Technical University of Crete, Greece

School of Electrical and Computer Engineering

Ensemble Neural Network Methods for the Sorting of Recyclables

Antonios Vogiatzis

Chania, January 2022

Πολυτεχνειο Κρητης

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Συνεργατικές Δομές Νευρωνικών Δικτύων για την Ταξινόμηση Ανακυκλώσιμων Υλικών

Βογιατζής Αντώνιος

Χανιά, Ιανουάριος 2022

Abstract

The classification of recyclable materials, particularly the recovery of plastic, is critical not only for economic sustainability, but also for environmental sustainability. Deep learning is a machine learning paradigm that employs artificial neural networks with multiple layers to progressively extract higher level features from raw input; a paradigm that in recent years has achieved extraordinary success in a wide variety of applications, including the classification of materials. This thesis introduces two novel ensemble neural network methods for image classification. Our methods build on the concept of "shared wisdom from data" in order to effectively classify recyclable materials. Specifically, in the first part of this thesis we introduce the so-called "Dual-branch Multi-output CNN" based on recent work in deep learning and waste classification. This is a custom convolutional neural network composed of two branches that aims to i) classify recyclables and ii) distinguish the type of plastic. The proposed architecture consists of two classifiers that have been trained on two distinct datasets in order to encode complementary characteristics of recyclable materials. Our approach allows the learning of disjoint label combinations, by making use of the datasets' joint utilization—but without requiring their mingling. In the second part of this thesis, we propose a generic classification architecture based on independent parallel CNNs that explicitly exploits a "mutual exclusivity" or "classifiers' mutually supported decisions" property that exists in many dataset domains of interest, namely that an image in a given dataset may almost unquestionably belong to only one class. Our framework incorporates several purpose-built opinion aggregation decision rules that are triggered when the mutual exclusivity property is satisfied or not; and it makes use of "weights" that intuitively reflect each CNN's confidence in correctly identifying its corresponding class. Thus, our framework can (a) take advantage of clearly defined class boundaries when they exist, and (b) successfully assign items to classes when clearly defined class boundaries do not exist. Our experiments, with two well-known problem-specific image datasets, confirm the effectiveness of our ensemble neural network methods in the classification of recyclables.

Περίληψη

Η ταξινόμηση των ανακυκλώσιμων υλικών, ιδιαίτερα η ανάκτηση του πλαστικού, είναι πολύ σημαντική όχι μόνο για την οικονομική βιωσιμότητα, αλλά και την οικολογική βιωσιμότητα. Η βαθιά μάθηση αποτελεί μια κατηγορία αλγορίθμων μηχανικής μάθησης που χρησιμοποιούν τεχνητά νευρωνικά δίκτυα με πολλαπλά επίπεδα για τη σταδιακή εξαγωγή χαρακτηριστικών υψηλότερου επιπέδου από την ακατέργαστη είσοδο. Πρόκειται για μια προσέγγιση που τα τελευταία χρόνια έχει χρησιμοποιηθεί με εξαιρετική επιτυχία σε μια μεγάλη ποικιλία εφαρμογών, συμπεριλαμβανομένης της ταξινόμησης υλικών. Η παρούσα μεταπτυχιακή διατριβή εισάγει δύο καινοτόμες μεθόδους ταξινόμησης εικόνων βασισμένες σε συνεργατικές δομές νευρωνικών δικτύων τις οποίες προτείνουμε. Οι μέθοδοι μας βασίζονται στην έννοια της λεγόμενης “διαμοιραζόμενης σοφίας από δεδομένα”, ώστε να ταξινομούν αποτελεσματικά τα ανακυκλώσιμα υλικά. Συγκεκριμένα, στο πρώτο μέρος της παρούσας μεταπτυχιακής εργασίας παρουσιάζουμε το αποκαλούμενο « Dual-branch Multi-output CNN» βασιζόμενοι σε σύγχρονες έρευνες στη βαθιά μάθηση και στην ταξινόμηση αποβλήτων. Αυτό είναι ένα προσαρμοσμένο συνελικτικό νευρωνικό δίκτυο που αποτελείται από δύο κλάδους και αποσκοπεί i) στο να ταξινομήσει τα ανακυκλώσιμα υλικά και ii) στο να διακρίνει τον τύπο του πλαστικού. Η προτεινόμενη αρχιτεκτονική αποτελείται από δύο ταξινομητές που έχουν εκπαιδευτεί σε δύο ξεχωριστά σύνολα δεδομένων, προκειμένου να αξιοποιήσουν παραπλήσια χαρακτηριστικά από τα ανακυκλώσιμα υλικά. Η προσέγγισή μας επιτρέπει την εκμάθηση διαφορετικών συνδυασμών χαρακτηριστικών με ταυτόχρονη αξιοποίηση των διαφορετικών συνόλων δεδομένων, χωρίς να απαιτεί την ανάμειξή τους. Στο δεύτερο μέρος της εργασίας μας, προτείνουμε μια γενική αρχιτεκτονική ταξινόμησης βασισμένη σε ανεξάρτητα παράλληλα συνελικτικά νευρωνικά δίκτυα (CNNs), η οποία αξιοποιεί την ιδιότητα της «αμοιβαίας αποκλειστικότητας» ή αλλιώς των «αμοιβαία υποστηριζόμενων αποφάσεων ταξινομητών» που χαρακτηρίζει πολλά σύνολα δεδομένων από τομείς ενδιαφέροντος: εν συντομία, συχνά μια εικόνα που περιλαμβάνεται σε ένα σύνολο δεδομένων ανήκει, κατά κοινή παραδοχή των ταξινομητών, σχεδόν αποκλειστικά σε μία μόνο κλάση. Παρέχουμε με άλλα λόγια ένα πλαίσιο λήψης αποφάσεων ταξινόμησης, το οποίο ενσωματώνει διάφορους προτεινόμενους κανόνες απόφασης, που ενεργοποιούνται όταν η ιδιότητα αμοιβαίας αποκλειστικότητας ικανοποιείται ή όχι. Ταυτόχρονα, χρησιμοποιεί «βάρη» που α-ντικατοπτρίζουν διαισθητικά το ποσοστό εμπιστοσύνης του κάθε CNN όσον αφορά την ταξινόμηση ενός υλικού στην κλάση στην οποία το δίκτυο είναι εξειδικευμένο. Έτσι, η προσέγγιση μας μπορεί (α) να εκμεταλλευτεί τα σαφώς καθορισμένα όρια κλάσεων, όταν τέτοια υπάρχουν, και (β) να αναθέτει στοιχεία επιτυχώς σε κλάσεις, ακόμα κι όταν δεν υπάρχουν σαφώς καθορισμένα όρια κλάσεων. Τα πειράματά μας με δύο γνωστά σύνολα δεδομένων εικόνων από τον συγκεκριμένο τομέα εν-διαφέροντος, επιβεβαιώνουν την αποτελεσματικότητα των προτεινόμενων μεθόδων και αρχιτεκτονικών συνεργατικών δομών νευρωνικών δικτύων στην ταξινόμηση των ανακυκλώσιμων υλικών.

Acknowledgements

First, I would like to thank my advisor Georgios Chalkiadakis for his trust and guidance during the course of this thesis, and Michailis Zervakis for the help with the computer vision domain.

My friends from Chania, Niko, Dimitri, Eriko and Maria for chatting interesting ideas.

Last, but not least, I would like to thank my family for their love, support, and constant encouragement.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Image classification refers to a method which is capable of classifying an image according to its visual content. For example, an image detection algorithm may be developed to tell whether an image contains a human being or not. While it is simple for humans to recognize an object, accurate image recognition is still a problem in computer vision applications. An effective means of treating recyclable waste can be used to distinguish recyclable products into types of recycling. The current recycling process often requires recycling facilities to sort by hand. Consumers can also be confused about the correct way to recycle materials. By using computer vision, we can predict the category of recycling of an object based on an image of it. The classification of recyclable materials is an interesting task that helps to protect the environment.

Material classification using neural networks and deep learning is in line with the current era (the era of deep learning). Deep learning has seen tremendous success in a variety of machine learning and computer vision applications. In contrast to traditional machine learning approaches, which are unable to process data in its natural form, deep learning allows multiple processing layers to learn features on their own. Deep convolution networks have demonstrated outstanding performance in image and video processing.Due to the dominance of deep learning (mostly deep neural networks) in various artificial intelligence applications, ensemble learning based on deep neural networks (ensemble deep learning) has recently demonstrated significant performance in improving learning system generalization. However, because modern deep neural networks typically have millions to billions of parameters, the time and space overheads for training multiple base deep learners and testing with the ensemble deep learner are significantly higher than those for traditional ensemble learning. To address this issue, it is critical to understand how ensemble learning evolved during the deep learning era.

There are various approaches to solving machine vision problems available today. Federated learning (FL) and split learning (SL) are two prominent approaches to distributed machine learning.Both use a model-to-data approach, which means that clients train and test machine learning methods without sharing raw data. Because the machine learning model architecture is split between clients and the server, SL provides better model privacy than FL.Furthermore, because of the split model, SL is a better choice for resource-constrained environments, such as waste recycling. All the aforementioned learning approaches were used in this thesis to contribute to novel attacks/treatments of the classification problem, along with voting mechanisms for classification tasks such as a hard voting ensemble, which involves summing the votes for crisp class labels from other models and predicting the class with the most votes, or a soft voting ensemble, which involves summing the predicted probabilities for class labels and predicting the class label with the largest sum probability.

In this thesis, we explore the problem of material classification by using deep learning to select the right ways to classify recyclable materials for environmental purposes. In particular, we developed two novel ensemble deep neural network methods for recyclable classification. Our methods make use of an effective convolution model, which can help to create useful applications for sorting trash materials without the need for human interaction with recycling equipment. In addition, we make use of transfer learning approaches, in particular the fine-tuning approach [1]. These transfer learning models are coupled with ensemble learning to make the whole system more robust.

We are motivated by two use-case scenarios, and we put forward a different approach for each. In one case, we have two datasets that have common features, and one is a subcategory of the other. In more detail, these are the Trashnet dataset (which contains glass, paper, metal, plastic, and trash material) [2] and the WaDaBa dataset [3] (which contains different labelled types of plastic, but augmented with non-plastic material in our case). We then propose that a Stacked Ensemble of CNNs branch is used for waste sorting, and a traditional CNN branch be used for plastic classification. The approach that we introduce, so called "Dual-branch Multi-output CNN", is a custom convolutional neural network composed of two branches aimed to i) classifying recyclables and ii) distinguishing the type of plastic. In particular, our approach makes use of the joint utilization of the datasets, making it possible to learn disjoint label combinations.

As a second use-case of interest, for the same datasets, we test a modification of one-vs-rest classification mechanism that utilizes the mutual features of every class. The proposed architecture consists of a set of independent dedicated binary classifiers, each one with two output nodes, with each output node expressing the probability of the item under consideration belonging to this class or not. We put forward a complete framework of soft voting methods, partly inspired by social choice theory, with the potential of being useful for a variety of datasets (as we can base our classification decisions on classifiers that are individually best suited to different datasets). We evaluated our framework with promising early results. Specifically, we show that our method allows for the effective separation of items based on distinct class features and characteristics; and is able, in its generality, which will become apparent below, to (a) take advantage of clear class boundaries when they exist, and (b) effectively assign items to classes with increased confidence, even when clear class boundaries do not exist.

1.1 Thesis Contributions

Our motivation was to find a method for the automated sorting of recyclable material. It has the potential to make manufacturing facilities more efficient and to help eliminate recyclable material, since staff can not typically process anything with 100% accuracy. This will not only have positive impacts on the ecosystem but will also have significant economic impacts.

Our classification problem involves capturing photos of a single object and classifying it as a recycled content category in order to mimic a stream of products in a recycling facility or a consumer taking a photo of a material to identify it. This research was conducted as part of a wider research effort, specifically the ANASA project [4]. The ANASA project aims to create, integrate, and commercialize an autonomous robotic system for categorizing and separating recyclable materials. The development of an automated procedure for recyclable waste separation will significantly contribute to increasing the (currently low) recycling rates in Greece, to the benefit of local societies and the economic enhancement of recycling activities throughout the country. Another need addressed in this thesis is the scarcity of real-world datasets. One major component of the effort was the creation of datasets from independent small datasets, as well as the development of multi-input systems that used many existing datasets and performed augmentation in order to obtain data for our methods to train on. Finally, we used the methods developed in this master's thesis for object detection to meet the project's requirements.The core of our approach was, of course, the development of the aforementioned ensemble learning methods.

We now present our main contributions in some more detail. First, we introduce a novel two-branches multi-output CNN, which integrates multiple source information at the final level, for recyclable waste classification. The need for this component arises from the need to address and exploit the peculiarities of the data. We show, through extensive testing, that our approach results in a significant improvement in performance. Transfer learning is demonstrated, and material classification is significantly improved, by leveraging the output of the plastics classifier, whose performance boosts that of the overall network.The success of our approach emphasizes the importance of utilizing information derived from various attributes and experimental settings. As a result, we can effectively learn from multiple sources and accumulate knowledge without mingling the data samples. At the classification level, the combination scheme exploits the individual results of subnets in a way that boosts confidence in the final decision-making process. Our results show that we can effectively use independent datasets to achieve high classification performance.

In the second part of our thesis we proposed an architecture that consists of a set of independent dedicated binary classifiers, each one with two output nodes, with each output expressing the probability of the item under consideration to belong to this class or not. A key novel intuition is that when only one dedicated classifier puts a "large enough" probability on an item belonging to its class, and all others believe the item cannot be classified. Then we can be confident that the class to select as the output of our system is indeed the one predicted by the k-th classifier. Typically, many image datasets contain photos with wildly disparate attributes. This enables the training of several classifiers with a high degree of confidence.

In more detail, the main contribution of this second aspect of our work is putting forward a generic architecture that explicitly exploits a *"mutual exclusivity property"* underlying many dataset domains of interest: the fact that in many cases an image in a given dataset might almost unquestionably belong to one class only. Our proposed architecture is able to separate individual features specific to each class, from the pool of shared features; and includes *designed-to-purpose decision rules* that are able to select the final class of the item to be classified. To this end, and to allow for flexibility and easy adaptation to the specific properties of the datasets of interest, our approach also makes use of "weights" associated to each independent classifier, and which intuitively mirror the confidence each dedicated classifier has in identifying its corresponding class.

An additional, in a sense "emergent" contribution, is the fact that the proposed framework may be utilized to characterize a given dataset regarding its "homogeneity" and "balancedness" properties. Two research papers were published as a result of this work: *Dual-Branch CNN for the Identification of Recyclable Materials* [5] and *A Novel One-vs-Rest Classification Framework for Mutually Supported Decisions by Independent Parallel Classifiers* [6] with one of them [5] receiving the best student paper award in the in the proceedings of the 2021 IEEE International Conference on Imaging Systems and Techniques (IST-2021).

1.2 Thesis Overview

Finally, we provide a concise summary of our thesis's content:

- In Chapter 2 we present all the background information needed for this thesis. We provide an explanation of the terms and concepts used in Waste Sorting and Multi-Class Classification, as well as the ensemble and transfer learning techniques that will be used. Additionally, we will discuss the appropriate metrics for evaluating the various strategies.

- In Chapter 3 we outline our Dual Branch CNN for classification of recyclables, evaluate its performance and compare it with a specific baseline method.

- In Chapter 4 we describe our, one-vs-rest classification framework for mutually supported decisions by independent parallel classifiers. The performance of the proposed approach is assessed and compared to other benchmark approaches.

- Finally, Chapter 5 serves as an epilogue to this theory, summarizing our findings and outlining recommendations for future improvements.

Chapter 2

Background

In this chapter, we provide some necessary background information. The rest of this chapter is organized as follows: In Section 2.1 we provide some information about what it is, image classification and how deep learning incorporate in it. In Section 2.2 we explain the case of Multi-class Image classification.In Section 2.3 we introduced the Recyclables Classification domain. In Section 2.4 we provide all the necessary information about the Architecture of an Artificial Neural Network and the basic components. Additionally, we present a brief summary of Gradient Descent optimization algorithms. In Section 2.5 we explain a calibration technique for Neural Networks refers as *Temperature Scaling*. In Section 2.6 we specify the *Transfer Learning* approach, that utilized in every network used in this thesis. In Section 2.7 we introduced the *Ensemble Learning* techniques, especially the *Voting Mechanisms*.In Section 2.8 we presented and analyzed the evaluation metrics we used to access the performance of the proposed networks. In Section 2.9 we present a selection of works that are related to this thesis.

2.1 Image Classification

The objective of image classification is to categorize and identify groups of pixels or vectors inside an image using predefined rules. The law of categorization can be implemented via one or more spectral or textural characterizations.

Image classification techniques are mainly divided into two categories: Supervised and Unsupervised image classification techniques.

Supervised image classification techniques employ previously identified reference samples (the ground truth) to train the classifier and then classify unknown input.

Thus, the supervised classification approach is the process of visually selecting training data samples from an image and assigning them to pre-defined categories, such as plants, roadways, natural resources, and houses. This is done in order to provide statistical metrics that can be used to analyze the overall image.

A computer examines a pixel-based picture. It accomplishes this by treating the picture as a collection of matrices, the size of which is determined by the image resolution. Simply said, picture classification is the statistical examination of this data utilizing algorithms from a computer's perspective. Image classification is accomplished in digital image processing by automatically categorizing pixels into specific groups, referred to as "classes".

2.1.1 How Image Classification Works

Image classification is a supervised learning problem: identify a set of target classes (objects to be recognized in images) and train a model to recognize them using the example photos labelled. Early computer vision models depend on raw pixel data as a sample input. This meant that computers must first convert an image into discrete pixels. The problem is that two images of the same object can appear to look very different. However, raw pixel data on its own does not offer a sufficiently stable representation to encompass the various variations of an entity as captured in a photograph. The position of the object, the background behind the object, the ambient illumination, the orientation of the camera and the focus of the camera will all cause variations in raw pixel data; These differences are so severe that they cannot be resolved by averaging the RGB pixel values. This influenced the accuracy of computer vision algorithms to categorise image objects correctly.

To represent things more flexibly, traditional computer vision models integrated additional properties obtained from pixel data such as color histograms, textures, and forms. The limitation of this strategy was that feature design became extremely time-consuming due to the large number of inputs that needed to be adjusted. For instance, which colors were most significant for a goat classifier? How flexible should the textures definitions be? Since the features required to be calibrated very finely, constructing robust models remained challenging, and the accuracy suffered.

2.1.2 Adding deep learning

Machine learning-based image recognition takes use of algorithms' ability to extract hidden information from a collection of structured and disorganized samples (Supervised Learning). The most widely used machine learning approach is deep learning, which employs a large number of hidden layers in a model.

Deep learning is a form of machine learning; a subset of artificial intelligence (AI) that enables machines to learn from data. Deep learning includes the use of computing structures called neural networks. In neural networks, input filters via hidden node layers. These nodes each process the input and transmit the results to the next node layer. This procedure is repeated until the computer reaches the output layer and responds. There are several forms of neural networks, each with a distinct operation of the hidden layers. Convolutional neural networks, or CNNs, are typically used in deep learning picture classification. In CNNs, nodes in hidden layers do not always share their output with another node in the next layer (known as convolutional layers).

Computers use deep learning to detect and extract characteristics in images. This implies that they may develop an understanding of the traits they need to look for in photos by analyzing numerous images. Thus, programmers are not required to input these filters manually, but simply to tweak them.

2.2 Multi-class Image classification

The purpose of this study is to build an algorithm for the automatic classification of potentially recyclable waste items, a task that is significantly challenging due to the state (e.g. level of dirtiness, shape deformation, etc.) of the items during their sorting in waste treatment facilities.

Image classification is a term referring to a technique for classifying images based on their visual content. For instance, an algorithm for image classification may be built to determine whether an image contains a human person. Although people easily recognize objects, reliable image classification remains a challenge in computer vision applications.

The majority of image recognition problems involve the use of a CNN to solve a multi-class classification problem. In the machine learning publications, diverse methods have been introduced for attempting to deal with numerous classes [7]; in deep learning, the One-vs-All classification approach is commonly used.

Multi-class classification models are available in a variety of types, in the machine learning domain. The most common classification method is One-Versus-One (OVO) or One-Versus-Rest (OVR) utilize bi-class model [8, 9]. Others approaches include One-class classification [10], error-correcting output codes [11, 12] and hierarchical methods [13].

As is usually used in the literature, the term "one-vs-rest" describes a generic classification paradigm that uses binary classification algorithms for multi-class classification. It entails approaching the multi-class problem via the lens of many "binary" classification ones. A dedicated "binary" classifier (that potentially uses a sigmoid function for classification into the most probable of the two classes) is then trained on the original dataset, and solves a "binary" classification problem; and final predictions are usually made using the classifier that is the most confident, or using a simple or absolute majority rule. One advantage of this method is that machine-learning algorithms developed for binary classification could easily be adapted to represent multi-class classification problems. The disadvantage is that when the training data contains a disproportionate number of negative examples to positive ones, the classifiers become unbalanced.

On the other hand, the "one-vs-one" is a categorical classification paradigm that employs binary classification algorithms to perform pair-wise multi-class classification. That is, the one-vs-one approach uses a "binary" classifier per pair of potential classes. With the OvO scheme, the binary classifier is capable of distinguishing between examples of one class and examples belonging to another class. Thus, when there are K classes, the OvO approach requires training and storing $\frac{k*(k-1)}{2}$ distinct classifiers, which can be considered as a drawback when K is high. Then, final predictions are usually made using the classifier that is the most confident, or using some weighted voting method [14]. These methods, however, have high computational costs.

Deep neural networks for Multi-class classification. Due to the fact that deep neural networks are utilized for multi-class classification tasks, the output layer is typically chosen to use a "softmax" function, and one output unit for each separate class. Therefore, since the final units share the same hidden layers (performing features extraction), this is a one-vs-rest classification design. An interesting idea is the *Attribute* *learning* [15, 16], in which various attributes are forecast, and their fusion is used to predict a class, is an enticing multi-class learning method.

2.3 Recyclables Classification

Waste classification may be handled in a variety of ways, examining either the civilian's behavior towards circular economy [17] or the operation conditions of the recycling facilities [18]. The main trend in utilizing image classification in the recycle waste industry is focused on the identification of the recyclable waste material and moreover on the type in case of plastic material. The plastic waste recycling and recovery industry has a great impact in the U.S. and Europe [19]. Thus, many attempts have been made, mainly in the area of waste material identification using CNNs. An image classification approach to classifying recyclable material into types of recycling may be an efficient means of handling recyclable waste. The purpose of this thesis is to photograph one piece of recycling or rubbish and classify it into six categories consisting of: glass, paper, metal, plastic, cardboard, and trash. This study presents two novel image classification models that can be used to distinguish recyclable materials.

Due to the rapid advancement of deep learning technology, a number of network models for classification have been developed, which is advantageous for achieving intelligent waste categorization. However, there are significant issues with existing waste classification models, such as low classification accuracy or a significantly longer run time. To address these issues, this article [20] proposes a waste classification approach based on a multilayer hybrid convolutional neural network (MLH-CNN). This approach uses a network structure similar to VggNet, but with fewer parameters and a higher classification accuracy. The performance of the suggested model can be improved by adjusting the number of network modules and channels. Subsequently, this article determines the right parameters for classifying waste images and selects the optimum design as the final model.

Another innovative strategy for improving the accuracy of waste classification driven by CNNs was to apply data augmentation; nevertheless, fine-tuning the CNN's fully connected-hyper-parameters layer's optimum was never used. Thus, in addition to data augmentation, this study [21] intends to optimize the fully connected layer of DenseNet121 using a genetic algorithm (GA) in order to increase the classification accuracy of DenseNet121 on "TrashNet" and offers the optimized DenseNet121.

2.3.1 Datasets which used in Recyclables Classification

Yang and Thung (2016) constructed a dataset of waste photos. Six classes are represented in the dataset: glass, paper, cardboard, plastic, metal, and trash. Currently, the dataset contains 2527 photos commonly called TrashNet [2]: 501 photographs of glass, 594 images of paper, 403 images of cardboard, 482 images of plastic, 410 images of metal, and 137 images of trash. The photographs were shot by putting the object on a white whiteboard and utilizing natural and/or fluorescent lights. The models, which included a support vector machine (SVM) with scale-invariant feature transform and a ResNet50 with SVM, were used to categorize garbage images in the TrashNet, with the two models achieving a classification accuracy of 63% and 87%, respectively.

The WaDaBa database [3] has 4000 photos divided into five categories: PET (01-polyethylene terephthalate), PE-HD (02-high-density polyethylene), PP (05-polypropylene), and PS (06-polystyrene) (07). Each image is composed of a single object that has been deformed to various degrees to approximate natural settings. After positioning the objects in the research posture, they were photographed using two different types of light sources: a fluorescent lamp and an LED-bulb. There were ten images taken, each with a different viewpoint of the turnover for each object (in the vertical axis). Following that, the object was deformed in three different ways: small, medium, and large. Ten images were taken for each category of destruction.

Another dataset that is used to classify recyclables is "Waste Classification Data", which comes in two versions [22]. The first is the categorization of samples into two broad categories: organic and recyclable materials, which is expanded in the second version of the dataset by the addition of a new category "N" - Nonrecyclable materials.

2.4 Artificial Neural Networks

Neural networks (NNs) are a computer technique that is loosely patterned on the biological brain. Brains can be thought of as an interconnected neuron network that transmits complex patterns of electrical impulses: input signals are sent to dendrites, and the output signal is fired via an axon in response to those inputs. Neural Networks emulate this effect by employing artificial neurons that can be formally defined as a graph. NNs receive input from the outside world through the input neuron and propagate forward-looking: the input is transmitted to other neurons in the network, and each neuron modifies the signal according to its internal rules until the signal passes through the entire structure and reaches the output neurons. Such systems 11learn" to perform tasks through examples, often without having to be programmed with task-specific restrictions.

2.4.1 The Architecture of an Artificial Neural Network

An Artificial Neural Network (ANN), as shown in Figure 2.1 is a set connected neurons in layers:

- **Input layer:** Introduces the initial data for further treatment into the system through subsequent artificial neuron layers.
- **hidden layer:** A layer of weighted inputs from artificial neurons to produce the output via activation function between input layers and output layers.
- **Output layer:** The last neuron layer generating the outputs for the network.

2.4.2 Multi-layer ANN

Due to their hidden layers (for example, Convolutional Network, Recurrent Neural Network, etc...), multilayer ANNs can solve more complicated classification and regression tasks.

There are many ways, neural multilayer networks can be set up. They typically have at least one input layer, which sends weighted inputs to a number of hidden layers. These advanced settings are also related to non-linear builds that use sigmoids and other functions to direct artificial neurons firing or activation. Although some of these systems are physically constructed with physical materials, most of them are designed with neural activity software functions.

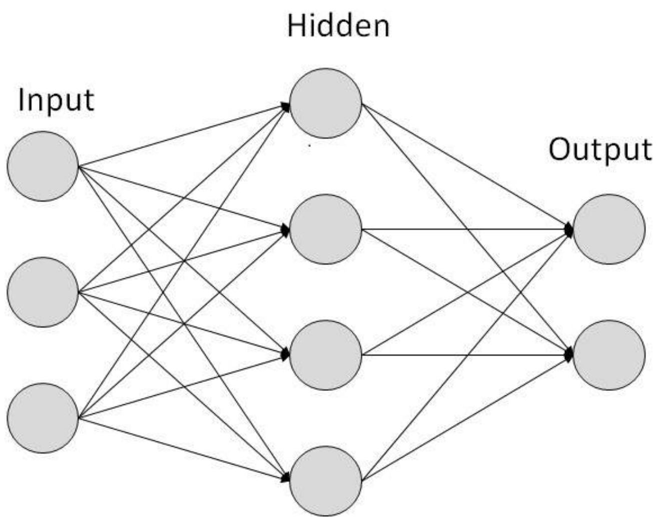


Figure 2.1: Artificial Neural Network

2.4.3 Convolutional Networks

Convolutional [23] networks are a distinct type of neural network for processing data that has a known grid-like topology, also known as convolutional neural networks or CNNs. Time-series data, for example, can be thought of as a one-dimensional grid taking measurements at regular time intervals, whereas image data can be thought of as a two-dimensional grid of pixels. Convolutional networks have demonstrated outstanding performance in practical applications. The phrase "convolutional neural network" refers to a network that makes use of a mathematical operation known as convolution. Convolution is a specialized type of linear operation. Convolutional networks are essentially neural networks that employ convolution in place of traditional matrix multiplication in at least one of their layers.

Convolution Operation

Convolution, in its simplest form, is the action of a real-valued argument on two functions. Assume we are using a laser sensor to track the location of a satellite. Our laser sensor outputs a single value $x(t)$, the satellite's position at time t. Both x and t are real-valued, which means that we can obtain a different reading from the laser sensor at any point in time.

Now consider the possibility that our laser sensor is kind of noisy. We'd like to combine multiple measurements in order to obtain a less noisy satellite location estimation. While more recent observations are obviously more significant, we want this to be a weighted average that prioritizes recent values. This is accomplished by the use of the weighting function $w(a)$, where a denotes the measurement time. If we perform this weighted average operation at any point in time, we gain a new function that approximates the satellite's position smoothly:

$$s(t) = \int x(a)w(t-a)da \tag{2.1}$$

This operation is called convolution. The convolution operation is usually demoted with an asterisk:

$$s(t) = (x * w)(t) \tag{2.2}$$

Convolution is defined in general for any functions for which the aforementioned integral is specified, and can be used for reasons other than weighted averages.

The first parameter (in this aforementioned paradigm, the function x) is often referred to as the **input**, whereas the second argument (in the same paradigm, the function w) is referred to as the **kernel**. The output is typically referred to as the **feature map**.

The concept of a laser sensor capable of performing measurements at any time is unfeasible. When we work with data on a computer, time is typically discrete, and our sensor will collect data at regular intervals. In our scenario, it makes more sense to suppose that our laser performs the measurement once every second. The time index t, on the other hand, can only take integer values. Given that x and w are now defined only for the integer t, we may construct a discreet convolution:

$$s(t) = (x * w)(t) = \sum_{\alpha=-\infty}^{\infty} x(a)w(t-\alpha) \tag{2.3}$$

Input is typically a multidimensional array of data in machine learning tasks, while *kernel* is normally a multidimensional array of parameters tweaked by the learning process. We're going to call these multidimensional arrays as *tensors*. Finally, we always utilize convolutions that span many dimensions. For instance, consider the scenario of a two-dimensional image I as input tensor, and we apparently want to use a two-dimensional kernel K as well:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(i-m, j-n) I(m,n) \tag{2.4}$$

Convolution is rarely used alone in machine learning; convolution is frequently employed in conjunction with the other functions described in the following subsection 2.4.4. In Figure 2.2 is shown an illustration of convolution applied to a 2-D tensor.

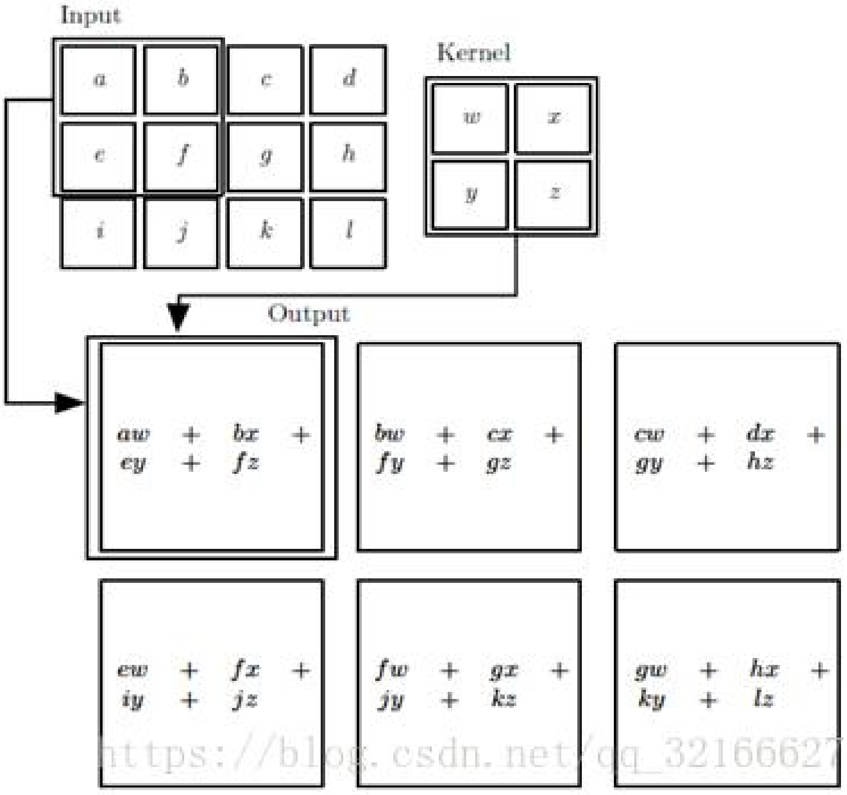


Figure 2.2: Convolution applied to a 2D tensor [2]

Convolution Components

Convolution integrates three main concepts that can help develop the machine learning system: **sparse interactions**, **parameters sharing** and **equivariant representations**. Convolution also yield a way to work with variable size inputs. Now we're going to explain each of these ideas in turn.

Conventional neural network layers employ matrix multiplication of parameters, with each input unit and output unit represented by a different parameter. This means that each output unit is connected to each input unit. However, convolutional networks appear to have *sparse interactions*. This is accomplished by maintaining a kernel that is smaller than the input tensor. When processing a picture, for example, the input image may contain thousands or millions of pixels, but we may discover minor, significant features

[2]taken from https://www.deeplearningbook.org/contents/convnets.html

such as edges by utilizing kernels that are only tens or hundreds of pixels in size. This requires us to keep fewer parameters, which reduces the model's memory requirements and improves its statistical performance. Additionally, it assures that the outcome is calculated with fewer operations. For schematic demonstrations of sparse connection, see Figure 2.3. In a deep convolutional network, units in the deeper layers can interact with a bigger proportion of the input indirectly. This enables the network to explain complicated relationships between a range of variables effectively by deriving interactions from fundamental building elements that each represent only sparse interactions.

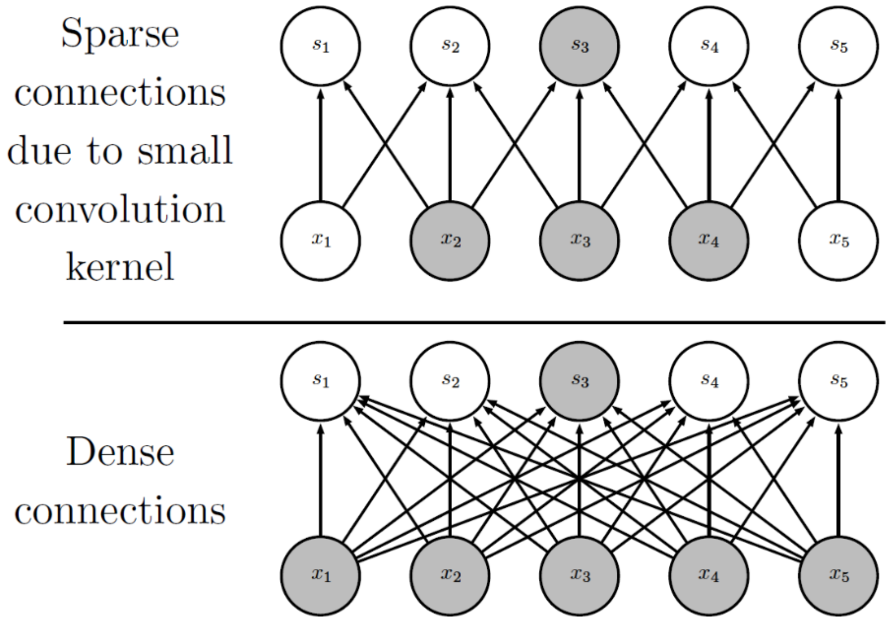


Figure 2.3: Sparse connectivity vs Dense connectivity [3]

The term *shared parameter* refers to the fact that the same parameter is used by multiple functions in the model. Each component of the kernel is utilized at each input area in a convolutional neural network (except perhaps some boundary pixels, depending on the design decisions regarding the boundary). The shared parameter utilized in convolution implies that, rather than learning a unique set of parameters for each area, we learn a single set. Thus, convolution is significantly more efficient than dense matrix multiplication in terms of memory limitations and statistical efficiency.

[3]taken from https://www.deeplearningbook.org/contents/convnets.html

In the case of convolution, the specific type of shared parameter enables a property known as translation *equivariance* to be applied to the layer. Equivariant means that if the input changes, the output changes in the same way. Particularly, if $f(g(x)) = g(f(x))$, a function $f(x)$ is equivariant to a function g. In the case of convolution, if we consider g to be a function that modifies the input, more precisely shifts it, the convolution function is identical to g. Convolution is, of course, not equivalent to any other transformations, such as a shift in the image's scale or rotation. Other techniques are required to manage these transitions.

2.4.4 Activation Functions

The node activation function regulates the output of this node in response to an input or combination of inputs in artificial neural networks.

Sigmoid

A sigmoid feature is a mathematical function that has a characteristic "S" or sigmoid curve [24]. The sigmoid function is widely used to refer to a specific case of the logistic function, which produces a set of probability outputs ranging from 0 to 1. In binary classification, sigmoid activation is a common feature.

$$Sigmoid(x) = \frac{1}{1 + e^{(-x)}} = \frac{e^x}{1 + e^x} \tag{2.5}$$

Tan-h

The hyperbolic tangent, or tan-h function, is a function that could be used in place of the logistic sigmoid [25]. The tan-h function, like the Sigmoid logistic function, is a Sigmoid function, but its range is [-1,+1]. This means that strongly negative tan-h inputs map negative outputs. Similarly, strongly positive.

$$tanh(x) = \frac{2}{1 + e^{(-2x)}} - 1 = 2 \times Sigmoid(2x) - 1 \tag{2.6}$$

Softmax

In contrast to Sigmoid activation functions, the Softmax activation function is utilized for multi-class classification [26]. The Softmax function computes the event's probability distribution over "n" distinct events. As a general rule, this function determines the probabilities of each target class over all potential target classes. Following that, the determined probabilities are employed to regulate the data's target class.

$$Softmax(x) = \frac{e^i}{\sum_i e^{(i)}} \tag{2.7}$$

ReLU

Rather than a sigmoid function, modern artificial neural networks make use of rectified linear units (ReLUs) [26]. When the input is less than zero, the linear unit with the rectified output is zero; otherwise, it is the raw input. This means that if the input value is larger than zero, the outcome is identical to the input value.

$$ReLU(x) = \max(0, x) \tag{2.8}$$

Leaky ReLU

The leaky ReLU function operates similarly to ReLU [27], except that the latter is replaced with a small alpha value rather than the negative data from the inputs in order to prevent the "dying ReLU" problem.

$$LeakyReLU(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & otherwise \end{cases} \tag{2.9}$$

In general, without the use of an activation function, non-linearity is not introduced into a network. We can use an activation function to model a response variable that varies nonlinearly in relation to its explanatory variables (target variable, class label, or score). The term ”non-linear” refers to an output that cannot be recreated by a linear combination of inputs. Another way to think about this is that without a nonlinear activation function, artificial neural networks, regardless of their number of layers, behave identically to a single layer Neural Network. Certain activation functions may have an effect on the disappearing gradient problem (the problem with the disappearing gradient occurs when the gradient becomes so small in prior layers of a deep-neural network that it does little to improve the weights of the earlier layers).

2.4.5 Pooling

A standard convolutional layer is composed of three main structural levels. The layer executes numerous convolutions in parallel to generate a series of linear activations in the first step.Each linear activation is routed through a nonlinear activation function, such as the softmax activation function, at the second level. This phase is sometimes referred as **detector stage**. At the third level, we apply the pooling function to fine-tune the layer’s output.

The pooling function substitutes the output of the network at a given position with a detailed statistic of the neighboring outputs. For instance, the *max pooling* formula measures the maximum performance inside a rectangular field. And there are quite a few more pooling functions to pick one in any case , e.g average or L^2 norm of a rectangular neighborhood.

In all cases, pooling results in a representation that is roughly invariant to minor input translations. The term “translation invariance” refers to the fact that the values of the majority of pooled outputs remain constant when the input is converted by a small volume. The usage of pooling can be interpreted as applying an immensely powerful prior that the function the layer learns must be constant over time to minor translations.

While pooling over spatial regions results in translation invariance, pooling over the outputs of independently parametrized convolutions informs the features which transformations to become invariant. Due to the fact that pooling aggregates the reactions across the neighborhood, it is possible to use many fewer pooling units than detector units.

Pooling is required for several tasks in order to manage inputs of varying sizes. For instance, if we wish to identify images of varying sizes, the classification input layer must be defined in size. This is usually accomplished by altering the offset between pooling parts to ensure that the classification layer gets the same amount of overview statistics regardless of the size of the input. For example, regardless of the size of the image, the final pooling layer of the network can be set to provide four collections of summary statistics, one for each quadrant of the image.

2.4.6 Fully Connected Layers

At the convolutional neural network's end, one or more completely connected layers are placed (when two layers are "fully connected", each node in the first layer is connected to every node in the second layer). Their objective is to perform a classification using the transformed features. Typically, the last fully connected layer contains a softmax activation function that generates a probability value ranging from 0 to 1 for each classification label predicted by the model. In Figure 2.4 we can see the overall end-to-end Convolutional Neural Network[3].

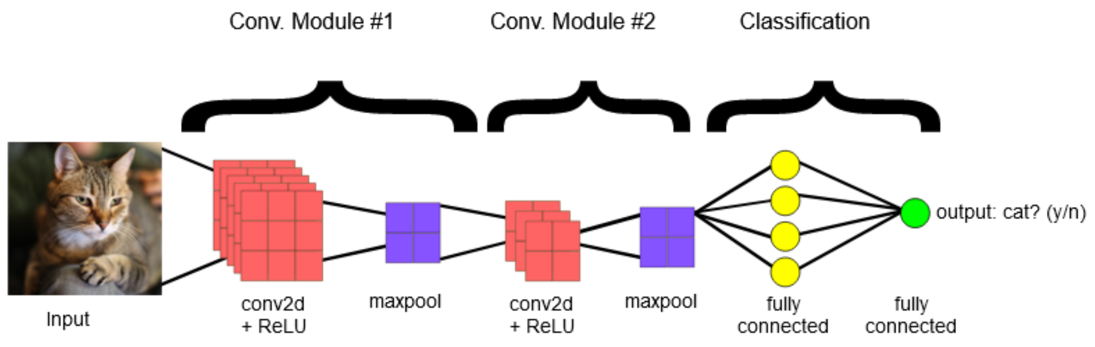


Figure 2.4: End-to-end structure of a CNN

[3]Figure 2.4 taken from https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

2.4.7 Dropout

Dropout works by probabilistically omitting, or ”dropping out”, inputs to a layer. These inputs might be variables from the data sample or activations from a previous layer. It has the effect of simulating a huge number of networks with significantly diverse network structures and, as a result, making nodes in the network more resilient to inputs in general. We use dropout for handling the over-fitting problem of our network.

2.4.8 Backpropagation

Backpropagation [28] is a technique for determining the gradient required to calculate the network weight in artificial neural networks. The term ”backward propagation of error” refers to the process by which an output error is calculated and transferred backward across network levels. It is used to train deep neural networks.

Due to backpropagation, the rule of delta may be applied to multi-layer feedforward networks in order to iteratively calculate gradients for individual layers using the chain rule. It is inextricably tied to the Gauss-Newton method and is being investigated as part of continuing study into the neural background.

Backpropagation is a general concept for the approach referred to as automatic differentiation. Backpropagation is frequently employed in the learning process to optimize the mass of neurons using a downward gradient method in order to calculate the loss function progression.

2.4.9 Gradient Descent Optimization algorithms

In optimization, gradient method is an algorithm to solve problems of the form:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.10}$$

Search directions at the current point defined by the function gradient.

Gradient downward is a well-known method for optimizing neural networks. Each modern Deep Learning library also offers implementations of numerous gradient-descent optimization methods (see, for example, the documentation for lasagne, caffe, and keras).

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The η learning rate determines the steps to reach a (local) minimum. In other words, we walk downwards to reach a valley in the pendulum of the surface created by the objective function.

Gradient descent is categorized into three versions based on the quantity of data points utilized to determine the gradient. Depending on the amount of data, we trade off the accuracy of the updated parameter for the time required to update it.

Batch gradient descent

Vanilla descent, known as the batch descent, calculates the cost function gradient with respect to the full θ data set parameters:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.11}$$

Because the gradient levels for the entire dataset must be generated in order to produce a single update, batch gradient descent is slow and incompatible with memory-free datasets.

Stochastic gradient descent

By contrast, Stochastic gradient descent (SGD) updates each training example $x^{(i)}$ with a parameter, and label $y^{(i)}$:

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.12}$$

Batch gradient descent is not appropriate for integrating massive volumes of data because gradients have to be recalculated for similar samples prior to updating each parameter. SGD reduces this redundancy by completing one update at a time. As a result, it is often significantly faster and can also be utilized for online learning.

Mini-batch gradient descent

Finally, the mini-batch descent takes the best of the two worlds and makes an update for every small batch of n training examples:

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.13}$$

This means that a) it minimizes the parameter's update variance, hence increasing its stability, and b) the widely used sophisticated deep learning libraries for highly optimized matrix optimization enable the gradient to be calculated very efficiently. Mini-batch sizes typically range from 50 to 256, but might vary depending on the purpose. The chosen algorithm for neural network training is mini-batch gradient descent.

Below are some algorithms that the deep-learning community often uses to optimize the task.

Adagrad

Adagrad [29] is a gradient optimization algorithm that simply adjusts the rate according to the parameters, with smaller updates for frequently occurring characteristics (i.e. low learning rate) and greater updates for rarely occurring qualities (i.e. high learning rate) (i.e. high learning rates). This is why it is appropriate for data processing that is insufficient. When data are sparse and descriptive, this technique frequently increases convergence performance in comparison to ordinary stochastic gradient descent. It has a base learning rate η, but this is multiplied using the $G_{j,j}$ diagonal elements of the outer matrix of the product.

$$G = \sum_{\tau=1}^{t} g_\tau g_\tau^\mathsf{T} \tag{2.14}$$

where $g_\tau = \nabla Q_i(W)$ the gradient, at iteration τ. The diagonal is given by

$$G_{j,j} = \sum_{\tau=1}^{t} g_{\tau,j}^2 \tag{2.15}$$

After each iteration, this vector is updated.

One of the primary advantages of Adagrad is the absence of the need to manually modify the learning rate. For the majority of implementations, the default value is 0.01 and it is left at that.

The primary drawback of Adagrad is the aggregation of the square denominator gradients: because each additional period is positive, the accumulated amount grows throughout training. This decreases the learning rate until it reaches a point where no new knowledge can be acquired from the algorithm.

Adagrad customizes the general η learning rate for each step θ_i in its update rules, based on past gradients calculated for θ_i:

$$\theta_{t+1,i} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_t \tag{2.16}$$

Adadelta

Adadelta [30] is an Adagrad extension that reduces the aggressive monotonous learning rate of Adagrad. Adadelta restricts the winder of accumulated previous gradients to a specified size, w rather than all previous square gradients. The sum of gradients is defined repeatedly as the decreasing mean of all gradients, which does not inefficiently save the preceding gradients. The running average $E[g2]t$ then only rely upon the previous average and current gradient (as the fraction γ):

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \tag{2.17}$$

The Adagrad update parameter, which we previously derived therefore takes shape:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \times g_t \tag{2.18}$$

RMSprop

Geoff Hinton proposed RMSprop, an unpublished adaptive learning rate approach, in Lecture 6e of his Coursera Class [31].

Both RMSprop and Adadelta have indeed been independently refined, but the necessity to deal with Adagrad's dramatically lowered learning rates has arisen. In fact, RMSprop is equivalent to the first vector of the Adadelta update:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{2.19}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \tag{2.20}$$

Additionally, RMSprop divides the learning rate by an exponentially decreasing average of squares. Hinton advises that *gamma* be defined as 0.9, and that an excellent default value for η be 0.001.

Adam

Adaptive Moment Estimation (Adam) [32] is another method for calculating adaptive learning rates for each parameter. Adam keeps a decaying average of the prior squared paths v_t as well as an exponentially decaying average of the previous paths m_t. The decaying averages of past m_t and past squared gradients v_t have been calculated in the following manner:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.21}$$

m_t and v_t are the first (average) and second (uncentered) moment of gradient, therefore the name of the method. Due to the fact that m_t and v_t are initialized as vectors of zeros, Adam's creators observe that they are skewed toward zero, particularly in the primary phases and, more specifically, when decadence rates are low (i.e. β_1 and β_2 are close to 1).
By calculating first and second instant biases, they counteract these biases

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-(\beta_1)^t} \\ \hat{v}_t &= \frac{v_t}{1-(\beta_2)^t} \end{aligned} \tag{2.22}$$

These parameters are then used as we saw in Adadelta and RMSprop, which gives the Adam update rule:

$$\theta_{(t+1)} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{2.23}$$

Default values of 0.9 are proposed by the authors for β_1, 0.999 for β_2, and 10^{-8} for ϵ. They demonstrate empirically that Adam is effective and compares favorably to other adaptive learning methods.

2.5 Calibration of Neural Networks

On datasets such as CIFAR100, more sophisticated and complex models, such as ResNet, are more accurate than their simpler versions. But while they are better at classifying images, we do not have as much faith in their confidence. Most neural networks for classification use softmax as the last activation: it generates a distribution of probabilities for each target class (i.e. cat, dog, boat, etc.). These probabilities add up to one. We may assume that if our model assigns a score of 0.8 to the target "dog" for a given picture, our model is certain at 80 per cent that this is the right target. Over 100 images that have been detected as dogs, we can expect about 80 images to be true dogs, while the remaining 20 were false positives. Deeper models were more accurate than the shallower models, but their confidences were *decorrelated* with "actual confidence". This inconsistency between the confidence of the model and the actual accuracy is called *miscalibration*.

To cope with the miscalibration we employ *Temperature Scaling* [33] that is the simplest but the most effective way. In more detail ,instead of compute the standard softmax activation function like Equation 2.7 we modify the function like this:

$$Softmax(x) = \frac{e^{y_i/T}}{\sum_i e^{y_i/T}} \tag{2.24}$$

All logits (values before the final activation, here softmax) are divided by the same temperature value T. Analogous to [34], this temperature "relax" the probabilities. Extreme probabilities (high confidence) soften more than smaller probabilities (low confidence).

2.6 Transfer Learning

Transfer learning (TL) is a machine learning (ML) research area that focuses on storing the knowledge gained while solving one problem and applying it to a different but related problem.

It is a widely utilized strategy in deep learning in which pre-trained models are used as the basis for computer vision and natural language tasks, given the enormous amount of computation and time required to construct neural network modeling on these issues, as well as the enormous skills gained. Transfer learning is an optimization technique which allows incremental progress or improved performance when modeling the second task.

We first train a base network on a known dataset and task, and then repurpose or transmit the learned features to a second target network for evaluation on a fresh dataset and task. This solution would continue to work since the features are general, i.e. they apply to both the base and target jobs, rather than just the base task (Figure 2.5).

2.6.1 Pre-trained Model Approach

1. **Select Source Model:** From the available models, a pre-trained source model is picked. Multiple research institutions publish models on huge and demanding datasets, which can be added to the pool of potential models from which to choose.

2. **Reuse Model:** Depending on the modeling methods employed, the model pre-trained model can then be utilized as a starting point for a model on the second priority task.

3. **Tune Model:**Alternatively, depending on the task given, the model may need to be changed or optimized based on the input-output pair data.

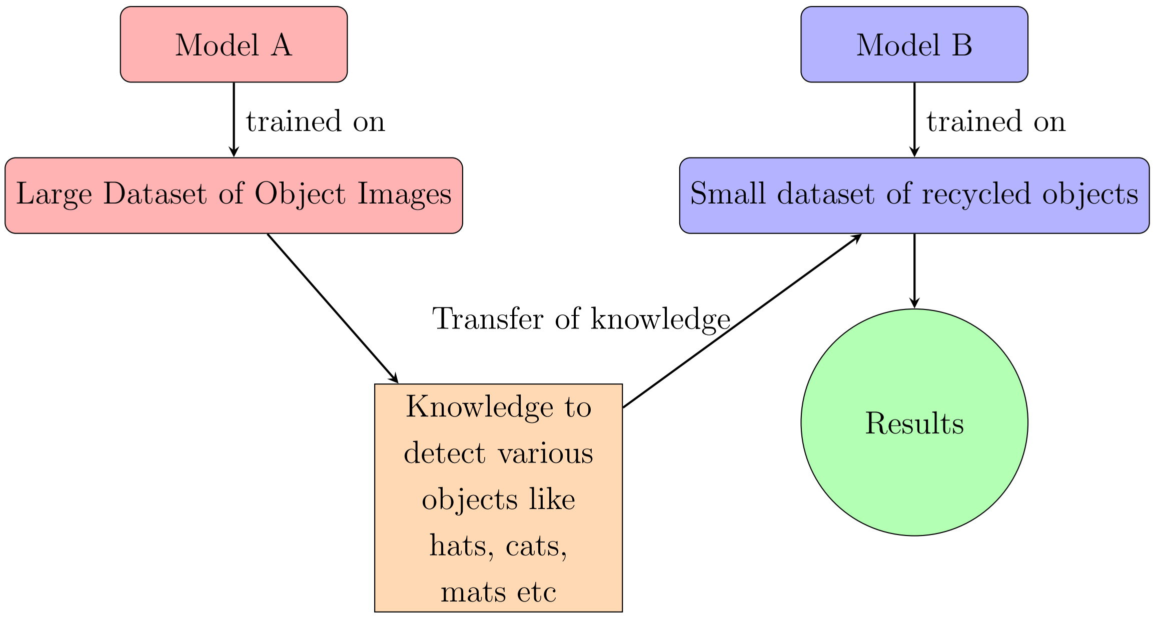

Model A
trained on
Large Dataset of Object Images
Model B
trained on
Small dataset of recycled objects
Transfer of knowledge
Knowledge to detect various objects like hats, cats, mats etc
Results

Figure 2.5: Transfer Learning process

2.6.2 Pre-trained Models as Feature Extractors

Typically, features in machine learning are hand-crafted by academics and domain experts. Luckily deep learning could automatically extract features. This does not of course mean feature engineering and domain intelligence is no longer relevant - you do have to consider which features to incorporate into your network. Neural networks can learn which features are really important, and which features are not.

Afterwards the learned representation can also be used for other tasks. Simply using the initial layers to spot proper feature representation, but do not use the network data because it is too task-specific. Rather, feed data into the network and use the output layer as one of the intermediate layers. The layer can then be interpreted as a raw data representation (Figure 2.6).

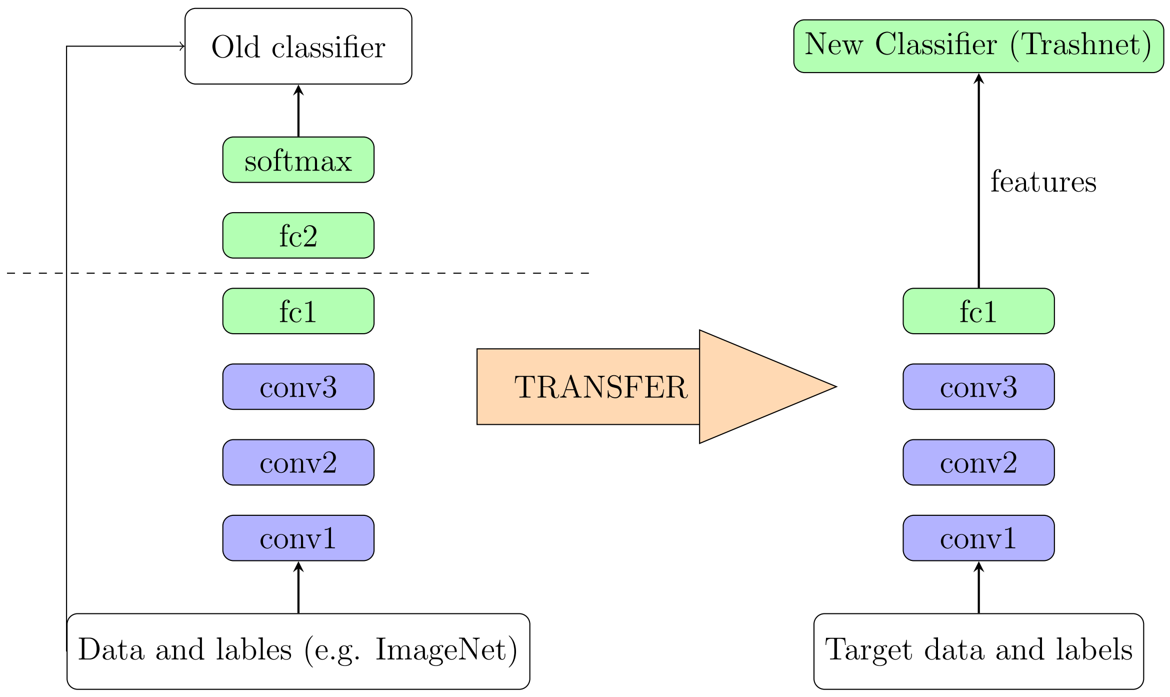


Figure 2.6: Feature Extraction approach

2.6.3 Fine Tuning the pre-trained weights

In the above case, we did not attempt to modify the weights, rather they were used as it is for feature extraction. But this is not what we want in case the images are bit different. We would like the model to relearn some or all of the features it has learned.

We may use the new dataset to fine-tune the CNN that has already been trained. Consider how comparable the new dataset is to the pre-training dataset. Due to the similarity between the two datasets, the same weights could be used to extract the new dataset's features.

- If the new dataset is very small, it is preferable to train only the final layers of the network in order to avoid overfitting, while keeping the other layers unchanged, following the below steps:
    1. The final layers of the pre-trained network should be removed.
    2. Add new layers
    3. Only the newly formed layers require retraining.
- If the new dataset is quite large, retrain the entire network using weights from the previously trained network.

The above 3 methods are summarized and shown schematically in the Figure 2.7.

1. **Strategy 1:** Initialized only the weights from pre-trained network and allow all layer to train.
2. **Strategy 2:** Fine-tuning some "later" layers while all the other early layers being frozen.
3. **Strategy 3:** Freeze all the layers expect from the final layers which are the custom full connected layers (Feature Extractors).

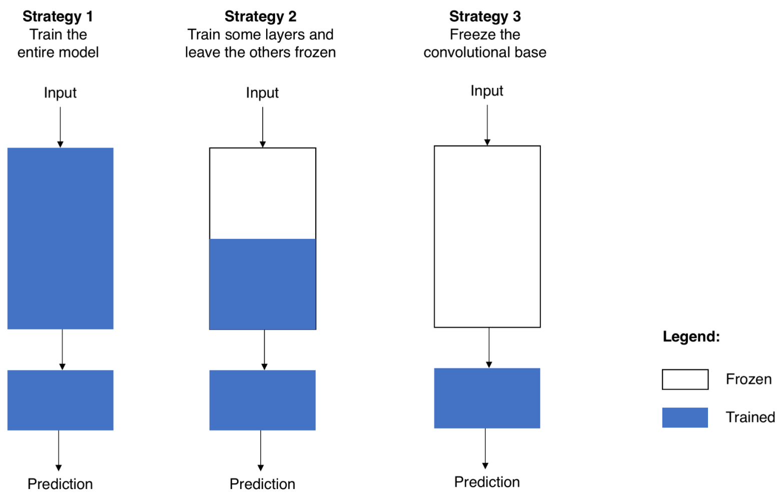


Figure 2.7: Transfer Learning Strategies

2.7 Ensemble Learning

Ensemble learning is a technique for intentionally creating and combining several models, such as classifiers or experts, to tackle a particular computational intelligence problem. Ensemble learning is typically used to improve the efficiency of a model (classification, prediction, feature approximation, etc) or to mitigate the tendency of a weak model. Ensemble Learning can also be used to assign confidence to the model's decisions, to pick optimal (or near-optimal) features, to combine data, to do incremental learning, non-stationary learning, and to perform error correction. While we focus on Ensemble Learning applications for classification, many of the basic concepts discussed below can simply be expanded to work on approximation or prediction-related subjects as well.

The Ensemble-based design is accomplished by integrating a number of models (henceforth classifiers). As a result, these systems are sometimes referred to as multiple classifier systems or ensemble systems. There are some situations in which the use of the Ensemble-based Method makes mathematical sense, which are described in depth below. For example, before committing to a medical operation, we normally ask for the advice of multiple doctors, read customer feedback before buying an item (especially expensive travel ticket), assess potential workers by reading their references, etc. In each situation, a final decision is determined by weighing the individual perspectives of a diverse group of experts. The fundamental objective is to avert the undesired option of an unneeded medical procedure, a bad merchandise, or even an unskilled worker.

2.7.1 Categories of Ensemble Methods

Ensemble approaches split into two distinct groups, namely *sequential* ensemble techniques and *parallel* ensemble techniques. Another criteria for grouping ensemble methods are the homogeneity, there are the *Homogeneous* and *Heterogeneous* methods.

Sequential ensemble techniques yield a sequence of basic learners, e.g. Adaptive Boosting (AdaBoost). The sequential development of base learners fosters interdependence. The model's efficiency is then enhanced by increasing the weights assigned to previously skewed learners.

In **parallel ensemble techniques**, base learners are developed in a parallel format, e.g. random forest [35]. Parallel approaches employ parallel generation of basic learners to foster independence between these learners. The independence of base learners significantly reduces the inaccuracy introduced by the use of averages.

The **Homogeneous Ensemble Approach** is a mixture of the same classifier forms. But for each classifier, the dataset is different. This would make the unified model operate more precisely after aggregating the effects of each model. That Ensemble System form work for a wide number of datasets. In the homogeneous process, the method of selection of features is the same for the separate training data. Computationally, it's costly.

The **Heterogeneous Ensemble approach** combines multiple classifiers or machine learning techniques, each of which is trained on the same data. This strategy is effective for smaller data sets. In heterogeneous terms, the process of picking a feature with identical training data is different. The Ensemble Approach achieves its cumulative outcome by aggregating the results of each integrated model.

2.7.2 Types of Voting Ensembles

The Voting Ensemble (or "Majority Voting Ensemble") is a machine learning ensemble that combines the predictions of multiple models. It is an approach which can be used to enhance model performance, particularly to the point where it outperforms all other models in the ensemble. The voting ensemble algorithm works by combining the forecasts of multiple models. It is suitable for classification and regression tasks. This involves estimating the average sample predictions in the case of regression. When it comes to classification, the guesses for each category are summed together and the label with the most votes is chosen.

The Voting Ensemble approach is a powerful tool that comes in handy when a single model exhibits bias [36, 37, 38]. Additionally, the Voting Ensemble may produce a higher overall score than the best base estimator, as it aggregates the predictions of different models and attempts to compensate for the individual models' flaws. Making the base estimators as diverse as possible is one technique to increase the ensemble's efficiency.The base estimators will be different pre-trained models, fine-tuning in the same dataset as shown in Figure 2.8.

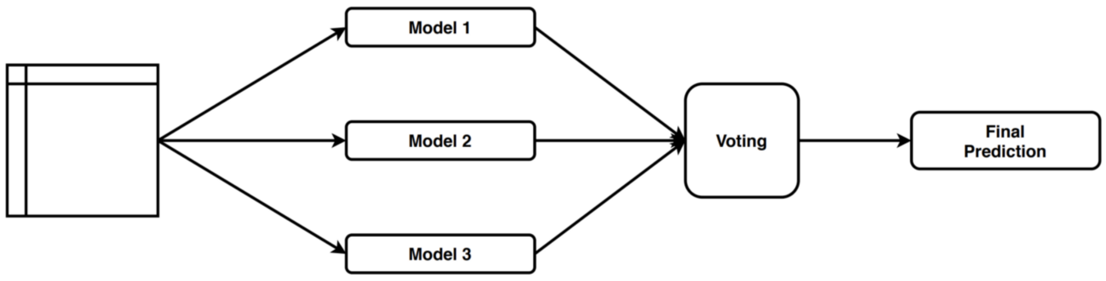


Figure 2.8: Ensemble Learning Voting Classifier

Two different voting schemes are typical among voting classifiers:

- **Hard Voting:** Every single classifier will vote for a class, and the majority will win. In mathematical terms, the desired target label of the set is the mode of distribution of independently predicted labels.
- **Soft voting:**Each classifier assigns a probability to each data point that it belongs to a particular target class. Predictions are summarized and weighted according to the classifier value. The vote is then cast on the preferred label with the highest weighted probability.

The voting ensemble does not ensure that its results will be superior than those of any other model utilized in the ensemble. If any proposed method in the ensemble outperforms the voting ensemble, it is assumed that it will be used instead of the voting ensemble. A voting ensemble is incredibly beneficial for machine learning models that use stochastic learning methods and generate a unique final model each time they are trained on the same dataset. For instance, neural networks that use stochastic gradient descent to find the optimal solution. Another example in which voting ensembles are particularly effective is when numerous instances of the same machine learning algorithm are combined with slightly varied hyperparameters.

2.7.3 Extension of Voting Ensemble

The disadvantage of the voting framework is that it treats all models equally, which implies that every model has contribute equally to prediction. This is a problem if certain models perform poorly in some situations but perform admirably in others. To address this issue, a voting ensemble extension that utilizes a weighted average or a weighted voting system for the contributing models has been proposed in the literature. Typically, this is called *blending* [12]. Another expansion is to use a machine learning model to determine when and how much to assist each model when making predictions. This is refereed as stacked generalization, *stacking* [39].

Voting ensemble extensions::

- Weighted Average Ensemble (blending)
- Stacked Generalization (stacking)

Stacking

Stacking (or stacked generalization) creates a series of models employing a variety of learning methodologies (e.g., one neural network, one decision tree, ...), in contrast to bagging or boosting, which train multiple versions of the same learner (e.g. all decision trees). For example, think of the scenario where m models are trained on a dataset of n samples. We intend to train m binary classifiers h_j sequentially in order to later combine them for the purpose of selecting new instances x as their weighted majority vote :

The model outputs are used to calculate the final prediction for any instance x:

$$\hat{y}(x) = \sum_{j=1}^{m} w_j \cdot h_j(x) \tag{2.25}$$

where the introduced a level-1 **meta-learner**, for optimizing the weights w_i of the level-0 base learners. That is, the individual m predictions associated with each training instance x_i are forwarded as training data to the level-1 learner, as shown in figure 2.9.

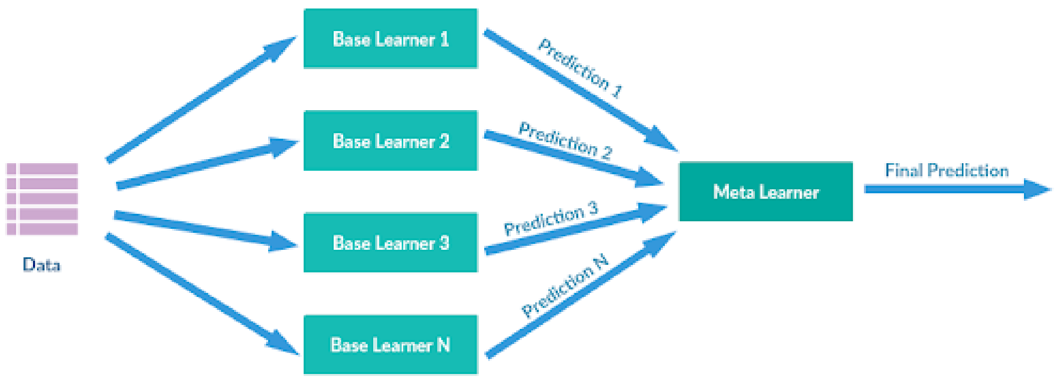


Figure 2.9: Stacking Architecture

2.7.4 Opinion Aggregation and Voting Mechanisms

Weighted voting games originated in the areas of *computational social choice* [40], and *cooperative game theory* [41].They model decision-making scenarios in which a set of electors must make a binary (yes/no) decision on a specific issue; a numeric weight is assigned to each elector and a decision is made if the total of the weights of the electors in favour of it matches or exceeds a certain threshold called a *quota*. Weighted voting games have many implementations outside the principle of social choice. For example, they can be used to model settings where each player has a certain amount of a given resource (say, time, money, or manpower) and there is a target that can be accomplished by any coalition that has a certain amount of that resource [42]. Weighted voting games can be viewed as a method of representation for yes/no voting schemes (i.e., simple cooperative games). They provide a condition on the yes/no voting system that is both necessary and adequate to ensure that such a system can be interpreted as a weighted voting game.

Social choice theory is a theoretical framework for analyzing the process of merging individual preferences and opinions in order to attain a collective decision or, in some cases, social welfare. The analysis begins with a set of seemingly plausible axioms of social choice and builds a social welfare function [43] using components of formal logic for generality.

2.8 Evaluation Metrics

1. **k-Fold Cross-Validation:** The process takes a single input, k, which specifies the number of groups to separate a data sample into. As such, it is frequently referred to as k-fold cross-validation. When a specific number for k is chosen, it may be used in place of k in the model comparison, such as k=*10* for a ten-fold cross-validation. Cross-validation is primarily used in applied machine learning to measure a model's capacity to learn from previously unseen data. It is a common strategy since it is straightforward and generally results in a less biased or positive estimate of the model's ability than other methods, such as a basic split train/test. The following is the primary way [44]:

   - Randomly shuffle the dataset.
   - Divide the sample into k distinct groups
   - For each individual group:
     - Consider the group as a stand-in or test data set.
     - Utilize the remaining groups as a source of training data.
     - On the training set, fit a model and evaluate it on the test set.
     - Keep the evaluation score and throw away the model
   - Summarize the model's ability by examining a sample of model evaluation scores

2. **Classification report through sklearn:** A classification report is used to evaluate the accuracy of an algorithm's predictions. How many of your guesses were

correct and how many were incorrect. There are four distinct ways of determining whether or not the forecasts are accurate:

(a) **TN / True Negative:** when a case was negative and predicted negative

(b) **TP / True Positive:** when a case was positive and predicted positive

(c) **FN / False Negative:** when a case was positive but predicted negative

(d) **FP / False Positive:** when a case was negative but predicted positive

- **Precision** refers to a classifier's ability to avoid labeling a negative instance as positive. It is defined as the ratio of true positives to the sum of true and false positives for each class. Out of all the positive classes we have predicted correctly, how many are actually positive. $Precision = \frac{TP}{(TP+FP)}$
- The **Recall** of a classifier refers to its ability to identify all positive examples. It is described in terms of the ratio of true positives to the sum of true positives and false negatives for each class. How much we predicted accurately out of all the positive classes. $Recall = \frac{TP}{(TP+FN)}$
- The **F1-score** is a weighted harmonic mean of *precision* and *recall* such that the best score is 1.0 and the worst is 0.0. Generally speaking, f1-score is a quantity that tries to combine precision and recall in one number to penalize models that despite having high accuracy for example have low recall for one class. To compare classifier models, the weighted average F1 should be used as a guide, not the global accuracy. $F_1 - Score = 2 * \frac{(Recall*Precision)}{(Recall+Precision)}$
- **Support:** The number of real occurrences of the class in the provided dataset is called support. Support does not vary between models; rather, it serves as a diagnostic tool for the evaluation process.
- **Accuracy:** How frequently does the classifier get it right on average. $Accuracy = \frac{(TP+TN)}{(Total)}$
- The **Macro-average** technique can be used to determine the system's overall performance across data sets. $macro\text{-}avg = \frac{(\sum_{n=1}^{\#classes} P_n)}{(\#classes)}$
- **Weighted-average:**Calculate metrics for each label and determine their weighted average in terms of support (the number of true instances for each label).

3. **Confusion matrix:** A confusion matrix is a set of outcomes from classification problem predictions. The number of correct and wrong predictions is reported using count values and broken down by class. The confusion matrix demonstrates how confused the classification model becomes when making predictions. It provides insight into not only the classifier's errors, but also the types of errors made.

4. **Sensitivity-Specificity Metrics:**

   - **Sensitivity**refers to the quantity of positive prediction and summarizes the accuracy with which positive classes are forecasted. $\textit{Sensitivity} = \frac{TP}{(TP+FN)}$
   - **Specificity** relates to the predictability of the negative class and is a complement to sensitivity, or the true-negative rate. $\textit{Sensitivity} = \frac{TN}{(FP+TN)}$

   Sensitivity is a more relevant aspect in cases of imbalanced classification.

5. **Probabilistic Metrics for Imbalanced Classification:** Probabilistic metrics are structured for quantifying the uncertainty of predictions. These are important for problems where we don't care whether the model is right or wrong, but want the model to correctly reflect the ambiguity in its forecast. Probabilistic definitions of error, including probability, are used to calculate the divergence from the true probability. These indicators are particularly helpful when we want to determine the efficiency of the classifier, whether it has misidentified the correct class with high or low probability [45]. The *Brier score* is a popular measure of predicted probabilities:

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2 \tag{2.26}$$

   where f_t is the predicted value and o_t is the observed. N is the number of observations. Brier score measures the mean square error between the estimated probabilities and the observed values (actual).

   The advantage of the Brier score is that it is organized in such a way that it favors the positive over the negative. The Brier score is calculated by evaluating the model's ability to predict the positive class and its accuracy in doing so. The mean squared error is calculated as the average of all squared errors between two numbers.

2.9 Related Work

The sections that follow explain several advancements in the classification of recyclable materials using machine learning algorithms, most notably CNN approaches, to provide context for the current work.

2.9.1 CNN Techniques for Waste Classification

A study conducted by Rahmi Arda Aral, Seref Recep Keskin and others explored different well-known Deep Learning models an after fine-tuned them to find the most efficient approach [46]. They implemented data augmentation due to relatively few images available for boosting classification accuracy. After tried different optimizer and DL architecture, they achieve 95% with Adam optimizer.

In CompostNet [47] is a waste classification system which utilize a deep learning network to classify compostable, recyclable, and landfill materials. The system provide two different design approaches, a custom model and one augmented pre-trained image classification model (MobileNetV2). The novelty of this approach is that come to an app version which enable the user to take a picture of any material and identify the correct class for better sorting in the waste receptacles.They demonstrate that even with a small dataset, a model based on transfer learning is capable of accurately classifying waste pictures.

Another similar application is the SpotGarbage [48] which employ a pre-trained Alexnet network modify to output a binary probability to classify a material can be either garbage or not-garbage. In details, they introduce an Android app, which employs a CNN called GarbNet to not only detect but also localize garbage in real world images. The authors proposed an optimization in the network architecture.

A Multi-layer Hybrid approach which uses Deep Learning models to identify wastes as recyclable or others material [49] are utilized a CNN-based algorithm for extracting image features and several multi-layer perceptrons (MLP) to reinforce these images features with other features information. A 90% classification accuracy is achieved under two distinct scenarios in comparison with only image classification CNN-based method.

The goals of this study [50] were to identify a wide range of recyclable materials, which affects waste management and recycling systems. Using intelligent systems instead of humans to sort recycled materials is an important necessity for an economical and sustainable recycling industry. In order to ensure efficiency, they worked with well-known deep neural network architectures. The suggested model, *RecycleNet*, is a meticulously designed deep convolutional neural network model optimized for recycling object classification. This unique model reduces the number of parameters in a 121-layer network known as DenseNet121 from seven million to just three million.

2.9.2 Ensemble Techniques for Classification

In this work [51] an architecture for remote Sensing Data Classification proposed using feature fusion, referred as two-tunnel CNN framework. In details the authors integrated two different CNN-based method train in separate datasets by concatenate the final outputs layer from each branch conclude to a new join layer that combined different features. Experiments show that this approach outperform the existing method.

In the work of Hinata and Takahashi [52] a complicated network approach refer as EnsNet consists of one main CNN and numerous Fully Connected Sub-Networks are proposed for classification purposes. All the Sub-Networks with the base CNN construct a feature map in the last convolutional layer, and a majority vote rule are in charge of the final selection of class. All SubNetorks train independent of others ant take part in an ensemble in final layer. Experiments shown that achieves the lowest error rate among some of the state-of-the-art models.

A novel approach present ii this work [53], they describe a One-vs-One (OvO) training methodology for neural networks that trains each output unit to differentiate between a specific pair of classes. Additionally, as compared to the One-vs-All classification scheme, this approach results in a greater number of output units. To determine the suggested approach's advantages, they compared it to the results of One-vs-One and One-vs-All class classifiers on three different plant recognition datasets and a dataset including photos from several monkey classes. Two deep convolutional neural network (CNN) architectures are developed from scratch or using pre-trained weights: Inception-V3 and ResNet-50. This experiment demonstrates that when all CNNs are generated from scratch, the One-vs-One classification outperforms the One-vs-All classification. However, fine-tuning the two pre-trained CNNs using the One-vs-All system gets the best results, as each of them was fine-tuned using the One-vs-All scheme.

2.9.3 Other Techniques

In the work [54] presents a triple-histogram classification model for the classification of plastic waste. The key objective of this paper is the automated segregation of waste. A gradient based features vector will be used to differentiate between four different classes: PE-HD and PET, PS, and PP.

In the approach of [55] the authors seeks to describe the four types of garbage waste comprised of glass, paper, metal and, plastic. They use a garbage data set that includes 400 samples of each class. The models used in the experiment are Pre-trained VGG-16, AlexNet, Support Vector Machine, K-Nearest Neighbor and Random Forest (RF). In their study present results that their models achieved an accuracy of 93%.

In the study of [56] the authors presents a method for automatically sorting two types of materials: polycoat containers and PET (Polyethylene Terephthalate) bottles. This article presents a high-speed approach for automatically recognizing regions inside a picture that may include these elements and extracting such parts. These regions are merged to form complete containers, which are classed as polycoat or PET bottles. The model is trained on the histogram of image pixels using a linear support vector machine (SVM). The proposed approach achieved a recognition rate of 93% and is capable of real-time operation at high frame rates utilizing a field-programmable gate array.

Chapter 3

Dual Branch Multi-output Approach

The classification for the achievements of recyclable materials, and in particular plastic, is critical both for the economy and for the achievement of climate balance. In this chapter we present a novel image classification model that can be used to distinguish recyclable materials, building on recent work in deep learning and waste classification.

The approach that we introduce is the so called "Dual-branch Multi-output CNN", a custom convolutional neural network composed of two branches aimed to i) classify recyclables and ii) distinguish the type of plastic. In particular, our approach makes use of the joint utilization of the available datasets, making it possible to learn disjoint label combinations. The proposed architecture is composed of two independent classifiers operating on two different datasets. In our work, the Densenet121, ResNet50, VGG16 architectures [57, 58, 59] were used on the Trashnet dataset [2], along with data augmentation techniques, as well as on the WaDaBa dataset [3] with physical variation techniques. Several experiments are used to assess their effectiveness in the classification of waste material images.

3.1 Overview and Contributions

The aim of this study is to develop an algorithm for the automated classification of potentially recyclable waste items, a task that is challenging due to the condition (e.g. degree of dirtiness, shape deformations, etc.) of the items during their sorting in waste treatment facilities. A second objective is to better identify the form of plastic used, as this type of plastic comprises a number of different chemical mixtures (e.g. PET, PP, PS, HDPE, etc.). Additionally, the fact that current methods of optical separation of recyclable waste face difficulties in distinguishing all recyclable materials at once without using hyper-spectral imaging [56, 55, 54], provides a stronger incentive for our work. Our long-term goal is to increase the productivity and revenue of waste treatment plants. To this end, we build on recent advances in deep learning neural net architectures [60].

In detail, we receive images of each recyclable material and annotate each one with two labels. The first label distinguishes one of five objects-namely glass, paper, metal, plastic, and trash. The second label categorizes specific "plastic subclasses", that is, PET, PE-HD, PP, PS and a non-plastic. The final output is the characterization of the material with one of these two labels.

To accomplish this classification task, we build a novel Dual-Branch Parallel CNN architecture. In our study, we combine numerous features from various data sources, which is useful for classification, and the joint utilization data helps make this possible. A crucial factor is that a combination of the Trashnet dataset (which contains glass, paper, metal, plastic and trash materials) [2] and the WaDaBa dataset [3] (which contains different labelled forms of plastic, but supplemented with non-plastic material in our case) would lead to a more robust model, one with potentially higher classification accuracy compared to using Trashnet alone.

As such, our Dual-Branch Multi-output CNN approach features two different classifiers, each one trained on unique branches. One branch is trained with the recyclable waste materials (from Trashnet), and the other with images of plastic types (from the augmented WaDaBa). We follow the classifications of the recyclable waste classifier, but increase our confidence on those classifications or revise them entirely based on the classification of the plastic classifier. In this sense, if the "Trashnet branch" has a large "confidence" (probability) of classifying an object as "plastic" and the "WaDaBa branch" assigns a large probability to a certain type of plastic, then we may favor this particular class of plastic with increased confidence. Similarly, if the "Trashnet branch" predicts that any material is in the non-plastic class (e.g. "glass"), and the "WaDaBa branch" confirms this prediction by agreeing that this material is "non-plastic", then we have greater confidence in the prediction. On the other hand, if the "Trashnet branch" predicts a non-plastic class with low probability p and the "WaDaBa branch" predicts a certain plastic form with a probability greater than p then we use the information from the "WaDaBa branch" as the basis for our classification. The last case is when the

"Trashnet branch" predicts a plastic class with a probability p and the "WaDaBa branch" predicts a non-plastic form with a probability, q then we compare the probabilities q and p and if p ("plastic") is higher than q ("non-plastic') we choose the plastic class for the final label, and respectively the non-plastic in the opposite case.

Since the "Trashnet branch" (i.e., the waste classifier) of the Dual-Branch architecture plays a vital role in the initial material detection, it consists of a *Stacked ensemble of CNNs* attempting to achieve better accuracy with less false predictions. By integrating different architectures [61], Stacked Ensemble learns the optimal combination of predictions from a variety of well-performing machine learning models as explained in subsection 2.7.3.

Our problem formulation shares certain characteristics with the incremental learning of networks from diverse datasets that cannot be brought together. This is the case, for instance, in medical multicenter datasets of different modalities, or of the same type but addressing different properties of the population. In these domains, the concept of shared wisdom from the data forms the trade-off between learning efficiency and privacy level [62, 63]. Towards this direction, the concept of Federated learning implements different training of networks at each data site and then performs an averaging combination of these networks at the final central point [64]. Alternatively, the concept of Split learning, splits a neural network into multiple components, and the trains them separately at a different site, up to the level of feature extraction, whereas all networks are combined towards a final centralized classification network [62]. Our approach of Dual-Branch learning builds on these paradigms and formulates different classifier networks as in Federated learning, with adaptive combination at the classifier label, similar to Split learning.

We compare our approach to a framework that employs two separate CNNs, each involved in a different classification assignment, which are run in sequence. We conduct comprehensive experiments on our benchmark datasets, which demonstrate that our proposed network achieves over 97% accuracy, and in some cases manages to exceed the accuracy results of state-of-the-art methods for our benchmark datasets.

Our contributions in this chapter can be summarized as follows:

- We expand on the concept of shared wisdom from data and explore how various datasets can be combined to improve accuracy and create a stable network architecture.

- We introduce a novel two-branches multi-output CNN, which integrates multiple source information at the final level, for recyclable waste classification. The need for this component arises from the need to address and exploit the peculiarities of the data.

- We show that our method results in a major increase in efficiency of the main constituent branch of the network. In addition, material classifier accuracy is greatly improved by leveraging the output of the plastic classifier.

- Moreover, our classifiers can substantially outperform the performance of state-of-the-art networks operating on our benchmark datasets.

- The success of our approach effectively highlights the importance of exploiting emerging information for ongoing action.

3.2 Proposed Classification Framework

Our proposed classification framework is the Dual-Parallel CNN for waste Classification, shown in Fig. 3.1. This includes a Stacked ensemble of CNNs branch for waste sorting and a classic CNN branch for plastic classifying. In our experiments we pit this against a CNN pipeline, where explicitly after the first CNN predicts waste classes, the second CNN predicts the form of plastic 3.3.

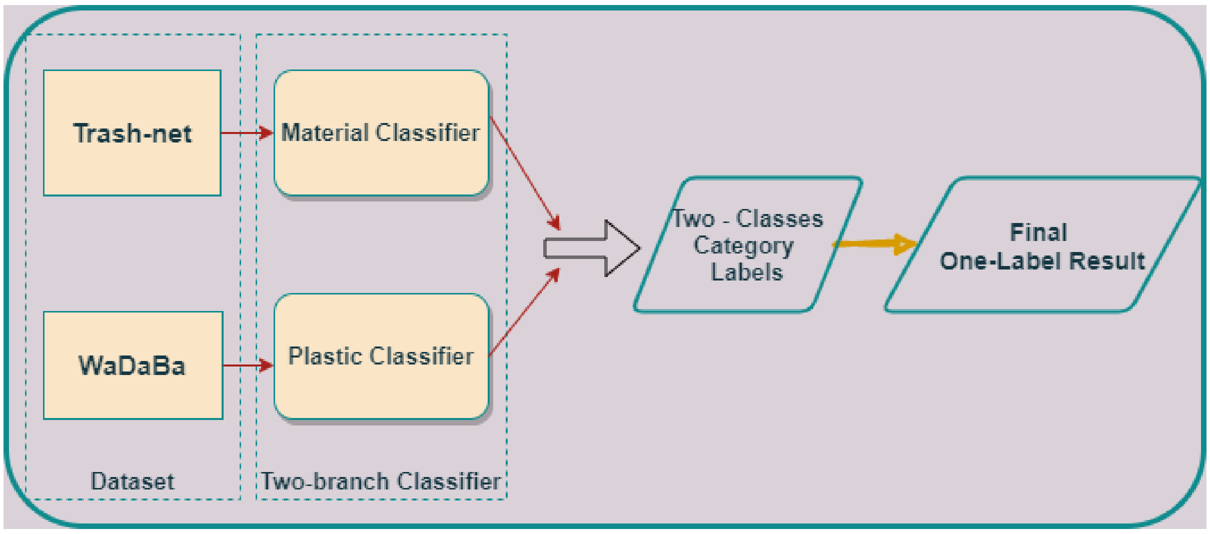


Figure 3.1: Dual-Branch Architecture

3.2.1 Dual-Parallel CNN for recyclable waste classification

The CNN architecture we put forward in this work, consists of two branches corresponding to two independent classifiers[4], depicted in Fig. 3.1. The first of these is a dedicated branch for a plastic classifier trained with the WaDaBa dataset, albeit supplemented with an extra class consisting of non-plastic images from the Trashnet dataset for the classifying four types of plastic or not plastic at all. For a more comprehensive vision of the network, we use a CNN network which has been pretrained on ”Imagenet”; and then replaced the final fully connected layers with 2 dense layers of 512 neurons. The output of the convolutional layer is passed through a *softmax with temperature scaling* Equation 2.24 activation function. The final network is fine-tuned using the WaDaBA images samples.

The second branch is an independently trained Stacked Ensemble of different CNN architectures, specializing in classifying (potentially recyclable) waste into five classes. This branch was trained on Trashnet. The base-learners are 3 different pretrained CNNs (DenseNet121, ResNet50, VGG16). The outputs of each model can then be combined.

[4]The network branches at least twice (sometimes more) in the multi-output classification to create several sets of fully connected heads at the network's end. The network then predicts a series of class labels for each head, allowing it to learn label combinations with disjoint labels. We have at least two heads attached to the same body, each in charge of a separate, specialized classification task.

We use a straightforward merge of concatenation, where a single 15-element vector is generated from the five class probabilities predicted by each of the three models. We then establish a hidden layer to translate this "input" to the meta learner and the output layer to produce its own probabilistic prediction. Finally, we create a stacked neural network generalization model based on a list of trained sub-models, Fig. 3.2.

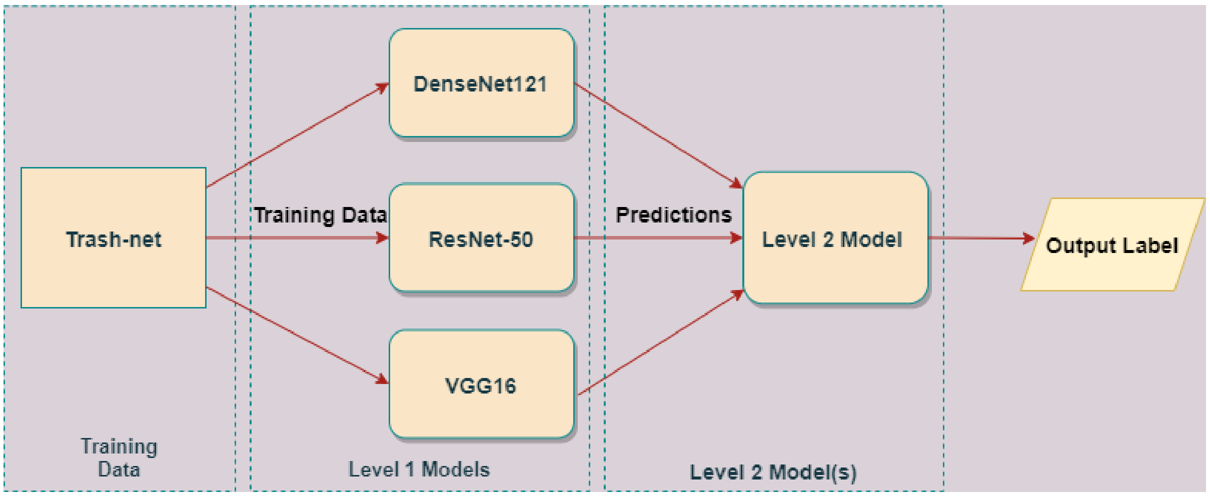


Figure 3.2: The "Material Branch" - Trained in Trashnet

In each branch, a series of convolution, activation, batch normalization, pooling and drop-out operations are performed until the final output is reached. The branch at the bottom of the network is a slightly shallower (not as deep) than the branch at the top of the network (Fig. 3.1). Predicting plastic type is much easier than predicting the waste category and therefore the plastic branch is shallower.

Now, it is important to assess the transfer learning capability of a given architecture. To test transfer learning , in one of our experiments we created a mixed test set from the two Datasets for our Dual-Branch CNN. In detail, the testing set is fed with images from five classes: (a) glass, (b) paper, (c) metal, (d) trash, (e) plastic. These consist of image samples from the Trash-net dataset except the plastic class which take samples from WaDaBa dataset. The purpose of DualBranch CNN is to test the transfer learning capability of this architecture from one dataset to another.

Our experimentation showed that transfer learning does not reach in this particular case the performance of the separately trained classifiers. The reason behind this is that the "material classifier" branch (or "Trashnet branch") is trained with Trashnet images that may be labeled as “trash”, while they depict plastic, and are thus similar to the plastic images originating from the WaDaBa dataset. Naturally, the Trashnet branch decides that these are “trash”, as shown in the confusion matrix we report. This phenomenon mirrors a wider, key problem faced in transfer learning: namely, the challenge to be able to classify correctly images that might look very similar to training data that originated from a class with a different, misleading label.

To counter this problem, we equip our Dual-Branch CNN with an additional step performed after we receive the two-labels output. In this step we exploit the known fact that we have a classifier specialized in plastic separation, and use this knowledge to increase our confidence in the “material class” (Trashnet branch) predictions, or even revise these completely, given the output of the “plastic” classifier (WaDaBa branch).

Specifically, we convert the double-labeled result of the proposed system into a final one-label result as follows. If the “Trashnet branch” has predicted “plastic”, and the “WaDaBa branch” gives a greater probability for a certain type of plastic, then we assign this probability as the probability of “plastic” - and predict that particular type of plastic. However, if the “Trashnet branch” has predicted any other particular class with probability p, but the “WaDaBa branch” predicts a certain type of plastic with a greater than p probability, then we depend our final classification on this “assumed-to-be-expert-on-plastic” prediction, and assign the material to be classified in the corresponding “plastic type” class predicted by WaDaBa.

Finally, if the “Trashnet branch” has predicted the item is in some particular non-plastic class with probability q, while the “WaDaBa branch” has assigned the item a probability $q' \geq q$ for it to be in its “non-plastic” category, then we are more confident in identifying the material class as the one predicted by the “Trashnet branch”, and produce exactly that class as our final prediction for the input image.

3.2.2 Training Process

First, the distinct branches of the Dual-Branches architecture are trained on their separate data using fine-tuning that requires a pre-trained model. Fine-tuning can configure the weights of the pretrained model to minimize the computation time.

The Stacked Ensemble CNN material classifier and the CNN plastic classifier are first trained on individual training set. More precisely, as defined in Algorithm 1, two separate

Algorithm 1 Training process of the Dual-Branch CNN for the classification of waste branch

```
1: Initialize all weights
2: for #epochs
3:     Stage 1:
4:     • Train the material classifier branch
5:     • Train the plastic classifier branch
6: for #epochs
7:     Stage 2:
8:     • Merge the separated trained models
9:     • Test two-branch CNN
```

branches are first trained in stage 1 using a large learning rate (lr=0.01). When the two components are combined in stage 2, the pre-trained models extract the respective classes from the training data pairs and we obtain the two-class output labels.

Additionally, we used data augmentation techniques in order to generate samples of images and thus enrich the initial training dataset. The augmentation transformations we use are a) random flipping, b) random rotating, c) gaussian distortion, d) shearing and skewing, e) random zooming in order to render variations within the dataset, so it can correctly generalize the unseen data. All data is scaled to 0-1 order to expedite the convergence time. On the other hand, fine-tuning is typically used to transfer a pretrained model for large-scale data to small-scale and related data. In consideration of the characteristics extracted from each branch which are relevant for a particular branch, we use the transfer learning strategy of [65] to train the two separate branches in a specific and very similar training set.

Finally, we compare the proposed Dual-Parallel CNN with the sequential implementation of the two CNN networks. The first network is used for material classification and the second one for the plastic type categorization as shown in Figure 3.3.

3.3 Experimental Setup

In this section, we will discuss how we collect data, set up the environment, and construct the CNN architecture that we proposed using Tensorflow and Python.

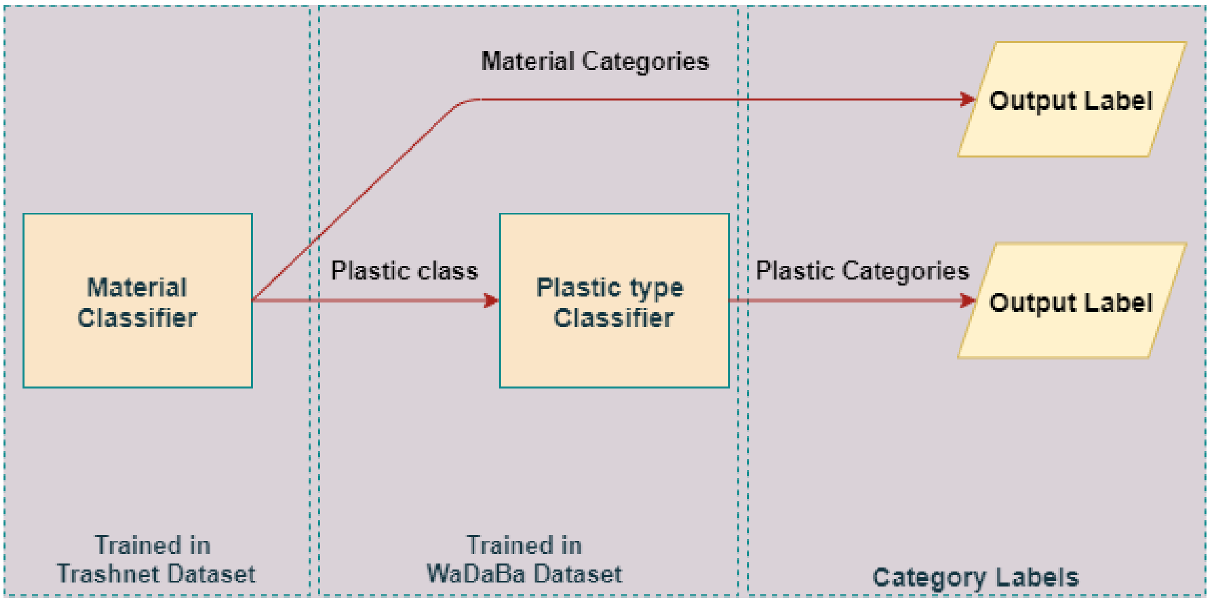


Figure 3.3: Sequential CNN Architecture

We experiment to the identification of recyclables materials, using two datasets. The approaches is decomposed in 4 parts:

- Data Pre-processing
- Evaluation Metrics selection and Fine Tuning
- Classification Framework Design
- Training and Testing of the selected method

In the below subsections we describe the datasets, metrics, and parameter fine tuning used in our work here,

3.3.1 Experiment Datasets

As mentioned above, we test our network on two datasets:

1. Plastic Waste DataBase of Images - WaDaBa
2. Augmented Trashnet Dataset

Plastic Waste DataBase of images WaDaBa

The WaDaBa consists of 3960 images obtained after transformations, such as change of perspective, damage effect, and light sources shifting. The specific categories of plastic waste are the: a) PET, b) PE-HD, c) PP, d) PS [3].

Augmented Trash-net dataset

The Trash-net data collection consists of images of six classes: glass, paper, cardboard, plastic, metal and trash. The dataset currently consists of 2527 images [2]. We picked five labels for classification, i.e. Paper, Plastic, Metal, Glass, Trash and used data augmentation that resulted to a dataset of 10,000 images equally divided among the 5 classes.

3.3.2 Evaluation Metrics selection

In order to examine the efficiency of the proposed two channel parallel CNN for recyclable waste classification in comparison to the sequential CNN classifier we take into account the number of feature maps and the fully connected layers in the network. Each classification prediction of the proposed system is compared to the manually classified label, which is described as the "truth". The efficiency of the proposed model is evaluated with accuracy, precision and recall. Cross-validation is one of the most commonly used data re-sampling techniques for calculating the true model prediction error and tuning the model parameters. In our study, we used 4-fold cross-validation. All the aforementioned metrics described in section 2.8.

A Classification report is used to measure the quality of predictions from a classification algorithm. In addition, we report confusion matrix for the experiments. There are four ways to check if the predictions are right or wrong: 1) precision, 2) recall, 3) f1-score, 4) accuracy.

Parameter and Fine Tuning

Classification performance is closely linked to the design architecture of the deep learning network. We tried three different architectures for the CNN network of our "WaDaBa (plastic) branch". Specifically, we tried the Visual Geometry Group network (VGG16) [13]; the Densely Connected Convolutional Network [16]; and the Deep Residual Learning for Image Recognition [15]. The bottom three layers, i.e. full connection, classify, and data input layers, are removed following the "fine-tune" approach described in subsection 2.6.3 from Transfer Learning Strategies.

The rate of learning is one of the factors that determines the convergence of speeds that may affect the performance of training. The learning rate is set at 0.01 with Adam's policy [14]. Different learning rates were tested on WadaBa dataset and our empirical study showed that the accuracy of classification may not be improved if a higher learning rate is used. Since the fine-tune strategy can significantly reduce the complexity of the computation, the low-level features are extracted by pre-trained layers, which then fit into the custom layers for the task of classification. In fact, fine-tune helps to achieve higher classification accuracy and to build a more robust network.

3.3.3 Evaluation Performance

We will present in this section the different evaluation metrics of the two approach that we use and the measurements that we display. There are 2 types of CNN Networks:

- **Dual_Branch_CNN**: consists of two independent branch (classifiers) that fuse in the final level and provide one final output.
- **Sequential_CNN**: consists of two CNN (classifiers), operating sequentially, with a double outcome at the end.

We choose some metrics to evaluate the performance of the classification models in each fold (we use 4-fold validation) which were described in section 2.8 of training period in the validation set:

- Accuracy
- Precision
- Recall
- f1-score
- Sensitivity
- Specificity

3.3.4 Implementation in Tensorflow

Our coding employed Python and Tensorflow through Keras high-level application programming interface. The Tensorflow is an open source machine intelligence library that uses a dataflow graph. Keras is written in Python and is capable of running at the top of Tensorflow and Theano, which focuses on allowing fast experiments. All tests are conducted on Windows 10, Intel i7-9700, 32 GB ram, and Nvidia GTX 1080 GPUs.

3.3.5 Data augmentation

We implemented python *Augmentor* [66] package for data augmentation, increasing the number of data for the Trashnet and WaDaBa datasets. The following general procedure is followed:

- We instantiate a Pipeline object which contain the initial image data set.
- Define a number of transformations to perform on this data set (operations are added sequentially in order to generate a pipeline).
- Augmentor applies operations to images stochastically as they pass through the pipeline, according to a user-defined probability value for each operation.

Augmented Operations

The operations we applied in our Datasets are the following: *a) Perspective skewing:* includes changing the image such that it appears to be viewed from a different viewpoint, *b) Elastic distortions:* allowing to distort an image while keeping the aspect ratio of the image intact, *c) Rotating:* simply rotate the images by arbitrary degrees, *d) Mirroring:* mirroring images (translating them through the x any y axes),

3.3.6 Construction of Fine-tuned Pre-trained CNN

Dual-Branch architecture consists of two Independent Branches (classifiers). As mentioned earlier, one branch consists of 3 fine-tune pre-trained CNN architecture, we tested the performance of three architecture in this branch. In order to do this, we pre-trained each one of them for material classification. The second branch consists of a single pre-trained CNN. Every one of these fine-tune pre-trained architecture follow the below process during training.

The Process 2 is shown the steps which required to train each of the CNN architectures used in the material and the plastic branches.

Process 2 Fine-tuned Pre-trained CNN Design

```
1: Download a CNN pre-trained on the ImageNet dataset
2:     Fine-tune Stage:
3:     • Remove the fully connected nodes at the end of the network
4:     • Replace the fully connected nodes with newly custom one following by a "Relaxed
       Softmax" activation function.
5:     • Initialized the first CONV layer with pre-trained weights
6:     • Start training all the network.
7: Call Adam Optimizer to minimize the loss function
```

3.3.7 Entire Framework Implementation

We sum up our approach in this area and describe the classification Framework proposed. Our classification approach is based on a fine-tuned pre-trained CNN with models fusion as an *ensemble approach*. Process 3 described all the steps building the Unified proposed Architecture.

For testing the Dual-Branch CNN network we construct a unique hold-out dataset consists of images from the 2 dataset, with images that none of the 2 have seen during the training stage

In "Trashnet" branch, we integrated 3 CNN architecture, namely ResNet50, DenseNet121 and VGG16. On "WaDaBa" branch, we employ a VGG16 architecture.

3.4 Results

We evaluated the three CNN architectures for the "plastic branch" with images from the four plastic classes in the WaDaBa dataset, and from an extra class for non-plastic samples taken from Trash-net. The number of samples per class is displayed in Table 3.1. The CNN models' accuracy is demonstrated in Table 3.2. DenseNet121 is the

Process 3 Dual-Branch CNN Implementation

```
 1:     Material Classifier Training Stage:
 2: for $e = 0$ in range ($n\_models$) do
 3:     Trained (using Process 2) and save a number of models on “Trashnet”
 4: end for
 5: Load all the models
 6: Construct stacked ensemble model integrating the load models
 7: Train stacked ensemble model on “Trashnet” dataset (Material Classifier).
 8:     Plastic Classifier Training Stage:
 9: Train a CNN model on “WaDaBa” dataset (Plastic Classifier).
10: Merge the independent Branches constructing the unified network
11: Test on the Dual-Branch network on hold-out dataset.
```

network exhibiting the highest accuracy. Note that the accuracy of all models surpasses the 75.68% accuracy of the only known-to-date model used on WaDaBa [54]. We use 4-stratified fold validation for generalization. A confusion matrix for DenseNet121 on the augmented WaDaBa dataset described above is shown in Fig. 3.4 and the respective Confusion matrix on Table 3.3. We observe there that almost all instances from each actual class are mapped to the corresponding predicted classes.

Class	Number of samples	
Name	Train	Validation
PE_HD	1650	550
PET	450	152
PP	480	160
PS	390	130
non-plastic	1264	316

Table 3.1: Number of Train/Val Samples for the WaDaBa Dataset

Deep Learning model	4-fold accuracy
DenseNet121	99.33% (+/- 0.17%)
VGG16	99.25% (+/- 0.30%)
ResNet50	99.12% (+/- 0.17%)

Table 3.2: Accuracy for the CNN models on the WaDaBa Dataset

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
PET	99%	100%	94%	550	99%	100%
PE_HD	100%	99%	95%	152	100%	98%
PP	96%	97%	98%	160	99%	96%
PS	98%	97%	97%	130	99%	96%
non_plastic	100%	100%	100%	316	100%	100%
accuracy			99%	1308		
macro avg	99%	98%	98%	1308		
weighted avg	99%	99%	99%	1308		

Table 3.3: Classification Report for DenseNet121 architecture on WaDaBa

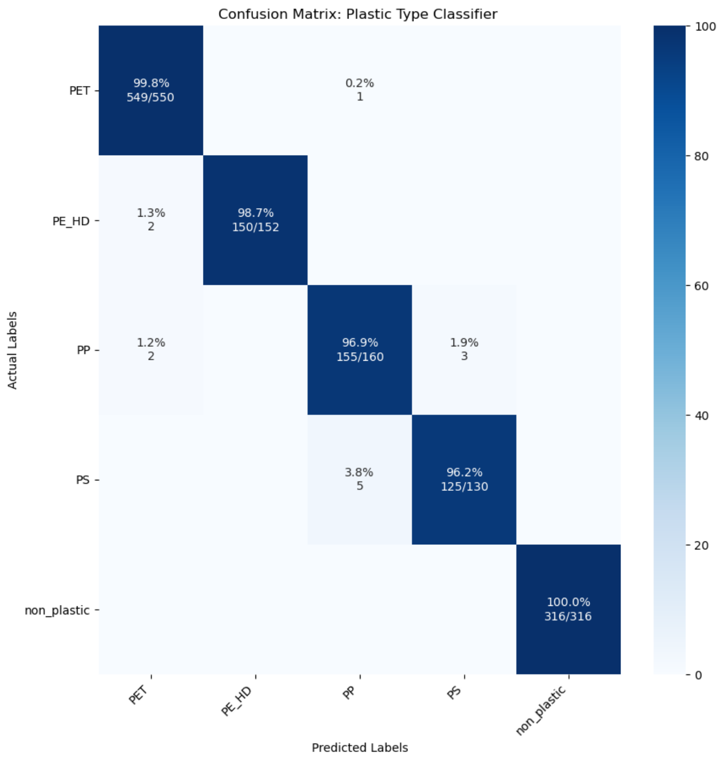


Figure 3.4: Confusion Matrix for DenseNet121 on WaDaBa

Now, in order to test the Stacked Ensemble CNN used for the “Trashnet (material) branch”, we use 87.5% of the samples in the augmented Trashnet dataset for training and testing the base learners, as shown in Table 3.4; and keep the remaining 12.5% as a hold-out set for testing the Stacked Ensemble CNN. We test the pre-trained base learners, and the performance of each of these architectures is shown in Table 3.5. We then evaluated the “Trashnet branch” model to assess whether it can exhibit a better performance. Its accuracy is shown in Table 3.6, and the corresponding confusion matrix in Fig. 3.5. Here again, the algorithm provides a good match between the actual and the predicted class for each instance, while the 4-fold cross validation accuracy remains high to each one of the three proposed DL models. We note that the Stacked Ensemble model’s accuracy surpasses, by more than 2%, several state of the art models [46, 50] applied on Trashnet.

Class	Number of samples	
Name	Train	Validation
Glass	1500	500
Metal	1500	500
Paper	1500	500
Plastic	1500	500
Trash	1500	500

Table 3.4: Number of Train/Val Samples for the Trashnet Dataset

Deep Learning model	4-fold accuracy
DenseNet121	96.73% (+/- 0.59%)
VGG16	96.22% (+/- 0.61%)
ResNet50	96.21% (+/- 0.22%)

Table 3.5: Accuracy for the base-learner models on the Trash-net Dataset

Material	Precision	Recall	f1-score	support	Specificity	Sensitivity
glass	95%	94%	95%	250	99%	94%
metal	98%	97%	97%	250	99%	97%
paper	96%	98%	97%	250	99%	98%
plastic	96%	94%	95%	250	99%	94%
trash	98%	100%	99%	250	99%	100%
accuracy			97%	1250		
macro avg	97%	97%	97%	1250		
weighted avg	97%	97%	97%	1250		

Table 3.6: Classification Report for Stacked Network architecture

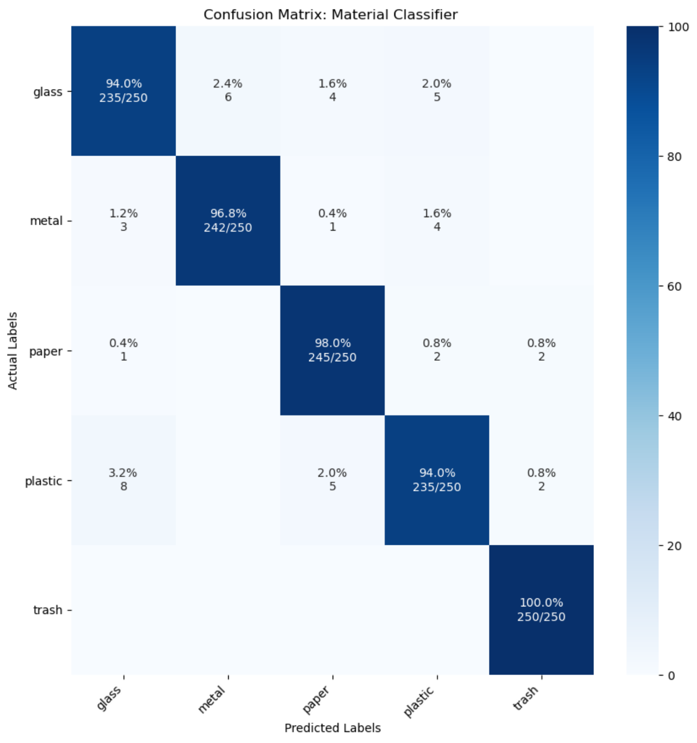


Figure 3.5: Fine-tune Stacked Material Classifier Confusion Matrix

Here we will analyze the performance of base-learners architecture regardless of the stacked ensemble. The evaluation metrics which we display are the metrics that explained in section 2.8. Firsly in figure 3.6 we can see the confusion matrix of ResNet50 architecture test on Trashnet and in table 3.7 the respectively confusion matrix. Respectively in table 3.8 and confusion matrix in fig. 3.7 the same metrics for the VGG16 architecture. Finally the DenseNet121 Architecture are the best among the 3 which tested, the classification report is shown in table 3.9 with the appropriately confusion matrix in fig. 3.8.

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
glass	92%	95%	93%	500	98%	94%
metal	95%	96%	96%	500	98%	96%
paper	98%	97%	98%	500	99%	97%
plastic	97%	92%	94%	500	99%	92%
trash	98%	98%	98%	500	99%	98%
accuracy			96%	2500		
macro avg	96%	96%	96%	2500		
weighted avg	96%	96%	96%	2500		

Table 3.7: Classification Report for ResNet50 as base-learner

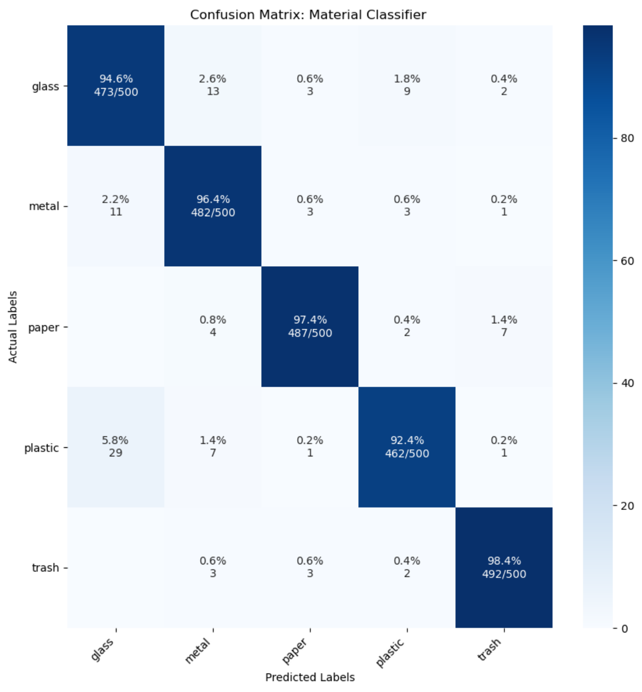


Figure 3.6: Fine-tune ResNet50 Material Classifier Confusion Matrix

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
glass	92%	94%	93%	500	98%	94%
metal	95%	97%	96%	500	98%	97%
paper	100%	94%	97%	500	99%	94%
plastic	95%	95%	95%	500	99%	95%
trash	96%	99%	98%	500	99%	99%
accuracy			96%	2500		
macro avg	96%	96%	96%	2500		
weighted avg	96%	96%	96%	2500		

Table 3.8: Classification Report for VGG16 as base-learner

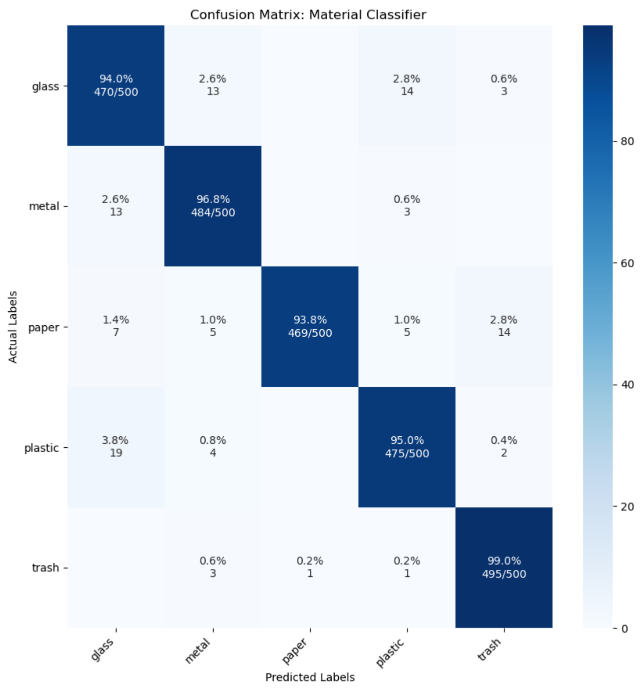


Figure 3.7: Fine-tune VGG16 Material Classifier Confusion Matrix

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
glass	99%	94%	96%	500	99%	95%
metal	97%	98%	98%	500	99%	98%
paper	98%	98%	98%	500	99%	98%
plastic	96%	98%	97%	500	99%	98%
trash	98%	99%	99%	500	99%	99%
accuracy			98%	2500		
macro avg	98%	98%	98%	2500		
weighted avg	98%	98%	98%	2500		

Table 3.9: Classification Report for DenseNet121 as base-learner

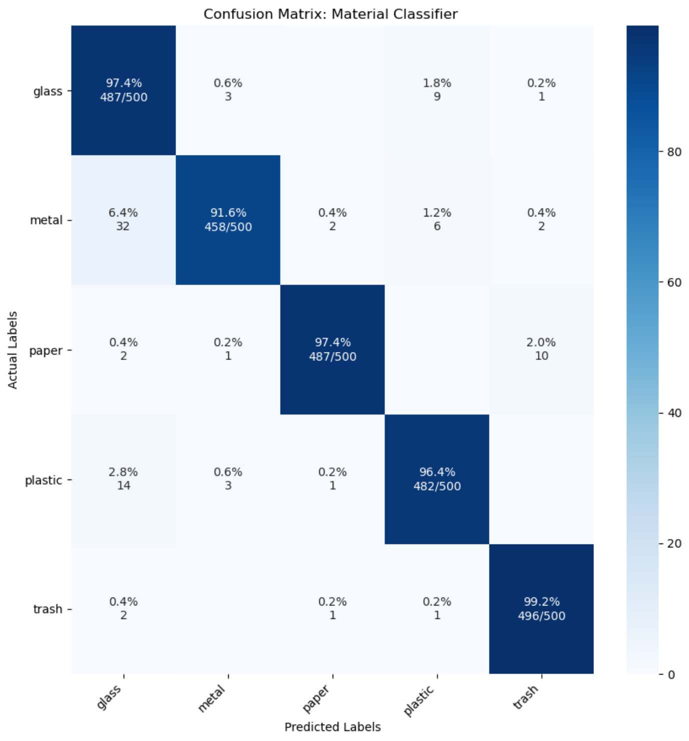


Figure 3.8: Fine-tune DenseNet121 Material Classifier Confusion Matrix

Finally, we created a new (different to the one described above) hold-out set which involves replacing the plastic class images in the original (Trashnet) hold-out set with plastic images originating from WaDaBa. In the multi-output classification, the network branches at least twice (sometimes more) to build several sets of fully-connected heads at the end of the network. The network then predicts a series of class labels for each head, making it possible to learn disjoint label combinations. We have at least two heads attached to the same body who each oversee a different, specialized classification tasks. We use the two independent networks (branches) as a "frozen classifier" to determine images and calculate multiple outputs obtaining the multi-output classifications. Then, we perform the *final one-label* classification step to predict the class of the input image. Fig. 3.9 and 3.10 summarize the performance of the proposed classification scheme in material separation and plastic identification, respectively.

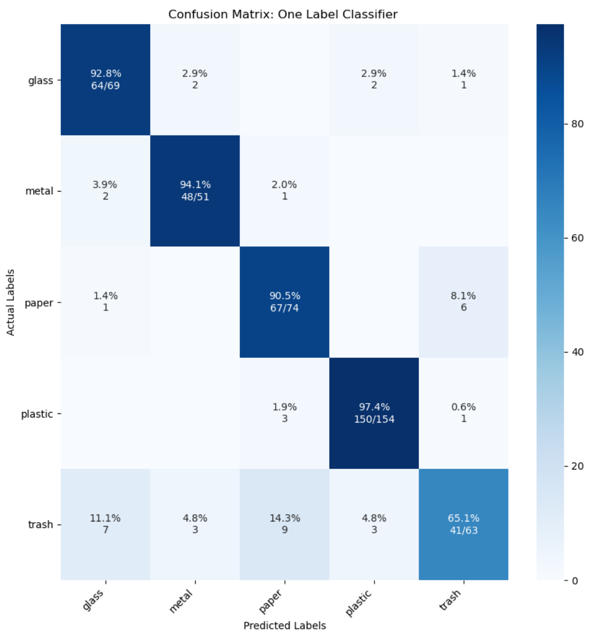


Figure 3.9: Confusion Matrix of the Dual-Branch classifier for materials

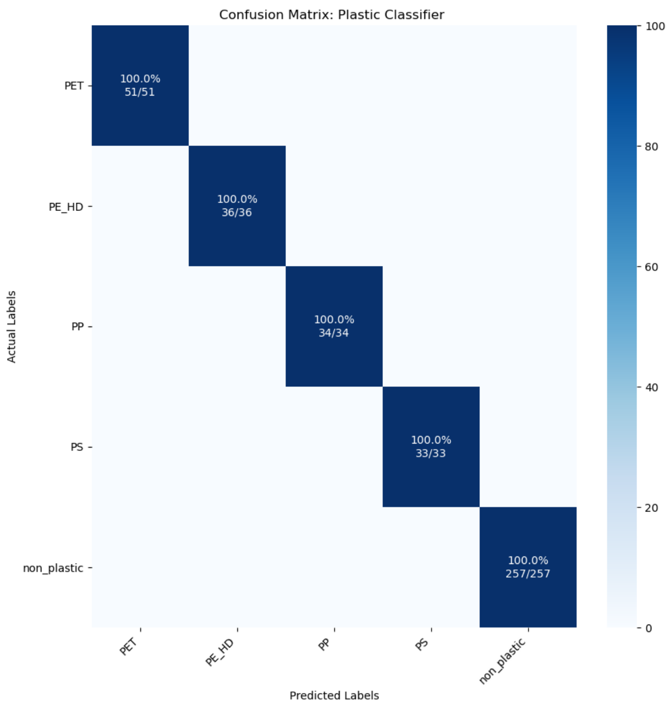


Figure 3.10: Confusion Matrix of the Dual-Branch classifier for plastics

By contrast, Fig. 3.11 demonstrates the independent performance of the material classifier (i.e., the “Trashnet branch”). In particular, we observe the inability of the material branch to correctly identify several plastic items that are erroneously classified as trash, paper, or metal as appeared. This underscores the need for our proposed extra step, the one that combines the two results into one final output. Intuitively, the final extra step manages to enhance the performance of the material classifier, based on the excellent performance of the “WaDaBa branch” in the separation of plastics.

Finally, to test the transfer learning ability of our proposed Dual Parallel CNN architecture, and showcase its ability to combine different datasets to improve accuracy, we employ a second, mixed, hold-out set, different to the Hold-Out set #1 mentioned above. This mixed Hold-Out set #2 contains only 411 images, and is built by replacing the plastic images of the original (augmented TrashNet) hold-out set with 154 plastic images originating from WaDaBa; while the 257 images used for the other materials come from TrashNet but are different to the ones used in Hold-Out set #1 above (which had

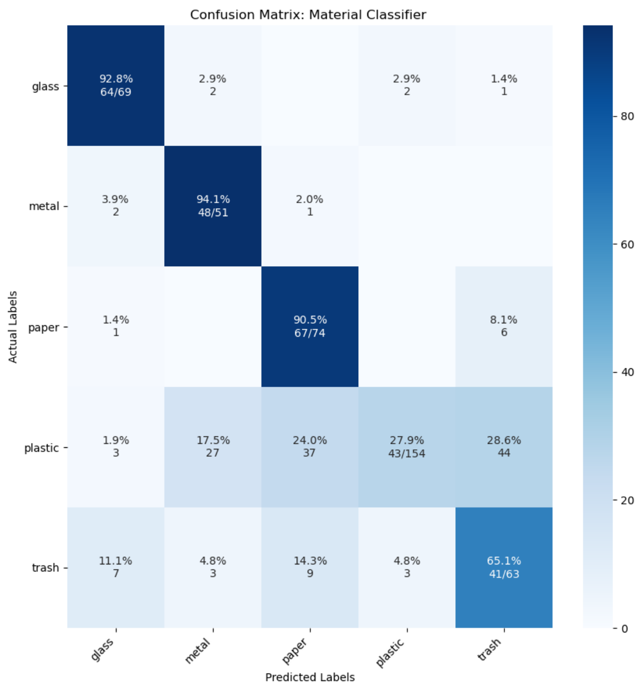


Figure 3.11: Confusion Matrix for independent Material Classifier

1250 images in total). As such, this is a rather demanding transfer learning experiment. To accomplish this task, we use the two independently trained networks (branches) as "frozen" classifiers to determine example images and obtain the multi-output classifications. Then, we perform the final one-label classification step to predict the class of the input image. Tables 7 & 8 summarize the performance of the overall classification scheme in material separation and plastic identification, respectively. The overall model's accuracy is 90.02%.

3.5 Conclusions and Future Work

In Chapter 3 we presented a dual-branch CNN architecture for the classification of recyclables, which achieves incremental learning from disjoint datasets. Our architecture takes advantage of the combined knowledge of its sub-nets across the various datasets, with local training only on the site of each dataset. Thus, we can learn from multiple sources and accumulate knowledge without mingling the data samples. The combination scheme at the classification level exploits the individual outcomes of subnets in a way that increases confidence in final decision making process. Our results indicate that we can indeed achieve efficient use of independent datasets, as to obtain accurate classification performance. Moreover, our classifiers outperform state-of-the-art ones used on our evaluation datasets. In subsequent stages of this research, we intend to use this approach in real-time applications, as an adaptively modified Federated learning scheme.

Chapter 4

Independent Parallel CNN Approach

The aim of our in this chapter work is to solve a single label multi-class classification problem which has been decomposed into as many parallel binary classifiers as the different classes. Our approach can be thought of a belonging in the *one-vs-rest* of classification methods family.[5]

The rest of this chapter is organized as follows. In Section 4.1 we provide an overview of our approach and highlight the contributions of the specific architecture. In the Section 4.2 we describe the architecture of our system and analyze all the required notation for the decision rules employ in voting mechanism. In the section, 4.3 we present the experiments on datasets that we will conduct and the respective cases to be explored. Finally, in the 4.4 we show the results of the proposed architecture and discuss the outcome and the models' behavior in different cases.

[5]As is usually used in the literature, the term "one-vs-rest" describes a generic classification paradigm that uses binary classification algorithms for multi-class classification. It involves tackling the multi-class problem as multiple "binary" classification ones. A dedicated "binary" classifier (that potentially uses a sigmoid function for classification into the most probable of the two classes) is then trained on the original dataset, and solves a "binary" classification problem; and final predictions are usually made using the classifier that is the most confident, or using a simple or absolute majority rule.

On the other hand, the "one-vs-one" is a categorical classification paradigm that employs binary classification algorithms to perform pair-wise multi-class classification. That is, the one-vs-one approach uses a "binary" classifier per pair of potential classes. Then, final predictions are usually made using the classifier that is the most confident, or using some weighted voting method [14]

4.1 Overview and Contribution

Specifically, our proposed architecture consists of a set of independent dedicated binary classifiers, each one with two output nodes, with each output node expressing the probability of the item under consideration to of belong in this class or not. Our method allows for the effective separation of items based on distinct class features and characteristics; and is able, in its generality which will become apparent below, to (a) take advantage of clear class boundaries when these exist, and (b) to effectively assign items to classes with increased confidence, even when clear class boundaries do not exist.

Our key intuition is that when only one of the independent dedicated classifiers, say k, puts a "large enough" probability on the item under consideration belonging to its class, while all others believe the item cannot be classified in their respective class, then we can be confident that the class to select as the output of our system is indeed the one predicted by the k-th classifier. We act upon this intuition via a decision rule we put forward and which implements it.

Arguably, there are many settings and problems where the assumption behind this approach is valid and helpful. For instance, many image datasets typically contain images with drastically different characteristics, allowing for the training of different classifiers with high confidence levels. Examples include *a)* recyclable materials: for instance, glass objects are very different from plastic, making it easy to differentiate between the materials; *b)* facial recognition problems: e.g., the color of the skin and the shape of the eyes are powerful characteristics which facilitate the required separation; *c)* X-ray datasets: in such images, the bone and tissue density provide strong signals that denote certain medical conditions.

Of course, there exist many cases in which features might be such that it is harder to distinguish among classes. That is, either the class boundaries are unclear, or the confidence of certain classifiers is low. The latter could occur, for instance in the case of *highly unbalanced datasets.* Our approach is generic enough to tackle such cases, via the incorporation of *(i)* classifier-specific "confidence-related weights", and *(ii)* several alternative decision rules and methods we propose, on top of the main decision rule mentioned earlier.

In more detail, the main contribution of our work in this chapter is putting forward a generic architecture that explicitly exploits a *"mutual exclusivity property"* underlying many dataset domains of interest: the fact that in many cases an image in a given dataset might almost unquestionably belong to one class only. In such cases, one would expect the predictions of dedicated independent binary classifiers to “support each other”: if the glass classifier would predict “glass”, while the plastic classifier would predict “not plastic”. We start from this observation, and build on it to provide a generic architecture that employs an independent parallel network of CNNs to provide accurate classification even if the aforementioned assumption does not hold. Our proposed architecture is able to separate individual features specific to each class, from the pool of shared features; and includes *designed-to-purpose decision rules* that are able to select the final class of the item to be classified. To this end, and to allow for flexibility and easy adaptation to the specific properties of the datasets of interest (e.g., on a dataset’s degree of “balancedeness” with respect to the types of images it contains), our approach also makes use of “weights” associated to each independent classifier, and which intuitively mirror the confidence each dedicated classifier has in identifying its corresponding class. Moreover, we propose the use of proper scoring rule, the brier score [67], for the automated re-adjustment of the aforementioned weights.

An additional, in a sense “emergent” contribution, is the fact that the proposed framework may be utilized to characterize a given dataset regarding its “homogeneity” and “balancedness” properties. That is, the framework can be used to apply a series of decision rules, potentially with varying weights and parameters and based on different intuitions and scenarios of interest, An exploratory qualitative and quantitative dataset analysis process can then be effectively carried out for each dataset, in parallel with and also following the classification decisions optimization one: by taking into account the frequency with which the conditions for the application of a specific *decision rule* is valid, one can analyze the dataset to decide its status—e.g., the degree by which contained items’ classes are related to each other. This exploratory dataset analysis, can be used to further drive the modeling process, for instance addressing certain questions regarding the choice of model parameters. For image classification problems, this may yield insights regarding the distributions of features, or for identifying meaningful trends of predictors in different classes. Notice that the same intuitions apply for generic classification problems: one can assess the use of simple “decision rules” to acquire meaningful knowledge regarding the dataset at hand, and further optimize the classification model.

4.2 System architecture and Decision Rules

We begin this section by describing the architecture of our system. The big picture of this architecture is shown in Figure 4.1

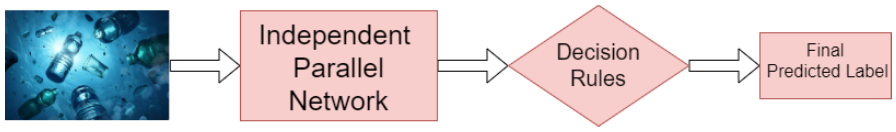


Figure 4.1: Proposed Architecture

There is a set S_C of M classifiers, one dedicated for each class, following the one-vs-rest classification method for decomposing the multi-class problem to M dedicated binary classifiers. In departure to what is the norm[6] in one-vs-rest classification approaches, the output of the network is a probability vector, each element of which contains the independent probabilistic estimate of each classifier representing its degree of certainty regarding an item belonging to its respective class. This vector will then be acted upon by the decision rules we put forward, and which we will be presenting in detail below, in order to come up with a final classification decision.

To explain further, Figure 4.2 depicts graphically the Independent Parallel Architecture in which we apply the different Decision Rules. The output of the architecture is a final vector of f_i^Y probabilities, signifying the degree of certainty by classifier i regarding an item belonging to its respective class; and our decision rules will be acting on this vector. We explain the f_i^Y probabilities, along with other required notation, immediately below :

Below we provide the notation required to explain the Decision Rules.

- $i \in M, |M| >= 2, M$: set of different classes
- p_i^Y : the dedicated classifier's output probability for (yes)
- p_i^N : the dedicated classifier's output probability for (no) $[p_i^N = 1 - p_i^Y]$

[6]The One-versus-All (or OvA) Strategy is a technique for categorizing numerous (more than two) classes using Binary (Two-Class) Classifiers (such as Support Vector Machines). Making choices entails applying all classifiers to an unknown sample and predicting the label for which the associated classifier provides the highest confidence score:

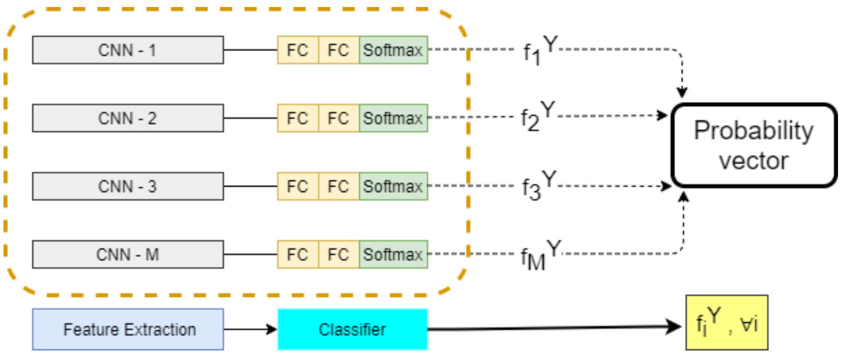


Figure 4.2: Independent Parallel Network (FC = Fully Connected Layer)

- w_i : the dedicated classifier's weight, with $\sum_{i=1}^{M} w_i = 1$
- $f_i^Y = w_i \cdot p_i^Y$ class i's final weighted probability ; $f_i^N = 1 - f_i^Y$
- $\hat{f}_i^Y$: the normalized f_i^Y values of all classifiers ; $\hat{f}_i^N = 1 - \hat{f}_i^Y$
- $\tilde{f}_i^Y$: the normalized f_i^Y values of classifiers which output $f_i^Y > f_i^N$; $\tilde{f}_i^N = 1 - \tilde{f}_i^Y$
- $\bar{f}_i^Y$: the custom f_i^Y values of classifiers which output $f_i^Y > f_i^N$
- c_k^s : the selected class according to a given decision rule k
- T : a required confidence threshold[7]

We note that deciding the w_i weights of each classifier is an interesting engineering problem: weights can represent our confidence to the specific classifier; or, more accurately, to the ability of the classifier to be effective on the particular dataset, given the features and other characteristics of the dataset—such as the number of its items, the degree of its items' homogeneity (e.g., features' distribution) and the number of items belonging to each possible class, and so on. As mentioned earlier, a dataset analysis process can be carried out multiple time for a particular dataset to determine w_i's: once before classification to determine the original w_i weights, and then again also again in an iterative fashion, in order to re-determine the w_i following a given number of classifications, as shown in Algorithm 4 depicting the iterative modeling optimization process

[7]The default T value used in our experiments is 80%.

required here. Figure 4.3 also shows the application of the decision rules on the f vector, and the iterative weight optimization process.

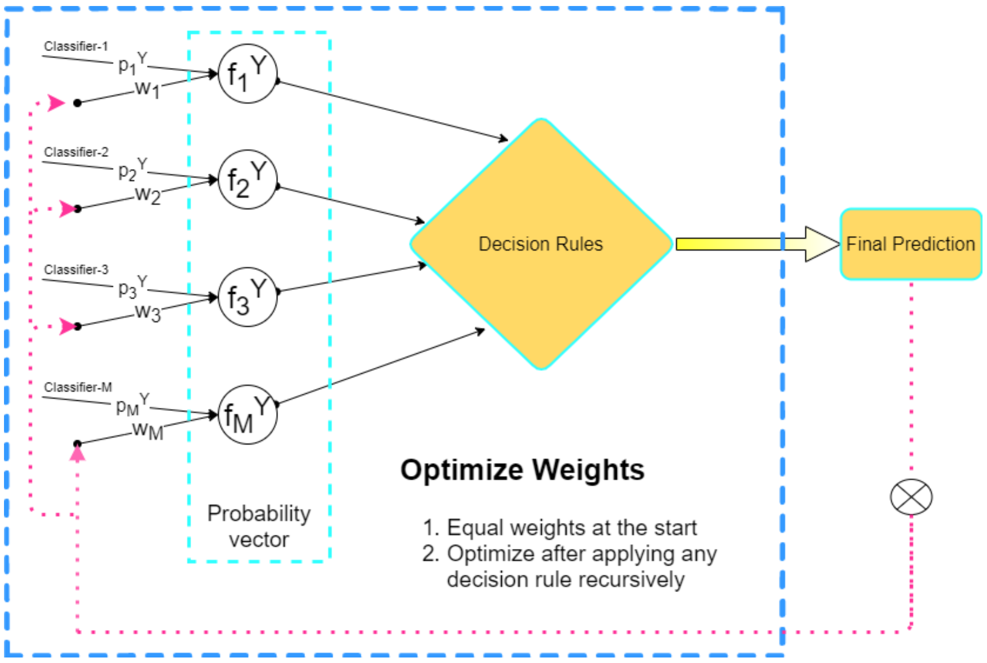


Figure 4.3: Weight Optimization for the predictions after Applying Decision Rules

Here, we will explained the algorithm 4, which used to compute the optimal weights for each independent classifier. Below we provide some information required to explain the algorithm.

- Weights are initialized at first iteration equally and may be optimize after the training phase.
- All weighted w_i output sum to 1.
- After the initial prediction we optimize the weights followed the algorithm steps.
- We use the Nelder-Mead [68] algorithm[8] for weights optimization in the experiments

[8]The Nelder-Mead optimization method is named after its creators, John Nelder and Roger Mead. The method was introduced in their 1965 work "A Simplex Method For Function Minimization," and has since been a commonly used standard approach for function optimization. It is suitable for functions with numerical inputs that are either one-dimensional or multidimensional. Nelder-Mead is a pattern

Algorithm 4 Finding optimal Ensemble Weights process

```
1: Input:    (a) Predictions probability of each model $p_i^Y$, (b) initial equal weights $w_i$,
   (c) initial label
2: Output:    $f_i^Y$ vector
3: while Condition != True do
4:     $f_i^Y = w_i \cdot p_i^Y$
5:     initial label = predict class
6:     if predict class probability > current probability then
7:         adjust the weights $w_i$
8:     else
9:         keep the same weights $w_i$
10:    end if
11: end while
12: return $p_i^Y * w_i$
```

In more details about the weight optimization step, there are techniques to bypass the optimization process entirely by utilizing knowledge gained during the training phase. A such case is to compute *a priori* information from representative samples that could be used to reinforce or dilute weightage values for a particular class, such as *a)* the frequency of any decisions rule are valid, *b)* the number of samples of every class, *c)* some evaluation metrics. In subsection 4.2.7 we proposed such a type of technique which adapt proper weight based on *Brier Score*.

With this overall architecture and notation at hand, we now proceed to describe our main opinion aggregation-based decision rule, namely the *Mutually Supported Decisions Rule (MSDR)* mathematically, along with variants we propose and which are used when the conditions for MSDR's application do not hold.

4.2.1 Mutually Supported Decisions Rule (MSDR)

As mentioned earlier, our main decision rule is based on the key intuition that the findings of the various independent, dedicated classifiers are mutually supported, thus we term

search optimization method, which implies that it does not require or use function gradient information and is thus acceptable for optimization scenarios in which the function's gradient is unknown or cannot be estimated properly.

it "the Mutually Supported Decisions Rule (MSDR)". We expect MSDR to be highly successful as a classification rule, when only one classifier (k) predicted $f_k^Y > f_k^N$, and all the others believe the item cannot be classified in their class, i.e., $f_i^Y < f_i^N, \forall i \neq k$. We can then be confident that the class c^s to select as our output is indeed the one predicted by the k-th classifier.

In Equation 4.1 below we show our first of three MSDR variants. We represent the class selected by this decision rule by c_1^s, since it is the class returned by our first decision rule:

$$c_1^s = k, \qquad \exists k : f_k^Y > f_k^N \quad and \quad f_k^Y \geq T \quad and \quad \forall j \neq k \quad f_j^Y < f_j^N \tag{4.1}$$

That is, c_1^s is returned if one classifier only answers "yes" to the question on whether it believes the item belongs to the class of items it is trained to identify, and believes this with a confidence that exceeds a prespecified threshold. Therefore, we give this MSDR variant the name "MSDR-ONE-YES-THRESHOLD-YES" decision rule.

Now, we employ two alternative courses of action in case the main condition ($f_k^Y > f_k^N$ for just one k) holds in Equation 4.1 above, but the threshold is not met. The first is to pick the class the k classifier predicts anyway. This is captured by Equation 4.2, which thus constitutes our "MSDR-ONE-YES-THRESHOLD-NO-PICK-YES" decisions rule:

$$c_{1.1}^s = k, \qquad \exists k : f_k^Y > f_k^N \quad and \quad f_k^Y < T \quad and \quad \forall j \neq k \quad f_j^Y < f_j^N \tag{4.2}$$

The second alternative course of action for MSDR is to disallow the deterministic determination of the outcome by k. Instead, we normalize the f_i^Y values of all classifiers to respective $\hat{f}_i^Y$ values so as to sum to 1, and then randomly sample the class to be selected out of this distribution. This is our "MSDR-ONE-YES-THRESHOLD-NO-PICK-RANDOM" decision rule, shown below:

$$\begin{aligned} c_{1.2}^s = \arg random_{i \in S_C}(\hat{f}_i^Y), \qquad \exists k : f_k^Y > f_k^N \quad & and \quad f_k^Y < T \\ & and \quad \forall j \neq k \quad f_j^Y < f_j^N \end{aligned} \tag{4.3}$$

Now, if the main MSDR condition of having only one dedicated classifier predicting its corresponding class *does not* hold, we use alternative rules and methods to pick a class to output as the one predicted by our system. These rules and methods are described in the following sections.

4.2.2 Simple Max Rule and variants

When more than one classifiers have predicted a $f_i^Y > f_i^N$ probability, we can simply select the corresponding class with maximum probability, as long as a confidence threshold is met:

$$c_2^s = \operatorname*{argmax}_i f_i^Y \quad \text{where } i: \quad f_i^Y > T \tag{4.4}$$

That is, we simply select the class c_2^s corresponding to the classifier with maximum f_i^Y across all classifiers, as long as $f_i^Y > T$. This is the "MANY-YES-THRESHOLD-YES-PICK-MAX" rule.

Now, if the threshold is not met, we do not possess the required confidence level to decide, so we can either *(a)* accept the class with the highest probability, i.e., the one selected by the following "MANY-YES-THRESHOLD-NO-PICK-MAX" rule:

$$c_{2.1}^s = \operatorname*{argmax}_i f_i^Y \tag{4.5}$$

or *(b)* we can normalize the f_j^Y values of only the classifiers which output $f_j^Y > f_j^N$ to $\tilde{f}_j^Y$ so as to sum to 1, and then we randomly sample the selected class out of this distribution. This is the "MANY-YES-THRESHOLD-NO-PICK-RANDOM":

$$c_{2.2}^s = \arg random_j(\tilde{f}_j^Y) \tag{4.6}$$

As mentioned, the $\tilde{f}_j^Y$ vector contains the normalized softmax elements of the classifiers' final output probability $\forall j \in C_S$ with $f_j^Y > f_j^N$. So, the chosen class $c_{2.2}^s$ is randomly selected from this particular subset of classes, according to their $\tilde{f}_j^Y$ probabilities.

4.2.3 Weighted Soft Voting

When more than one classifiers have predicted a $f_i^Y > f_i^N$ probability, we implement a *weighted soft voting* method utilizing class-specific weights w_i, and calculate new custom probabilities only for these classifiers.

We predict the custom probabilities $\bar{f}_i^Y$ only for the classifiers which output $f_j^Y > f_j^N$ with the help of the others classifiers which output $f_j^Y < f_j^N$. That is, for every classifier which outputs $f_i^Y > f_i^N$ the custom probability $\bar{f}_i^Y$ is calculated according to the equation below:

$$\bar{f}_i^Y = w_i \cdot p_i^Y + \sum_{j:f_j^Y < f_j^N} w_j \cdot p_j^N, \quad \forall i : f_i^Y > f_i^N \tag{4.7}$$

and then we pick the highest among them, which thus constitutes our "MANY-YES-WEIGHTED-SOFT-VOTING" decisions rule:

$$c_3^s = \underset{i:f_i^Y > f_i^N}{\operatorname{argmax}} \bar{f}_i^Y \tag{4.8}$$

As mentioned, the $\bar{f}_j^Y$ vector contains the custom probabilities elements of the classifiers' final output probability, $\forall j \in M$ with $f_j^Y > f_j^N$. So, the chosen class c_3^s is selected from this particular subset of classes, according to their $\bar{f}_j^Y$ probabilities.

4.2.4 Negative Predictors Rule

Now, when all classifiers predict $f_i^Y < f_i^N$—i.e., when they are all "pessimistic" or "negative" regarding the item under consideration falling in their corresponding class—we simply choose the class with highest f_i^Y probability :

$$c_4^s = \underset{i:f_i^Y < f_i^N}{\operatorname{argmax}} f_i^Y \quad \text{, used when all } i \text{ predict } f_i^Y < f_i^N \tag{4.9}$$

This is the "ALL-NO-PICK-MAX" decision rule.

4.2.5 SVM Correction Rule

We also propose the following method to assist us with decision-making when there are many classifiers that predict an image to belong in the class they specialize in, or when there is only one such classifier, but the threshold is not met.

That is, we also implement we build $\frac{M \cdot (M-1)}{2}$ SVMs following the "one-vs-one" classification paradigm [8]. These SVMs are trained along with our M independent Classifiers. Thus, we have an SVM for every pair-wise classifier, which is trained with the actual

[8]The One-vs-One strategy splits a multi-class classification into one binary classification problem per each pair of classes.

one-hot label for the image, and the predicted label from the M independent parallel Classifiers, as shown in Figure 4.4. These SVMs are used for the final decision making instead of our “MANY-YES” or the “MSDR-ONE-YES-THRESHOLD-NO” decision rules variants.

In other words, we also propose and evaluate the following classification method, which can also be seen as a decision rule variant:

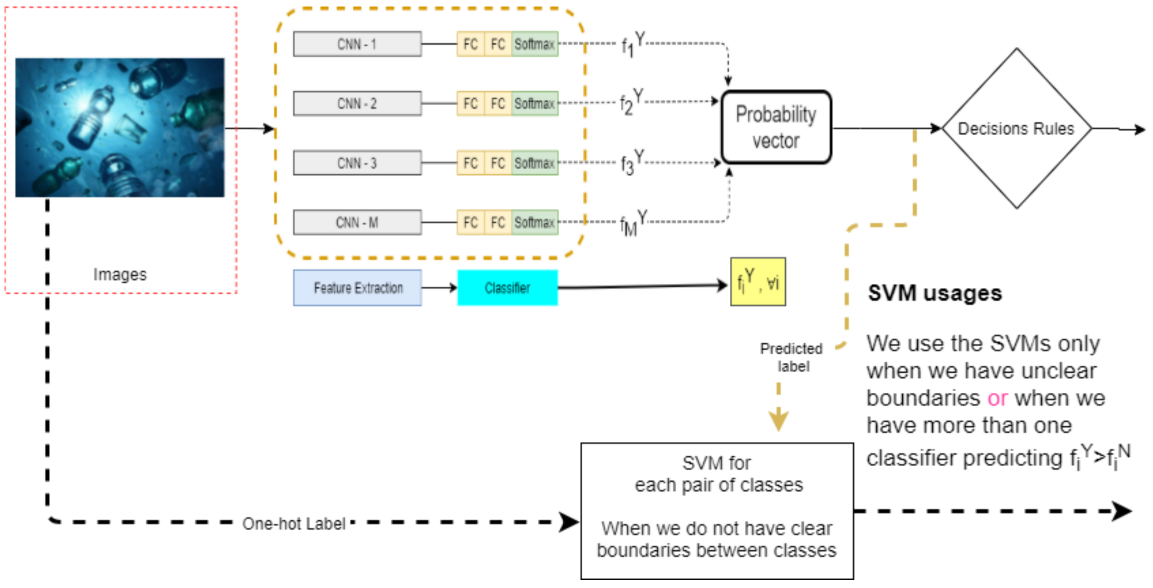

Images
CNN - 1
CNN - 2
CNN - 3
CNN - M
FC
FC
Softmax
f_1^Y
f_2^Y
f_3^Y
f_M^Y
Probability vector
Decisions Rules
Feature Extraction
Classifier
$f_i^Y, \forall i$
SVM usages
We use the SVMs only when we have unclear boundaries or when we have more than one classifier predicting $f_i^Y > f_i^N$
Predicted label
SVM for each pair of classes
When we do not have clear boundaries between classes
One-hot Label

Figure 4.4: SVM_Architecture

$$c_5^s = \begin{cases} k, & f_k^Y > f_k^N \quad and \quad f_k^Y \geq T \quad and \quad \forall j \neq i \quad f_i^Y < f_i^N \\ \text{argmax}_i \, f(SVM_j), & else \end{cases} \tag{4.10}$$

Thus, if more than one classifier predicted, $f_i^Y > f_i^N$ we pick the higher probability from the SVM_k, with (k) being one class from the pair of classes in the corresponding SVM.

4.2.6 Decision Rules Summary

The approach and decision rules, discussed in this chapter, can be summarized in the following Table 4.1:

Name of Decision Rule	Description	Equation
MSDR-ONE-YES-THRESHOLD-YES	Only one classifier consider the item to belong in one class and also confident enough	4.1
MSDR-ONE-YES-THRESHOLD-NO-PICK-YES	Only one classifier consider the item to belong in one class but not confident, so pick the class (k)	4.2
MSDR-ONE-YES-THRESHOLD-NO-PICK-RANDOM	Only one classifier considers the item to belong in one class but is not confident, so we pick randomly among all classifier outcomes after normalization	4.3
MANY-YES-THRESHOLD-YES-PICK-MAX	More than one classifier consider the item to belong in one class, and we simply pick the one with the highest f_i score if exceeds the confidence threshold	4.4
MANY-YES-THRESHOLD-NO-PICK-MAX	More than one classifiers consider the item to belong in one class, and we simply pick the highest score since no classifier exceed the confidence score	4.5
MANY-YES-THRESHOLD-NO-PICK-RANDOM	More than one classifiers consider the item to belong in one class, and we select randomly among only these after normalize	4.6
MANY-YES-WEIGHTED-SOFT-VOTING	More than one classifiers consider the item to belong in one class, and we select the higher custom probability among only these classifiers	4.8
ALL-NO-PICK-MAX	All classifiers consider the item to not belong in their class, and we simply pick the one with the highest f_i^Y probability	4.9

Table 4.1: Cumulative Decision Rules Description.

The decision rules above can help resolve indecisive cases such as the ones in the examples of Table 4.2 and Table 4.3.

Classifier: f_i^Y	Probability, T=90%		
	Case 1	Case 2	Case 3
1	0.9	0.3	0.23
2	0.4	0.84	0.3
3	0.3	0.4	0.8
4	0.35	0.3	0.4
5	0.2	0.35	0.3

Table 4.2: Different MSDR predictions cases with Threshold 90%

In Table 4.2 we can see 3 cases in which we apply a different "MSDR" decision rule. In more detail, given the f_i^Y for every classifier, $i \in [1, 5]$, we select the appropriate rule. For example in *case 1* we select the classifier-1 predicted class following the equation 4.1 due to the fact that the classifier's probability reaches the 90% Threshold; in *case 2* the one classifier that predicted $f_i^Y > f_i^N$, here classifier-2, has not reached the threshold, and we pick simply the higher probability following the Equation 4.2. Finally, in *case 3*, we normalize all probabilities and sample following the Equation 4.3), since there is no one classifier which surpass the Threshold. Cases 2-3 both fall under the category "MSDR-ONE-YES-THRESHOLD-NO" but a different rule applies.

Classifier: f_i^Y	Probability, T=90%		
	Case 4	Case 5	Case 6
1	0.8	0.3	0.63
2	0.4	0.84	0.3
3	0.92	0.4	0.86
4	0.35	0.67	0.4
5	0.2	0.35	0.78

Table 4.3: Different MANY-YES predictions cases with Threshold 90%

In the Table 4.3 we can see 3 cases in which we apply a different "MANY-YES" decision rule. In more detail, given the f_i^Y for every classifier, $i \in [1,5]$, we select the appropriate rule. For example in *case 4* we select the classifier-3 predicted class following the Equation 4.4 due to the fact that the classifier's probability exceeds the Threshold 90% and we pick the higher probability among the classifier which predicted $f_i^Y > f_i^N$. In *case 5* we select the class of classifier_2 following Equation 4.5, here classifier-2 output. Finally in *case 6*, we normalize all probabilities from classifiers which predicted $f_i^Y > f_i^N$ (green cells) and sampling following Equation 4.6.

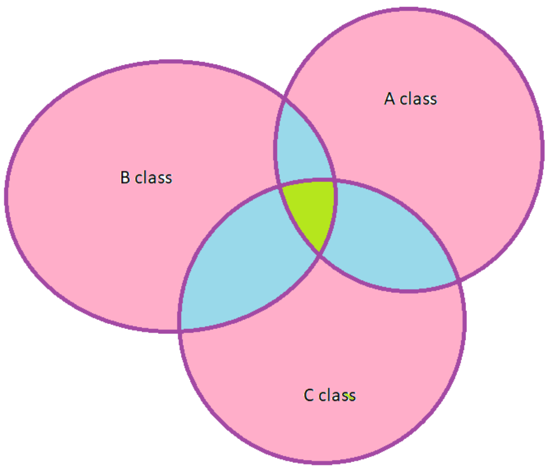


Figure 4.5: Features of class, depicted as a Venn Diagram

Figure 4.5 depicts graphically the different cases of the classification mechanism. In more detail, the 3 cases from Table 2 are the pink areas which we have clear boundaries among all different class. In these cases we have only one classifier to believe that the object belongs to its own class. If this classifier output exceeds the confidence level we are in case 1, if not we choose between case, 2 or 3 as alternative methods.

On the contrary, the blues areas belongs to more than one class (especially green to all classifiers) and the features may classify to uncertain outcome with more than one classifier predicted that the object belongs to its own class. In these cases we have the case 4-6 from Table 4.3. In more detail, in case 4 we pick the max probability on these classifiers if it exceeds the confidence level. If the confidence score is not reached we simply pick the "most confident" among these classifiers' that have predicted the item belongs to their class, (case 5), or normalize only their respective probabilities and sample from the normalized distribution (case 6).

4.2.7 Re-adjusting the Classifier Weights

Here we address a simple automated method for modifying the classifier' weights. The approach evaluates our classifiers' effectiveness according to a classic criterion from the scoring rules literature, and uses the results to adjust their original weights.

Specifically, we propose to employ the *Brier Score* [67], which is a *strictly proper* [10] scoring rule function that evaluates the accuracy of probabilistic predictions [69].

Strictly proper scoring rules have many applications, and often appear in the mechanism design literature to help guarantee that agents declare their true preferences: if they do so they are rewarded, or are "punished" by the rule otherwise [70, 71]. In our case here, strict propriety is desirable since we wish to reward classifiers that produce accurate predictions: only predictions that are truly accurate receive the maximum score assigned by the rule.

We champion the use of the Brier rule, since it is applicable to assignments where forecasts must allocate probability to a set of mutually exclusive discrete outcomes or categories. The Brier score computed the mean squared error between the model's expected probabilities and the actual values. The score quantifies the magnitude of the prediction error and is optimized for binary classification problems. It evaluates the probability for the positive class, where the positive class is coded as 1, and negative class 0. As such, it is a key measurement for unbalanced classification problems [72].

The Brier score may be thought of as a cost function. In our case if the classifier makes the wrong prediction with high probability, it should incur a significant penalty. The lower the Brier score, the lower the average "penalties" assigned to the classifier.

[10]Formally, if P is the true underlying distribution of the random variable x, a scoring rule S is strictly proper if $S(P,x) \geq S(\hat{P},x)$, with the equality holding if and only if $\hat{P} = P$. (By contrast, the scoring rule is said to be proper if $S(P,x) \geq S(\hat{P},x)$, but the prediction $\hat{P} = P$ is not the only one that maximises $S(\hat{P},x)$)

The typical formulation of the Brier score is :

$$BS = \frac{1}{N}\sum_{i=1}^{N}(p_i - o_i)^2 \tag{4.11}$$

In our case, p_i in Equation 4.11is the prediction probability for i' image sample, the term o_i is equal to 1 if the image sample classify correct and 0 if not, and N is the number of predicting images samples. The Brier score is a proper scoring rule [73] and therefore the optimum score corresponds to a perfect prediction.[11]

In conclusion, we propose the use of the Brier score to determine the predictive accuracy of binary prediction models, and as such to enable the calibration of such models. The calibration follow the rule of thumb that we normalize the *Brier Scores* probability BS_i of all the classifiers to sum to 1, and then we adjust the dedicated weights according to this distribution.

4.3 Experiments

In this section, we will be describing the experiments conducted to evaluate our approach.

Our goals in this section can be summarized as follows:

1. To assess the performance variations of the different decision rules
2. To characterize the dataset “status” regarding its “homogeneity” and “balancedness” properties.

Additionally, we intend to do an assessment of the potential of this framework to readjust the weights to achieve an even better prediction accuracy as future work

For evaluating our decision rules, we will run tests with two types of Datasets: *a)* balanced and *b)* unbalanced. In more details, we consider a balanced dataset to have

[11]There are some decompositions of the Brier score that offer a greater insight into the behavior of the binary classifier. Especially a convenient formulation is BS = *uncertainty - resolution + reliability* [74, 75, 76], where uncertainty is the marginal uncertainty about labels, resolution measures the divergence of individual forecasts from marginal, and reliability measures calibration as an average violation of long-term true label frequencies. *Drawbacks:* Brier score is insensitive to predicted probabilities associated with in/frequent events [69]. Since it does not adequately differentiate among minor shifts in forecast that are important for rare events.

equal samples of images in every class. This does not apply for the "unbalanced dataset". Specifically, the term "unbalanced data" refers to the case where the number of observations for every class in a classification dataset is unequal, in the sense that there is major and/or significant variance among the examples in each class of the problem [72]. Improvements in learning from unbalanced data have been inspired by various real-life applications in which we encounter the issue of unequal data representation. The minority class is the more important one, and therefore we need methods to increase its representation in the results [77]. Unbalanced classifications pose a challenge to predictive modeling, since most of the machine learning techniques used mostly for classification are based on the expectation of an equal number of instances for each class. This results in models with poor predictive efficiency, especially for the minority class [78]. This is a problem because the minority class is more important, generally speaking, and therefore the problem is more susceptible than the majority class to errors in classification for the minority class [79].

It is also the case that the assessment is easier for balanced datasets because there is no implicit bias in that case.

4.3.1 Datasets and Data Augmentation Techniques

In this subsection, we describe two originally *unbalanced* datasets (Trash-net and WaDaBa) and in order to produce balanced versions of these we employ data augmentation techniques, to obtain an *augmented* Trash-net and an augmented WaDaBa respectively as referred in Chapter 3.

Trash-net and Augmented-Trash-net

- The **Trash-net** data collection consists of images of six classes: glass, paper, cardboard, plastic, metal and trash. The dataset currently consists of 2527 images and is a slightly unbalanced dataset. We pick only the first five classes, i.e. Paper, Plastic, Metal, Glass, Cardboard.

- The **Augmented Trash-net** was created after we picked five labels for classification, i.e. Paper, Plastic, Metal, Glass, Cardboard from the original *trash-net* and used data augmentation as previous explained in subsection 3.3.5 that resulted in an augmented dataset of 10,000 images equally divided among the 5 classes.

Plastic Waste DataBase of images WaDaBa

- As explained in Chapter 3.3.1 the **WaDaBa** consists of 3960 images obtained after transformations, such as change of perspective, damage effect, and light sources shifting. The specific categories of plastic waste are the: a) PET, b) PE-HD, c) PP, d) PS. The *PET* class has as many images as all the other classes combined. So we have a case of outnumber samples of one class over the remaining classes, and is a highly unbalanced dataset.

- In a similar manner to what we did for the *augmented trash-net* we used data augmentation on the original *WaDaBa dataset* to result to a dataset of 8800 images equally divided among the 4 classes referred as **augmented WaDaBa**.

Cases to be explored

- **Slightly Unbalanced:** A problem of unbalanced classification where the distribution of examples is uneven by a small number.

  To obtain the needed custom dataset for this case, we created a dataset subnet. Our chosen dataset is the TrashNet data collection [2], which contains images of six different types of waste: glass, paper, cardboard, plastic, metal, and trash. We concentrate our efforts on images labeled as belonging to one of the five initial classes (i.e., we exclude Trash). The final experiment dataset contains 799 images: 140 images of cardboard, 165 images of glass, 142 images of metal, 190 images of paper, and 162 images of plastic; it is thus "slightly" unbalanced. To begin, we train a baseline CNN model on a *"slightly unbalanced"* training set of 530 TrashNet images. We then train the binary Independent Parallel (IP) CNN classifiers on their respective sub-datasets of the training set (specifically, the training set contains 87 cardboard, 112 glass, 89 metal, 135 paper, and 107 plastic images). Both the baseline and IP CNNs are based on the ResNet50 architecture (but with the baseline CNN performing multiclass classification, i.e., having as many output units as the number of classes).

  The training on this custom dataset presented in subsection 4.4.1 and the evaluation of the decision rules in subsection 4.4.2.

- **Highly Unbalanced:** An unbalanced classification problem where the distribution of examples is uneven by a large number.

  To obtain the needed custom dataset for this case, we created a dataset subnet. Our chosen dataset is the WaDaBa data collection [3], which contains images of four different types of plastic: PET, PE_HD, PS, and PP with *PET* class having as many images as all others combined. The final experiment dataset contains 1222 images: 630 images of PET, 203 images of PE_HD, 210 images of PP, and 179 images of PS; it is thus "highly" unbalanced. To begin, we train a baseline CNN model on a *"highly unbalanced"* training set of 915 WaDaBa images. We then train the binary Independent Parallel (IP) CNN classifiers on their respective sub-datasets of the training set (specifically, the training set contains 523 PET, 134 PE_HD, 144 PP, and 114 PS images). Both the baseline and IP CNNs are based on the ResNet50 architecture (but with the baseline CNN performing multiclass classification, i.e., having as many output units as the number of classes).

  The training on this custom dataset presented in subsection 4.4.3 and the evaluation of the decision rules in subsection 4.4.4.

In the experiments, there are three possible outcomes which will need to map with an acceptable decision rule. First, if there are clear margins across all classes, then the *"MSDR"* rules will be sufficient. Secondly, if there are shared features between two or more classes, there is a need for *"MANY-YES"* rules or special case of *"Weighted-soft-voting"*. Finally, if no classifier is able to classify the object as part of its own corresponding class, hence this requires the use of *"ALL-NO-PICK-MAX"* rule.

In Figure 4.6, we analyze the various combinations of decision rules that we will examine in order to construct an appropriate combination of decision rules to address the required three spots. First it should be mentioned that 3 categories of cases should be covered as mentioned above. Specifically the case of the discrete boundaries that only one classifier produces ($f_i^Y > f_i^N$), when this is not the case we need to cover the case of multiple classifiers that have ($f_i^Y > f_i^N$) and the worst case that we do not have any classifiers with ($f_i^Y > f_i^N$). Thus, we can see there are 4 potential combinations which follow the "MSDR-ONE-YES-THRESHOLD-YES" with two alternatives for the case of

"THRESHOLD-NO", followed by "MANY-YES-THRESHOLD-YES" with two alternatives ("MANY-YES-THRESHOLD-NO-PICK-MAX" and "MANY-YES-THRESHOLD-NO-PICK-RANDOM") for the case of "THRESHOLD-NO" and in the last part complemented with "ALL-NO-PICK-MAX" rule for the case that all classifiers predict that the item does not belong in their respective class. All the combinations mentioned above are tested in this thesis, but there is one that remains untested. Finally, we have one extra combination after "MSDR-ONE-YES-THRESHOLD-YES" the "MANY-YES-WEIGHTED-SOFT-VOTING" followed by "ALL-NO-PICK-MAX" in the combination.

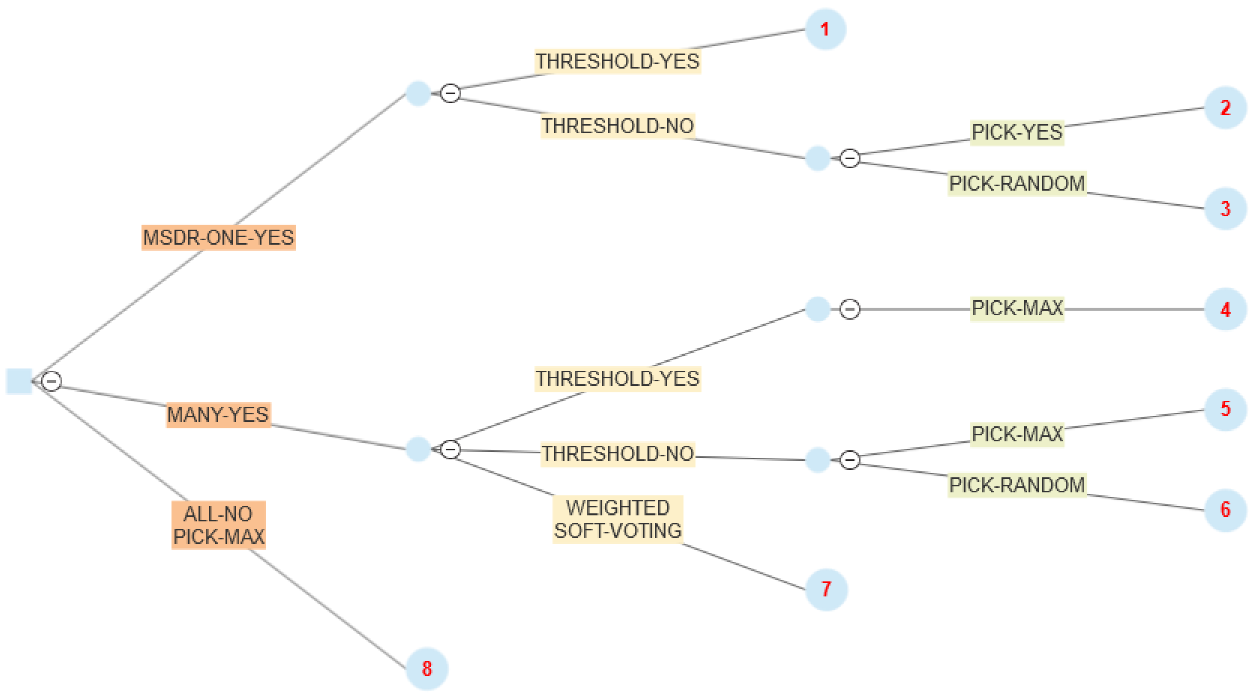


Figure 4.6: Decision Tree for different Decisions Rules

In more detail, we will refer as a combination number (e.g. Comb 2) a specific sequence of decision rules that maps to each of the three main *Discrete Families* of decision rules. In Table 4.4 we describe these combinations.

Discrete Family of Rules	Specific Decision Rules	Combinations				
		1	2	3	4	5
MSDR-ONE-YES	THRESHOLD-YES	●	●	●	●	●
	THRESHOLD-NO-PICK-YES	●		●		
	THRESHOLD-NO-PICK-RANDOM		●		●	
MANY-YES	THRESHOLD-YES-PICK-MAX	●	●	●	●	
	THRESHOLD-NO-PICK-MAX	●	●			
	THRESHOLD-NO-PICK-RANDOM			●	●	
	WEIGHTED-SOFT-VOTING					●
ALL-NO	PICK-MAX	●	●	●	●	●

Table 4.4: Decision Rules participation on different combinations

4.4 Results

In this section, we will build and assess our proposed approach for the datasets under consideration. In the Process 5 we outline the process of training and then testing the *IP_Network* described in Figure 4.2 with applying different combinations of rules as shown in Figure 4.6. We test this approach on every dataset we depict in subsection 4.3.1.

Process 5 Train Independent Parallel Network With Decision Rules

```
1: Modify the dataset for training IP classifiers
2:    Baseline Model Training:
3:    • Train a baseline model with the typical multi-class representation
4:    IP classifiers Training:
5:    • Train #classes independent binary CNN to act as classifiers in the respective
      sub-dataset.
6:    Evaluation Phase:
7:    • Testing the baseline model and the IP_network with different decision rules com-
      binations
8: Evaluate different combinations of decisions rules, assessing the frequency of each
   rule.
```

4.4.1 Training on the Trash-net Dataset

The dataset comprises five classes, such as Cardboard, Metal, Glass, Plastic, Paper. Since the ratio of images per class is not equal, it is a slightly unbalanced. The accuracy of each Independent classifier is shown in Table 4.5 in comparison to the multi-class baseline CNN model accuracy. In Table 4.6 the sensitivity and specificity metrics for the individual class provide by the baseline CNN model. We can see that the overall accuracy from Baseline_CNN is 3% lower for the “smaller” accuracy on independent classifiers. However, its sensitivity for each class is usually higher than of the IP_classifiers. This is due to the larger number of samples processed by the baseline CNN model, as well as the fact that the baseline model learns features for every class, rather than just two, as an independent classifier, allowing it to learn more accurate boundaries between classes. The Brier Score for the *IP_classifiers* stays below 7% with the best possible Brier score to be 0, for total accuracy.

Model Name	4-fold Accuracy	Specificity	Sensitivity	Brier Score
Multi-Class Baseline_CNN	89.58% (+/- 1.24%)			
IP_Cardboard classifier	96.94% (+/- 0.88%)	99.3%	78.3%	4.3%
IP_Glass classifier	93.40% (+/- 0.64%)	97.1%	78.2%	6.8%
IP_Metal classifier	93.30% (+/- 1.04%)	97.2%	84.1%	6.2%
IP_Paper classifier	96.42% (+/- 0.48%)	98.2%	87.3%	4.1%
IP_Plastic classifier	93.68% (+/- 1.13%)	98.3%	79.4%	5.6%

Table 4.5: Metrics for every independent classifier in Training phase

Class Name	Specificity	Sensitivity
Cardboard class	98.1%	93.1%
Glass class	92.2%	91.0%
Metal class	96.3%	85.2%
Paper class	98.2%	91.2%
Plastic class	98.3%	78.3%

Table 4.6: Specificity and Sensitivity of the multi-class Baseline Model for each Class

The confusion matrix of baseline multi-class CNN are shown in Figure 4.7. For comparison, the corresponding confusion matrices during training for each IP_Network are shown in Figure 4.8 for Cardboard and Glass classifiers, in Figure 4.9 for Metal and Paper classifiers and the remaining Plastic class on Figure 4.10.

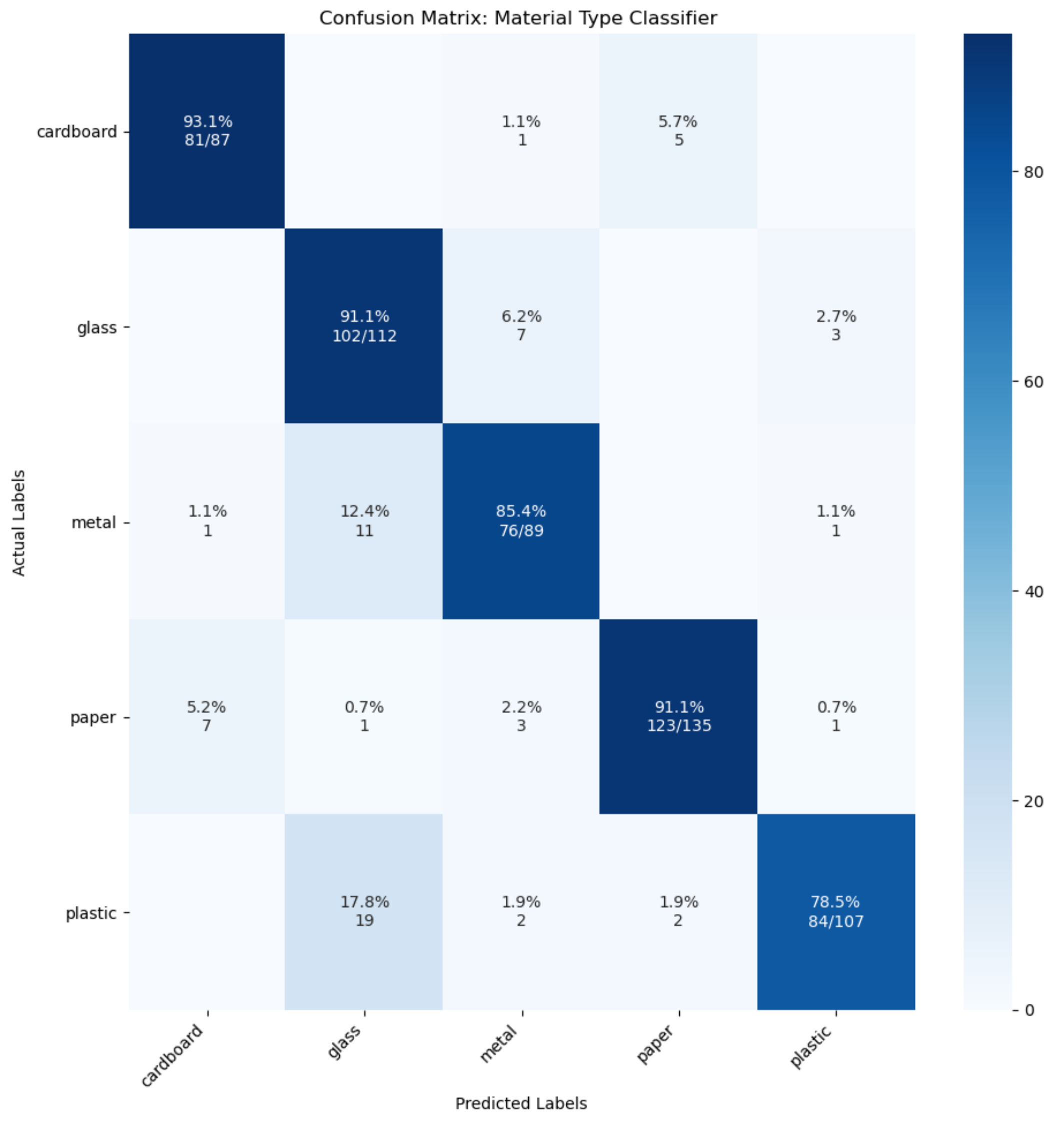
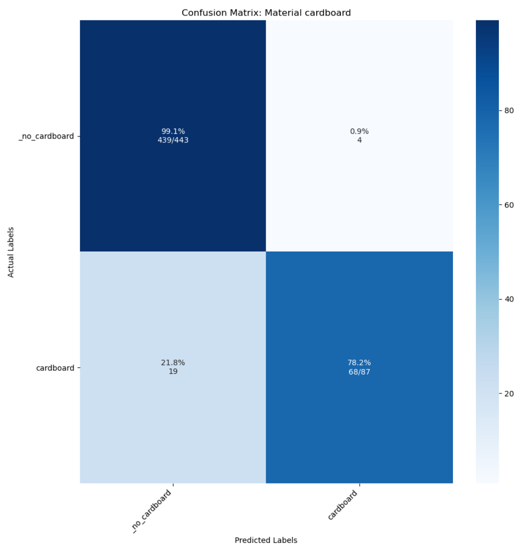
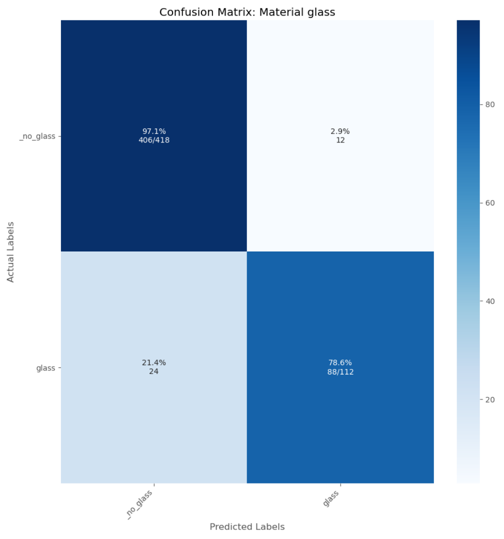


Figure 4.7: Multi-class Baseline Classifier Confusion Matrix

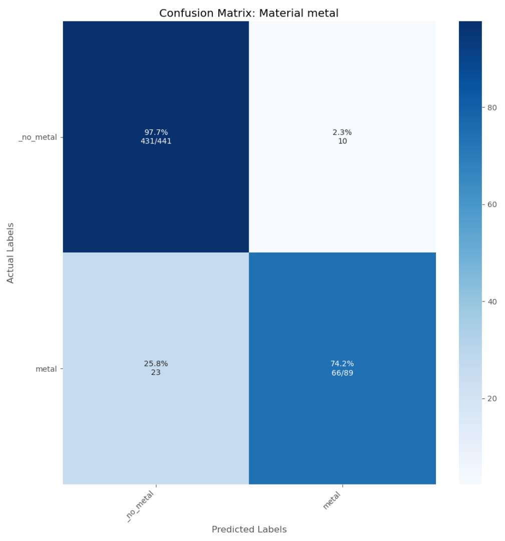


(a) Cardboard

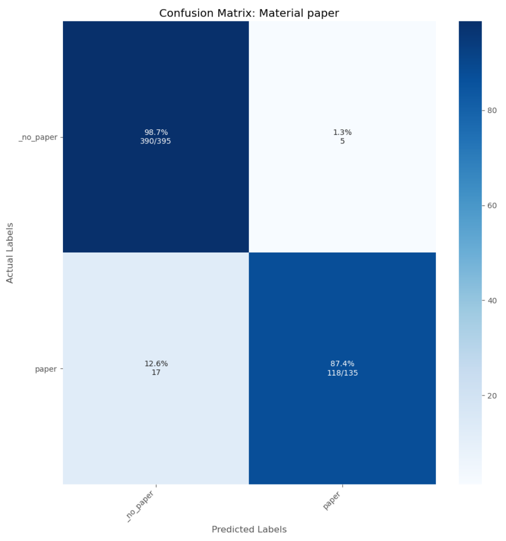


(b) Glass

Figure 4.8: Cardboard and Glass IP-Classifiers Confusion Matrix



(a) Metal



(b) Paper

Figure 4.9: Metal and Paper IP-Classifiers Confusion Matrix

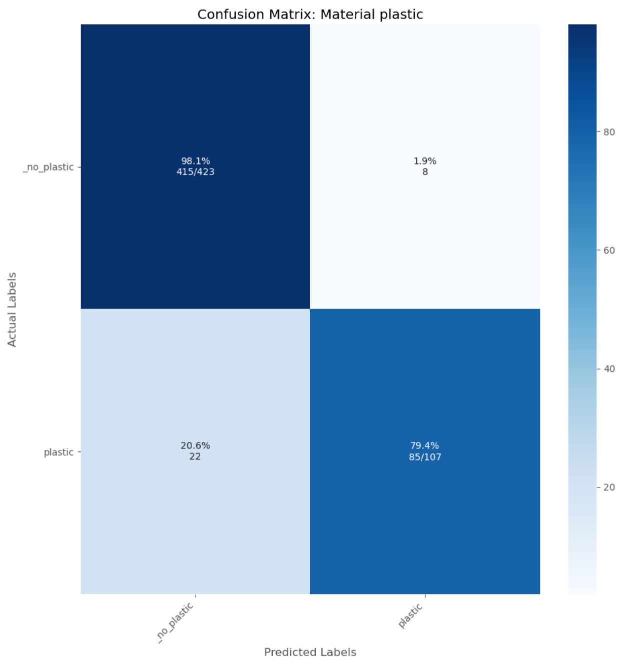


Figure 4.10: Plastic IP-Classifier Confusion Matrix

4.4.2 Decisions Rules Evaluation on Trash-net

Now we use the networks mentioned above as “frozen classifiers” and test in a testing set[12] of unseen images from both the multi-class baseline and IP networks, to evaluate the performance of the different combinations mentioned in Figure 4.6 against the baseline model. In Table 4.7 and Table 4.8 we can see the classification report metrics of *baseline network and Decision Rules Combinations* respectively. The results show that in every metric we have an increase ranging to 1-4 % from baseline model. Importantly, we have a 4% increase in overall accuracy.

We can infer the increase of the correct predictions in every class is due to the utilization of different targeted “decision rule paths”, instead of the classic multi-class CNN architecture. Figure 4.11 depicts the performance of Multi-class Baseline CNN and Figure 4.12 the performance of voting mechanism under decision rules implementation.

[12]consists of images that were chosen prior to the training of the networks under examination, ensuring that they have never seen it before and thus constituting a clean unseen test set.

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
cardboard	94%	92%	93%	53	98.2%	92.4%
glass	75%	89%	81%	53	91.3%	88.6%
metal	82%	85%	83%	53	94.4%	84.9%
paper	74%	91%	81%	55	90%	90.9%
plastic	77%	44%	56%	55	96.4%	43.6%
accuracy			80%	269		
macro avg	80%	80%	79%	269		
weighted avg	80%	80%	79%	269		

Table 4.7: Classification Report for Baseline network in Trash-net

Material	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
cardboard	94%	94%	94%	53	98.3%	94.3%
glass	82%	87%	84%	53	94.7%	86.7%
metal	83%	92%	88%	53	94.6%	92.4%
paper	75%	91%	82%	55	91.1%	90.9%
plastic	88%	55%	67%	55	97.9%	54.5%
accuracy			84%	269		
macro avg	84%	84%	83%	269		
weighted avg	84%	84%	83%	269		

Table 4.8: Classification Report for Decision Rules Combination [2-4] in Trash-net

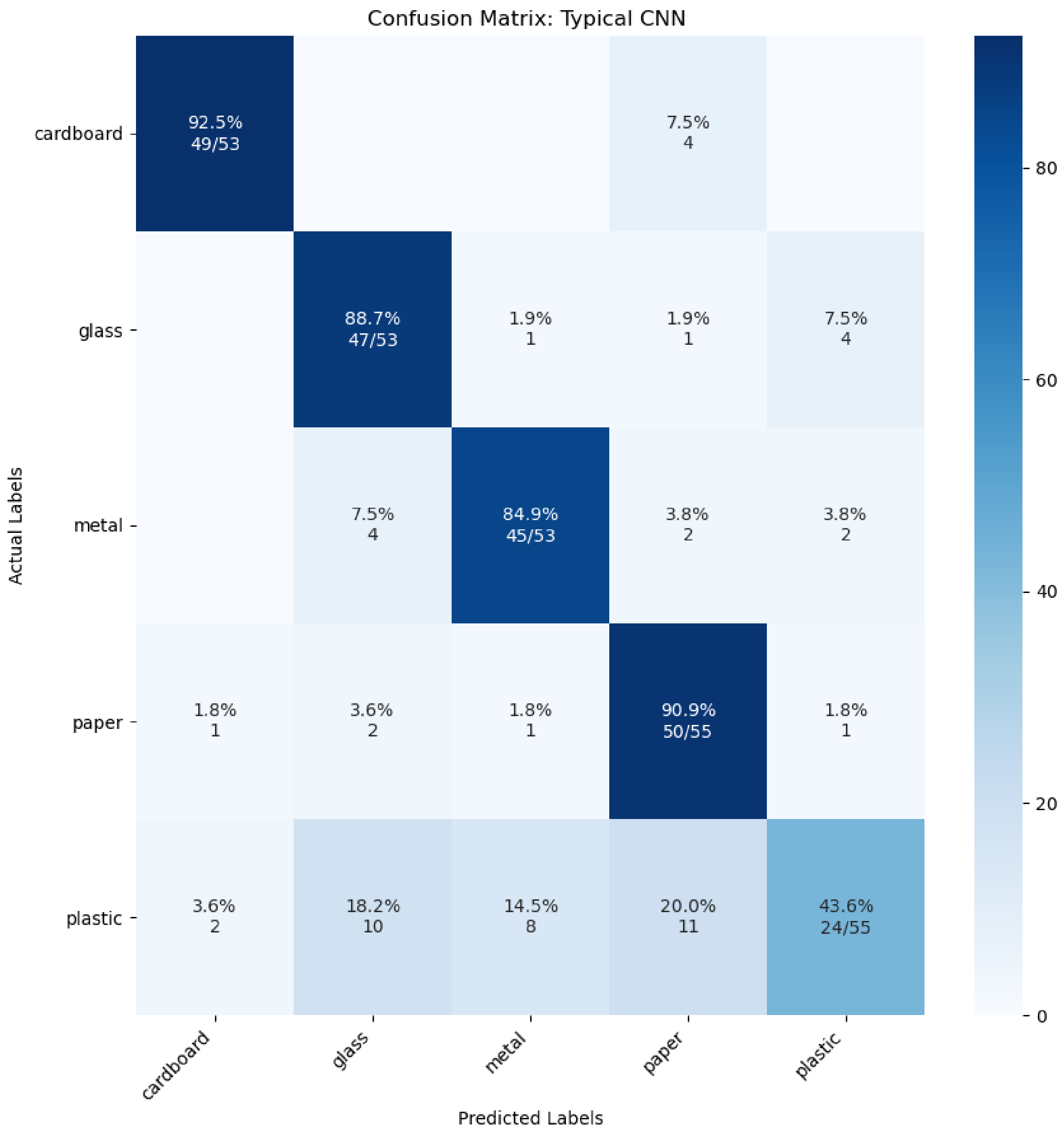


Figure 4.11: Multi-class Baseline Network Confusion Matrix

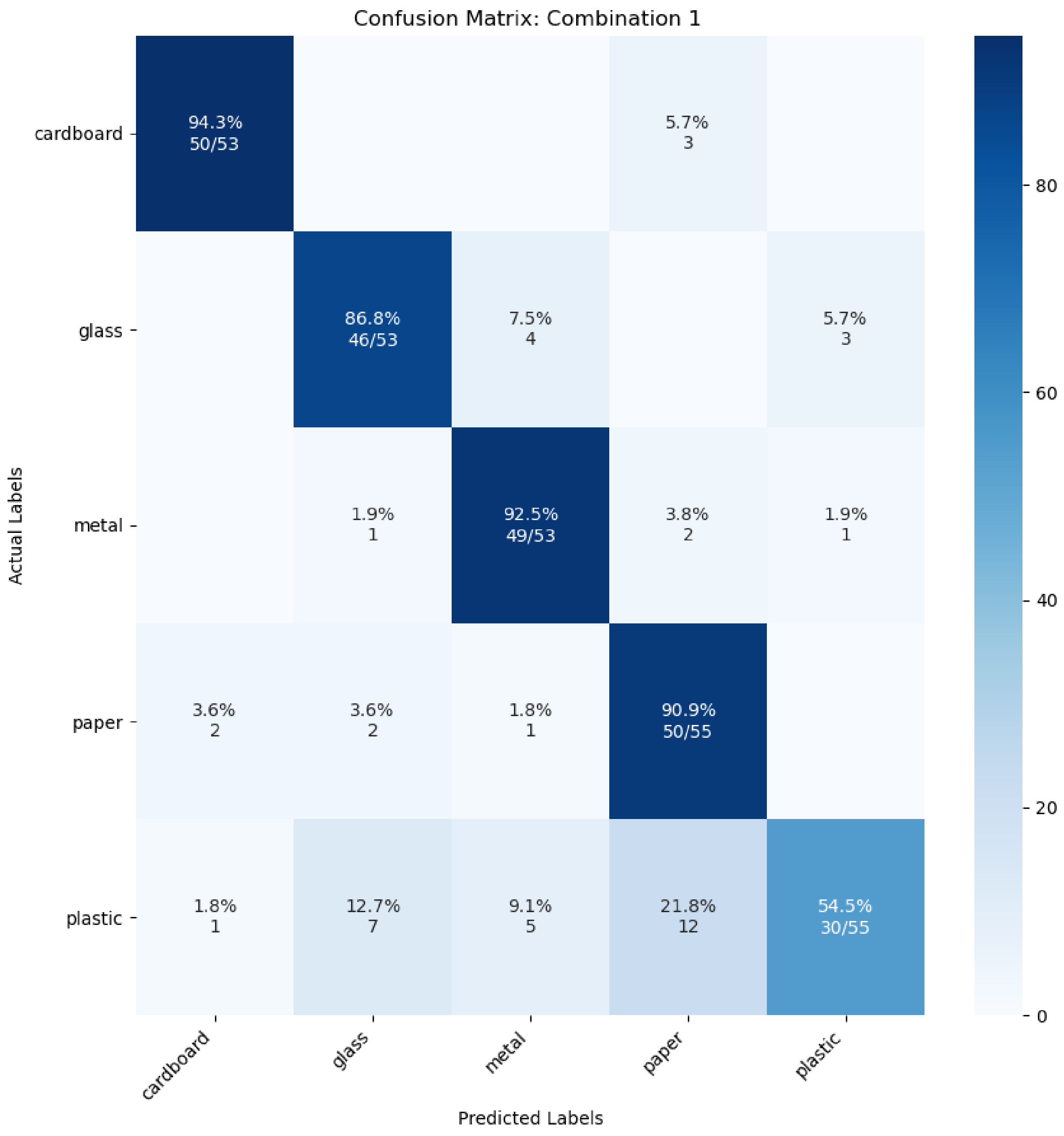


Figure 4.12: Decision Rules Combination Confusion Matrix

Name of Decision Rule	Decision Rules Frequency (sum=269)			
	Comb. 1	Comb. 2	Comb. 3	Comb. 4
MSDR-ONE-YES-THRESHOLD-YES	143	143	143	143
MSDR-ONE-YES-THRESHOLD-NO-PICK-YES	23	-	23	-
MSDR-ONE-YES-THRESHOLD-NO-PICK-RANDOM	-	23	-	23
MANY-YES-THRESHOLD-YES-PICK-MAX	42	42	42	42
MANY-YES-THRESHOLD-NO-PICK-MAX	9	9	-	-
MANY-YES-THRESHOLD-NO-PICK-RANDOM	-	-	9	9
ALL-NO-PICK-MAX	52	52	52	52

Table 4.9: Frequency of Decision Rules Activation

A decision set is analogous to a rule-based democracy, except that some rules may have a greater voting power. The rules in a set are either mutually exclusive or there is a strategy for resolving conflicts, such as majority voting, which may be weighted by the frequency with which the individual rules are activated or by other quality measures. We can estimate the frequency of occurrences of these decision rules by counting the number of occurrences one by one. The frequency of *decision rules* activation's are shown in table Table 4.9. As we stated in paragraph 4.3.1, each combination of decision rules must include rules that cover the three main cases (MSDR-YES, MANY-YES, ALL-NO). Furthermore, for the first two cases, we must add two extra rules for the state in which the probability threshold (THRESHOLD-NO case) is not reached, resulting in a sequence of five rules. This results in four distinct rule combinations, refer as Comb [1-4].

According to table 4.10, the "MSDR-ONE-YES" decision rule family has a 61.8 % activation rate, followed by the MANY-YES decision rule family with an 18.9 % activation rate and the "ALL-NO" decision rule family with a 19.3 % activation rate. In greater detail, the subcase of THRESHOLD-NO triggered a substantially lower proportion of the

Main Case of Decision Rules	Percentage of Activations			
	THRESHOLD-YES	THRESHOLD-NO	NO-THRESHOLD	Total (%)
MSDR-ONE-YES	53.2 %	8.6 %	-	61.8 %
MANY-YES	15.6 %	3.3 %	-	18.9 %
ALL-NO	-	-	19.3 %	19.3 %

Table 4.10: Decision rules frequency activation on Trashnet Dataset

first two cases, “MSDR-ONE-YES” and “MANY-YES”, activating at a rate of 6 times and 5 times lower, respectively.

As a result, the dataset exhibits a rather high percentage of ambiguous boundaries between the various classes. Particularly in these instances, only around 1 in 6 will require the employment of a decision rule to settle the problem when the defined threshold is not met (80% to consider the classifier result as “YES”). To support the perception that class boundaries are indistinguishable, the “ALL-NO” decision rule is engaged only when no independent classifier predicts that the item belongs to its own class. Here we have a similar activation rate as in the previous case (“MANY-YES”), which leads us to the conclusion that there is a large degree to which contained items’ classes are related to each other.

Finally, we can see that in one of the two prediction, there are distinct boundaries between the classes, but we also have a prediction with a higher probability of 80%. This is a relatively low prediction rate with distinct margins between classes for a slightly unbalanced dataset, but it demonstrates the importance of statistical analysis through the frequency of decision rule activation.

4.4.3 Training on the WaDaBa Dataset

The dataset comprises four type of plastic, such as PET, PE_HD, PP, PS with the *PET* class having as many images as all others combined (see subsection 4.3.1). Since the ratio of images in one class is the same with the sum of all the others, this is a case of *highly unbalanced* dataset. The accuracy of each Independent classifier shown in Table 4.11 in comparison to the multi-class baseline CNN model accuracy. In Table 4.12 the sensitivity and specificity metrics for the individual class provide by the baseline CNN model. We can see that the overall accuracy from Baseline_CNN is in average equal with any accuracy on independent classifiers. However, its sensitivity for each class is usually higher than of the IP_classifiers. This is due to the larger number of samples processed by the baseline CNN model, as well as the fact that the baseline model learns features for every class, rather than just two, as an independent classifier, allowing it to learn more accurate boundaries between classes. The Brier Score for the *IP classifiers* stays below 3% with the best possible Brier score to be 0, for total accuracy.

Model Name	**4-fold Accuracy**	**Specificity**	**Sensitivity**	**Brier Score**
Multi-class Baseline_CNN	98.39% (+/- 0.61%)			
PET classifier	98.66% (+/- 0.61%)	99.2%	99.3%	0.5%
PE_HD classifier	99.45% (+/- 0.17%)	97.1%	100.0%	0.4%
PP classifier	96.28% (+/- 0.92%)	87.3%	98.3%	3%
PS classifier	97.93% (+/- 0.65%)	99.1%	79.2%	2.7%

Table 4.11: Metrics for every independent classifier in Training phase

Class Name	**Specificity**	**Sensitivity**
PET class	99.3%	99.1%
PE_HD class	99.4%	100.0%
PP class	99.1%	95.2%
PS class	99.2%	98.4%

Table 4.12: Specificity and Sensitivity of the Baseline Model for every class

The confusion matrix of baseline multi-class baseline CNN are shown in Figure 4.13. For comparison the corresponding confusion matrixs during training for each IP_Network are shown in Figure 4.14 for PET and PE_HD classifiers, in Figure 4.15 for PP and PS classifiers.

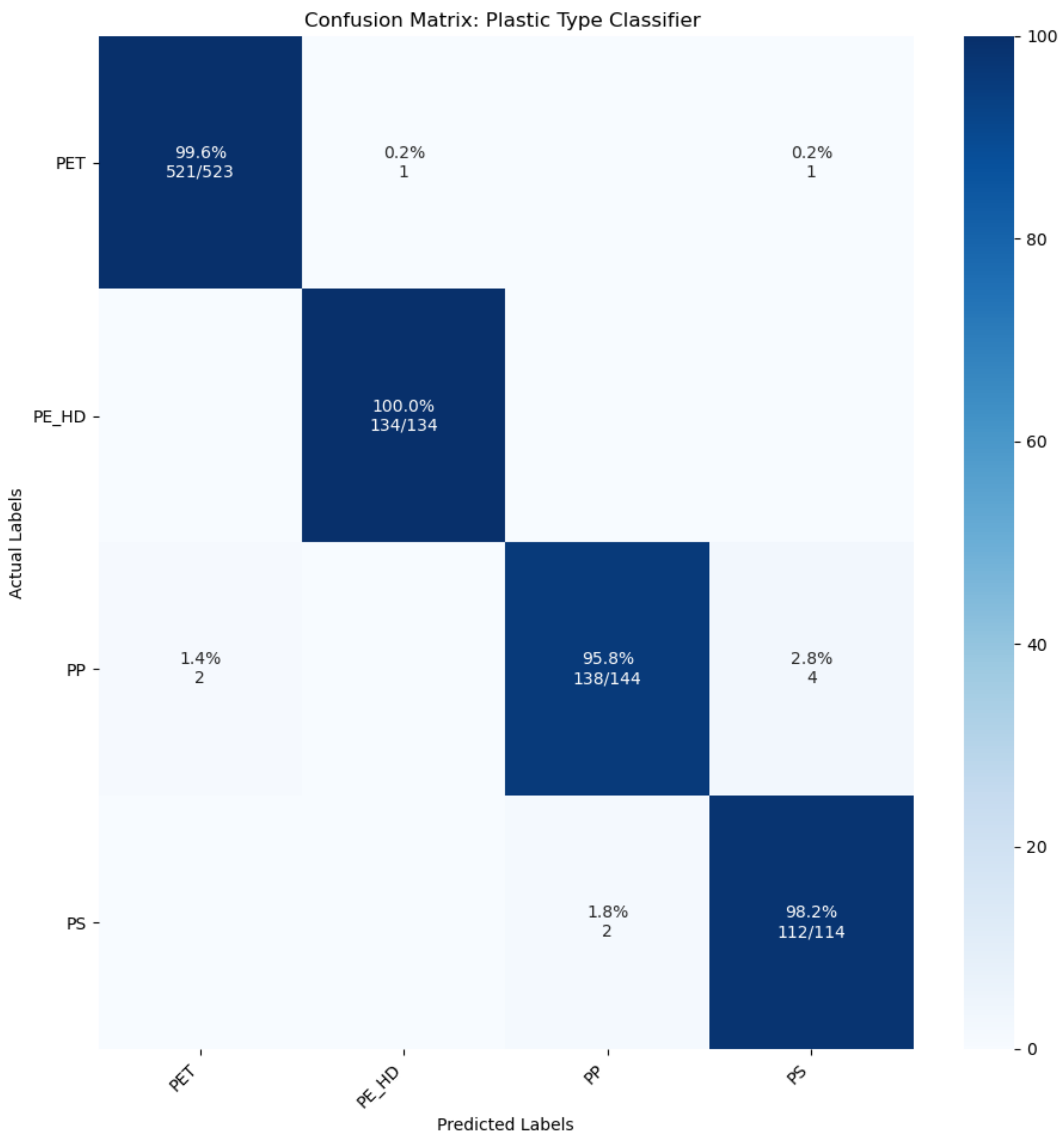
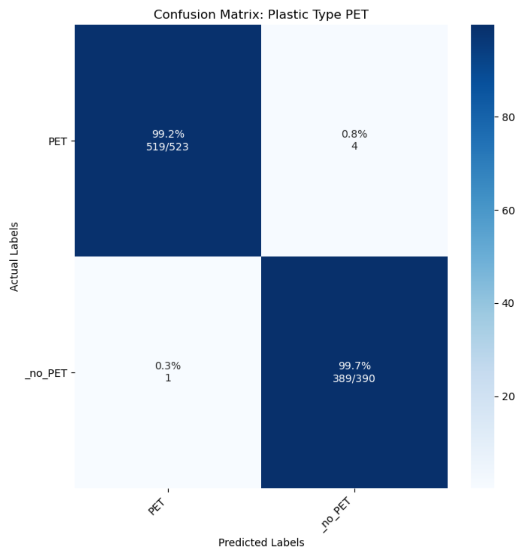
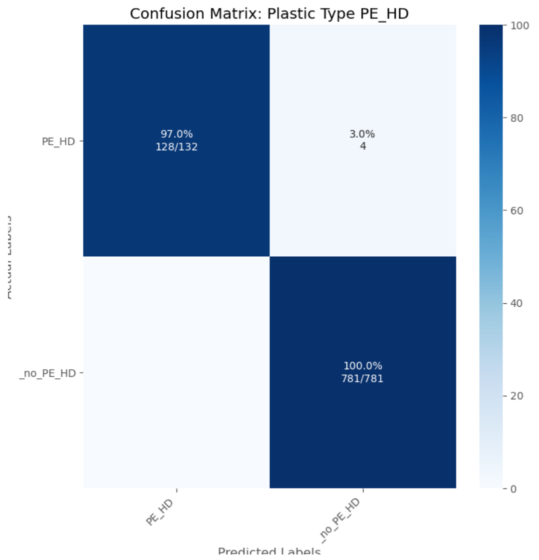


Figure 4.13: Baseline Classifier Confusion Matrix

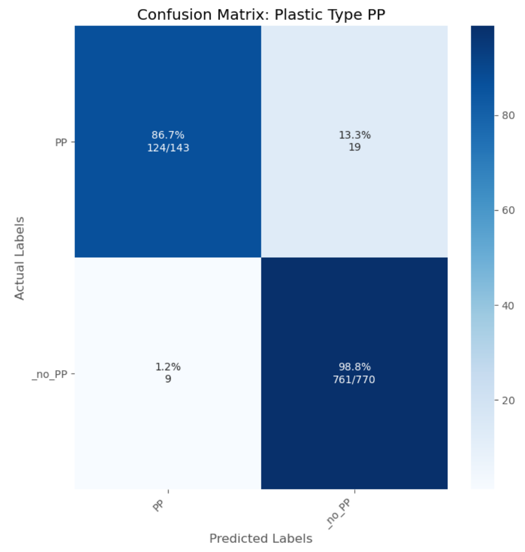


(a) PET

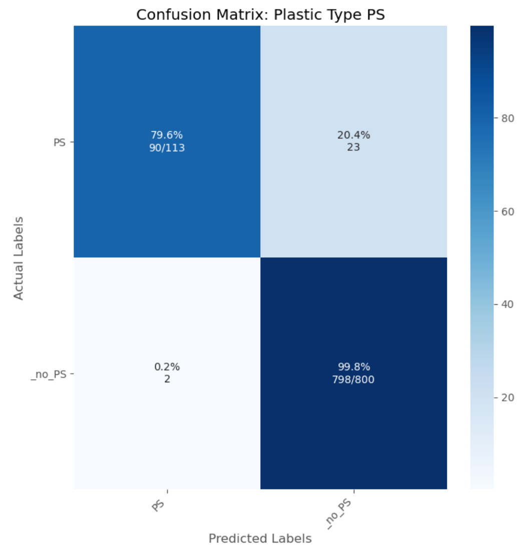


(b) PE_HD

Figure 4.14: PET and PE_HD IP-Classifiers Confusion Matrix



(a) PP



(b) PS

Figure 4.15: PP and PS IP-Classifiers Confusion Matrix

4.4.4 Decision Rules Evaluation on WaDaBa

Now we use the networks mentioned above as "frozen classifiers" and test in a hold_out set of unseen images from both the baseline and IP networks, to evaluate the performance of the different combinations mentioned in Figure 4.6 and Table 4.4, specifically comb 1-4, against the baseline model. In Table 4.13 and Table 4.14 we can see the classification report metrics of *baseline network and Decision Rules Combinations* respectively. The results show that in every metrics, the Decision rules does not manage to surpass in any way the baseline model metrics. Thus, in this case the primary purpose of this framework is not met and only the secondary objectives, those of characterizing the dataset regarding its "homogeneity" and "balancedness" properties, may be achieved.

Figure 4.16 depicts the performance of the Multi-class Baseline CNN, and Figure 4.17 the performance of voting mechanism with the decision rules implementation.

Plastic Type	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
PET	74%	97%	84%	107	79%	97.1%
PE_HD	97%	93%	95%	69	98.8%	92.6%
PP	56%	53%	54%	66	87.9%	53%
PS	97%	57%	72%	65	99%	56.9%
accuracy			78%	307		
macro avg	81%	75%	76%	307		
weighted avg	80%	75%	78%	307		

Table 4.13: Classification Report for Baseline network in WaDaBa

Plastic Type	**Precision**	**Recall**	**f1-score**	**support**	**Specificity**	**Sensitivity**
PET	74%	100%	85%	107	76.3%	100%
PE_HD	95%	91%	93%	69	98.2%	91.3%
PP	51%	48%	50%	66	86.6%	48.5%
PS	57%	48%	64%	65	99%	47.6%
accuracy			76%	307		
macro avg	75%	72%	73%	307		
weighted avg	78%	76%	75%	307		

Table 4.14: Classification Report for Decision Rules Combination [2-4] in WaDaBa

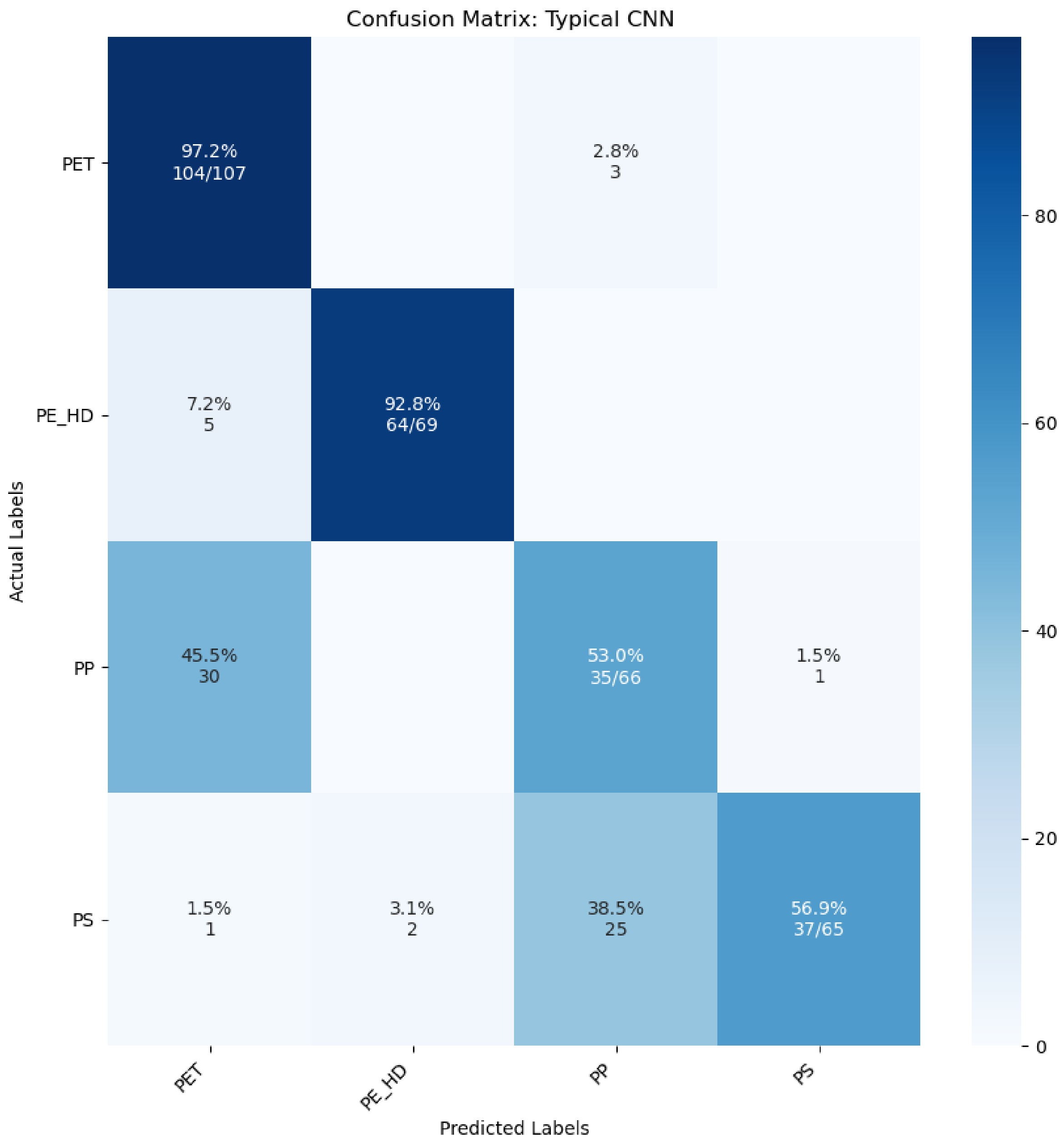


Figure 4.16: Baseline Network Confusion Matrix

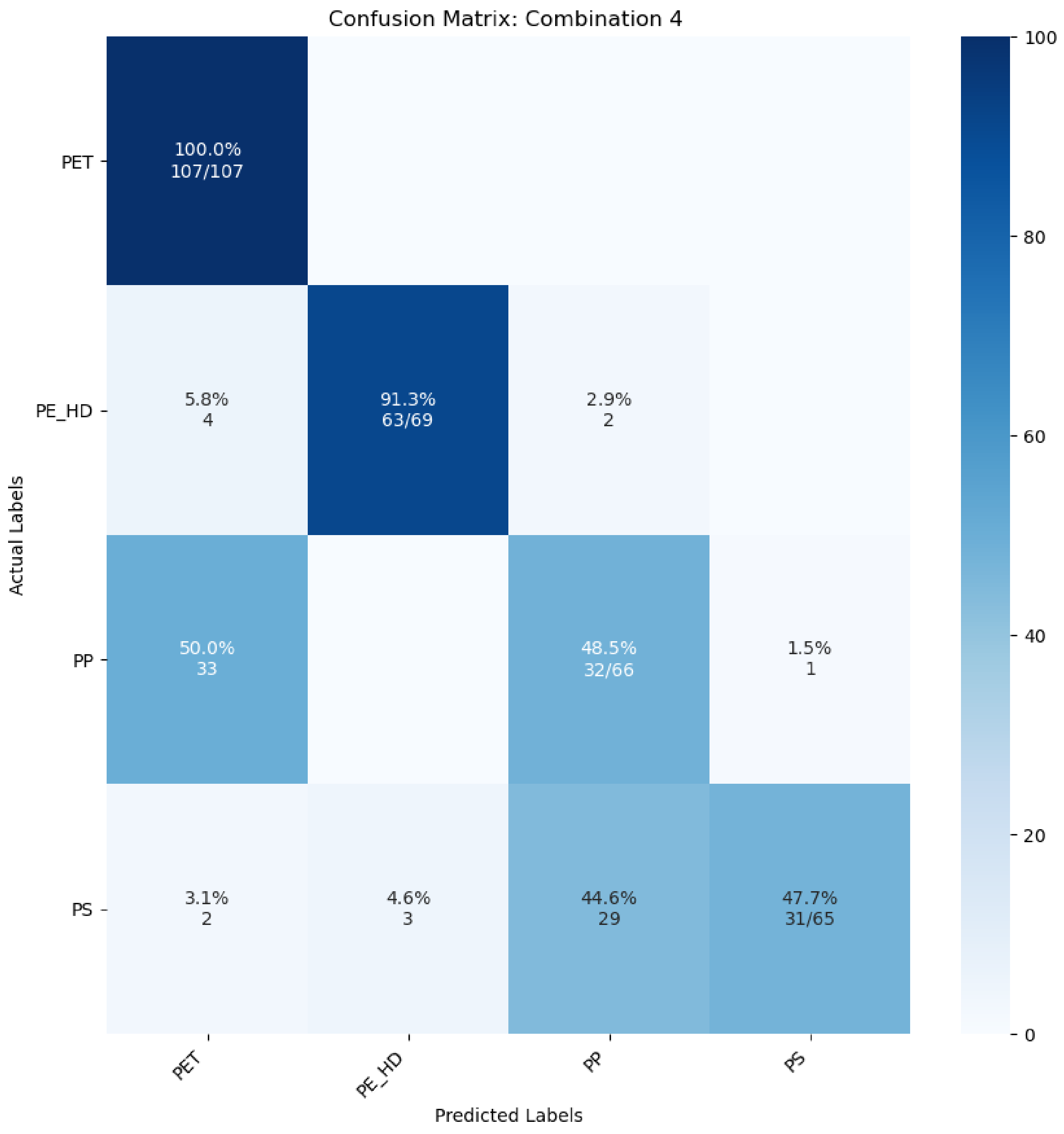


Figure 4.17: Decision Rules Combination Confusion Matrix

Name of Decision Rule	Decision Rules Frequency (sum=307)			
	Comb. 2	Comb. 3	Comb. 4	Comb. 5
MSDR-ONE-YES-THRESHOLD-YES	212	212	212	212
MSDR-ONE-YES-THRESHOLD-NO-PICK-YES	25	-	25	-
MSDR-ONE-YES-THRESHOLD-NO-PICK-RANDOM	-	25	-	25
MANY-YES-THRESHOLD-YES-PICK-MAX	13	13	13	13
MANY-YES-THRESHOLD-NO-PICK-MAX	4	4	-	-
MANY-YES-THRESHOLD-NO-PICK-RANDOM	-	-	4	4
ALL-NO-PICK-MAX	53	53	53	53

Table 4.15: Frequency of Decision Rules Activation

A decision set is analogous to a rule-based democracy, except that some rules may have a greater voting power. The rules in a set are either mutually exclusive or there is a strategy for resolving conflicts, such as majority voting, which may be weighted by the frequency with which the individual rules are activated or by other quality measures. We can estimate the frequency of occurrences of these decision rules by counting the number of occurrences one by one. The frequency of *decision rules* activation's are shown in table Table 4.15. As we stated in paragraph 4.3.1, each combination of decision rules must include rules that cover the three main cases (MSDR-YES, MANY-YES, ALL-NO). Furthermore, for the first two cases, we must add two extra rules for the state in which the probability threshold (THRESHOLD-NO case) is not reached, resulting in a sequence of five rules. This results in four distinct rule combinations, refer as Comb [1-4].

According to table Table 4.16, the "MSDR-ONE-YES" decision rule family has a 77.2 % activation rate, followed by the MANY-YES decision rule family with an 5.5 % activation rate and the "ALL-NO" decision rule family with a 17.3 % activation rate. In

Main Case of Decision Rules	Percentage of Activations			
	THRESHOLD-YES	THRESHOLD-NO	No-THRESHOLD	Total (%)
MSDR-ONE-YES	69.1 %	8.1 %	-	77.2 %
MANY-YES	4.2 %	1.3 %	-	5.5 %
ALL-NO	-	-	17.3 %	17.3 %

Table 4.16: Decision rules frequency activation on WaDaBa Dataset

greater detail, the subcase of THRESHOLD-NO triggered a substantially lower proportion of the first two cases, "MSDR-ONE-YES" and "MANY-YES", activating at a rate of 8.5 times and 3 times lower, respectively.

As a result, the dataset exhibits a rather high percentage of clear boundaries between the various classes. Particularly in these instances, there are not many samples images that activated the "THRESHOLD-NO" decision rule to resolve the problem when the pre-defined threshold is not met (80% to consider the classifier result as "YES"). To support the perception that class boundaries are distinct between plastic type class, the "ALL-NO" decision rule is engaged only when no independent classifier predicts that the item belongs to its own class, and all these predictions are the wrong predictions for the PP class that predicted as PET and PS plastic type that wrongly predict as PP class type respectively as shown in Figure 4.17.

Finally, we can see that in 77.2 % of prediction, there are distinct boundaries between the classes, but we also have a prediction with a higher probability of 80%. The WaDaBa dataset is highly unbalanced but has very clear boundaries between distinct classes, indicating that the dataset largely possesses the mutual exclusivity property, and all that is required is to better adjust the weights for PP and PS classifiers according to the corresponding Bries score to achieve better accuracy and minimize errors.

Making a final comparison between Table 4.9 and Table 4.15 we see that there is a significant difference in the activation frequency of the "MSDR-ONE-YES" and "MANY-YES" decision rules, with fluctuations depending on how distinct/clear the boundaries are among classes and the degree of balance of the dataset, but almost no difference in the activating decision rules, "ALL-NO". Even in the classic case of the base model there are cases where the probability of choosing a class is less than 50% but it remains the highest of all the others.

4.5 Conclusions and Future Work

In Chapter 4 we present a novel, generic one-versus-all classification framework in this work that incorporates multiple non-trivial decision rules for effectively aggregating the opinions of several independent parallel CNNs. Several of the proposed rules explicitly exploit the mutual exclusivity property inherent in numerous interesting image datasets, while others are inspired by randomized social choice schemes and assign a class to the item under consideration by selecting from a distribution of alternatives. When clear class boundaries exist, the framework takes advantage of them, while still being able to confidently assign items to classes when they do not. The proposed methodology paves the way for a plethora of fascinating future study. To begin, the effectiveness of various combinations of decision rules (e.g., combining max-picking rules with randomized ones in various cases) must be verified via systematic experimentation on a variety of relevant datasets; and statistics on the frequency with which different rules are triggered in various datasets must be compiled. The results obtained in this section can be used to implicitly characterize a dataset in terms of “balancedness” and “homogeneity” (e.g., if a “MANY-YES” rule regularly applies, this can indicate that a dataset is mostly have blurry boundaries between classes and the mutually exclusive property is not very dominant). Finally, in terms of developing effective and efficient methods for setting and adjusting the Independent Classifier’s weights, we intend to use uncertainty metrics from the scoring rules literature [80] to objectively characterize our confidence in a classifier’s performance.

Chapter 5

Conclusion

5.1 Conclusions and Future Work

Two ensemble learning approaches for recyclables classification are described in this thesis, implemented and incorporated using Tensorflow, Keras and Python. In the first approach , reffered as "Dual-branch" CNN we exploit the shared wisdom from data and explore how various datasets can be combined to improve accuracy and create a stable network architecture. The second approach implements a novel classification framework that employs voting mechanism, based on outputs from Independent Parallel classifiers. We make use of pre-trained Convolution Neural Networks for classification, and replace "normal softmax" with softmax with temperature scaling for better calibration on neural network. We provide a thorough evaluation of our methods, and discuss the impact of our work on both recycle waste classification and optimization of ensemble learning techniques.

5.2 Future Work

We have already mentioned ideas for future work in the end of Chapters 3 and 4. We now discuss further ideas for extending the work described in this thesis. The lack of interpretablity discourages practitioners from using these solutions even though the strategies developed improves performance [69]. As a result, the learned strategies should be explored and interpreted through the opening of the deep "black box".

It is very important to manage reliability because no algorithm can be flawless in all cases. In this context, in this work of [81] examines the issue of end-to-to-end object detectors that where a probabilistic confidence score is explicitly pre-multiplied into the incoming activations to modulate confidence. They use a relaxed version of the "softmax" function that can be incorporate in out approaches.

A interesting approach is to employ the voting mechanism of Independent Parallel classifiers on object detection system, combining with ensemble learning to create new algorithm for object detection like what is done in the approach of [82].

Our architecture takes advantage of the combined knowledge of its sub-nets across the various datasets, with local training only on the site of each dataset. We can learn from multiple sources and accumulate knowledge without mingling the data samples. The combination scheme exploits the individual outcomes of subnets in a way that increases confidence in final decision making process. We propose to deploy this approach in real-time applications as an adaptively updated Federated learning strategy in following stages of this research.

References

[1] Pan, S., Yang, Q.: A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering **22**(10) (2010) 1345–1359 2

[2] Thung, G., Yang, M.: "Classification of Trash for Recyclability Status", Stanford University CS229 [online]. (2016) http://cs229.stanford.edu/proj2016/report/ThungYang-ClassificationOfTrashForRecyclabilityStatus-report.pdf. 2, 12, 43, 44, 52, 86

[3] Bobulski, J., Piatkowski, J.: Pet waste classification method and plastic waste database - wadaba. In Choraś, M., Choraś, R.S., eds.: Image Processing and Communications Challenges 9, Cham, Springer International Publishing (2018) 57–64 2, 12, 43, 44, 52, 87

[4] Anasa Robotic Sorter: Autonomous robotic system for urban waste recycling (2022) 3

[5] Vogiatzis, A., Chalkiadakis, G., Moirogiorgou, K., Livanos, G., Papadogiorgaki, M., Zervakis, M.: Dual-branch cnn for the identification of recyclable materials. In: 2021 IEEE International Conference on Imaging Systems and Techniques (IST). (2021) 1–6 5

[6] Vogiatzis, A., Chalkiadakis, G., Moirogiorgou, K., Zervakis, M.: A novel one-vs-rest classification framework for mutually supported decisions by independent parallel classifiers. In: 2021 IEEE International Conference on Imaging Systems and Techniques (IST). (2021) 1–6 5

[7] Aly, M.: Survey on Multiclass Classification Methods. Tech. rep., Caltech, USA (2005) 10

[8] Sejnowski, T., Rosenberg, C.: Parallel networks that learn to pronounce english text. Complex Syst. **1** (1987) 10

[9] Lorena, A., Carvalho, A., Gama, J.: A review on the combination of binary classifiers in multiclass problems. Artificial Intelligence Review **30** (12 2008) 19–37 10

[10] Ban, T., Abe, S.: Implementing multi-class classifiers by one-class classification methods. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE (2006) 327–332 10

[11] a. Bagheri, M., Montazer, G.A., Escalera, S.: Error correcting output codes for multiclass classification: Application to two image vision problems. In: The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012). (2012) 508–513 10

[12] Zhou, Z.H.: Ensemble Methods: Foundations and Algorithms. 1st edn. Chapman , Hall/CRC (2012) 10, 35

[13] Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Analysis & Applications **5**(2) (2002) 210–220 10

[14] Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006) 10, 69

[15] ul Hassan, M., Mulhem, P., Pellerin, D., Qu©not, G.: Explaining visual classification using attributes. In: 2019 International Conference on Content-Based Multimedia Indexing (CBMI). (2019) 1–6 11

[16] He, S., Schomaker, L.: Open set chinese character recognition using multi-typed attributes. arXiv preprint arXiv:1808.08993 (2018) 11

[17] Tian, M., Pu, B., Chen, Y., Zhu, Z.: Consumer's waste classification intention in china: An extended theory of planned behavior model. Sustainability **11** (12 2019) 11

[18] Bonifazi, G., Serranti, S., Bonoli, A., DallAra, A.: Innovative recognition-sorting procedures applied to solid waste: The hyperspectral approach. Sustainable Development **120** (2009) 885–894 11

[19] Flores, M., Tan, J.: Literature review of automated waste segregation system using machine learning: A comprehensive analysis. International journal of simulation: systems, science and technology (2019) 11

[20] Shi, C., Tan, C., Wang, T., Wang, L.: A waste classification method based on a multilayer hybrid convolution neural network. Applied Sciences **11**(18) (2021) 11

[21] Mao, w.l., Chen, W.C., Wang, C.T., Lin, Y.H.: Recycling waste classification using optimized convolutional neural network. Resources, Conservation and Recycling **164** (01 2021) 105132 11

[22] Sashaank, S.: Waste classification data. https://www.kaggle.com/techsash/waste-classification-data (June 2019) 12

[23] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4) (December 1989) 541551 14

[24] Narayan, S.: The generalized sigmoid activation function: Competitive supervised learning. Information Sciences **99**(1) (1997) 69–82 19

[25] Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S.: Activation functions: Comparison of trends in practice and research for deep learning (12 2020) 19

[26] Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018) 20

[27] Pedamonti, D.: Comparison of non-linear activation functions for deep neural networks on mnist classification task. ArXiv **abs/1804.02763** (2018) 20

[28] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323** (1986) 533–536 23

[29] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12** (July 2011) 2121–2159 25

[30] Zeiler, M.D.: ADADELTA: an adaptive learning rate method. **abs/1212.5701** (2012) 26

[31] Hinton, G.: Rmsprop algorithm (2010) [online] `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. 26

[32] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. **abs/1412.6980** (2014) 27

[33] Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. CoRR **abs/1706.04599** (2017) 28

[34] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015) 28

[35] Ho, T.K.: Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. Volume 1., IEEE (1995) 278–282 33

[36] Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems, Berlin, Heidelberg, Springer Berlin Heidelberg (2000) 1–15 34

[37] Ueda, N., Nakano, R.: Generalization error of ensemble estimators. Proceedings of International Conference on Neural Networks (ICNN'96) **1** (1996) 90–95 vol.1 34

[38] Gandhi, I., Pandey, M.: Hybrid ensemble of classifiers using voting. In: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT). (2015) 399–404 34

[39] Wolpert, D.: Stacked generalization. Neural Networks **5** (12 1992) 241–259 35

[40] Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D., eds.: Handbook of Computational Social Choice. Cambridge University Press (2016) 36

[41] Chalkiadakis, G., Wooldridge, M.J.: Weighted voting games. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D., eds.: Handbook of Computational Social Choice. Cambridge University Press (2016) 377–396 36

[42] Chalkiadakis, G., Wooldridge, M., Moulin, H. In: Weighted Voting Games. Cambridge University Press (2016) 377396 36

[43] Arrow, K.: Social Choice and Individual Values. Wiley: New York (1951) 37

[44] Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society. Series B (Methodological) **36**(2) (1974) 111–147 37

[45] Ferri, C., Hern'ndez-Orallo, J., Modroiu, R.: An experimental comparison of performance measures for classification. Pattern Recognition Letters **30**(1) (2009) 27–38 39

[46] Aral, R.A., Keskin, .R., Kaya, M., Hac±´Amerolu, M.: Classification of trashnet dataset based on deep learning models. In: 2018 IEEE International Conference on Big Data (Big Data). (2018) 2058–2062 40, 59

[47] Frost, S., Tor, B., Agrawal, R., Forbes, A.: Compostnet: An image classifier for meal waste. (10 2019) 1–4 40

[48] Mittal, G., Yagnik, K., Garg, M., Krishnan, N.: Spotgarbage: smartphone app to detect garbage using deep learning. (09 2016) 940–945 40

[49] Chu, Y., Huang, C., Xie, X., Tan, B., Kamal, S., Xiong, X.G.: Multilayer hybrid deep-learning method for waste classification and recycling. Computational Intelligence and Neuroscience **2018** (11 2018) 1–9 40

[50] Bircanolu, C., Atay, M., Beer, F., Gen§, ., K±zrak, M.A.: Recyclenet: Intelligent waste sorting using deep neural networks. In: 2018 Innovations in Intelligent Systems and Applications (INISTA). (2018) 1–7 40, 59

[51] Xu, X., Li, W., Ran, Q., Du, Q., Gao, L.: Multisource remote sensing data classification based on convolutional neural network. IEEE Transactions on Geoscience and Remote Sensing **PP** (10 2017) 1–13 41

[52] Hirata, D., Takahashi, N.: Ensemble learning in cnn augmented with fully connected subnetworks (03 2020) 41

[53] Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L.R., Wiering, M.A.: One-vs-one classification for deep neural networks. Pattern Recognition **108** (2020) 107528 41

[54] Bobulski, J., Kubanek, M.: The triple histogram method for waste classification. AIP Conference Proceedings **2293** (11 2020) 420004 42, 44, 56

[55] Costa, B., Bernardes, A., Pereira, J., Zampa, V., Pereira, V., Matos, G., Soares, E., Soares, C., Silva, A.: Artificial intelligence in automated sorting in trash recycling. (10 2018) 198–205 42, 44

[56] House, B.W., Capson, D.W., Schuurman, D.C.: Towards real-time sorting of recyclable goods using support vector machines. In: Proceedings of the 2011 IEEE International Symposium on Sustainable Systems and Technology. (2011) 1–6 42, 44

[57] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2018) 43

[58] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015) 43

[59] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015) 43

[60] Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: A comprehensive review. Neural Computation **29** (06 2017) 1–98 44

[61] Li, S., Yao, Y., Hu, J., Liu, G., Yao, X., Hu, J.: An ensemble stacked convolutional neural network model for environmental event sound recognition. Applied Sciences **8**(7) (2018) 45

[62] Vepakomma, P., Gupta, O., Swedish, T., Raskar, R.: Split learning for health: Distributed deep learning without sharing raw patient data (2018) 45

[63] Vepakomma, P., Balla, J., Pal, Ph.D., S., Raskar, R.: Splintering with distributions and polytopes: Unconventional schemes for private computation (10 2020) 45

[64] McMahan, H.B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data (2017) 45

[65] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? (2014) 50

[66] Bloice, M.D., Stocker, C., Holzinger, A.: Augmentor: An image augmentation library for machine learning (2017) 54

[67] BRIER, G.W.: Verification of forecasts expressed in terms of probability. Monthly Weather Review **78**(1) (01 Jan. 1950) 1 – 3 71, 83

[68] Gao, F., Han, L.: Implementing the nelder-mead simplex algorithm with adaptive parameters. Computational Optimization and Applications **51** (05 2012) 259–277 74

[69] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J.V., Lakshminarayanan, B., Snoek, J.: Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In: NeurIPS. (2019) 83, 84, 109

[70] Akasiadis, C., Chalkiadakis, G.: Agent cooperatives for effective power consumption shifting. In: AAAI. (2013) 83

[71] Robu, V., Kota, R., Chalkiadakis, G., Rogers, A., Jennings, N.: Cooperative virtual power plant formation using scoring rules. In: Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12) (22/07/12). (August 2012) 370–376 83

[72] Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Learning from imbalanced data sets. Volume 11. Springer (2018) 83, 85

[73] Predd, J.B., Seiringer, R., Lieb, E.H., Osherson, D.N., Poor, H.V., Kulkarni, S.R.: Probabilistic coherence and proper scoring rules. IEEE Transactions on Information Theory **55**(10) (Oct 2009) 47864792 84

[74] DeGroot, M.H., Fienberg, S.E.: The comparison and evaluation of forecasters. Journal of the Royal Statistical Society. Series D (The Statistician) **32**(1/2) (1983) 12–22 84

[75] Murphy, A.H.: A new vector partition of the probability score. Journal of Applied Meteorology and Climatology **12**(4) (01 Jun. 1973) 595 – 600 84

[76] Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association **102**(477) (2007) 359–378 84

[77] Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5** (2016) 221–232 85

[78] Branco, P., Torgo, L., Ribeiro, R.: A survey of predictive modelling under imbalanced distributions (2015) 85

[79] He, H., Ma, Y.: Imbalanced learning: foundations, algorithms, and applications. (2013) 85

[80] Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association **102** (2007) 359 – 378 108

[81] Neumann, L., Zisserman, A., Vedaldi, A.: Relaxed softmax: Efficient confidence auto-calibration for safe pedestrian detection. (2018) 110

[82] Solovyev, R., Wang, W., Gabruseva, T.: Weighted boxes fusion: Ensembling boxes from different object detection models. Image and Vision Computing **107** (2021) 104117 110