahmad, S. (2011). « Detection and classification of leaf diseases using K-means-based segmentation and ». In: *Information Technology Journal* 10.2, pp. 267–275 (cit. on p. 41).

Alejo, R., Garcia, V., Sotoca, J. M., Mollineda, R. A., and Sanchez, J. S. (2007). « Improving the performance of the RBF neural networks trained with imbalanced samples ». In: *International Work-Conference on Artificial Neural Networks*. Springer, pp. 162–169 (cit. on p. 26).

Alemu, K., Adugna, G., Lemessa, F., and Muleta, D. (2016). « Current status of coffee berry disease (Colletotrichum kahawae Waller & Bridge) in Ethiopia ». In: *Archives of Phytopathology and Plant Protection* 49.17-18, pp. 421–433 (cit. on pp. 2, 43).

Amazon (2019). *Registry of Open Data on AWS*. https://registry.opendata.aws//. (Visited on 11/27/2019) (cit. on p. 2).

Aran, M., Nath, A. G., and Shyna, A. (2016). « Automated Cashew kernel grading using machine vision ». In: *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*. IEEE, pp. 1–5 (cit. on p. 47).

Arivazhagan, S., Shebiah, R. N., Ananthi, S., and Varthini, S. V. (2013). « Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features ». In: *Agricultural Engineering International: CIGR Journal* 15.1, pp. 211–217 (cit. on p. 44).

Awate, A., Deshmankar, D., Amrutkar, G., Bagul, U., and Sonavane, S. (2015). « Fruit disease detection using color, texture analysis and ANN ». In: *2015 International Conference on Green Computing and Internet of Things (ICGCIOT)*. IEEE, pp. 970–975 (cit. on p. 45).

Ayitenfsu, B. M. (2014). « Method of coffee bean defect detection ». In: *International Journal of Engineering Research & Technology* 3, pp. 2355–57 (cit. on pp. 41, 48).

Bah, M. D., Hafiane, A., and Canals, R. (2018). « Deep learning with unsupervised data labeling for weeds detection on uav images ». In: *arXiv preprint arXiv:1805.12395* (cit. on p. 39).

Barbedo, J. G. A. (2013). « Digital image processing techniques for detecting, quantifying and classifying plant diseases ». In: *SpringerPlus* 2.1, p. 660 (cit. on pp. 40, 41).

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). « Speeded-up robust features (SURF) ». In: *Computer vision and image understanding* 110.3, pp. 346–359 (cit. on p. 23).

El-Bendary, N., El Hariri, E., Hassanien, A. E., and Badr, A. (2015). « Using machine learning techniques for evaluating tomato ripeness ». In: *Expert Systems with Applications* 42.4, pp. 1892–1905 (cit. on p. 39).

Benediktsson, J. A., Pesaresi, M., and Amason, K. (2003). « Classification and feature extraction for remote sensing images from urban areas based on morphological transformations ». In: *IEEE Transactions on Geoscience and Remote Sensing* 41.9, pp. 1940–1949 (cit. on p. 23).

Bengio, Y., Courville, A., and Vincent, P. (2013). « Representation learning: A review and new perspectives ». In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828 (cit. on p. 9).

Bock, C., Poole, G., Parker, P., and Gottwald, T. (2010). « Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging ». In: *Critical Reviews in Plant Sciences* 29.2, pp. 59–107 (cit. on p. 41).

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, J., et al. (2016). « End to end learning for self-driving cars ». In: *arXiv preprint arXiv:1604.07316* (cit. on p. 1).

Breiman, L. (1996). « Bagging predictors ». In: *Machine learning* 24.2, pp. 123–140 (cit. on pp. 35, 36).

Buchsbaum, G. (1980). « A spatial processor model for object colour perception ». In: *Journal of the Franklin institute* 310.1, pp. 1–26 (cit. on p. 77).

Camargo, A. and Smith, J. (2009). « Image pattern classification for the identification of disease causing agents in plants ». In: *Computers and Electronics in Agriculture* 66.2, pp. 121–125 (cit. on p. 44).

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). « SMOTE: synthetic minority over-sampling technique ». In: *Journal of artificial intelligence research* 16, pp. 321–357 (cit. on pp. 26, 29).

Chen, H., Lundberg, S., and Lee, S.-I. (2017). « Checkpoint Ensembles: Ensemble Methods from a Single Training Process ». In: *arXiv preprint arXiv:1710.03282* (cit. on pp. 3, 83).

Chollet, F. et al. (2015). *Keras* (cit. on p. 58).

Chung, C.-L., Huang, K.-J., Chen, S.-Y., Lai, M.-H., Chen, Y.-C., and Kuo, Y.-F. (2016). « Detecting Bakanae disease in rice seedlings by machine vision ». In: *Computers and Electronics in Agriculture* 121, pp. 404–411 (cit. on p. 44).

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). « Deep, big, simple neural nets for handwritten digit recognition ». In: *Neural computation* 22.12, pp. 3207–3220 (cit. on p. 29).

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). « Fast and accurate deep network learning by exponential linear units (elus) ». In: *arXiv preprint arXiv:1511.07289* (cit. on p. 21).

Daviron, B. and Ponte, S. (2005). « The Coffee Paradox: Global markets ». In: *Commodity Trade and the Elusive promise of Development Zed books: London* (cit. on p. 42).

Demelash, T., Kifle, B., et al. (2018). « A review of coffee diseases research in Ethiopia. » In: *International Journal of Agriculture and Biosciences* 7.2, pp. 65–70 (cit. on pp. 2, 43).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). « ImageNet: A large-scale hierarchical image database ». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255 (cit. on pp. 20, 34, 73).

Deng, L. (2012). « The MNIST database of handwritten digit images for machine learning research [best of the web] ». In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142 (cit. on pp. 27, 73).

Dietterich, T. G. (1998). « Approximate statistical tests for comparing supervised classification learning algorithms ». In: *Neural computation* 10.7, pp. 1895–1923 (cit. on p. 86).

Dietterich, T. G. (2000). « Ensemble methods in machine learning ». In: *International workshop on multiple classifier systems*. Springer, pp. 1–15 (cit. on pp. 3, 83).

Dietterich, T. G. et al. (2002). « Ensemble learning ». In: *The handbook of brain theory and neural networks* 2, pp. 110–125 (cit. on p. 16).

Durmuş, H., Güneş, E. O., and Kırcı, M. (2017). « Disease detection on the leaves of the tomato plants by using deep learning ». In: *2017 6th International Conference on Agro-Geoinformatics*. IEEE, pp. 1–5 (cit. on pp. 46, 47).

Elkan, C. (2001). « The foundations of cost-sensitive learning ». In: *International joint conference on artificial intelligence*. Vol. 17. 1. Lawrence Erlbaum Associates Ltd, pp. 973–978 (cit. on p. 26).

Everitt, B. S. (1992). *The analysis of contingency tables*. Chapman and Hall/CRC (cit. on p. 87).

Faridah, F., Parikesit, G. O., and Ferdiansjah, F. (2011). « Coffee bean grade determination based on image parameter ». In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 9.3, pp. 547–554 (cit. on pp. 41, 48).

Faridah, F., Parikesit, G. O., and Ferdiansjah, F. (2013). « Coffee bean grade determination based on image parameter ». In: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 9.3, pp. 547–554 (cit. on pp. 47, 48).

Ferentinos, K. P. (2018). « Deep learning models for plant disease detection and diagnosis ». In: *Computers and Electronics in Agriculture* 145, pp. 311–318 (cit. on pp. 45–47).

Freund, Y., Schapire, R. E., et al. (1996). « Experiments with a new boosting algorithm ». In: *icml*. Vol. 96. Citeseer, pp. 148–156 (cit. on p. 36).

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). « Synthetic data augmentation using GAN for improved liver lesion classification ». In: *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE, pp. 289–293 (cit. on p. 29).

Fujita, E., Kawasaki, Y., Uga, H., Kagiwada, S., and Iyatomi, H. (2016). « Basic investigation on a robust and practical plant diagnostic system ». In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, pp. 989–992 (cit. on pp. 45, 46).

Fukushima, K. (1980). « Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position ». In: *Biological cybernetics* 36.4, pp. 193–202 (cit. on p. 19).

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). « Domain-adversarial training of neural networks ». In: *The Journal of Machine Learning Research* 17.1, pp. 2096–2030 (cit. on p. 102).

Gidaris, S. and Komodakis, N. (2018). « Dynamic few-shot visual learning without forgetting ». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375 (cit. on p. 102).

Gijsenij, A., Gevers, T., and Van De Weijer, J. (2011). « Computational color constancy: Survey and experiments ». In: *IEEE Transactions on Image Processing* 20.9, pp. 2475–2489 (cit. on p. 77).

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press (cit. on pp. 2, 19).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). « Generative adversarial nets ». In: *Advances in neural information processing systems*, pp. 2672–2680 (cit. on pp. 3, 29, 31).

Greenspan, H., Van Ginneken, B., and Summers, R. M. (2016). « Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique ». In: *IEEE Transactions on Medical Imaging* 35.5, pp. 1153–1159 (cit. on p. 1).

Haralick, R. M., Shanmugam, K., et al. (1973). « Textural features for image classification ». In: *IEEE Transactions on systems, man, and cybernetics* 6, pp. 610–621 (cit. on pp. 23, 78).

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). « ADASYN: Adaptive synthetic sampling approach for imbalanced learning ». In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, pp. 1322–1328 (cit. on p. 26).

He, H. and Garcia, E. A. (2008). « Learning from imbalanced data ». In: *IEEE Transactions on Knowledge & Data Engineering* 9, pp. 1263–1284 (cit. on p. 25).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). « Deep residual learning for image recognition ». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (cit. on pp. 17, 33, 35).

Al-Hiary, H., Bani-Ahmad, S., Reyalat, M., Braik, M., and ALRahamneh, Z. (2011). « Fast and accurate detection and classification of plant diseases ». In: *International Journal of Computer Applications* 17.1, pp. 31–38 (cit. on p. 44).

Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*. Vol. 398. John Wiley & Sons (cit. on p. 10).

Hu, M.-K. (1962). « Visual pattern recognition by moment invariants ». In: *IRE transactions on information theory* 8.2, pp. 179–187 (cit. on pp. 23, 78).

Hubel, D. H. and Wiesel, T. N. (1962). « Receptive fields, binocular interaction and functional architecture in the cat's visual cortex ». In: *The Journal of physiology* 160.1, pp. 106–154 (cit. on p. 19).

Hung, C., Xu, Z., and Sukkarieh, S. (2014). « Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV ». In: *Remote Sensing* 6.12, pp. 12037–12054 (cit. on p. 39).

ICO (2019). *Annual Review*. http://www.ico.org/documents/cy2018-19/annual-review-2017-18-e.pdf. (Visited on 12/05/2019) (cit. on pp. 41, 42).

Ioffe, S. and Szegedy, C. (2015). « Batch normalization: Accelerating deep network training by reducing internal covariate shift ». In: *arXiv preprint arXiv:1502.03167* (cit. on p. 37).

Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). « Artificial neural networks: A tutorial ». In: *Computer* 29.3, pp. 31–44 (cit. on p. 16).

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*. Vol. 112. Springer (cit. on p. 24).

Japkowicz, N. (2000). « The class imbalance problem: Significance and strategies ». In: *Proceedings of the International Conference on Artificial Intelligence* (cit. on p. 26).

Japkowicz, N. and Stephen, S. (2002). « The class imbalance problem: A systematic study ». In: *Intelligent data analysis* 6.5, pp. 429–449 (cit. on p. 26).

Kaggle Inc. (2019). *Kaggle Datasets*. https://www.kaggle.com/datasets/. (Visited on 11/27/2019) (cit. on p. 2).

Kamilaris, A. and Prenafeta-Boldú, F. X. (2018). « Deep learning in agriculture: A survey ». In: *Computers and electronics in agriculture* 147, pp. 70–90 (cit. on p. 47).

Kingma, D. P. and Ba, J. (2014). « Adam: A method for stochastic optimization ». In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 19, 22).

Koehrsen, W. (2017). *Random Forest Simple Explanation*. URL: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d (visited on 11/29/2019) (cit. on p. 17).

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer (cit. on p. 22).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). « Imagenet classification with deep convolutional neural networks ». In: *Advances in neural information processing systems*, pp. 1097–1105 (cit. on pp. 1, 17, 21, 23, 34, 58).

Krogh, A. and Hertz, J. A. (1992). « A simple weight decay can improve generalization ». In: *Advances in neural information processing systems*, pp. 950–957 (cit. on p. 36).

Kumar, R. A., Rajpurohit, V. S., and Nargund, V. (2017). « A neural network assisted machine vision system for sorting pomegranate fruits ». In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, pp. 1–9 (cit. on p. 47).

Kussul, N., Lavreniuk, M., Skakun, S., and Shelestov, A. (2017). « Deep learning classification of land cover and crop types using remote sensing data ». In: *IEEE Geoscience and Remote Sensing Letters* 14.5, pp. 778–782 (cit. on p. 39).

LeCun, Y., Bengio, Y., et al. (1995). « Convolutional networks for images, speech, and time series ». In: *The handbook of brain theory and neural networks* 3361.10, p. 1995 (cit. on p. 19).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). « Deep learning ». In: *nature* 521.7553, p. 436 (cit. on p. 22).

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). « Gradient-based learning applied to document recognition ». In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 20, 23, 33).

Lee, S. H., Chan, C. S., Wilkin, P., and Remagnino, P. (2015). « Deep-plant: Plant identification with convolutional neural networks ». In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 452–456 (cit. on p. 52).

Liang, M. and Hu, X. (2015). « Recurrent convolutional neural network for object recognition ». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3367–3375 (cit. on p. 101).

Liaw, A., Wiener, M., et al. (2002). « Classification and regression by randomForest ». In: *R news* 2.3, pp. 18–22 (cit. on p. 16).

LifeCLEF (2019). *Multimidea retrival in CLEF*. https://www.imageclef.org//. (Visited on 12/10/2019) (cit. on p. 52).

Liu, B., Zhang, Y., He, D., and Li, Y. (2018). « Identification of apple leaf diseases based on deep convolutional neural networks ». In: *Symmetry* 10.1, p. 11 (cit. on p. 46).

Lowe, D. G. (2004). « Distinctive image features from scale-invariant keypoints ». In: *International journal of computer vision* 60.2, pp. 91–110 (cit. on p. 23).

Lu, J., Hu, J., Zhao, G., Mei, F., and Zhang, C. (2017). « An in-field automatic wheat disease diagnosis system ». In: *Computers and electronics in agriculture* 142, pp. 369–379 (cit. on p. 46).

Ma, J., Du, K., Zheng, F., Zhang, L., Gong, Z., and Sun, Z. (2018). « A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network ». In: *Computers and Electronics in Agriculture* 154, pp. 18–24 (cit. on pp. 45, 46).

Maaten, L. v. d. and Hinton, G. (2008). « Visualizing data using t-SNE ». In: *Journal of machine learning research* 9.Nov, pp. 2579–2605 (cit. on p. 69).

Mao, R., Zhu, H., Zhang, L., and Chen, A. (2006). « A new method to assist small data set neural network learning ». In: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 1. IEEE, pp. 17–22 (cit. on p. 25).

Milioto, A., Lottes, P., and Stachniss, C. (2018). « Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging

background knowledge in cnns ». In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2229–2235 (cit. on p. 39).

Mitiku, F., Mey, Y. de, Nyssen, J., and Maertens, M. (2017). « Do private sustainability standards contribute to income growth and poverty alleviation? » In: *Sustainability* 9.2, p. 246 (cit. on pp. 1, 42).

Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). « Using deep learning for image-based plant disease detection ». In: *Frontiers in plant science* 7, p. 1419 (cit. on pp. 45, 47).

Moreno-Torres, J. G., Raeder, T., Alaiz-RodriGuez, R., Chawla, N. V., and Herrera, F. (2012). « A unifying view on dataset shift in classification ». In: *Pattern Recognition* 45.1, pp. 521–530 (cit. on p. 71).

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). « Reading digits in natural images with unsupervised feature learning ». In: (cit. on p. 27).

Neumann, M., Hallau, L., Klatt, B., Kersting, K., and Bauckhage, C. (2014). « Erosion band features for cell phone image based plant disease classification ». In: *2014 22nd International Conference on Pattern Recognition*. IEEE, pp. 3315–3320 (cit. on p. 44).

Ng, A. Y. (2004). « Feature selection, L 1 vs. L 2 regularization, and rotational invariance ». In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, p. 78 (cit. on p. 36).

Nguyen, C. H. and Ho, T. B. (2005). « An imbalanced data rule learner ». In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 617–624 (cit. on pp. 26, 27).

Nguyen, G. H., Bouzerdoum, A., and Phung, S. L. (2009). « Learning pattern classification tasks with imbalanced data sets ». In: *Pattern recognition*. IntechOpen (cit. on pp. 26, 27).

Odena, A. (2016). « Semi-supervised learning with generative adversarial networks ». In: *arXiv preprint arXiv:1606.01583* (cit. on pp. 29, 76).

Olaniyi, E. O., Adekunle, A. A., Odekuoye, T., and Khashman, A. (2017). « Automatic system for grading banana using GLCM texture feature extraction and neural network arbitrations ». In: *Journal of food process engineering* 40.6, e12575 (cit. on p. 47).

Oliveira, E. M. de, Leme, D. S., Barbosa, B. H. G., Rodarte, M. P., and Pereira, R. G. F. A. (2016). « A computer vision system for coffee beans classification based on computational intelligence techniques ». In: *Journal of Food engineering* 171, pp. 22–27 (cit. on pp. 47, 48).

Osorio, N. (2004). « Lessons from the world coffee crisis: A serious problem for sustainable development ». In: *London: International Coffee Organization* (cit. on p. 41).

Osuna, E., Freund, R., and Girosit, F. (1997). « Training support vector machines: an application to face detection ». In: *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, pp. 130–136 (cit. on p. 14).

Padmanabhan, S. (1973). « The great Bengal famine ». In: *Annual Review of Phytopathology* 11.1, pp. 11–24 (cit. on p. 40).

Padol, P. B. and Yadav, A. A. (2016). « SVM classifier based grape leaf disease detection ». In: *2016 Conference on advances in signal processing (CASP)*. IEEE, pp. 175–179 (cit. on p. 44).

Pan, S. J. and Yang, Q. (2009). « A survey on transfer learning ». In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359 (cit. on p. 33).

Pandey, R., Gamit, N., and Naik, S. (2014). « A novel non-destructive grading method for Mango (Mangifera Indica L.) using fuzzy expert system ». In: *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, pp. 1087–1094 (cit. on p. 47).

Picon, A., Alvarez-Gila, A., Seitz, M., Ortiz-Barredo, A., Echazarra, J., and Johannes, A. (2018). « Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild ». In: *Computers and Electronics in Agriculture* (cit. on pp. 45, 46).

Pinto, C., Furukawa, J., Fukai, H., and Tamura, S. (2017). « Classification of Green coffee bean images basec on defect types using convolutional neural network (CNN) ». In: *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*. IEEE, pp. 1–5 (cit. on p. 48).

Qin, F., Liu, D., Sun, B., Ruan, L., Ma, Z., and Wang, H. (2016). « Identification of alfalfa leaf diseases using image recognition technology ». In: *PLoS One* 11.12, e0168274 (cit. on p. 45).

Quinlan, J. R. (1986). « Induction of decision trees ». In: *Machine learning* 1.1, pp. 81–106 (cit. on p. 13).

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press (cit. on p. 27).

Ramakrishnan, M. et al. (2015). « Groundnut leaf disease detection and classification by using back probagation algorithm ». In: *2015 International Conference on Communications and Signal Processing (ICCSP)*. IEEE, pp. 0964–0968 (cit. on p. 45).

Reyes, A. K., Caicedo, J. C., and Camargo, J. E. (2015). « Fine-tuning Deep Convolutional Networks for Plant Recognition. » In: *CLEF (Working Notes)* (cit. on p. 45).

Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge university press (cit. on p. 11).

Ruder, S. (2016). « An overview of gradient descent optimization algorithms ». In: *arXiv preprint arXiv:1609.04747* (cit. on p. 19).

Sabrol, H. and Satish, K. (2016). « Tomato plant disease classification in digital images using classification tree ». In: *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, pp. 1242–1246 (cit. on p. 45).

Sankaran, S., Mishra, A., Ehsani, R., and Davis, C. (2010). « A review of advanced techniques for detecting plant diseases ». In: *Computers and Electronics in Agriculture* 72.1, pp. 1–13 (cit. on p. 40).

Sannakki, S. S., Rajpurohit, V. S., Nargund, V., and Kulkarni, P. (2013). « Diagnosis and classification of grape leaf diseases using neural networks ». In: *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. IEEE, pp. 1–5 (cit. on pp. 44, 45).

Savary, S., Ficke, A., Aubertot, J.-N., and Hollier, C. (Dec. 2012). « Crop losses due to diseases and their implications for global food production losses and food security ». In: *Food Security* 4. DOI: 10.1007/s12571-012-0200-5 (cit. on p. 40).

Semary, N. A., Tharwat, A., Elhariri, E., and Hassanien, A. E. (2015). « Fruit-based tomato grading system using features fusion and support vector machine ». In: *Intelligent Systems' 2014*. Springer, pp. 401–410 (cit. on pp. 39, 47).

Shorten, C. and Khoshgoftaar, T. M. (2019). « A survey on image data augmentation for deep learning ». In: *Journal of Big Data* 6.1, p. 60 (cit. on p. 30).

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). « Best practices for convolutional neural networks applied to visual document analysis. » In: *Icdar*. Vol. 3. 2003 (cit. on p. 29).

Simonyan, K. and Zisserman, A. (2014). « Very deep convolutional networks for large-scale image recognition ». In: *arXiv preprint arXiv:1409.1556* (cit. on pp. 33, 35, 77).

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., and Stefanovic, D. (2016). « Deep neural networks based recognition of plant diseases by leaf image classification ». In: *Computational intelligence and neuroscience* 2016 (cit. on p. 46).

Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. (2013). « Zero-shot learning through cross-modal transfer ». In: *Advances in neural information processing systems*, pp. 935–943 (cit. on p. 102).

Son, N.-T., Chen, C.-F., Chen, C.-R., and Minh, V.-Q. (2018). « Assessment of Sentinel-1A data for rice crop classification using random forests and support vector machines ». In: *Geocarto international* 33.6, pp. 587–601 (cit. on p. 39).

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). « Striving for simplicity: The all convolutional net ». In: *arXiv preprint 1412.6806* (cit. on pp. 63, 83).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). « Dropout: a simple way to prevent neural networks from overfitting ». In: *The journal of machine learning research* 15.1, pp. 1929–1958 (cit. on p. 36).

Stellmacher, T. (2007). *Governing the Ethiopian coffee forests: a local level institutional analysis in Kaffa and Bale Mountains*. Shaker (cit. on p. 1).

Studio, A. (2017). « Android Studio ». In: *The Official IDE for Android* (cit. on p. 93).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al. (2015). « Going deeper with convolutions ». In: Cvpr (cit. on pp. 17, 20, 33, 34, 77).

Tao, Q., Wu, G.-W., Wang, F.-Y., and Wang, J. (2005). « Posterior probability support vector machines for unbalanced data ». In: *IEEE Transactions on Neural Networks* 16.6, pp. 1561–1573 (cit. on pp. 26, 27).

Tian, J., Hu, Q., Ma, X., and Han, M. (2012). « An improved KPCA/GA-SVM classification model for plant leaf disease recognition ». In: *Journal of Computational Information Systems* 8.18, pp. 7737–7745 (cit. on p. 44).

Torrey, L. and Shavlik, J. (2010). « Transfer learning ». In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, pp. 242–264 (cit. on p. 3).

Trade, C. on, Nations Unies sur le commerce et le développement, C. des, Conference, U. N., Staff, D., et al. (2003). *Economic Development in Africa: Trade Performance and Commodity Dependence*. United Nations Publications (cit. on p. 41).

Tuia, D., Flamary, R., and Courty, N. (2015). « Multiclass feature learning for hyperspectral image classification: Sparse and hierarchical solutions ». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 105, pp. 272–285 (cit. on p. 39).

Turi, B., Abebe, G., and Goro, G. (2013). « Classification of Ethiopian coffee beans using imaging techniques ». In: *East African Journal of Sciences* 7.1, pp. 1–10 (cit. on pp. 48, 101).

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). « Adversarial Discriminative Domain Adaptation ». In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 28).

Ullstrup, A. (1972). « The impacts of the southern corn leaf blight epidemics of 1970-1971 ». In: *Annual review of phytopathology* 10.1, pp. 37–50 (cit. on p. 40).

University, P. S. (2019). *PlantVillage*. https://plantvillage.psu.edu//. (Visited on 12/10/2019) (cit. on p. 52).

Waghmare, H., Kokare, R., and Dandawate, Y. (2016). « Detection and classification of diseases of Grape plant using opposite colour Local Binary Pattern feature and machine learning for automated Decision Support System ». In: *2016 3rd international conference on signal processing and integrated networks (SPIN)*. IEEE, pp. 513–518 (cit. on p. 44).

Wallelign, S., Polceanu, M., and Buche, C. (2018). « Soybean Plant Disease Identification Using Convolutional Neural Network ». In: *The Thirty-First International Flairs Conference*, pp. 146–151 (cit. on p. 55).

Wallelign, S., Polceanu, M., Jemal, T., and Buche, C. (2019). « Coffee Grading with Convolutional Neural Networks using Small Datasets with High Variance ». In: *Journal of WSCG* 27, pp. 113–120. DOI: 10.24132/JWSCG.2019.27.2.4 (cit. on p. 75).

Wang, H., Li, G., Ma, Z., and Li, X. (2012). « Image recognition of plant diseases based on principal component analysis and neural networks ». In: *2012 8th International Conference on Natural Computation*. IEEE, pp. 246–251 (cit. on p. 45).

Wu, S. G., Bao, F. S., Xu, E. Y., Wang, Y.-X., Chang, Y.-F., and Xiang, Q.-L. (2007). « A leaf recognition algorithm for plant classification using probabilistic neural network ». In: *2007 IEEE international symposium on signal processing and information technology*. IEEE, pp. 11–16 (cit. on p. 52).

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. (2018). « Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly ». In: *IEEE transactions on pattern analysis and machine intelligence* (cit. on p. 102).

Xu, B., Wang, N., Chen, T., and Li, M. (2015). « Empirical evaluation of rectified activations in convolutional network ». In: *arXiv preprint arXiv:1505.00853* (cit. on p. 21).

Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., and Yu, Y. (2015). « HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition ». In: *Proceedings of*

the IEEE international conference on computer vision, pp. 2740–2748 (cit. on p. 79).

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). « How transferable are features in deep neural networks? » In: *Advances in neural information processing systems*, pp. 3320–3328 (cit. on p. 33).

Zappone, A., Di Renzo, M., and Debbah, M. (2019). « Wireless networks design in the era of deep learning: Model-based, AI-based, or both? » In: *arXiv preprint arXiv:1902.02647* (cit. on p. 25).

Zeru, A., Assefa, F., Adugna, G., and Hindorf, H. (2012). « Occurrence of fungal diseases of Coffea arabica L. in montane rainforests of Ethiopia ». In: *Journal of Applied Botany and Food Quality* 82.2, pp. 148–151 (cit. on p. 43).

Zhao, B., Feng, J., Wu, X., and Yan, S. (2017). « A survey on deep learning-based fine-grained object classification and semantic segmentation ». In: *International Journal of Automation and Computing* 14.2, pp. 119–135 (cit. on p. 72).

Zheng, Z., Zheng, L., and Yang, Y. (2017). « Unlabeled samples generated by gan improve the person re-identification baseline in vitro ». In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3754–3762 (cit. on p. 29).

Zhou, Z.-H. and Liu, X.-Y. (2006). « Training cost-sensitive neural networks with methods addressing the class imbalance problem ». In: *IEEE Transactions on Knowledge and Data Engineering* 1, pp. 63–77 (cit. on p. 26).

Titre : Un système intelligent pour le classement du café et l'identification des maladies

Mots clés : Apprentissage approfondi, Réseau neuronal convolutif, Apprentissage d'ensemble, Ensemble de données, Café.

Résumé :

L'un des facteurs clés du succès des modèles d'apprentissage profond est leur capacité d'apprendre automatiquement des représentations importantes à partir des données d'entrée, sans qu'il soit nécessaire que des experts humains conçoivent des caractéristiques spécifiques aux tâches. Cependant, pour apprendre ces représentations, les méthodes d'apprentissage en profondeur nécessitent généralement de grandes quantités de données, qui sont coûteuses à obtenir, surtout en raison des efforts requis pour recueillir et étiqueter les données. Cette thèse examine l'applicabilité de l'apprentissage en profondeur à des situations du monde réel où les données existent en petites quantités, recueillies à différents endroits (laboratoires), en utilisant différentes techniques d'acquisition et dans des conditions minimales contrôlées. Il y a deux contributions principales. Tout d'abord, nous nous attaquons au problème de la détection des maladies du caféier, qui jusqu'à présent n'était pas abordé dans la littérature, en utilisant un ensemble de données contenant des images de maladies du caféier téléchargées sur Internet et que nous avons capturées à la ferme, et une approche transfert - apprentissage. Il a été possible de concevoir un modèle qui classifie les maladies du caféier avec une précision de 90,18 % en utilisant seulement 562 images du caféier**.** Deuxièmement, nous nous attaquons au problème du classement des grains de café. Un ensemble de données pour le classement des grains de café est créé en capturant des images de grains de café à la branche Jimma du Commodity Exchange (ECX) de l'Éthiopie en deux séries. Afin de tenter de résoudre le décalage de l'ensemble de données qui s'est produit dans les deux ensembles en raison des différences d'éclairage et de caméra, des algorithmes de correction des couleurs ont été ajoutés au pipeline existant de traitement des images et d'augmentation des données. Toutefois, ces techniques n'ont pas amélioré les performances du modèle par rapport aux méthodes de prétraitement couramment utilisées. Cela indique que la différence dans les techniques d'acquisition d'images n'était pas la seule raison du décalage de l'ensemble de données. Nous avons proposé et évalué une architecture de réseau qui, combinée à des techniques d'augmentation des données et d'apprentissage d'ensemble, a mené à un classificateur amélioré (précision de 89,1 % sur l'ensemble des données d'essai) qui évalue les grains de café et qui est plus performant que les méthodes classiques d'apprentissage machine (amélioration de 25,47 %) et que les modèles prêts-à-servir (18 %). L'ensemble de données sur les grains de café et les modèles seront mis à la disposition du public pour appuyer la poursuite des recherches sur ces sujets importants.

Title : An intelligent system for coffee grading and disease identification

Keywords : Deep learning, Convolutional neural network, Ensemble learning, Small datasets, Coffee.

Abstract :

One of the key factors in the success of deep learning models is their ability to learn important representations from the input data automatically, without the need for human experts designing task specific features. However, to learn these representations, deep learning methods usually require large amounts of data, which are expensive to obtain especially because of the efforts required for gathering and labeling data. This thesis investigates the applicability of deep learning for real world situations where data exist in small amounts, collected at different locations (labs), using different acquisition techniques and with minimally controlled conditions. There are two main contribution. First, we tackle the problem of coffee plant disease detection, which up to now was not approached in the literature, using a dataset containing images of coffee plant disease downloaded from the internet and that we captured in the farm, and a transfer-learning approach. It was possible to design a model that classifies coffee plant diseases with test accuracy of 90.18% using only 562 images of coffee plant. Second, we tackle the problem of coffee beans grading. A dataset for coffee beans grading is created by capturing images of coffee beans at the Jimma branch of Ethiopia's Commodity Exchange (ECX) in two rounds. In an attempt to solve the dataset shift that occurred in the two sets because of illumination and camera differences, color correction algorithms were added to the existing image processing and data augmentation pipeline. However, these techniques did not improve the performance of the model when compared to the commonly used preprocessing methods. This indicates that the difference in the image acquisition techniques was not the only reason for the dataset shift. We proposed and evaluated a network architecture that combined with data augmentation and ensemble learning techniques, led to an improved classifier (89.1% accuracy on the test dataset) that grades coffee beans, which performs better than the classical machine learning approaches (25.47% improvement) and off-the-shelf deep learning models (18% improvement). The coffee beans dataset and the models will be made publicly available to support further research on these important topics.