A Flexible Packing Optimization Approach to Improve Product Quality and Worker Training

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Roshanak Akram

August 2019

To the love of my life, Reza

To my parents, Mozaffar and Sousan

To my late grandmother, Touran, who always wished to see my graduation

Acknowledgments

The success and outcome of this dissertation required a lot of guidance and assistance from many people, and I am incredibly privileged to be given so many great opportunities throughout the course of my graduate studies. Above all, I am deeply indebted to my advisor, Dr. Sawhney, for his continuous support during my Ph.D. life. I have learned that success is truly achieved when you improve one's life. He has inspired me to become an independent researcher and helped me realize the power of critical thinking. I am also extremely grateful to my doctoral committee members, Dr. Kobza, Dr. Zaretzki, and Dr. Yu, for reviewing my work and providing me with valuable feedback that has improved the quality of my research. I want to express my sincerest thanks to Dr. Ninad Pradhan, who helped me in this project from the initial phases and always provided me with the most valuable feedbacks. None of this would have been possible without such supervision and assistance.

I am thankful to my fellow graduate students at the Center for Advanced Systems Research and Education (CASRE) for making this journey so productive and memorable. Thank you, Carla Angelo-Arbogast, for your unconditional support.

I sincerely thank my parents, Mozaffar and Sousan, for their unconditional love, timely encouragement, and endless support. My hope is to make their life happier by accomplishing what they have always wished for me.

Last and foremost, I would like to thank my lovely husband, Reza, for his unconditional and endless support, inspiration and encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without him. Thank you for making each day of my life colorful.

Abstract

Placing items in cases and then forming pallets are among significant steps for distribution and storage of final goods. Suboptimal packing may also lead to bulged cases, which can lead to pallet instability and other transportation hazards. Delicate items such as food packets may suffer quality damage at this step, by way of forced packing. Optimized Standard Operating Procedures (SOPs), which assume all bags are rigid bodies, improve the situation but do not resolve it. The insight offered by this research stems from the observation that there is a growing use of flexible packaging for food items. The flexibility of packages can be used in creating new configurations of the same bag, by folding it in various ways. The presented framework delivers a mathematical model capable of dealing with the increased complexity of flexible configurations. The model is linearized to reduce the run-time. The results are validated by a case study at a packing facility for a large governmental organization in the United States. A user-friendly interface generates an animated SOP to simplify the training process. The impact of flexibility is demonstrated using two metrics: "utilized height" of a packed case, and "unutilized space" in the case. The model easily outperforms rigid body optimization in all examined situations. An automatic visual inspection model is then developed based on a deep learning algorithm that can classify packed cases to acceptable and defective classes. Further analysis has shown that the defect can be localized, which is helpful in identification of the packing step that led to defect formation. The proposed model is a manifest of the smart factory. The intelligent automation model allows packing facilities to be responsive to product innovations and resultant packaging changes. The theoretical contributions made in allowing "flexibility" in the packing model have broad implications for container-loading problems. The integrated SOP generation and worker training framework have a direct benefit to workers as well as managers.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Material handling is one of the significant steps in production and supply chain management. The main preparatory step is to pack smaller items – called "bags" – into larger containers – called "cases" or "cartons" or "containers". This is followed by stacking cases onto pallets, or "palletization" (see Figure 1.1). Packing and material handling have an impact on quality assurance, which is a significant element of overall digitalization.

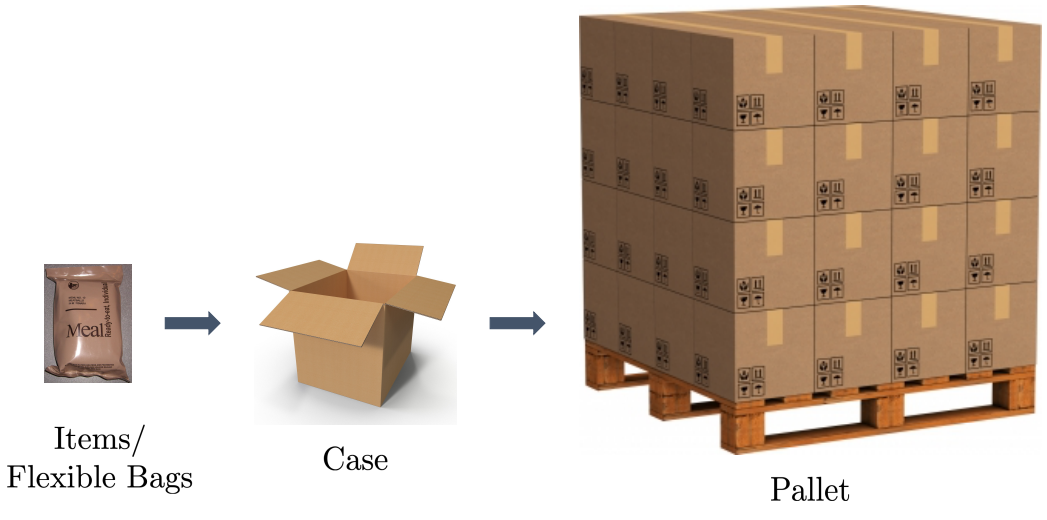


Figure 1.1: Packing and Shipping Units

The total packaging expenditure for 2015 was an aggregate of 839 billion (USD) globally in which North America has about 27% share of this amount (see Figure 1.2). US packaging industry valued 227 billion (USD) (Neil-Boss and Brooks, 2012) against the total Gross Domestic Product (GDP) of 18.037 trillion (USD). The packaging industry forms 1.25% of the GDP, and without packaging, the GDP of a country would significantly decrease in value. This is simply because, without proper packaging, no product can be shipped from supplier to manufacturer, retailer or consumer, through the supply chain.

Operational Excellence (OE) is the implementation of the business strategy to attain superior performance in productivity, quality, and delivery of services or products across the value chain. OE spans product design and development, manufacturing, packaging, shipping as well as operational effectiveness of people and processes. Since OE is an ongoing effort, the journey is the thing that matters, not the destination. With the emergence of Industry 4.0, the entire value chain can be facilitated by using approaches such as Smart Manufacturing and the Internet of Things (IoT). By transforming the existing raw data into actionable intelligence, it is possible to enable decision makers to take the right action at the right time and continuously improve systems performance. Industries dealing with producing products that are subject to dynamic change can benefit the most out of having an integrated system

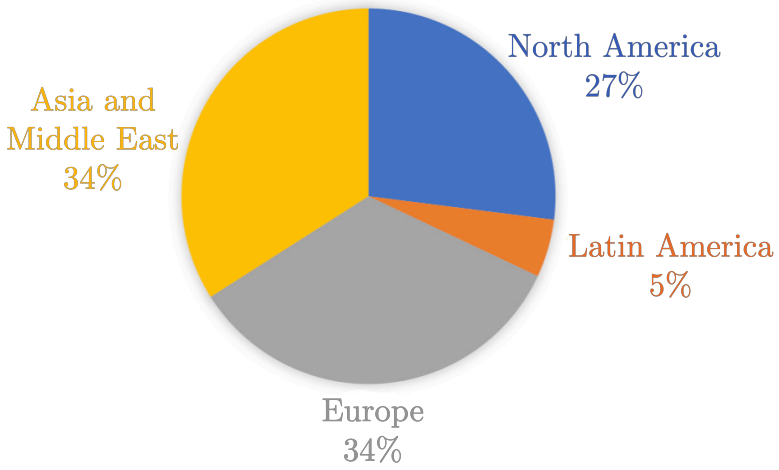


Figure 1.2: Global Packaging based on Geography

in which the decisions are made based on the most updated information shared through digitalized processes.

"Packing" refers to the way that the items are placed into the bigger box or container and "packaging" relates to the specific wrapper of the elements. The *absence of a packing strategy* leads to interrelated quality issues. First, ad hoc packing practices cause an overfilling of cases, leading to a case deformation reported in industry studies as a "bulge", which has been directly linked to transportation stability issues in pallets (Orloski, 2005). Second, compression machines used to mitigate bulge deformations (Leblanc, 2015) have been reported to accelerate the failure of a case (Hall, 2011). Third, the use of forced packing adversely affects the transportation of delicate goods; for example, the food industry reports that the physical damage to cookies or crackers affect a customer's sensory perception of the product quality (Booth et al., 2003). *Inadequate worker training* exacerbates these quality issues (Prinz et al., 2016) and has been the focus of solutions; for example, for a packing case study at a 'Liz and Claiborne' facility (Orloski, 2005).

As stated earlier, operational excellence must focus on delivering ***quality*** products (Flynn et al., 1995; Basu, 2004; Oakland, 2014), which makes the quality degradation resulting from improperly planned and executed packing unacceptable. The lack of efficient information sharing between manufacturing and packing department as well as inadequate or inappropriate training have adverse effects on the quality of the product in the bags. This is a consequence of using compression equipment such as rollers and pressers (Leblanc, 2015), and is especially pronounced in food packing. Contents inside the bag, especially delicate or brittle items such as cookies or crackers, are susceptible to getting crushed or powdered. The textural change influences the perceived sensory quality of the food; this has been documented in the cases that include cookies (Booth et al., 2003). The stated problem is addressed by presenting an automated solution to the packing problem that simultaneously addresses the core issues of developing a packing strategy and improving worker training. The solution is consistent with the "smart factory" emphasis on quality and organizational ability to quickly adapt to packing requirements for new products with dynamic alternations (Burke et al., 2017).

This dissertation is an attempt to address conceptual and practical issues in the area of "smart factory" within the broader framework of Industry 4.0. (Zuehlke, 2010; Hessman, 2013; Alessi and Gummer, 2014). Items with flexible packaging are becoming more popular, and more companies set the goal of using more flexible packaging (more environmentally friendly). The application of smart factory principles to "flexible packing" – in which bags packaged with flexible material are packed into cases – has become increasingly relevant with the growing popularity of flexible packaging material and decreased product lifecycles. The optimization of flexible packing processes and designed training is aligned with Industry 4.0 principles of *transparency* - since the packing procedure is outputted as a standard operating procedure for the workers - being *proactive* - by tracing a mistake in a packing step to the observed defect in the final product - and *agility* - by helping organizations adapt to fast changes in the product design and to a shortened lifecycle (Burke et al., 2017).

One of the significant differences between flexible and rigid packaging is that the items can have different configurations, with different dimensions. Moreover, the flexible nature of the items enables us to change the configuration in a way that is more efficient in utilizing space – when packing them in a case or a larger container. It is intended to find out the best configuration of bags, by folding them, as well as the best placement strategy (as an operational plan) to ensure that space is efficiently used and the quality is not degraded by any extra force. The proposed model illustrates the best packing strategy for the workers of the line by training them interactively. The presented solution in this dissertation is aligned with the solution proposed in the Liz and Claiborne center (Orloski, 2005), where they concluded that occurrence of overfilling could be reduced by improving worker training which results in case quality improvement. In a nutshell, this work identifies four purposes which address the automation and integrated system for improving quality in the presented context:

1. Efficient utilization of the space in the case by means of an improved packing strategy
2. Development of a Standard Operating Procedure (SOP) to simplify training
3. Controlling the quality of cases after implementing the packing strategy with an automated visual inspection technique

4. Associating any detected defect of cases with the most influential packing step; presenting an accountable model for packing and worker training

The presented approach systematically identifies the optimal sequence and placement of bags inside a case. This methodology satisfies the objective of efficiently utilizing the space inside a case in addition to the elimination of the need for mechanical compression of the case and its contents. The optimal sequence is rendered as a simple graphical visualization which serves as a training tool. This serves as the SOP and achieves the second objective. It also establishes accountability on the assembly line, since the SOP guarantees a compact configuration of bags inside the case. Any deviation, in the form of a bulge, may then be traced back to the corresponding assembly station that was most influential. The identification of localized bulges motivates accountability in the system since the identified stations may then undergo additional training. A direct consequence of the presented approach is that it improves pallet stability since bulged cases are known to cause transportation stability issues in pallets (Orloski, 2005).

1.2 Problem Statement

Let us assume that in a production line, the produced goods are packed in bags which must be placed in a case afterward. The requirement is that cases must be sealed and palletized with no visible bulges and with no forced packing. This stipulation ensures that pallets are stable and that delicate contents inside bags are protected from damage. It is assumed that bags are made from flexible packaging material. This category of bags is typical in food packaging, where contents are filled in, and a certain head-space is not used per bag, followed by an optional step of vacuum sealing to reduce volume. It is assumed that the dimensions of the bags are approximately known or measurable. Shifting of contents inside bags which are not tightly packed can cause shape variations. It is assumed that the assignment of bags to a case is fixed and that the case dimensions are known. The primary objective of the work is to minimize the height occupied by packed bags inside a case, with no changes required for case length and width. This allows the production line to use the same cases as the current selection.

Three additional goals are introduced, which the approach must satisfy. First, the packing approach must consider all folding options for a bag, including no folds. This avoids the assumption that every single bag must be fully folded to reduce the unused spaces and allows the algorithm to explore and analyze all alternatives. Second, the optimization must be parameterized such that an arbitrary change in bag dimensions does not incapacitate the algorithm. This type of standardization is essential to production lines in which the contents of the bag or the size of the bag may change periodically. Thirdly, worker training must be simplified by requiring interactive training to be an output of the methodology. The standardization achieved in the previous step can be leveraged to create this type of training output. The advantage of including the training as an objective is that frequent packing errors traced back to individual assembly stations may then be corrected by retraining the workers at those stations.

In a case study, presented in Chapter 4, product modification is an inevitable activity among the continuous improvement endeavors to reach operational excellence for the selected organization.

1.3 Research Context

In the manufacturing environments where the products are subject to dynamic changes, we found out that the main focus is on product development and no such effort is spent on sustaining the quality of the delivered product. This misalignment between product/packaging development and packing strategy development – as a basis for transporting goods – will result in providing items to the customers with lower quality than promised which could be a drawback in OE. Therefore, this dissertation aims to fill this gap in the OE framework by proposing an optimization-based strategy rooted in Industry 4.0 goals of "quality" and "worker well-being". In other words, the purpose of this study is to improve the quality of delivered items by presenting a packing optimization and accountability model. The proposed model is designed and implemented for a governmental organization.

The central research contribution of this dissertation is the development of an optimized packing model which relaxes the assumption of the rigidity of bags. There is no model in

the literature that utilizes the flexibility of the items to use space more efficiently (please see Section 2.2.1). The proposed model uses the style of folding the bag – for example, length-wise or breadth-wise – as a decision variable. The outputs of this model are, therefore, the location, orientation, and folding style for a bag. The complexity of modeling the optimization increases significantly, but its solution has positive consequences for reduced overall space consumption and reduced bulges. The model is nonetheless practical in terms of time utilization since it has been linearized and can be solved using linear techniques. The output of the optimizer is the SOP: an animation of the packing steps is generated for each packing station in the line. A full-featured version of the model was implemented as a case study for a governmental organization.

The results of the model are then validated by an automated visual inspection (AVI) technique, to ensure the assumption that the proposed flexible packing model is reducing the odds of producing cases that are bulged, more importantly, avoid packing cases that include some items with low quality. AVI systems are image or video based inspection that is vastly used in industrial and manufacturing environments. Vision systems can measure items, identify the ones that are not in a correct or standard position and also identify the shapes. By the aid of Artificial Intelligence (AI) techniques, the vision systems can also make decisions such as passing or failing a specifically defined standard. The main reason for having the automated inspection system is to develop an accountable framework which means that in case any defect is identified, the system reveals the step in which the results of the optimization (the generated SOP) was not followed correctly. Therefore, the contributor of the defect or bulge will be re-trained, and there will be less bulged cases in the future.

The presented research contributions are evaluated in a comparative context with existing packing methods which assume bags to be rigid bodies. The following questions are evaluated for the comparison: 1. Does the flexibility assumption change the packing sequence compared to the rigidity assumption? and 2. To what extent does the flexibility assumption improve space consumption inside a case?

1.3.1 Scope and Limitation

This study covers an open dimension packing problem in cases of having flexible items. The presented methodology can be implemented in any packing industry with the goal of placing small flexible packages into the larger rigid case. The problem was originated from food supplying facilities for a governmental company but could also be used in any other food industry that has the challenge of delivering high-quality food to the end customers.

Given that the dimensions of the packages are measurable, a solution for optimally packing the small bags into the cases can be found. Since human errors are inevitable in the implementation plan, an automated inspection module is developed which is capable of classifying the cases to acceptable and not acceptable groups. The goal for this classification is to build an accountable model being capable of identifying the source of any error. In the cases that these errors are human error, retraining could be done to avoid further defects in the system.

The limitation of this study is that only the most doable configurations for folding the flexible bags are considered, while there could be more options. As it was mentioned, human error exists in the proposed methodology, and more in-depth studies can be done in this area. One possibility is to use a depth camera and identify the packing step that is no in accordance with the SOP. Moreover, the problem of how to pack the contents inside the bags is not studied in this research since it was out of the scope of the case study.

1.4 Approach

The selected approach in this dissertation is classified into two tasks that are explained in the remainder of this section and more details are presented in Chapter 3.

1.4.1 Optimization

Based on the previous sections, the problem is to optimally place bags (small items) to a case (container). After the literature survey, it has been concluded that the problem is in the family of open dimension problems, where we want to minimize the height of the case

(please see Chapter 2 that covers a detailed literature review). The nature of this problem necessitates us to decide on the placement of the bags in a different manner from traditional models for rectangular items. In the related literature, there are plenty of studies on the placement of rectangular items, and some other studies on irregular objects, such as spheres or circular items. There is a lack of focus on the items that are similar to rectangular objects but made with flexible materials which make it hard to be measured in a single set of deterministic dimensional values. A graphical representation of flexible packaging is illustrated in Figure 1.3. Based on the nature of this problem, it was understood that different folding options help in reduction of un-utilized space. So a new set of decision variables is introduced for knowing the best folding option for each bag.

1.4.2 Inspection

An automated visual inspection system is designed to identify the cases whose shapes fall outside acceptable thresholds. A deformed case, when placed on a pallet for transportation, can upset the spatial distribution of the pallet and make an entire load of cases unstable. An unstable pallet is likely to topple and ultimately damage the contents of one or more cases. This makes it essential to have an inspection system which can spot a deformation at the vendor site before it propagates along the supply chain and leads to broader quality defects.

A schematic of the inspection system is shown in Figure 1.4. The packed and sealed case is placed in a designated inspection area which has an inspection rig with a downward facing camera. The camera is connected to a computer which runs the inspection algorithm and classifies a case as **passed inspection/good** or **failed inspection/bad** depending on its

Figure 1.3: Representation of Flexibility in MRE Bags

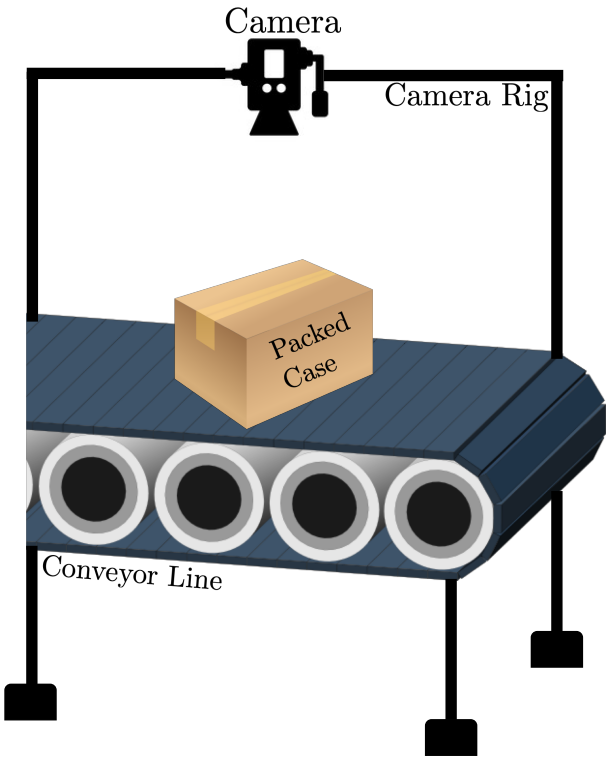


Figure 1.4: Visual Inspection Setup

level of deformation. Then a visual alarm shows if the case is ready for palletization or it should be taken out of the line for finding out the root cause of the problem.

The objectives of the inspection system from the point of view of the camera algorithm are illustrated in Figure 1.5. To recognize deformation, the shape of a case needs to be captured and then fed to the Convolutional Neural Network (CNN). CNNs are among the models that are inspired by biology (Matsugu et al., 2003), and are classified as a category of multi-layer neural networks. It has to be noted that CNNs perfectly fit for the tasks such as visual pattern recognition directly from color images where minimal pre-processing effort is required. Moreover, CNNs have shown promising success in the domain of image classification, and more generally, in computer vision.

The visual inspection problem may be simply stated as follows: Given shape information for a sufficient number of examples of good and bad cases, develop a learning algorithm capable of classifying a new, previously unseen, case as **failed inspection** or **passed inspection**. The development strategy begins with a data collection phase. In this phase, the algorithm is exposed to examples which may be **positives**, i.e., cases which pass

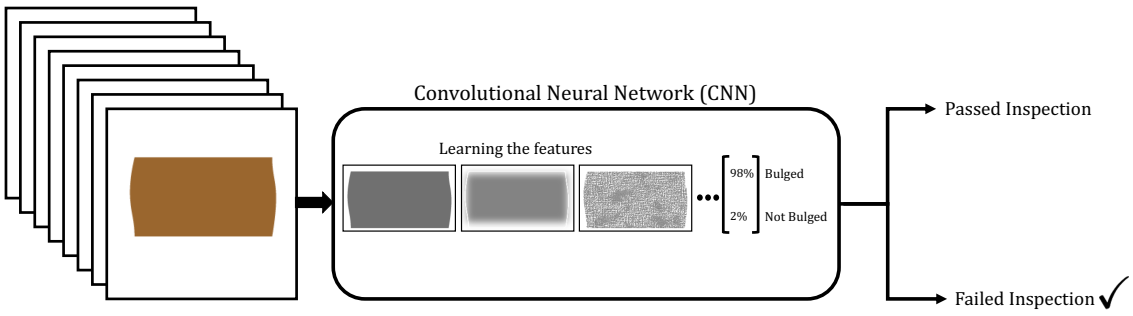


Figure 1.5: Conceptual Representation of Inspection Objectives

inspection, and **negatives**, i.e., cases which fail inspection. A human provides information about pass/fail; hence this machine learning technique is called supervised learning.

Samples from the data collection phase are processed during the **training** phase. In this phase, the learning algorithm is required to internally develop a definition for the level of deformation which will place a case in the pass or fail category. Various training algorithms are implemented, and the one which performs best for pass/fail classification is selected. The training phase is followed by the **validation** phase. During testing and validation, cases are placed in front of the camera. Their pass/fail labels are known by the human and the test of the learning algorithm is to attach the correct label to each test case. Failing this, the algorithm reenters training followed by testing until it has been trained to perform at satisfactory accuracy levels.

It is intended to connect defects observed during inspection to actions during packaging which is their likely cause. This is a statistical inference type of problem, in which an observation is connected to the event which has the maximum likelihood of being its cause. The developed model will consider its observation to be: (1) to identify the defect, (2) its location on the packed case, and (3) the packing steps that most likely caused this defect. The causative events for this are derived from the optimization model. The overview of the planned methodology is presented in Figure 1.6.

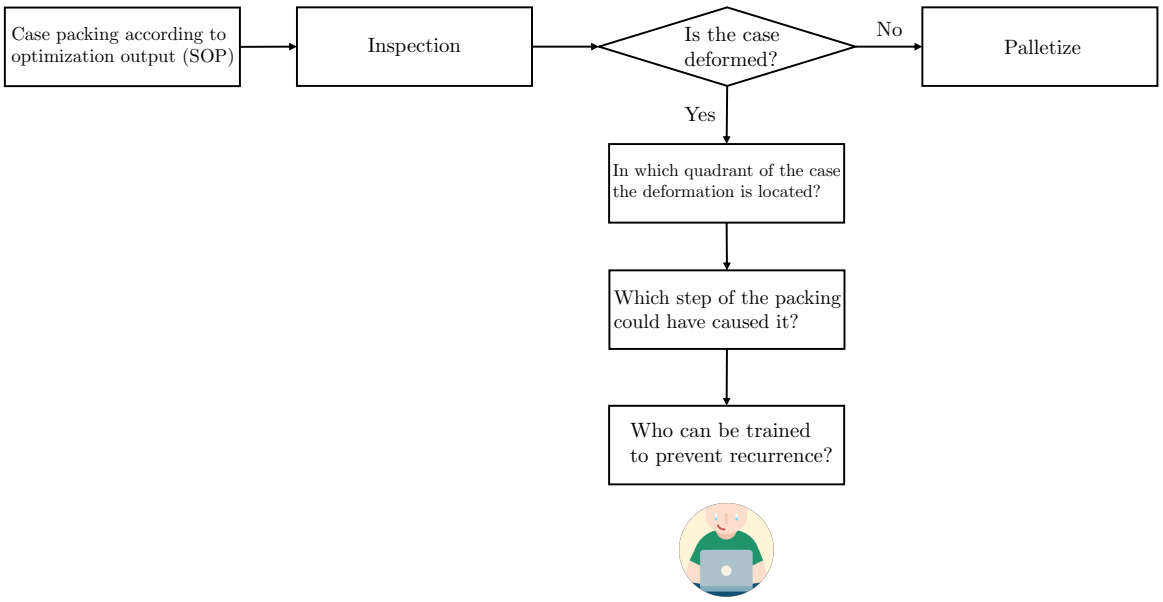


Figure 1.6: Methodology Outline

1.5 Summary

To summarize, this research has the following significant steps:

1. A **flexible-object optimization model** is designed. It permits bags to be folded in specific ways to improve space utilization inside the case. This is a realistic model of packing flexible bags inside rigid cases. By allowing them to fold (considering variable dimensions), we make a technical contribution to the area of optimization while providing the partner company with a practical approach for placing bags into cases.

2. The optimization model will **immediately accommodate changes to any alternations in the bag sizes**. Dimensions of new bags are input to the model. It calculates the packing solution for the new items and creates instructions and a visualization video to assist assemblers at vendor sites. Thus, **SOP generation is automated**.

3. Deep learning algorithms are utilized to identify not only whether a case has a defect, but also to **classify the location of defect observed**. Assemblers will get access to a

lot more information about the types of defects observed, their trends and seasonality. This will generate a powerful feedback mechanism for managers to improve their processes.

4. A multi-label classification model to link packaging defects to root causes is developed. This will connect an observed defect to the packaging step which most likely caused it, and the person or team accountable for it. **Accountability for packing quality** is embedded in the process and will lead to fewer defects and an added sense of responsibility.

5. **Focused training can be provided** if a packing step or process is found to have repeated quality issues at a vendor site, then training related to those issues may be provided to a team or individuals.

1.6 Outline

This dissertation includes five chapters. Chapter 1 introduces the context of the studied problem. Chapter 2 presents an extensive literature review. Chapter 3 and Chapter 4 explains the methodology chosen for solving this problem and the results respectively. Chapter 5 summarizes the findings, the implementation of the model, and future work.

Chapter 2

Literature Review

This chapter summarizes the literature review in two main categories of optimization and inspection. The first category presents a summary of optimization models/techniques used in the family of packing problem. The second category covers the deep learning techniques that are used to detect any defect in the manufacturing line where the main focus is on convolutional neural networks (CNNs). In each section, the related publications have been reviewed, and research gap is identified.

2.1 Optimization

The family of packing problems has been extensively researched. One of the earliest review papers in this field is the survey by Dowsland and Dowsland (1992) on the operations research applications for packing problems such as pallet loading and container loading. The authors show that most variants of packing problems are NP-complete, the solutions to which may be effectively discovered only by heuristic methods. Wäscher et al. (2007) present an operations research-based typology for cutting and packing problems in which objects are organized into homogeneous categories based on their characteristics. This work improves over previous categorization efforts by Dyckhoff (1990). The survey by Bortfeldt and Wäscher (2013) presents various constraints in container loading problems and reviews modeling approaches, including exact and heuristic algorithms. Zhao et al. (2016) review solution methodologies, provide an analysis of critical algorithmic design issues and compare

state-of-the-art algorithms in container loading. Literature in the field, as understood from the sources cited above, reveals the following classes of optimization problems:

1. Selection of case size suitable for the bags
2. Selection of bag sizes suitable for the case
3. Clustering of bags to fit the case
4. Allocation of subsets of bags to collectively fit multiple cases
5. Configuration of bags inside cases

The work presented in this dissertation belongs to the fifth class of problems from the above list. The configuration problem is further defined by the notion of "heterogeneity", presented by Bortfeldt and Wäscher (2013). Bags are considered to be "weakly" heterogeneous when they can be grouped into classes based on shape and size similarity. Each item in a class is present in relatively large quantities. On the other hand, bags are considered "strongly" heterogeneous if none or only very few share an identical shape and size. These definitions form the basis of classification of container loading problems into two families (Bortfeldt and Wäscher, 2013) - **output maximization** and **input minimization**, outlined below with terminology modified to be consistent with this dissertation:

1. In **output maximization**, subsets of bags must be allocated to a limited set of cases. Output maximization includes the following type of problems:
    - Single Stock-Size Cutting Stock Problem: Packing a weakly heterogeneous set of bags into a minimum number of identical cases
    - Multiple Stock-Size Cutting Stock Problem: Packing a weakly heterogeneous set of bags into a weakly heterogeneous set of cases in a way that the value of the cases used is minimized
    - Residual Cutting Stock Problem: Packing a weakly heterogeneous set of bags into a strongly heterogeneous set of cases where the value of the cases used is minimized

- Single Bin-Size Bin Packing Problem: Packing a strongly heterogeneous set of bags into a minimum number of identical cases
- Multiple Bin-Size Bin Packing Problem: Packing a strongly heterogeneous set of items into a weakly heterogeneous set of cases in a way that the value of the cases used is minimized
- Residual Bin Packing Problem: Packing a strongly heterogeneous set of bags into a strongly heterogeneous set of cases such that the value of the cases used is minimized
- Open Dimension Problem: Packing a set of bags into a single case with one or more variable dimensions such that the occupied volume of container is minimized

2. In **input minimization**, the selection of bags for allocation into cases is not needed, since it is assumed that unlimited containers are available. Input minimization includes the following type of problems:

   - Identical Item Packing Problem: Loading a single case with the maximum possible number of bags
   - Single Large Object Placement Problem: Loading a single case with a selection from a weakly heterogeneous set of bags in order to maximize the value of the loaded items
   - Multiple Identical Large Object Placement Problem: Loading a set of identical cases when selecting from a weakly heterogeneous set of bags to maximize the value of the loaded items
   - Multiple Heterogeneous Large Object Placement Problem: Loading a heterogeneous set of cases when selecting from a weakly heterogeneous set of bags to maximize the value of the loaded items
   - Single Knapsack Problem: Loading a single case by selecting from a strongly heterogeneous set of bags to maximize the value of the loaded items

- Multiple Identical Knapsack Problem: Loading a set of identical cases with a selection from a strongly heterogeneous set of bags such that the value of the loaded items is maximized.
- Multiple Heterogeneous Knapsack Problem: Loading a set of heterogeneous cases with a selection from a strongly heterogeneous set of bags to maximize the value of the loaded items

Based on this classification, the presented work belongs to the group of **open dimension problems** in **output maximization**, in which a **heterogeneous** set of bags is packed into a case. Related optimization papers are classified into three groups: exact methods, heuristic algorithms, and meta-heuristic algorithms.

2.1.1 Exact Methods

The work by Chen et al. (1995) is among the first exact solutions to packing problems. The authors present a binary mixed integer linear programming model for the general 3D case loading problem. The problem involves packing a set of heterogeneous items into heterogeneous containers. The model considers various possible orientations, multiple item sizes, numerous case sizes, avoidance of items overlapping aimed at the minimization of non-utilized space.

Martello et al. (2000) address the packing of a given set of rectangular-shaped items that are orthogonal to each other into the minimum number of three-dimensional rectangular bins. A discussion is presented about the complexity of the problem, and lower bounds are shown to tackle this issue. The resultant algorithm leads to the incorporation of the original approximation algorithm with a branch-and-bound algorithm for 3D bin packing problem. Packing of up to 90 items can be solved to optimality within a reasonable time-frame.

Faina (2000) present a geometrical model which reduces the general 3D packing problem to a finite enumeration scheme. The objective is to find the optimal way of placing a given set of rectangular cartons within a minimum volume rectangular container. The model is validated for several test cases. The algorithm is shown to be efficient when the number of items is relatively small.

Eley (2003) present a set partitioning formulation for the container loading problem. Initially, the authors generated solutions for a single container loading problem. The solutions can be joined together in the second stage to be generalized for multiple container loading problems. The inclusion of realistic constraints such as load stability and box orientation and the performance of their model made their approach unique. Moreover, the authors compared the computational results with existing benchmark problems and proved their presented integer program can be solved to optimality within five seconds. In a review paper published by Zhao et al. (2016) it is mentioned that proposed algorithm by Eley (2003) outperforms two of the previous algorithms in the literature presented by Ivancic (1988) and Bortfeldt (2000).

Li et al. (2003) propose a slightly different solution method in which the problem is reformulated to a model with non-overlapping constraints. The objective function is then linearized, and decomposed sub-problems are solved by a distributed computation algorithm to find the global optimum solution.

Stoyan et al. (2003) study the packing of solid spheres into a parallelepiped with the objective of height minimization. The solution strategy has three different phases and is capable of finding packing solutions for up to 60 spheres. Birgin and Sobral (2008) perform a similar study to find the smallest object within which the heterogeneous items can be packed. In this work, multiple item shapes are considered, including circle, triangle, square, rectangle, and strips in two and three-dimensional geometrical space.

Martello et al. (2007) explore orthogonal packing of a given set of rectangular boxes into the minimum number of three-dimensional rectangular bins. This is an extension of their previous work (Martello et al., 2000) which combines the original enumerative approach with a new constraint programming approach to develop a generalized single-bin packing procedure. The outcome of the study enables robot-packable solutions to the problem.

An approximation algorithm that could be applied to 2D and 3D bin packing as well as a 3D strip packing problem is presented by Miyazawa and Wakabayashi (2009). A general parametric packaging is developed which can be extended to other problems or dimensions.

Junqueira et al. (2012) present a mixed integer linear programming model for the container loading problem considering the cargo stability horizontally and vertically as well

as the load bearing strength of the cargo, including fragility. The performance is analyzed using optimization software with 100 randomly generated instances.

De Queiroz et al. (2012) present algorithms for 3D guillotine cutting problems in an unbounded knapsack, cutting stock, and strip packing. Two cases were studied; in the first, items have fixed orientation, and in the second, orthogonal rotations around all axes are permitted.

Zhu et al. (2012) present a new column generation method - called prototype column generation - which is applied to container loading problem. Since the model is formulated as a set cover problem with a huge number of variables rather than constraints, the direct application of the column generation is not practical. Therefore, the original problem is decomposed to pricing subproblems where instead of actual column generation (computationally costly), an approximation method is used. This strategy is helpful in reducing the average gap from the optimal solutions by 50% when compared to Che et al. (2011).

2.1.2 Heuristics

The second family of studies is focused on developing heuristics for specific applications. Bischoff and Marriott (1990) provide a composite case study on development of composite heuristics presented up to 1990, including Dowsland (1984); George and Robinson (1980); Bischoff and Dowsland (1982); Conover and Conover (1980); Langston (1987). One implication from the composite heuristic is that the sequence of using heuristics is not important and packing efficiency depends on the number of boxes to be packed.

Li and Cheng (1990) investigate a 3D packing problem to minimize the height of the container. The authors base their contribution to multiple studies related to on-line packaging algorithms, known as level-strip algorithms. A particular case of 2D and 3D column packing problems is solved in their papers where the efficiency of those algorithms was still unknown.

Bischoff and Ratcliff (1995b) present one of the earliest contributions to heuristic literature for container loading problems. Two criteria are selected to generate a separate heuristic. The first criterion is even distribution of packing arrangements, whereas the second

targets multi-drop situations. Their study on randomly generated examples shows that the developed heuristics are complementary to each other when coping with different practical requirements. Bischoff and Ratcliff (1995a) study the problem of loading pallets with non-identical items which also evaluates the conditions under which the consignment cannot be accommodated on a single pallet. A greedy procedure is developed and evaluated as a solution. Ratcliff and Bischoff (1998) extended the previous study by considering the weight distribution in the container. The proposed model is solved with a heuristic approach in which boxes are iteratively packed in layers (with at least two boxes of identical type and orientation) from the floor of the container. The algorithm checks the load bearing constraint at each step which prevents the packing of layers with low load bearing at the early stages.

Lai et al. (1998) present two packing solutions as well as a graph-theoretic model and compare the results with a heuristic. The work does not assert optimality due to the complexity of the problem but concludes that the quality of heuristic solutions is high.

Chien and Wu (1998) introduce three-dimensional guillotine cutting technique applied to the problem of packing items with different sizes into a container with a predefined size. In the same context, Pisinger (2002) present a heuristic-based method which is called "tree search" and is an extension of the wall-building approach. Based on the presented results, the implementation of the heuristic resulted in utilizing more than 95% of the available space.

Lodi et al. (2002) introduce a new constructive heuristic within the framework of tabu search. The algorithm was tested on the existing benchmarks in the literature of packing problems (Martello et al., 2000) and compared with the results of guided local search heuristic that was previously published by Faroe et al. (2003). In the majority of the test cases, the approach proposed by Lodi et al. (2002) dominates other heuristics.

The heuristic method presented by Chien and Deng (2004) is based on spatial matrix representation for reducing the empty spaces. Unlike the papers that have been summarized, the authors present a simulation platform for visualizing the results and have a step-by-step user guide of packing pattern.

Moura and Oliveira (2005) improve the heuristic presented by George and Robinson (1980) in a way that instead of considering an infinite-length container, authors considered

a length equal to or less than the container's length. The presented results reveal promising effectiveness for achieving cargo stability and volume utilization.

Bortfeldt and Mack (2007) present a heuristic for 3D strip packing problem derived from a branch-and-bound approach for container loading problem. The performance of this heuristic exceeds other methods in the literature in terms of utilized volume and computational efforts.

In the framework of greedy heuristics, Ren et al. (2011) present a tree search method with five evaluation metrics. Authors propose packing block of boxes (made with identical items that have the same orientation) into a container. Selection of different blocks is evaluated with the evaluation metrics. The algorithm was tested with benchmark test problems (Bischoff and Ratcliff, 1995b; Loh, 1992) and show high average utilization of the volume.

Liu et al. (2011) present a hybrid tabu search composed of tabu search and loading heuristics iteratively. The proposed algorithm can handle the computational complexity of larger scale problems as well as outputting the loading arrangements of the boxes by block construction. This model considers the weight distribution of boxes in the container which exists in many real cases. The algorithm was tested with real-world data and not only showed promising results in volume utilization but also in container stability.

Allen et al. (2011) present a hybrid placement strategy for 3D strip packing problem to minimize the container length. The efficiency of the solutions is examined when only a heuristic is used. Integration of this heuristic with some meta-heuristic methods further enhance the solutions.

Lim et al. (2012) present a tree search heuristic that uses dynamic prioritization to handle the "trouble-making" box types. The priority associated with each box is revised at the end of each iteration as a measure to reveal how trouble-making each box is. In the single container case, the objective is to maximize the volume utilization with consideration of load stability, weight distribution and the load bearing strength of boxes. The results of the model are then visualized to be easily implemented in the target industry.

Zhang et al. (2012) introduce an efficient heuristic originated from multi-layer search. Authors presented a new strategy called "composite block"; Unlike traditional blocks in multi-layer packing, composite blocks are composed of boxes of different types. Moreover,

the proposed algorithm is developed based on a depth-first search in each packing iteration to guaranty that the algorithm is moving into the direction of optimality. The model was tested on 1500 instances where in most cases the proposed study outperforms other similar algorithms. In large-scale problems, this method requires more computational time which can be counted as a shortcoming.

Liu et al. (2014) propose a new binary tree search heuristic that takes full support, orientation and guillotine cutting constraints into account. Forming strips and then layers are necessary for building the binary tree where each node in the tree represents a container loading pattern.

Sheng et al. (2017) present a heuristic with consideration of expiration dates for the items to be loaded and sent. The proposed method is composed of four steps in which the expiring orders are handles in the first and second step by applying simulated annealing.

In block-building based heuristic approaches, Araya et al. (2017) propose a new evaluation function to rank boxes. The proposed function rewards the boxes that fit well in the container and also penalizes the placements that result in volume waste. Authors mention that for iteratively placing blocks, a covered surface area of the blocks is more important than wasted volume in the container.

2.1.3 Meta-heuristics

The last group of studies is focused on meta-heuristic algorithms for the packing problem. Many of these algorithms are based on a specific order of boxes. In addition to making the boxes sorted, some algorithms are based upon the permutation of the boxes. It has to be noted that meta-heuristics do not guarantee that the solution found is globally optimal.

Corcoran III and Wainwright (1992) present the first 3D bin packing problem with the genetic algorithm. Moura and Oliveira (2005) present a new GRASP (greedy randomized adaptive search procedure) algorithm for solving the container loading problem when the dimensions of the container are known and fixed, and the objective is to optimize the volume use as well as the load stability. The results illustrated that the quality of the solutions made by their algorithm surpasses the other ones presented by the other researchers of the field.

Bortfeldt and Gehring (2001) used the heuristic that was presented by Gehring et al. (1990) to generate the initial solutions that are constructed based on Layer Determining Box (LDB). Then they presented their genetic algorithm by applying crossover and mutation function to produce new offsprings. The top offsprings are then selected based on their stowage plan, and the final re-sequencing of the layers is implemented for determining the best weight distribution. The authors presented a parallel genetic algorithm to improve the efficiency of the algorithm in addition to the quality of the solutions found by their previous algorithm(Gehring and Bortfeldt, 2002).

Takahara and Miyamoto (2005) propose an evolutionary approach in addition to a heuristic method for multiple container loading problems. The sequence for loading pairs of packages is determined by a genetic algorithm, followed by a heuristic procedure to identify loading position of each box. The evaluation of algorithm performance demonstrates the superiority of meta-heuristics. In a subsequent study, Takahara (2006) address multiple containers and pallet loading problems. The objective of their research is to provide meta-heuristics that determine the sequence of packages, and the strategic procedure determining the series of containers and pallets with reference to the search process.

Egeblad et al. (2007) apply a meta-heuristic method called "Guided Local Search", which is capable of searching all the neighborhoods in polynomial time to produce robust results.

Liang et al. (2007) present a two-phased meta-heuristics. Initially, ant colony optimization is employed for extracting useful information that could support the genetic algorithm. The problem is solved with the objective of improving the utilization ratio.

Wu et al. (2010) study the 3D bin packing problem, in which items are allowed to have different possible orientations. Considering orientations adds to the complexity of the problem. The work includes the design of a special bin packing algorithm based on packing index. A genetic algorithm is developed to find solutions to this version of the bin packing problem.

Dereli and Das (2011) propose a hybrid algorithm with the objective to maximize the total volume of packed boxes. The algorithm is inspired by foraging behavior of honey bees, and therefore it is called "bee algorithm" (BA). Like genetic algorithm, BA also works with a population of solutions and evaluates their fitness values. The initial set of solutions are

created by following the heuristic proposed by Pham et al. (2006). Then the best solutions and some newly generated ones are selected to create the next population. The focus of this permutation procedure is to change the rotation of the boxes in a way to find the alternative width of layers for better performance.

Maleki (2013) reveal that no existing analytical model in packing literature considers moldable or flexible items. A hybrid operations research model is presented, where moldable and rigid pieces have to be packed with each other. The shortcoming in that research is that it merely defines separate sets for moldable and rigid items; however, the formulation of the optimization does not treat moldable items differently from rigid items.

Can and Sahingoz (2014) illustrate a simulated annealing algorithm for solving the 2D container loading problem. The work demonstrates that an optimal choice of variable T (temperature) in simulated annealing leads to a global optimum and is therefore critical to the success of the algorithm.

In some of the cases in the industrial and commercial application, it might be necessary to consider multiple objectives such as the study presented by Zheng et al. (2015). In this research, a multi-objective multi-population biased random-key genetic algorithm is illustrated that recognizes space utilization of the container and the total value of boxes as the objectives.

Based on a multi-population biased random-key genetic algorithm, Ramos et al. (2016) propose two versions of a hybrid genetic algorithm that considers static stability constraint within the three-dimensional rectangular container loading problems. The static stability constraint is based on the static mechanical equilibrium conditions derived from Newton's laws of motion.

2.2 Inspection

In this dissertation, the results of the optimization model are validated, and an accountable model for training the workers is developed on the basis of a machine vision system. Computer vision systems mainly rely on tools such as digital sensors or industrial cameras

to obtain images. The images are the input data to the computer algorithm for the analysis of various characteristics used for further decision making.

Since quality control is one of the most critical pillars of the operational excellence, automatic inspection is becoming a significant area in the manufacturing and packing industries. Due to the slow pace of conventional defect detection methods and the risk of making errors, more enterprises now opt for automatic inspection systems. Moreover, higher production speed and less labor cost paved the way for growth in the popularity of these methods. Increased expectation of high-quality products from customers made the industries more responsible, primarily if they are committed to OE. As a solution to these problems, artificial vision based automatic inspection systems arrived. Since the size and cost of computer vision systems have decreased (Jain et al., 1995), at the same time power efficiency and ease of use of such systems made it an inevitable device for use in industry (Neethu and Anoop, 2015).

In a survey published by Huang and Pan (2015), the authors classified image-processing-based inspection to four classes based on the techniques they proposed. The classification is listed below:

1. Projection methods
2. Filter-based approaches
3. Learning-based approaches
4. Hybrid methods

The work that is done in this dissertation lies in the third category where machine learning and pattern recognition algorithms are used. According to LeCun et al. (2015a), neural nets and backpropagation methods were ignored by the computer vision community. The reason was that experts in these areas thought that simple gradient descent is not able to get out of the local optima. Being trapped in the local optima can result in weight configurations in a way that no small change would decrease the average error. Practically, it has been shown that being trapped in the local optima is rarely a case in large networks.

On the other hand, there are multiple saddle point is the solution region where the gradient is zero. Due to the fact that high error plateaus surround saddle points, the learning procedure is prone to becoming drastically slower, and give the false impression of the existence of a local optima (Dauphin et al., 2014). No matter what the pattern around the saddle points looks like, it has been shown that almost all of them have very similar values of their objective functions. Therefore, it does not matter if the algorithm gets stuck at the saddle points.

Since 2006, deep feedforward networks have attracted the attention of the researchers in the field of image classification. An unsupervised learning procedure was introduced by multiple researchers to create layers of feature detectors without having labeled data. Learning each layer of feature detector enables us to reconstruct the activities of feature detectors in the layer below. This reconstruction technique makes the pretraining of several layers of complicated feature detectors possible. As a result, the weights of a deep network can be initialized to reasonable values and then by attaching a final output layer to the top of the system, the whole network can be fine-tuned by applying a round of backpropagation. For instance, Hinton et al. (2006) introduced an efficient and novel approach for training very deep neural networks by pretraining one hidden layer at a time while using unsupervised learning procedure. Bengio et al. (2007) present that the unsupervised pretraining method that was previously introduced (Hinton et al., 2006) significantly improves performance on the test data. Moreover, the presented method is generalized to the other unsupervised representation learning techniques.

Based on the format and complexity of the raw data as the input of machine learning algorithms, conventional methods were restricted and required significant domain expertise. The experts needed to design a feature extractor for the transformation of raw data that could be the pixel values, where having images as the inputs, into an appropriate feature vector. These feature vectors or more generally, any internal representation, sets the stage for the learning procedure that enables the detection or classification in the input data. To make machine learning methods easier for more people to use, representation learning was introduced that is composed of a set of techniques allowing the machine to be fed with the raw data (Bengio et al., 2013). Instead of having the expertise for transforming raw data

into an internal representation, these learning methods automatically discover the suitable representation for any detection or classification tasks.

Deep learning methods are among representation learning methods, and they have multiple levels of representation. Each level of observation is obtained by a combination of non-linear modules that are not complex. Then the representations at each level (starting from the input data) are transformed into representation at a higher level. By combining or stacking these transformations, it becomes possible to learn very complicated functions. In the area of image classification, since the input data is given in a format of pixel arrays, higher layers of representation intensify parts of the input that is vital for discrimination and reduce the variation. Generally, the first layer looks for the absence or presence of the edges at particular locations of an image. The the second layer usually detect for any known pattern or theme, and in the third layer, the patterns are grouped to detect familiar objects. It has to be emphasized that the layers are not designed by humans and only learned from the data. In the domain of image recognition, Krizhevsky et al. (2012) made the error rate of object recognition in half while using convolutional neural networks. The research presented by Farabet et al. (2013), Tompson et al. (2014), and Szegedy et al. (2015) were among other breakthroughs of convolutional neural nets.

Here, one of the most well-known techniques, Convolutional Neural Network (CNN), in deep learning that fits computer-vision is used for learning the features as well as the classification of the images. CNNs are a unique class of neural networks in the biological processes as stated by Matsugu et al. (2003). In the area of image classification, and more generally, computer vision, CNNs have shown promising results. One of the significant differences between CNN and a feed-forward neural network is the way that the inputs are fed to the algorithm. The input of the CNN is in a format of a two-dimensional array (or a matrix), therefore using the convolution operation becomes practical in a sense that the number of free parameters reduces significantly (Simard et al., 2003). Convolutional layers apply convolution operations to the input, passing the result to the next layer. By applying a convolution operation with a filter to the input data, adding the bias term and applying the non-linear function, the result is passed to the next layer (LeCun et al., 2015b).

2.2.1 Research Gap

Literature reveals that there is a high volume of research, with considerable depth, related to packing small rigid items in cases (Table 2.1). On the other hand, there is sparse research which addresses explicitly how flexible or moldable items change the problem formulation or solution. The motivating example for the presented work is that practical packages – such as bags containing food – may have non-utilized space that can be reduced by folding. These folds are easy to execute by workers on an assembly or production line. It is therefore pertinent to study the effect of folding on the packing position, orientation, and sequence of bags. The presented work addresses this gap in the following ways. The optimization generalizes the problem by considering three possible folding configurations for each bag. A set of associated decision variables is identified to enable the search for an optimal option. To the best of our knowledge, this dissertation is the first research that presents optimization for flexible packages which is related to a unique and challenging problem in packing literature.

In the case study that is studied here, the meal bags (small items) have some un-utilized space that could be shrunk by folding. These folds are easy to be implemented once the best scenario for folding is decided. Therefore, in this research three possible folding configurations are introduced for each bag with a set of decision variables associated with them enabling us to find the optimal option. Other than this contribution, since there is the possibility of variation in assembly stations, an inspection module is developed to find out any defects in the final packages. To the best of our knowledge, no such a model in the literature combines optimization and machine learning to implement a decision successfully. To summarize, this dissertation makes the following scientific or technological contributions:

- Optimization for flexible packaging addresses a unique and challenging problem in packaging.
- Inspection system uses machine learning to classify packaging errors.
- Accountability model identifies the teams or personnel who may be retrained to enhance the productivity and reliability of the packaging process.

Table 2.1: Summary of Literature Survey

Publication	Packing Parameters						Usability	
	Item Shape					Orientation	Automatic Update	
	Rigid	Flexible	Rectangular	Circular	Irregular		Visualization	Training with SOP
Bischoff and Marriott (1990)	✓		✓					
Li and Cheng (1990)	✓		✓			✓		
Scheithauer (1991)	✓		✓			✓		
Li and Cheng (1992)	✓		✓			✓		
Corcoran III and Wainwright (1992)	✓		✓					
Mukhacheva and Shehtman (1997)	✓		✓			✓		
Miyazawa and Wakabayashi (1997)	✓		✓			✓		
Lai et al. (1998)	✓		✓					
Bortfeldt and Gehring (1999)	✓		✓			✓		
Faina (2000)	✓		✓					
Li et al. (2003)	✓		✓					
Stoyan et al. (2003)	✓			✓				
Yeung and Tang (2005)	✓		✓			✓		
Jansen and Solis-Oba (2006)	✓		✓			✓	✓	
Bansal et al. (2007)	✓		✓			✓		
Bortfeldt and Mack (2007)	✓		✓			✓		
Egeblad et al. (2007)	✓			✓		✓		
Miyazawa and Wakabayashi (2007)	✓		✓			✓		
Birgin and Sobral (2008)	✓			✓			✓	
Egeblad et al. (2009)	✓				✓	✓	✓	
Fujiyoshi et al. (2009)	✓		✓			✓		
Miyazawa and Wakabayashi (2009)	✓		✓					
Junqueira et al. (2012)	✓		✓			✓	✓	
De Queiroz et al. (2012)	✓		✓			✓		
Allen et al. (2011)	✓		✓			✓	✓	
He and Huang (2011)	✓		✓			✓	✓	
He et al. (2012)	✓		✓			✓		
Hasni and Sabri (2013)	✓		✓					
Araya and Riff (2014)	✓		✓			✓		
Li and Zhang (2015)	✓		✓			✓		
Huang et al. (2016)	✓		✓			✓	✓	
Mostaghimi Ghomi et al. (2017)	✓		✓			✓	✓	
This Study	✓	✓	✓		✓	✓	✓	✓

Chapter 3

Methodology

This chapter is composed of two main parts. In the first part, the methodology of the optimization model is described in detail. This part intends to present a model that enables the assembler to have a better placement strategy when packing the bags inside the cases. The details of the mathematical model, are presented in this chapter, and the results are shown in Chapter 4. The second part of this chapter covers the model used for the visual inspection of the packed cases. This part serves as the validation for the optimization solutions. Having packing optimization and inspection in an integrated framework leads to having an accountable model which automates the steps from packing bags inside the case to the final quality inspection of the cases before releasing them for palletization. Additionally, depending on the location of the defect detected by the inspection, focused training of the workers will reduce the odds of having common flaws in the future. The results of the inspection are also presented in Chapter 4.

3.1 Optimization

3.1.1 Problem Statement

Let us assume that in a production line, the produced and packaged goods are needed to be placed in a case/carton. The cartons have to be sealed and moved to the palletization process in a way that no bulges could be observed. Since these cartons have to be stacked together

for palletization and shipment, the bulges can cause lack of stability. More importantly, the content of these boxes is more prone to defection in case of bulge existence. In this dissertation, it is assumed that the number of bags (with flexible packaging covers) to be placed in every case is fixed and known in advance. Since the length and width of the pallets are known and fixed, it is not intended to change these dimensions of the cases but to minimize the height of them. Lack of having a systematic approach in placing items inside the cases lead to some problems that are listed below:

1. The bags are usually filled with various content items, and they are not vacuum packed; therefore they have some unused space.

2. The bags do not have a fixed configuration and dimensions, and there is a possibility that the contents are jammed in one side of a bag.

3. To avoid the occurrence of the lumps in the cases, some companies use multiple pressing stations at the end of the line (roller from the sides and vertical pressing machine). The harm of this step is that some of the contents might be crushed and in cases where the content is food, the content will not be edible anymore. Having not consumable items could be translated to waste in the system.

One of the purposes of this study is to present a systematic way to optimize the packing process with three significant impacts. First of all, it is intended to have a clear approach on how to place the flexible bags inside a case where each bag has some empty space, and simple foldings can cause the bag more rigid and reduce the unused space. Secondly, we want to make sure that the available space is used efficiently and no more mechanical pressings are needed in the packing process. Lack of a standard way to place bags in a case leads to having bulged boxes. Thus one of the essential outputs is to provide the company with a methodology that can be used whenever they decided to change the contents of bags.

Last but not least, it is desired to improve the workers' well-being in the system. Interactive training is the output of the model which could be referred to in the times that workers make some mistake. The proposed method is accountable since it establishes a baseline that can identify which step of the packing leads to the bulge in the final packed case.

3.2 Mathematical Formulation

Operations research studies involve the construction of a mathematical model, which is a collection of logical and numerical relationships that represents the characteristics of the problem under consideration. These models describe the significant relationships between variables. Additionally, they include an objective function for evaluation of the alternative solutions and a set of constraints for restricting solutions to get feasible values.

In this section, it is intended to find the best placement strategy for putting the small items (bags) into the large items (cases). The mathematical model uses the (x, y, z) coordinates of the bags relative to the coordinates of the case to produce the optimal positions. Additionally, since the orientation makes a difference for the packing pattern, a set of variables are used to take that into account. It is necessary to make sure that all the bags lie entirely within the cases and small items do not overlap with each other. The notation is similar to Wu et al. (2010) and are presented below:

3.2.1 Sets

$\boldsymbol{A}$	set of bags
$\boldsymbol{K}$	set of possible foldings

3.2.2 Parameters

$\boldsymbol{(l_{ik}, w_{ik}, h_{ik})}$	$(i,j) \in A,\ k \in K$	length, width and height of bag i in scenario k respectively
$\boldsymbol{L, W, H}$		The length, width and of the case/container respectively
$\boldsymbol{M}$		A big number

3.2.3 Continuous Variables

$\boldsymbol{(x_i, y_i, z_i)}$	$i \in A$	left-bottom-behind corner coordinates of bag i
$\boldsymbol{\tilde{H}}$		The height of the case/container

3.2.4 Binary Variables

$\boldsymbol{N_{ik}}$	$i \in A, k \in K$	folding scenario k for bag i
$\boldsymbol{X_{li}}$	$i \in A$	1 if the length direction of bag i is parallel to the case's X axis; 0 otherwise
$\boldsymbol{Z_{li}}$	$i \in A$	1 if the length direction of bag i is parallel to the case's Z axis; 0 otherwise
$\boldsymbol{Y_{wi}}$	$i \in A$	1 if the width direction of bag i is parallel to the case's Y axi; 0 otherwise
$\boldsymbol{Z_{hi}}$	$i \in A$	1 if the height direction of bag i is parallel to the case's Z axis; 0 otherwise
$\boldsymbol{a_{ij}}$	$(i,j) \in A$	relative positioning of bag i to bag j. 1 if bag i is is in front of bag j; 0 otherwise
$\boldsymbol{b_{ij}}$	$(i,j) \in A$	relative positioning of bag i to bag j. 1 if bag i to the right of bag j; 0 otherwise
$\boldsymbol{c_{ij}}$	$(i,j) \in A$	relative positioning of bag i to bag j. 1 if bag i is is on top of bag j; 0 otherwise

3.2.5 Objective Function

$$\text{Minimize} \quad \tilde{H} \tag{3.1}$$

3.2.6 Constraints

$$x_i + \sum_{k \in K} X_{li} l_{ik} N_{ik} + \sum_{k \in K} w_{ik} N_{ik} (Z_{li} - Y_{wi} + Z_{hi}) + \sum_{k \in K} h_{ik} N_{ik} (1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi}) \leq x_j + M(1 - a_{ij}) \quad \forall \{i,j\} \in A,\ i \neq j \tag{3.2}$$

$$y_i + \sum_{k \in K} Y_{wi} w_{ik} N_{ik} + \sum_{k \in K} l_{ik} N_{ik} (1 - X_{li} - Z_{li}) + \sum_{k \in K} h_{ik} N_{ik} (X_{li} + Z_{li} - Y_{wi}) \leq y_j + M(1 - b_{ij}) \quad \forall \{i,j\} \in A,\ i \neq j \tag{3.3}$$

$$z_i + \sum_{k \in K} Z_{hi} h_{ik} N_{ik} + \sum_{k \in K} w_{ik} N_{ik} (1 - Z_{li} - Z_{hi}) + \sum_{k \in K} l_{ik} N_{ik} (Z_{li}) \leq z_j + M(1 - c_{ij}) \quad \forall \{i,j\} \in A,\ i \neq j \tag{3.4}$$

$$x_i + \sum_{k \in K} l_{ik} N_{ik} X_{li} + \sum_{k \in K} w_{ik} N_{ik} (Z_{li} - Y_{wi} + Z_{hi}) + \sum_{k \in K} h_{ik} N_{ik} (1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi}) \leq L \qquad \forall i \in A \tag{3.5}$$

$$y_i + \sum_{k \in K} w_{ik} N_{ik} Y_{wi} + \sum_{k \in K} l_{ik} N_{ik} (1 - X_{li} - Z_{li}) + \sum_{k \in K} h_{ik} N_{ik} (X_{li} + Z_{li} - Y_{wi}) \leq W \qquad \forall i \in A \tag{3.6}$$

$$z_i + \sum_{k \in K} h_{ik} N_{ik} Z_{hi} + \sum_{k \in K} w_{ik} N_{ik} (1 - Z_{li} - Z_{hi}) + \sum_{k \in K} l_{ik} N_{ik} (Z_{li}) \leq \tilde{H} \qquad \forall i \in A \tag{3.7}$$

$$a_{ij} + a_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} \geq 1 \qquad \forall \{i, j\} \in A,\ i \neq j \tag{3.8}$$

$$X_{li} + Z_{li} \leq 1 \qquad \forall i \in A \tag{3.9}$$

$$Z_{li} + Z_{hi} \leq 1 \qquad \forall i \in A \tag{3.10}$$

$$Z_{li} - Y wi + Z_{hi} \geq 0 \qquad \forall i \in A \tag{3.11}$$

$$1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi} \leq 1 \qquad \forall i \in A \tag{3.12}$$

$$1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi} \geq 0 \qquad \forall i \in A \tag{3.13}$$

$$X_{li} + Z_{li} - Y_{wi} \leq 1 \qquad \forall i \in A \tag{3.14}$$

$$X_{li} + Z_{li} - Y_{wi} \geq 0 \qquad \forall i \in A \tag{3.15}$$

To clarify the use of binary variables, Figure 3.1 illustrates a case that has two bags inside of it. Since bag i is rotated the binary variables that define this orientation are $X_{li} = Y_{wi} = Z_{hi} = 0, Z_{li} = 1$, and for bag j which is not rotated the binary variable assignments are $X_{lj} = Y_{wj} = Z_{hj} = 1, Z_{lj} = 0$. As for the relative placements, since bag i is in front, right hand side of bag j the variables are $a_{ij} = b_{ij} = 1$ and $c_{ij} = 0$.

The objective of this model is to minimize the height of each case (Equation 3.1) since it will not be feasible to use the same pallets anymore if we change the case's length and width. Equations 3.2 – 3.4 are to make sure that the items do not overlap with each other. Equations 3.5 – 3.7 are to make sure that the items are within the case dimensions. Equation 3.8 is to

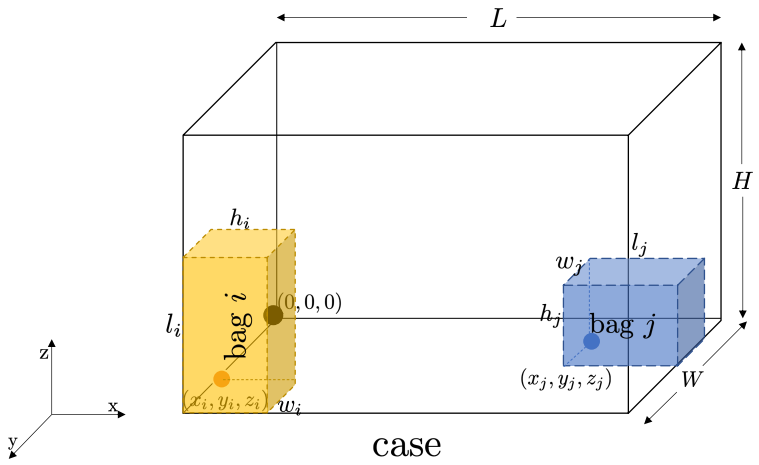


Figure 3.1: Representation of Relative Placements and Variable Explanation

make sure that only one of the relative positioning variables is true at a time. Equations 3.9 – 3.15 are to make sure that the binary variables are controlled correctly, and no more than a single orientation variables is equal to one. As it can be seen in Equations 3.2 – 3.7, we have non-linear terms such as $N_{ik}X_{li}$. Since these terms are the multiplication of two binary variables, with further endeavor, it is possible to make the model linearized, and the model is presented in the next section.

3.3 Linearized Model

In order to make the model linear, the initial step is to substitute the nonlinear terms with new ones. The new variables that have to be added to the model are listed below:

3.3.1 Binary Variables

$\boldsymbol{P_{ik}}$ $i \in A, k \in K$ to be substituted with $X_{li}N_{ik}$

$\boldsymbol{Q_{ik}}$ $i \in A, k \in K$ to be substituted with $Z_{li}N_{ik}$

$\boldsymbol{R_{ik}}$ $i \in A, k \in K$ to be substituted with $Y_{wi}N_{ik}$

$\boldsymbol{S_{ik}}$ $i \in A, k \in K$ to be substituted with $Z_{hi}N_{ik}$

To make sure that the variables are controlled correctly, we have to add three extra constraints to the model (in addition to the non-negativity constraint) presented in the previous section. Thus, similar to the previous model the objective of this model is to minimize the height of each case (Equation 3.16). Equations 3.17 – 3.19 are to make sure that the items do not overlap with each other. Equations 3.20 – 3.22 are to make sure that the items are within the case dimensions. Equation 3.23 is to make sure that only one of the relative positioning variables is right at a time. Equations 3.24 – 3.30 are to make sure that the binary variables are controlled correctly, and no more than a single orientation variable is equal to one. Finally, Equations 3.31 – 3.42 are added to the previous model to make sure that the new set of binary variable substitution is appropriately handled.

3.3.2 Objective Function

$$\text{Minimize} \quad \tilde{H} \tag{3.16}$$

3.3.3 Constraints

$$x_i + \sum_{k\in K} l_{ik}P_{ik} + \sum_{k\in K} w_{ik}(Q_{ik} - R_{ik} + S_{ik}) + \sum_{k\in K} h_{ik}(N_{ik} - P_{ik} - Q_{ik} + R_{ik} - S_{ik}) \leq x_j + M(1 - a_{ij}) \quad \forall\{i,j\} \in A,\ i \neq j \tag{3.17}$$

$$y_i + \sum_{k\in K} w_{ik}R_{ik} + \sum_{k\in K} l_{ik}(N_{ik} - P_{ik} - Q_{ik}) + \sum_{k\in K} h_{ik}(P_{ik} + Q_{ik} - R_{ik}) \leq y_j + M(1 - b_{ij}) \quad \forall\{i,j\} \in A,\ i \neq j \tag{3.18}$$

$$z_i + \sum_{k\in K} h_{ik}S_{ik} + \sum_{k\in K} w_{ik}(N_{ik} - Q_{ik} - S_{ik}) + \sum_{k\in K} l_{ik}Q_{ik} \leq z_j + M(1 - c_{ij}) \quad \forall\{i,j\} \in A,\ i \neq j \tag{3.19}$$

$$x_i + \sum_{k\in K} l_{ik}P_{ik} + \sum_{k\in K} w_{ik}(Q_{ik} - R_{ik} + S_{ik}) + \sum_{k\in K} h_{ik}(N_{ik} - P_{ik} - Q_{ik} + R_{ik} - S_{ik}) \leq L \quad \forall i \in A \tag{3.20}$$

$$y_i + \sum_{k\in K} w_{ik}R_{ik} + \sum_{k\in K} l_{ik}(N_{ik} - P_{ik} - Q_{ik}) + \sum_{k\in K} h_{ik}(P_{ik} + Q_{ik} - R_{ik}) \leq W \quad \forall i \in A \tag{3.21}$$

$$z_i + \sum_{k\in K} h_{ik}S_{ik} + \sum_{k\in K} w_{ik}(N_{ik} - Q_{ik} - S_{ik}) + \sum_{k\in K} l_{ik}Q_{ik} \leq \tilde{H} \quad \forall i \in A \tag{3.22}$$

$$a_{ij} + a_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} \geq 1 \qquad \forall \{i,j\} \in A,\ i \neq j \tag{3.23}$$

$$X_{li} + Z_{li} \leq 1 \qquad \forall i \in A \tag{3.24}$$

$$Z_{li} + Z_{hi} \leq 1 \qquad \forall i \in A \tag{3.25}$$

$$Z_{li} - Y_{wi} + Z_{hi} \geq 0 \qquad \forall i \in A \tag{3.26}$$

$$1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi} \leq 1 \qquad \forall i \in A \tag{3.27}$$

$$1 - X_{li} - Z_{li} + Y_{wi} - Z_{hi} \geq 0 \qquad \forall i \in A \tag{3.28}$$

$$X_{li} + Z_{li} - Y_{wi} \leq 1 \qquad \forall i \in A \tag{3.29}$$

$$X_{li} + Z_{li} - Y_{wi} \geq 0 \qquad \forall i \in A \tag{3.30}$$

$$P_{ik} \leq X_{li} \qquad \forall i \in A\ \forall k \in K \tag{3.31}$$

$$P_{ik} \leq N_{ik} \qquad \forall i \in A\ \forall k \in K \tag{3.32}$$

$$P_{ik} \geq X_{li} + N_{ik} - 1 \qquad \forall i \in A\ \forall k \in K \tag{3.33}$$

$$Q_{ik} \leq Z_{li} \qquad \forall i \in A\ \forall k \in K \tag{3.34}$$

$$Q_{ik} \leq N_{ik} \qquad \forall i \in A\ \forall k \in K \tag{3.35}$$

$$Q_{ik} \geq Z_{li} + N_{ik} - 1 \qquad \forall i \in A\ \forall k \in K \tag{3.36}$$

$$R_{ik} \leq Y_{wi} \qquad \forall i \in A\ \forall k \in K \tag{3.37}$$

$$R_{ik} \leq N_{ik} \qquad \forall i \in A\ \forall k \in K \tag{3.38}$$

$$R_{ik} \geq Y_{wi} + N_{ik} - 1 \qquad \forall i \in A\ \forall k \in K \tag{3.39}$$

$$S_{ik} \leq Z_{hi} \qquad \forall i \in A\ \forall k \in K \tag{3.40}$$

$$S_{ik} \leq N_{ik} \qquad \forall i \in A\ \forall k \in K \tag{3.41}$$

$$S_{ik} \geq Z_{hi} + N_{ik} - 1 \qquad \forall i \in A\ \forall k \in K \tag{3.42}$$

$$x_i, y_i, z_i, \tilde{H}, N_{ik}, X_{li}, Z_{li}, Y_{wi}, Z_{hi},$$
$$a_{ij}, b_{ij}, c_{ij}, P_{ik}, Q_{ik}, R_{ik}, S_{ik} \geq 0 \qquad \forall \{i,j\} \in A,\ \forall k \in K \tag{3.43}$$

The objective of this model is to minimize the height of each case (Equation 3.16). Equations 3.17 – 3.19 are to make sure that the items do not overlap with each other.

Equations 3.20 – 3.22 are to make sure that the items are within the case dimensions. Equation 3.23 is to make sure that only one of the relative positioning variables is true at a time. Equations 3.24 – 3.30 are to make sure that the binary variables are controlled correctly, and no more than a single orientation variable is equal to one. To control the non-linear terms in Equations 3.2 – 3.7, for each of the new variables, three set of constraints are added to the model. Equations 3.31 – 3.33 are corresponding to P_{ik} that is replaced with the multiplication of X_{li} and N_{ik}. Similarly, Equations 3.34 – 3.36 are corresponding to Q_{ik} that is replaced with the multiplication of Z_{li} and N_{ik}. and Equations 3.37 – 3.39 are corresponding to R_{ik} that is replaced with the multiplication of Y_{wi} and N_{ik}. Lastly, Equations 3.40 – 3.42 are corresponding to S_{ik} that is replaced with the multiplication of Z_{hi} and N_{ik}.

The presented model is solved with Gurobi solver (Optimization, 2017), and the results are rendered in an animated format for each station in charge of packing. Additionally, a user interface is developed that simplifies the usage of the model for the end users and outputs the results in a format of SOP in an automated way. Having such a model is useful in the companies that pack products with a short life cycle since it can adapt to the changes quickly and efficiently.

3.4 Inspection

3.4.1 Problem Statement

In the previous section (3.1.1), an operations research framework was presented for the packing of cases encompassing items with flexible packaging. The approach is involved with different scenarios for folding the bags. It is true that the personnel in charge of packing bags inside cases are now faced with an extra step which is folding the bags, therefore the risk of making mistakes is increasing. To have a measurement system, an automatic inspection module was developed with the goal of identifying any final case with a bulge.

Automated visual recognition has been an intimidating task for computer and machine vision. The lack of large volumes of labeled data historically made it difficult for machine

learning algorithms to perform recognition tasks accurately. The emergence of the ImageNet database (Krizhevsky et al., 2012) boosted this area of research. The ImageNet project creates and maintains an extensive database of images that would have hand-annotated labels through crowdsourcing and serves as a benchmark dataset for most computer vision problems. As of now, ImageNet hosts links to almost ten million hand-annotated images. The availability of data has led to extensive research on computer vision algorithms to classify objects in the pictures in the ImageNet database automatically. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) provided a further boost to this research, which forms the basis of the idea being tested for the case study presented in this dissertation. Up to 2012, hand-engineered features were used to facilitate object classification. In 2012, researchers at the University of Toronto utilized a Convolutional Neural Network (CNN) to achieve a revolutionary error rate of 15.3%. Since then, every winner of the ILSVRC each year has utilized a CNN with different architectures to improve visual recognition accuracy. The most significant accomplishment from the perspective of the presented case study is that the winner of the 2015 challenge achieved an error rate which was lower than the human error rate for a visual recognition challenge. This leads us to believe that, with sufficient data, the inspection system has the potential to perform sophisticated classification tasks at least as well as human inspectors. The general explanation about how CNNs perform is presented in the next section.

3.5 Convolutional Neural Networks

3.5.1 Objective

For analyzing visual imagery, CNN is vastly used which is in a class of deep, feed-forward – where there are no loops in the network and information is always fed forward – artificial neural networks. One of the main applications of CNN is "Image Classification" where an image is inputted to the system, and a class or label is expected as an output of the algorithm. For human beings, recognition is a natural task which can happen very quickly without spending too much time analyzing or thinking. Moreover, humans can assign a label

to the objects with generalizing prior knowledge and adapt to different environments. On the other hand, machines do not have the same capability, but the same idea is the primary driver to be implemented in the area of image classification.

3.5.2 Background and Formulation

Similar to neural networks, CNNs are also related to neuroscience. Biologically speaking, CNNs look similar to the visual cortex where small regions of cells are sensitive to specific regions of the visual field. One of the oldest types of artificial neuron is called **perceptron** which was developed by Rosenblatt (1958). A perceptron's task is to take a set of binary inputs such as $x_1, x_2, ..., x_n$ and produce a single binary output in return as it is depicted in Figure 3.2.

The mechanism of the perceptron is that each input should have a **weight**, which is a real value to express the importance of each input $(w_1, w_2, ..., w_n)$. The output takes the value of zero or one based on the results of the weighted sum of the inputs. To be more precise the outcomes of the output are:

$$output = \begin{cases} 0, & \text{if } \sum_j w_j x_j \leq \text{threshold.} \\ 1, & \text{if } \sum_j w_j x_j > \text{threshold.} \end{cases} \tag{3.44}$$

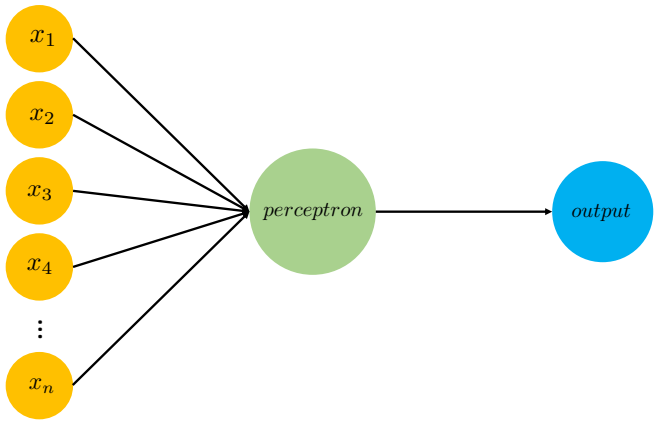


Figure 3.2: Overview of a Perceptron Neuron

Since perceptron is very simplistic, and the results are highly dependent on the weights and threshold, it is not practical to be used in the domain of complex problems. But it has been proved that a network of perceptrons (stacked layers of perceptrons presented in Figure 3.3) perform better in real-world problems. Each layer takes care of a decision-making problem, and by adding layers, more complex and abstract decisions can be made.

To make the notation used in Equation 3.44 easier, instead of using weighted sum we will use a dot product operation for the arrays. Also, we set b equal to (-threshold) and will refer to it as the bias term. Therefore, the equations are rewritten as:

$$output = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0. \\ 1, & \text{if } w \cdot x + b > 0. \end{cases} \tag{3.45}$$

It needs to be mentioned that the weights and biases of the network are usually initialized and then with a learning algorithm, they will both be tuned in the network of artificial neurons or perceptrons. It is expected for the learning algorithm to reflect a small change in the output if a small change is happening in the bias or weight value in the network. In other

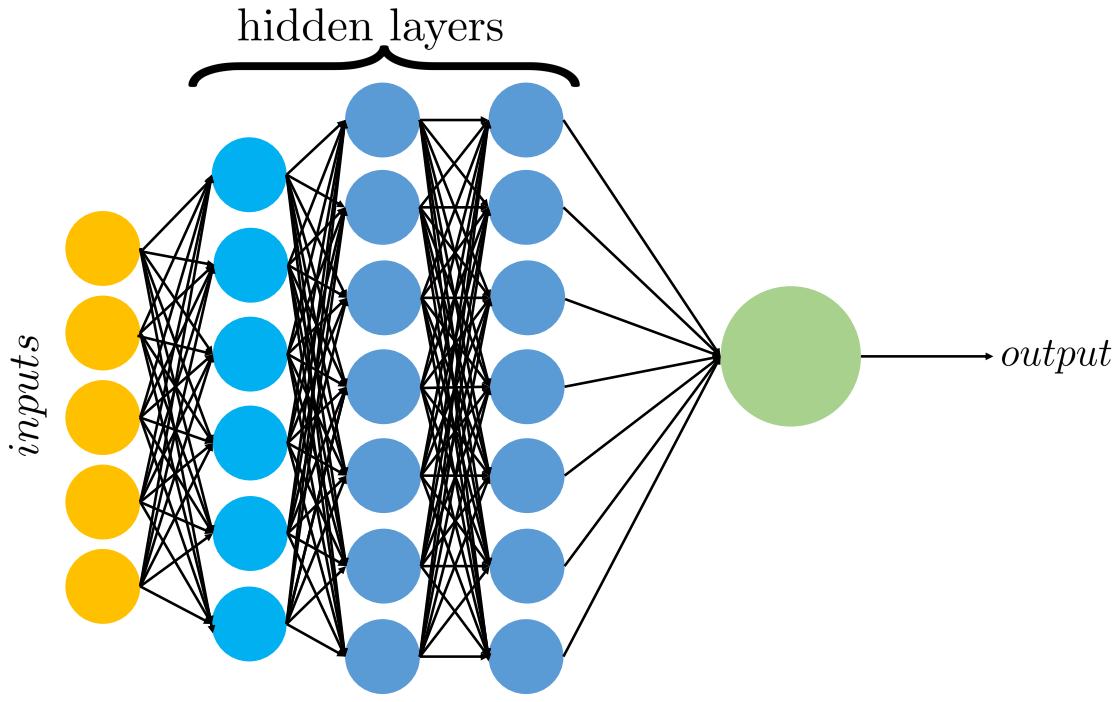


Figure 3.3: Overview of a More Complex Perceptron Neuron

words, if the output of the problem is a classification problem, then we expect that with minimal change in the weights, the output still remains in the same class. With having a network that is not influenced much by small changes, we would be able to train and modify the network so as to get the behavior that is desired. It worth mentioning that, sometimes, depending on the complexity of the problem the training would be extremely hard.

A network of perceptrons lacks the quality of the desired networks where the outcome has to be slightly sensitive with respect to the bias or weights. As a matter of fact, even a minute change of any perceptron can cause in a flip for the output of the network. This problem is mainly related to the way a perceptron performs; Therefore, in the modern neural networks, it is more common to use another neuron model which is called **sigmoid**. A sigmoid neuron is similar to the perceptron in terms of the components (see Figure 3.2). Although there is one significant difference between the outputs of these neurons. In perceptron, the output is a binary value, whereas in sigmoid neurons that the outputs have any probability value (any number between 0 and 1). To produce the output in each sigmoid neuron, instead of following the Equation 3.45, a sigmoid function must be applied to the $(w \cdot x + b)$ term, where the sigmoid function is written as:

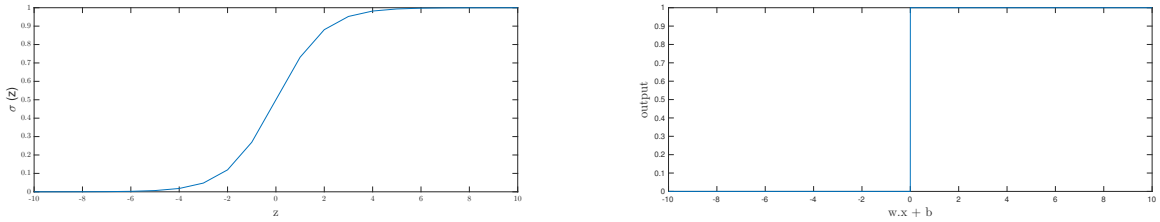
$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.46}$$

and the output of each sigmoid neuron with $x_1, x_2, ..., x_n$ inputs and $(w_1, w_2, ..., w_n)$ weights can be written as Equation 3.47 in its expanded form and Equation 3.48 using vectorized notation:

$$\sigma(z) = \frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{3.47}$$

$$\sigma(z) = \frac{1}{1 + exp(-w \cdot x - b)} \tag{3.48}$$

It could easily be seen that the behavior of sigmoid and perceptron is similar to each other by looking at the plots of sigmoid and stepwise functions presented in Figure 3.4. Lets assume $w \cdot x$ is a large positive number in a perceptron, then in sigmoid $\sigma(z) \approx 1$. In



(a) Sigmoid **(b)** Stepwise

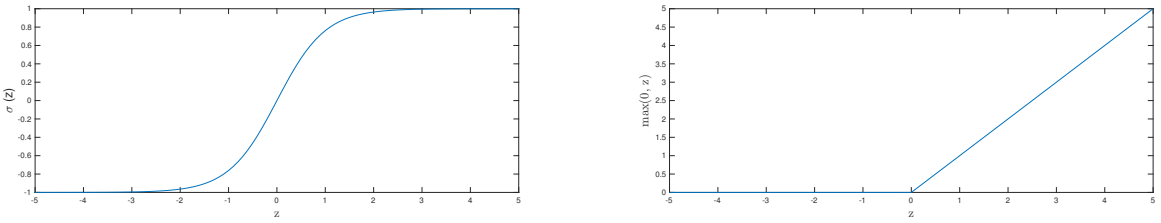
Figure 3.4: Comparison of Sigmoid and Perceptron Neurons

other words, a very large output resulted from perceptron corresponds to probability of 1 in sigmoid. In a same fashion, when $w \cdot x$ is a large negative number in perceptron, then in sigmoid $\sigma(z) \approx 0$.

Based on the characteristics of each problem, a different activation function is used. Another one, *tanh*, is commonly used which has the ability to center the data, rather than sigmoid. Generally speaking, sigmoid is preferred over *tanh* only in binary classification problems. A representation of the *tanh* is shown in Figure 3.5a.

Both of these activations have a downside which is a weakness of performance when values of z are very large. To cope with this issue, ReLU (rectified linear unit) activation function was introduced (see Figure 3.5b).

In some cases having the derivative of the activation function as zero might cause problems. Therefore, one last well-established activation function is called Leaky ReLU, where the slope of the function for negative z is close to zero but not accurately 0. A graphical representation of this function is presented in Figure 3.6.



(a) tanh **(b)** ReLU

Figure 3.5: tanh and ReLU Activation Functions

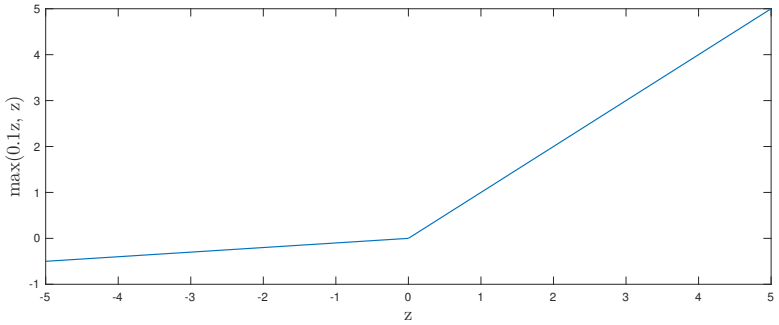


Figure 3.6: Leaky ReLU Activation Function

Next step is to define a cost function which enables us to train the data. Suppose that we want to set a loss function based on a single observation, then one candidate can be written as the following ($\hat{y}, y$ refer to the predicted label and actual label respectively):

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{3.49}$$

The proposed loss function is not very good because it is not convex (finding global optima is not possible with well-known algorithms such as gradient descent). The cost or loss function is a metric that says how well the algorithm is performing on the whole data set and at the same time, we should be capable of optimizing it (within a reasonable timeframe). For instance, in binary classification, a reasonable loss function can be defined as Equation 3.50. We call it reasonable because we expect $\hat{y}$ to have a large value when $y = 1$, on the other hand, we want $\hat{y}$ to have a tiny value when $y = 0$.

$$L = -\big[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\big] \tag{3.50}$$

It must be noted that $\hat{y}$ refers to the predicted label and y refers to the actual label of a single observation in the training sample. Now if we want to extend the loss function to all observations in the training set (where $\hat{y}^{(i)}$ and $y^{(i)}$ refers to the predicted label and the actual label of the i-th observation, respectively), the loss function is defined in Equation 3.51.

$$J(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right] \tag{3.51}$$

Now that the loss function is defined, we have to minimize $J(W,b)$ and find the optimal values of W^* and b^*. Since $J(W,b)$ is a convex function, we are sure that W^* and b^* are the global optimal points. One of the popular algorithms for the optimization is **Gradient Decent** which starts from an initial point (solution) and moves toward the global optima. This process can be mathematically expressed as Equation 3.52 for the weights and bias terms. In these equations, α represents the learning rate which is among the user inputs (it can be tuned as a hyperparameter).

$$\begin{aligned} W &:= W - \alpha\frac{\partial J(W,b)}{\partial W} \\ b &:= b - \alpha\frac{\partial J(W,b)}{\partial b} \end{aligned} \tag{3.52}$$

There are three different classes of gradient descent with respect to the number of training patterns that are used for error calculation. Each of the classes is explained briefly in the following:

1. Batch gradient descent

   Like the original gradient descent optimization technique, all the errors are calculated for each example in the training sample, but the model is only updated after all of these training examples have been evaluated. In other words, batch gradient descent updates the model at the end of each training epoch, where epoch is the term referring to a single cycle through the entire training dataset.

2. Mini-batch gradient descent

   In this method, the training dataset is split into small batches, and the errors are calculated for each example in the training set. Unlike the previous method, the model coefficients are updated after calculations are done for each mini-batch.

3. Stochastic gradient descent

   In this variant of the gradient descent method, the model coefficients are updated after error calculation for each example in the training sample is done. Although the model coefficients are more frequently updated, it is noisier with more oscillations around the optimal which can lead to variation in loss and accuracy.

4. Gradient descent with momentum

   The basis of this approach is rooted from the exponentially moving average method which works well for the estimation of noisy data. Given a set of data points over time (t), the exponentially weighted moving average of data at time t can be written as Equation 3.53. V_t can be specified as approximately averaging over $\frac{1}{1-\beta}$ time units of data (or iterations). The formula could be interpreted in a way that larger values of β adapt to the data more slowly, but it is smoother and more shifted to the last data points $(0 \leq \beta \leq 1)$.

$$V_t = \beta V_{t-1} + (1 - \beta)\theta_t \tag{3.53}$$

   To address the issue as mentioned above (where the initial values of the estimate are minimal), bias correction is done to make the computations more accurate. In cases of having large β values, the initial estimates will be very close to V_0 (which is initialized as 0). As a solution, instead of taking V_t, we take $\frac{V_t}{1-\beta^t}$. When $t \to \infty$ then $(1-\beta^t) \to 1$ which let us make sure that the bias correction is only active in the initial phases.

   The main idea of the momentum in the gradient descent algorithm is to get the exponential moving average of the gradients, and then use this information to update the weights and bias terms in the network. Figure 3.7 shows a possible path to the optimal solution where leaning is desired to be faster in the horizontal direction and slower in the vertical direction, in other words, the oscillation in the vertical direction has to be reduced.

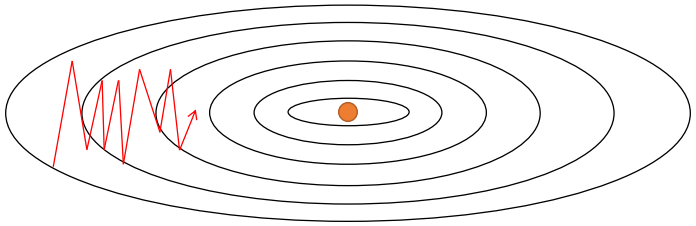


Figure 3.7: Illustration of an Oscillatory Solution Path Using Gradient Descent Optimizer

To add momentum to the gradient descent, in each iteration, dW and db are calculated and then they are exponentially averaged as shown in Equations 3.54 and 3.55 respectively. α and β are hyperparameters which could be tuned but in practice, $\beta = 0.9$ is commonly used and proved to be a robust value (Hinton et al., 2012).

$$V_{dW} = \beta V_{dW} + (1 - \beta) dW \tag{3.54}$$

$$V_{db} = \beta V_{db} + (1 - \beta) db \tag{3.55}$$

$$W := W - \alpha V_{dW} \tag{3.56}$$

$$b := b - \alpha V_{db} \tag{3.57}$$

In the presented formulation, there is no essential need to have a bias correction since with setting the value of $\beta = 0.9$ after about ten iterations the system warms up, and the initialization will be corrected.

As the formulation shows, in SGD, the exact derivatives of the loss functions are not computed, but instead, they are estimated on a small batch. Therefore, it is possible that the algorithm does not stay in the optimal direction because of the derivatives' noisiness. However, one way of having a better derivative estimation is the exponentially weighted averages. This is one of the main reasons that SGD with momentum might perform better than the traditional method. The other reason for the popularity of this method is that it helps the gradients to move in the right direction

toward the optimal solution whereas SGD might oscillate around a locally optimal point.

5. Root Mean Squared Prop (RMSprop)

   In this technique, it is desired to damp out the oscillation in the path of reaching the optimal point. As Figure 3.7 shows, for simplicity let us assume that the vertical direction corresponds to bias terms and the horizontal is related to weights. Therefore we want the algorithm to speed up in the horizontal direction and slows down in the vertical direction. To reach this goal, the following formulation is presented where S is used instead of V in the exponentially weighted moving average method. At iteration t the formulation can be written as:

$$S_{dW} = \beta S_{dW} + (1-\beta) dW^2 \tag{3.58}$$

$$S_{db} = \beta S_{db} + (1-\beta) db^2 \tag{3.59}$$

$$W := W - \alpha \frac{dW}{\sqrt{S_{dW}}} \tag{3.60}$$

$$b := b - \alpha \frac{db}{\sqrt{S_{db}}} \tag{3.61}$$

   The intuition from Equation 3.58 is that since dW are small, then when updating W with Equation 3.60 it will get updated faster. On the other hand, based on Equation 3.59, since db values are large, the process of update in b becomes slower as desired. It has to be noted that in real cases, W and b are both really high dimensional, but the separation of the parameters helps smooth the oscillation and make the algorithm faster (Tieleman and Hinton, 2012).

   To ensure the numerical stability in the Equations 3.60 and 3.61, a very small term is added to the denominator (ϵ) where the value is set as 10^{-8} (see Equations 3.62, 3.63).

$$W := W - \alpha \frac{dW}{\sqrt{S_{dW}} + \epsilon} \tag{3.62}$$

$$b := b - \alpha \frac{db}{\sqrt{S_{db}} + \epsilon} \tag{3.63}$$

6. Adaptive Moment Estimation (ADAM)

   This technique mixes the ideas in momentum and RMSprop techniques. In practice, ADAM optimizer has shown significant success, and that is the reason it got selected as the optimizer in the inspection tasks in this dissertation. To present the formulation, similar to the previous methods, let us assume that at iteration t, dW and db are calculated. For initializing the algorithm, we set $V_{dW}, S_{dW} = 0$ and $V_{db}, S_{db} = 0$. To make the formulations presented in Equations 3.64-3.67 more readable, β_1 refers to the coefficient in the momentum approach, and β_2 refers to the RMSprop coefficient.

$$V_{dW} = \beta_1 V_{dW} + (1 - \beta_1) dW \tag{3.64}$$

$$V_{db} = \beta_1 V_{db} + (1 - \beta_1) db \tag{3.65}$$

$$S_{dW} = \beta_2 S_{dW} + (1 - \beta_2) dW^2 \tag{3.66}$$

$$S_{db} = \beta_2 S_{db} + (1 - \beta_2) db^2 \tag{3.67}$$

   As it was explained in the bias correction step, to make the computations more accurate, the following corrections are applied to V_{dW}, V_{db} and S_{dW}, S_{db} according to Equations 3.68, 3.69, 3.70 and 3.71.

$$V_{dW}^{Corrected} = \frac{V_{dW}}{(1-\beta_1^t)} \tag{3.68}$$

$$V_{db}^{Corrected} = \frac{V_{db}}{(1-\beta_1^t)} \tag{3.69}$$

$$S_{dW}^{Corrected} = \frac{S_{dW}}{(1-\beta_2^t)} \tag{3.70}$$

$$S_{db}^{Corrected} = \frac{S_{db}}{(1-\beta_2^t)} \tag{3.71}$$

After the bias correction, weight and bias terms can be updated following Equations 3.72 and 3.73. As mentioned earlier, ADAM optimizer has shown promising results which is the reason that it is heavily used in deep learning tasks. The developers of this outstanding optimization technique presented the best and default values for the hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Moreover, α can be tuned depending on the context (Kingma and Ba, 2014).

$$W := W - \alpha \frac{V_{dW}^{Corrected}}{\sqrt{S_{dW}^{Corrected}} + \epsilon} \tag{3.72}$$

$$b := b - \alpha \frac{V_{db}^{Corrected}}{\sqrt{S_{db}^{Corrected}} + \epsilon} \tag{3.73}$$

3.5.3 Inputs and Outputs

Since our goal is to have an image classification algorithm, it is obvious that the inputs of the model are images. However, computers accept and process the images as an array of pixel values. Depending on the resolution of the image, each input volume has three dimensions:

1. **Width** which refers to the number of pixels in the image width-wise.
2. **Height** which refers to the number of pixels in the image height-wise.
3. **Depth** refers to the number of channels or colors. The most obvious depth is **RGB** which refers to red, green and blue channels in a single image.

To provide an example of the inputs, consider that we have a color image in a format of JPG with size 224×224, then the array that the computer has to process has the dimension of $224 \times 224 \times 3$. Each element of this array has the value between 0 and 255 which reflects the pixel intensity at a point.

The outputs of the model will be a vector of probabilities that an image belongs to a specific class. To clarify, suppose that an image of a laptop is given to the algorithm. So, the output array includes probability values that the input image belongs to all defined classes such as a laptop, water bottle, cellphone, desk lamp, and a book. Then the array would be like $(0.90, 0.02, 0.02, 0.02, 0.02, 0.02)$.

3.5.4 Convolution Layer

In the context of neural networks, the inputs are 1D arrays. Suppose we are dealing with RGB images that have a minimal size (64×64), then the input dimension is $64 \times 64 \times 3 = 12288$ which makes the computations highly expensive. Considering higher quality RGB images in the order of 1000 leads to solving a neural network with an input size of $3M$. Lets only think about the first hidden layer with 1000 units, then the dimension of the first layer weight matrix will be $(1000, 3M)$ which is enormous. To make the size of the problem manageable, it is necessary to follow the **convolution** operation which is explained in the remainder of this section.

In Figure 3.8 an example of a 6×6 image is presented as an input to the neural network. The results of the convolved image are presented. The main purpose of the filter (or kernel) is to extract some specific features. It shines over a region of the image which is called the receptive field, and then the element values in the filter are multiplied with the original pixel values of the image (computing element-wise multiplications). Then the filter is slid all over the image as shown in Figure 3.9 and the multiplications are summed up to a single number.

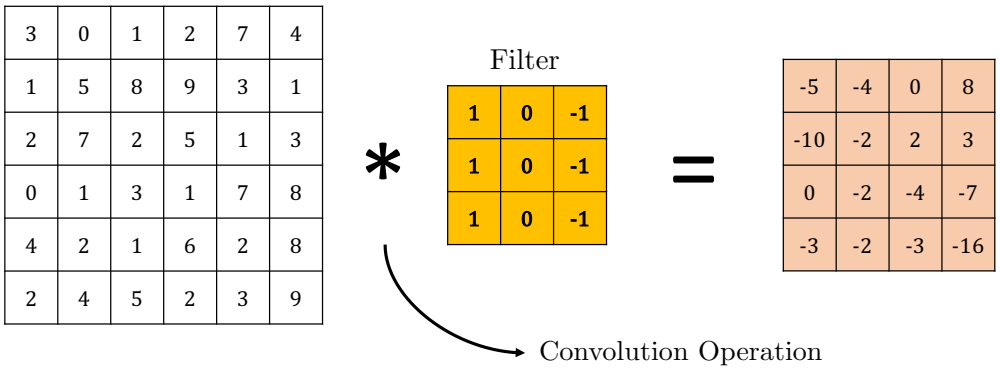


Figure 3.8: Convolution Operation

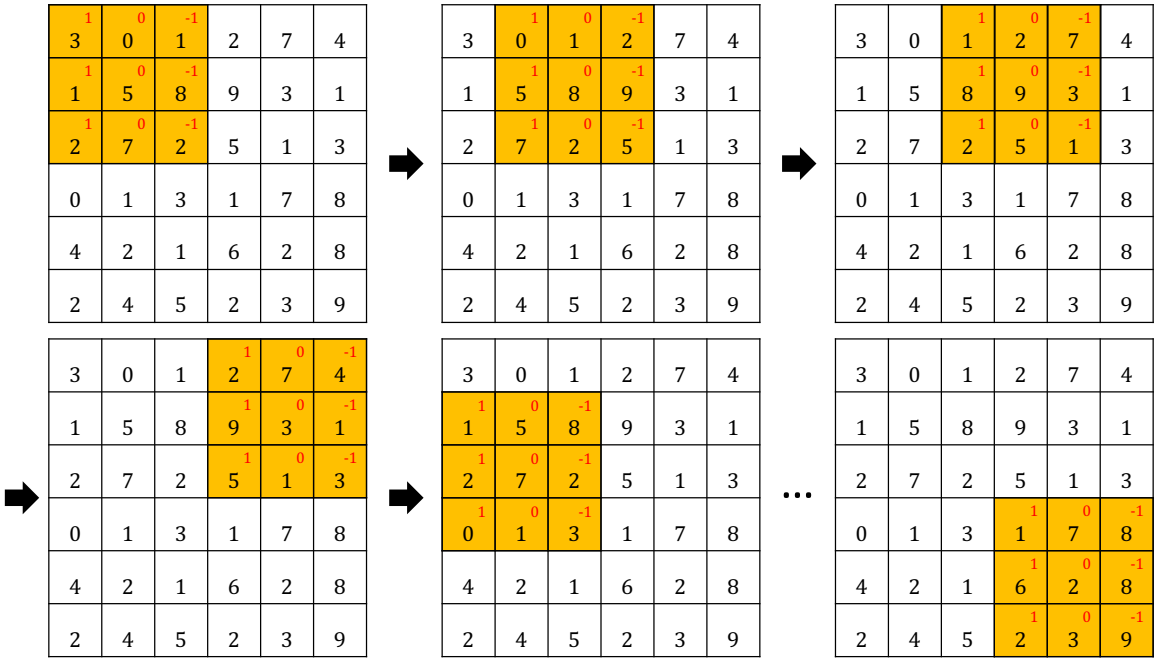


Figure 3.9: Convolving between a Filter and an Image

Each of the locations in the input matrix produces a number. After sliding the filter over all the locations, we get a 4×4 matrix (the size of the input is reduced, but the main features are still there), which is called an activation map or feature map. The output of an $(n \times n)$ image that is convolved by a $(f \times f)$ filter has the size $(n - f + 1) \times (n - f + 1)$.

It worths mentioning that the pixels in the corners of the input image are only touched once and there is a risk that some information is missed when convolving the filter through the edges of the image. Moreover, the output has a smaller size when compared to the input image. To address the mentioned issues, the **padding** approach has been introduced that adds some rows and columns to the original input image with pixel values of 0. The output of an $(n \times n)$ image that is convolved by a $(f \times f)$ filter with padding p has the size $(n + 2p - f + 1) \times (n + 2p - f + 1)$.

As it was discussed earlier, to add non-linearity to the network, it is essential to have activation units through the network. Figure 3.10 represents how a single layer of a CNN works where the input is an RGB image of 6×6. Filters serve as the weight matrices in neural networks. Since the purpose of the CNN for us is the image classification, the activation function is set as ReLU. Layers activated with ReLU perform significantly better than other activations layers in terms of the pace in training samples with a high level of accuracy. In the lower layers of the neural networks, usually, the training process is slow because the

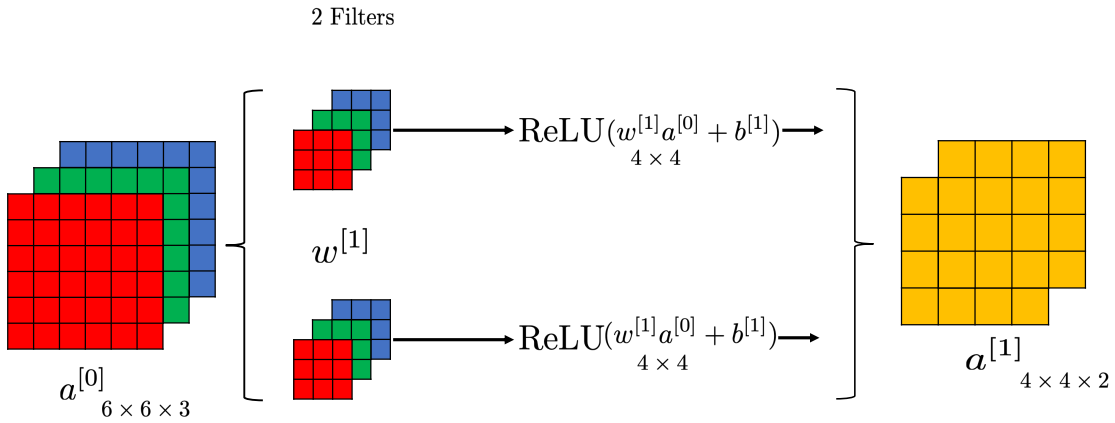


Figure 3.10: Single Layer of CNN

gradient is vanishingly small which prevents the weight to be updated. It is reported that ReLU suffers less from this issue (also known as vanishing gradient problem).

$$z^{[1]} = w^{[1]}a^{[0]} + b^{[1]} \tag{3.74}$$

An activation function is applied to the output for adding non-linearity (as a convention in CNNs or any other neural networks) which works like:

$$a^{[1]} = g(z^{[1]}) \tag{3.75}$$

In addition to the convolution layer in CNN, two other types of layer exist that are explained briefly in the following.

3.5.5 Pooling Layer

After applying the activation layer, usually, **pooling** is implemented which is also known as a **down-sampling layer**. Pooling layer takes a filter and a stride and depending on which pooling function is used (let us say max pooling), the maximum function is applied to the input volume and outputs the corresponding value in each of the sub-regions. An example of applying max pooling and the average pooling is presented in Figure 3.11.

Pooling layers are applied for two purposes. The first one is to make the network computationally more efficient (pooling results in less number of parameters through the network). The second reason is to reduce the risk of overfitting (when the model fits too well to the training set and is not so generalizable). Reducing the number of parameters is translated into less complexity of the network.

3.5.6 Fully Connected Layer

The last layer of the network is called the **fully connected** layer which takes an input volume that can be the output of a convolution layer, activation or a pooling layer and

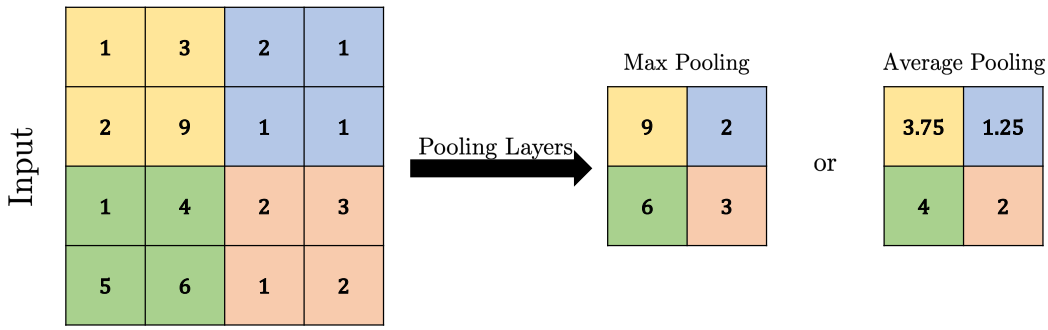


Figure 3.11: Pooling Layer Example

outputs a 1D array with N (the number of classes that the program has to choose from) elements. Depending on the properties of the problem, if it is desired to classify images of packed boxes to defective and non-defective classes, the output of the fully connected layer would be a vector with two elements, each of which represents the probability that the image belongs to each of the classes. This layer looks at the output of the previous layer in which it contains the high-level features and then determines which feature has the highest correlation with a particular class.

Overall, the convolution layer has two advantages over fully connected layer:

1. Parameter Sharing

   A filter or a feature detector that is used in a specific region of the image might be useful in any other region of the image.

2. Sparsity of Connections

   In each layer, each output value is dependent upon a small number of inputs. So, training with smaller training samples is possible in which the trained layer is less prone to overfitting.

3.6 Multi-Label Classification of Cases: Accountability Model

In this section, it is intended to localize any defect that is detected in the cases. In other words, based on the fact that we have the top-down view images of the cases, we want to identify the location of the defect and find out in which quadrant of the case the defect is seen. To reach this specific aim, multi-label classification methods have to be explained. In modern applications, such as semantic scene classification (Boutell et al., 2004) when an image can belong with more than a single class, music categorization (Li and Ogihara, 2003) when a song can be associated with more than one genre, protein function classification (Zhang and Zhou, 2005), etc., multi-label classification methods have become more popular (Tsoumakas and Katakis, 2007). In the traditional classification techniques, learning is based on a data set of examples with a single label from a set of disjoint labels associated with each. The size of the set of disjoint labels defines if the problem is a binary classification (when we have two disjoint labels) or multi-class classification (in case of having more than two disjoint labels).

However, in multi-label classification, each example in a test set is associated with a set of labels. This task is originated from text categorization and medical diagnosis field in which a single sample can have more than a single label. For instance, a patient can be diagnosed with high blood pressure and diabetes at the same time. Multi-label classification is very similar to the task of ranking in supervised learning in which we have to order a set of labels in a way that the greatest labels are more relevant to the new instance. It has to be mentioned that in label ranking, post-processing is required to present the outputs in a set of labels.

In the multi-label classification task, we want to answer the following question:

"Can we identify the regions of an image which activate a particular label while doing the task of classification with CNN?"

The most significant difference between multi-class and multi-label classification is that in the first one, an observation/sample can only be assigned to a single class. For instance, an image of a single animal could be classified into a cat or a dog, but not both at the same

time. Whereas in the multi-label classification, in a single image, we are looking for different objects. For instance, an image can include a rabbit and a carrot. In summary, in multi-label classification, there is no hard wrong misclassification, and there is a possibility that only a subset of actual classes is predicted. Therefore, the more labels that are predicted in an instance correctly, the better the prediction is.

Let us assume that we want to have a multi-label classification model based on the outputs of the model presented as Equation 3.76 for a sample image of a case.

$$z = [-1.0, 5.0, -0.5, 4.7] \tag{3.76}$$

$$softmax\,(z) = [0.0014152405960, 0.5709488061, 0.002333337273, 0.4229692786] \tag{3.77}$$

After applying softmax activation, the outputs are given in Equation 3.77, which clearly shows that class two and four should be selected. The main problem is that by using softmax, we do not have a deterministic value for the number of labels; in other words, we do not have a threshold. In this dissertation, we assumed that the probability of having a defect in a quadrant of a packed case is independent of the probability of having a defect in another quadrant. In other words, defects in each of the quadrants are equally likely to be observed. This assumption makes the use of another activation function (sigmoid) that is used mainly for binary classification possible. After applying sigmoid function 3.46 to the output vector z, we get:

$$sigmoid\,(z) = [0.2689414213, 0.9933071490, 0.3775406687, 0.9933071490] \tag{3.78}$$

By applying the sigmoid activation function to the output layer, we get the probability of label l_j as a Bernoulli distribution where the probability of having each label is independent of other label probabilities. In common practice, the threshold is set as 0.5 to get the binary outputs for label assignments.

$$P(l_j|x_j) = \frac{1}{1 + exp(-z_j)} \tag{3.79}$$

The model is called accountable since it enables us to connect the defects with the steps of the placement strategy and the SOP created by the optimization model. The results of the models are presented and discussed in the next Chapter 4.

Chapter 4

Results within a Case Study

4.1 Case Study

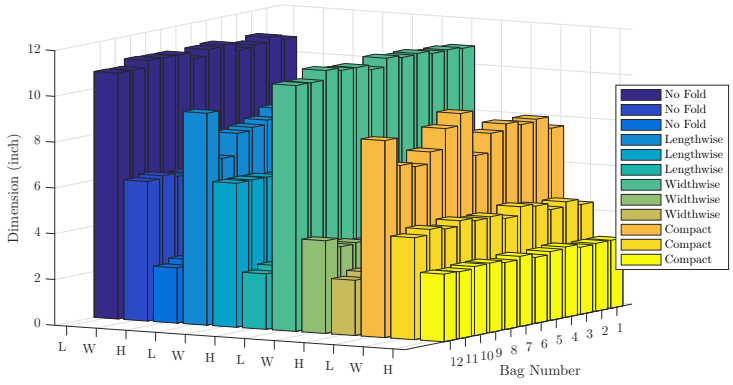
4.1.1 Explanation of Case Study and Measurements

This study is based on a project that was done for a company in charge of distributing items to the demand nodes. The company had three packing facilities (assemblers), each of which has their own standards for placing the bags inside the cases, but all of them have to follow a regulation (about the items that should be in a case) that is given to them multiple times a year. The identity of the company discussed in this case study has been protected.

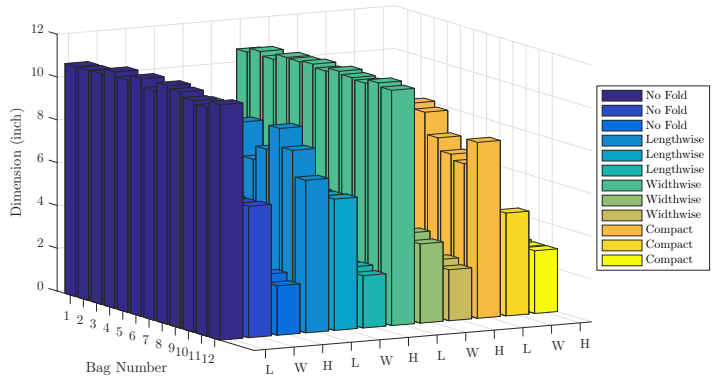
Currently, each time a change in the system happens, or a new package has to be produced, each assembler spends some time (up to a week or more) on figuring out the best way to place the bags inside a case. After coming up with an acceptable approach, the workers of the line are trained. The training is just by showing them the change in the current system and pointing out the differences in their daily routine. One of the goals of this dissertation is to propose a standardized way to find out the best configuration in a short time in an automated way. Thus a GUI is developed only needing the dimension of the new bags and by pushing buttons, a set of training videos will be generated for different stations.

In this section, 10 cases were selected, and the contents (12 bags which include food items and have different sizes) were measured. These cases were collected from three different

packing facilities; therefore, a variation in the sizes was among the initial observations. As explained in the previous sections, the contents of each case are some bags that have near-rectangular shape but with some unused space. As part of the study, we propose to fold the bags from different possible ways (from length, width or both) to make the packs more compact and avoid any bulges in the final packed case. To make this problem and its characteristics clearer, a presentation of how foldings have an impact on the sizes of bags is included in Figure 4.1 for only two of the 10 cases. When the bags were folded from a specific side, for instance, the length, the other dimensions (width and height) changed slightly. This is because each bag includes various content items and squeezing it from a side makes the content to be jammed in the other parts of the bag. In the next section, the data is used as inputs of the model, and the results are explained.



(a) A Demonstration of Data Set 1: Dimension of Different Bags in Various Folding Scenarios



(b) A Demonstration of Data Set 2:Dimension of Different Bags in Various Folding Scenarios

Figure 4.1: Input Data Set to the Model

4.1.2 Numerical results

Packing for all 10 cases was solved by the optimization model in Section 3.2 and 3.3 using Gurobi solver (Optimization, 2017). The stopping conditions for the solver were set to whichever out of "optimality" or "after 60 minutes" came first (the optimality gap for the proposed model is presented in Table 4.1). To analyze and interpret the results, a metric named "improvement region" in introduced for proving that the model is capable of giving the placement strategy that uses folding of bags in a way that all of them are within the case dimension considering a safety margin.

The current height of the cases that used by the assemblers is 10.5 inches, and when solving the model the lower bound is set to 9.5 which is the best height value we expect to get from the model (one inch is considered as a safety margin). The expectation is set to make sure that all the bags are placed within the case when following the results of our proposed model and no extra force is required for closing and sealing the case. The results reveal that in all the instances that have been studied in the model, the case height is between 9.5 and 10.5 that assures that no extra compression is needed for packing the cases. In contrary, when testing other scenarios with the assumption of rigidity, some results show that the placement solution is not feasible, therefore external compression is required to be able to close the cases.

Three scenarios are defined for the assessment of the optimization model:

- **Scenario 1**: In this scenario, we do not fold any of the bags. So the contents of the bags are loose, and plenty of empty space is there. Since the bags have the near rectangular shapes, we solved the problem with the model that was presented by (Wu et al., 2010). The height of the case in this scenario has the largest value, and unutilized space is the largest as well. We call this scenario as **"No Folds"** since we are not changing the bag configurations at all.

Table 4.1: The Optimality Gap of the Proposed Model (Optimal Fold Scenario) when the Lower Bound is 9.5 Inches

Test Number	1	2	3	4	5	6	7	8	9	10
Optimality Gap (%)	2.68	1.97	2.30	11.04	5.26	1.06	6.77	0.00	6.79	6.29

- **Scenario 2**: In this scenario, we wanted to test our folding strategy; thus we call this scenario as **"Optimal Folds"**. For this scenario, we solved the problem with our proposed model, and it was observed that the optimal case hight is almost the smallest. Similarly, for the un-utilized space objective, this scenario leads to the best results in almost all cases.

- **Scenario 3**: In the last scenario, we wanted to test the hypothesis whether making each bag as its most compact configuration leads to the minimum case height or not. Therefore this scenario is called **"Compact Bags"**. The results will be explained in details in the next sections.

Objective values (Height)

The optimization objective of the model is to minimize the case height occupied by the packing configuration. Since the current case height was known, the results are presented in a way to visualize and quantify the improvement. Reduction of the occupied height, with no flexibility afforded to the case length and width, implies that there are no bulges and the bags comfortably occupy the case volume. The shaded region in Figure 4.2 illustrates that any objective value between the limits – from the lower bound to the current case height $[9.5, 10.5]$ – reflect improvement compared to the currently used case height. The results of the proposed model remain in the "improvement region", meaning that all bags can be placed inside the cases without the need for external compression.

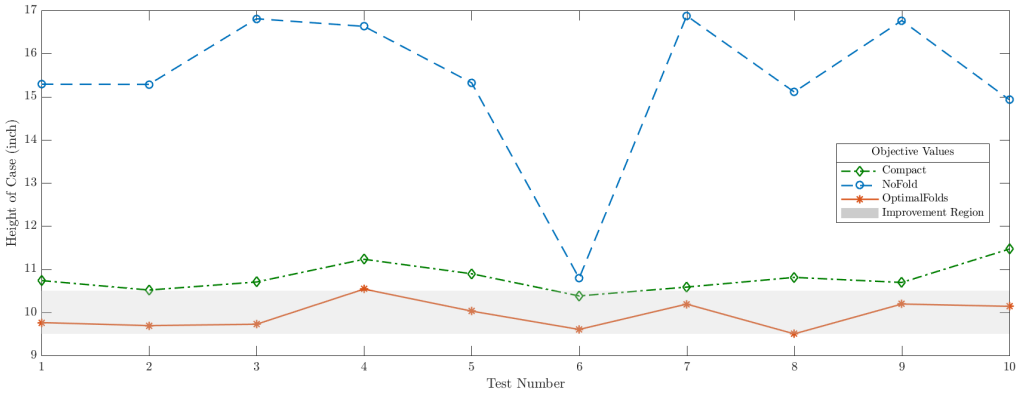


Figure 4.2: Results of 10 Tests in Terms of Objective Values (Case Height) with respect to Each Scenario for Comparing the Benefit of Using Bag's Flexibility

As anticipated, the no folds scenario performs the worst. For all examples except case 6, there is a significant difference in the occupied height between this scenario and the other two scenarios. Figure 4.2 shows that compact folds and optimal folds generate comparable output for the occupied height, with optimal folds performing better on all the test cases.

Un-utilized space

With respect to the value of the optimal height of the case presented in the previous section (see Figure 4.2), the non-utilized space in the cases are calculated according to Equation 4.1. Unlike the results presented in the previous section, there is no specific improvement region as an evaluation metric shown here. The non-utilized space can be interpreted as the dead space in which the contents of the case can slide relative to each other, where cases with more unused space are more prone to quality degradation. Therefore, the interpretation is that the closer the values of the non-utilized space to zero, the better is the placement. Figure 4.3 illustrates the non-utilized space per case in each scenario. Expectedly, no folds results in the largest non-utilized space. The results of the proposed model in all test cases have the closest value to zero which verifies that following the optimal folding is beneficial in case space utilization.

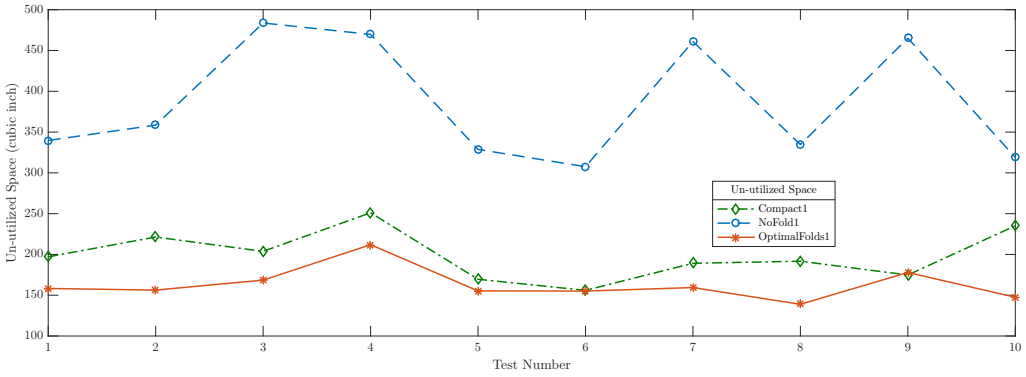


Figure 4.3: Results of 10 Tests in Terms of non-utilized Space Values (Case Height) with respect to Each Scenario for Comparing the Benefit of Using Bag's Flexibility

$$\textit{Non-utilized Space}_{Case_j} = L \times W \times \tilde{H}_j - \sum_{i \in A} \sum_{k \in K} (l_{ik} N_{ik}) \times (w_{ik} N_{ik}) \times (h_{ik} N_{ik}) \qquad (4.1)$$

Choosing the Right Folding Scenario

Optimal folds perform better on both evaluative metrics (in height minimization (Figure 4.2) as well as non-utilized space (Figure 4.3)). Which scenario is then the suitable choice for packing flexible bags?

The choice may be guided by the consideration of packing time. Each fold adds minimally to the packing time, but the overall effect is to slow down the process. No folds seems to be the most practical choice from this perspective. However, no folds must be accompanied by an additional packing station for mechanical compression, which has a balancing effect on the effective packing time. It also reduces overall throughput because of quality rejects resulting from excessively bulged cases or cases damaged during mechanical pressing.

Compact folds is the most time consuming, since all bags must be fully folded. For each case containing 12 bags, as seen in the packing facility, 24 folds are required. Packing according to optimal folds, on the other hand, considers the number of folds to be a decision variable. Figure 4.4 shows that there is no optimal folding instance, in the dataset, which required 24 folds. This provides empirical evidence that optimal folds will reduce the amount of work needed for packing, while providing better space utilization. There is, therefore, a time advantage gained by the adoption of optimal folds.

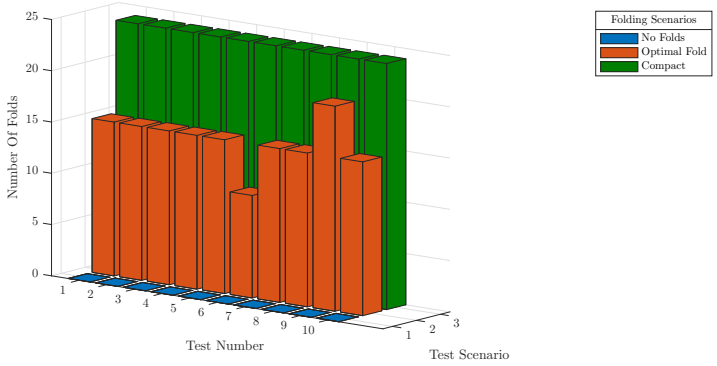


Figure 4.4: The Number of Required Folds in Each Scenario in Each Test Case

Model Scalability

In order to test the effect of an arbitrary increase in the number of items and study the scalability and computational complexity, extra tests have been performed. In the case study presented in this paper, the cases must contain only 12 bags. Therefore, instances with more number of bags have been created by random data generation based on the existing data on hand for cases with 12 bags. It is assumed that in the case with 48 bags, four cases with the current dimension is placed side by side to form a larger case with the length and width two times larger than the current cases but with the same height. In the same fashion, in the case containing 72 bags, the length dimension is three times larger than the currently used case, and the width is twice larger, but the height is kept unchanged. The largest instance considers stacking two cases containing 48 bags; therefore the length and width are similar to that, but the height is doubled. The choice of setting the lower bound is identical to the previous instances. The stopping conditions for the solver in all the large instances were set to whichever out of "optimality" or "after 60 minutes" came first. Based on a parameter study and various tests, it has been concluded that to have a faster convergence, it is beneficial to change the "symmetry" and "presolve" parameter to its "aggressive" mode. The results that are presented in Table 4.2 reveal that the model is scalable and therefore it is applicable to the situations with larger cases.

4.2 Sensitivity Analysis

As it has been mentioned, the selected objective function for the optimization is the case height. The reason for making this choice among all other possible objective functions is the feasibility of implementation for our partner company. Since changing the case height

Table 4.2: Scalability of the Proposed Model

Number of Bags	Case Height (Inch)	Current Case Height (Inch)	Lower Bound (Inch)	Optimality Gap (%)
48	10.34	10.5	9.5	8.84
72	9.5	10.5	9.5	0.00
96	20.62	21	19	8.52

is possible with minimal cost, a comprehensive study has been done to check the effect of the proposed model that uses the flexibility of the bags to utilize the space in the cases efficiently. However, to check out the model performance when altering other dimensions of the case is allowable, sensitivity analysis has been performed. The purpose of doing the sensitivity analysis is to find the answer the following questions "what would the best case length and case height when bag folding is followed?" and "how much is this improvement if the dimension can be reduced at all?"

To study the performance of the proposed model, the objective function is evaluated when the length and the width of the cases are minimized. The same test cases are used (these cases were obtained by collecting them from different assemblers that were in charge of packing the items for preparing them to be shipped) and the choice of the objective lower bound has been made in a similar fashion to the height minimization. The lower bound is only used to calculate the optimality gap of the model. Since the current case length and width is easily measurable, any result that is less than the current case dimension can be referred to as an improvement. Each of the tests were stopped after 60 minutes of runtime.

$$\textit{Case Length Lower Bound} = \textit{Current Case Length} - 1 \qquad = 14.6875$$

$$\textit{Case Width Lower Bound} = \textit{Current Case Width} - 1 \qquad = 8.0625$$

Table 4.3: Sensitivity Analysis for Testing the Effect of Folding on Case Length

Experiment	Improvement (%)	Optimal Solution	Optimality Gap (%)
Test 1	1.30	15.48	5.14
Test 2	1.78	15.41	4.68
Test 3	1.64	15.43	4.81
Test 4	NA	16.51	11.00
Test 5	NA	15.70	6.44
Test 6	6.37	14.69	0.00
Test 7	2.97	15.22	3.51
Test 8	NA	16.56	11.30
Test 9	NA	16.67	11.90
Test 10	0.35	15.63	6.05

The summary of the results when the case length is minimized can be found in Table 4.3. Among the available cases, the results show that the proposed model can improve the case length in six tests out of ten. The results are achieved when keeping the height and the width of the cases unchanged and equal to their current configurations. One possible reason for explaining the incapability of the model in improving the length in some cases is that minimizing the length leads to the highest space reduction and therefore, the hardest to be solved. Based on the observations during case packing in the facilities, most of the times the bulges appeared from the width and height sides of the cases which explains the use of compression heads and side rollers. In a nutshell, with the model that utilizes folding of the bags to save some space in the cases, it might be better to focus on improving the width and the height of the cases that most likely cause deformations to form.

Similar analysis has been performed to check the effect of item flexibility on the case width. The case width is optimized when keeping the length and height unchanged. The results that are presented in Table 4.4 illustrate that in nine out of ten tests, the case width is improved.

Table 4.4: Sensitivity Analysis for Testing the Effect of Folding on Case Width

Experiment	Improvement (%)	Optimal Solution	Optimality Gap (%)
Test 1	5.26	8.59	6.10
Test 2	4.26	8.68	7.07
Test 3	8.25	8.31	3.03
Test 4	NA	9.86	18.30
Test 5	4.45	8.66	6.89
Test 6	7.43	8.39	3.89
Test 7	9.23	8.23	1.98
Test 8	3.45	8.75	7.86
Test 9	9.21	8.23	2.01
Test 10	2.33	8.85	8.91

4.3 Software Implementation

In this case study, a graphic user interface (GUI) for both input and output is developed using *Python*. The outline of the GUI is presented in Figures 4.5, 4.6 which show the only thing the users need to do is to measure the dimensions of the bags that need to be placed inside the case and input it to the GUI. The last step is tuning the optimization solver and run the model. As soon as the results are gotten out of the model, in a post-processing step using *VPython*, the results are visualized in a format of a video that is explained in more detail in the next section.

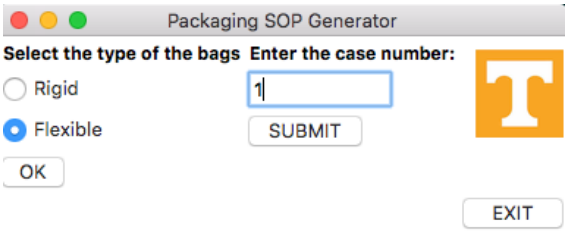


Figure 4.5: The First Window of the GUI

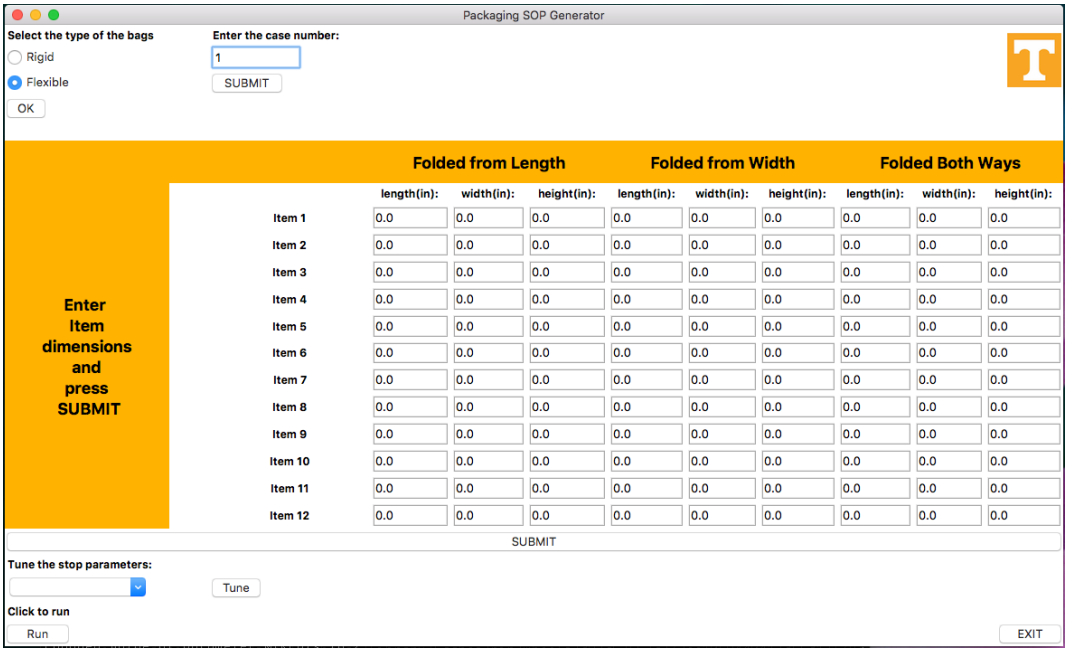


Figure 4.6: The Second Window of the GUI

4.3.1 Automatic SOP Generation

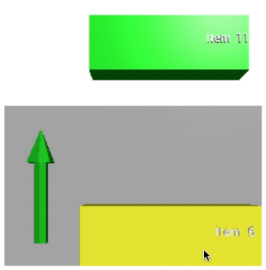
As discussed earlier, the presented optimization model takes advantage of the flexible nature of bags by suggesting ways of folding them before packing inside a case, which leads to further improvement in space utilization. A Standard Operating Procedure (SOP) is automatically generated using the optimization model, to make sure that the results of the model are implementable in its easiest way. Moreover, it creates a common standard tool for all the users that are in different facilities in the case study. The SOP has the advantage of self-updating based on any changes regarding the bag sizes which has a direct impact on improving the training regimen and reducing overall labor time invested in training.

Multiple screenshots of SOP are presented in Figure 4.7 where each color shows a specific type of folding. *Yellow* means that no folds are needed, *green* shows that the bag has to be folded from width, and *orange* tells that the bag must be folded from both sides. The color *blue*, which is not part of the results in this specific test number (6), is a sign to fold a bag lengthwise.

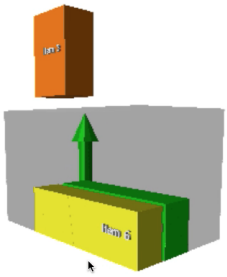
4.4 Inspection Results

Convolutional neural networks (CNN) are a type of neural networks that have shown much success in image classification. Inspired by the neurological concept of receptive fields, CNNs essentially process the images in patches and its neurons "fire" when a certain type of feature is detected in the image such as edges or shapes. The neurons learn how to detect and when to "fire" through an optimization step (gradient descent is among the most popular ones as explained earlier in this chapter). So, CNNs do not require extensive manual preprocessing or feature extraction, but they do need a lot of images to be able to predict on real-world data.

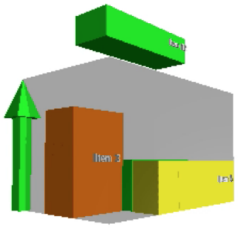
In 2012, a convolutional neural network achieved a revolutionary error rate of 15.3% in the ImageNet challenge which consisted of classifying around 1.2 million images. CNN architectures continued to win subsequent ImageNet challenges, and by 2015, CNNs had achieved an error rate lower than the human error rate. This leads us to believe that,



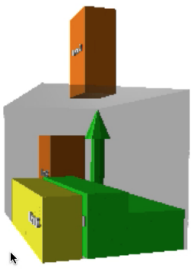
(a) Step 1



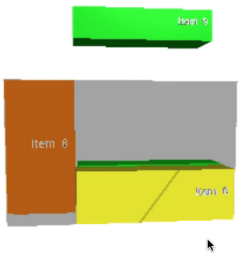
(b) Step 2



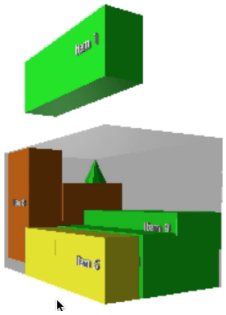
(c) Step 3



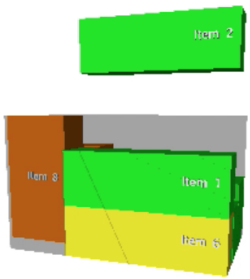
(d) Step 4



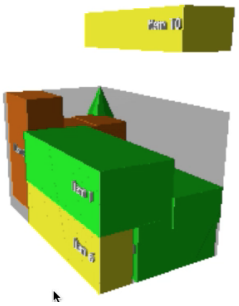
(e) Step 5



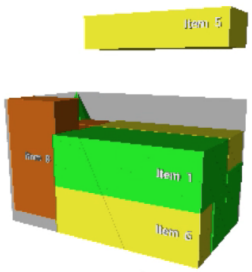
(f) Step 6



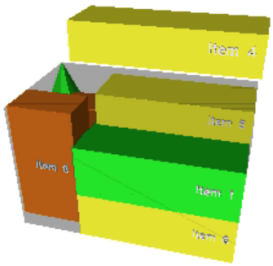
(g) Step 7



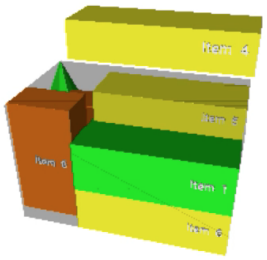
(h) Step 8



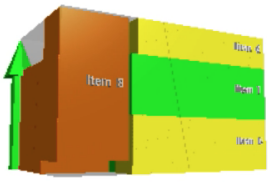
(i) Step 9



(j) Step 10



(k) Step 11



(l) Step 12

Figure 4.7: A Step-by-Step Representation of SOP

with sufficient data, a CNN inspection system has the potential to perform sophisticated classification tasks at least as well as human inspectors.

There have been several types of CNN architectures developed, with each newer iteration adding more layers while trying to maintain a reasonable size. ResNet is one such architecture that manages to reduce the size and still increase accuracy in image classification. Developed at Microsoft Research, it achieved an error rate of 3.75% on the ImageNet challenge. A ResNet architecture was used in the initial stages of the inspection portion of the project. However, the chosen architecture was changed to MobileNet later to have a smaller model with similar accuracy. MobileNet was developed at Google with specifically mobile and embedded applications in mind. It achieves an accuracy very close to ResNet with a fraction of the parameters and weights required.

4.4.1 Model Evaluation Metrics

The metrics used to evaluate the performance of the CNN in detecting defects are listed below:

- **Accuracy**

  Accuracy is just the number of accurate predictions divided by the total number of predictions. The accuracy can be misleading of the performance (Provost et al., 1998) when there is a class imbalance, i.e., the number of positive images are significantly less than the negative ones. Moreover in problems where the cost of having misclassified true positives (or false negative) is very high, then the accuracy is not the best metric to judge the model based upon. Therefore, it is essential to introduce precision and recall.

- **Precision**

  Precision is a metric that shows how accurate the model is based on the number of times that observations are actual positive out of those predicted positive overall. In cases that the cost of having false positive is high (like the prediction of spam emails where the user might lose some valuable information), making a conclusion based on precision is

appropriate.

$$Precision = \frac{True \quad Positives}{True \quad Positives \quad + \quad False \quad Positives} \tag{4.2}$$

- **Recall**

  Recall is another metric designed for calculating how many of the true positives are labeled as positive (true positive) using the prediction model. With the same notion in the previous metric, in cases that the cost of having false negative is high (like fraud detection), making a conclusion based on recall is proper.

$$Recall = \frac{True \quad Positives}{True \quad Positives \quad + \quad False \quad Negatives} \tag{4.3}$$

- **F1 Score**

  In cases of seeking a balance between precision and recall, F_1 score is calculated. The main difference between F_1 score and accuracy is that as explained previously, accuracy is largely contributed by a large number of true negatives. In some cases, it is more suitable to focus on false positives and false negatives. Thus, F_1 score is a more appropriate metric to be used if having a balance between precision and recall is sought. Moreover, when the class distribution is uneven in the data, choosing models based on F_1 score is better.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

- **AUROC**

  The Receiver Operator Characteristic (ROC) curve is a fundamental tool for diagnostic test evaluation and is commonly used to present results for binary decision problems in machine learning. In a ROC curve, the true positive rate is plotted in function of the false positive rate for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. It worth noting that sensitivity is equivalent to recall and specificity is a true negative

rate as written in Equation 4.5.

$$Sensitivity = \frac{True\ \ Negatives}{True\ \ Negatives\ +\ False\ \ Positives} \tag{4.5}$$

The area under the ROC curve (AUROC or AUC) is a measure of how well a parameter can distinguish between two diagnostic groups (defective or not defective). A model with high discrimination ability will have high sensitivity and specificity simultaneously, leading to a ROC curve which goes close to the top left corner of the plot. A model with no discrimination ability will have a ROC curve which is the 45-degree diagonal line. Provost et al. (1998) show that using ROC curves when evaluating binary classification can be overly optimistic due to the fact that the number of correctly classified positive instances alternates with the number of incorrectly classified negative instances.

- **AUPRC**

  The Area under the Precision-Recall Curve (AUPRC) is a metric obtained by the summation of the integral of the Precision-Recall Curve. The precision-recall curve represents the trade-off between precision and recall for different thresholds. Precision is a ratio of true positives to predicted positives; in the inspection context, it tells us how many of the predicted defective cases were actually defective. Whereas recall is another ratio involving true positives and false negatives; it tells us how many actually defective boxes were correctly classified as defective.

  The AUROC for an imbalanced dataset tends to be higher than it would be if the dataset were balanced. However, the AUPRC does not suffer from this discrepancy, which makes it suitable for binary classification tasks involving highly imbalanced datasets. The maximum value of both AUROC and AUPRC is one which signifies that the classifier is a perfect classifier.

The inspection task was done in two main phases. The initial phase of the development was broad and unrestricted in nature. The aim was to assess the approach of using CNNs over traditional machine vision techniques for their lack of hand-engineering while keeping the data-heavy requirement of CNNs in mind. It involved testing the capability of convolutional

neural networks to detect defective boxes with no other restrictions. Two main experiments were done that are listed below and thoroughly explained in the remainder of this chapter in Sections 4.4.2 and 4.4.3 respectively.

1. Binary Classification

   In this phase, the goal is to assess how successful CNNs are in identifying any defect in the packed cases. Cases are classified into "defective" or "normal" based on the algorithm. More details of this part can be found in Section 4.4.2.

2. Multi-Label Classification

   In this phase, it is intended to identify in which quadrant of the case the defect is seen. In multi-label classification used here, there is no constraint on the assignment of each instance to classes. In other words, each case is allowed to be assigned to more than a single class. In this case study, it means that the defect can be detected in more than a single quadrant. The detailed explanation of this part is presented in Section 4.4.3.

4.4.2 Binary Classification

One of the main objectives of this phase is to identify whether CNNs are suitable for identifying defective cases from normal ones. In order to study the feasibility of using these models, three major steps were followed: design of experiments, dataset creation and modeling that are explained precisely.

I. **Dataset Creation**

   Images of boxes, both normal and defective were downloaded from Google. They were labeled with a '0' if they were normal and '1' if they were defective, making this a binary classification. They were split into training and testing sets with an equal class distribution. This dataset is referred to as the *downloaded* images in the following sections.

   The metal frame that had been built earlier for data collection at site visits was utilized (Please see Figure 4.8) to take any pictures in the lab. The horizontal bar was affixed to the middle of the frame, rather than the top so that the boxes could be placed

Figure 4.8: Data Collection Apparatus

on the ground. A WiFi-enabled camera was then attached to the horizontal bar and connected to a smartphone so that the camera could be controlled remotely, and the images would get saved to the smartphone. The ground was marked with tape as a guideline for placing the box so that it was ensured that the image captured was centered around the box. The following procedure was followed for image capture:

i. The box was placed under the camera

ii. Image was captured

iii. The box was then rotated 180 degrees

iv. Image was captured

v. The box was then flipped upside down

vi. Image was captured

vii. Flipped box was rotated 180 degrees

viii. Image was captured

The box was rotated and/or flipped so as to capture all different views of the box possible from the top. This way four images of the box were captured. Next, a strategy was developed on how to utilize the cases present in our lab for obtaining all the different kinds of defects and bulges, with keeping the limited number of boxes available to us in mind. In our lab, there were three kinds of cases for a total of 15 boxes (please see Table 4.5)

Since quite a number of opened normal boxes were available, it was decided to use those boxes to simulate defects, along with one unopened box. The data collection procedure for the normal and defective boxes is described below:

For the normal boxes

(a) Images were captured of all the unopened boxes using the procedure described above to catch all views

(b) The opened boxes were taped shut with no gap between the flap and the contents unaffected, i.e., no additional packing was done, and images were captured

(c) Gaps of different sizes (but within approved range) were simulated on the opened boxes to capture images of normal boxes with an **acceptable** gap – defined by assemblers as being up to 2 cm.

(d) Then, two boxes were packed with different placement of bags to simulate a natural **acceptable** bulge and taped shut and then images were captured

For the defective boxes

Table 4.5: Description of Available Cases for Developing the Tests

Type	Number	Description
Normal unopened	5	These were packed cases that we had received from both of our site visits.
Normal opened	8	These were normal and undamaged cases that had been received prior to this project and were opened. They were packed in different degrees of fullness.
Defective	2	These were severely damaged boxes that we had received from one of the assemblers on our site visit.

(a) The damaged boxes collected from one assembler were captured in all views. Then, these boxes were damaged even further to capture the extremity of damage.

(b) Bulges were created in opened boxes by over-stuffing or changing the placement of the bags haphazardly and taping them shut with considerable pressure applied on the top. These bulged boxes were then captured. Further bulges were created by applying even more pressure on the taped boxes, either manually or through tools and then the boxes would be imaged.

(c) Some of these bulged boxes were flipped, and defects were introduced on the bottom side of the box and then all the views captured.

(d) A single unopened box was opened slightly to create deformities in the flap.

All in all, around 60 images of normal boxes and 80 images of defective boxes were captured. These images were then cropped, and horizontally flipped copies were made to create a final dataset of 280 images, termed as "lab images". Some examples of normal and defective boxes are given in Figure 4.9.

(a) Lab Images of Good Cases

(b) Lab Images of Defective Cases

Figure 4.9: Sample of Images Taken in the Lab Using the Apparatus

II. **Design of Experiments**

Two experiments were formulated on the basis of the following two questions:

(a) Box detection: can a CNN tell if there is a cardboard box in the image?

(b) Defect detection: can a CNN tell apart between defective and normal boxes?

The dataset to train the classifier was created by joining the lab images with the images of boxes downloaded from the Internet; the final size of the fine-tuned dataset was 650 with roughly 60% defective and 40% normal. The data was divided into cross-validation sets with the test set always comprising of only lab images. This was done to get the performance that would be closest to a real-life inspection setting for the users.

There were 6 experiments performed. Each was based on the percent of lab images added to the train set; starting from 0% and increment by 10% till 50%. Any proportion higher than 50% was not considered as the test set size would decrease to the point that the results would not be very meaningful and generalizable. The results regarding each experiment is presented in Table 4.6 with respect to the precision and recall metric.

Table 4.6: Results from Feeding ResNet with a Mixture of Real Images and Images from Google

Percent	Accuracy on Test Set	Confusion Matrix		Recall	Precision	Test Set Size
0	0.47	23	101	0.69	0.52	282
		49	109			
10	0.74	53	55	0.95	0.69	237
		7	122			
20	0.86	57	29	1	0.81	211
		0	125			
30	0.92	81	1	0.87	0.99	185
		13	90			
40	0.9	49	10	0.94	0.9	159
		6	94			
50	0.83	51	5	0.78	0.92	132
		17	59			

III. Modeling

For image classification tasks, deep convolutional neural networks have achieved significant improvements (accuracy of these models are presented in Figure 4.10 and Table 4.7). Like image classification, other tasks have also benefited from very deep models. That being said, by the advancement of technology, there is a trend to develop deeper models to be able to solve more complicated tasks and at the same time improve the accuracy. However, when the models get deeper, not only training such networks becomes much more demanding but also the accuracy starts saturating and sometimes degrades as well. To resolve these issues residual learning was introduced.

In the deep convolutional neural network, multiple layers are stacked up and then trained for a desirable task. Then the different level of features is learned at the end of each layer. On the other hand, in residual learning, it is intended to learn the residuals rather than the features. Residuals can be expressed as the subtraction of feature learned from an input of that layer. One of the best performing residual networks is ResNet which learns the residuals by using shortcut connections. Shortcut connection directly connects the input of the n^{th} layer to some $(n + x)^{th}$ layer. He

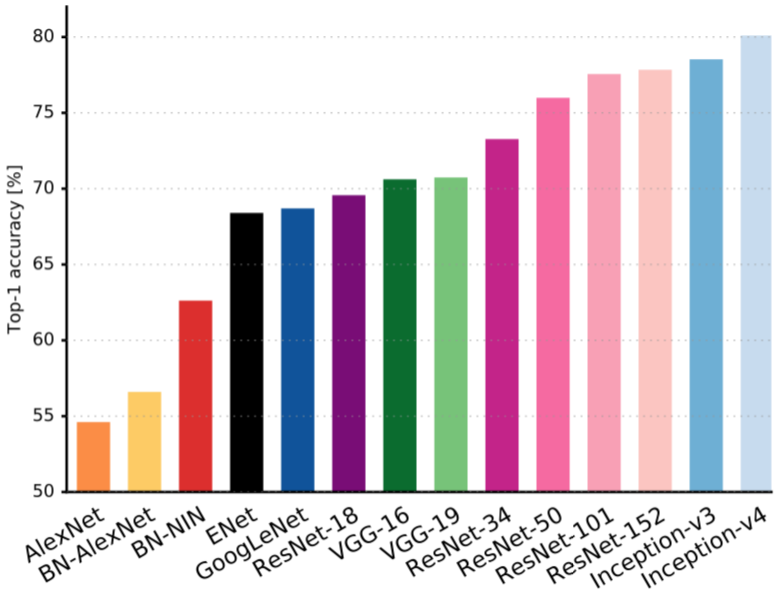


Figure 4.10: Model's performance on the ImageNet Validation Dataset – This figure is extracted from Canziani et al. (2016)

Table 4.7: Comparison of Accuracy in Deep Networks (Chollet et al., 2015)

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.79	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.9	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.75	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.96	88,949,818	-

et al. (2016) has proved that training residual networks are easier and less demanding than training a simple deep convolutional neural networks. Additionally, in training the residual networks, the problem of the accuracy degradation is solved. Therefore, in this dissertation ResNet architecture is utilized.

A ResNet architecture was created and loaded with ImageNet weights to save resources and time instead of training the model from scratch. In 2015, in a classification competition in ImageNet Large Scale Visual Recognition Challenge (ILSVRC), ResNet won first place with a top-5 error rate of 3.57%. Similarly, in Common Objects in Context (COCO), ResNet won the first place in large-scale object detection, segmentation, and captioning (He et al., 2016).

For the box detection task, the original model was used which tries to classify an image into one of 1000 categories such as car, cat, airplane, etc. Out of these 1000 categories, the "carton" and "crate" categories were taken to be the correct labels for detecting boxes. On feeding just the images to the model, it was able to correctly identify boxes in an image around 77% of the time. This was promising so the next step of identifying defective boxes was done.

For the defect detection task, the ResNet was modified such that it could classify an image into only two categories. The ResNet model was trained with both the dataset and the labels to output either "normal" or "defective" for each image. It achieved a highly desirable accuracy of 89%. Some images and their predictions are given below,

the two boxes in Figure 4.11a were classified as "normal" and the two Figure 4.11b as "defective".

In Table 4.6, "percent" refers to the percent of images taken in the lab mixed with downloaded from the Internet (as it was explained in the design of experiments). Test set comprises of only lab images.

The ResNet model performs quite well on differentiating between defective and normal boxes. However, these boxes are from any point-of-view. The results of classifying cases to "normal" and "defective" are presented in the following.

Based on the results in Table 4.6, having 30% of the images taken in the lab in the training set, leads to the best accuracy level. It worth mentioning that the model ran for ten epochs. Therefore this model is selected for further scrutinization. This model was trained further with more number of epochs (50), and the results are discussed in the following:

The confusion matrix is presented in Figure 4.12. For the test set of the size of 185, only three of them were misclassified (the boxes were defective, but the model identified

(a) Good Cases

(b) Defective Cases

Figure 4.11: Sample of Downloaded Images

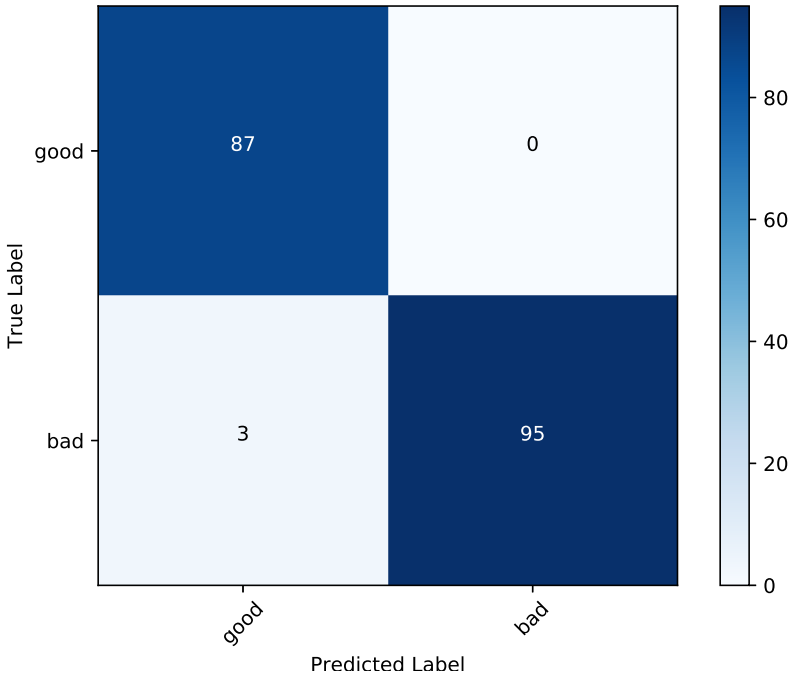


Figure 4.12: Confusion Matrix, without Normalization

them as normal). This performance is still satisfactory, but the model still missed some defects. It confirms the hypothesis that a significant component of the test accuracy can be attributed to randomness rather than pure deterministic learning. This random component cannot be eliminated completely from the model, but its effect can be reduced. The only way of decreasing the randomness of a model is to expose the model to more and more representative images of the boxes such that the randomness would have an infinitesimal effect on the testing accuracy.

Hence for the successful operation of the inspection module, it is imperative to obtain numerous real-life images of the boxes on the conveyors at the suppliers and train the model on them which is done later in this chapter.

Following the metrics explained in Section 4.4.1, the accuracy of the model can be calculated as:

$$Accuracy = 1 - \frac{Misclassified\ Samples}{Total\ Number\ of\ Samples\ in\ the\ Test\ Set} = 1 - \frac{3}{185} = 0.9838 \quad (4.6)$$

ROC curve is presented in Figure 4.13 where the area under it is calculated as 1.0 which is perfect. But, as mentioned earlier, it is better to make a conclusion about the goodness of fit based on precision-recall curves.

There is a metric in precision-recall curves called average precision score (AP). This metric is calculated according to Equation 4.7 where P_n and R_n are the precision and recall at the n^{th} threshold. The results are presented in different cut-off values in Figure 4.14. As the plots show, increasing the cut-off slightly reduces the average precision score of the classification task.

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{4.7}$$

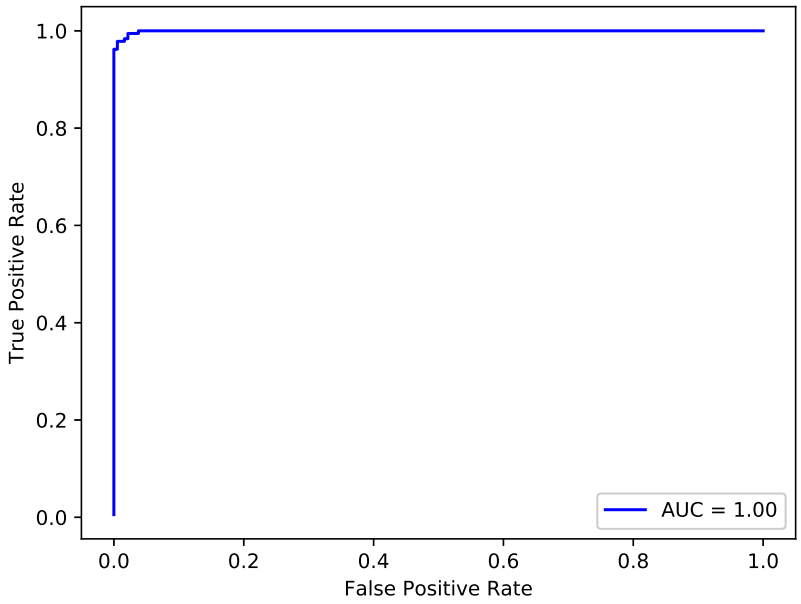
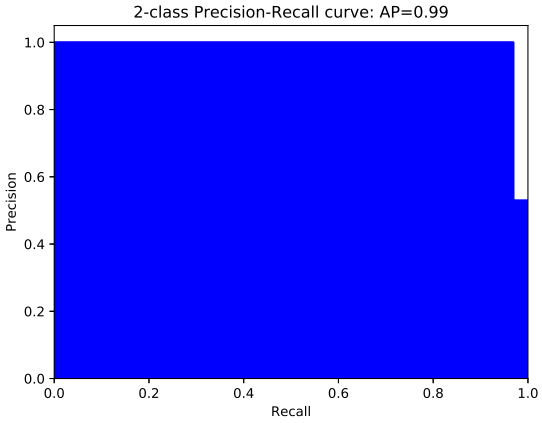
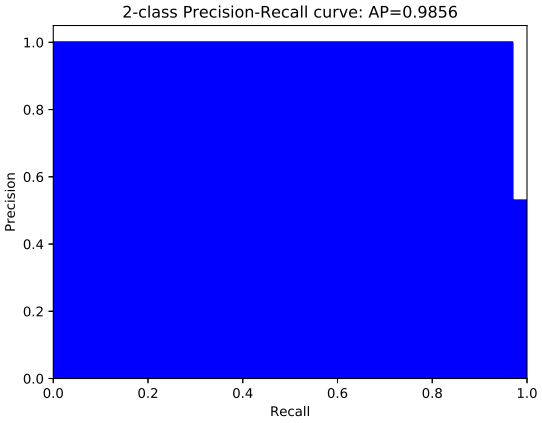


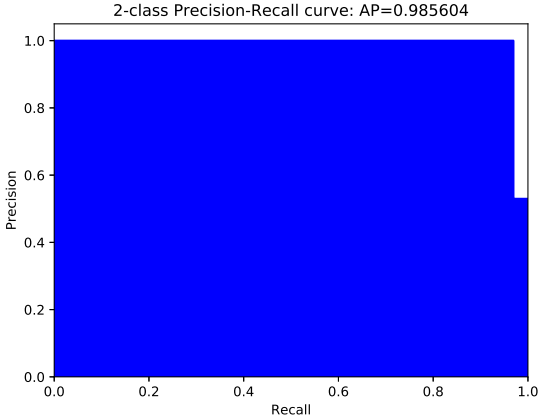
Figure 4.13: Receiver Operating Characteristic (ROC) Curve



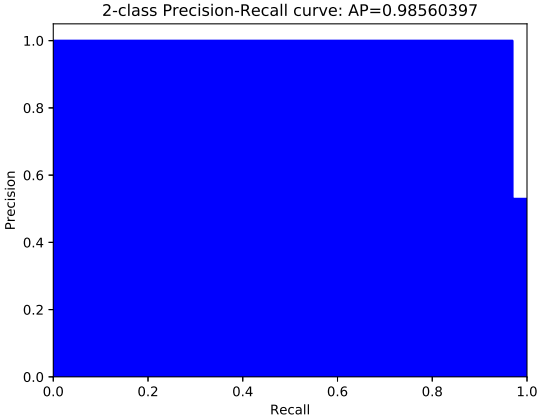
(a) Cut-off = 0.2



(b) Cut-off = 0.4



(c) Cut-off = 0.6



(d) Cut-off = 0.8

Figure 4.14: Precision-Recall Curves for Different Cut-off Limits

The results that were presented to this point were all based on lab images. The main reason was to test the success of the CNN models on defect detection. Now that the feasibility of the models is proven, the model is implemented on the data set that we collected from one of the food packing assemblers. From this point on, it is desired to check how the performance is affected when the ResNet is given only a top-down view of the box.

Results of the Binary Classification Based on Real Data

Data Collection

Camcorders were utilized to record video, and they were placed over the conveyor line using a specialized, modular frame that was presented in Figure 4.8. The camera was attached to the frame by means of an L-bracket that was fixed to the frame. The data collection was split into two periods: a test period and a full data collection period. The test period lasted one hour of line operation with the objective of finding the appropriate location for data collection on the line, resolve issues related to frame stability – fixed by using additional screws – and to see the optimal camera configuration to provide full visibility of the packed cases on the packaging line. The final choice for the camera assembly is depicted in Figure 4.15 with a green circle.

The other candidate position is depicted using a red circle, which was located after the A and B cases converge into a single line. However, this was not selected because:

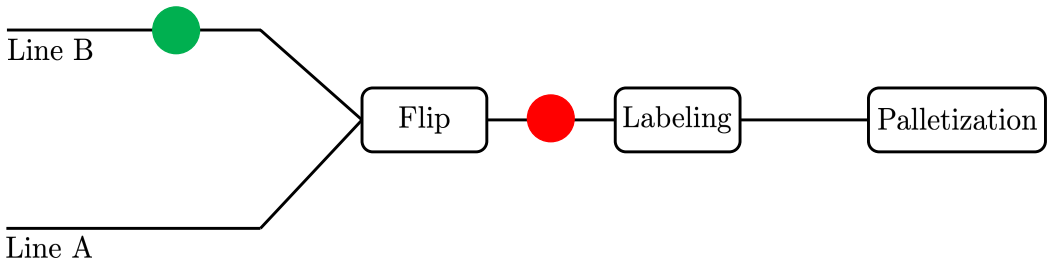


Figure 4.15: Overview of the Assembly Lines

- The cases were flipped to their side for label application before the position – which would potentially hide the packaging bulge as well as the gaps between the flaps of the cases, and

- In the current situation, the defective cases were already pulled from the individual lines – by a person in charge of inspection – before the lines converged which would hide all defective cases from the data.

Potential measures to mitigate these issues would require changes to the workflow. This was outside the scope and intent of this project. Therefore, the position on the line B – which had higher observed defects – was selected.

Data collection spanned several hours over a single day. The inspection assembly was installed at the previously selected location. By being present near the assembly at all times it was insured that ongoing work was not disrupted, and to address recording errors immediately. Approximately three hours of video was captured, and the line was continuously operating over this period.

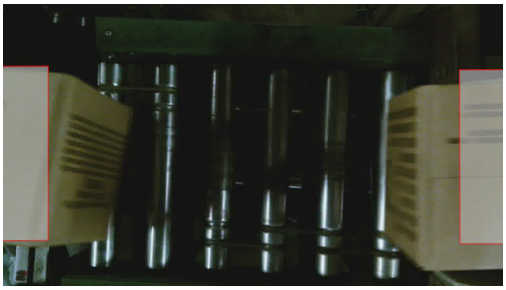
The collected videos were processed to extract all the images of the packed cases. Special care was taken to not include "transition" images in the data – that is, images in which cases were partially visible or had motion blur. The image extraction requirements were therefore stated as:

- The entire case should be visible in the image

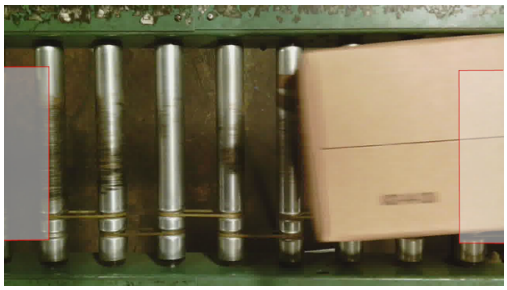
- The image should be centered around the case.

- Previous and next cases may be visible in an image but may not overlap with the central area as described above.

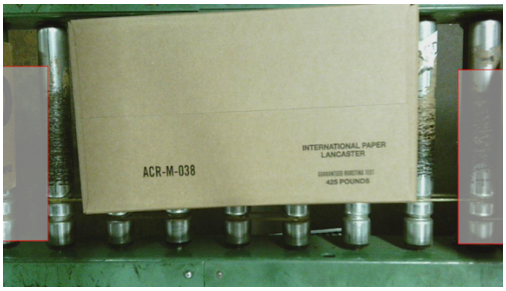
The video was sampled every second to reduce duplicates – the same case being counted twice. A simple, color-based heuristic was used to automate the identification of centralized cases in an image. The idea was to check at the borders of an image for brown color – denoting a case – in the white shaded regions shown below:



(a) Transition Image Sample



(b) Transition Image Sample



(c) Acceptable Image Sample

Figure 4.16: Image Extraction from the Recorded Video

This led to images in Figure 4.16a and Figure 4.16b being ignored, while image in Figure 4.16c is selected for further processing After preprocessing, we found 3500 usable images from the data. These images were labeled and then used for training the deep learning algorithm.

Results

The images that were extracted from the videos formed the dataset and then fed to the ResNet50 model that was performing well for the lab data. The results of the model are explained in detail in the following sections.

The confusion matrix is presented in Figure 4.17. For the test set of a size of 1382, only 21 of them were misclassified (20 of the boxes were defective but the model identified them as normal, and one of them was good and detected as a defective case). This performance is still satisfactory, but the model still missed some defects. It confirms the hypothesis that a significant component of the test accuracy can be attributed to randomness rather than pure deterministic learning. This random component cannot be eliminated completely from the model, but its effect can be reduced. The only way of decreasing the randomness of a model is to expose the model to more and more representative images of the boxes. The

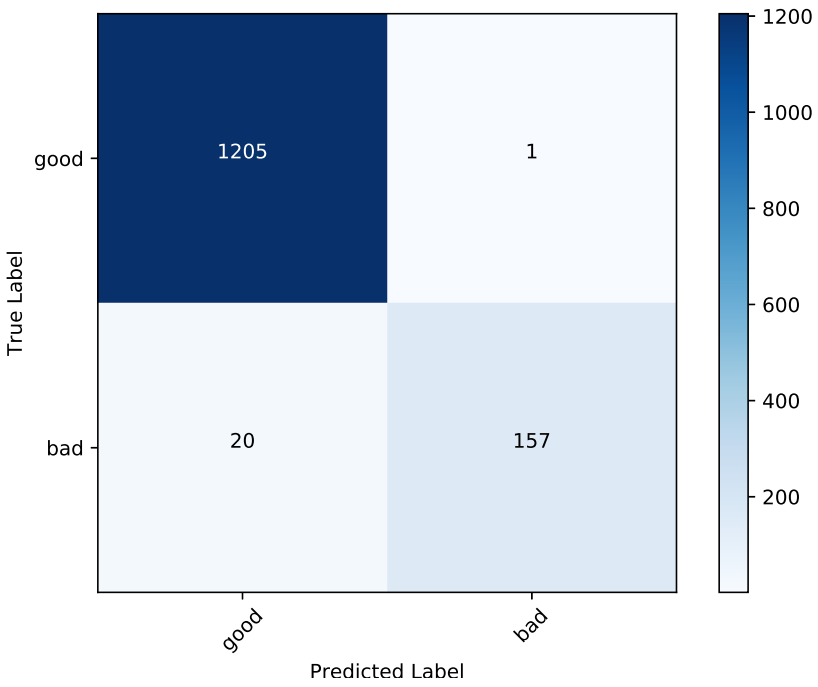


Figure 4.17: Confusion Matrix, without Normalization

data that is used was captured for almost four hours in a video format and was used as a proof of concept. Therefore by retraining the model with more data, the accuracy can be improved.

Following the metrics explained in Section 4.4.1, the accuracy of the model can be calculated as:

$$Accuracy = 1 - \frac{Misclassified\ Samples}{Total\ Number\ of\ Samples\ in\ the\ Test\ Set} = 1 - \frac{21}{1383} = 0.9848 \qquad (4.8)$$

ROC curve is presented in Figure 4.18 and the area under it is calculated as 1.0 which is perfect. But, as mentioned earlier, it is better to make a conclusion about the goodness of fit based on precision-recall curves.

There is a metric in precision-recall curves called average precision score (AP). This metric is calculated according to Equation 4.7. The results are presented for different cut-off values in Figure 4.19. As the plots show, increasing the cut-off reduces the average precision score of the classification task, and the best results are achieved at the cut-off value of 0.2.

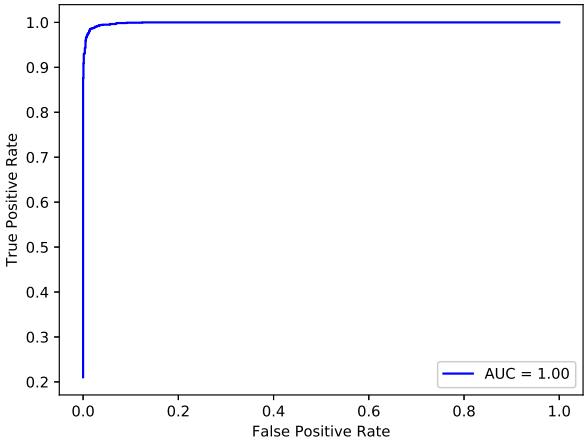
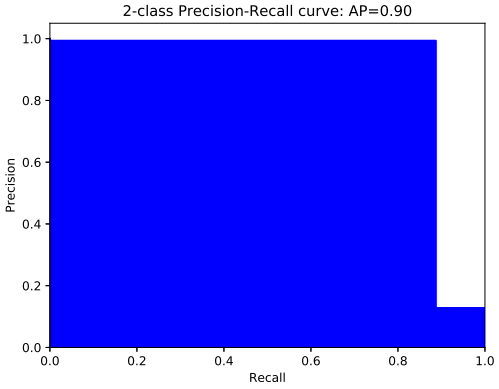
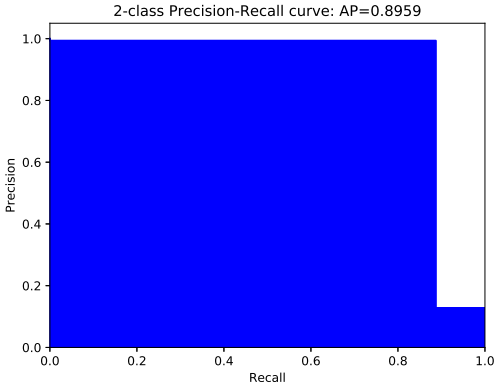


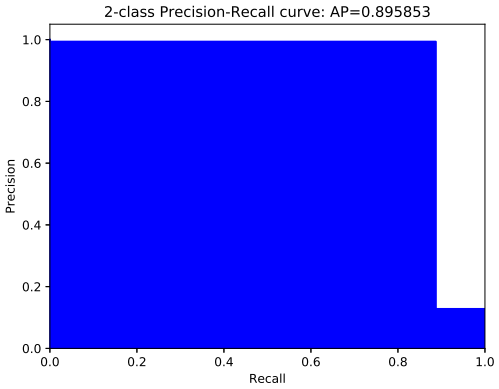
Figure 4.18: Receiver Operating Characteristic (ROC) Curve



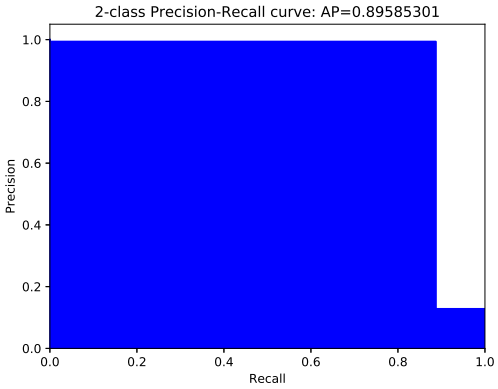
(a) Cut-off = 0.2



(b) Cut-off = 0.4



(c) Cut-off = 0.6



(d) Cut-off = 0.8

Figure 4.19: Precision-Recall Curves for Different Cut-off Limits

4.4.3 Multi-Label Classification of Cases: Accountability Model

In this section, it is intended to localize any defect that is detected in the packed cases. In other words, based on the fact that we have the top-down view images of the cases, we want to identify in which quadrant of the case the defect is seen. In contrary with multi-class classification task, we want to answer the following question:

"Can we identify the regions of an image which activate a particular label while doing the task of classification with CNN?"

The most significant difference between multi-class and multi-label classification is that in the first one, an observation/sample can only be assigned to a single class. For instance, an image of a single animal could be classified into a cat or a dog, but not both at the same time. Whereas in the multi-label classification, in a single image, we are looking for different objects. For instance, an image can include a rabbit and a carrot. In summary, in multi-label classification, there is no hard wrong misclassification, and there is a possibility that only a subset of actual classes is predicted. Therefore, the more labels that are predicted in an instance correctly, the better the prediction is.

Model

In order to have a more efficient model that has the capability to be run on mobile devices, with the reasonable accuracy values, in this section we implemented and retrained one of the most recent models that was developed in Google. As it was presented in Table 4.7, MobileNet is one of the models with a much smaller size than ResNet50, and the accuracy is in the same order. To be more precise, the top-1 accuracy that was reported in Chollet et al. (2015), ResNet50 is 74.9% accurate with a size of 99 MB, while MobileNetV2 is 71.3% accurate with the size of only 14 MB. Since the accuracy is not compromised when reducing the size of the model, we chose to implement this model for the last automatic visual inspection task in this dissertation.

The main idea behind the first version of MobileNet is to replace the convolutional layers by "depth-wise separable" convolutions to make the model computationally more efficient. The tasks of each convolutional layer can be summarized into two groups. As shown in

Figure 4.20, the first task is a depth-wise convolution layer for filtering the inputs. The second task is then to apply a 1×1 – commonly known as a point-wise convolution layer – for combining the filtered values of the input and create new features. The combination of the explained two tasks is called depth-wise separable convolution block that is capable of what traditional convolution layers did but in a much more efficient and faster manner (Howard et al., 2017). By means of depth-wise separable convolutions, MobileNets can have comparable accuracy results with the other larger neural networks but with about nine times less work which makes them feasible for mobile devices as well.

The first version of MobileNet is composed of a regular 3×3 convolution as its first layer which is followed by 13 times of the depth-wise separable convolutional block. In the architecture of this model, no pooling layer is used for spatial dimension reduction. In cases that spatial dimension reduction is needed, the depth-wise layer is applied with a stride of two. In contrary with other architectures, the activation function that is used in MobileNet is a specific version of ReLU which is called ReLU6 which is pretty much similar to the original activation but prevents is from becoming too large.

The second version of the MobileNet (MobileNetV2) still takes advantage of depth-wise separable convolutions, but it's main building block looks a little bit different. In the outline

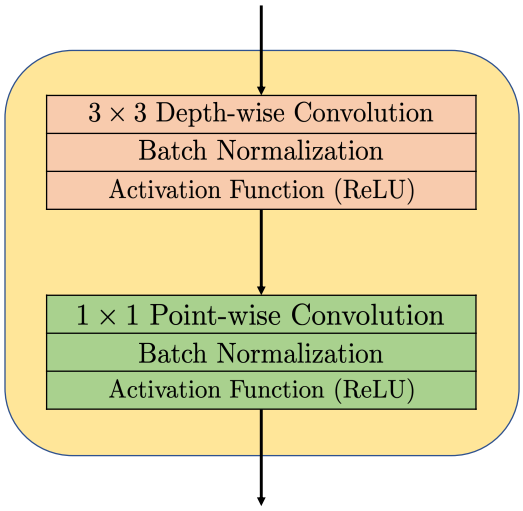


Figure 4.20: One Block of Depth-wise Separable Convolution

of the main block which is shown in Figure 4.21, the last two blocks are similar to the first version of MobileNet except the fact that in this latest model, the 1×1 convolution layer has a different task. In MobileNet, the 1×1 point-wise convolutional layer kept the number of channels the same or made them double. In contrary, in MobileNetV2, the 1×1 reduces the dimension (channels) and thus known as the "projection layer". Since the amount of the data that flows through the network is shrunk with the projection layer, this layer is also referred to as a "bottleneck layer".

In order to have a more efficient model with an acceptable accuracy level, in this section of the dissertation, MobileNetV2 was selected, and the results that are presented in the following parts are obtained from retraining this model. Initially, all the input images to the algorithm were labeled, and to make the sample size more extensive, the data was augmented. The images of the packed cases were simply rotated and flopped with the preprocessing function of Keras library.

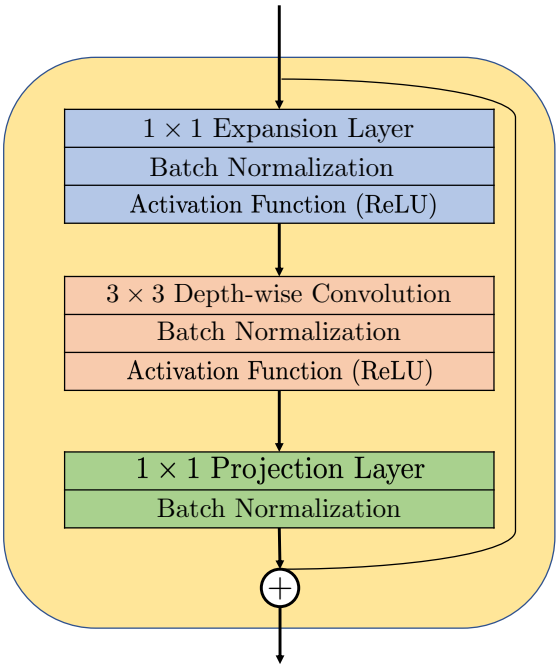


Figure 4.21: One Block of Bottleneck Residual Block

Results

In the case study presented in this dissertation, multi-label prediction enables us to say that the defect is detected in more than a single quadrant. Since the problem that we are solving is different from a traditional multi-class classification, another set of evaluation metrics are required as well. As it was discussed in Section 4.4.1, in single-label classification simple metrics such as accuracy, precision, recall, etc., were described. However in multi-label classification, we have to use a set of metrics that can evaluate the number of labels that were predicted correctly. Here, we use three different methods that can evaluate metrics for multi-label classification by somehow averaging the labels.

Lets assume L is the set of all labels we have which can be written as $L = \{\lambda_i : i = 1;\ ...;\ q\}$ and binary evaluations are calculated regarding each label. For instance, if a label is predicted correctly, it is written as $True\ Positive(\lambda_i)$ and D to denote a set of multi-label training examples that is written as $D = \{(x_j, Y_j) : j = 1;\ ...;\ m\}$. To be more precise, x_j is a feature vector and Y_j is the set of labels of the j^{th} example and $Y_j \in L$.

- **Micro Averaging**

  In micro-averaging method, we sum up the individual true positives, false positives, and false negatives of the system for different sets and then insert them in the formulas used for calculation of precision and recall.

$$Precision_{Micro\text{-}Averaged} = \frac{\sum_{\lambda_i \in L} True\ Positives(\lambda_i)}{\sum_{\lambda_i \in L} True\ Positives(\lambda_i) + False\ Positives(\lambda_i)} \tag{4.9}$$

$$Recall_{Micro\text{-}Averaged} = \frac{\sum_{\lambda_i \in L} True\ Positives(\lambda_i)}{\sum_{\lambda_i \in L} True\ Positives(\lambda_i) + False\ Negatives(\lambda_i)} \tag{4.10}$$

  The micro-averaged PRC curve of the multi-labeled classification is presented in Figure 4.22. In this plot, the precision and recall score are calculated with Equations 4.9, 4.10 which gives each sample-label pair an equal contribution to the overall metric. Typically, precision and recall are inversely related, for instance, when precision increases, recall falls and vice-versa. Therefore, a balance between these two needs to be achieved.

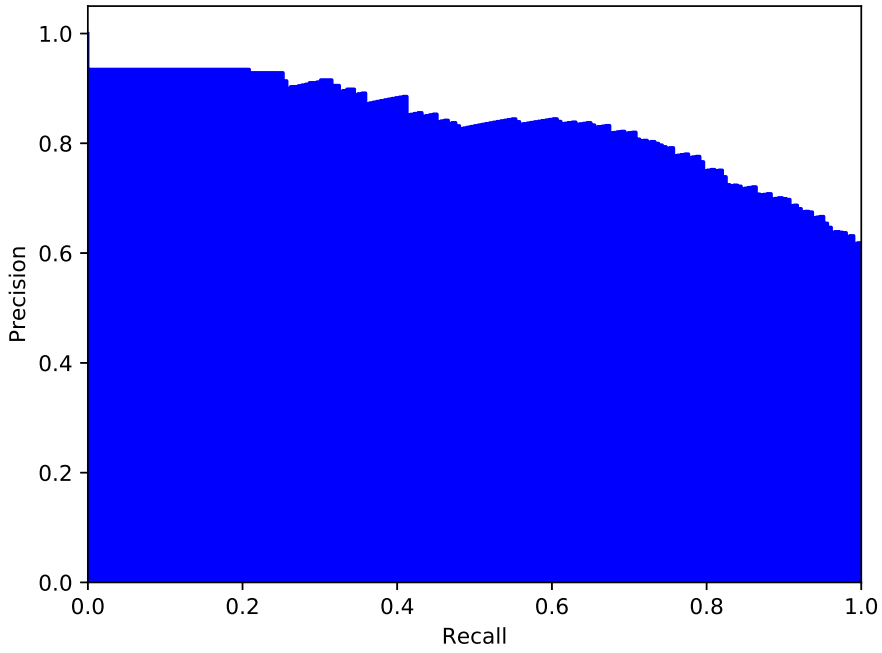


Figure 4.22: Micro-Averaged PRC

- **Macro Averaging**

  Macro-averaging is simple and straight forward. It is only required to take the average of the precision and recall of the system on different sets where $q = |L|$. The macro-averaging method is usually used to get an overall view of the system performance across the sets of data. Contrarily, micro-averaging is an informative measure when the dataset varies in size.

$$Precision_{Macro\text{-}Averaged} = \frac{\sum_{\lambda_i \in L} Precision(D, \lambda_i)}{q} \tag{4.11}$$

$$Recall_{Macro\text{-}Averaged} = \frac{\sum_{\lambda_i \in L} Recall(D, \lambda_i)}{q} \tag{4.12}$$

  The macro-averaged PRC curve of the multi-labeled classification is presented in Figure 4.23. In this plot, the precision and recall score are calculated with Equations 4.11, 4.12 which gives each sample-label pair an equal contribution to the overall metric.

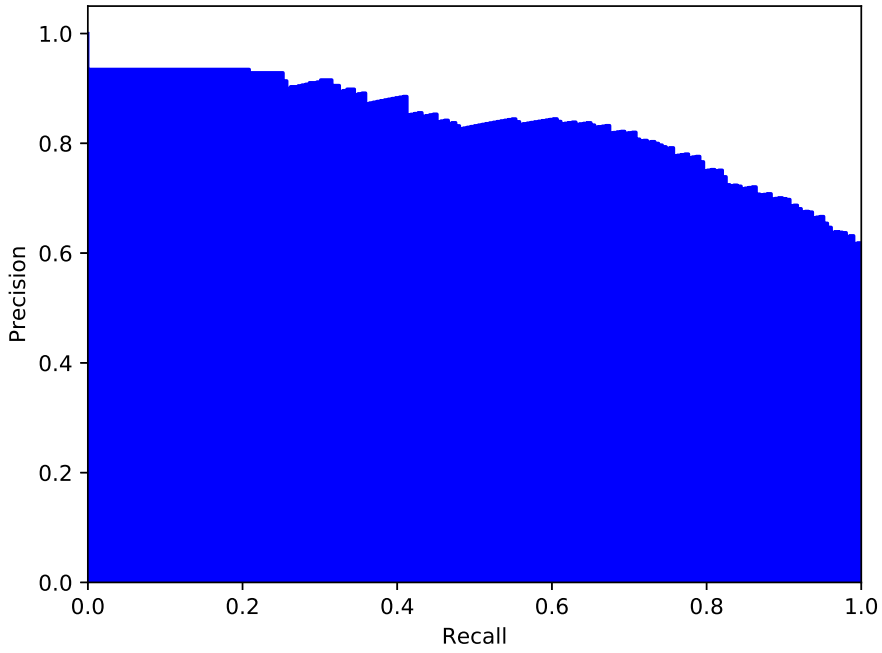


Figure 4.23: Macro-Averaged PRC

The performance of the classifiers are usually evaluated with the ROC curves, and for the experiments in this section, the results are reported in Figure 4.24. While ROC plots can be visually appealing and give an overview of the classifier's performance for a wide range of specificities, in situations with class imbalance, this plot could be misleading. Since making performance interpretation of the model could be deceptive, the performance evaluations are done based on the PRC plots. Unlike ROC curves, PRC plots can provide the decision maker with an accurate prediction of the future classification performance. The main reason for this is that PRC approach evaluates the fraction of true positives among positive predictions (Saito and Rehmsmeier, 2015).

- **F1 Score**

  Similar to the explanations in Section 4.4.1, to make a balance between precision and recall, F_1 score is calculated. The main difference between F_1 score and accuracy is that as explained previously, accuracy is largely contributed by a large number of true negatives. Thus, F_1 score is a more relevant metric to be used if having a balance between precision

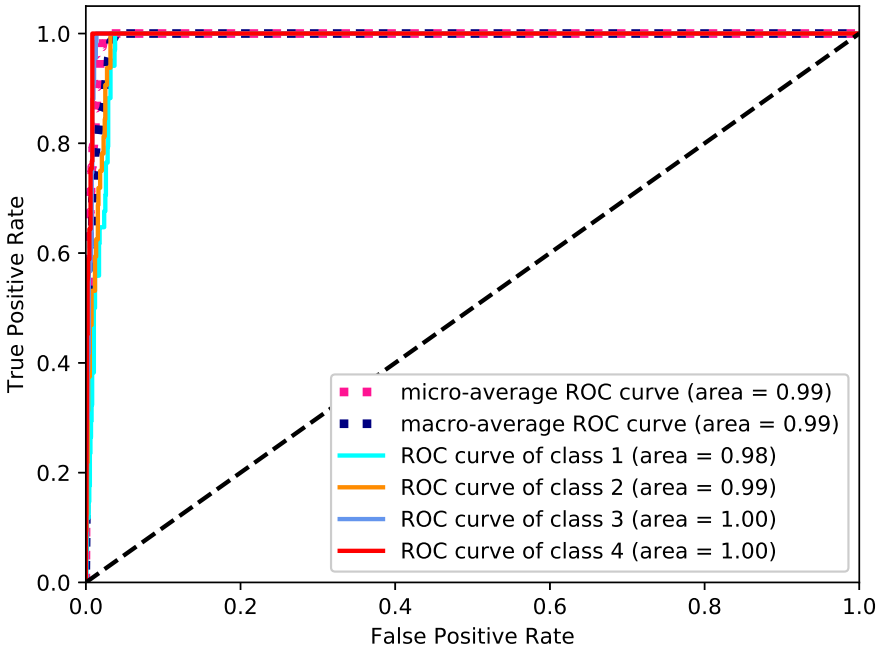


Figure 4.24: ROC Plot for Multi-Label Classification Task

and recall is sought. The F_1 score is simply the harmonic mean of the micro or macro-averaged precision and recall presented in Equations 4.9, 4.10, 4.11 , 4.12 respectively.

$$F_1 = 2 \times \frac{Precision_{Micro\text{-}Averaged} \times Recall_{Micro\text{-}Averaged}}{Precision_{Micro\text{-}Averaged} + Recall_{Micro\text{-}Averaged}} \tag{4.13}$$

$$F_1 = 2 \times \frac{Precision_{Macro\text{-}Averaged} \times Recall_{Macro\text{-}Averaged}}{Precision_{Macro\text{-}Averaged} + Recall_{Macro\text{-}Averaged}} \tag{4.14}$$

The micro and macro-averaged Iso-F_1 curves of the multi-labeled classification are presented in Figure 4.25 and 4.26 respectively. Each of these plots illustrates the area under the PRC for each of the labels associated with the samples. For instance, according to the micro-averaging method, the AUPRC for all the labels is 82% (just as an example the AUPRC for the fourth label is 80% which associated with the defects in the fourth quadrant of the case). The same interpretation is true for the other curves.

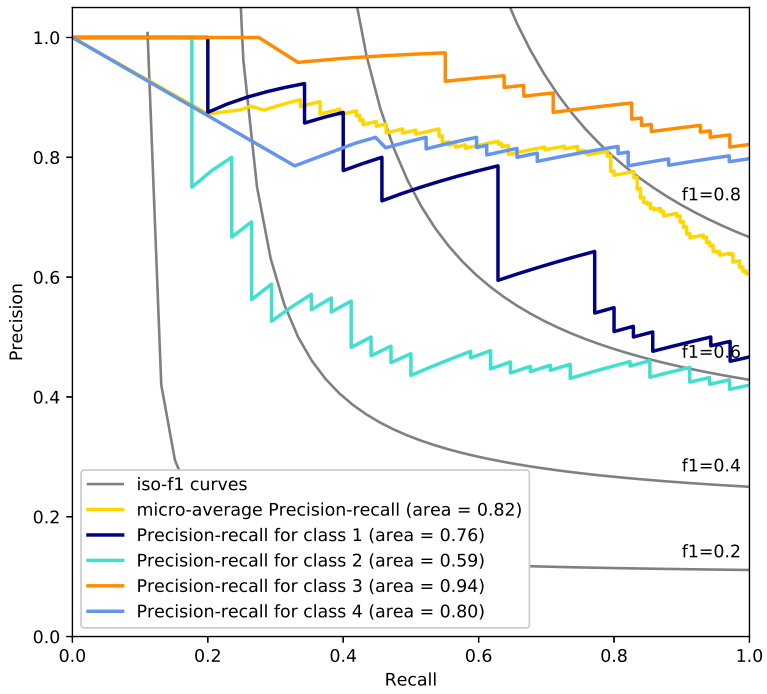


Figure 4.25: Micro-Averaged Iso-F_1 Curves

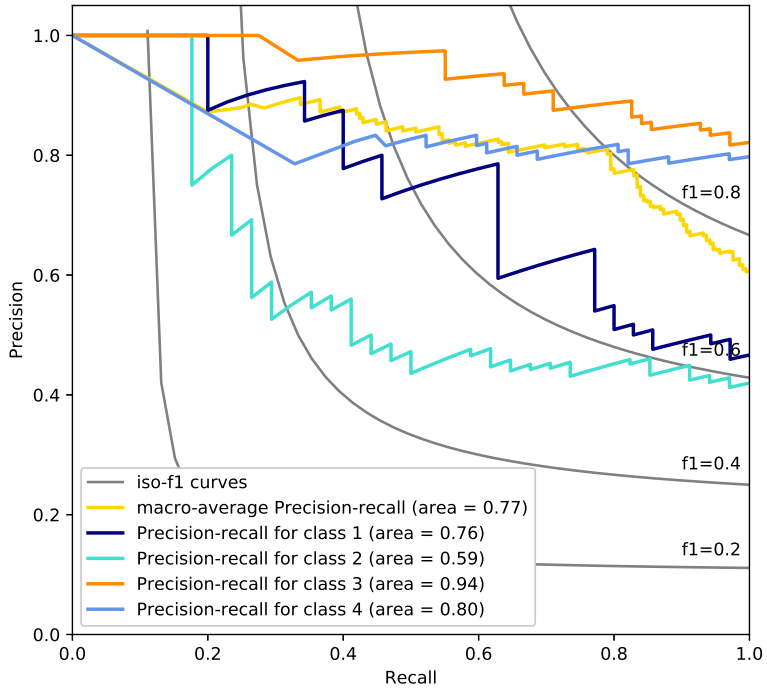


Figure 4.26: Macro-Averaged Iso-F_1 Curves

Lastly, the relationship between observing a defect in a specific quadrant of the packed case and the packing station that most likely caused it is presented. The quadrants of a case are shown in an image taken with the top-view camera (see Figure 4.27). It has to be mentioned that, the analysis is based on the assumption that if the deformation is observed in a quadrant, it is mainly caused by the layer adjacent to the case flaps. The assumption can be relaxed in the future by changing the inspection task. Depth cameras can be used in each station making sure that the SOP is followed correctly. It is expected that the defect will be detected in each station no matter if it is in the closest layer to the case flaps or at the bottom of the case.

In the case study presented in this dissertation, we focused on the line that produced case B as discussed in Section 4.4.2. The bags that are placed in this case are numbered from 13 to 24. The optimization task outputs the best placement strategy based on the dimensions of the bags that are measurable. Following the optimal results for one of the sample cases, the closest layer to the case flaps was identified from two perspectives; the bags that are located there as well as the stations that are in charge of putting them inside the case. As stated previously, three assemblers produce these cases, and each of them has their own process.

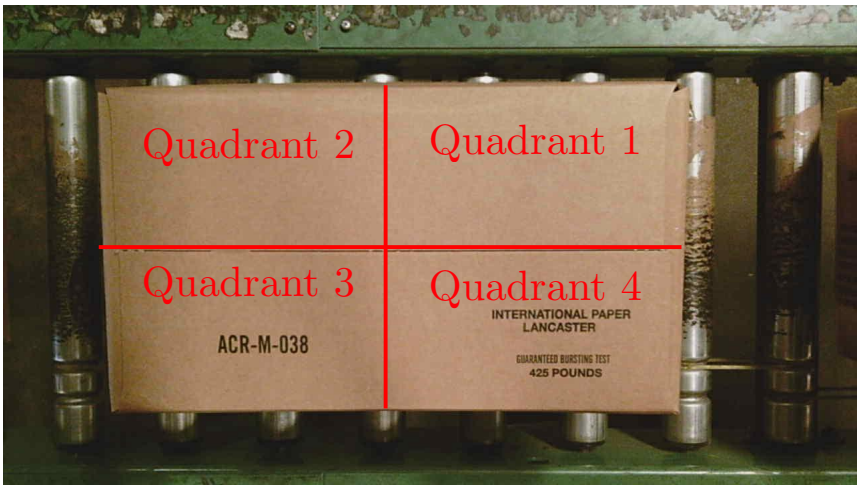


Figure 4.27: Labels Used for Localization of Defects in Cases

Two of these assemblers have six packing stations where the person is in charge of placing two bags inside the case. However, another assembler has 12 stations with one person in each, in charge of placing one bag inside the case. Since in this dissertation the data was collected from the latter assembler, the relationship between identifying a deformation in a quadrant of the case and its relevant bag/station is presented in Table 4.8.

The interpretation from the results that are presented in Table 4.8 is that since the defects can be localized, it can be associated with the step of packing which most likely created it. Finding this relationship can lead to having an accountable model where the quality issues can be related to their most likely causes. This model can be an example of having mistake proofing (Poka-Yoke) in the system which is making the errors visible once they occurred. Moreover, it notifies the person in charge that an error has happened and recommend them to follow the SOP to make sure that the chances of identifying the same mistake will be reduced in the long run.

Table 4.8: Relationship between Localized Deformation and Packing Step

Bulge detected in	Bag (s) and Station (s) that most likely caused the defect	
Quadrant 1	Bag 20	Station 7
Quadrant 2	Bag 17	Station 8
Quadrant 3	Bag 13	Station 11
Quadrant 4	Bag 16	Stations 12

Chapter 5

Conclusion and Future Work

In the framework of the Operational Excellence (OE), we found out that the main focus is on product development and no such effort is spent on the movement of the products. This misalignment between product development and packaging and transportation development will result in delivering items to the customers with lower quality than promised which could be a drawback in OE. Therefore, this dissertation aims to fill this gap in the OE framework.

In other words, the purpose of this study is to improve the quality of delivered items via a packing optimization and accountability model for a governmental organization. The proposed work is the first to develop such abilities based on flexible packaging containers. The proposed research will lead to improvements in the following manner:

1. An optimization algorithm is developed to utilize packaging space to avoid bulges and other deformations.

2. The correct sequence of packaging items is computed to prevent damage to delicate items.

3. The optimization model takes advantage of the moldable nature of bags by suggesting ways of folding bags before packaging, further improving space utilization.

4. A standard operating procedure is automatically generated using the optimization model, creating a standard common to all the users in different assemblers.

5. The SOP self-updates based on the changes of the items, improving the training regimen and reducing overall labor time invested in training.

6. An automated inspection system is developed using the convolutional neural network to detect packaging errors before a case is loaded on a pallet, saving quality costs in transportation logistics.

7. Inspection results are linked to specific packaging sequence errors, creating a platform for first-of-its-kind accountability model for the SOP.

This dissertation presents a new placement approach with a mathematical model when having items with flexible packages. The model could be used even in design stages to come up with the best size for bags among some possible scenarios. The model is an extension of Wu et al. (2010) paper when having flexible items. The presented results which are based on a case study revealed that by strategically using the flexibility by foldings, we could reduce the occupied space in cases. Since one option for folding of bags is to shrink them to the minimal size by folding it from both sides, the results of our model are compared with the results of the model with consideration of the most compact bag dimensions. The results show that our model requires less number of folding and leads to the smallest case height by minimal effort. To make sure that the results are usable to the end users, an interface was developed which requires the dimension of the items in each folding scenario. The results of the model are then visualized and presented as standard operating procedure videos (SOP) to the workers of the line. Whenever a change happens regarding the size of bags, the inputs are changed, and new SOPs are generated within few minutes.

As part of validating the results of the presented model and having an accountable model, a deep learning based inspection is presented which can automatically detect any bulges in the final packed case with convolutional neural network algorithm. In the event of having some bulged instances, the location of the swelling is identified and gets connected to a step of optimization that most likely caused it. The station that was in charge of creating this bulge will be retrained with the hope that the learning curve improves over time. The main reason for doing this is that foldings could be hard for the workers of the line and adds more complexity to the usual activities they have. By having a standard operating procedure, the

risk of having defects in the system will be reduced, peoples' quality of life will be improved, and the system reacts in a more agile way to any alternations.

We did not consider computational complexities in this study, however the scalability of the model is tested. Developing a heuristic method or an exact solution strategy for the times that the number of items is more substantial than the case presented in this paper may be a proper direction for future research. Usage of depth-camera instead of normal cameras will make the model more realistic.

Bibliography

Alessi, C. and Gummer, C. (2014). Germany bets on 'smart factories' to keep its manufacturing edge. *The Wall Street Journal*. 4

Allen, S. D., Burke, E. K., and Kendall, G. (2011). A hybrid placement strategy for the three-dimensional strip packing problem. *European Journal of Operational Research*, 209(3):219–227. 21, 29

Araya, I., Guerrero, K., and Nuñez, E. (2017). Vcs: A new heuristic function for selecting boxes in the single container loading problem. *Computers & Operations Research*, 82:27–35. 22

Araya, I. and Riff, M.-C. (2014). A beam search approach to the container loading problem. *Computers & Operations Research*, 43:100–107. 29

Bansal, N., Han, X., Iwama, K., Sviridenko, M., and Zhang, G. (2007). Harmonic algorithm for 3-dimensional strip packing problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1197–1206. Society for Industrial and Applied Mathematics. 29

Basu, R. (2004). *Implementing quality: a practical guide to tools and techniques: enabling the power of operational excellence*. Cengage Learning EMEA. 3

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828. 26

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160. 26

Birgin, E. G. and Sobral, F. (2008). Minimizing the object dimensions in circle and sphere packing problems. *Computers & Operations Research*, 35(7):2357–2375. 18, 29

Bischoff, E. and Dowsland, W. (1982). An application of the micro to product design and distribution. *Journal of the Operational Research Society*, 33(3):271–280. 19

Bischoff, E. and Ratcliff, M. (1995a). Loading multiple pallets. *Journal of the Operational Research Society*, 46(11):1322–1336. 20

Bischoff, E. E. and Marriott, M. D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44(2):267–276. 19, 29

Bischoff, E. E. and Ratcliff, M. (1995b). Issues in the development of approaches to container loading. *Omega*, 23(4):377–390. 19, 21

Booth, D., Earl, T., and Mobini, S. (2003). Perceptual channels for the texture of a food. *Appetite*, 40(1):69 – 76. 3

Bortfeldt, A. (2000). Eine heuristik für multiple containerladeprobleme. *OR-Spektrum*, 22(2):239–261. 18

Bortfeldt, A. and Gehring, H. (1999). Two metaheuristics for strip packing problems. In *Erscheint in: Proceedingsband der 5th International Conference of the Decision Sciences Institute, Athen*. 29

Bortfeldt, A. and Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161. 22

Bortfeldt, A. and Mack, D. (2007). A heuristic for the three-dimensional strip packing problem. *European Journal of Operational Research*, 183(3):1267–1279. 21, 29

Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20. 14, 15

Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. (2004). Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771. 57

Burke, R., Mussomeli, A., Laaper, S., Hartigan, M., and Sniderman, B. (2017). The smart factory: Responsive, adaptive, connected manufacturing. *Deloitte Insights, August*, 31. 3, 4

Can, O. and Sahingoz, O. K. (2014). Solving container loading problem with simulated annealing algorithm. In *Computational Intelligence and Informatics (CINTI), 2014 IEEE 15th International Symposium on*, pages 379–383. IEEE. 24

Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*. xi, 80

Che, C. H., Huang, W., Lim, A., and Zhu, W. (2011). The multiple container loading cost minimization problem. *European Journal of Operational Research*, 214(3):501–511. 19

Chen, C., Lee, S.-M., and Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76. 17

Chien, C.-F. and Deng, J.-F. (2004). A container packing support system for determining and visualizing container packing patterns. *Decision Support Systems*, 37(1):23–34. 20

Chien, C.-F. and Wu, W.-T. (1998). A recursive computational procedure for container loading. *Computers & Industrial Engineering*, 35(1-2):319–322. 20

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras. ix, 81, 91

Conover, W. J. and Conover, W. J. (1980). Practical nonparametric statistics. 19

Corcoran III, A. L. and Wainwright, R. L. (1992). A genetic algorithm for packing in three dimensions. In *Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing: technological challenges of the 1990's*, pages 1021–1030. ACM. 22, 29

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941. 26

De Queiroz, T. A., Miyazawa, F. K., Wakabayashi, Y., and Xavier, E. C. (2012). Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers & Operations Research*, 39(2):200–212. 19, 29

Dereli, T. and Das, G. S. (2011). A hybrid 'bee (s) algorithm'for solving container loading problems. *Applied Soft Computing*, 11(2):2854–2862. 23

Dowsland, K. A. and Dowsland, W. B. (1992). Packing problems. *European journal of operational research*, 56(1):2–14. 14

Dowsland, W. B. (1984). *Two and three dimensional packing problems and solution methods*. Operational Research Society of New Zeland. 19

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159. 14

Egeblad, J., Nielsen, B. K., and Brazil, M. (2009). Translational packing of arbitrary polytopes. *Computational Geometry*, 42(4):269–288. 29

Egeblad, J., Nielsen, B. K., and Odgaard, A. (2007). Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 183(3):1249–1266. 23, 29

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25(1):45–60. 17, 18

Faina, L. (2000). A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research*, 126(2):340–354. 17, 29

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929. 27

Faroe, O., Pisinger, D., and Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *Informs journal on computing*, 15(3):267–283. 20

Flynn, B. B., Schroeder, R. G., and Sakakibara, S. (1995). The impact of quality management practices on performance and competitive advantage. *Decision sciences*, 26(5):659–691. 3

Fujiyoshi, K., Kawai, H., and Ishihara, K. (2009). A tree based novel representation for 3d-block packing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(5):759–764. 29

Gehring, H. and Bortfeldt, A. (2002). A parallel genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 9(4):497–511. 23

Gehring, H., Menschner, K., and Meyer, M. (1990). A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44(2):277–288. 23

George, J. A. and Robinson, D. F. (1980). A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147–156. 19, 20

Hall, A. J. (2011). Gauge repeatability and reproducibility study for a bulge test fixture for corrugated boxes. 3

Hasni, H. and Sabri, H. (2013). On a hybrid genetic algorithm for solving the container loading problem with no orientation constraints. *Journal of Mathematical Modelling and Algorithms*, pages 1–18. 29

He, K. and Huang, W. (2011). An efficient placement heuristic for three-dimensional rectangular packing. *Computers & Operations Research*, 38(1):227–233. 29

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*, pages 770–778. 80, 81

He, Y., Wu, Y., and de Souza, R. (2012). A global search framework for practical three-dimensional packing with variable carton orientations. *Computers & Operations Research*, 39(10):2395–2414. 29

Hessman, T. (2013). The dawn of the smart factory. *Industry Week*, 14:14–19. 4

Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, page 14. 48

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554. 26

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. 92

Huang, S.-H. and Pan, Y.-C. (2015). Automated visual inspection in the semiconductor industry: A survey. *Computers in industry*, 66:1–10. 25

Huang, Y.-H., Hwang, F., and Lu, H.-C. (2016). An effective placement method for the single container loading problem. *Computers & Industrial Engineering*, 97:212–221. 29

Ivancic, N. J. (1988). *An integer programming based heuristic approach to the three dimensional packing problem*. PhD thesis, Case Western Reserve University. 18

Jain, R., Kasturi, R., and Schunck, B. G. (1995). *Machine vision*, volume 5. McGraw-Hill New York. 25

Jansen, K. and Solis-Oba, R. (2006). An asymptotic approximation algorithm for 3d-strip packing. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 143–152. Society for Industrial and Applied Mathematics. 29

Junqueira, L., Morabito, R., and Yamashita, D. S. (2012). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85. 18, 29

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 51

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 27, 40

Lai, K., Xue, J., and Xu, B. (1998). Container packing in a multi-customer delivering operation. *Computers & industrial engineering*, 35(1-2):323–326. 20, 29

Langston, M. A. (1987). A study of composite heuristic algorithms. *Journal of the Operational Research Society*, 38(6):539–544. 19

Leblanc, R. (2015). Reducing product damage in the supply chain – here's how. 3

LeCun, Y., Bengio, Y., and Hinton, G. (2015a). Deep learning. *nature*, 521(7553):436. 25

LeCun, Y. et al. (2015b). Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, page 20. 27

Li, H.-L., Tsai, J.-F., and Hu, N.-Z. (2003). A distributed global optimization method for packing problems. *Journal of the Operational Research Society*, 54(4):419–425. 18, 29

Li, K. and Cheng, K.-H. (1990). On three-dimensional packing. *SIAM Journal on Computing*, 19(5):847–867. 19, 29

Li, K. and Cheng, K. H. (1992). Heuristic algorithms for on-line packing in three dimensions. *Journal of Algorithms*, 13(4):589–605. 29

Li, T. and Ogihara, M. (2003). Detecting emotion in music. 57

Li, X. and Zhang, K. (2015). A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers. *Computers & Industrial Engineering*, 90:305–313. 29

Liang, S.-C., Lee, C.-Y., and Huang, S.-W. (2007). A hybrid meta-heuristic for the container loading problem. *Communications of the IIMA*, 7(4):9. 23

Lim, A., Ma, H., Xu, J., and Zhang, X. (2012). An iterated construction approach with dynamic prioritization for solving the container loading problems. *Expert Systems with Applications*, 39(4):4292–4305. 21

Liu, J., Yue, Y., Dong, Z., Maple, C., and Keech, M. (2011). A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 38(4):797–807. 21

Liu, S., Tan, W., Xu, Z., and Liu, X. (2014). A tree search algorithm for the container loading problem. *Computers & Industrial Engineering*, 75:20–30. 22

Lodi, A., Martello, S., and Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420. 20

Loh, T. (1992). A packing algorithm for hexahedral boxes. In *Proceedings of the Conference of Industrial Automation, Singapore*, pages 115–126. 21

Maleki, S. (2013). *Moldable Items Packing Optimization*. PhD thesis, University of Tennessee, Knoxville. 24

Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267. 17, 18, 20

Martello, S., Pisinger, D., Vigo, D., Boef, E. D., and Korst, J. (2007). Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software (TOMS)*, 33(1):7. 18

Matsugu, M., Mori, K., Mitari, Y., and Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559. 10, 27

Miyazawa, F. K. and Wakabayashi, Y. (1997). An algorithm for the three-dimensional packing problem with asymptotic performance analysis. *Algorithmica*, 18(1):122–144. 29

Miyazawa, F. K. and Wakabayashi, Y. (2007). Two-and three-dimensional parametric packing. *Computers & operations research*, 34(9):2589–2603. 29

Miyazawa, F. K. and Wakabayashi, Y. (2009). Three-dimensional packings with rotations. *Computers & Operations Research*, 36(10):2801–2815. 18, 29

Mostaghimi Ghomi, H., St Amour, B. G., and Abdul-Kader, W. (2017). Three-dimensional container loading: A simulated annealing approach. *International Journal of Applied Engineering Research*, 12(7):1290. 29

Moura, A. and Oliveira, J. F. (2005). A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 20(4):50–57. 20, 22

Mukhacheva, E. and Shehtman, I. (1997). Decomposition method of two-and-three-dimensional rectangular bin packing. *Decision Making under Conditions of Uncertainty (Cutting–Packing Problems), Ufa State Aviation Technical University, Ufa*, pages 155–171. 29

Neethu, N. and Anoop, B. (2015). Role of computer vision in automatic inspection systems. *International Journal of Computer Applications*, 123(13). 25

Neil-Boss, N. and Brooks, K. (2012). Unwrapping the packaging industry. *Ernst & Young Global Limited*. 2

Oakland, J. S. (2014). *Total quality management and operational excellence: text with cases*. Routledge. 3

Optimization, G. (2017). Gurobi optimizer version 7.0. 2. 39, 62

Orloski, A. (2005). Liz claiborne upgrades packaging at center. 3, 4, 5

Pham, D., Koç, E., and Ghanbarzadeh, A. (2006). Optimization of the weights of multi-layered perceptions using the bees algorithm. PROCEEDINGS OF INTERNATIONAL SYMPOSIUM ON INTELLIGENT MANUFACTURING SYSTEMS. 24

Pisinger, D. (2002). Heuristics for the container loading problem. *European journal of operational research*, 141(2):382–392. 20

Prinz, C., Morlock, F., Freith, S., Kreggenfeld, N., Kreimeier, D., and Kuhlenkötter, B. (2016). Learning factory modules for smart factories in industrie 4.0. *Procedia CiRp*, 54:113–118. 3

Provost, F. J., Fawcett, T., Kohavi, R., et al. (1998). The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453. 72, 74

Ramos, A. G., Oliveira, J. F., Gonçalves, J. F., and Lopes, M. P. (2016). A container loading algorithm with static mechanical equilibrium stability constraints. *Transportation Research Part B: Methodological*, 91:565–581. 24

Ratcliff, M. and Bischoff, E. E. (1998). Allowing for weight considerations in container loading. *Operations-Research-Spektrum*, 20(1):65–71. 20

Ren, J., Tian, Y., and Sawaragi, T. (2011). A tree search method for the container loading problem with shipment priority. *European Journal of Operational Research*, 214(3):526–535. 21

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. 41

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432. 96

Scheithauer, G. (1991). A three-dimensional bin packing algorithm. *Elektronische Informationsverarbeitung und Kybernetik*, 27(5/6):263–271. 29

Sheng, L., Xiuqin, S., Changjian, C., Hongxia, Z., Dayong, S., and Feiyue, W. (2017). Heuristic algorithm for the container loading problem with multiple constraints. *Computers & Industrial Engineering*, 108:149–164. 22

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE. 27

Stoyan, Y. Y., Yaskow, G., and Scheithauer, G. (2003). *Packing of various radii solid spheres into a parallelepiped*. PhD thesis. 18, 29

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9. 27

Takahara, S. (2006). A simple meta-heuristic approach for the multiple container loading problem. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 3, pages 2328–2333. IEEE. 23

Takahara, S. and Miyamoto, S. (2005). An evolutionary approach for the multiple container loading problem. In *Hybrid Intelligent Systems, 2005. HIS'05. Fifth International Conference on*, pages 6–pp. IEEE. 23

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*. 49

Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807. 27

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13. 57

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European journal of operational research*, 183(3):1109–1130. 14

Wu, Y., Li, W., Goh, M., and de Souza, R. (2010). Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, 202(2):347–355. 23, 32, 62, 102

Yeung, L. H. W. and Tang, W. K.-S. (2005). A hybrid genetic approach for container loading in logistics industry. *IEEE Transactions on industrial electronics*, 52(2):617–627. 29

Zhang, D., Peng, Y., and Leung, S. C. (2012). A heuristic block-loading algorithm based on multi-layer search for the container loading problem. *Computers & Operations Research*, 39(10):2267–2276. 21

Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *Granular Computing, 2005 IEEE International Conference on*, volume 2, pages 718–721. IEEE. 57

Zhao, X., Bennell, J. A., Bektaş, T., and Dowsland, K. (2016). A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320. 14, 18

Zheng, J.-N., Chien, C.-F., and Gen, M. (2015). Multi-objective multi-population biased random-key genetic algorithm for the 3-d container loading problem. *Computers & Industrial Engineering*, 89:80–87. 24

Zhu, W., Huang, W., and Lim, A. (2012). A prototype column generation strategy for the multiple container loading problem. *European Journal of Operational Research*, 223(1):27–39. 19

Zuehlke, D. (2010). Smartfactory—towards a factory-of-things. *Annual reviews in control*, 34(1):129–138. 4

Appendices

A Summary of Input Data

The dimension of the bags are presented in a tabular format for all the sample cases presented in this paper in the following.

Table A.1: Sample Case 1- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.76	6.06	2.03	8.91	6.49	2.31	10.90	3.58	2.60	7.69	4.44	2.97
10.97	6.03	2.23	9.16	6.32	2.57	10.97	3.95	2.44	8.21	4.69	2.97
10.78	6.15	2.35	7.23	6.22	2.35	10.94	3.68	2.50	8.11	4.49	2.93
10.68	6.10	2.52	8.17	6.14	2.52	11.08	3.94	2.52	8.27	4.77	3.08
11.03	6.08	2.19	8.66	6.08	2.40	11.03	3.69	2.48	7.99	4.88	3.01
10.98	6.06	2.55	7.20	6.17	2.55	11.14	3.80	2.59	7.16	4.50	2.88
10.70	6.07	2.66	8.80	6.45	2.66	10.77	3.77	2.66	9.12	4.70	3.07
10.99	6.07	2.12	8.43	6.16	2.29	10.99	4.17	2.26	8.60	4.67	2.93
11.00	6.08	2.27	8.24	6.21	2.43	11.00	3.60	2.32	7.71	4.79	3.04
11.03	6.06	2.23	8.11	6.29	2.54	11.13	3.70	2.37	7.20	4.57	3.02
10.73	6.21	2.64	7.15	6.30	2.64	10.73	3.66	2.64	7.38	4.69	2.93
10.74	6.09	2.41	9.26	6.29	2.42	10.74	4.05	2.41	8.61	4.45	2.95

Table A.2: Sample Case 2- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.74	6.11	2.26	8.51	6.48	2.61	10.74	4.18	2.37	8.68	4.45	2.90
10.75	6.20	2.55	7.11	6.39	2.55	10.96	3.67	2.55	7.01	4.66	2.89
10.78	6.04	2.07	7.74	6.05	2.31	10.78	3.69	2.41	7.42	4.64	3.04
11.05	6.12	2.43	9.31	6.19	2.43	11.05	3.86	2.57	7.42	4.44	2.93
10.86	6.21	1.97	8.80	6.24	2.28	11.04	3.85	2.46	7.76	4.69	2.82
11.11	6.17	2.19	7.25	6.34	2.50	11.11	4.08	2.46	8.90	4.78	2.83
10.73	6.09	2.63	8.92	6.39	2.63	10.97	3.52	2.63	9.13	4.69	3.01
11.15	6.14	1.98	7.37	6.41	2.52	11.15	4.05	2.40	8.89	4.43	2.83
11.03	6.20	2.02	8.02	6.20	2.34	11.03	4.11	2.32	7.88	4.60	2.83
10.82	6.11	2.12	9.18	6.11	2.64	10.96	3.75	2.33	7.30	4.47	2.99
10.68	6.09	2.66	8.34	6.09	2.66	11.16	4.03	2.66	7.04	4.43	2.93
11.00	6.14	2.31	7.13	6.14	2.44	11.00	3.70	2.38	8.22	4.81	2.93

Table A.3: Sample Case 3- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
11.09	6.09	2.05	11.09	3.82	2.39	10.74	4.18	2.37	8.34	4.82	2.96
10.87	6.06	2.45	10.87	4.02	2.45	10.96	3.67	2.55	9.12	4.86	3.07
11.10	6.20	2.60	11.10	3.83	2.60	10.78	3.69	2.41	8.66	4.83	2.93
11.06	6.06	2.36	11.06	3.86	2.36	11.05	3.86	2.57	8.64	4.81	2.92
10.69	6.14	2.37	10.69	3.79	2.37	11.04	3.85	2.46	7.21	4.48	2.91
11.13	6.22	2.33	11.13	4.07	2.59	11.11	4.08	2.46	7.71	4.52	2.97
10.91	6.13	2.43	10.91	4.20	2.47	10.97	3.52	2.63	9.00	4.74	2.84
10.91	6.19	2.38	10.91	3.84	2.38	11.15	4.05	2.40	8.02	4.72	2.99
10.77	6.16	2.02	10.86	3.72	2.36	11.03	4.11	2.32	6.91	4.46	3.03
11.00	6.18	2.08	11.00	3.98	2.54	10.96	3.75	2.33	7.91	4.69	2.86
10.93	6.03	2.65	10.93	4.06	2.65	11.16	4.03	2.66	6.91	4.85	2.95
10.96	6.14	2.66	10.96	3.94	2.66	11.00	3.70	2.38	6.89	4.48	2.85

Table A.4: Sample Case 4- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.68	6.19	2.61	8.29	6.32	2.61	10.94	4.01	2.61	7.85	4.68	2.85
10.79	6.14	2.49	7.51	6.39	2.49	10.79	3.77	2.49	7.74	4.87	2.82
10.77	6.22	2.60	8.29	6.22	2.60	11.08	3.81	2.60	8.59	4.86	2.91
10.82	6.20	2.42	7.88	6.20	2.42	10.82	3.74	2.55	7.51	4.52	2.92
11.11	6.17	1.98	8.24	6.45	2.50	11.11	4.20	2.32	8.40	4.60	2.98
11.12	6.05	2.03	8.88	6.21	2.32	11.12	4.01	2.32	8.69	4.82	2.95
10.91	6.06	2.63	8.09	6.38	2.63	10.91	4.15	2.63	8.55	4.74	2.91
10.75	6.09	2.59	7.74	6.09	2.59	10.75	4.18	2.59	9.17	4.57	2.90
10.76	6.23	2.29	9.20	6.23	2.47	11.03	4.05	2.29	8.13	4.74	2.97
10.76	6.21	2.52	7.17	6.28	2.52	10.76	4.11	2.62	8.40	4.49	2.83
10.91	6.13	2.11	9.01	6.16	2.24	10.91	4.09	2.26	7.50	4.83	2.92
10.71	6.07	2.19	8.09	6.23	2.40	10.74	4.15	2.44	7.78	4.83	2.89

Table A.5: Sample Case 5- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
11.12	6.19	2.27	8.28	6.29	2.36	11.12	3.77	2.58	7.81	4.54	2.92
10.81	6.07	2.42	7.96	6.07	2.44	10.99	3.65	2.42	9.16	4.45	2.97
10.84	6.20	2.14	7.59	6.33	2.25	11.13	3.98	2.57	7.19	4.62	2.89
11.13	6.12	2.17	7.21	6.18	2.58	11.13	3.69	2.61	8.95	4.77	3.05
10.87	6.09	2.62	7.86	6.12	2.62	10.87	4.16	2.62	7.31	4.48	3.08
10.94	6.23	2.06	8.28	6.46	2.64	10.94	3.76	2.57	8.31	4.87	3.07
11.05	6.18	2.30	7.91	6.18	2.42	11.16	4.17	2.31	8.72	4.79	3.08
10.90	6.12	2.53	9.33	6.22	2.61	10.90	3.87	2.54	8.58	4.87	2.99
11.09	6.08	2.20	7.20	6.49	2.22	11.10	3.93	2.42	8.62	4.44	2.90
11.01	6.10	2.35	7.26	6.20	2.35	11.01	4.04	2.61	8.08	4.68	3.05
10.73	6.08	2.00	8.62	6.46	2.38	10.83	3.68	2.49	8.33	4.60	3.00
10.98	6.18	2.45	7.46	6.18	2.45	10.98	3.74	2.45	7.92	4.63	2.95

Table A.6: Sample Case 6- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.79	4.21	2.53	9.26	6.22	2.36	11.12	3.77	2.58	8.95	4.45	3.05
11.09	3.67	2.35	8.51	6.20	2.31	10.99	3.65	2.42	7.76	4.63	2.88
11.09	3.69	2.48	9.00	6.04	2.58	11.13	3.98	2.57	7.08	4.58	3.02
10.71	3.64	2.31	7.37	6.15	2.42	11.13	3.69	2.61	8.75	4.80	3.01
10.78	4.08	2.32	8.35	6.38	2.36	10.87	4.16	2.62	7.13	4.66	2.84
11.08	3.59	2.29	8.84	6.40	2.28	10.94	3.76	2.57	8.65	4.62	2.94
11.07	3.85	2.36	7.68	6.11	2.59	11.16	4.17	2.31	7.14	4.45	2.85
10.84	3.58	2.42	8.22	6.46	2.25	10.90	3.87	2.54	8.86	4.53	2.89
10.90	4.18	2.57	8.41	6.06	2.57	11.10	3.93	2.42	8.11	4.60	2.88
11.04	3.66	2.39	8.19	6.44	2.56	11.01	4.04	2.61	8.25	4.47	2.86
10.90	4.03	2.49	8.47	6.26	2.49	10.83	3.68	2.49	8.88	4.61	2.82
10.90	4.20	2.48	7.14	6.07	2.44	10.98	3.74	2.45	8.89	4.53	3.07

Table A.7: Sample Case 7- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.87	6.09	2.29	7.14	6.31	2.43	10.87	3.62	2.32	8.94	4.43	3.01
10.79	6.12	2.41	8.39	6.42	2.62	10.79	3.89	2.60	7.39	4.48	2.89
10.72	6.16	2.27	8.24	6.16	2.27	11.03	3.84	2.63	8.14	4.73	2.99
11.08	6.13	2.64	7.75	6.39	2.64	11.08	3.89	2.64	7.34	4.45	3.08
10.74	6.02	2.16	7.72	6.04	2.53	10.79	3.58	2.44	8.11	4.88	2.96
10.93	6.20	2.65	8.46	6.26	2.65	10.93	3.94	2.65	9.13	4.47	2.93
10.91	6.14	2.44	7.97	6.43	2.49	10.91	4.01	2.49	8.74	4.71	3.01
10.82	6.21	2.60	7.75	6.49	2.60	10.90	3.91	2.60	7.33	4.43	2.86
11.00	6.22	2.43	7.96	6.22	2.43	11.10	3.96	2.43	7.79	4.45	2.90
11.03	6.21	2.26	8.94	6.24	2.51	11.03	3.73	2.52	8.80	4.51	2.92
11.13	6.17	2.62	7.62	6.21	2.62	11.14	3.72	2.62	7.89	4.79	2.83
10.84	6.19	2.07	9.31	6.46	2.26	10.84	3.95	2.51	7.78	4.80	2.89

Table A.8: Sample Case 8- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.90	6.19	2.17	7.11	6.47	2.59	11.02	4.00	2.31	9.10	4.44	3.04
10.75	6.21	2.37	9.27	6.21	2.59	11.13	3.58	2.51	8.47	4.60	3.00
10.86	6.17	2.07	7.86	6.21	2.48	10.99	4.11	2.58	6.87	4.56	2.90
11.06	6.12	2.31	7.16	6.27	2.31	11.06	3.54	2.39	9.11	4.62	3.05
11.12	6.04	2.22	7.90	6.14	2.36	11.12	3.60	2.27	8.02	4.58	3.04
10.75	6.17	2.25	8.54	6.21	2.59	11.07	3.87	2.45	8.12	4.76	2.92
10.84	6.21	2.16	7.74	6.22	2.44	11.05	4.00	2.30	6.99	4.86	3.05
11.01	6.03	2.26	9.31	6.17	2.61	11.01	3.95	2.59	7.23	4.52	2.96
11.13	6.05	2.29	7.28	6.05	2.64	11.13	4.16	2.34	8.64	4.80	2.97
10.87	6.10	2.55	7.38	6.48	2.55	10.87	3.74	2.55	7.61	4.43	2.82
10.73	6.05	2.22	7.69	6.06	2.22	11.06	3.91	2.39	8.31	4.85	3.06
11.06	6.05	2.27	7.58	6.28	2.39	11.06	4.04	2.55	8.25	4.87	2.99

Table A.9: Sample Case 9- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.89	6.17	2.44	8.82	6.43	2.44	10.90	4.19	2.44	7.56	4.87	3.02
11.10	6.13	2.54	7.12	6.13	2.54	11.10	4.00	2.54	7.23	4.48	3.01
10.95	6.08	2.22	7.62	6.25	2.45	10.95	3.53	2.48	8.66	4.75	3.02
10.70	6.15	2.40	8.67	6.24	2.41	10.79	4.12	2.46	9.18	4.66	2.99
10.73	6.23	2.21	9.13	6.27	2.56	10.73	3.89	2.47	8.87	4.86	3.04
10.74	6.13	2.49	7.30	6.47	2.49	10.74	3.82	2.54	8.44	4.75	2.93
11.08	6.21	2.36	8.59	6.27	2.37	11.08	4.13	2.52	8.76	4.84	3.07
11.13	6.06	2.40	8.54	6.27	2.60	11.15	3.94	2.57	7.99	4.56	2.89
11.10	6.12	2.24	9.20	6.32	2.44	11.10	3.57	2.62	7.93	4.71	2.85
10.72	6.15	2.43	9.13	6.28	2.43	10.72	3.62	2.43	7.58	4.52	2.98
11.05	6.06	2.44	7.16	6.06	2.44	11.12	3.81	2.58	6.90	4.81	2.82
10.67	6.11	2.51	7.83	6.15	2.51	10.67	4.11	2.51	7.82	4.44	2.84

Table A.10: Sample Case 10- Bag Dimensions in Different Scenarios of Folding

No Fold			Length-wise Fold			Width-wise Fold			All Folds		
l	w	h	l	w	h	l	w	h	l	w	h
10.97	6.13	1.99	8.78	6.34	2.44	11.14	4.20	2.45	6.91	4.68	3.03
10.97	6.06	1.99	9.31	6.06	2.32	11.02	4.06	2.57	8.91	4.64	2.95
10.69	6.09	2.36	7.71	6.11	2.62	10.69	4.11	2.42	9.05	4.51	2.96
10.68	6.08	2.59	7.41	6.46	2.59	11.13	3.86	2.59	9.04	4.51	2.82
10.70	6.20	2.35	9.29	6.48	2.49	10.83	4.03	2.59	6.96	4.80	3.03
10.81	6.22	2.10	7.79	6.22	2.64	10.90	4.15	2.37	8.82	4.83	2.91
10.73	6.12	2.52	8.38	6.21	2.52	11.15	3.57	2.52	8.82	4.68	2.94
11.08	6.11	2.63	8.31	6.40	2.63	11.08	4.16	2.63	8.17	4.59	3.01
10.95	6.08	2.02	9.26	6.44	2.28	10.95	3.54	2.33	8.93	4.53	3.02
10.93	6.08	2.23	8.11	6.48	2.37	10.93	3.75	2.26	8.70	4.60	3.06
10.94	6.19	2.43	7.95	6.32	2.43	11.09	3.61	2.43	8.98	4.68	2.90
10.72	6.03	1.97	7.24	6.12	2.25	11.12	3.60	2.44	8.53	4.69	2.84

Vita

Roshanak was born in Tehran, Iran. She graduated from high school in 2007. From there, she went to the Amirkabir University of Technology (Tehran Polytechnic), and received a B.S. in Industrial and Systems Engineering in 2008. She was selected as the top 30 student in Iran for continuing her education, and participated in a logistic and supply chain Master's program in the University of Tehran. While doing research in the area of inventory management in supply chain and publishing papers, she got admitted in doctorate program in Industrial and Systems Engineering at University of Tennessee, Knoxville, and simultaneously a Masters in Statistics from the college of business at the University of Tennessee, Knoxville. Roshanak is a member of the Institute of Industrial and Systems Engineers (IISE), Society for Industrial and Applied Mathematics (SIAM), and the institute for Operations Research and the Management Sciences (INFORMS). One of her papers was selected as the best track paper in Industrial Engineering and Operations Management (IEOM) conference.