The identification of garbage dumps in the rural areas of Cyprus through the application of convolutional neural networks to satellite imagery.

Andrew Wilkinson

Supervisor: Tony Knowles

June 2023

Word Count: 8782 via overleaf

Thesis submitted in partial fulfilment for the degree of Master of Science in Computer Science with Artificial Intelligence

Department of Computer Science

Contents

Acknowledgements iv

Executive Summary v

List of Figures vi

List of Tables viii

1 Introduction 1

1.1 Background and Motivations 1

1.2 Aims and Objectives 2

1.3 Approach 2

1.4 Results and Contributions 3

1.5 Outline 3

2 Theoretical Background 4

2.1 Motivations 4

2.2 Machine Learning 4

2.3 Deep Learning 5

2.4 Convolutional Neural Networks 5

2.5 Machine Learning and Satellite Imagery 6

2.6 Residual Networks 7

2.7 Pretrained Zoo Models 8

2.8 Siamese Networks 8

2.9 Hyperparameter Optimisation 9

2.10 Deep Learning Frameworks 9

2.11 Data Collection 10

2.12 Data Augmentation 11

3 Research Methodology **12**

3.1 Research Design 12

3.2 Data Collection 13

3.3 Data Augmentation 17

3.4 Machine Learning 18

3.4.1 Choosing the model 18

3.4.2 Tuning Hyperparameters 19

3.4.3 Modelling Individual Datasets 21

3.4.4 Modelling Pipelined Datasets 21

3.5 Statistics Gathered 22

3.6 Frameworks/Tools 25

3.7 Limitations 26

4 Results and Discussion **28**

4.1 Model Selection Results 28

4.2 Hyperparameter Tuning Results 29

4.3 ResNet-50 Results 30

4.3.1 Singly Augmented Datasets 30

4.3.2 Pipeline Augmented Datasets 31

4.3.3 Summary Tables 32

4.4 Running Predictions 34

5 Conclusion **35**

5.1 Limitations and Future Work 36

A Appendix **37**

A.1 Software Versions 37

A.2 Data Collection Web App 38

A.3 Data Augmentation Samples 42

A.4 Weka ARFF Samples 44

A.5 Weka GUI Samples 45

A.6 Weka Command Line Scripts 48

A.7 Weka Prediction Results 49

Acknowledgements

I would like to take this opportunity to express my gratitude to the following people for their support throughout my MSc research.

I would like to thank my supervisor Tony Knowles for his guidance and feedback, which helped shape the direction of my research and encouraged me to think critically about my findings.

I would also like to express my sincere gratitude to my wife Jeanette, who provided unwavering support and care during my illness, which made it possible for me to continue my research. Her encouragement and understanding were invaluable, and it would have been difficult to complete this dissertation without her help.

I would also like to thank Andreas Yiannaki for assistance in collecting verified garbage coordinates.

Executive Summary

In much of the developed world responsible garbage disposal is a challenging problem. In Cyprus, as elsewhere, illegal garbage disposal via ”fly-tipping” is a significant issue, especially in rural areas where few legal garbage disposal options exist. However, there is a lack of studies that attempt to measure the scale of this problem, and little resource available to address it. A method of automating the process of identifying garbage dumps would help counter this and provide information to the relevant authorities.

The aim of this study was to investigate the degree to which artificial intelligence techniques, together with satellite imagery, can be used to identify illegal garbage dumps in the rural areas of Cyprus. This involved collecting a novel dataset of images that could be categorised as either containing, or not containing, garbage. The collection of such datasets in sufficient raw quantities is time consuming and costly. Therefore a relatively modest baseline set of images was collected, then data augmentation techniques used to increase the size of this dataset to a point where useful machine learning could be applied.

From this set of images an artificial neural network was trained to recognise the presence or absence of garbage in new images. A type of neural network especially suited to this task known as ”convolutional neural networks” was used. The efficacy of the resulting model was evaluated using an independently collected dataset of test images.

The result was a ”deep learning” model that could correctly identify images containing garbage in approximately 90% of cases. It is envisaged that this model could form the basis of a future system that could systematically analyse the entire landscape of Cyprus to build a ”garbage” map of the island.

This study has no legal, social, ethical, professional, or commercial issues.

List of Figures

1 ResNet Architecture 7

2 Big garbage dump 14

3 Medium garbage dump 15

4 Small garbage dump 15

5 Augmentation via clipping 16

6 Web App - Index Page 38

7 Web App - Create New Garbage Record 39

8 Web App - List Garbage Records 40

9 Web App - Database Schema 41

10 Sample Augmentated Images 42

11 Sample Augmentated Images (cont) 43

12 Sample Weka Training ARFF 44

13 Sample Weka Prediction ARFF 44

14 Weka - upload a dataset 45

15 Weka - Associate an Image Iterator with a dataset 46

16 Weka - configure hyperparameters 47

17 Weka - view results 47

18 Sample Weka Training Script 48

19 Sample Weka Prediction Script 48

20 Weka Prediction Results - pipeline 3 model 49

List of Tables

1 Pipelined Models 22

2 Model Evaluation 28

3 Single Augmentation Evaluation 30

4 Pipelined Evaluation 31

5 mini-batch size = 1 33

6 mini-batch size = 8 33

7 mini-batch size = 32 33

8 mini-batch size = 64 33

9 Software Versions 37

1 Introduction

1.1 Background and Motivations

Illegal waste dumping is a social and environmental issue throughout the developed and developing world [1]. The consequences include environmental pollution, health hazards, and negative impacts on local ecosystems and aesthetics. The island of Cyprus has particular issues with waste disposal in rural areas [2], where little formal recycling exists. As a result, citizens tend to make their own impromptu garbage dumps. Assessing the scale of this issue is difficult [3]. However, if there was a way of identifying how many such garbage dumps have accumulated, together with their location, then this could be used to inform and exert pressure on local, regional and national government bodies to address the issue more proactively [4].

Performing this sort of task by hand would be laborious and error prone. If, for example, the land was divided up into $200m^2$ patches, then even for a relatively modest sized country such as Cyprus covering around $9250km^2$ this would mean at least 231,250 such image patches to be analysed. If a human could retrieve an image, visually analyse it, and store the classification result as an atomic operation in around one minute, then to classify all image patches would require at least 3,850 person hours.

Some previous work has employed AI techniques to automatically detect the presence of garbage dumps, such as [5] and [6]. These projects have focused on large land-fills occupying an area of several square kilometers. Few such projects exist focusing on small scale sites covering just tens or hundreds of square metres. However, it is such small areas that Cyprus garbage dumps tend to occupy. With the existence of public access high resolution remote sensing imagery, such as satellite [7] [8], it is possible to focus on techniques to recognise these smaller scale features. This study utilised a type of Artificial Neural Network (ANN) shown to be effective in the classification of satellite imagery [9] and applied it to these small scale features.

1.2 Aims and Objectives

The aim of this study was to develop and evaluate an AI based approach for identifying garbage heaps in Cyprus. This involved the collection and labeling of satellite imagery. However, collection of such image sets is a laborious task and it is unfeasible to collect such ”raw” data in sufficient quantities to perform useful machine learning [10]. To counter this data augmentation techniques such as sharpening, cropping, rotating, and flipping were used to significantly increase the dataset sizes.

The questions it sought answers to are:

- **[RQ 1]** What are the most effective machine learning algorithms and techniques for developing a model that can accurately identify and classify small scale garbage dumps in the rural areas of Cyprus from satellite imagery?
- **[RQ 2]** To what extent can data augmentation techniques improve a machine learning model for such small scale satellite image classification?

1.3 Approach

To provide answers to the question above the following steps were completed. Data was collected in the form of satellite images of known garbage locations in Cyprus, forming a baseline set of images. This dataset was expanded through various data augmentation techniques [11]: sharpening, rotating, cropping, and flipping. A suitable machine learning model was selected for the study by training and evaluating various candidates against the baseline. The chosen model was then trained on the augmented datasets with care given to the tuning of hyperparameters [12] so that the performance of the model was optimised. Finally, the accuracy of the model was evaluated by applying various validation methods and statistics in order to arrive at conclusions with regards the efficacy of the model.

The study focused on a type of deep learning model referred to as convolutional neural networks (CNNs). These work by applying various filters to the images which learn to pick out low level then progressively higher level features from the images [13], culminating in, for this study, the identification of garbage. Dimensionality of the input data is reduced to a manageable level by use of such techniques as pooling [14]. CNNs have proven effective at image recognition, and have been applied successfully to the area of remote sensing [15].

1.4 Results and Contributions

The results and outputs of the study are manyfold:

- A novel baseline set of satellite image patches labelled as containing/not containing garbage, which might find use outside of this study.
- Several augmented datasets of similarly labelled images which may also find wider use within the machine learning community.
- A set of CNN models which provide an accurate basis for garbage heap identification with the potential to inform waste management policies in Cyprus.

1.5 Outline

Section 2 develops a theoretical framework to the study, mostly through a review of pertinent literature. Section 3 establishes the research methodology used to answer the research questions. This is followed by a discussion of the results in section 4 and finally a conclusion and recommendations for future work in section 5.

2 Theoretical Background

2.1 Motivations

Webb et al., 2006 [3] provide a comprehensive study into the incidence of fly-tipping. They describe in detail the who, how, where, and why of such practices and their environmental and social impact. One limitation of the study is that it is primarily UK based. However, it seems probable that it mirrors the issues experienced in Cyprus. One issue highlighted is the difficulty authorities have in mapping the location of fly-tipping sites - a main driver behind this research and one of the main objectives it seeks to address by producing models which can recognize garbage dumps from satellite images.

2.2 Machine Learning

Standard machine learning (ML) techniques such as Support Vector Machines (SVM) and Random Forest (RF) provide an alternative paradigm to traditional feature-based image recognition, and have been successfully applied to many image recognition tasks. Sheykhmousa et al., 2020 [16] describes a meta-study of 251 SVM and RF projects. It aimed to develop a comparative analysis of the two methods, but much of the data is qualitative obtained through a case study approach. Any quantitative data that was obtained does not seem to be directly comparable as the types of project that the SVM and RF were applied to are somewhat different. Nevertheless, the paper highlights that such methods are still being utilised in some domains.

Useful as these approaches are in some limited contexts, they have limitations when images are complex and large scale. For these deep learning approaches have been found to be more useful.

2.3 Deep Learning

The original deep learning research into image recognition was undertaken thirty years ago when LeCun used the Multilayer Perceptron with back-propagation to recognize handwritten digits [17]. He describes back-propagation as an iterative algorithm used to adjust the weights of the connections between nodes to minimize the difference between the predicted output and the actual output. It forms the basis by which ANNs are trained on input data. Since then deep learning has become the dominant method for all image recognition tasks, including those involving remote sensing sources, such as satellite imagery. One of the main reasons for this is that an ANN can model complex non-linear relationships, as described by LeCun in his later 2015 paper [18].

Standard ANNs can be effective for classifying small image patches, but for complex large satellite images this is an impractical method. Each pixel of an image is a separate input to the network. Once multiple layers are added to the network then the number of parameters in the training process grows exponentially such that the training time and memory requirements become prohibitively high, as Aggarwal details in his influential text book [19]. To address this a type of ANN called the Convolutional Neural Network (CNN) has become very popular.

2.4 Convolutional Neural Networks

The first CNN was proposed by Fukushima in the foundational 1980 paper [20] in which the neocognitron is described as an ”improved neural network”. This is essentially a hierarchy of networks composed of layers of nodes with ”variable connections between adjoining layers”. LeCun took this idea further in 1998 [21] in which back-propagation and gradient descent were added and he showed how these strategies could solve pattern recognition involving high dimensional images better than existing hand-designed algorithms. This lay the groundwork for future CNN algorithms and their various variations such as LeNet, AlexNet, Google LeNet, and ResNet. [22]. Much of the earlier work focused on recognition of small scale images,

particularly that of handwritten characters [23]. The next section shows how this was adapted for larger scale satellite imagery.

2.5 Machine Learning and Satellite Imagery

Satellite imagery offers particular challenges to machine learning. Forkuor et al., 2018 [24] discus these at depth in a paper that describes land-use and land-cover mapping in Burkino Faso. In addition to being much bigger, satellite images are multi-spectral. They describe the Sentinel-2 satellites from the Copernicus programme as providing images that encompass 13 spectral bands; in addition to the usual RGB channels, data is provided as channels in the near infra-red. This increases the options for analysing the data, allowing inferences to be made about differences in thermal characteristics, which could be useful for certain types of garbage dumps, especially if they contain biological matter. However, the more channels used to train a model then the more parameters must be handled, increasing the learning time. Consequently, it was decided for this study to use three channel RGB images.

Vaishnnave et al., 2019 [25] provide a survey of various methods of classifying satellite datasets. The methods covered include popular CNN algorithms such as AlexNet, ResNet-50, GoogleNet, and CaffeNet. Of particular interest is a summary of the accuracy achieved for each approach, with the CNN methods performing best with quoted accuracy of 93% to 99%. It should be noted that these figures are drawn from different studies with different datasets and different aims, so are not really comparable; nevertheless they do provide some support of the approach taken in this study.

A particularly relevant study is that of Rajkumar et al., 2022 in which a variety of ML algorithms are applied to landfills taken from WorldView and GeoEye satellite missions [5]. A modest dataset of 245 images was collected, with each image being 512x512 pixels. No indication is given of the area covered, however it is clear that the landfills are significantly bigger than the ones addressed by this study. The results showed “considerable” accuracy with between 76%

and 83%. This is well below the reported accuracy figures of [25]; perhaps the difference being due to the smaller dataset. The paper claims that models trained on high-definition imagery generalised well to detecting low resolution, although no evidence is given for this. The paper is particularly interesting because: it addresses the same class of images as this study, it gives some indication of useful image ”patch” size, uses a similar dataset size, suggests some possible CNN algorithms, and gives a baseline for what accuracy might be expected for a modest sized dataset.

2.6 Residual Networks

One subset of CNN that has been particularly successful in the area of remote sensing is that of Residual Networks (ResNets). Introduced by He et al. in a 2016 paper [26], this influential architecture addresses the vanishing gradient problem. This problem occurs in deep networks when the value changes back-propagated become so small that no meaningful weight updates (and, therefore, learning) occurs. The paper describes an architecture where shortcut connections can be made between layers, allowing nodes to be treated as blocks during the learning process (see figure 1 from the paper). This helps better move the gradients through very deep networks. The paper describes networks of over 1000 layers being able to be trained in this way.

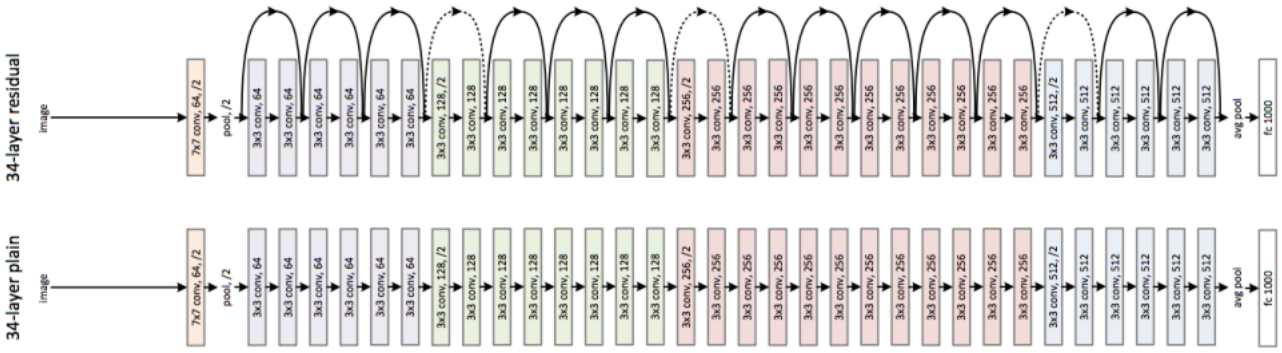


Figure 1: ResNet Architecture

2.7 Pretrained Zoo Models

Pretrained models are specific implementations of ML algorithms that have been trained on large amounts of data. They can be used to ”shortcut” the training time of similar problems and result in better models. The paper by Yosinski et al., 2014 [27] details early work that showed how even when the training data is significantly different from that of the target problem, that models designed around them perform better than those that are not pretrained. It is not fully established why this is, but it might have something to do with the general structures that make up typical objects in 2D images being learned, such as edges, irrespective of what the target image is. A limitation of the study was that it was performed against a narrow set of network structures (CNNs) and a single (if substantial) data source - ImageNet [28]. How well the results would generalise to other network and data types is unclear. However, as both CNNs and the ImageNet dataset are pertinent to this study, the paper remains a valuable input.

2.8 Siamese Networks

An interesting alternative/complement to CNN is that of Siamese networks, as described by Chicco (2021) [29]. This paper describes a siamese neural network as an ensemble of two ANNs which work in parallel, comparing the two outputs for similarity, usually via a cosine distance calculation [30]. This has been found to deliver good results even with quite modest sized datasets. The paper provides an overview of areas to which Siamese neural networks have been applied, such as audio and signal processing. It also describes their use in image analysis, particularly to signature and fingerprint recognition. Brief mention is given of their application to remote sensing projects - analysis of satellite images. However, the paper does not explore the potential limitations of such ”twin” networks or provide practical guidance in their implementation.

Whilst an interesting alternative/addition to basic CNN approaches to image classification, the additional complexity of the siamese approach would have been impractical given the

resources and time constraints of the Cyprus garbage study, so was not taken further.

2.9 Hyperparameter Optimisation

Aggarwal (2018) [31] describes hyperparameters as neural network parameters external to those trained by the model. They encompass such aspects as number of network layers, dropout, early stopping, and various normalisation methods as described in this 2012 Bengio treatise [32]. Selecting the most appropriate of these parameters can have great effect on the performance of the trained model.

However, there are often many such parameters and the values they might take also many. To choose appropriate parameters one of two approaches tend to be adopted. A systematic method might be used which effectively performs a search through the search space of the possible hyperparameter combinations, as described by Andonie et al., 2020 [12]. However, for a large network this might require a large amount of time and processing power because the model must be at least partially trained each time to enable comparisons to be made. Alternatively, a more ad-hoc experimental approach might be taken based on judgement and experience. Although much less rigorous than the systematic approach, this requires less time and is most commonly used. The various constraints of this study suggested the latter approach the more pragmatic.

2.10 Deep Learning Frameworks

Parvat et al., 2017 [33] provides a brief summary of popular deep learning frameworks and their respective capabilities. It does not explicitly define what a deep learning framework is, but from the capabilities it reviews one can infer it to be a set of tools and components that aid in the development, training and validation of ANNs. They typically provide a GUI and/or command-line interface, algorithm implementations, pretrained models and an API with which

to construct the models, set hyperparameters, pass data and receive the results. The frameworks covered by the paper are Theano, DeepLearning4J, Caffe, Torch and Tensorflow. The programming interfaces for these are a mix of Python, C/C++ and Java. The paper suggests that it evaluates the frameworks, but it mostly provides a capabilities comparison. The conclusion is that they mostly supply similar features, and much therefore depends on personal choice of (for example) Python versus Java versus C/C++.

Of the frameworks referred to above, the Java-based deep learning framework DeepLearning4j was of particular interest. It provides implementations of a large a set of neural network models, algorithms, and utilities together with extensive hyperparameter tuning capabilities [34]. It has also been integrated with the Weka machine learning workbench [35]. This allows a rapid prototyping approach to be taken by leveraging the easy to use GUI to upload datasets, select machine learning models, tune hyperparameters, run the models and obtain visualisations of the results. Sample screenshots of the Weka GUI can be found in appendix A.5. Re-tuning and re-running models based on the results from earlier iterations is trivial [36]. This made Weka/DeepLearning4j the obvious choice for the experimental nature of this study.

The next section moves away from machine learning models and looks at actual sources of the data and their collection and preprocessing.

2.11 Data Collection

There are several potential sources of high resolution satellite data. The US geological survey provides Earth Explorer - a user interface into the Landsat satellite system [37]. Sentinel data is available from the ESA open access hub [38]. Forkuor (2018) [24] describes a study of these two sources for land use and land cover, and applied three ML algorithms to compare their relative performance. Of the two, the Sentinel data proved slightly more accurate than the Landsat, although the algorithms were not ANN based, which tempers the applicability of the

results for this study.

An additional potential source of images is the Google Earth engine [39]. This provides both a browser and application front-end into a mix of Landsat and Sentinel imagery, which are freely available for third-party use. Zhao et al., 2021 [8] summarises the Google Earth engine and categorises the types of use to which the engine has been applied, including land classification.

2.12 Data Augmentation

Obtaining relevant labelled satellite imagery for most ML is difficult and time consuming [10]. It can be especially challenging to source images in sufficient numbers to obtain meaningful results. One way to address this is to use image augmentation techniques to enlarge the base set of images. Shorten et al., 2019 [11] offers a comprehensive overview of the various approaches, including geometric and photometric. Geometric techniques focus on changing an image by rotating, cropping, and scaling. Photometric focuses on changing an image by altering the contrast, sharpening, and brightness.

These two approaches are studied in more detail by Taylor et al. (2018) [40] in which the results of performing ML experiments on both are described. The DeepLearning4J framework [36] and CNNs are used on a dataset enlarged using a variety of techniques and a comparison made. The study does not record the results of applying all techniques together at once, but does suggest that geometric cropping provides the biggest improvement from a baseline of 48.13% to 61.95%.

3 Research Methodology

3.1 Research Design

In order to achieve the aims and objectives introduced in section 1.2 the following approach was adopted:

- To collect a set of satellite images from a publicly available source.
- To appropriately label each image as either ”garbage” or ”not garbage”.
- To apply several promising supervised deep learning approaches to this ”baseline” set of images and to choose the best performing for the remainder of the study.
- To evaluate the effects of varying various key hyperparameters against the baseline set and selected model to determine a near optimal model configuration.
- To apply various data augmentation techniques to the baseline dataset in order to derive enlarged datasets.
- To evaluate each enlarged dataset against the best performing machine learning model and make appropriate performance comparisons.

This follows the classical machine learning approach as described in [41], against a dataset novel both in terms of geographical location and in the size of features of interest. This represents a quantitative experimental approach involving the systematic analysis of data to produce probabilistic numeric outputs. This is the best approach to address the research questions under study as it represents a rigorous testing and evaluation of the various models. It is a method that can systematically vary different aspects of the machine learning process, such as the size and type of dataset, the types of algorithm used, the tuning of hyperparameters, and to evaluate how these all impact on the performance of the models to arrive at an optimal solution.

Other methods might have been chosen to address the research questions, such as classic image processing and pattern recognition techniques. However, these tend to be oriented around computer vision and real-time interaction and lack the ability to automatically learn and extract features at different levels of abstraction in the way deep learning algorithms can. The problem could also have been addressed using standard machine learning as described in section 2.2. However, these would be unlikely to be able to handle the complex and ambiguous image data as well as a deep learning approach or be able to match their classification accuracy.

The sections that follow describe the above process in detail together with limitations of the study.

3.2 Data Collection

The data collected for this study was RGB colour satellite images centred on latitude/longitude coordinates known to contain garbage dumps. These points were collected over the last year and are taken from a variety of rural locations. They provide a representative set of images of the sort of garbage dumps found throughout Cyprus, taken from a variety of land types, such as forest, scrubland, mountain, and arable. An alternative to 3 channel RGB images could have been used; multispectral images are available which provide more channels, up to 13 in the case of Sentinel 2 sensors [24]. These make use of wavelengths such as near-infrared (NIR) and thermal infrared (TIR) which can provide useful complementary information about landcover types. However, these tend to be most useful for detecting vegetation cover, and soil types. They also add significantly to the data volume to be processed and risked being beyond the computational resources available to the project.

For every image collected of a garbage dump site, a corresponding image of a nearby location that did not contain garbage was collected; this ensured a balanced dataset of both classes of interest (garbage, not-garbage). A balanced dataset is important to prevent biases in the

learned model, as discussed in [42]. A baseline set of 100 images was collected - 50 labelled as ”garbage” and 50 labelled as ”not garbage”.

Various possible sources of satellite imagery exist, as detailed in 2.11. It was decided for this study that the best source would be Google Earth [39]. This decision was made due to its ease of use and availability, and because it covers the areas of interest with little cloud cover which would otherwise obscure the features of interest. Google earth has also been used successfully on various other similar land classification projects, such as [6] and [43].

One aspect of image based ML of great importance is the image size of the samples being used to train the model [44]. Several popular CNN algorithms have been designed against specific images sizes. The original LeNet designed for the MNIST dataset was intended for 32x32x3 images [45]. AlexNet, ResNet, and VGG architectures assume image size of 224x224x3 [46] [26] [13]. The Inception (GoogleNet) algorithm is optimised for 299x299x3 images [47]. For this study the images collected are 200x200x3 files and upscaled as necessary for the model. These were a convenient size to work with and allowed the features of interest to be placed within a single image per occurrence.

Figure 2 3 and 4 illustrates sample garbage/not garbage image pairs, drawn from ”big”, ”medium” and ”small” samples.

Figure 2: Big garbage dump

Figure 3: Medium garbage dump

Figure 4: Small garbage dump

Once research began it became clear that the chosen image size/land area combination had a problem. Images were collected that centralised the garbage features of interest. Obviously randomly selected images beyond the training set would not possess this characteristic and would more likely occupy some non-central position within the image, or for larger garbage features be ”clipped” by the image boundary. Training on the data as was would probably perform poorly against a real-life set of images [40]. There were various ways to address this, such as expanding the dataset to include images towards and straddling the image boundary; however this would be quite laborious and a main aim of the study was to leverage a minimal dataset. Consequently an approach was taken to use geometric cropping to produce five

images per cropped image - one centered (the original image) and one for each quadrant. Figure 5 attempts to illustrate this process by using a different colour square for each quintile. Each of these five images were fed into the subsequent augmentation steps. To prevent production of ”garbage” image patches without any actual garbage features, this process was only applied to images containing garbage features of a certain minimum size.

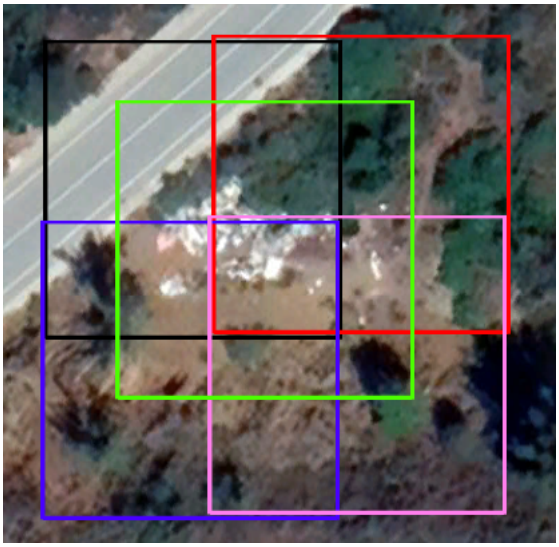


Figure 5: Augmentation via clipping

Each 200x200px image maps onto a land area of around 20mx20m area, with one pixel representing around 10x10cm. Other corresponding land classification studies tend to use each image pixel representing a larger land dimension, such as [6] using 150x150m patches. However, these studies are focusing on garbage sites covering much bigger areas. Most of the sites in Cyprus are smaller, so a smaller image patch can be used.

Various meta-data was collected pertaining to each image-pair and stored in a MySQL database, together with a file system reference to the location of the image. A simple PHP web-interface provided easy access into this database, allowing entries to be added, viewed and updated. The database schema is shown in the appendix together with screen shots of the web forms (appendix A.2).

3.3 Data Augmentation

Labelled satellite imagery for ML projects of this sort is expensive and laborious to collect in large enough quantities to produce accurate and robust models [48]. Studies have frequently relied on image augmentation to enlarge the dataset whilst preserving the labelling characteristics of the images. A major aim of this study was to explore how ML accuracy can be improved by such techniques, when applied to the Cyprus garbage satellite image set. Section 2.12 reviewed methods of increasing an image set set via image augmentation. [11] categorize such techniques as being one of: geometric, photometric or deep learning based. This study focused on the geometric and photometric techniques. Deep learning techniques could have been used as an alternative, or supplemental, approach to augmentation. However, deep learning augmentation requires significant computational resources when compared to conventional geometric and photometric procedures. Consequently this did not form part of the study.

The geometric augmentations most likely to offer improvement according to [40] were applied as follows (inclusive of the original image):

- cropping: each suitable garbage labelled image and their non-garbage counterpart was subject to the cropping method described in the previous section. This increased the baseline set x2.
- rotating: each image was rotated by 90, 180, and 270 degrees. This increased the baseline set x4.
- flipping: each image was flipped horizontally and vertically: this increased the baseline set x3.

The most promising photometric technique seemed likely to be sharpening. The nature of the typical garbage image is one of rectangular shapes, often slightly blurred due to the available resolution. Sharpening can improve these and enhance the edge-detection capabilities, as

outlined in [10]. The images were all subjected to a sharpening ”kernel” filter, increasing the baseline set from 100 to 200 (inclusive).

Appendix A.3 provides a few illustrations of the result of augmenting an image. How these various data augmentations were made to the baseline set and made available to the ML algorithms is described in the next section.

3.4 Machine Learning

Once a suitably augmented set(s) of data were obtained, the ML process was carried out as follows:

- choose model instance to go forward with
- tune hyperparameters
- run model against each augmented dataset
- run model against pipelined datasets - carefully selected combinations of augmented datasets
- evaluate the results

This iterative process of tuning hyperparameters, training on augmented datasets, combining them in pipelined datasets, and evaluating the results helped in improving the model’s performance and ability to handle novel scenarios.

3.4.1 Choosing the model

Only convolutional neural network (CNN) models were considered for the actual machine learning. When a neural network is applied to a 224x224x3 image, then there are 150,528

input values for each image. This is too high a dimensionality to train a neural network using fully connected layers given the modest computational resources available to the project [49]. CNNs use such techniques as pooling to reduce this dimensionality [50]. They also use filters to pick out features of interest in an image. These work by highlighting low level features such as edges which can be fed into further filters that can reveal higher level features, such as polygons, which can fed into yet more filters until (in this case) garbage dumps might be recognised [19].

There are many possible CNN models to choose from. It was decided to evaluate a range of those that had been successfully applied to image classification tasks, including several winners of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [51]. The process was to apply each candidate model to the baseline (unaugmented) and compare the relative performance of the model, using the metrics described in section 3.5. [5] and [25] suggest that Residual Network model might be most effective when applied to this dataset, which proved to be the case. The results section 4.1 summarises the outcome of this model selection process, including a list of the candidate models evaluated.

All the models evaluated were pretrained. These have already been trained against an image dataset, in this case the publicly available ImageNet dataset [28]. Whilst not consisting exclusively of satellite imagery (in fact only a small proportion is) it has been found that even models pretrained on non-satellite images have an advantage over those not pretrained at all [27].

3.4.2 Tuning Hyperparameters

There are two types of parameter that neural networks work with. The model parameters are those that are internal to the model and which are estimated by the algorithm as it works through the nodes in the network, calculating the weights and biase. There are also parameters external to the model which can be set manually - referred to as

hyperparameters [31]. Setting these appropriately can greatly improve performance of the model, reducing the execution time and resulting in better accuracy. This ”tuning” process is mostly experimental, guided by experience of understanding of what the various parameters are intended to accomplish, as reviewed in section 2.9. For the Cyprus garbage dataset, the following hyperparameters were chosen for further evaluation against ResNet-50 [32]:

- early stopping: terminate the training process if no improvement is found (potentially speeds up the training process)
- optimisation algorithm: methods of applying the loss (error) function to arrive at a set of optimal weights and biases (potentially improves the accuracy)
- L1 and L2 regularisation factors: makes adjustments to values of coefficients (potentially reduces overfitting)
- dropout: using various criteria to exclude some nodes of the network from the training process (again, potentially reduces overfitting)

These hyperparameters are not an exhaustive set of those available, but represent a typical collection of those that might be expected to improve the performance of the model if selected judiciously.

The parameters were tuned by selecting appropriate values for each and training and evaluating accuracy of the model in an iterative way. This was done for each parameter in isolation. An assumption was made that parameter settings would not be interconnected. This is probably a slightly naive assumption, but to assume otherwise would have resulted in an unmanageable set of possible parameter combinations.

3.4.3 Modelling Individual Datasets

The following datasets were modelled to establish the relative effectiveness of each data augmentation technique via the selected model, as determined in the results, section 4.1:

- baseline dataset of 100 samples
- geocropped dataset of 200 samples
- sharpened dataset of 200 samples
- flipped dataset of 300 samples
- rotated dataset of 400 samples

For each such dataset performance metrics were collected, as described in section 3.5. The aim of this was to establish a set of baselines for comparison of the various data augmentation techniques and for later evaluation of improvements (or otherwise) as they were pipelined together.

3.4.4 Modelling Pipelined Datasets

There are numerous ways of combining the individual datasets, and it is only practicable to model a subset of these. For this study, to give a reasonable variation of datasets the following pipelines were modelled:

Note that rotation operations multiply each sample by 3 (90°, 180°, 270°) and flipping operations multiply each by 2 (horizontal and vertically flipped) - in each case excluding the original sample (to avoid repeatedly including it in the dataset).

There are clearly many other ways the datasets could be combined together, for example by creating datasets without sharpening the images, or omitting/combining image augmentations

Table 1: Pipelined Models

Pipeline Label	Description	#Samples
pipeline_1	Add each individual dataset into total dataset	800
pipeline_2	For each sample, rotate the sample, flip the sample, add those 6 samples to total dataset. Then sharpen the sample and rotate and flip that sharpened sample, add those 6 samples to the total dataset	1200
pipeline_3	Perform as per pipeline_2 but also do the same with the cropped dataset, doubling the number of samples	2400
pipeline_4	Combine the datasets to their fullest, by making the rotation and flipping operations multiplicative rather than additive	4800

in different ways. However, the sets chosen provide a representative set of different sample sizes with which to compare the model performance, both as a means of assessing the impact of data augmentation on the results of the model and of how well a model can be trained to recognise garbage.

3.5 Statistics Gathered

The research undertaken represents a binary classification problem. The model produced classifies image patches as either containing garbage (the ”positive” outcome) or not (the ”negative” outcome). This implies that for any attempt to classify a particular image, when compared to the actual classification, there are four possibilities [52]:

1. the classification is ”contains garbage” and the image does actually contain garbage: a ”true positive” (TP)
2. the classification is ”contains garbage” but the image does not actually contain garbage: a ”false positive” (FP)
3. the classification is ”does not contain garbage” and the image does not actually contain any garbage: a ”true negative” (TN)

4. the classification is ”does not contain garbage” but the image does actually contain garbage: a ”false negative” (FN)

Most of the statistics used to assess the efficacy of a binary classification problem make use these four quantities. The ones used in the study, either directly or buried within another statistic are detailed below.

Accuracy: the percentage of correctly classified predictions with respect to the entire dataset. This gives a good quick indication of how well the model is performing, especially for balanced datasets such as we had here. See equation 1 for how it is calculated from the figures referred to previously.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

Precision: how well the model predicts positive results. This metric does not give a complete picture as it does not take into account false negatives, in this case predictions of image patches not containing garbage when they actually do. See equation 2.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

Recall: the fraction of true positive instances that are correctly identified by the model. This metric does not take account false positives, in this case the predictions of image patches identified as containing garbage which do not. See equation 3.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

These metrics give valuable insight into how well a model is performing in terms of predictive accuracy, but they do not provide a complete picture [53]. There are various useful ways in

which these statistics can be combined to produce a more complete picture.

F-Score: this balances precision and recall by calculating their harmonic mean. See equation 4.

$$\text{FScore} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precion} + \text{Recall}} \tag{4}$$

However, it can be seen that with the F-Score the true negative quantity is not taken into account (because neither precision nor recall do so). The implication is that images predicted to contain garbage, but actually do not, will not effect the stated accuracy of the model via the F-Score.

MCC A statistic which takes account of all parts of the confusion matrix is the Matthew Correlation Coefficient (MCC) [54]. This measures the correlation between the actual and predicted classes, and yields a value between -1 and 1. An MCC of 1 indicates perfect classification, 0 random performance, and -1 a wholly negative correlation. See equation 5. This takes equal account of all four parts of the confusion matrix, so it is one of the preferred statistics used on this study.

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

Various other metrics could have been used to those above. Some of these, such as root mean square error (RMSE), are more suitable to regression problems where the target variable is represented by a continuous value [52]. Others, such as Cohen's Kappa statistic, are most useful for classifications where the dataset is imbalanced [53]. As this study is neither a regression problem nor imbalanced, it was felt that those listed above represent a good set of measures applicable to the balanced binary classification nature of the study.

3.6 Frameworks/Tools

The research was conducted using the Weka machine learning toolkit (Waikato Environment for Knowledge Analysis) [35]. This is an extensible Java based toolkit that provides support for data preprocessing, implementations of many popular machine learning algorithms, visualisation tools, results statistic reporting, and much more.

For this research Weka was used in conjunction with DeepLearning4J [55], which provides Java implementations of many deep learning models, including convolutional neural networks. Various popular models are provided, such as AlexNet, LeNet, ResNet and Inception. One especially useful feature is that many of these come as ”zoo models” [56] pretrained on several publicly available image datasets, including ImageNet [28]. These pretrained models significantly shorten the training time required for new datasets [27].

Both Weka and DeepLearning4j provide programmatic API and command line interfaces. Weka also offers a user friendly GUI with which to supply datasets, invoke models, and visualise the results.

Several alternative frameworks could have been used in place of Weka/DeepLearning4j [33]. Keras is a Python programming framework for constructing neural networks in relatively few lines of code. Pytorch is another powerful Python programming framework that provides similar features to Keras. Either of these two frameworks could have been employed for the research, but it was felt that the Weka integrated GUI combined with the various pretrained models made it particularly suitable for the prototyping aspects of the project.

In addition to the frameworks employed for the core machine learning tasks, various other utilities and techniques were used for data collection and augmentation. Data augmentation was performed using a set of simple bash shell scripts. These in turn utilised both the ImageMagick image processing software [57] and the powerful ffmpeg multimedia software [58]. Various other systems could have been used, such as the built-in Keras

augmentation capabilities [59], but ImageMagick and ffmpeg were simple to use and performed all the functions required.

A simple web interface was used to aid in collection of the image data samples, written in the PHP programming language. This utilised a MySQL database for data persistence [60]. A Java program was written to turn the images that form a dataset into a file in the Weka ”arff” format [35] that can be processed by Weka. A sample file in this format is represented in appendix A.4.

Appendix A.1 lists each software package used and their versions.

3.7 Limitations

A CNN project requires large and diverse datasets for effective training [10]. This project has been based on a modest baseline set of 100 images. However, data augmentation techniques have boosted the dataset size up to 4800. Nevertheless, additional images added to the baseset would add to the predictive performance of the model.

The images were gathered at a resolution of 1 pixel representing 10x10cm of land coverage. The model might not perform as well against different resolution images. In addition, during the research the same land area was found to have different chromatic characteristics depending on what date satellite images were taken. An area for future exploration is to examine how the developed model performs against images that differ in this way. Possibly a model developed against a grey scale could be more robust against such variability. Converting the images to grey scale and rerunning the tests would be an interesting area for future study.

Many of the images were collected in mountainous wooded areas, which tend to predominate on the island. The effects of this were mitigated to an extent by using an independently collected test dataset from other areas of the island without these characteristics. Nevertheless a larger more diverse dataset would only improve the accuracy and robustness of the model.

It is also the case that wooded areas may obscure garbage dumps, reducing the effectiveness of the model in detecting these and boosting the ”false negative” numbers. It might be the case that expanding the search beyond the 3-channel RGB into, for example, the infrared, might improve this aspect of the model by detecting thermal differences.

4 Results and Discussion

This section evaluates the results of the various experiments to first choose the best performing model type, then optimise the hyperparameters and finally run the model against the various datasets, both unaugmented and augmented. At each stage statistics such as accuracy, F-Score and MCC were collected (see section 3.5) and used as the basis for comparison.

4.1 Model Selection Results

The results of evaluating the various models to determine a single candidate to move forward with, as described in section 3.4.1, are shown in table 2. This shows the accuracy, F-Score and MCC metrics for a run of each model on the 100 image baseline dataset. In all cases the ”Dl4J” flavours of model were used from Weka/Deeplearning4j.

Table 2: Model Evaluation

Model	Accuracy	F-Score	MCC
LeNet	53.00	0.53	0.06
VGG	N/A	N/A	N/A
AlexNet	50.00	0.45	0.00
KerasInceptionV3	47.00	0.47	-0.06
Xception	50.00	0.37	0.00
ResNet50	67.00	0.66	0.36

From this it can be seen that the ResNet-50 implementation gives the best performance against all measures [1]. This is consistent with the results of other satellite based land classification studies that had evaluated several CNN models, as highlighted in section 2.5. The ResNet-50 model was therefore selected for subsequent steps in the process. This has 48 convolutional layers, one MaxPool layer, and one average pool layer and a total of of around 23.5 million trainable parameters [26].

[1] The VGG implementations yielded arithmetic underflows, possibly due to the ”vanishing gradient” problem appearing for these models [61]

4.2 Hyperparameter Tuning Results

The following hyper-params were experimented with (as described in section 3.4.2):

- mini-batch size
- early stopping
- optimisation algorithm
- L1/L2 regularisation factors
- dropout

Of these, the only hyperparameter that had any noticeable impact on the results was the mini-batch size. As described in section 3.4.2 the purpose of this parameter is to split up the dataset into small batches which are fed into the gradient descent algorithm. It lies between stochastic gradient descent and batch descent, trying to find a reasonable balance between the two [62].

There are various possible reasons why other parameters might have so little impact on the outputs. It may be that the dataset is of insufficient size for the model to respond well to adjustments in the hyperparameters, or the ResNet-50 model itself might be architecturally so well regularized, especially as we are using the pretrained zoo model variant, that hyperparameter tuning has little effect. However, similar lack of impact was observed when modifying these hyperparameters for other models, such as KerasInceptionV3. It may also be that be that these features as implemented in the DeepLearning4J are not being propagated from the Weka API. In fact, examining the Weka/Deeplearning4j source code uncovered evidence to this effect [2] and further investigation suggests that the most recent version of Weka/Deeplearning4j is not using the most current version of the Deeplearning4j backend.

Because of this, all tuning experiments focused on the mini-batch parameter, as shown in tables 5, 6, 7, 8.

[2] This comment in the org.deeplearning4j.ui.module.train.TrainModule class seems relevant (the ”TODO” is from the source code, not this report!) ***”//TODO: Maybe L1/L2, dropout, updater-specific values etc”***

4.3 ResNet-50 Results

This section describes the results of the various experiments outlined in sections 3.4.3 and 3.4.4 run on the ResNet-50 model, pretrained on the ImageNet image database [28].

4.3.1 Singly Augmented Datasets

These are results of applying a single augmentation technique to the baseline dataset and adding the augmented dataset to the baseline dataset; the ResNet50 algorithm was applied to each of these datasets in turn. The trained model used a mini-batch size of 8 and 5x cross-fold validation as described in [63]. This mini-batch size was used because it had been shown to produce good accuracy results whilst cutting down the training time without causing memory issues for the hardware used. 5x cross-fold validation was used in preference to the common alternative of 10x because it cut down the training/validation time by nearly 50% whilst producing metrics quite similar to the larger validation approach.

Table 3: Single Augmentation Evaluation

Dataset	# samples	Accuracy	MCC	F-Score
Collected baseline	100	56.00	0.12	0.56
Baseline + Cropped	200	83.50	0.68	0.83
Baseline + Sharpened	200	74.00	0.05	0.74
Baseline + Flipped	300	90.33	0.82	0.90
Baseline + Rotated	400	95.25	0.91	0.95

From this it can be seen that each of the augmentation techniques have yielded significant accuracy improvements over just the baseline dataset. The bigger the augmented dataset the greater the improvement, which is broadly in line with expectations. The flipping and rotating augmentation techniques in particular gave results suggestive of very good performance, with accuracy levels 90 percent and above, with similarly good MCC and F-score metrics. Perhaps these results are too good, especially on such small datasets. This might be suggestive of overfitting, a topic which is discussed in the next sections in the context of the pipelined model runs.

4.3.2 Pipeline Augmented Datasets

This section summarises the results of the various pipelined experiments. Table 4 lists the performance measures collected as a result of running the experiments against the datasets described in section 3.4.4, using a mini-batch size of 8 and 5x cross-validation used previously.

Table 4: Pipelined Evaluation

Pipeline	# samples	Accuracy	MCC	F-Score
1	800	96.65	0.93	0.97
2	1200	98.25	0.97	0.98
3	2400	98.29	0.966	0.98
4	4800	99.81	0.99	0.99

This show that the accuracy improves as the dataset size increases, which was the expectation. However, the accuracy results are again very high in comparison to that reported in other similar land classification studies. For example [5] reported a maximum accuracy of 83%. The results are also better than competitors have achieved in the various ILSVRC competition [51]. It seemed possible, therefore, that some form of overfitting was occurring [64]. To investigate this further three more experiments were performed. One was to run the model against different mini-batch sizes to examine the impact of this. Large batch sizes have been found to sometimes result in loss of generalisation of models [65]. Another was to run the model and validate using a percentage split approach instead of using cross-fold. In theory, percentage-split might be more prone to overfitting as a smaller training set is used, but it was felt that it would give an additional basis for comparison. The third was to collect an additional dataset of garbage/not garbage images independent of the training set and to use these as a test dataset. The results of applying the model against this test dataset would provide further insight. This results of running these experiments are described in the next section.

4.3.3 Summary Tables

The tables in this section summarise the results of running the model against each pipeline set of data, as detailed in section 3.4.4 [3]. For each dataset the model was evaluated using each of the three validation methods: 5 times cross-fold, 70% split, and using the independent test dataset. In addition this exercise was replicated for four mini-batch sizes: 1, 8, 32 and 64. The mini-batch parameter had been previously shown to have most impact on the model of all the various hyperparameters, and this set was chosen to obtain a representative spread of results. All other hyperparameters were left at their default values.

The tables show the accuracy and MCC metrics for each permutation. The F-score metric was dropped for reasons of clarity in interpreting the tables, and because it takes into account less information than does the MCC statistic. An average was taken for each measure against the validation methods as a way of combining the results into a single value. The most accurate model for each mini-batch, as determined by their average, is highlighted in bold.

These tables illustrate that in all cases, even for the unaugmented baseline, that the trained models perform significantly better than chance in their ability to correctly classify an image as containing/not containing garbage. In many cases it does so with greater than 80% accuracy.

The experiments show that the model tends to perform worse against the independent test dataset than when validated against the training data using the cross-fold and percentage-split methods. There are various possibilities for this. The model may be overfitting to the training set - which is why a separate independent dataset was added. The test dataset is fairly small and unaugmented, which might be resulting in less representative results. However, even with these caveats the models show promising predictive capabilities.

The model which gives the best results is the 2400 instance dataset combined with a 64 sample size mini-batch. However, the combined results of the 32 size mini-batch gives the

[3] The 4800 size pipeline dataset was dropped because the associated training time proved prohibitive. It is felt that the performance of the lesser augmented sets was such that this had little impact on the overall results.

Table 5: mini-batch size = 1

Dataset	#samples	5x cross-fold		70% split		test dataset		Averaged	
		Acc	MCC	Acc	MCC	Acc	MCC	Acc	MCC
baseline	100	55.00	0.10	60.00	0.16	67.00	0.44	60.67	0.23
pipeline 1	800	75.10	0.52	81.70	0.63	64.00	0.30	73.60	0.48
pipeline 2	1200	79.20	0.58	74.20	0.51	37.00	0.04	63.47	0.18
pipeline 3	2400	73.20	0.46	79.58	0.59	99.40	0.99	**84.06**	**0.68**

Table 6: mini-batch size = 8

Dataset	#samples	5x cross-fold		70% split		test dataset		Averaged	
		Acc	MCC	Acc	MCC	Acc	MCC	Acc	MCC
baseline	100	56.00	0.20	46.70	0.01	74.00	0.49	58.90	0.23
pipeline 1	800	93.75	0.88	95.83	0.92	77.00	0.57	88.86	**0.79**
pipeline 2	1200	98.25	0.97	78.21	0.58	79.00	0.60	85.15	0.72
pipeline 3	2400	98.29	0.97	95.65	0.91	73.00	0.49	**88.98**	**0.79**

Table 7: mini-batch size = 32

Dataset	# samples	5x cross-fold		70% split		test dataset		Averaged	
		Acc	MCC	Acc	MCC	Acc	MCC	Acc	MCC
pipeline 1	800	98.00	0.96	96.70	0.93	79.00	0.60	90.23	0.83
pipeline 2	1200	99.50	0.99	98.60	0.97	75.00	0.49	91.03	0.82
pipeline 3	2400	99.30	0.98	99.30	0.99	76.0	0.54	**91.53**	**0.84**

Table 8: mini-batch size = 64

Dataset	# samples	5x cross-fold		70% split		test dataset		Averaged	
		Acc	MCC	Acc	MCC	Acc	MCC	Acc	MCC
pipeline 1	800	96.12	0.92	48.00	-0.01	78.00	0.58	74.04	0.75
pipeline 2	1200	98.75	0.98	71.67	0.50	78.00	0.53	82.81	0.67
pipeline 3	2400	99.12	0.98	99.44	0.99	80.00	0.58	**92.85**	**0.85**

most consistent results overs all datasets, with a combined average accuracy more than 90% for each dataset, and is the one that the authors would suggest adopting for any future extensions of the study or for making predictions.

4.4 Running Predictions

The previous sections describe the results of training models and evaluating them using various validation techniques and against a set of test data. Once a model has been trained it can be used to make predictions against new data. This is achieved by creating a Weka ".arff" prediction file - which is identical to a Weka training file, but with the label replaced by a "?" to signify that the outcome is not yet known. An example of such a file is shown in appendix A.4. This refers to 20 images that we want to make predictions against. The results of using the model trained against the baseline are shown in appendix A.7.

5 Conclusion

This study sought to answer two research questions, as detailed in section 1:

- **[RQ 1]** What are the most effective machine learning algorithms and techniques for developing a model that can accurately identify and classify small scale garbage dumps in the rural areas of Cyprus from satellite imagery?
- **[RQ 2]** To what extent can data augmentation techniques improve a machine learning model for such small scale satellite image classification?

To answer these questions a baseline dataset of 100 training satellite image patches was collected using Google Earth [39] and labelled as either ”garbage” or ”not garbage”. Several popular convolutional neural network implementations were trained and evaluated on this baseline set, with the ResNet-50 model showing the most promise in terms of accuracy.

The baseline dataset was then augmented by applying various combinations of sharpening, rotating, flipping, and cropping. This resulted in ”pipelines” of 800, 1200, 2400, and 4800 sized image sets. The ResNet-50 model, pretrained on the ImageNet dataset [46], was trained on each pipelined collection of images and the performance of each model evaluated with respect to the others. In addition, various combinations of mini-batch size was used to improve generalisation of the models and reduce overfitting [66]. Various validation techniques were used to evaluate the accuracy and generalisation capabilities of the models, including cross-fold validation, employing a holdout dataset, and testing against a separate independently collected set of images.

The experiments showed that ResNet-50 provides an effective machine learning model in the detection of garbage dumps ([RQ 1]). With just the small baseset of 100 images, a model could be trained that correctly predicts around 70% of novel images. When the size of this baseline sets was expanded using data argumentation ([RQ 2]) then the predictive capabilities of the model increased dramatically, correctly classifying images in more than 90% of cases.

5.1 Limitations and Future Work

One limitation of the project is that the images that the models were trained on were mostly gathered from a specific landscape type - wooded mountainous areas (which cover much of the island). This baseline dataset also tended to lack such features as buildings, vehicles, etc. The independent test dataset does include such elements, together with more diverse terrain types such as arable. Despite this, the models still perform well against this more varied dataset, obtaining around 80% accuracy. Nevertheless, the predictive capabilities of the model would be expected to improve further if the training dataset was expanded both in size and land type.

The data augmentation techniques have proven to be very successful, boosting accuracy from around 70% to over 90%. However, the techniques are applied quite systematically - each one is administered to each image in the same way. As a consequence, there is some evidence of overfitting creeping into the results. This could perhaps be improved by adding variance into the process. For example randomly applying the sharpness process, or varying how rotation and cropping are applied. This would be expected to improve the generalisation of the model.

The model utilises the RGB channels of the image sets. Visually reviewing the images which have garbage shows that the garbage features tend to be surprisingly regular, especially in terms of their light colouring. It might therefore be the case that using pure greyscale images improves the model as well as significantly reducing training time. This could be explored in future work.

A Appendix

A.1 Software Versions

The following lists all the software packages used in the project, together with versions.

Software Package	Version
Java	11
Weka	3.9.6
Weka deepLearning4J	1.7.2
ImageMagick	7.1.1-7
apache	2.0
mysql	7.4.3-4
php	7.4

Table 9: Software Versions

A.2 Data Collection Web App

To aid in data collection a simple web-app was developed that allows the user to record key features about an image - such as latitude/longitude position, verification status, etc. and store these in a MySQL database, as well as allowing the image to be uploaded. Whilst not forming part of the main research strand as such, it did make the data collection easier to manage.

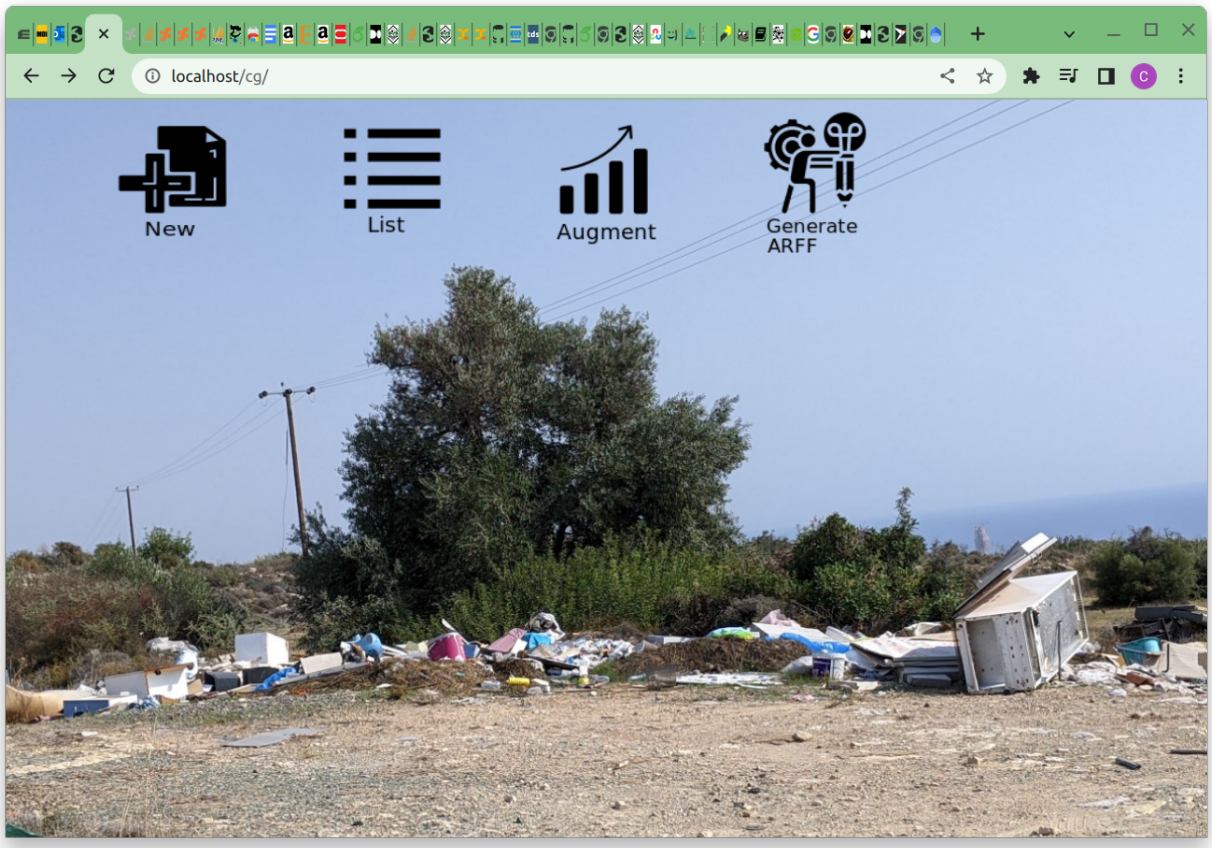


Figure 6: Web App - Index Page

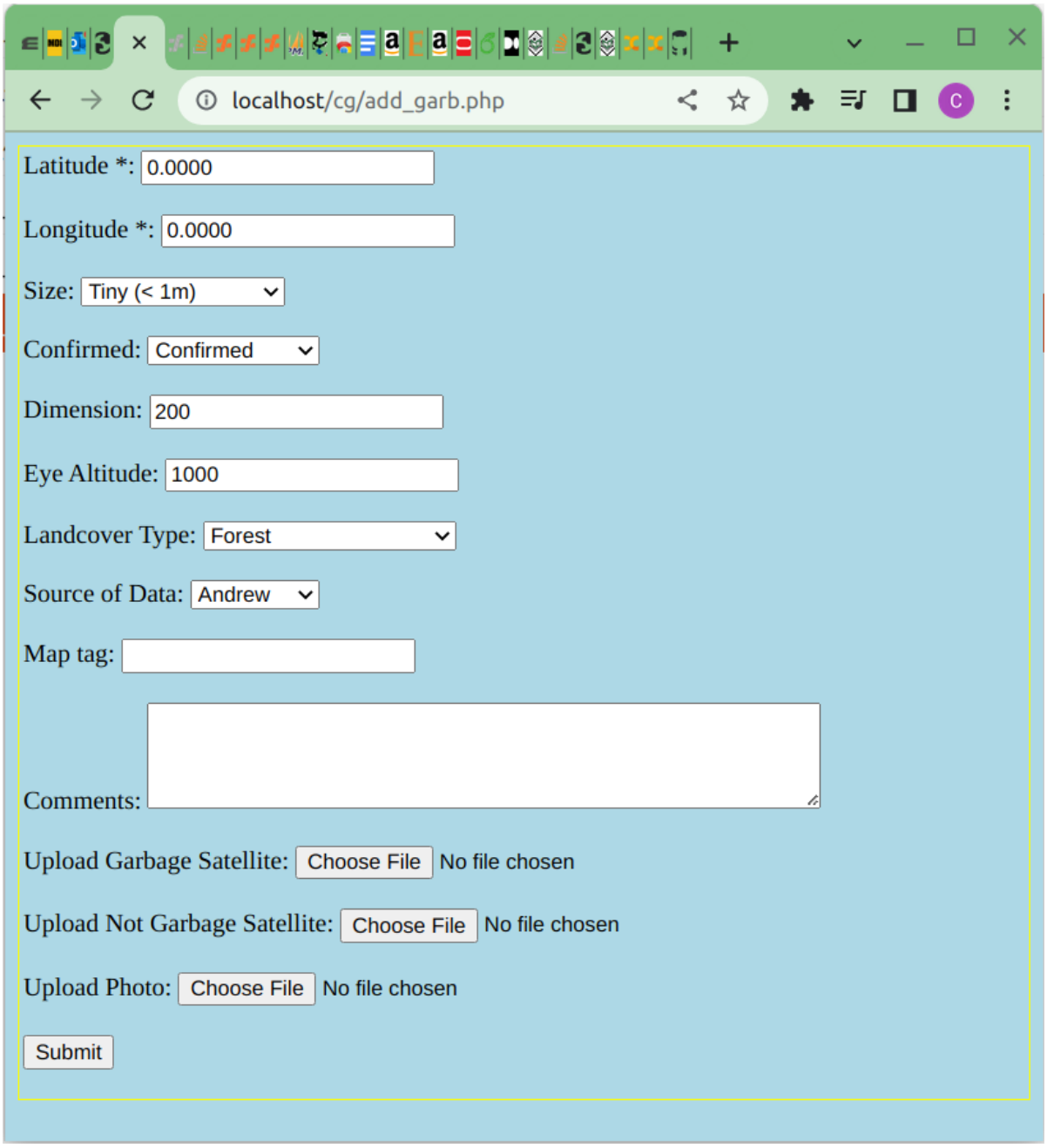
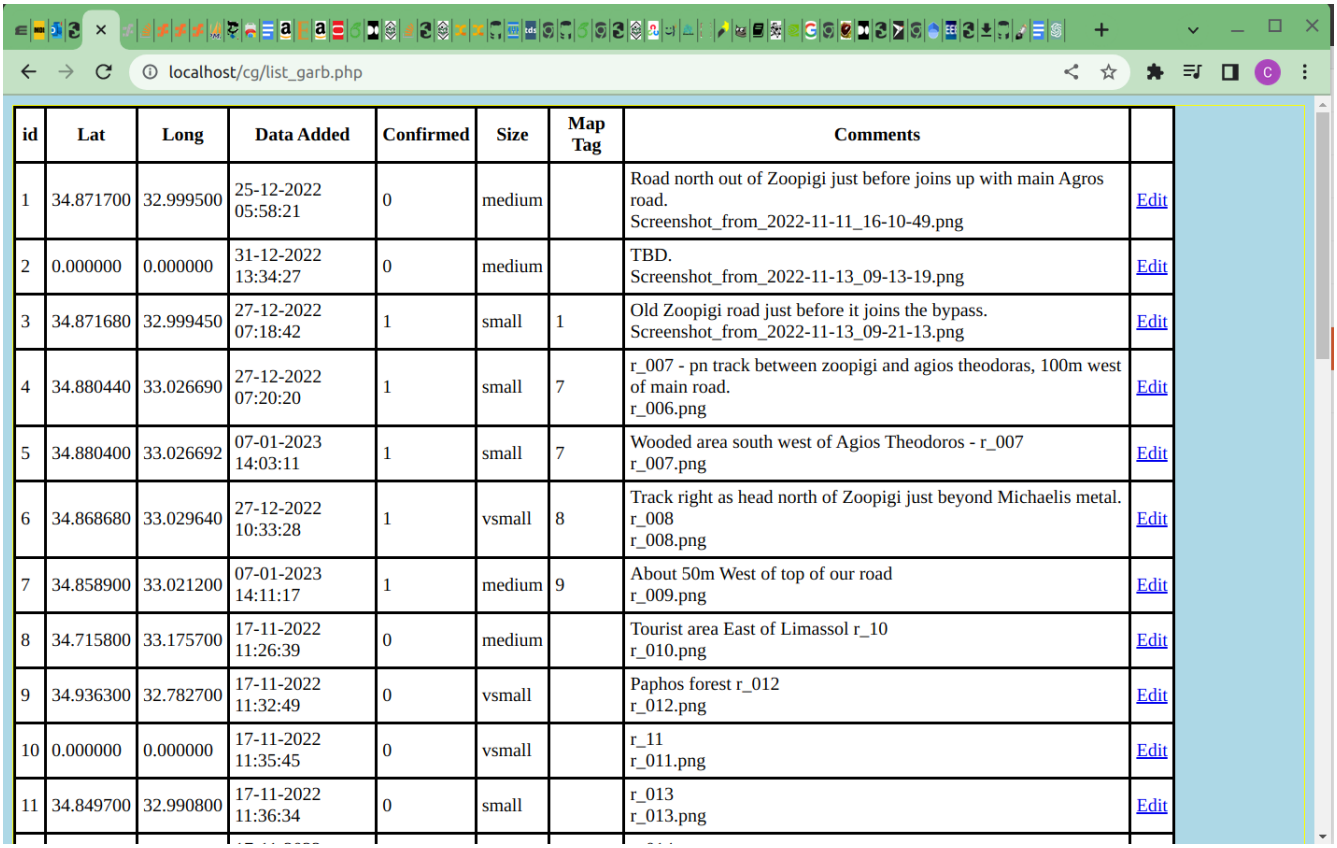


Figure 7: Web App - Create New Garbage Record



id	Lat	Long	Data Added	Confirmed	Size	Map Tag	Comments	
1	34.871700	32.999500	25-12-2022 05:58:21	0	medium		Road north out of Zoopigi just before joins up with main Agros road. Screenshot_from_2022-11-11_16-10-49.png	Edit
2	0.000000	0.000000	31-12-2022 13:34:27	0	medium		TBD. Screenshot_from_2022-11-13_09-13-19.png	Edit
3	34.871680	32.999450	27-12-2022 07:18:42	1	small	1	Old Zoopigi road just before it joins the bypass. Screenshot_from_2022-11-13_09-21-13.png	Edit
4	34.880440	33.026690	27-12-2022 07:20:20	1	small	7	r_007 - pn track between zoopigi and agios theodoras, 100m west of main road. r_006.png	Edit
5	34.880400	33.026692	07-01-2023 14:03:11	1	small	7	Wooded area south west of Agios Theodoros - r_007 r_007.png	Edit
6	34.868680	33.029640	27-12-2022 10:33:28	1	vsmall	8	Track right as head north of Zoopigi just beyond Michaelis metal. r_008 r_008.png	Edit
7	34.858900	33.021200	07-01-2023 14:11:17	1	medium	9	About 50m West of top of our road r_009.png	Edit
8	34.715800	33.175700	17-11-2022 11:26:39	0	medium		Tourist area East of Limassol r_10 r_010.png	Edit
9	34.936300	32.782700	17-11-2022 11:32:49	0	vsmall		Paphos forest r_012 r_012.png	Edit
10	0.000000	0.000000	17-11-2022 11:35:45	0	vsmall		r_11 r_011.png	Edit
11	34.849700	32.990800	17-11-2022 11:36:34	0	small		r_013 r_013.png	Edit

Figure 8: Web App - List Garbage Records

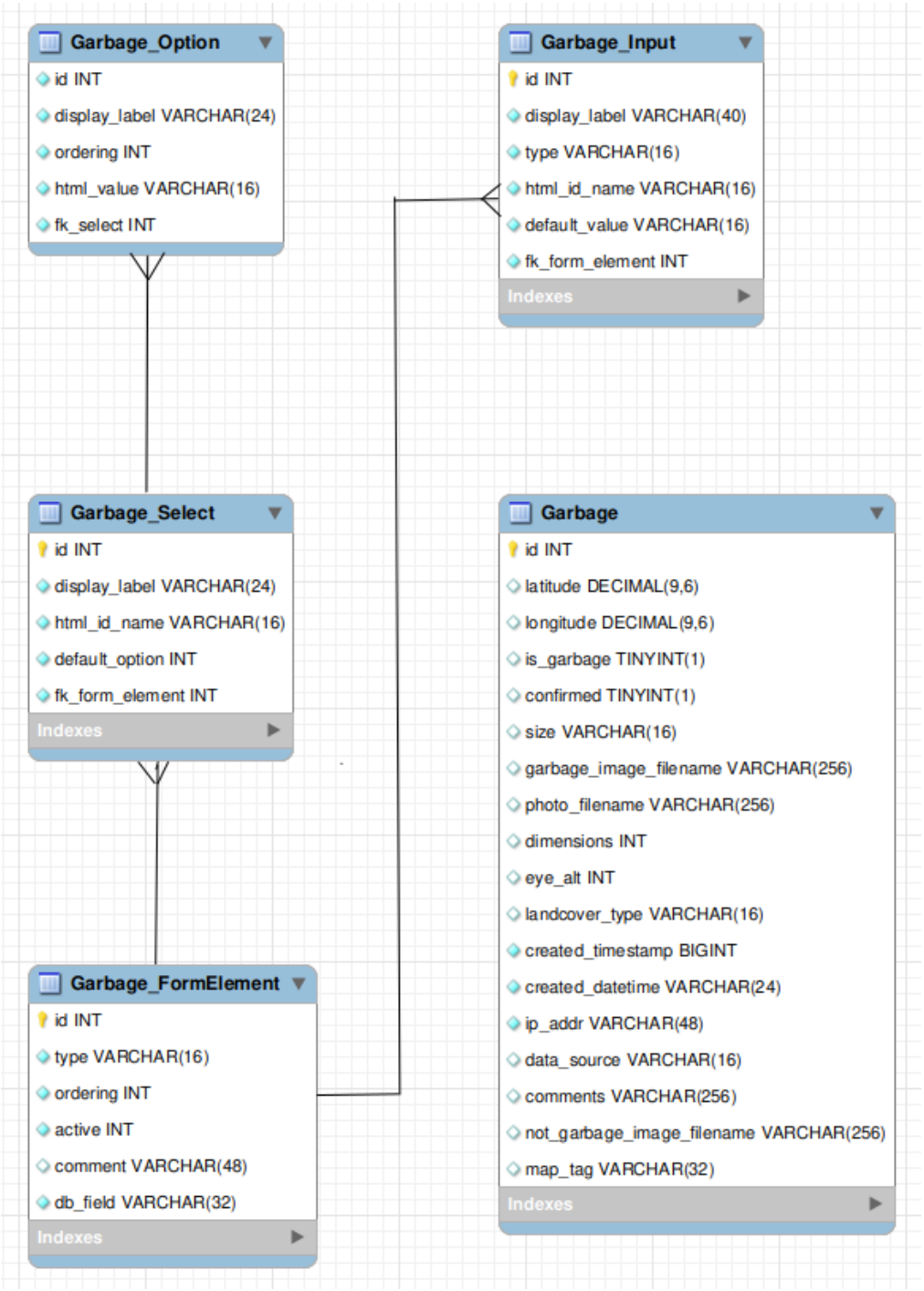


Figure 9: Web App - Database Schema

A.3 Data Augmentation Samples

(a) Unaugmented original

(b) Sharpened

(c) Horizontally Flipped

(d) Vertically Flipped

Figure 10: Sample Augmentated Images

(a) Rotated 90°

(b) Rotated 180°

Figure 11: Sample Augmentated Images (cont)

A.4 Weka ARFF Samples

The following contains snippets from the Weka ”arff” files used as input into the training and prediction processes.

```
@relation Rubbish

@attribute filename string
@attribute class {1,0}

@data
10_r_011.png,1
11_r_013.png,1
12_r_014.png,1
13_r_015.png,0
14_r_016.png,0
15_r_017.png,0
16_r_018.png,1
17_r_019.png,1
```

Figure 12: Sample Weka Training ARFF

```
@relation Rubbish

@attribute filename string
@attribute class {1,0}

@data
"rubbish_70.png",?
"rubbish_71.png",?
"rubbish_72.png",?
"rubbish_73.png",?
"rubbish_74.png",?
"rubbish_75.png",?
"rubbish_76.png",?
"rubbish_77.png",?
```

Figure 13: Sample Weka Prediction ARFF

A.5 Weka GUI Samples

The following provides a few screenshots illustrating use of the Weka GUI to input a training dataset, select a model, configure hyperparameters, and view the results.

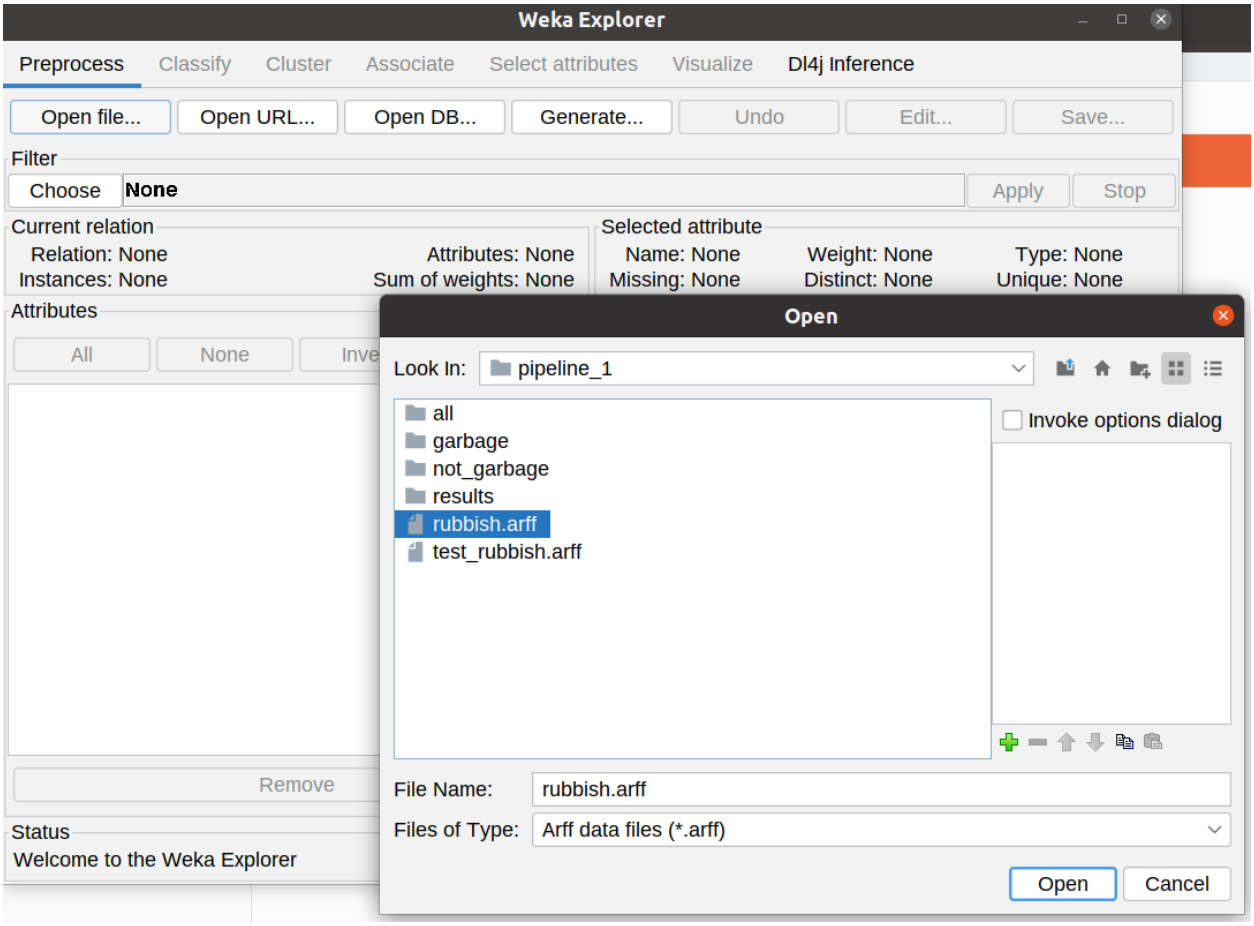


Figure 14: Weka - upload a dataset

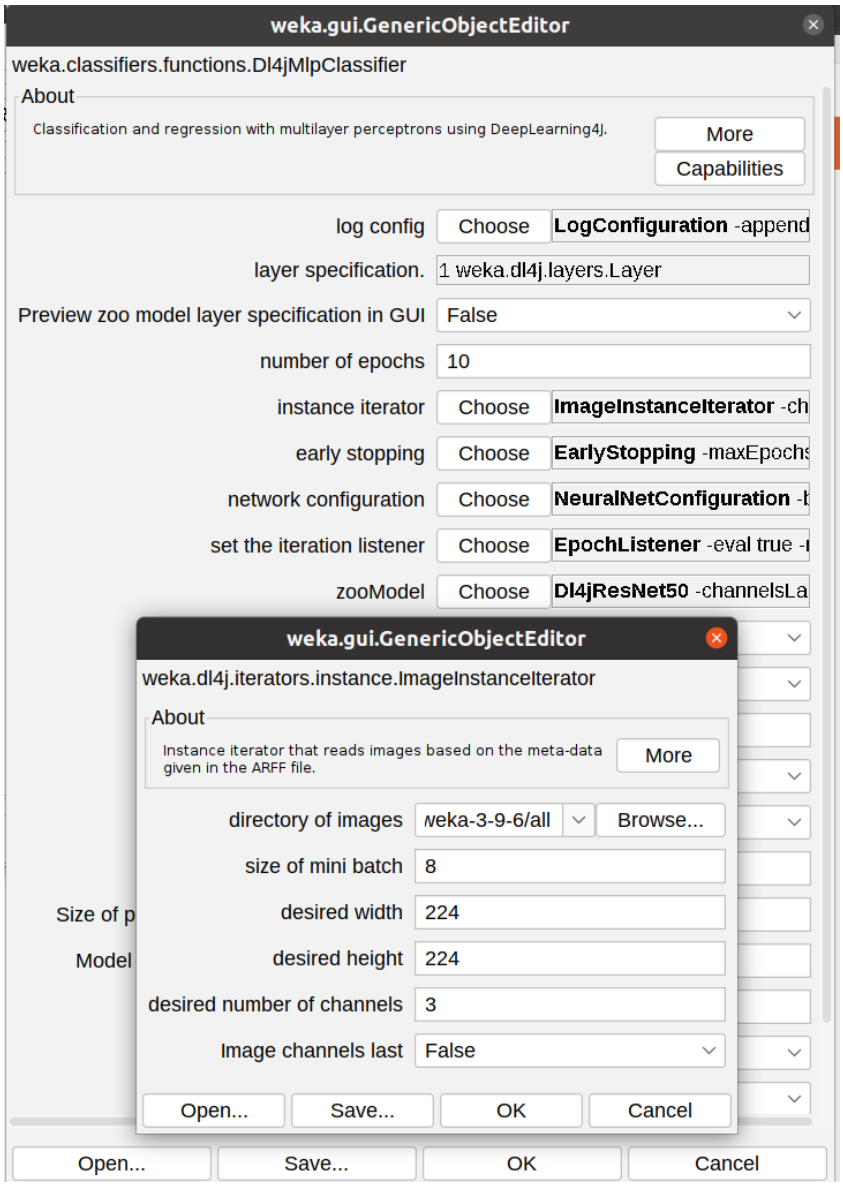


Figure 15: Weka - Associate an Image Iterator with a dataset

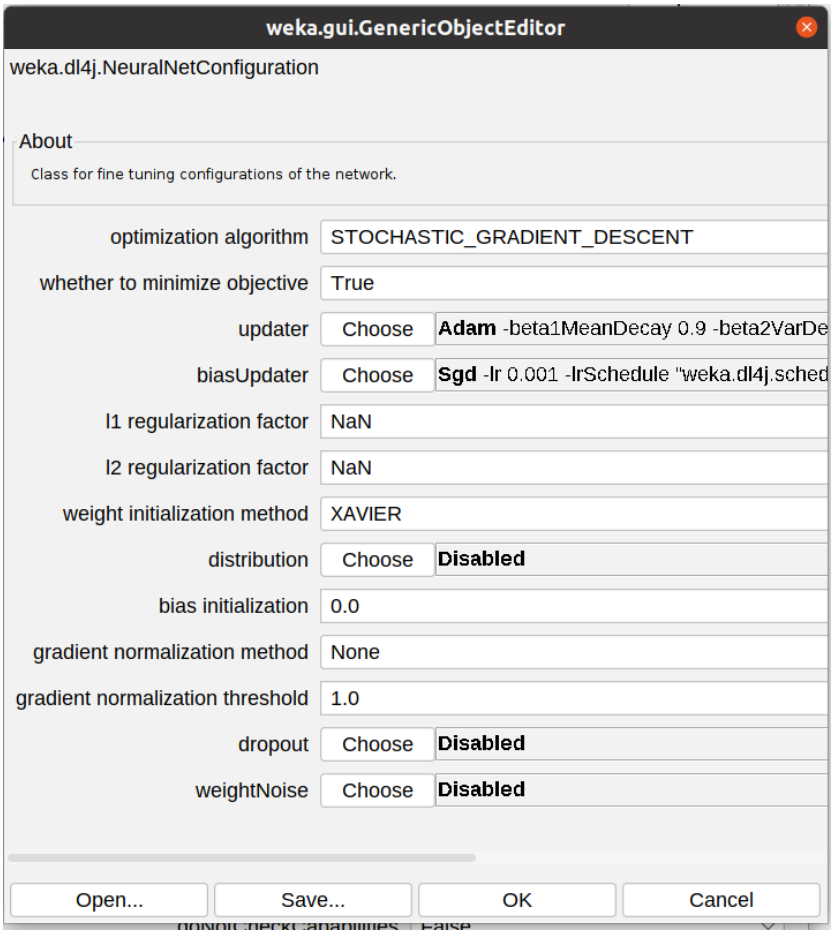


Figure 16: Weka - configure hyperparameters

```
Time taken to build model: 1141.42 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          55               55      %
Incorrectly Classified Instances        45               45      %
Kappa statistic                          0.1
Mean absolute error                      0.459
Root mean squared error                  0.5419
Relative absolute error                 91.7911 %
Root relative squared error            108.3857 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area
                 0.500    0.400    0.556      0.500    0.526      0.101    0.576     0.611
                 0.600    0.500    0.545      0.600    0.571      0.101    0.576     0.563
Weighted Avg.    0.550    0.450    0.551      0.550    0.549      0.101    0.576     0.587

=== Confusion Matrix ===

  a  b   <-- classified as
 25 25 |  a = 1
 20 30 |  b = 0
```

Figure 17: Weka - view results

A.6 Weka Command Line Scripts

The following extract from a command line (bash) script illustrates how a model can be trained by running the DeepLearningClassifier on a dataset.

```
# The following gives examples of the most important parameters in the script:
#       - use mini-batch size = 8, and 5 fold cross-validation:        -bs 8, -x 5
#       - use split percentage instead:         -split-percentage 20
#       - the location of the arff (or csv) file:       -t ../rubbish.arff
#       - the location of the image files:      -imagesLocation ./all
#       - desired size of the images:  -height 224 -width 224
#       - where to stored the resultant model:  -d test.model
# For a full list of parameters run it without any parameters set:
#       java -Xmx16g -cp weka.jar:wekaDeeplearning4j-1.7.2.jar:lib/* weka.classifiers.functions.Dl4jMlpClassifier
# Or visit the documentation at:

java -Xmx16g -cp ../java/weka.jar:../java/wekaDeeplearning4j-1.7.2.jar:../java/lib/* weka.classifiers.functions.Dl4jMlpClassifier -S 1 -cache-mode MEMORY -early-
stopping "weka.dl4j.earlystopping.EarlyStopping -maxEpochsNoImprovement 0 -valPercentage 0.0" -normalization "Standardize training data" -iterator
"weka.dl4j.iterators.instance.ImageInstanceIterator -channelsLast false -height 224 -imagesLocation ./all -numChannels 1 -width 224 -bs 8" -iteration-listener
"weka.dl4j.listener.EpochListener -eval true -n 5" -layer "weka.dl4j.layers.OutputLayer -lossFn \"weka.dl4j.lossfunctions.LossMCXENT \" -nOut 2 -activation
\"weka.dl4j.activations.ActivationSoftmax \" -name \"Output layer\"" -logConfig "weka.core.LogConfiguration -append true -dl4jLogLevel WARN -logFile ./-
wekaDeeplearning4j.log -nd4jLogLevel INFO -wekaDl4jLogLevel INFO" -config "weka.dl4j.NeuralNetConfiguration -biasInit 0.0 -biasUpdater \"weka.dl4j.updater.Sgd -lr
0.001 -lrSchedule \\\"weka.dl4j.schedules.ConstantSchedule -scheduleType EPOCH\\\"\" -dist \"weka.dl4j.distribution.Disabled \" -dropout
\"weka.dl4j.dropout.Disabled \" -gradientNormalization None -gradNormThreshold 1.0 -l1 NaN -l2 NaN -minimize -algorithm STOCHASTIC_GRADIENT_DESCENT -updater
\"weka.dl4j.updater.Adam -beta1MeanDecay 0.9 -beta2VarDecay 0.999 -epsilon 1.0E-8 -lr 0.001 -lrSchedule \\\"weka.dl4j.schedules.ConstantSchedule -scheduleType EPOCH\
\\\"\" -weightInit XAVIER -weightNoise \"weka.dl4j.weightnoise.Disabled \"" -numEpochs 10 -numGPUs 1 -averagingFrequency 10 -prefetchSize 24 -queueSize 0 -zooModel
"weka.dl4j.zoo.Dl4jResNet50 -channelsLast false -pretrained IMAGENET" -x 5 -t ../datasets/singly_augmented/baseline/rubbish.arff -d test.model
```

Figure 18: Sample Weka Training Script

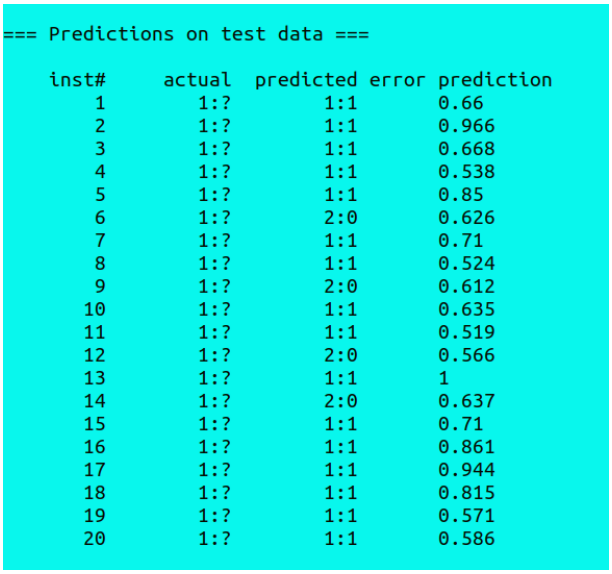
The following extract from a command line (bash) script illustrates how a prediction can be run on a set of new images.

```
# This script runs a set of predictions contained in rubbish_pred.arff using the model test.model trained previously
using train_dl4j.sh
# Note that a limitation of the Weka implementation is that the images being predicted against *must* be in the same
# class directory name as the images against which they were trained, in this case "all".
java -Xmx16g -cp ../java/weka.jar:../java/wekaDeeplearning4j-1.7.2.jar:../java/lib/*
weka.classifiers.functions.Dl4jMlpClassifier -T ./rubbish_pred.arff -l ./test.model -p 0
```

Figure 19: Sample Weka Prediction Script

A.7 Weka Prediction Results

The following figure illustrates the prediction results of applying the ”best” model (pipeline 3, trained on 2400 samples with mini-batch size of 32) to a set of twenty new images. The prediction of garbage (1) or not garbage (0) is given for each images, together with a confidence of how correct that prediction is, expressed as a percentage.



```
=== Predictions on test data ===

    inst#     actual  predicted error prediction
        1        1:?        1:1       0.66
        2        1:?        1:1       0.966
        3        1:?        1:1       0.668
        4        1:?        1:1       0.538
        5        1:?        1:1       0.85
        6        1:?        2:0       0.626
        7        1:?        1:1       0.71
        8        1:?        1:1       0.524
        9        1:?        2:0       0.612
       10        1:?        1:1       0.635
       11        1:?        1:1       0.519
       12        1:?        2:0       0.566
       13        1:?        1:1       1
       14        1:?        2:0       0.637
       15        1:?        1:1       0.71
       16        1:?        1:1       0.861
       17        1:?        1:1       0.944
       18        1:?        1:1       0.815
       19        1:?        1:1       0.571
       20        1:?        1:1       0.586
```

Figure 20: Weka Prediction Results - pipeline 3 model

References

[1] M. Vrijheid, “Health effects of residence near hazardous waste landfill sites: a review of epidemiologic literature.” *Environmental Health Perspectives*, vol. 108, no. suppl 1, pp. 101–112, Mar. 2000.

[2] G. Gabrielides, A. Golik, L. Loizides, M. Marino, F. Bingel, and M. Torregrossa, “Man-made garbage pollution on the Mediterranean coastline,” *Marine Pollution Bulletin*, vol. 23, pp. 437–441, 1991, publisher: Elsevier.

[3] B. Webb, B. Marshall, S. Czarnomski, and N. Tilley, “Fly-tipping: Causes, incentives and solutions,” *Internet: http://archive. defra. gov. uk/environment/quality/local/flytipping/documents/flytipping-causes. pdf (5. 8. 2011)*, 2006.

[4] T. Kinnaman and D. Fullerton, “Garbage and Recycling in Communities with Curbside Recycling and Unit-Based Pricing,” National Bureau of Economic Research, Cambridge, MA, Tech. Rep. w6021, Apr. 1997.

[5] A. Rajkumar, C. A. Kft, T. Sziranyi, and A. Majdik, “Detecting Landfills Using Multi-Spectral Satellite Images And Deep Learning Methods,” *Proceedings from ICLR 2022*, 2022.

[6] M. R. Devesa and A. V. Brust, “Mapping illegal waste dumping sites with neural-network classification of satellite imagery,” 2021. [Online]. Available: https://arxiv.org/abs/2110.08599

[7] M. A. Wulder, T. R. Loveland, D. P. Roy, C. J. Crawford, J. G. Masek, C. E. Woodcock, R. G. Allen, M. C. Anderson, A. S. Belward, W. B. Cohen, J. Dwyer, A. Erb, F. Gao, P. Griffiths, D. Helder, T. Hermosilla, J. D. Hipple, P. Hostert, M. J. Hughes, J. Huntington, D. M. Johnson, R. Kennedy, A. Kilic, Z. Li, L. Lymburner, J. McCorkel, N. Pahlevan, T. A. Scambos, C. Schaaf, J. R. Schott, Y. Sheng, J. Storey, E. Vermote, J. Vogelmann, J. C. White, R. H. Wynne, and Z. Zhu, “Current status of Landsat program, science, and applications,” *Remote Sensing of Environment*, vol. 225, pp. 127–147, May 2019.

[8] Q. Zhao, L. Yu, X. Li, D. Peng, Y. Zhang, and P. Gong, “Progress and trends in the application of Google Earth and Google Earth Engine,” *Remote Sensing*, vol. 13, no. 18, p. 3778, 2021, publisher: MDPI.

[9] G. Patowary, M. Agarwalla, S. Agarwal, and M. P. Sarma, “A Lightweight CNN Architecture for Land Classification on Satellite Images,” in *2020 International Conference on Computational Performance Evaluation (ComPE)*. Shillong, India: IEEE, Jul. 2020, pp. 362–366.

[10] M. Ghaffar, A. McKinstry, T. Maul, and T. Vu, “Data augmentation approaches for satellite image super-resolution,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 47–54, 2019, publisher: Copernicus GmbH.

[11] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019, publisher: SpringerOpen.

[12] R. Andonie and A.-C. Florea, “Weighted random search for CNN hyperparameter optimization,” *arXiv preprint arXiv:2003.13300*, 2020.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[14] P. Shamsolmoali, D. Kumar Jain, M. Zareapoor, J. Yang, and M. Afshar Alam, “High-dimensional multimedia classification using deep CNN and extended residual units,” *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 23 867–23 882, 2019, publisher: Springer.

[15] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, “Land use classification in remote sensing images by convolutional neural networks,” *arXiv preprint arXiv:1508.00092*, 2015.

[16] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, “Support vector machine versus random forest for remote sensing image

classification: A meta-analysis and systematic review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 6308–6325, 2020, publisher: IEEE.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989, publisher: MIT Press.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015, publisher: Nature Publishing Group UK London.

[19] C. C. Charu C. Aggarwal, "The Basic Structure of a Convolutional Network," in *Neural networks and deep learning*, 1st ed. Springer, 2018, pp. 318–322, publisher: Springer.

[20] K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, publisher: IEEE.

[22] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[23] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with mnist and emnist," *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019, publisher: MDPI.

[24] G. Forkuor, K. Dimobe, I. Serme, and J. E. Tondoh, "Landsat-8 vs. Sentinel-2: examining the added value of sentinel-2's red-edge bands to land-use and land-cover mapping in Burkina Faso," *GIScience & remote sensing*, vol. 55, no. 3, pp. 331–354, 2018, publisher: Taylor & Francis.

[25] M. Vaishnnave, K. S. Devi, and P. Srinivasan, “A study on deep learning models for satellite imagery,” *International Journal of Applied Engineering Research*, vol. 14, no. 4, pp. 881–887, 2019.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” *Advances in neural information processing systems*, vol. 27, 2014.

[28] S. V. Labs, “ImageNet,” 2020. [Online]. Available: https://www.image-net.org/

[29] D. Chicco, “Siamese neural networks: An overview,” *Artificial neural networks*, pp. 73–94, 2021, publisher: Springer.

[30] B. Li and L. Han, “Distance Weighted Cosine Similarity Measure for Text Classification,” in *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, and X. Yao, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 8206, pp. 611–618.

[31] C. C. Charu C. Aggarwal, “Tuning Hyperparameters,” in *Neural networks and deep learning*, 1st ed. Springer, 2018, p. 125, publisher: Springer.

[32] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Neural Networks: Tricks of the Trade: Second Edition*, pp. 437–478, 2012, publisher: Springer.

[33] A. Parvat, J. Chavan, S. Kadam, S. Dev, and V. Pathak, “A survey of deep-learning frameworks,” in *2017 International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2017, pp. 1–7.

[34] “DeepLearning4J.” [Online]. Available: https://github.com/deeplearning4j/deeplearning4j

[35] Eibe Frank, Mark A. Hall, and Ian H. Witten, “The WEKA Workbench. Online Appendix for ”Data Mining: Practical Machine Learning Tools and Techniques,” in *The WEKA Workbench. Online Appendix for ”Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Morgan Kaufmann, 2016.

[36] S. Lang, F. Bravo-Marquez, C. Beckham, M. Hall, and E. Frank, “Wekadeeplearning4j: A deep learning package for weka based on deeplearning4j,” *Knowledge-Based Systems*, vol. 178, pp. 48–50, 2019, publisher: Elsevier.

[37] United States Geological Survey, “USGS Earth Explorer.”

[38] European Space Agency, “Copernicus Open Access Hub.” [Online]. Available: https://scihub.copernicus.eu/

[39] Google Corp., “Google Earth Engine.” [Online]. Available: https://earthengine.google.com/

[40] L. Taylor and G. Nitschke, “Improving deep learning with generic data augmentation,” in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 1542–1547.

[41] R. Ashmore, R. Calinescu, and C. Paterson, “Assuring the machine learning lifecycle: Desiderata, methods, and challenges,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021, publisher: ACM New York, NY, USA.

[42] “Handling imbalanced datasets in machine learning.” [Online]. Available: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

[43] M. Skogsmo, “A Scalable Approach for Detecting Dumpsites using Automatic Target Recognition with Feature Selection and SVM through Satellite Imagery,” 2020.

[44] M. L. Richter, W. Byttner, U. Krumnack, A. Wiedenroth, L. Schallner, and J. Shenk, “(Input) Size Matters for CNN Classifiers,” in *Artificial Neural Networks and Machine*

Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30. Springer, 2021, pp. 133–144.

[45] Yann LeCun, “The MNIST database of handwritten digits.” [Online]. Available: http://yann.lecun.com/exdb/mnist/

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, place: New York, NY, USA Publisher: Association for Computing Machinery.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[48] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, “Deepsat: a learning framework for satellite imagery,” in *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, 2015, pp. 1–10.

[49] D. L. Donoho and others, “High-dimensional data analysis: The curses and blessings of dimensionality,” *AMS math challenges lecture*, vol. 1, no. 2000, p. 32, 2000, publisher: Citeseer.

[50] C. C. Charu C. Aggarwal, “Pooling,” in *Neural networks and deep learning*, 1st ed. Springer, 2018, pp. 326–327, publisher: Springer.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and others, “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, pp. 211–252, 2015, publisher: Springer.

[52] A. Müller and S. Guido, “Model evaluation and improvement in Introduction to Machine Learning with Python, 1e (ed. Müller, AC) 262–263,” 2017.

[53] D. M. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” *arXiv preprint arXiv:2010.16061*, 2020.

[54] Boaz Shmueli, “Matthews Correlation Coefficient is The Best Classification Metric You’ve Never Heard Of,” Nov. 2019. [Online]. Available: https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a

[55] R. Raj, *Java Deep Learning Cookbook: Train neural networks for classification, NLP, and reinforcement learning using Deeplearning4j*. Packt Publishing Ltd, 2019.

[56] Y. Shu, Z. Kou, Z. Cao, J. Wang, and M. Long, “Zoo-tuning: Adaptive transfer from a zoo of models,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 9626–9637.

[57] I. Team, “ImageMagick.” [Online]. Available: https://imagemagick.org/

[58] F. team, “FFmpeg.” [Online]. Available: https://ffmpeg.org/

[59] R. Dandavate and V. Patodkar, “CNN and Data Augmentation Based Fruit Classification Model,” in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. Palladam, India: IEEE, Oct. 2020, pp. 784–787.

[60] X. Yu and C. Yi, “Design and Implementation of the Website Based on PHP & MYSQL,” in *2010 International Conference on E-Product E-Service and E-Entertainment*. IEEE, 2010, pp. 1–4.

[61] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, Apr. 1998.

[62] Brownlee, Jason (last), “A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size,” Jul. 2017. [Online]. Available: A Gentle Introduction to Mini-Batch Gradient Descent

[63] C. C. Charu C. Aggarwal, “Evaluating with Hold-Out and Cross-Validation,” in *Neural networks and deep learning*, 1st ed. Springer, 2018, pp. 179–180, publisher: Springer.

[64] C. C. Aggarwal, “Teaching Deep Learners to Generalize,” in *Neural networks and deep learning*, 1st ed. Springer, 2018, pp. 169–214.

[65] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *arXiv preprint arXiv:1609.04836*, 2016.

[66] S. J. Russell, “Generalization and overfitting,” in *Artificial intelligence a modern approach*, 3rd ed. Pearson Education, Inc., 2010, pp. 705–706.