Bachelor Thesis
Creative Technology

DumpingMapper:
Illegal dumping detection from high spatial resolution satellite imagery

Jochem G.D. Sallander

S2324903

University of Twente

Supervisor: Dr. A. Kamilaris

Critical Observer: Dr. J.W. Kamminga

February 2023

Abstract

In the consumption-based society of today, illegally dumped waste constitutes a real issue. Illegal dumping comes with several environmental, health, and economic issues and thus should be reduced. Additionally, recycling dumped waste would be a great step towards a circular economy. Literature suggests that deep learning models, more specifically convolutional neural networks (CNNs), trained on high-resolution satellite imagery can be a possible solution to this problem. CNNs are successfully used to classify materials or objects on satellite imagery in similar fields. This project aims to find out how deep learning models can be used to classify illegally dumped materials on satellite imagery. Initially, a few state-of-the-art CNN models are trained to classify between just images that have dumping and images that do not. This is done at the hand of over 20.000 manually annotated satellite images of Cyprus, where ground truth data was collected. The Inception V3 CNN model was able to reach an overall classification accuracy of 83%, where the size of the dataset and training time had the largest impact on performance. Then, several CNN models are trained to classify two very distinct classes of waste material. This is done at the hand of a much smaller pre-existing dataset which is annotated with the required details of less than 600 images. This dataset was used as the training data while a subset of the earlier Cyprus dataset was re-annotated and used to evaluate the models. Using data augmentation techniques to slightly reduce the limitations of the small dataset, the Inception V3 CNN model was able to reach a classification accuracy of only 58%. While the approach used in this project does not produce confident performance figures for CNN models at classifying illegally dumped waste material, the earlier results and increased performance over different iterations of models suggest that with a different approach and implementation of new techniques, CNN deep learning models combined with high-spatial resolution satellite imagery might still pose a possible solution to illegal waste dumping in the future.

Acknowledgments

I would like to thank my project supervisor, Dr. Andreas Kamilaris. First of all for the educational opportunity that he provided through this project, and secondly, for his guidance, the meetings, the feedback, and most importantly his enthusiasm that helped me complete this project. I would also like to thank one of Dr. Kamilaris' associates, Chirag P. Padubidri for showing support, sharing his knowledge on the subject, and giving ample opportunities for feedback and ideation.
I want to show gratitude toward my critical observer for this project, Dr. Jacob W. Kamminga for the opportunity and insightful feedback. Additionally, I would like to thank my fellow student Laurens F.A. van Helvoort for his contribution to the dataset used in this project.

Finally, I would like to show appreciation toward my friends and family, for the support, patience, and loyalty that they have shown me during the period of this graduation project as it helped me cross the finish line.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the consumption-based society we live in, a considerable amount of waste material is generated. In the EU alone, 203.660.000 tons of waste material were generated just in 2020 [1]. Over the past decades, the amount of material being disposed of as waste has only been increasing. Not all of this waste is properly disposed of through the right channels, however, as illegal dumpings of waste materials occur. These illegal dumpings will have definite negative effects on the surrounding environment and near communities in various ways. One example of this is the pollution of soil and groundwater as shown by Paz et al. [2] critical in agriculture. Also, there are health risks associated with living close to a waste-dumping location as found by Singh et al. [3]. Furthermore, illegal dumping has a negative impact on tourism, which could be a major problem for many communities which rely on tourism as a sizable part of their economies. Due to the large amount of waste material and the diversity of the types of materials dumped, it is difficult for governmental bodies and institutions to track down and take care of the illegally dumped material through the proper channels.

In order for authorities and institutions to deal with illegal dumping spots efficiently and with minimal damage to the environment and surrounding communities, the illegally dumped material must be located and the type of material properly classified. The classification of dumped waste materials is of great importance in making steps closer to a circular economy. The type of material that is dumped must be known in order for the authorities to contact the proper institutions to be responsible for the cleanup. These institutions are knowledgeable about the recycling process of those specific materials, maximizing the amount of material that can be recycled. This circular economy approach to the materials also has economic benefits, such as cost savings and new possible business opportunities as described by Kobza and Schuster [4]next to the obvious environmental ones.

An application of technology needs to be developed to automatically detect the illegal dumping grounds' locations and to accurately classify which types of material are dumped. A possible solution might be found in the field of Earth Observation (EO). The usage of deep learning in combination with Remote Sensing (RS) which is observing earth trough sensors from satellites or aircraft, has seen successful implementation in various similar applications, such as land cover classification as shown by Kussel et al. [5] as well as other specific implementations. In this project, we will find out how these techniques can be applied to the application of classification of dumped waste material.

1.1 Research Objectives

The purpose of this paper is to investigate the application of state-of-the-art deep learning classification techniques in combination with satellite imagery to classify waste material in illegal waste dumping grounds. Both classifying between satellite images that show dumping and background satellite images that do not show any dumping, as well as between different types of waste material on satellite images showing dumping. This can be formulated to form the main research question of this project as the following:

RQ: How can deep learning be used for the classification of illegally dumped materials from satellite imagery?

In order to answer this main research question, we first need to know how to structure a deep learning model to work optimally for this specific application as well as know how such a deep learning model can be best trained to be accurate in classifying the illegally dumped waste material. This can be formulated into the following sub-questions:

SQ1: Which deep learning models are structured optimally to classify illegally dumped materials?

SQ2: How can such a deep learning model be trained to perform accurately?

1.2 Report Outline

In the following report, the full process of structuring, training, and evaluating a deep learning model for the given application will be described. It will be structured like the following. In Chapter 2 the background research will be outlined. This includes the state-of-the-art of deep learning for image classification, as well as satellite imagery. Chapter 3 will outline the approach taken and the methods and techniques used in this project. The realization of the project will then be documented in chapter 4, after which the evaluation of the proposed solution will be discussed in chapter 5 and the results will be shown. Chapter 6 will be discussing limitations as well as possible future directions in this field. Finally, Chapter 7 will conclude the report followed by the appendixes and references.

2 Background Research

The aim of this chapter in the project is to get acquainted with previous work in the field as well as the state of the art to inform decisions later in the project. First, a literature review regarding the state of satellite imagery is conducted. Next, preliminary research was done on the state-of-the-art of deep learning models in the field of RS with the use of a literature matrix which can be seen in Appendix A. Finally, the training of such models for application in RS is discussed.

2.1 Satellite Imagery

In order to effectively apply deep learning techniques to satellite imagery for the purposes of classifying illegally dumped materials, we first need to gain an understanding of existing satellite imagery datasets and their characteristics. This subchapter consists of an overview of relevant satellite datasets and their features. Additionally, the possible combining of satellite datasets will be briefly discussed.

2.1.1 Sensor Types

There are several different types of satellite sensors that are used to produce satellite datasets which all fall into one of two main categories. The first main category of sensors is imaging sensors. In the category of imaging sensors, Zhu et al. [6] name the subcategories of optical imaging sensors, thermal imaging sensors, and radar imaging sensors. What defines optical imaging sensors according to Zhu et al. [6] is that these sensors are able to sense electromagnetic (ER) waves in the visible part of the spectrum or waves that have been reflected from the earth's surface. On the other hand, thermal sensors operate infrared and microwave ranges. Finally, radar imaging sensors are active sensors in the microwave range as defined by Zhu et al. [6]. This means that they first transmit ER waves and then sense the reflection.

When comparing different satellite datasets for a given purpose, not all types are always relevant. Both optical and radar sensors are also mentioned by Kumar and Bhagat [7]. The reason Kumar and Bhagat [7] leave out thermal imaging sensors in their paper could be because they focus on land-based applications of satellite imaging, in which thermal imaging does not have many applications as described by Zhu et al. [6]. Additionally, Kumar and Bhagat [7] describe hyperspectral imaging as a separate data type in contrast to Zhu et al. [6] who have sub-categorized hyperspectral imagery under optical imagery. Along with hyperspectral Zhu et al. [6] also describe panchromatic and multispectral as sub-categories of optical imagery. These sub-categories are differentiated by the number of spectral bands and the spectral range in the sensor, which are the two main characteristics that differentiate the optical type of satellite imagery datasets.

The second main category of sensors is non-imaging type sensors. Zhu et al. [6] state that spectroradiometers, radiometers, and laser-based sensors belong in this category. Kumar and Bhagat

[7] also briefly describe LiDAR (Light Detection and Ranging) sensor-based datasets, which are laser-based. According to Zhu et al. [6], the types of datasets in the non-imaging category have no significantly land-based applications except for LiDAR. However, LiDAR can only really be used for topographic mapping according to Kumar and Bhagat [7] and Zhu et al. [6]. Given our use case for satellite imagery, non-imaging-based datasets might not be as useful to us as imaging-based datasets when used standalone.

2.1.2 Characteristics

There are four important main characteristics of satellite sensors which are reflected on the imagery datasets produced using these sensors. Liu [8] uses sensor type, spatial resolution, revisit time, and swath size as characteristics to differentiate the different satellites of which datasets are available. Sensor type, which was also used by Zhu et al. [6] and Kumar and Bhagat [7] to differentiate between satellites, intuitively means which type(s) of sensor(s) the satellite uses to capture data such as the types described above. As stated before, when the type of the sensor is optical, the two additional characteristics spectral range and the number of spectral bands are important. The spectral range is the range of wavelengths of the ER spectrum that a sensor can capture. This range is divided into different spectral bands. As defined by Zhu et al. [6], panchromatic-type sensors have a short spectral range and just one band, while multispectral sensors have a medium spectral range divided up into several bands. Intuitively hyperspectral sensors have a big spectral range divided up into a large number of small bands which can go up to a hundred spectral bands. Other characteristics apply to both optical as well as other types of satellite imagery sensors.

After sensor type, another significant dataset characteristic is spatial resolution. The characteristic of spatial resolution is used by Zhu et al. [6], Kumar and Bhagat [7], and Liu [8] as well as by Krueger [9] to differentiate satellites. The spatial resolution specifies the size of one pixel from a satellite image in relation to real life. This size indicates how small an object can be to still be distinguished by sensors as explained by Liu [8], it can vary between kilometers all the way down to centimeters. Liu [8] also explains that the revisit time is the amount of time it takes a satellite to recapture an exact location again after having just captured it, which is usually somewhere between a month and a day. This characteristic is also used by Zhu et al. [6] and Kumar and Bhagat [7]. Then there is swath size, which is also used to differentiate satellites by Zhu et al. [6] as well as Krueger [9]. A swath is the width of land that a sensor covers on every pass as made clear by Zhu et al. [6], which is usually a few hundred kilometers. Other factors not related to the sensor(s) used to create the dataset can factor into the decision on which datasets could be used for the given application such as the costs of the dataset as mentioned by Krueger [9], or the access and availability of a dataset as mentioned by Kumar and Bhagat [7]. The available datasets have many different characteristics which need to be taken into account depending on the use case, some characteristics are more influential than others.

When classifying satellite imagery using deep learning such as in this project, different characteristics have different influences on the accuracy of the classification. In general, the spatial resolution has the greatest effect, and spectral band selection after that, depending on the use case, with other characteristics having a relatively low influence on the classification accuracy. Momeni et al. [10] found that after changing the classification process, using a higher spatial resolution gave the highest rise in the accuracy of classification. This is in line with findings by Chen et al. [11] who also found changing the spatial resolution to have the biggest impact, although depending on the landscape being classified, higher resolution might not always lead to more accurate classification. After spatial resolution Momeni et al. [10] found the spectral band selection to also have a significant impact, stating that the spectral band selection must be deliberate as just adding more bands does not always result in higher accuracy. In contrast, Yu et al. [12] found the spectral band selection to have a larger effect on classification accuracy than spatial resolution supported by Bradter et al. [13] who found that using a large number of spectral bands gave the best results. This difference in findings could be due to the different landscapes and purpose in the different studies as both Momeni et al. [10] and Chen et al. [11] worked on urban landscapes while Yu et al. [12] and Bradter et al. [13] focused on vegetation classification. The effect different characteristics have on classification accuracy can differ depending on the use case, although the spatial resolution and spectral band selection are the most important characteristics.

As our objective is to classify dumped waste, it is necessary to use satellite imagery with a fine enough spatial resolution so that the waste can be identified, ideally under half a meter. Additionally, research in a similar field of urban land cover classification suggests that additional spectral bands can possibly enhance classification accuracy. Herold et al. [14] found that after bands in the visual range, near-infrared (NIR) and short-wave infrared (SWIR) have great importance for telling apart different types of ground cover materials. Additionally, Momeni et al. [15] found that urban land cover classification accuracy significantly increases if additional red-edge and NIR bands are included and potential improvements can be made by including yellow visible and coastal blue spectral bands.

2.1.3 Multi-modal learning

While for a standalone dataset optical imagery is most useful, it is possible to combine different types of satellite data, or others gathered through different means, in order to further improve the accuracy of classification by a deep learning model. Fang et al. [16] concluded that it is possible to improve classification accuracy by combining a high spatial resolution dataset with laser-based LiDAR data. Additionally, it was found by Orynbaikyzy [17] that adding radar data as well as optical features to the originally used bands results in a higher level of accuracy of classification. Orynbaikyzy [17] describe optical features as an additional layer of features that can be extracted from existing spectral bands. These optical features allow for an increase in classification accuracy using the existing dataset.

2.2 Convolutional Neural Networks

Neural networks (NNs) are a type of deep learning model directly inspired by biological neural networks or brains. A NN consists of layers of neurons or nodes that are connected between the layers. A schematic of a simple NN is visualized in figure 1. Each neuron can receive an input signal, apply a certain weight and bias to the received signal and then output the signal through an activation function. With the use of backpropagation and a cost function, changing the weights will allow the network to learn to recognize certain desired patterns. Convolutional Neural Networks (CNNs) are a type of NN that is particularly useful in learning patterns in images.

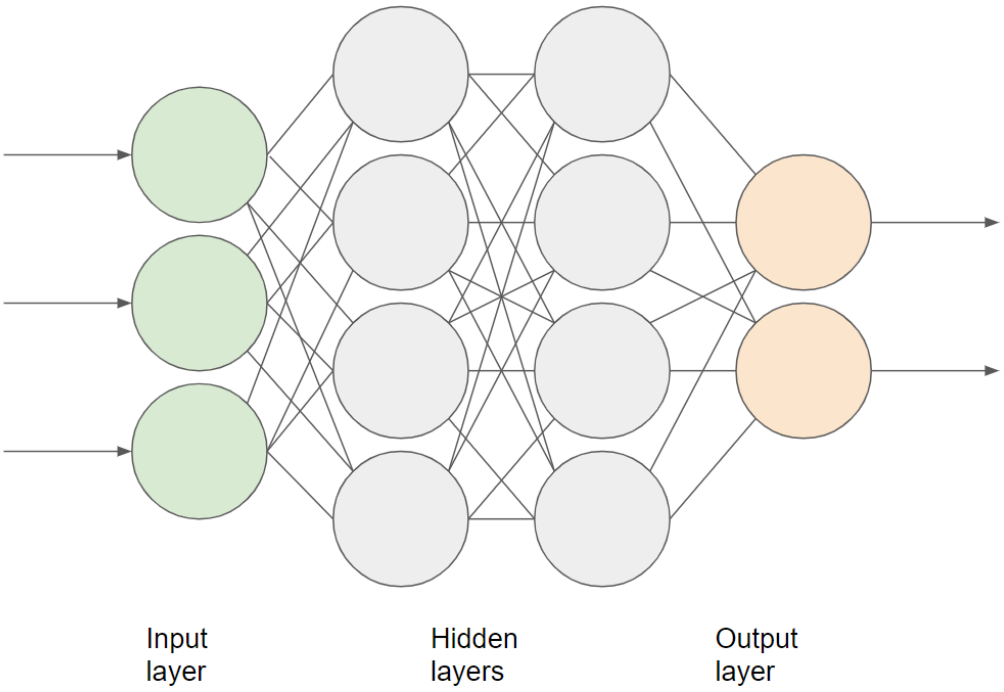


Figure 1. NN Diagram.

CNNs have seen a lot of success in the field of RS in similar applications. As Song et al. [18] describe, the advantage of CNNs in image classification is based on a few key characteristics. Firstly, the convolutional operation on groups of pixels allows for the extraction of certain abstract features from the data which can be used in the classification. Secondly, CNNs have the capability to rapidly acquire image information from heaps of data as allowed by the combination of pooling layers in their structure. Finally, the sparse connections, weight-sharing and spatial subsampling that characterize the simple structure of CNNs allow for more adaptability to the data structure of images.

CNNs have been found to be effective for the purpose of image classification in the field of RS. Kussel et al. [5] compared the land cover classification performance of a traditional NN as well as the Random Forrest (RF) deep learning model which is popular in the field of RS to two different CNN structures. It was found by Kussel et al. [5] that in this application the CNNs were able to reach a higher percentage of classification accuracy than both the RF and traditional NN deep learning models.

In this chapter, the CNN model structure will be investigated and the application of CNNs in the RS field will be explored.

2.2.1 CNN Structure

Similarly to a regular NN, the structure of a CNN is made up of different components which are stacked in layers. These layers are split up between the feature extractor part of the CNN structure and the feature classifier part of the CNN. The first part, the feature extractor, takes the input data and processes it into a set of features that are present in the input image. Things like lines, textures, or patterns. Then the feature classifier part of the CNN decides how to classify the input image based on those extracted features. The layers in the feature classifier part are fully connected, like in a deeply connected neural network (DNN). However, in the layers in the feature extractor, each neuron is, in contrast to in a DNN, only locally connected to a certain area of the previous layer. There are two types of layers in the feature extractor: convolutional layers, which give CNNs their name, and pooling layers. An image diagram of the structure of a CNN can be seen in Figure 2.

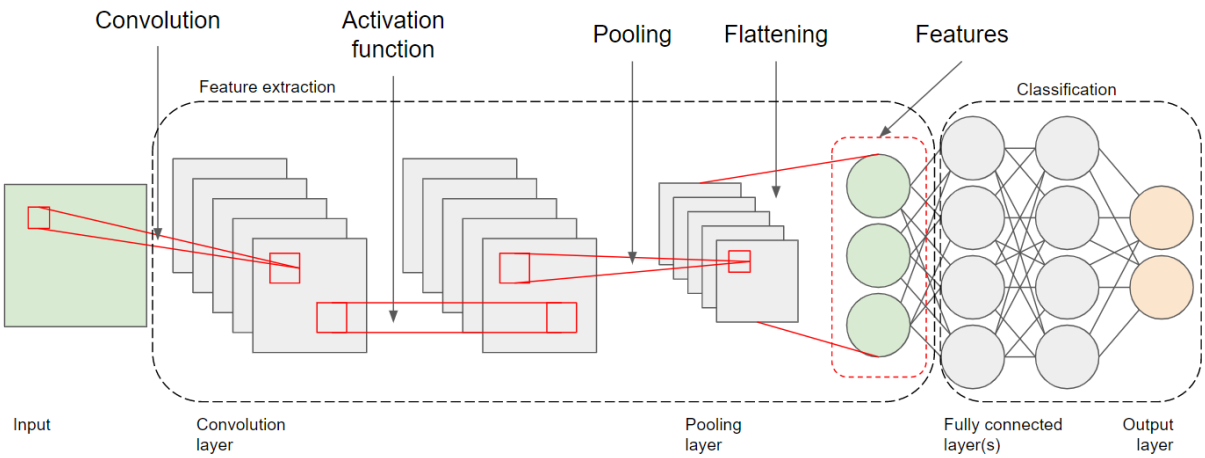


Figure 2. CNN Diagram.

A convolutional layer, intuitively, performs a convolutional operation which, like stated before, works on a small area of the input image defined by the convolutional kernel size. Portilla [19] explains that the convolutional kernel is a kind of filter that works as a matrix of different weights that are applied to the pixel values of each pixel within the convolutional kernel. The average of the weighted inputs forms the output of the convolutional layer at that particular location which, if done for the whole image, will form a convoluted image. This can then be used as an input for the next convolutional layer to extract complex features with each additional layer as Song et al. [18] describe.

A pooling layer, on the other hand, can be used to reduce the computational cost of training the network, by subsampling the image. Song et al. [18] explain that the usage of a pooling layer is to capture certain features while reducing the input data, so the model can still learn particular features effectively at a reduced computational load. Similarly to a convolutional layer, a pooling layer works

with a certain kernel size which determines how much of the input data is removed as explained by Portilla [19]. A pooling layer does not perform a convolutional operation but instead, a bunch of general pooling layers can be used such as maximum, average, and random pooling. These pooling layers either take the maximum or average pixel value inside the kernel respectively, while the random pooling selects a pixel value based on certain probabilities.

Activation and cost functions are also essential in a CNN. The activation as its name makes clear, is used to decide if a neuron is activated and thus whether it contributes to the network or not. Portilla [19] explains that traditionally this has usually been a Sigmoid function but recently Rectified Linear Unit (ReLU) by Hara et al. [20] and Maxout by Goodfellow et al. [21] functions proved more popular backed up by Song et al. [18]. The cost or loss function is used to measure the difference between a neuron's predicted value and the expected value. A popular cost function suggested by Portilla [19] is cross entropy as it allows for faster learning the more 'off' the neuron is from the expected value. Additionally, Song et al. [18] and Portilla [19] suggest extra additions to the loss function like L1 and L2 regularization can be used to prevent overfitting of the model.

The fully connected layer in the feature classifier part of the CNN has multiple hidden layers of fully connected neurons. This fully connected layer takes the feature images output by the convolutional and pooling layers in the form of a one-dimensional vector as an input and maps the features to the output layer which finally classifies the data through a classification function such as Support Vector Machines (SVMs) or Softmax by [22].

2.2.2 Existing CNN Models

There are many pre-existing CNN models already available in existing literature, each with different structures, depths, and features. An overview of popular CNN models from the reviewed existing literature has been created in Table 1.

Model	Model depth	Structural features
LeNet [23]	4 layers	Simple CNN structure
AlexNet [24]	7 layers	C-layers interconnected
VGG [25]	19 layers	Increased depth, max pooling
GoogleNet [26]	22 layers	Inception modules
ResNet [27]	34 layers	Skip-connections and batch normalization
Inception V3/V4 [28]/[29]	48 layers	Residual connections, 1 x 1 filters
DenseNet [30]	5 layers	Cross-layer connection

Table 1. CNN Models Overview.

LeNet [23] is the first successful application of a CNN, first designed in 1990 and improved in 1998 to classify handwriting. This model consists of 2 convolutional layers (C-layers) and 2 pooling layers (P-layers) which alternate each other. This was expanded upon with AlexNet [24] which was made up of 5 C layers and 3 P layers but this time, not every C-layer was followed by a P-layer. VGG or VGGNet [25] was designed to analyze the capability of deeper CNN models, initially created with 19 layers. GoogleNet [26] was based on a new architecture, made up of 22 Inception layers which improved the utilization of computer resources. ResNet [27] is inspired by a 19-layer VGG model but expanded to 34 layers with fewer P-layers than the VGG model. ResNet [27] also introduced skip connections combined with batch normalization. In later iterations, ResNet [27] was expanded up to more than 1000 layers. Inception V3/V4 [28]/[29] expanded on the GoogleNet [26] structure by introducing residual connections, accelerating the training process. DenseNet [30] is more applicable in applications where a very deep CNN is not able to be properly trained, like when a low amount of training data is available. DenseNet [30] offers a dense cross-layer connection, allowing every layer to directly connect to all previous layers. A shallower CNN model like this will possibly work well in cases where a low amount of training data limits the possible accuracy of a CNN model.

2.2.3. CNN Models in Remote Sensing

When adapting existing CNN models to the field of RS, there are some things to consider. Satellite imagery includes fine-grain textures and shapes as well as additional spectral information. A number of changes can be made to various aspects of a CNN model in order for the CNN model to better capture features from satellite imagery and thus increase the classification accuracy.

In the field of computer vision where most CNN structures have been developed, the input to the model usually consists of an image with RGB (Red, Green, Blue) values. As previously discussed however, satellite images sometimes contain information on more spectral bands than just visible light. This data more precisely describes the earth's features and can thus be recommended to be included as input for an RS CNN. In addition to this, it is also possible to use multisource data, Duarte et al. [31] were able to gain 4% of classification accuracy in classifying damaged buildings by combining feature images generated from multiple airborne datasets with the ones generated from satellite imagery. Meanwhile, utilizing a novel 3-D deep learning model, Hamida et al. [32] used a combination of three datasets each with over 100 spectral bands with the addition of spatial data as input, allowing them to reach up to 98,4% of groundcover classification accuracy. Additionally, textural features or vegetation indices can be extracted from the existing satellite imagery and added to the input of a CNN model as additional features. Xia et al. [33] added 4 textural features; mean value, standard deviation, consistency, and entropy to RGB images, and significantly improved the classification accuracy of vegetation and roads. Also, Pritt and Chern [34] include features from the satellite imagery metadata and include those features in the fully connected layer of the model. Overall the addition of more spectral, index, textural information, metadata or, multisource data to the satellite imagery input of a

CNN model could help the CNN to recognize additional patterns and features allowing for a higher level of accuracy in classification.

A limitation of using a CNN in RS is that CNNs contain more parameters than more traditional, shallow NNs. To accurately train these parameters more training data can ensure a faster convergence of the model. In contrast to the field of computer vision, there are few training datasets suitable for satellite image classification, let alone a specific application such as waste material classification. Therefore, reducing the number of parameters in the model can aid the model to converge accurately. A possible solution to this is to reduce the parameters in the fully connected layers with the use of a dropout function. According to Portilla [19] this randomly drops certain units to prevent overfitting of the model. Pritt and Chern [34] and Kussel et al. [5] both include a dropout with 0.6 and 0.5 probability respectively and reach high levels of classification accuracy on multispectral satellite imagery. Additionally, Zhong et al. [35] use a global average pooling layer in place of a fully connected layer for the classification of features in addition to a dropout function, reducing the parameters significantly.

Two popular classifier functions in the computer vision field are the Support Vector Machine (SVM) and Softmax [22] classifiers. In literature, the Softmax classifier is more prominent in papers covering successful classification through satellite imagery, such as in [32], [34], [36], [37]. Although the SVM classifier has seen use in similar use cases like by Zhang et al. [38]. Adedeji and Wang [39] also use a SVM for waste material classification although they use regular images instead of satellite imagery. However, Song et al. [18] argue that these classifiers might be inadequate for the RS field due to high noise levels in some cases. Bai et al. [40] used a conventional classifier for this reason.

The addition of L1 or L2 regularization as additions to the loss function of a CNN can help against model overfitting as it penalizes larger weights in the model as explained by Portilla [19]. This is also used by Kussel et al. [5] in land cover classification.

Increasing the number of layers of a CNN model can lead to a bump in classification accuracy level, however, adding layers to a CNN will also increase the number of parameters which, as previously discussed, requires more training data to train effectively. A possible workaround for this is to create multiple independent CNN models with different structures and then integrate them which can be done in various ways. Pritt and Chern[34] created an ensemble of 4 existing CNN models each with its own fully connected layer. Then through the unweighted average of the four different models, the classification is decided. This structure helped to classify buildings into 62 classes with an accuracy of up to %95 for some classes. Li et al.[37] suggest a different method where differently structured CNNs vote on the land use classification resulting in a 94% overall accuracy. The combining of various CNN models into a single structure might be a feasible solution to the classification of waste materials as a very low amount of training data is available.

Parameter initialization and optimization are important parts of any CNN. For parameter optimization, traditional parameter optimizers can be used. Both Song et al. [18] and Portilla [19] list RMSprop, Adam, Adadelta, Adagrad, and Stochastic Gradient Descent as possible optimizers. Parameter initialization is more complex, however. Initialization with random values has a low training efficiency and requires a large dataset according to Nogueira et al. [41]. This process might result in the most accurate model, however. A different strategy suggested by Nogueira et al. [41] is to take an existing model that has already been trained, so the parameters are already developed, and then retrain that model to fit it to the specific use case.

2.3 Training CNN Models

Like regular neural networks CNN deep learning models rely on a sufficiently large dataset to extract and learn features. While it is possible to train a CNN model using a relatively smaller dataset there is a large chance of overfitting the model resulting in low classification accuracy when evaluating. While some open RS training sets exist, our model will be trained for a highly specific purpose for which there is no open-access RS dataset available that fits our criteria. Additionally, there is only a low number of labeled images available that show illegal waste dumping. There are however ways in which smaller datasets can be augmented, such as the creation of synthetic data and existing data augmentation techniques which will be explored in this chapter.

2.3.1 Data Augmentation

Since gaining more samples through manual visual interpretation is very inefficient and costs resources, methods have been deployed that automatically expand a dataset by applying certain simple operations to the pre-existing images in the dataset. Examples of augmentations that have been used on remote sensing datasets from literature are rotations at certain degree intervals, mirroring or flipping images, random translations, or scaling of the images as used in [31], [34], [36], [42] but also operations such as color value shifting, blurring and, brightness adjustment such as used by Pan et al.[36] and Padubidri et al. [42]. Furthermore, Zhong et al. [35] developed a novel dataset augmentation in which a window of variable size is applied to different parts of scenes in satellite images and then sampling from that window. Although sometimes simple, all of these augmentations can be used to enlarge datasets for the training CNN deep learning models.

2.3.2 Synthetic Data

In addition to augmenting a dataset with the traditional augmentation methods that utilize the existing images in a dataset, it is also possible to create entirely new data. For dumped waste detection using a CNN, Padubidri et al. [42] generated synthetic training data by assorting 3D models of dumped materials into random piles and rendering them on top of a satellite image background. This allowed them to greatly increase the size of the training data while they reserved most real-world data

for evaluating the model. Additionally, Martinson et al. [43] use a similar method of rendering 3D models on top of satellite imagery to create a dataset depicting construction/ marine/ flying vehicles. Martinson et al. [43] describe an additional step in which they use a generative adversarial network (GAN) to blend the 3D models into the image. The GAN model learned which modifications it could make to images were most effective in bringing them closer to a real image. However, they conclude that for training purposes, images created without the use of a GAN gave bigger improvements. They suggest the GAN is better used for creating validation images. However, to properly train a GAN for such a procedure, a significantly large dataset is required. As the application of this project is highly specific, we will rely on a small real-world dataset and thus, as proven in a similar application in literature, synthetic data generation could have a positive impact on performance, given the application.

3 Methods and Techniques

Firstly, in this chapter, the structure of the project is set out and compared to the Creative Technology Design Process. The Creative Technology Design Process was used as an inspiration for the structure of this project. However, it was not fully utilized as it does not completely fit the project at hand. After this, the approach and the methods and techniques used in the different parts of the approach are set out and explained.

3.1 Design Process

As this is a graduation project for the bachelor of Creative Technology inspiration was taken from the Creative Technology Design Process (CreaTe DP) by Mader and Eggink [44] to guide and structure the project. This design process, depicted in figure 3, is split up into four phases. The first three phases of which, ideation, specification, and realization, contain both a divergence and a convergence phase. In the divergence phase, the design space is open and starts tacking a form, while in the convergence phase, a certain solution is reached by narrowing the design space by making design solutions. Additionally, the spirals found in different phases of the process allow for many iterations while the separation of the phases creates a more guided, clear structure to design by.

While this structure is not exactly copied in this particular project, it was the main inspiration behind the structure that is used. Central to both the CreaTe DP and this project is the feedback loops that allow for easy iteration of prototypes. Specifically, after implementation, the parameters of the models are iterated upon. When a model is trained and the performance is evaluated, it will be decided which parameters are changed and another iteration of the model is trained. The differences through these iterations allow to see the impact that the different parameters have on the performance of the models. The iterations of the models and differing results from these model iterations are central to understanding and improving the implementation. In contrast to the CreaTe DP however, in this project the Ideation and the Realization are much more intertwined. The main part of the ideation and specification are done for constructing an approach or methodology in which multiple iterations of models can be used to test the effects of different parameters. Main aspects specified here are, how the dataset should be structured, how it is annotated, and what data will be used by the models in what way. Additionally, ideation and specification takes place between the model iterations where small tweaks in the code of the models can already have great differences to the performance. After the evaluation of an iteration of the model, a change is ideated and immediately implemented. This leads to more experimentation but a less structured approach.

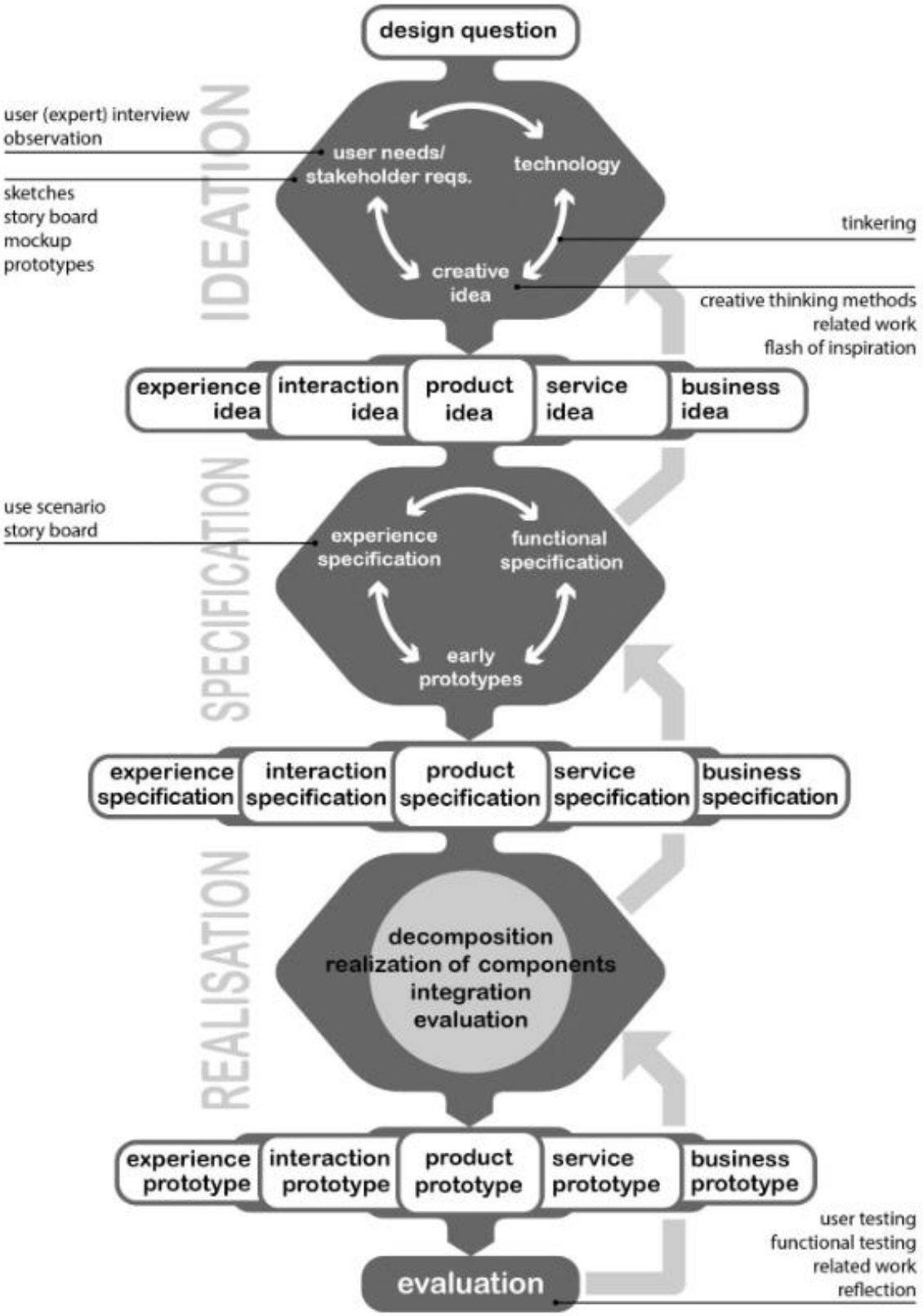


Figure 3. Diagram of the Creative Technology Design Process by Mader and Eggink [44].

3.1.1 Ideation and Specification

In the ideation phase of the CreaTe DP by Mader and Eggink [44] the design question will be turned into ideas that will answer the sub-questions. In the case of this project, this means that a possible approach to tackle the research questions will be thought of. The approach includes which deep learning models fit the implementation at hand and possible methods for training this model to perform the given purpose. This ideation end-point can be reached through an iterative creative thinking process that includes creative ideation, inspiration by technology, and assembling the requirements. For this project, the background research of the literature previously discussed will play a large role to this end. Literature on existing solutions to similar problems as well as literature on the state of the art in this field will inform the decision-making on potential deep learning models as well as learning methods. Different CNN structures and CNN training methods are considered and aspects from solutions for similar classification problems will be adapted to form the ideas.

In the specification of the CreaTe DP, the idea of the ideation phase will be further and more concretely defined into requirements. In this project, this phase is combined with the ideation to formulate the approach. Based on the literature but also the needs of the client, and the ideation, a clear and structured approach is set out. This approach is used as a plan for realization later in the project. These items will be documented in the second subchapter of the Methods and Techniques chapter as it will largely be a description of the methods used to implement, test, iterate, and evaluate the models.

3.1.2 Realization

In the CreaTe DP, these Ideation and Specification phases are followed by a Realization phase. This phase is also present in this graduation project. As the name implies, in the Realization phase of the project, the implementation of the various tested iterations are documented. The realization of the idea consists of a few parts. Firstly, the training data is prepared and pre-processing is done. This includes labeling, splitting training and test data, and formatting. Secondly, the chosen CNN models will be implemented and the models will be trained and tested. This phase will be reported on in the Realization chapter of this report.

3.1.3 Evaluation

The final phase of the CreaTe DP by Mader and Eggink [44] is the Evaluation phase. Again, this phase is also present in this project and is documented in the evaluation chapter of this report. Here, the performance of the trained models will be reported and evaluated. This evaluation is done at the hand of evaluation metrics specified in the evaluation chapter. The implications of these results between iterations are also discussed. Finally, the results and implications of the project as a whole are set out.

3.2 Approach

To give a structured approach to the project it is decided to split the project into two main phases. First, a more simple binary classification is attempted, in order to test the viability of utilizing the proposed technologies for the given purpose of classifying illegally dumped waste materials. For this binary classification problem models will be trained and tested for the ability to classify between two classes of images. One class of image where dumping can be distinguished and one class of images where no dumping can be distinguished visually. The results of this more basic classification problem will show us if the used CNN models are able to effectively recognize the textural, shape, color, and size features of waste dumpings. After this, a more complex classification is attempted in which the deep learning models are trained and evaluated on the ability to classify between different types of waste materials. For this more complex classification problem, a different method is required than for the binary dumping classification.

3.2.1 Binary Dumping Classification

As previously stated, a binary classification between the class of "Dumping" and "No Dumping" will be attempted first. This is done through multiple iterations of different models in order to reach a level of accuracy that inspires confidence to continue with a more complex classification problem.

3.2.1.1 Dataset

In order to train CNNs to an acceptable level a significant amount of training and testing data is required. In hyper-specific applications such as the classification of dumped waste material the lack of such datasets can become a limiting factor. While a large amount of satellite imagery is available, the images need to be properly labeled before they can be used for training a CNN. To label and gather satellite images, ground truth data is required. The gathering of ground truth and labeling the data is usually the most costly and time-consuming part of RS research.

For this project, ground truth data and an unlabeled dataset were acquired through the project coordinator. Ground truth was gathered on the island of Cyprus by volunteering citizens who recorded coordinates of locations where they had found dumping. In total 1016 distinct locations were recorded which are represented in figure 4. Now that the coordinates of a large amount of dumping locations have been gathered, satellite images from these locations can be captured. Through the google earth engine, a time series of 15 images were captured for each of the recorded locations at six-month intervals. This means that for every location an image was captured every six months, going back seven and a half years. This results in a dataset of over 15.000 images. However, as we have captured a time series over a relatively long amount of time, it is possible that some of the images were captured before the dumping occurred or after the dumping had been resolved. This means that not all of the images are guaranteed to contain dumping. So, manual annotation of the images is necessary.

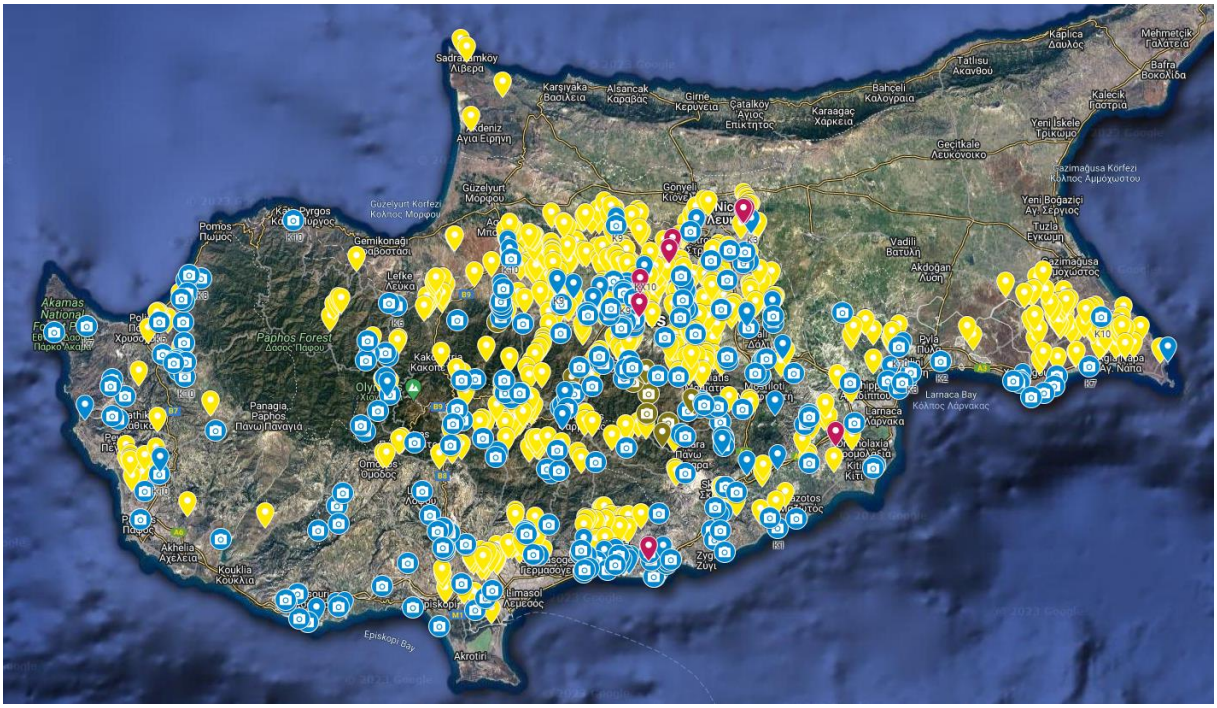


Figure 4. Map of Cyprus showing recorded dumping locations.

The images retrieved from the google earth engine have a high spatial resolution of around 0.30 m per pixel. Such high spatial resolution is necessary to be able to recognize waste from satellite imagery as usually dumped waste items are smaller than a square meter. A disadvantage of such high spatial resolution however is that the images of the given captured area will have a very high pixel count, being 6663 by 8192 pixels. This is a problem as such images require a huge amount of computational power. Additionally, most CNN models are limited to images of 512 by 512 pixels in size. Therefore it is required that the images are cropped. This is done using an automatic batch action in Adobe Photoshop. The images are cropped to 512x512 in the center of the images, as can be seen in figure 5. This is because the coordinates of the recorded dumping location are at the center of the captured image and thus, cropping to the center of the image ensures that the dumping location is not cropped out. Additionally, if it is required to balance out the classes after the annotation process, it is possible to acquire more images that do not show any dumping by taking croppings from the corners of the images. Since the dumping locations are centered in the images the corners should not show any dumping. Another advantage of the cropping of the images is that it will ease the manual annotation process as it reduces the overall captured area shown in the image. This reduces the area that needs to be manually searched for dumping.

Figure 5. Original image (Left) and cropped image (Right, enlarged).

The process of manually annotating the images consists of looking at each of the images and spotting whether there is dumping visible on the image or not. Additionally, some of the images are filtered out entirely if the cropped area is covered by cloud cover or other artifacts that make the image unusable. Since there is a significant amount of images to be annotated and manual annotation is a time-consuming task, the annotation of all of the images will be an ongoing process for the first half of the project. The annotation will be a process running parallel to the first few iterations of the model. This approach has the benefit of showing us how a growing training dataset influences the results of the models.

After a desired amount of images has been annotated the dataset is split before being used to train and test models. This train/ test split is common in deep learning projects and allows for the models to be correctly evaluated after training. The test data is kept separate from the training data, only being shown to the model after the training has been completed. This allows for the evaluation of the models to show how a model performs on entirely new data. In the case of this project, a percentage split of 80% train and 20% test is chosen, as this is the norm in the existing literature. This split is done using a Python script.

3.2.1.2 Model Iterations

When a certain initial amount of images have been annotated the models are run. Initially, a few different models will be run in the same circumstances with the same amount of data. This allows for a comparison of the performance between the different models. The chosen models are ResNet50 [27], Inception V3 [28], and VGG16 [25]. These particular CNN models have been chosen as they are all state-of-the-art models that have proven to be popular and effective in the existing literature. Additionally, these models have quite different architectures, allowing us to compare a wide spread of model structures. As training and testing models takes time and computational power, the best-performing model is selected and used for further iterating. This performance is judged with the evaluation metrics of accuracy, F-score, and confusion matrixes, as defined in the evaluation chapter. Eventually, the other models will also be evaluated again after several changes have been made through the iterations to make sure the originally selected model still performs the best under the new circumstances. Through the iterations, the parameters that will be experimented with are the size of the dataset, the balance of the dataset, and the training time. The models are implemented in Python scripts, utilizing the Tensorflow and Keras libraries. They are then run on the University of Twente High Performance Computing (HPC) cluster. This is further discussed in the Realization chapter.

3.2.2 Waste Material Type Classification

As previously set out, after the binary dumping/ no dumping classification the next part of the project focuses on exploring the effectiveness of deep learning models in classifying between different types of waste material. A similar approach is taken to the dumping/ no dumping classification although a different dataset is also utilized and different parameters are changed between iterations.

To make the classification less complex the models will initially be trained and tested to classify two types of waste materials. This will allow us to see the viability of using these models for such a complex classification task as well as see how the performance might be optimized. If time and scope allows for it, the classification of more material types can be explored later.

3.2.1.1 Dataset

As previously stated a different dataset is utilized in this more complex classification problem. This is because in order to train models to recognize different types of waste material we need images that are properly labeled with different types of waste material that are confirmed with ground truth data. Unfortunately, the dataset gathered and annotated earlier has not been annotated to this level of detail, and neither is there a possibility to do so since no ground truth data is collected on the types of material at the dumping sites. The AerialWaste dataset by Torres and Fraternali [45] was selected as a subset of this dataset has multi-class annotation. A total of 715 unique images have been labeled with waste types and storage types, some being labeled with multiple categories. A list of the different categories and the number of images with those labels can be seen in table 2. Note that this list

includes some duplicates between the categories and that the dataset also already has a train/ test split. As we will initially be classifying between two classes of waste materials, these categories can be divided into two overarching classes that resemble certain waste material types intuitively.

Label	Total Images	Train Images	Test Images
Rubble/ excavated earth and rocks	294	228	66
Bulky items	286	242	44
Fire wood	173	135	38
Scrap	167	140	27
Plastics	126	102	24
Vehicles	53	27	26
Tires	45	32	13
Domestic appliances	24	19	5
Paper	26	21	5
Sludge-Zootechnical waste-Manure	19	15	4
Foundry waste	9	8	1
Stone/marble processing waste	13	12	1
Asphalt milling	12	9	3
Corrugated sheets	11	10	1
Glass	8	6	2
Heaps not delimited	448	355	93
Full container	167	113	54
Big bags	50	31	19
Full pallets	50	43	7
Delimited heaps	69	38	31
Cisterns	35	26	9
Drums bins	18	16	2

Table 2. AerialWaste [45] dataset labels.

While this dataset can be used as training data, to see the viability of this implementation of technology in the target area of Cyprus the previously annotated dataset will be used for validating the models. However, like stated before, we do not have ground truth data on waste material type for these dumping locations. It should however be feasible to manually collect a small set of images for two classes based on context clues from the image. For example, a waste dump might be recognized as construction waste if it is right next to an actual construction site. This manual process will be difficult and time-consuming like the other annotation. However, a small set should be enough for testing data.

3.2.1.2 Model Iterations

For this part of the project, the best-performing model from the previous part will again be used. Additionally, the DenseNet [30] CNN model will also be evaluated. This is decided because of the much smaller size of the training and testing datasets, for which the broader structure of the DenseNet model should be optimized. Like the previous part of the project, A number of iterations of the models will be trained and evaluated with tweaked parameters in between. It is then possible to evaluate the effect of the changes in these parameters on the performance of the models. Like in the previous part, the training time will be changed. Additionally, data augmentation techniques will be utilized. These data augmentation techniques might produce interesting results regarding the limitation of the smaller dataset. Again the models will be implemented in Python with the Tensorflow and Keras scripts and run on the University of Twente HPC cluster.

4 Realization

In this chapter of the project report, the implementation of the methods previously discussed is documented. This will include the insights that came up during the implementation as well as how tools were used, including the pseudocode of the Python scripts. Additionally, the changes made between iterations of the models are further discussed. Again, this chapter is split into two parts, the binary dumping/ no dumping classification and the waste type classification.

4.1 Binary Dumping Classification

For the first part of the project, the realization consists of the pre-processing of the dataset and the actual implementation which are discussed one after another. These “phases” are actually parallel running processes, however, and the changes made between iterations shall be discussed afterward.

4.1.1 Dataset pre-processing

As previously stated, the first step of implementing a deep learning project is the pre-processing of the dataset. Due to the large number of images in the dataset, the work of pre-processing was split with a fellow student who would work with the same dataset. This was a helpful strategy for making this project feasible within the time constraints of a graduation project. After equally splitting the number of images the pre-processing went underway. Firstly, the images were cropped to 512x512 in the center of the image by using an automatic batch action in Adobe Photoshop. This tool allows you to record the changes you make to a single and then automatically apply them to a large size batch of similar images. Due to the large number of images and large file sizes of the images before cropping they were downloaded from the source in smaller batches. Each batch was then cropped before the next batch was downloaded, as hard drive space would allow. After the downloading and cropping of the images, the annotation process could start.

This annotation process consisted of visually inspecting every image and deciding if there was a visually recognizable waste dumping present or not. An example of what such images might look like can be seen in figure 6. For some images, this could be particularly hard due to terrain or hardly visible differences between images from the same location time series, such as shown in figure 7. The images were sorted into two files, each labeled accordingly. In addition to annotating between images showing dumping and no dumping, some images came up that could not be used for the training and evaluation of the models. This could be due to factors such as cloud cover, extremely dark images, or other artifacts. These images were removed from the dataset, a few examples of which can be seen in figure 8. Since even after splitting the work, there was still a large number of images to be annotated a tool was selected to make the process of sorting the images after visual analysis easier. PhotoSift was selected as it allowed for the easy sorting of images into different files with the press of a button. Another helpful feature of the PhotoSift tool is the accelerated loading of the images, which allows

fast scrolling through the dataset without any loading hiccups. Overall the manual annotation took about a month for the whole dataset. During this process, iterations of the models were already being trained on different sizes of the dataset. After the full annotation is completed, the dataset consists of 10.505 images in the “Dumping” class and 4.212 images in the “No Dumping” class.

Figure 6. An image showing dumping (Left) and an image showing no dumping (Right).

Figure 7. Two images from the same location with difficult terrain, one with dumping (Left).

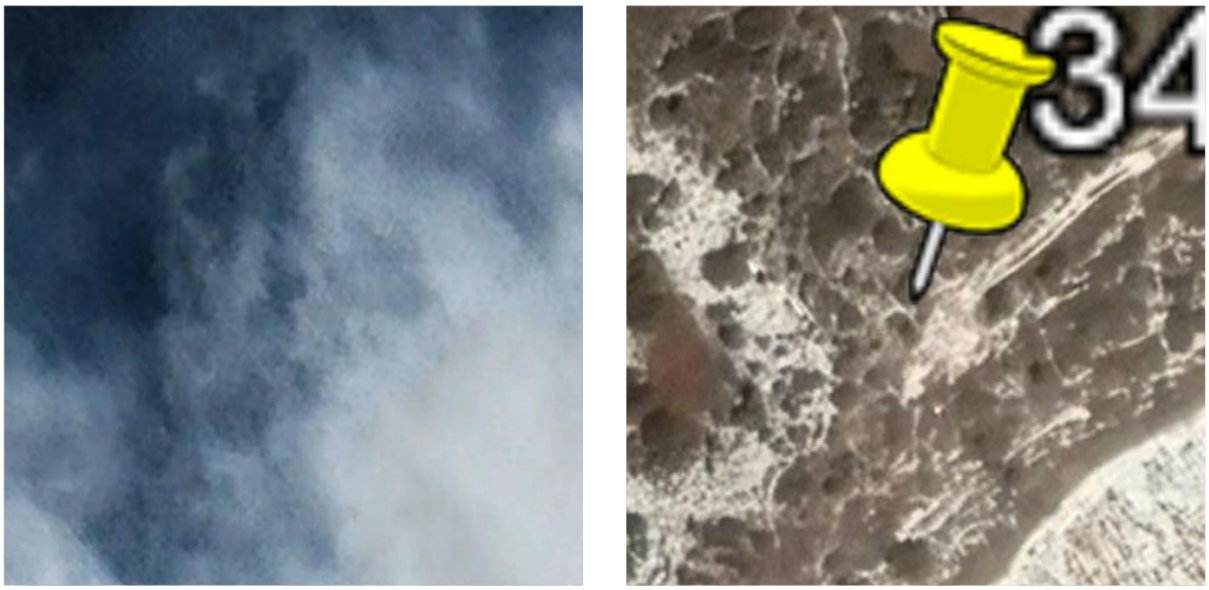


Figure 8. Two images deemed unusable.

Since there is a big difference between the number of images that show dumping and that do not show dumping new images are created to add to the "No Dumping" class. This is done by using the previously acquired full-sized satellite images and cropping sections of 512x512 out of the corners of the images, again using Adobe Photoshop automated batches. Since the coordinates of the dumping locations are in the center of the images, the corners should not contain dumping. 6.293 additional images are acquired through this process, ending up with 10.505 "Dumping" and 10.505 "No Dumping" images in the final iteration of the dataset.

When a certain amount of images are annotated for a certain iteration, the train/ test split is performed before the dataset is used. As previously set out, the split was realized through a small Python script with the use of a library called "splitfolders". This library has an easy function for splitting as well as allowing for a "seed" to be input. This makes sure that the split is done in the same way every time and thus, reducing the variance between iterations of the models. The script can be seen in Appendix B. After annotation and splitting, the dataset will have a file structure as can be seen in table 3.

Dataset		
	Train (80%)	
		Dumping
		No_Dumping
	Test (20%)	
		Dumping
		No_Dumping

Table 3. Dataset file structure.

4.1.2 Model Implementation

As previously discussed multiple different iterations of multiple CNN models are trained and evaluated. Fortunately, the implementations for these different models and the iterations between them are nearly identical based on code. Python is used to realize the models and by using the TensorFlow library with the addition of the Keras library, the models can be easily loaded. Thanks to these libraries the different models are easy to implement and do not have to be coded from scratch. Additionally, these libraries allow for the importing of pre-trained weights which allows for a faster converging of the model. This is beneficial, especially in cases where the dataset is smaller. While a script of one of the models is included in Appendix C as an example, the pseudocode of what these scripts look like can be seen in table 4. Of note here is that the final classification layer added on line 24 of the pseudocode has only a single node and a sigmoid activation function, this is because we are doing a binary classification. Another point of interest is that when compiling the model, on line 25 of the pseudocode, the desired evaluation metrics can be selected. In the case of this project, the evaluation metrics are accuracy, precision, recall, and the necessary components for a confusion matrix. These components are the true positives, the false positives, the true negatives, and the false negatives. These evaluation metrics will be further discussed in the evaluation chapter of this project. Finally, the batch size, which can be changed in line 26 of the pseudo-code is set to 16 for this project. This parameter determines the number of images that pass through the model at a time during training.

Line	Pseudocode
1	Import the relevant libraries and utilities
2	Import chosen model from Tensorflow and Keras library
3	**Create "prepare dataset" function**
4	Create array for training images
5	Create array for training labels
6	Get a path to dataset directory
7	For all the images in the directory:
8	- Load the image
9	- Convert image to right format
10	- Add image to image array
11	- Add label to label array
12	Return training images and training label arrays
13	"No Dumping" training data equals what **"prepare dataset" function** returns for "No Dumping" file directory
14	"Dumping" training data equals what **"prepare dataset" function** returns for "Dumping" file directory
15	Training dataset equals a combination of "No Dumping" training data & "Dumping" training data
16	"No Dumping" test data equals what **"prepare dataset" function** returns for "No Dumping" file directory
17	"Dumping" test data equals what **"prepare dataset" function** returns for "Dumping" file directory
18	Test dataset equals a combination of "No Dumping" training data & "Dumping" test data
19	Load chosen model with pre-trained weights
20	Remove last layer
21	Add flattening layer
22	Add densely connected layer
23	Add dropout layer
24	Add final classification layer
25	Compile the model with selected evaluation metrics
26	Train and evaluate model with given parameters

Table 4. Pseudocode of the used Python script.

In order to meet the high computational and storage needs that are required to run the prepared scripts, the University of Twente HPC cluster was utilized. This cluster allows students to access far greater computing power than what they might be able to access personally. Even with access to these powerful resources, some of the models took over 24 hours to train and evaluate. This is a good indication of how computationally heavy these types of deep learning models can be. For accessing the server and submitting tasks, the tool Windows Terminal was used while for uploading the dataset

and scripts to the server, WinSCP was used. Since the University of Twente HPC cluster the Slurm workload manager software to schedule the tasks and divide resources a Slurm job script needs to be created. This script tells the Slurm workload manager which scripts to run and what resources are required. Fortunately, an almost identical Slurm job script could be used for the different models, an example of which can be seen in Appendix D. For the purpose of this project, 16 processing cores are requested, each coming with 7 gigabytes of working memory. Additionally, 2 graphics processing units (GPUs) are requested. After several iterations of the dataset, these GPUs were specifically requested to be Nvidia Titan-Xs. This is because using lower-grade GPUs resulted in memory issues with a certain size of the dataset.

4.1.3 Iterations

Many iterations of training and evaluating models are realized. In this subchapter, I will be setting out these iterations. The performance of these different iterations will be later discussed in the evaluation chapter of the report. In this first part of the project, the main difference between these iterations is the size of the dataset. The first iteration is more like a test to see if the implementation can run successfully. This "test run" is a ResNet50 model [27] and the dataset consisted of only 1.380 images. After this "test run" is completed successfully a comparison of the three previously selected models is done. The ResNet50 [27], Inception V3 [28], and VGG16 [25] CNN models are trained and evaluated on a dataset of now 10.220 images. After this, the best-performing model is used for further iterations, which is the Inception V3 model [28]. The next iteration is just the Inception V3 model [28] on a dataset of 13.780 after which another two iterations are realized with the same model and dataset except that this time a "class weight" parameter is added to the model. This parameter, which can be added to the model for training at line 26 of the pseudocode, allows giving more weight to a class with fewer images to balance the effect the classes have on the model training. As the extra "No Dumping" images have not yet been generated, adding this parameter could yield improved performance. For the first of these two iterations, the difference between the number of images was approximated while for the second it was calculated exactly. The next iteration is another increase in dataset size, to 14.717 images which is the whole of the dataset annotated. For the next iteration, again, just the Inception V3 model [28] is used, and this time the additional "No Dumping" images have been added, for a total dataset of 21.010 images. Additionally, the number of epochs is increased to 3 from the default 1 in previous iterations. This parameter can also be changed at line 26 of the pseudocode and represents the number of times that the model will go through the entire dataset while training. Intuitively, a larger number of epochs should increase the performance. For the penultimate iteration, the number of epochs is increased to 5, and all three of the models are trained and evaluated again. Finally, the best-performing model is iterated on a final time by increasing the number of epochs to 10, allowing us to see if the performance will rise significantly or if the model has reached its limit.

4.2 Waste Material Type Classification

As previously stated it is decided to train and validate the models to classify between two classes of waste material types to make the classification problem less complex. To make sure that the manual annotation of these two classes from the previous dataset is viable, two waste very distinct waste material types are selected. The classes “Construction waste” and “Regular trash” should have textural and contextual features that differ greatly from each other, allowing for more easy manual annotation as well as a less complex task for the deep learning models.

Like the previous part, the realization of this part of the project consists of data pre-processing and model implementation. Again, these processes run parallel and the differences between iterations will be discussed.

4.2.1 Dataset pre-processing

As set out previously, the AerialWaste dataset from Torres and Fraternali [45] is used for the training dataset. As shown above, in table 2, a subset of the images in this dataset have been labeled with various waste type categories. In order to fit this dataset to the selected classes of “Construction waste” and “Regular trash” an intuitive selection of the categories is made to fit with each of the overarching classes. The division of categories and the number of images related to this can be seen in table 5. It is important to note again that some images have multiple labels and thus there are duplicates between the categories, the penultimate row of the table shows the number of images with these duplicates removed. Additionally, the AerialWaste dataset [45] includes images from the google earth engine, the worldview-3 satellite mission, and the AGEA airborne photography campaign. Since the images from the AGEA campaign are aerial photography and have a higher spatial resolution than the testing images, they should also be removed from the dataset. The last row of table 5 shows the number of images left. Note that since our own dataset is used for testing the models, both the training and testing parts of the AerialWaste dataset [45] are combined and used for training. Also, the images have been cropped centrally to 512 by 512 pixels using the same method in Adobe Photoshop.

Construction waste Class		**Regular trash Class**	
Label	Nr. of images	Label	Nr. of images
Rubble/excavated earth and rocks	294	Scrap	167
Stone/marble processing waste	13	Plastic	126
Asphalt milling	12	Domestic appliances	24
Corrugated sheets	11	Paper	26
Full Container	167	Big bags	50
Full pallets	50	Drums bins	18
Total	547		411
Total no duplicates	456		283
Total no duplicates and no AGEA	338		260

*Table 5. Division of categories from AerialWaste Dataset **[45]**.*

For the testing dataset for this waste material type classification problem, the manually annotated dataset previously utilized is used again. Since such distinct classes have been chosen it should be feasible to manually annotate a small number of images for both of the classes. The selected classes for this classification problem should also be recognizable from contextual information and the landscapes seen in the images. The “Regular trash” can be recognized by urban surroundings and small piles, while the “Construction waste” is very likely if the dumping is located right next to a construction site. An additional contextual clue to recognize construction is by utilizing the time series. If a building is present in one image but not in another image from the same location but at a different time, this might suggest that the building has been constructed or demolished between images and thus indicating construction. Like the previous annotation process, the manual selection was complex and time-consuming. In the end, from the 10.505 images with visible dumping sites, 100 distinct images were selected. 50 images in the “Construction waste” class and 50 images in the “Regular trash” class. An example from each of the classes can be seen in figure 9.

Figure 9. An image showing construction waste (Left) and an image showing regular trash (Right).

4.2.2 Model Implementation

Since the approach in this part of the project is largely the same as the previous part, the Python script used in the previous part of the project can be quickly adapted to work in this new classification problem. However, the AerialWaste dataset [45] that is used for the training data comes in a proprietary format and requires some bundled Python utilities to filter and extract the images. Additional code is added to the beginning of the script to adapt to this. A representation of the additional code is shown as pseudocode in table 6. This pseudocode replaces lines 13 to 15 of the pseudocode seen in table 4.

Line	Pseudocode
1	Import AerialWaste dataset utilities
2	Load list of training images from AerialWaste dataset with desired “Construction waste” categories using utility
3	Load list of test images from AerialWaste dataset with desired “Construction waste” categories using utility
4	Create list of “Construction waste” training images from both lists
5	Load list of training images from AerialWaste dataset with desired “Regular trash” categories using utility
6	Load list of test images from AerialWaste dataset with desired “Regular trash” categories using utility
7	Create list of “Regular trash” training images from both lists
8	“Construction waste” training data equals what **“prepare dataset” function** returns for “Construction waste” training images list
9	“Regular trash” training data equals what **“prepare dataset” function** returns for “Regular trash” training images list
10	Training dataset equals a combination of “Construction waste” training data & “Regular trash” training data

Table 6. Pseudocode for loading the AerialWaste dataset as training data.

For some iterations of models in this classification problem, data augmentation techniques are implemented. With the use of the TensorFlow and Keras libraries, data augmentation techniques can easily be added to the existing script. This is done through a so-called “data generator” which only adds a single line of code before training and evaluating the model. This line of code creates a “data generator” with the desired data augmentation techniques selected. Then the “data generator” is selected as one of the parameters in line 26 of the pseudocode seen in table 4. An example script with both the added code for the AerialWaste dataset and the data augmentation techniques can be found in Appendix E.

The iterations in this part of the project are again run on the University of Twente HPC cluster. For this purpose, the same basic Slurm job script can be used.

4.2.3 Iterations

As the Inception V3 model [28] performed the best in the previous classification problem, it is used again here in most of the iterations although two other models are also evaluated. Other than this, the differences between the iterations consist of the number of epochs used, the batch size, and the data augmentation techniques used.

The first iteration realized was the Inception V3 model [28] with 10 epochs. However, this iteration was conducted before the AGEA images were filtered out. The second iteration is conducted with the same parameters but this time the AGEA images are filtered out. For the next iteration, the

number of epochs is increased to 25 to try and increase model performance. For the next iterations, again with the Inception V3 model [28], data augmentation techniques are implemented to try and reduce the limitation of the small dataset size. The data augmentation techniques used are rotation of the images as well as vertical and horizontal flipping. These techniques for this iteration are relatively mild in changing the images and are selected as they do not have any effect on the textural features that might be present. This allows us to build up the augmentation techniques later on. This iteration was "repeated" but this time the number of epochs are increased again to 30. With the same data augmentation techniques and the number of epochs, the DenseNet CNN model [30] was evaluated. This allows us to directly compare this broad structured model to the more traditionally deep Inception model.

For the next iteration, we are going back to the Inception model again. This time the number of epochs was increased dramatically to 50 epochs. Increasing the number of epochs allows us to see if the limitations of the other parameters in the current setup are a limiting factor or if the performance still increases just by increasing the training time. In the following iteration, we change up these parameters by increasing both the number of data augmentation techniques used and using somewhat more drastic augmentation techniques. The techniques used in this iteration are again rotation and flips but additionally, width and height shifts, brightness shifts, shearing, zooming, and channel shifts are used. Like before, it is attempted to find the limitations of these augmentation techniques in the following iterations. One iteration is evaluated where the number of epochs is increased even further to 75 and another iteration for which the batch size is doubled from the 16 used in all other iterations in this project to 32. In the final iteration, a significantly different structured deep learning model is evaluated. The ResNet Feature Pyramid Network (FPN) model was used by the creators of the AerialWaste dataset, Torres and Fraternali in [45], with good performance. By evaluating this model with the methods and dataset used in the project, limitations can be identified as compared to using just the AerialWaste data for both training and testing.

5 Evaluation

In this chapter of the project report, the evaluation metrics of all iterations of both of the classification problems in this project are documented. The performance of the different iterations is also discussed and compared at the hand of some data visualizations.

5.1 Evaluation Metrics

For the purpose of evaluating a deep learning model after training, there are several possible metrics to choose from. These different metrics indicate the performance of the models in different ways. The most common and intuitive performance metric is the overall accuracy. This metric indicates the percentage of (test) images that the model has correctly predicted the label of. This metric is useful for quickly gauging the performance of models as well as for communication of performance to an audience. Because of these reasons, this metric is chosen as the main performance metric in this project. Other metrics have also been used between iterations, namely the F-score and confusion matrixes. The F-score is the harmonic mean of two other metrics, precision and recall. The precision tells us how many of the images that are predicted to show dumping by the model actually show dumping, while recall tells us how many of the images that actually show dumping are predicted to show dumping by the model. The F-score can be used similarly to accuracy. The final metric used was the confusion matrix. This is a 2 by 2 matrix showing the true positives, false positives, true negatives, and false negatives. This means that for the testing data, the confusion matrix tells how many dumping images are predicted right, how many are predicted wrong, how many non-dumping images are predicted right, and how many are predicted wrong.

5.2 Binary Dumping Classification

To easily compare the performance of the iterations from the dumping/ no dumping classification, table 7 is compiled. The table shows the test accuracy for all of the iterations as well as the parameters that change between the different iterations. Additionally, figure 10 visually sets out the test accuracy of the different models as compared to the size of the dataset.

What strikes from these results, especially iterations 2A, B, and C and 8A, B, and C, is that the Inception V3 model performs significantly better than the other models evaluated in this classification problem. This is unexpected as in the existing literature the ResNet family of CNN models seems the be the most popular. Curiously, between iterations 2A and 8A, the performance of the ResNet50 model actually dropped significantly. I am unsure as to what the reason for this performance drop could be, although this could be an outlier. Something that becomes clear from both table 7 and figure 10 is that increasing the dataset also significantly increases the model accuracy. This is expected. Additionally, it seems that having a “balanced” dataset is also beneficial to the model's performance. This is most noticeable between iterations 3, 4, and 5, where the dataset is balanced by changing the

class weights. It is also noticeable how increasing the number of epochs, or training time, can have a positive influence on the accuracy. This can be seen in iterations 7, 8B, and 9.

Iteration	Model	Images	Epochs	Test Accuracy	Notes
1	ResNet50	1.380	1	40%	Test run
2**A**	ResNet50	10.220	1	72%	
2**B**	Inception V3	10.220	1	76%	
2**C**	VGG16	10.220	1	72%	
3	Inception V3	13.780	1	72%	
4	Inception V3	13.780	1	75%	Class weight approximated
5	Inception V3	13.780	1	79%	Class weight calculated
6	Inception V3	14.717	1	77%	Class weight calculated
7	Inception V3	21.010	3	80%	Extra “No Dumping” images generated
8**A**	ResNet50	21.010	5	62%	Extra “No Dumping” images generated
8**B**	Inception V3	21.010	5	82%	Extra “No Dumping” images generated
8**C**	VGG16	21.010	5	77%	Extra “No Dumping” images generated
9	Inception V3	21.010	10	83%	Extra “No Dumping” images generated

Table 7. Binary Dumping Classification Results.

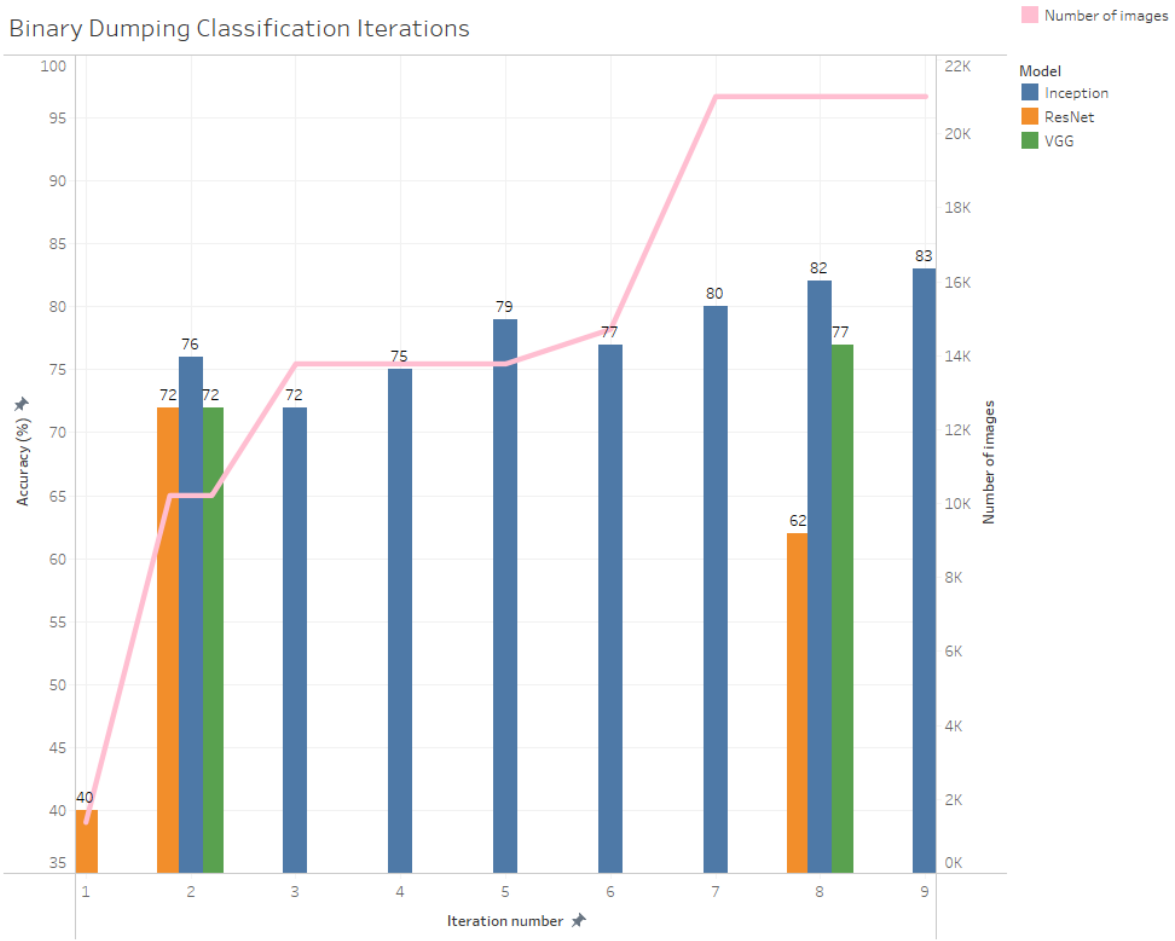


Figure 10. Model test accuracy and Dataset size.

Overall, the increased performance of the model after increasing the size of the dataset, balancing out the dataset, and increasing the training time of the model was expected from preliminary research and existing literature. The performance metrics of the final iteration which can be seen in table 8 show that there is definite merit to the usage of CNN models with this type of satellite imagery for the classification of dumping sites. This indicates that CNN models are able to learn to recognize the textural, contextual, and color features of dumping locations on high spatial resolution RGB satellite imagery.

Accuracy	0,8268
Precision	0,8376
Recall	0,8372
F-score	0,8374

Test data	Predicted "No Dumping"	Predicted "Dumping"
Actual "No Dumping"	1647	374
Actual "Dumping"	375	1929

Table 8. Performance metrics iteration 9.

5.2 Waste Material Type Classification

For the second classification problem of the project, the waste material classification, the same evaluation metrics are used. In this classification problem, next to the accuracy indicating the performance of the models, the confusion matrixes also gave some helpful insights.

The accuracy of the different iterations has been set out in table 9. This time both the training and testing accuracy are included as there is a significant difference between them and I feel that the comparison between them can lead to more insights. Additionally, batch size and data augmentation techniques are also included in the table as these change between some of the iterations. Furthermore, the training and test accuracy are visually set out over the iterations that implement the Inception V3 model in a graph in figure 11.

What is immediately noticeable from the performance metrics in table 9 is the stark difference between the training and testing accuracy. A theory as to why this could be is that the training and testing dataset are not gathered with the same process, so, logically, there are differences between the training and testing data. An example of this is that the training data is captured from a wide range of areas and terrains, while the testing dataset is limited to Cyprus. Furthermore, the categories of the training dataset are intuitively divided between the two classes of waste, however, the original categories might not accurately represent the same textural and contextual features as the classes they are assigned to. This also accounts partially for the difference between the training and testing data. It can be noticed in figure 11 however, that over the iterations up until iteration 7, the training and test accuracy are getting closer together until finally being equal in iteration 7, which is also the best-performing iteration based on test accuracy and F-score.

Iteration	Model	Epochs	Batch Size	Training Accuracy	Test Accuracy	Data Augmentation
1	Inception V3	10	16	71%	55%	None
2	Inception V3	10	16	67%	50%	None
3	Inception V3	25	16	70%	54%	None
4	Inception V3	25	16	56%	50%	Rotation, Flipping
5	Inception V3	30	16	58%	54%	Rotation, Flipping
6	DenseNet	30	16	58%	52%	Rotation, Flipping
7	Inception V3	50	16	58%	58%	Rotation, Flipping
8	Inception V3	50	16	58%	50%	Rotation, Shifts, Brightness, Shearing, Zooming, Channel shift, Flipping
9	Inception V3	50	32	57%	50%	Rotation, Shifts, Brightness, Shearing, Zooming, Channel shift, Flipping
10	Inception V3	75	16	56%	50%	Rotation, Shifts, Brightness, Shearing, Zooming, Channel shift, Flipping
11	ResNetFPN	50	16	57%	50%	Rotation, Shifts, Brightness, Shearing, Zooming, Channel shift, Flipping

Table 9. Binary Dumping Classification Results.

What can also be noticed from both table 9 and figure 11 is that over the first few iterations, the training accuracy drops. The drop from the first to the second iteration can be explained by the filtering out of the AGEA images from the training dataset, which further reduces the already small dataset. The further drop in training accuracy seems to be at the introduction of data augmentation techniques. These are unexpected results since the data augmentation should somewhat reduce the limitation of a small dataset. The reduction in accuracy from adding data augmentation is countered with an increase in the number of epochs, however, also allowing the test accuracy to match the training accuracy and leading to the best-performing iteration of this complex classification problem. Adding additional data augmentation techniques seems to confuse the models, reducing the test accuracy to a meager 50%. If we take a look at the confusion matrix of one of these models, such as table 10. We can see that the addition of these data augmentations has confused the model to the point where it predicts all of the test images to be in the same class, resulting in 50% accuracy. This suggests that there is a balance to be struck in the utilization of data augmentation techniques to find the best performance. A possible reason for this behavior is that these “more extreme” data augmentation techniques further increase the differences between training and testing data.

Iterations 9 and 10 show that this dataset limitation cannot be overcome by increasing the batch size or the number of epochs. Finally, the implementation of other models such as DenseNet and ResNetFPN does not improve the performance.

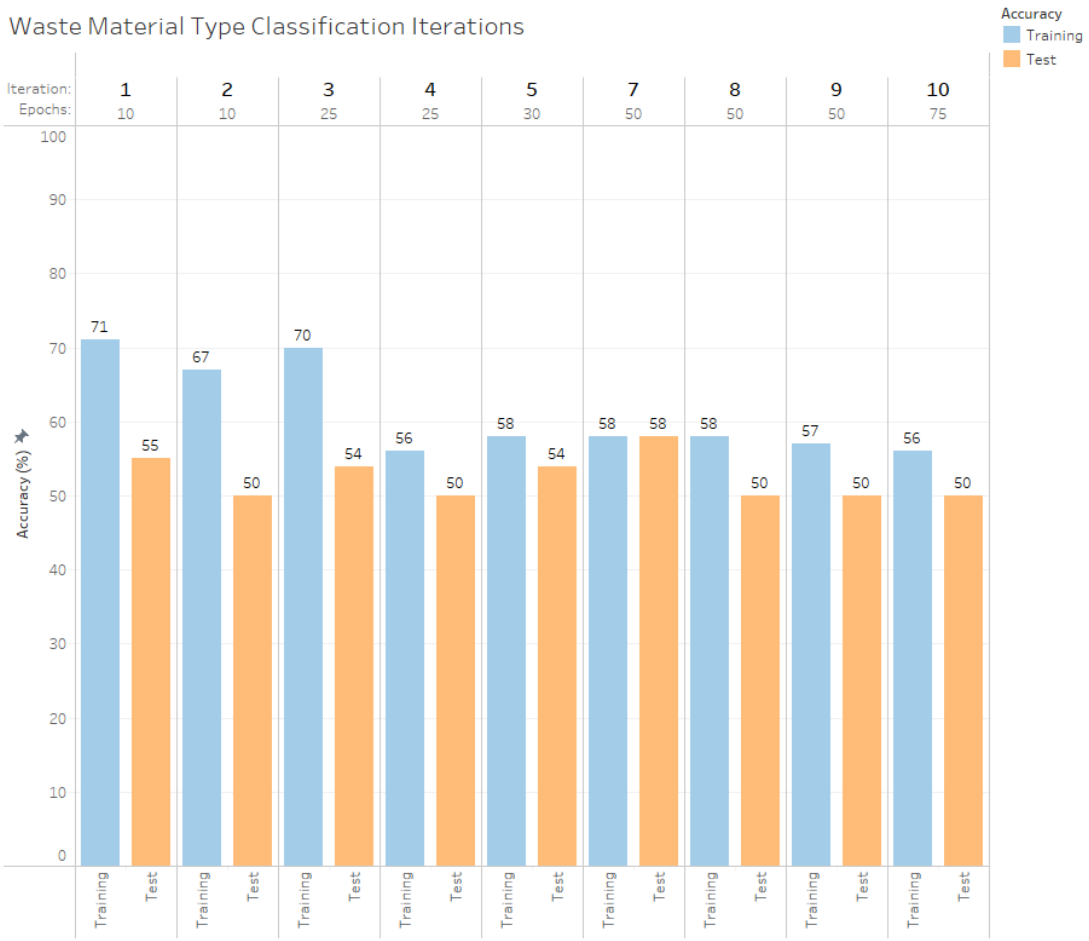


Figure 11. Inception V3 train and test accuracy.

Test data	Predicted "Construction waste"	Predicted "Regular trash"
Actual "Construction waste"	50	0
Actual "Regular trash"	50	0

Table 10. Confusion matrix Inception V3 model iteration 8.

Overall, the performance metrics of iteration 7 which are shown in table 11 indicate that there might be an implementation possible, where a CNN model could learn to distinguish some classes of waste material with this type of satellite imagery. Unfortunately, the approach and methods used in this project do not produce convincing performance metrics. This could be caused by several different factors, the limited dataset being one of them. Possible future directions to this classification problem will be discussed in the discussion & future work chapter.

Accuracy	0,5800
Precision	0,7857
Recall	0,2200
F-score	0,8374

Test data	Predicted “Construction waste”	Predicted “Regular trash”
Actual “Construction waste”	47	3
Actual “Regular trash”	39	11

Table 11. Performance metrics iteration 7.

6 Discussion & Future Work

In this chapter, the limitations of the process and the approach of this graduation project are discussed. Additionally, some possible future directions in this field are set out. The goal of this is to gain a better understanding of this project's place in the research field.

6.1 Limitations

A limitation in the methodology of the project, mainly the second half, is the non-availability of a big dataset, which includes labels for waste material types. This absence of annotated data limited the results achieved within the time limitations and scope of this graduation project. Although, if planned differently the significant difference between training and testing data in the latter half of the project could maybe have been avoided or reduced. Ideally, a new dataset with these types of annotations would be created. However, creating a dataset annotated with such granularity also requires more granular ground truth data and can be costly and time-consuming.

Another oversight in the methodology of the project is the neglect of a potentially useful aspect of the gathered dataset. The time series of 15 images for each recorded dumping location could have been incorporated into the methodology of the project. Potentially, the models could have learned to recognize certain temporal features to better classify certain waste types, such as the construction waste previously discussed. Additionally, using more than one image per time series in the dataset might have diluted the dataset, although the increase in the size of the dataset was very beneficial to the classification performance.

A big limitation of this project was the time constraint. While a project cannot move on eternally, especially a graduation project, I feel like some parts of the project took up a significant part of the allotted time. Namely the processing of the datasets, implementing the models, and waiting for models to finish training. This is of course the nature of this type of project and these are integral parts of the process. However, I feel that with more time, more experimentation could have been done, leading to different results.

Another limitation of the process of the project was the usage of the University of Twente HPC cluster. While the project would not have been possible in a realistic time frame without using an HPC solution, a lot of time was lost trying to get the implementation to work. From learning to write a Slurm script to properly installing all of the required correct versions of libraries and utilities. An additional limitation was the remote nature of the HPC cluster. Especially switching between datasets and troubleshooting the Python scripts took a lot of extra time because of this.

6.2 Future Directions

Since deep learning models such as the CNN models used in this project need a lot of data to be trained effectively, acquiring a big enough dataset can become a limiting factor for projects in this field. Especially for hyper-specific applications such as illegal waste dumping classifications collecting ground truth and image datasets can be a costly and time-consuming process. Unfortunately, not a lot of data exists for this specific application. The limitations of a dataset can be reduced by the application of data augmentation techniques as previously discussed. One powerful data augmentation technique not utilized in this project is synthetic data. By rendering 3D models of the desired subject, such as heaps of waste material, on top of satellite imagery, new data can be generated that imitates real images. These images can then be used to enhance and extend existing datasets. This technique has seen success, such as in works by Padubidri et al. [42] and Martinson et al. [43].

Another possible future research direction in this field could be the usage of multispectral or even hyperspectral satellite imagery. Some materials show more recognizable features in frequency bands outside of the visible spectrum. The utilization of these additional spectral bands can allow deep learning models to better classify the materials that are more recognizable in an expanded spectral range. Both Momeni et al. [15] and Herold et al. [14] found that including some additional bands increases performance in urban ground cover classification. A current limitation of this technology however is that high spatial resolution data in these additional bands is not widely available.

Other types of satellite data might also be useful if combined with high spatial-resolution satellite imagery, such as LiDAR height maps. Additionally, other classified maps could be incorporated, such as groundcover or road network maps. These features add contextual information that could unveil certain features that a deep learning model could learn to recognize.

Finally, aerial photography could be used instead of satellite imagery. The benefit of aerial photography is that the images are usually in an extremely high spatial resolution which can allow CNN models to recognize more features. This could be especially helpful for classifying material or objects with such fine granularity or size as dumped waste. The downsides of aerial photography are that it can be expensive and time-consuming to obtain such data, and there is not yet a lot of coverage. Although, through future advancements in technology, publicly available satellite imagery might reach a fine enough spatial resolution for this purpose. This would make the use of aerial photography unnecessary.

7 Conclusion

The goal of this graduation project is to answer the main research question, namely, how can deep learning be used for the classification of illegally dumped materials from satellite imagery? The motivation behind this research question is that by answering it, we can come closer to a solution for reducing, cleaning up, and recycling waste sites where waste materials have illegally been dumped. This problem should be solved as waste dumping comes with definite negative environmental effects such as pollution and diseases. Additionally, tourism and other economic factors can be negatively impacted. Recycling dumped waste will also bring society closer to a circular economy, which is necessary to continue to live with the luxuries of modern life. Deep learning models combined with satellite imagery have the potential to be part of a solution.

We can answer this research question in a structured manner by answering sub-questions derived from it. The first one being: which deep learning models are structured optimally to classify illegally dumped materials?

Preliminary research gives a hint towards the answer to this question. Existing literature shows that state-of-the-art convolutional neural networks perform very well in similar tasks. In order to see whether these CNN models can be adapted to the application at hand, several different CNN models are first trained to classify between satellite images that show visible waste dumping or no visible waste dumping. After evaluating a few iterations, the Inception V3 CNN model performed the best out of all tested CNN models with an overall accuracy of 83%. This performance suggests that CNN models are capable of learning to recognize the textural, contextual, and color features of waste dumps. Now, the Inception V3 and some other CNN models are trained to classify between two very distinct classes of waste materials, construction waste, and regular trash. Here the Inception V3 model again performs the best out of the tested deep learning model, reaching an overall accuracy of 58%. With the methodology used in this project, the CNN models do not perform well enough at classifying different classes of waste material to confidently say that they are the best deep model structure for the given application. However, there is an indication that with a different approach, CNN models, especially if structured like the Inception V3 model, could possibly be updated to classify illegally dumped waste material.

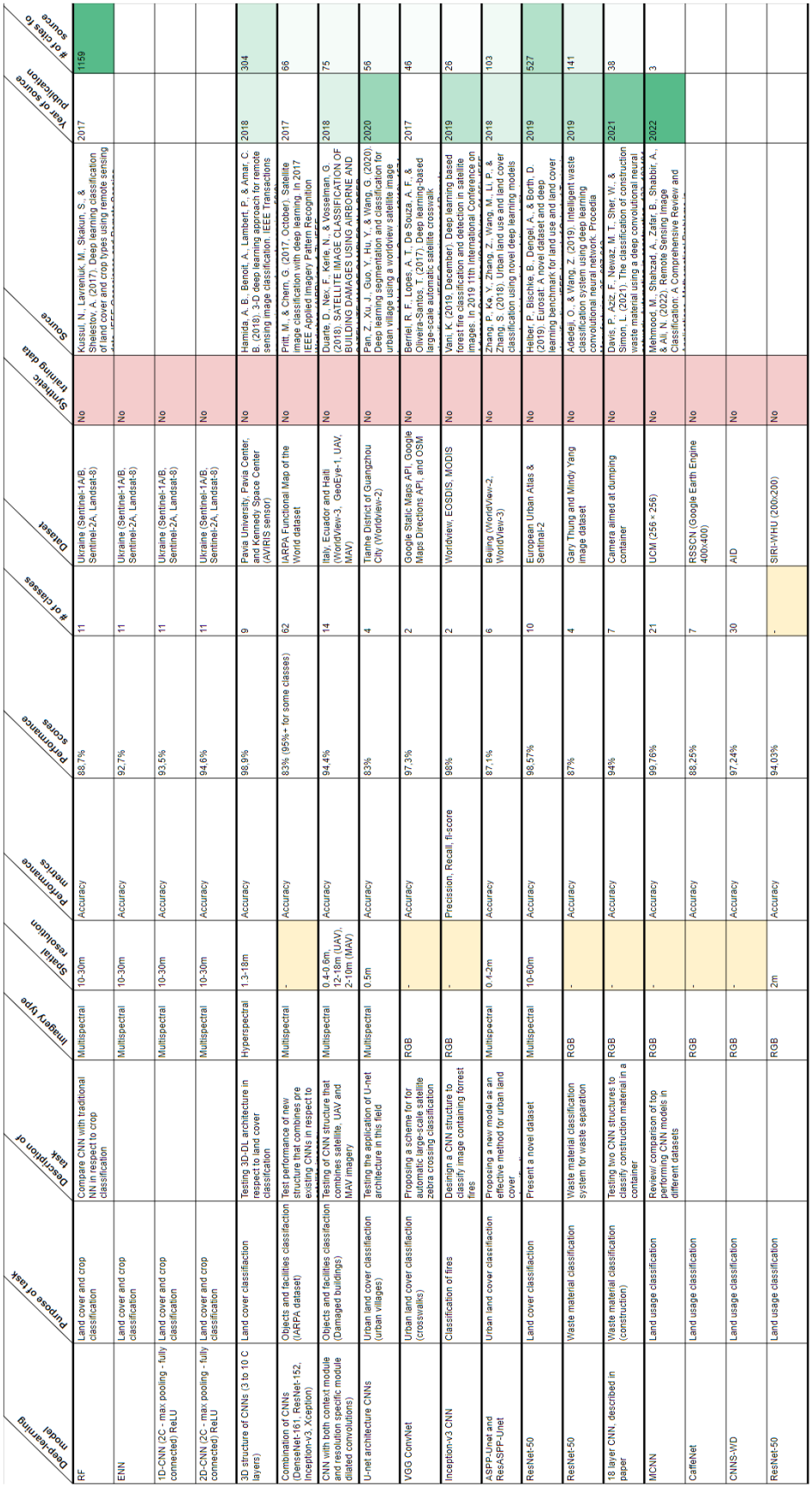
To fully answer the research question another sub-question should also be answered. This second sub-question is: how can such a deep learning model be trained to perform accurately?

preliminary research suggests that the most important part of training a deep learning model successfully is the dataset. In this project, a dataset of over 15.000 high satellite images with a spatial resolution of around 0.30 m per pixel, is manually annotated and additional images are generated. This dataset has images with and without dumping and the images are labeled as such. As this process took quite a long time, there was a chance to see through several iterations how the size of the training dataset affected the performance of the models. As expected, the larger the dataset got, the better the

models performed. Additionally, when the dataset is more balanced, meaning that the number of images in the different classes is similar or smaller classes are given more weight, the performance also increased. Finally, as is also suggested in existing literature, increasing the time, or the number of epochs, that the model is trained for also increases the classification performance. When the classification problem becomes more complex to classify between types of waste material, the granularity of the dataset will need to adapt to this. As data that is annotated with precise categories of waste is rare, the dataset used for training the models was greatly reduced to less than 600 images. This resulted in far worse classification performance. Data augmentation techniques can be utilized to slightly compensate for a smaller dataset, although they should be carefully considered and not overused if the original dataset is already so small. An additional limitation of the low availability of images annotated with high granularity in this project is that a different dataset is used for evaluation, resulting in a big disparity between the training and testing accuracy of the models. With moderate data augmentation usage and increased training time, this disparity is reduced.

Overall, to answer the main research question as a whole, CNN models can perform well with the classification of high spatial resolution satellite imagery. The results of the project suggest that the CNN model is able to be trained to recognize the features of waste dumping with fairly good performance when trained with a decently sized and well-balanced dataset. The CNN models have not performed well with the classification of waste material types with the used approach. This could partially or totally be due to a severely limited dataset as data annotated in sufficient granularity is very limited. Data augmentation techniques can slightly compensate for this if used precisely.

Appendix A: Literature matrix



Deep-learning model	Purpose of task	Description of task	Imagery type	Spatial resolution	Performance metrics	Performance scores	# of classes	Dataset	Synthetic training data	Source	Year of source publication	# of cities in source
RF	Land cover and crop classification	Compare CNN with traditional NN in respect to crop classification	Multispectral	10-30m	Accuracy	88.7%	11	Ukraine (Sentinel-1A/B, Sentinel-2A, Landsat-8)	No	Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep learning classification of land cover and crop types using remote sensing	2017	1159
ENN	Land cover and crop classification		Multispectral	10-30m	Accuracy	92.7%	11	Ukraine (Sentinel-1A/B, Sentinel-2A, Landsat-8)	No			
1D-CNN (2C - max pooling - fully connected) ReLU	Land cover and crop classification		Multispectral	10-30m	Accuracy	93.5%	11	Ukraine (Sentinel-1A/B, Sentinel-2A, Landsat-8)	No			
2D-CNN (2C - max pooling - fully connected) ReLU	Land cover and crop classification		Multispectral	10-30m	Accuracy	94.6%	11	Ukraine (Sentinel-1A/B, Sentinel-2A, Landsat-8)	No			
3D structure of CNNs (3 to 10 C layers)	Land cover classification	Testing 3D DL architecture in respect to land cover classifcation	Hyperspectral	1.3-18m	Accuracy	98.9%	9	Pavia University, Pavia Center, and Kennedy Space Center (AVIRIS sensor)	No	Hamida, A. B., Benoit, A., Lambert, P., & Amar, C. B. (2018). 3-D deep learning approach for remote sensing image classification. IEEE Transactions	2018	304
Combination of CNNs (DenseNet-161, ResNet-152, Inception-v3, Xception)	Objects and facilities classification	Test performance of new structure that combines pre existing CNNs in respect to	Multispectral	-	Accuracy	83% (95%+ for some classes)	62	IARPA Functional Map of the World dataset	No	Pritt, M., & Chern, G. (2017, October). Satellite image classification with deep learning. In 2017 IEEE Applied imagery Pattern Recognition	2017	66
CNN with both context module and resolution specific module dilated convolutions)	Objects and facilities classification (Damaged buildings)	Testing of CNN structure that combines satellite, UAV and MAV imagery	Multispectral	0.4-0.8m, 12-18m (UAV), 2-10m (MAV)	Accuracy	94.4%	14	Italy, Ecuador and Haiti (WorldView-3, GeoEye-1, UAV, MAV)	No	Duarte, D., Nex, F., Kerle, N., & Vosselman, G. (2018). SATELLITE IMAGE CLASSIFICATION OF BUILDING DAMAGES USING AIRBORNE AND	2018	75
U-net architecture CNNs	Urban land cover classification (urban villages)	Testing the application of U-net architecture in this field	Multispectral	0.5m	Accuracy	83%	4	Tianhe District of Guangzhou City (WorldView-2)	No	Pan, Z., Xu, J., Guo, Y., Hu, Y., & Wang, G. (2020). Deep learning segmentation and classification for urban village using a worldview satellite image	2020	56
VGG ConvNet	Urban land cover classification (crosswalks)	Proposing a scheme for for automatic large-scale satellite zebra crossing classification	RGB	-	Accuracy	97.3%	2	Google Static Maps API, Google Maps Directions API, and OSM	No	Berriel, R. F., Lopes, A. T., De Souza, A. F., & Oliveira-Santos, T. (2017). Deep learning-based large-scale automatic satellite crosswalk	2017	46
Inception-v3 CNN	Classification of fires	Desinign a CNN structure to classify image containing forrest fires	RGB	-	Precision, Recall, fi-score	98%	2	Worldview, EOSDIS, MODIS	No	Vani, K. (2019, December). Deep learning based forest fire classification and detection in satellite images. In 2019 11th International Conference on	2019	26
ASPP-Unet and ResASPP-Unet	Urban land cover classification	Proposing a navi model as an effective method for urban land cover	Multispectral	0.4-2m	Accuracy	87.1%	6	Beijing (WorldView-2, WorldView-3)	No	Zhang, P., Ke, Y., Zhang, Z., Wang, M., Li, P., & Zhang, S. (2018). Urban land use and land cover classification using novel deep learning models	2018	103
ResNet-50	Land cover classification	Present a novel dataset	Multispectral	10-60m	Accuracy	98.57%	10	European Urban Atlas & Sentinal-2	No	Helber, P., Bischke, B., Dengel, A., & Borth, D. (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover	2019	527
ResNet-50	Waste material classification	Waste material classification system for waste separation	RGB	-	Accuracy	87%	4	Gary Thung and Mindy Yang image dataset	No	Adedeji, O., & Wang, Z. (2019). Intelligent waste classification system using deep learning convolutional neural network. Procedia	2019	141
18 layer CNN, described in paper	Waste material classification (construction)	Testing two CNN structures to classify construction material in a	RGB	-	Accuracy	94%	7	Camera aimed at dumping container	No	Davis, P., Aziz, F., Newaz, M. T., Sher, W., & Simon, L. (2021). The classification of construction waste material using a deep convolutional neural	2021	38
MCNN	Land usage classification	Review comparison of top performing CNN models in different datasets	RGB	-	Accuracy	99.76%	21	UCM (256 x 256)	No	Mehmood, M., Shahzad, A., Zafar, B., Shabbir, A., & Ali, N. (2022). Remote Sensing Image Classification: A Comprehensive Review and	2022	3
CaffeNet	Land usage classification		RGB	-	Accuracy	88.25%	7	RSSCN (Google Earth Engine 400x400)	No			
CNNS-WD	Land usage classification		RGB	-	Accuracy	97.24%	30	AID	No			
ResNet-50	Land usage classification		RGB	2m	Accuracy	94.03%	-	SIRI-WHU (200x200)	No			

Figure 12. Literature matrix used in preliminary research.

Appendix B: Dataset spitting script

```
import splitfolders as sf

input_folder = "C:/…/Dataset_unsplit"
output = "C:/…/Dataset_split"
sf.ratio(input_folder, output=output, seed=42, ratio=(.8, .2))
```

Figure 13. Python script to perform train/ test split.

Appendix C: Model implementation script

```
#import libraries
import numpy as np
import glob
import tensorflow as tf

#import utilities
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras import Model, layers
from tensorflow.keras.optimizers import RMSprop

#import model
from tensorflow.keras.applications.inception_v3 import InceptionV3

def prepare_dataset(path, label):   #create function for loading dataset
    x_train = []
    y_train = []
    all_images_path = glob.glob(path + '/*.jpg')
    for img_path in all_images_path:    #load all images in given path
        img = load_img(img_path, target_size=(512, 512))
        img = img_to_array(img)
        img = img / 255.0
        x_train.append(img)
        y_train.append(label)
    return np.array(x_train), np.array(y_train)

#load classes and label training data with function
trainX_no_dumping, trainY_no_dumping =
prepare_dataset("dataset/train/No_Dumping/", 0)
trainX_dumping, trainY_dumping = prepare_dataset("dataset/train/Dumping/",
1)

#concatenate training datasets
x_train = np.concatenate((trainX_no_dumping, trainX_dumping), axis=0)
y_train = np.concatenate((trainY_no_dumping, trainY_dumping), axis=0)

#load classes and label test data with function
testX_no_dumping, testY_no_dumping =
prepare_dataset("dataset/val/No_Dumping/", 0)
testX_dumping, testY_dumping = prepare_dataset("dataset/val/Dumping/", 1)

#concatenate test datasets
x_test = np.concatenate((testX_no_dumping, testX_dumping), axis=0)
y_test = np.concatenate((testY_no_dumping, testY_dumping), axis=0)

#load model
pretrained_model = InceptionV3(input_shape=(512, 512, 3),
                               include_top=False,
                               weights='imagenet')
for layer in pretrained_model.layers:
    layer.trainable = False

#replace last layers
last_layer = pretrained_model.get_layer('mixed10')
last_output = last_layer.output

x = layers.Flatten()(last_output)             #add flatten layer
x = layers.Dense(1024, activation='relu')(x) #add densly connected layer
x = layers.Dropout(0.2)(x)                    #add dropout layer
```

```
x = layers.Dense(1, activation='sigmoid')(x) #final classification layer
model = Model(pretrained_model.input, x)

#compile model
model.compile(optimizer=RMSprop(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=['acc',           #select evaluation metrics
                       tf.keras.metrics.Precision(),
                       tf.keras.metrics.Recall(),
                       tf.keras.metrics.TruePositives(),
                       tf.keras.metrics.TrueNegatives(),
                       tf.keras.metrics.FalsePositives(),
                       tf.keras.metrics.FalseNegatives()]
              )

#train and evaluate model
model.fit(x_train, y_train,     #select model parameters
          batch_size=16,
          epochs=1,
          class_weight={0: 1, 1: 1},
          validation_data=(x_test, y_test),
          shuffle=True)
```

Figure 14. Python script implementing CNN model.

Appendix D: Slurm job script

```
#!/bin/bash
#SBATCH -J ClassifyJob          # job name
#SBATCH -c 16                   # number of cores
#SBATCH --gres=gpu:2            # number of gpus
#SBATCH -p main                 # partition
#SBATCH --constraint=avx2       # additional request
#SBATCH --constraint=titan-x    # additional request
#SBATCH -o outfile.out          # send stdout to outfile
#SBATCH -e errfile.err          # send stderr to errfile
#SBATCH --time=2-00:00          # maximum time limit

# display node name
echo "nodename :"
hostname

# check if gpu are assigned, if not create empty list
if [ -z "$CUDA_VISIBLE_DEVICES" ]; then
  export CUDA_VISIBLE_DEVICES=""
fi
# display which gpu's we are allowed to use
echo "CUDA_VISIBLE_DEVICES = '"$CUDA_VISIBLE_DEVICES"'"

# load nvidia cuda toolkit
module load nvidia/cuda-11.2

# run task
echo "Starting script: "
python3 Script.py
```

Figure 15. Slurm job script.

Appendix E: Updated model implementation script

```
#import libraries
import numpy as np
import glob
import tensorflow as tf

#import utilities
from utils import dataloader
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras import Model, layers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tqdm import tqdm

#variables
train_list ="training.json"
test_list ="testing.json"
dataset_root ="Cropped"
source_list = ["WV3", "GE"]
no_dumping = []
emptyList = []
dumping_const = []
dumping_trash = []

#create lists of construction waste images using utility
data_multilabel_construction = dataloader.ClassificationDataset(train_list,
dataset_root=dataset_root, binary=False,
                                                                classes=[1,
12, 13, 14, 17, 19])
data_multilabel_construction += dataloader.ClassificationDataset(test_list,
dataset_root=dataset_root, binary=False,
                                                                 classes=[1,
12, 13, 14, 17, 19])

#check if images are from google earth engine or worldview-3
for img in tqdm(data_multilabel_construction):
    if img["source"] in source_list:
        dumping_const.append(img["name"])

#create lists of regular trash images using utility
data_multilabel_trash = dataloader.ClassificationDataset(train_list,
dataset_root=dataset_root, binary=False,
                                                         classes=[4, 5, 8,
9, 18, 22])
data_multilabel_trash += dataloader.ClassificationDataset(test_list,
dataset_root=dataset_root, binary=False,
                                                          classes=[4, 5, 8,
9, 18, 22])

#check if images are from google earth engine or worldview-3
for img in tqdm(data_multilabel_trash):
    if img["source"] in source_list:
        dumping_trash.append(img["name"])

#print the amount of images in training data
print("Amount of images construction", len(dumping_const))
print("Amount of images trash", len(dumping_trash))
```

```
def prepare_dataset(path, label, AerielDataset, list_of_labels):
    x_train = []
    y_train = []
    all_images_path = glob.glob(path + '/*.jpg')
    if AerielDataset:                          #if from Aerialwaste dataset
        for img_path in all_images_path:       #load all images in given path
            split_path = img_path.split("/")
            imagelabel = split_path[-1].split(".")
            if (imagelabel[0] + '.png') in list_of_labels:
                img = load_img(img_path, target_size=(512, 512))
                img = img_to_array(img)
                img = img / 255.0
                x_train.append(img)
                y_train.append(label)
    else:                                      #if from regular dataset
        for img_path in all_images_path:       #load all images in given path
                img = load_img(img_path, target_size=(512, 512))
                img = img_to_array(img)
                img = img / 255.0
                x_train.append(img)
                y_train.append(label)
    return np.array(x_train), np.array(y_train)

#load classes and label training data with function
trainX_const, trainY_const = prepare_dataset("Croppedjpg/", 0, 1,
dumping_const)
trainX_trash, trainY_trash = prepare_dataset("Croppedjpg/", 1, 1,
dumping_trash)

#concatenate training datasets
x_train = np.concatenate((trainX_const, trainX_trash), axis=0)
y_train = np.concatenate((trainY_const, trainY_trash), axis=0)

#load classes and label test data with function
testX_const, testY_const = prepare_dataset("Test_Classes/Class_Construct/",
0, 0, emptyList)
testX_trash, testY_trash = prepare_dataset("Test_Classes/Class_Trash/", 1,
0, emptyList)

#concatenate test datasets
x_test = np.concatenate((testX_const, testX_trash), axis=0)
y_test = np.concatenate((testY_const, testY_trash), axis=0)

#load model
pretrained_model = InceptionV3(input_shape=(512, 512, 3),
                               include_top=False,
                               weights='imagenet')
for layer in pretrained_model.layers:
    layer.trainable = False

#replace last layers
last_layer = pretrained_model.get_layer('mixed10')
last_output = last_layer.output

x = layers.Flatten()(last_output)              #add flatten layer
x = layers.Dense(1024, activation='relu')(x) #add densly connected layer
x = layers.Dropout(0.2)(x)                     #add dropout layer
x = layers.Dense(1, activation='sigmoid')(x) #final classification layer
model = Model(pretrained_model.input, x)
```

```
#compile model
model.compile(optimizer=RMSprop(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=['acc',           #select evaluation metrics
                       tf.keras.metrics.Precision(),
                       tf.keras.metrics.Recall(),
                       tf.keras.metrics.TruePositives(),
                       tf.keras.metrics.TrueNegatives(),
                       tf.keras.metrics.FalsePositives(),
                       tf.keras.metrics.FalseNegatives()],
              run_eagerly=True)

#create data generator
datagen = ImageDataGenerator( #select augementation techniques
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=90,
    width_shift_range=10.0,
    height_shift_range=10.0,
    brightness_range=[0.2, 1.2],
    shear_range=0.5,
    zoom_range=0.2,
    channel_shift_range=20.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=True,
    vertical_flip=True,
    rescale=None,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    interpolation_order=1,
    dtype=None)

#fit and run data generator
datagen.fit(x_train)
generator_datagen = datagen.flow(x_train, y_train, batch_size=32,
shuffle=True)

model.fit_generator(generator_datagen, #select model parameters
                    steps_per_epoch=len(x_train) // 32,
                    epochs=50,
                    class_weight={0: 1, 1: 1.19},
                    validation_data=(x_test, y_test))
```

Figure 16. Updated Python script implementing CNN model.

References

[1] "Waste generated by households by year and waste category - Products Datasets - Eurostat." https://ec.europa.eu/eurostat/web/products-datasets/-/ten00110 (accessed Oct. 24, 2022).

[2] D. H. F. da Paz, K. P. V. Lafayette, M. J. de O. Holanda, M. do C. M. Sobral, and L. A. R. de C. Costa, "Assessment of environmental impact risks arising from the illegal dumping of construction waste in Brazil," *Environ Dev Sustain*, vol. 22, no. 3, pp. 2289–2304, Mar. 2020, doi: 10.1007/S10668-018-0289-6/TABLES/8.

[3] S. K. Singh, P. Chokhandre, P. S. Salve, and R. Rajak, "Open dumping site and health risks to proximate communities in Mumbai, India: A cross-sectional case-comparison study," *Clin Epidemiol Glob Health*, vol. 9, pp. 34–40, Jan. 2021, doi: 10.1016/J.CEGH.2020.06.008.

[4] N. Kobza and A. Schuster, "Building a responsible Europe - the value of circular economy," *IFAC-PapersOnLine*, vol. 49, no. 29, pp. 111–116, Jan. 2016, doi: 10.1016/J.IFACOL.2016.11.067.

[5] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, May 2017, doi: 10.1109/LGRS.2017.2681128.

[6] L. Zhu *et al.*, "A Review: Remote Sensing Sensors," *Multi-purposeful Application of Geospatial Data*, Dec. 2017, doi: 10.5772/INTECHOPEN.71049.

[7] S. Kumar and V. Bhagat, "Remote Sensing Satellites for Land Applications: A Review," *Remote Sensing of Land*, vol. 2, no. 2, pp. 96–104, Jul. 2019, doi: 10.21523/gcj1.18020203.

[8] P. Liu, "A survey of remote-sensing big data," *Frontiers in Environmental Science*, vol. 3, no. JUN. Frontiers Media S.A., Jun. 17, 2015. doi: 10.3389/fenvs.2015.00045.

[9] J. K. Krueger, D. Selva, M. W. Smith, and J. Keesee, "Spacecraft and constellation design for a continuous responsive imaging system in space," in *AIAA Space 2009 Conference and Exposition*, 2009. doi: 10.2514/6.2009-6773.

[10] R. Momeni, P. Aplin, and D. S. Boyd, "Mapping complex urban land cover from spaceborne imagery: The influence of spatial resolution, spectral band set and classification approach," *Remote Sens (Basel)*, vol. 8, no. 2, 2016, doi: 10.3390/rs8020088.

[11] D. Chen, D. A. Stow, and P. Gong, "Examining the effect of spatial resolution and texture window size on classification accuracy: An urban environment case," *Int J Remote Sens*, vol. 25, no. 11, pp. 2177–2192, Jun. 2004, doi: 10.1080/01431160310001618464.

[12] X. Yu *et al.*, “Examining the roles of spectral, spatial, and topographic features in improving land-cover and forest classifications in a subtropical region,” *Remote Sens (Basel)*, vol. 12, no. 18, Sep. 2020, doi: 10.3390/RS12182907.

[13] U. Bradter, J. O’Connell, W. E. Kunin, C. W. H. Boffey, R. J. Ellis, and T. G. Benton, “Classifying grass-dominated habitats from remotely sensed data: The influence of spectral resolution, acquisition time and the vegetation classification system on accuracy and thematic resolution,” *Science of the Total Environment*, vol. 711, Apr. 2020, doi: 10.1016/j.scitotenv.2019.134584.

[14] M. Herold, M. E. Gardner, and D. A. Roberts, “Spectral resolution requirements for mapping urban areas,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 9 PART I, pp. 1907–1919, Sep. 2003, doi: 10.1109/TGRS.2003.815238.

[15] R. Momeni, P. Aplin, and D. S. Boyd, “Mapping Complex Urban Land Cover from Spaceborne Imagery: The Influence of Spatial Resolution, Spectral Band Set and Classification Approach,” *Remote Sensing 2016, Vol. 8, Page 88*, vol. 8, no. 2, p. 88, Jan. 2016, doi: 10.3390/RS8020088.

[16] F. Fang, B. E. McNeil, T. A. Warner, and A. E. Maxwell, “Combining high spatial resolution multi-temporal satellite data with leaf-on LiDAR to enhance tree species discrimination at the crown level,” *Int J Remote Sens*, vol. 39, no. 23, pp. 9054–9072, Dec. 2018, doi: 10.1080/01431161.2018.1504343.

[17] A. Orynbaikyzy, U. Gessner, and C. Conrad, “Crop type classification using a combination of optical and radar remote sensing data: a review,” *Int J Remote Sens*, vol. 40, no. 17, pp. 6553–6595, Sep. 2019, doi: 10.1080/01431161.2019.1569791/SUPPL_FILE/TRES_A_1569791_SM4489.ZIP.

[18] J. Song, S. Gao, Y. Zhu, and C. Ma, “Big Earth Data A survey of remote sensing image classification based on CNNs A survey of remote sensing image classification based on CNNs,” 2019, doi: 10.1080/20964471.2019.1657720.

[19] J. Portilla, “Complete Guide to TensorFlow for Deep Learning with Python,” *Udemy Course*, Apr. 2020. https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/ (accessed Nov. 06, 2022).

[20] K. Hara, D. Saito, and H. Shouno, “Analysis of function of rectified linear unit used in deep learning,” *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-September, Sep. 2015, doi: 10.1109/IJCNN.2015.7280578.

[21] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout Networks,” 2013.

[22] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-Margin Softmax Loss for Convolutional Neural Networks,” Dec. 2016, doi: 10.48550/arxiv.1612.02295.

[23] 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. 2278–2323, 1998, doi: 10.1109/5.726791.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun ACM*, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[25] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Sep. 2014, doi: 10.48550/arxiv.1409.1556.

[26] C. Szegedy *et al.*, “Going Deeper With Convolutions.” pp. 1–9, 2015.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, Dec. 2016, doi: 10.1109/CVPR.2016.90.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision.” pp. 2818–2826, 2016.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[30] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks.” pp. 4700–4708, 2017. Accessed: Nov. 07, 2022. [Online]. Available: https://github.com/liuzhuang13/DenseNet.

[31] D. Duarte, F. Nex, N. Kerle, and G. Vosselman, “SATELLITE IMAGE CLASSIFICATION OF BUILDING DAMAGES USING AIRBORNE AND SATELLITE IMAGE SAMPLES IN A DEEP LEARNING APPROACH,” 2018, doi: 10.5194/isprs-annals-IV-2-89-2018.

[32] A. ben Hamida, A. Benoit, P. Lambert, and C. ben Amar, “3-D deep learning approach for remote sensing image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 8, pp. 4420–4434, Aug. 2018, doi: 10.1109/TGRS.2018.2818945.

[33] M. Xia, G. Cao, G. Wang, and Y. Shang, “Remote sensing image classification based on deep learning and conditional random fields,” *Journal of Image and Graphics*, vol. 22, no. 9, pp. 1289–1301, 2017.

[34] M. Pritt and G. Chern, “Satellite image classification with deep learning,” *Proceedings - Applied Imagery Pattern Recognition Workshop*, vol. 2017-October, Sep. 2018, doi: 10.1109/AIPR.2017.8457969.

[35] Y. Zhong, F. Fei, and L. Zhang, “Large patch convolutional neural networks for the scene classification of high spatial resolution imagery,” *https://doi.org/10.1117/1.JRS.10.025006*, vol. 10, no. 2, p. 025006, Apr. 2016, doi: 10.1117/1.JRS.10.025006.

[36] Z. Pan, J. Xu, Y. Guo, Y. Hu, and G. Wang, “Deep learning segmentation and classification for urban village using a worldview satellite image based on U-net,” *Remote Sens (Basel)*, vol. 12, no. 10, May 2020, doi: 10.3390/rs12101574.

[37] H. Li, K. Fu, G. Xu, X. Zheng, W. Ren, and X. Sun, “Remote Sensing Letters Scene classification in remote sensing images using a two-stage neural network ensemble model Scene classification in remote sensing images using a two-stage neural network ensemble model,” 2017, doi: 10.1080/2150704X.2017.1302104.

[38] P. Zhang, Y. Ke, Z. Zhang, M. Wang, P. Li, and S. Zhang, “Urban Land Use and Land Cover Classification Using Novel Deep Learning Models Based on High Spatial Resolution Satellite Imagery,” *Sensors 2018, Vol. 18, Page 3717*, vol. 18, no. 11, p. 3717, Nov. 2018, doi: 10.3390/S18113717.

[39] O. Adedeji and Z. Wang, “Intelligent Waste Classification System Using Deep Learning Convolutional Neural Network,” *Procedia Manuf*, vol. 35, pp. 607–612, Jan. 2019, doi: 10.1016/J.PROMFG.2019.05.086.

[40] B. Yu, J. Dongmin, P. Jiajun, Z. Ning, and B. Yu, “Application of an Improved ELU Convolution Neural Network in the SAR Image Ship Detection,” *Bulletin of Surveying and Mapping*, vol. 0, no. 1, p. 125, Jan. 2018, doi: 10.13474/J.CNKI.11-2246.2018.0024.

[41] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, “Towards better exploiting convolutional neural networks for remote sensing scene classification,” *Pattern Recognit*, vol. 61, pp. 539–556, Jan. 2017, doi: 10.1016/j.patcog.2016.07.001.

[42] C. Padubidri, A. Kamilaris, and S. Karatsiolis, “Accurate Detection of Illegal Dumping Sites Using High Resolution Aerial Photography and Deep Learning,” in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated*

Events, PerCom Workshops 2022, 2022, pp. 451–456. doi: 10.1109/PerComWorkshops53856.2022.9767451.

[43] E. Martinson, B. Furlong, and A. Gillies, “Training Rare Object Detection in Satellite Imagery With Synthetic GAN Images.” pp. 2769–2776, 2021.

[44] A. H. Mader and W. Eggink, “A Design Process for Creative Technology,” in *Proceedings of the 16th International conference on Engineering and Product Design, E&PDE 2014*, Sep. 2014, pp. 568–573.

[45] R. N. Torres and P. Fraternali, “Learning to Identify Illegal Landfills through Scene Classification in Aerial Images,” *Remote Sensing 2021, Vol. 13, Page 4520*, vol. 13, no. 22, p. 4520, Nov. 2021, doi: 10.3390/RS13224520.