Reconmatic

Automated Solutions for Sustainable and Circular Construction and Demolition Waste Management

Czech Technical University in Prague

D4.1 AI-based CDW classification software utilizing low-cost sensors' inputs

Contract №: 101058580

June, 2024

Funded by
the European Union

This project is funded by the European Union under Grant Agreement No. 101058580.

Document Control Sheet

Project	Automated solutions for sustainable and circular construction and demolition waste management
Call identifier	HORIZON-CL4-2021-TWIN-TRANSITION-01
Grant Agreement №	101058580
Coordinator	CESKE VYSOKE UCENI TECHNICKE V PRAZE Czech Technical University in Prague
Work package	4. Automated off-site CDW management and treatment
Work package leader	České učení technické v Praze (CVUT) (Czech Technical University in Prague)
Related tasks	4.2
Deliverable title	Deliverable 4.1 – AI-based CDW classification software utilizing low-cost sensors' inputs
Deliverable nature	Software
Dissemination level	PU
Lead Beneficiary	CVUT
Contributing partners	
Authors	Václav Nežerka, CVUT Tomáš Zbíral, CVUT
Reviewer(s)	Jan Trejbal, CVUT (internal) David Šilhánek, CVUT (external) Jan Valentin, CVUT (internal) Stanislav Vítek, CVUT (internal)
Version	1.3 (final, reviewed)
Total number of pages	43
Due date	June 30, 2024
Submission date	June 27, 2024

Contents

List of Figures **vi**

List of Tables **viii**

1 Introduction **1**

2 Dataset **3**

3 Convolution versus extraction of selected features **5**

3.1 Metrics 5

3.1.1 Mean intensity 6

3.1.2 Mean intensity of red color 7

3.1.3 Shannon’s entropy 8

3.1.4 Mean intensity gradient 8

3.2 Classifiers 9

3.2.1 Gradient boosting 10

3.2.2 Multi-layer perception 10

3.2.3 Convolutional neural network 11

3.3 Model Evaluation Metrics 12

3.4 Results and discussion 14

3.5 Application procedure 17

4 Deep neural networks for segmentation and classification **21**

4.1 Data preparation 21

4.1.1 Data cleaning 21

4.1.2 Ground truth masks 22

4.1.3 Data augmentation 23

4.1.4 Data splitting 24

4.2 Segmentation 24
4.2.1 U-Net architecture 24
4.2.2 Segmentation accuracy metrics 26
4.2.3 U-Net model 27
4.3 Classification 30
4.3.1 Model architecture 31
4.3.2 Data preparation for classification 33
4.3.3 Performance of the ResNet model 33

References **37**

Executive Summary

Proper sorting of construction and demolition waste (CDW) fragments is essential for its further valorization. In this document, we demonstrate the potential of machine-learning models for the recognition and classification of CDW fragments using computer vision-based algorithms. The approach was tested on four types of CDW material fragments commonly found in mixed debris from demolition sites: aerated autoclaved concrete (AAC), asphalt mix conglomerates (reclaimed asphalt or reclaimed asphalt pavement), ceramics (roof tiles and bricks), and concrete fragments. For that purpose, we examined:

1. Three basic machine-learning classification models, gradient boosting (GB), multi-layer perception (MLP), and convolutional neural network (CNN)
2. Advanced deep-learning models for segmentation (U-net) and classification (ResNet).

The links to image datasets, computer codes, and pre-trained models used in this study are open and links are provided in this document. We believe that the findings can promote the developments in robotics-assisted sorting of CDW fragments, enabling its efficient use in the production of new materials and products and reduction of the environmental burden associated with CDW disposal.

Results to date are promising for future research. We consider it crucial for further development to collect large, diverse datasets. For this purpose, we have developed a conveyor belt system with the capability of mounting various measurement instruments, facilitating the collection of comprehensive datasets that mirror the conditions of a real sorting yard. While visual data have proven effective, the incorporation of additional modalities such as depth measurements, ultrasound properties, or other electromagnetic spectrum intervals could further enhance accuracy, thus accelerating and refining the sorting process. In terms of machine learning, numerous innovative approaches remain to be explored. These include the integration of decision tree models with deep neural networks and averaging the outcomes from the same model trained on multiple random seeds to maximize accuracy. Such advancements could significantly advance the precision and efficiency of CDW sorting systems.

Source codes

The suite of scripts developed during the project is open-source and available online:

1. Codeocean capsule contains scripts for feature extraction and classification of image subsets as described by Nežerka et al. (2024).
2. Github repository contains scripts for segmentation using U-net and classification using ResNet models.

Document name:	AI-based CDW classification software utilizing low-cost sensors' inputs								
Reference:	D4.1	Dissemination:	PU	Version:	1.3	Status:	Final	Page:	v of 43

List of Figures

2.1 The site for collecting images, a CDW collection and sorting yard near Kladno, Czech Republic. 4

2.2 Examples of image datasets for the examined CDW materials. 4

3.1 Manual extraction of 200×200 px regions (image subsets) used for training and testing of selected classifiers. 6

3.2 Visualization of the image subset characteristics for individual materials (classes) as pairwise scatter plots; marginal distributions of each feature for each class are plotted on the diagonal. 7

3.3 Local coordinates (i, j) for a subset of pixels (right) arbitrarily located within an image of a CDW fragment (left). 8

3.4 Training and testing accuracy as a function of epoch recorded during CNN training. 13

3.5 Confusion matrices for different classifiers and comparison of their performance with manual classification done by five experts on building materials from the FCE CTU in Prague. 16

3.6 Speed (left) and accuracy (right) reached by individual classifiers on the validation (testing) datasets for different sizes of image subsets that were extracted by cropping the redundant portion of the images. 17

3.7 Localization of whole CDW fragments and their classification based on texture recognition using different classifiers; the size of image subsets 135×135 px. 18

3.8 A typical misclassification of AAC fragments by CNN during a comprehensive validation of the classification algorithms; size of image subsets 200×200 px with a 70 px overlap. 19

4.1 Examples of image datasets for the examined CDW (Nežerka et al., 2024). 22

4.2 Images with corresponding masks for each material. 22

4.3 U-net architecture (example for 32×32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations (Ronneberger et al., 2015). . . 26
4.4 Training and validation loss as a function of epoch reported during the training of the U-Net model. 28
4.5 Training and validation IoU as a function of epoch reported during the training of the U-Net model. 28
4.6 Training and validation $F1$ as a function of epoch reported during the training of the U-Net model. 29
4.7 Comparison of ground truth and predicted mask. The blue overlay displays true negatives, the red overlay displays false positives, the green overlay displays false negatives, and true positives are transparent. . . . 29
4.8 Input data for ResNet model. 33
4.9 Training and validation loss as a function of epoch reported during the training of the ResNet model. 34
4.10 Training and validation accuracy as a function of epoch reported during the training of the ResNet model. 34
4.11 Predicted probabilities for correct and incorrect predictions for AAC. . . 35
4.12 Confusion matrix for model in epoch 61. 36

List of Tables

3.1 Summary of extracted 200×200 px image subsets used for testing and training of selected classifiers. 5
3.2 Summary of hyperparameters for the GB classifier implemented in Scikit-Learn v.1.1.3 (*ensemble.GradientBoostingClassifier* model class). . . . 10
3.3 Summary of hyperparameters for the MLP classifier implemented in Scikit-Learn v.1.1.3 (*neural_network.MLPClassifier* model class). 11
3.4 Architecture of the CNN models; the individual layers were implemented in the Tensorflow Keras v.2.10.0 Python package, the *layers* class. . . . 13
3.5 Accuracy of different classifiers when recognizing whole CDW fragments by classifying several ($>$4) 200×200 image subsets with a 70 px overlap (Figure 3.8). 17
3.6 Comparison of the current study with previous significant works focused on machine-learning-based recognition of construction materials in terms of model performance, data type, and dataset size. 19
4.1 Mean IoU based on material calculated for the test data. 27
4.2 Accuracy comparison for models introduced in previous study by Nežerka et al. (2024) and ResNet model. 35

1 Introduction

In order to pursue sustainable development, it is imperative to manage waste in a prudent and cost-effective manner and adopt the principles of circular economy (Joensuu et al., 2020; Oluleye et al., 2022). Following this direction, the European Parliament and the Commission issued Directive No 98/2008 which required the EU member states to increase the overall recycling of waste to at least 70% by weight from 2020. Even though the rate of CDW recycling in the EU is almost constant, at about 90% on average[1], the lion's share is downcycled. At the global scale, rapidly developing countries, such as China with 2 bn tons/year, are even bigger CDW producers than all the EU states combined (Zheng et al., 2017).

The most commonly recycled CDW materials, besides soils, are concrete and ceramics. These waste materials are most often not recycled, but rather utilized directly for embankments, backfills, fillings, or beddings under foundation slabs or pavings; such an approach leads to deterioration of the material value. Less frequently, the recycled fragments are used as aggregates in the production of new concrete mixes or the finest fractions as micro-fillers (Hlůžek et al., 2020; Prošek et al., 2020; Valentin et al., 2021; Nežerka et al., 2023). The major limiting factor in the crushed CDW valorization in applications such as concrete manufacturing is improper sorting (Hoong et al., 2020). Su (2020) carried out a multi-agent evolutionary game study and concluded that research into CDW classification holds the greatest potential to promote CDW recycling and reuse. Davis et al. (2021) pointed out that the automatic classification of CDW materials would significantly reduce the costs associated with sorting.

At the pre-sorting stage, methods exploiting gravitational, magnetic, inertial, electrostatic, or buoyancy forces are very efficient in separating specific types of materials from a heterogeneous CDW mix (Gundupalli et al., 2017; Vincent et al., 2022). Leveraging big data in CDW management offers promising advancements. Yuan et al. (2021) utilized a dataset of 4.27 million truckloads of construction waste to estimate waste composition based on bulk density. Such techniques can significantly refine sorting processes and promote sustainable resource utilization.

Despite recent progress in advanced methods based on research into the development of various sensors (image, spectroscopic, spectral, UV sensitive, etc.) (Gundupalli et al., 2017; Lu and Chen, 2022), sorting of the remaining fragments is at the industrial scale most commonly accomplished manually and cannot be done properly due to their similarity. Therefore, it is desirable to replace manual sorting with robotic

[1] https://ec.europa.eu/eurostat/databrowser/view/cei_wm040/default/table

vision-based technologies such as RGB cameras, hyperspectral imaging, or X-ray sensors assisted with machine learning. This approach has been first employed for the purpose of municipal waste separation (Özkan et al., 2015; Wang et al., 2019b; Liang and Gu, 2021; Lu and Chen, 2022) and the extensive development led to the sorting accuracy exceeding 90% (Yang et al., 2021).

The robotic vision-based technology has also started to find its way into the CDW sorting (Wang et al., 2019a, 2020). However, automatic CDW recognition encountered its limitations in terms of accuracy and boundary identification. The latter issue was addressed by Dong et al. (2022), who proposed a boundary-aware model with the ability to distinguish and segment individual materials within structural debris. CNNs are specialized for image recognition, leveraging their ability to identify hierarchical patterns in visual data. Their design enables them to dissect images into components, enhancing classification accuracy, especially in intricate tasks like CDW sorting. For instance, Xiao et al. (2020) utilized CNNs to effectively classify different CDW materials, underscoring the potential of this approach in the domain. They classified different CDW materials (wood, brick, rubber, rock, concrete) with an accuracy exceeding 80%. Ku et al. (2020) built a robotic line that automatically recognized and classified the basic materials within CDW using hyperspectral and 3D cameras with an accuracy of about 90%. Machine-learning classification was also employed by Lin et al. (2022), who recognized visually different CDW fragments and achieved an accuracy ranging between 75 and 80%. The closest to our goal is the study by Hoong et al. (2020), who employed neural networks for the classification of recycled aggregates. They constructed a library of 36,000 images of individual aggregate grains and their model achieved accuracies of up to 97%.

While previous studies have employed CNN-based models for CDW classification, our research distinguishes itself in two primary ways. Firstly, we focus on the efficient extraction of features describing the textures captured using ordinary RGB cameras, a method not extensively explored in prior work. Secondly, we provide a comprehensive comparison between CNN and other machine-learning models, specifically GB models and MLP, showcasing the efficacy of feature extraction in enhancing both speed and accuracy. This paper presents a unique approach to CDW fragment recognition, emphasizing the power of feature extraction. In the following part, we focused on the use of advanced models to accomplish fast segmentation and accurate classification using U-net and ResNet models, respectively. We provide extensive datasets, computer codes, and pre-trained models, ensuring our methodology is transparent, reproducible, and can be built upon by other researchers or industry stakeholders.

2 Dataset

The capabilities and limitations of the selected feature extraction methods and machine-learning models are demonstrated on four types of CDW fragments. These were chosen because they are the most common fragments found in mixed debris from demolition sites in the Czech Republic: light-colored aerated autoclaved concrete (AAC), asphalt conglomerates, ceramics (roof tiles and bricks), and concrete. These materials not only represent a significant portion of the total waste but also pose a challenge in terms of their similarity, making their accurate classification crucial for efficient recycling and waste management.

The 1920×1280 px images of ∼30–250mm fragments were taken from a distance of about 70 cm using a handheld digital single-lens reflex camera (Canon EOS 70D with a Canon zoom lens EF-S 17-85 IS USM) in a CDW collection and sorting yard near Kladno, Czech Republic (Figure 2.1). The images were captured in a shade to minimize variations in illumination and to ensure consistent image quality. Importantly, the CDW fragments were used in their natural state from the yard, without any pre-sorting or cleaning, reflecting the real-world conditions of such waste. In a potential industrial deployment, techniques like air-flow cleaning could be introduced on conveyor belts to minimize dirt and dust, enhancing the image clarity. The fragments were placed on the ground while taking the images, or directly on the CDW piles.

Unlike clean structural elements, whose classification has been tackled in other studies (Zhu and Brilakis, 2010; Son et al., 2012; Dimitrov and Golparvar-Fard, 2014; Han and Golparvar-Fard, 2015; Braun and Borrmann, 2019; Mahami et al., 2020), recognition of CDW fragments is a more challenging task as their surface can be contaminated with dust and residues of other materials. Randomly selected samples of CDW fragments are presented in Figure 2.2, showing similar textures, especially in the case of AAC and concrete. The complete image datasets used for training of machine-learning classifiers and validation are open and provided as supplementary material (Nežerka et al., 2023).

Note that for ceramics, the dominant feature in this category is the red color (including orange and pink). Although not currently addressed, if models were trained on various colors of ceramic materials (especially roof tiles), they should handle this variation. The main goal now is to refine the methodology to achieve the best possible accuracy. The model's precision depends on the quantity and quality of the training data; it performs based on the data it is trained on.

Figure 2.1: The site for collecting images, a CDW collection and sorting yard near Kladno, Czech Republic.

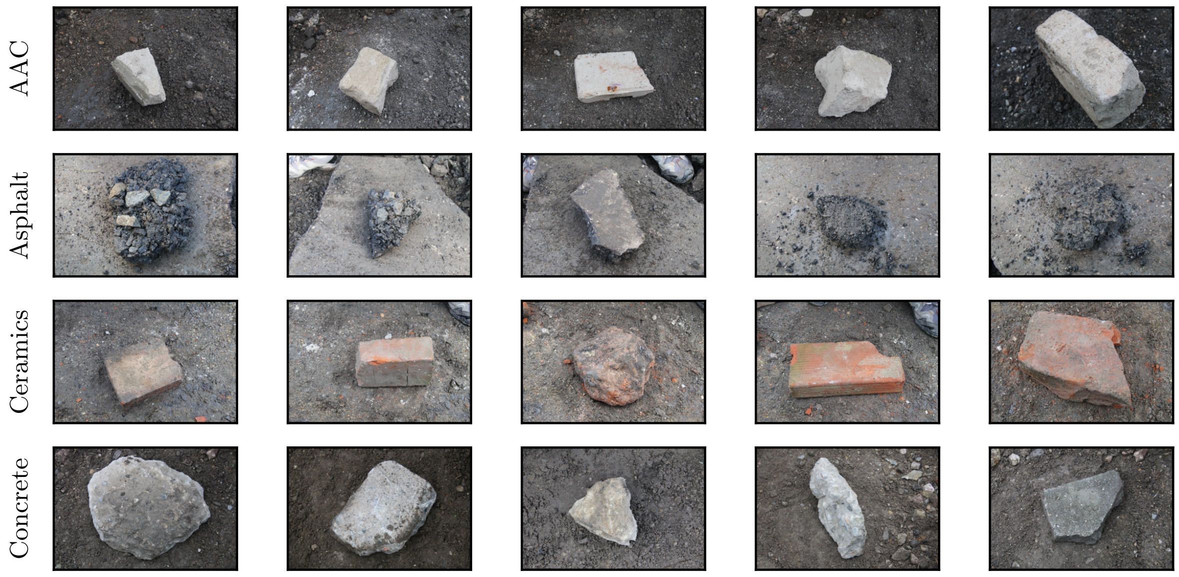

Figure 2.2: Examples of image datasets for the examined CDW materials.

3 Convolution versus extraction of selected features

The acquired image datasets were manually split to individual material classes. The annotated images within each class were divided into training and testing sets in a 4:1 ratio. Since the shape of fragments cannot be the key for classification and the classifiers were trained to recognize the CDW textures, 200×200 px regions (image subsets) were manually extracted for training and testing of the selected classifiers (Figure 3.1). The summary of these training/testing data is provided in Table 3.1.

Table 3.1: Summary of extracted 200×200 px image subsets used for testing and training of selected classifiers.

Material (class)	Number of training images	Number of testing images
AAC	939	235
Asphalt	902	226
Ceramics	620	155
Concrete	825	206

Images represent a high-dimensional input space with $D = N \times N \times C$ features, where $N \times N$ is the image subset size (px) and C is the number of color channels (equal to 3). Such large inputs can be tackled using CNNs, yet reducing the input space by extracting informative numeric features that describe the CDW texture (Figure 3.2) allows to use simple and efficient classification algorithms. In this chapter, we scrutinize the GB and MLP models for such a classification based on extracted features.

3.1 Metrics

The following metrics are proposed to describe the color and texture of CDW fragments, reducing the input space to $D = 4$: (i) mean intensity, (ii) mean intensity of a selected color channel, (iii) Shannon entropy, and (iv) mean intensity gradient. To calculate these quantities, local coordinates (i, j) are introduced for image subsets (Figure 3.3). The 3-dimensional matrix of intensities for individual color channels, $I(C, i, j)$,

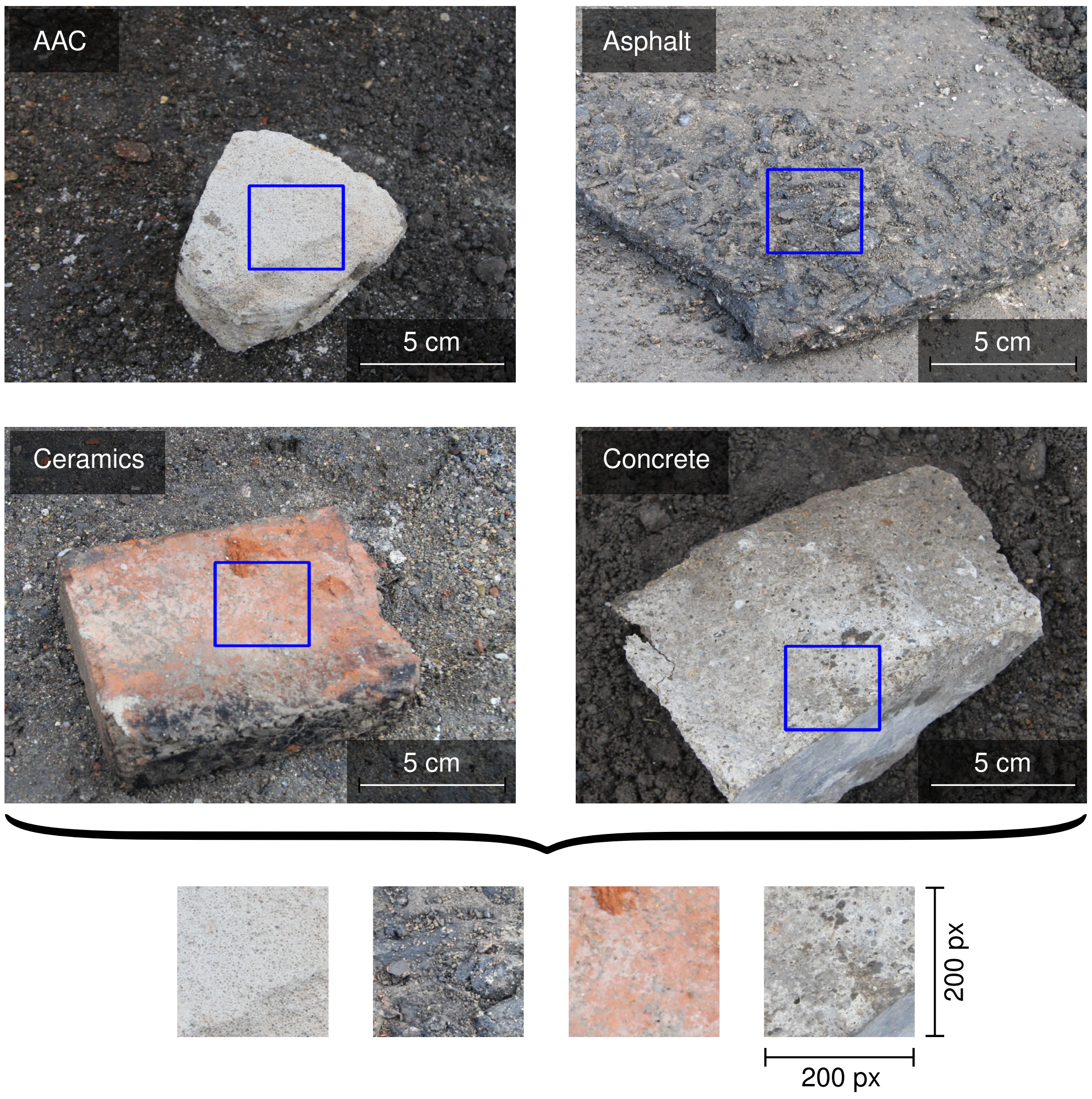

Figure 3.1: Manual extraction of 200×200 px regions (image subsets) used for training and testing of selected classifiers.

was reduced to a single-channel matrix $I(1,i,j) \equiv I_{\text{gray}}(i,j)$, representing a gray-scale image, as

$$I_{\text{gray}}(i,j) = 0.299\, I_{\text{red}}(i,j) + 0.587\, I_{\text{green}}(i,j) + 0.114\, I_{\text{blue}}(i,j), \tag{3.1}$$

where $I_{\text{red}}(i,j)$, $I_{\text{green}}(i,j)$, and $I_{\text{blue}}(i,j)$ represent the matrices of intensities for the red, green, and blue channel, respectively. The weights for individual channels follow luma encoding that reflects different human vision sensitivity to particular colors (Khudhair et al., 2023).

3.1.1 Mean intensity

Mean intensity, $\overline{I_{\text{gray}}}$, is strongly influenced by the illumination of a captured scene and cannot be considered a reliable feature if constant illumination is not ensured for

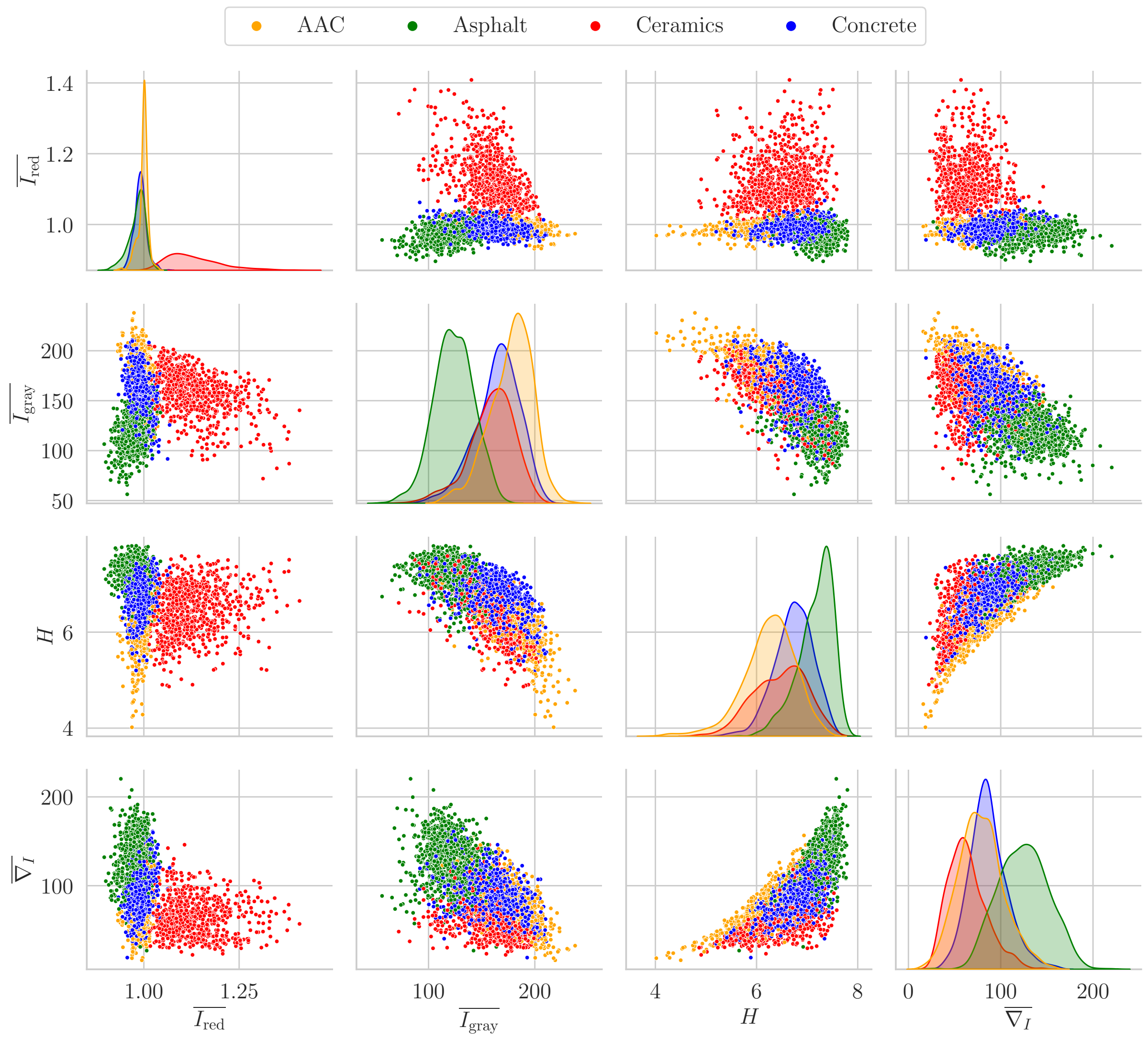

Figure 3.2: Visualization of the image subset characteristics for individual materials (classes) as pairwise scatter plots; marginal distributions of each feature for each class are plotted on the diagonal.

all (training, testing, and classified) images. Since this proof-of-the-concept study is intended as a cookbook for CDW fragments recognition on conveyor belts in an indoor environment, $\overline{I_{\text{gray}}}$ can be considered as one of the relevant features for classification and is calculated as

$$\overline{I_{\text{gray}}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{I_{\text{gray}}(i, j)}{N^2}. \tag{3.2}$$

3.1.2 Mean intensity of red color

The color distribution is one of the key features and many machine-learning models for material recognition were based purely on color-based classification (Son et al.,

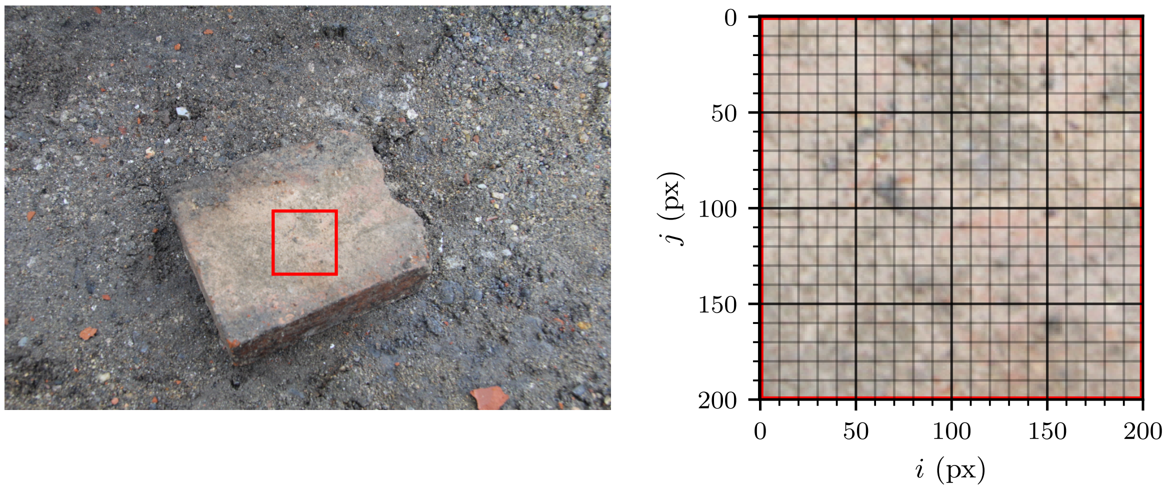

Figure 3.3: Local coordinates (i, j) for a subset of pixels (right) arbitrarily located within an image of a CDW fragment (left).

2012). It was found during a preliminary analysis that for CDW materials, it is sufficient to focus on the predominance of a specific color. Given the orange/reddish color of ceramic fragments, the mean intensity of the red channel, $\overline{I_{\text{red}}}$, relative to the mean intensity (brightness) was selected as the most appropriate color-related label and its value was calculated as

$$\overline{I_{\text{red}}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{I_{\text{red}}(i,j)}{N^2} \frac{1}{\overline{I_{\text{gray}}}}. \tag{3.3}$$

3.1.3 Shannon's entropy

Many distinct CDW materials have similar colors and color-based labeling may fail (Dimitrov and Golparvar-Fard, 2014; Bosché et al., 2015; Nežerka and Trejbal, 2019). To evaluate the randomness of a texture pattern as an additional feature, Shannon's entropy appears to be the most easy-to-calculate measure (Wu et al., 2013; Antoš et al., 2017; de Sousa Filho et al., 2022). It was first proposed by Claude Shannon in 1948 to evaluate the average level of uncertainty in a signal as (Shannon, 1948; Wu et al., 2011)

$$H = -\sum_{I_{\text{gray}}=0}^{255} P(I_{\text{gray}}) \log_2 P(I_{\text{gray}}), \tag{3.4}$$

where $P(I_{\text{gray}}) \in [0, 255]$ (8-bit images) is the frequency of gray pixels' intensity. High values of H indicate higher uncertainty (randomness) of the signal (image).

3.1.4 Mean intensity gradient

Mean intensity gradient ($\overline{\nabla_I}$) was proposed by Pan et al. (2010) as an indicator of stochastic pattern quality in regard to digital image correlation measurements. It evaluates the frequency and intensity of irregularities within an image. Such a measure is directly related to the texture roughness, being another crucial feature used for material classification (Yuan et al., 2020). In this study, the mean intensity gradient was calculated as

$$\overline{\nabla_I} = \sum_{i=1}^{N}\sum_{j=1}^{N} |\nabla I_{\text{gray}}(i,j)| \frac{1}{N^2}, \tag{3.5}$$

where $|\nabla I_{\text{gray}}(i,j)| = \sqrt{I_i(i,j)^2 + I_j(i,j)^2}$ is the modulus of local intensity gradient and I_i and I_j are the i-directional and j-directional derivatives of $I_{\text{gray}}(i,j)$ at each pixel location (i,j). The differentiation was accomplished using a Sobel operator with a 3×3 kernel (Nixon and Aguado, 2020).

3.2 Classifiers

The machine-learning models used for classification are only briefly introduced in the following sections, along with a presentation of input parameters for each model. Detailed descriptions and analyses of the models are beyond the scope of this paper. The curious reader is referred to comprehensive books on machine learning such as ones by Géron (2022) and Murphy (2022).

The choice of classifiers in this study was driven by the aim to span a spectrum of algorithmic complexity and to capture the strengths of different types of models. Specifically:

1. GB is renowned for its efficiency in classification tasks. It excels in handling structured data and can seamlessly navigate the non-linear relationships between features, making it a robust choice for our dataset (Zhou, 2021).
2. MLP, as a basic form of artificial neural networks (ANNs), bridges the gap between traditional machine learning and deep learning techniques (Ho et al., 2023). Its inclusion allowed us to gauge the efficiency of a simpler neural network architecture in the context of CDW recognition.
3. CNN was incorporated as a benchmark due to its inherent layered analysis capabilities. By automatically extracting features, CNNs detect edges and intricate patterns. Its performance provides insights into how deep learning techniques interpret the visual features in the CDW fragments.

The performance of individual classifiers was tested on a custom-built desktop computer equipped with an Intel 4 core i3-8350K CPU, 16 GB RAM, 250 GB SSD hard drive, Windows 10 operating system, and Python 3.10.9. The Python codes and pre-trained models are provided along with this paper (Zbíral and Nežerka, 2023).

3.2.1 Gradient boosting

GB is a machine learning algorithm that typically uses decision trees (Salzberg, 1994) as its base models (Friedman, 2001). The decision tree is a flowchart-like tree structure where each internal node tests an attribute, and the connected branches represent an outcome of the test. Analogically to leaves, the terminal nodes hold class labels (Friedman, 2002).

At each iteration, GB trains a weak decision tree model on the residual errors between the true and predicted labels of the previous iteration. The final prediction is made by adding up the predictions of all the decision trees, where the contribution of each tree depends on its weight, determined by the improvement in the loss function after adding the tree to the ensemble. The loss function is minimized using gradient descent. The algorithm usually outperforms random forest classifiers in terms of speed and accuracy of the predictions (Hastie et al., 2008; Piryonesi and El-Diraby, 2021).

The GB classifier used in this study was implemented in the Scikit-Learn v.1.1.3 Python package. Standardization of features was performed using the *preprocessing.StandardScaler* class. Cross-validation was accomplished using the *model_selection.StratifiedShuffleSplit* class that provides randomly selected indices to split datasets into test/train data and preserves the percentage of samples for each class. The hyperparameters for the *ensemble.GradientBoostingClassifier* model class were defined as summarized in Table 3.2. The optimum parameters were selected based on the prediction accuracy and speed. The optimization was done using the Scikit-learn's *model_selection.GridSearchCV* class that provides an exhaustive search over specified values of model parameters (learning rate $\in [0.2, 0.8]$, maximum depth $\in [3, 5]$, and a number of estimators $\in [100, 200]$). Other hyperparameters were kept in their default settings.

Table 3.2: Summary of hyperparameters for the GB classifier implemented in Scikit-Learn v.1.1.3 (*ensemble.GradientBoostingClassifier* model class).

Input parameter	Keyword argument	Value	Note
Random state	random_state	0	Fixing the random state ensures deterministic behavior during fitting
Learning rate	learning_rate	0.4	Learning rate shrinks the contribution of each tree
Maximum depth	max_depth	4	Maximum depth of individual regression estimators, limiting the number of nodes in decision trees
Number of estimators	n_estimators	125	Number of boosting stages to perform

3.2.2 Multi-layer perception

MLP is a type of artificial neural network that consists of an input layer, a specified number of hidden layers, and an output layer (Rumelhart et al., 1986; Hinton et al., 2006). The input layer represents the features of the input data, while the output layer represents the predicted probability for all classes. The hidden layers are used to learn the non-linear transformations of the input features that lead to the final prediction. In our implementation, the MLP model consists of a single hidden layer; this hidden layer consists of neurons, where each neuron applies a weighted sum of the input features and a bias term, followed by an activation function, such as sigmoid or hyperbolic tangent (tanh). The weights and biases are learned through backpropagation, where the gradients of the loss function are computed to update the weights and biases using gradient descent.

Also, the MLP classifier was implemented in the Scikit-Learn v.1.1.3 Python package. The training procedure was similar to that of the GB model: the standardization of features was performed using the *preprocessing.StandardScaler* class and *model_selection.StratifiedShuffleSplit* class was used for cross-validation. The hyperparameters for the *neural_network.MLPClassifier* model class were defined as summarized in Table 3.3. The search for optimum parameters was also accomplished using the Scikit-learn's *model_selection.GridSearchCV* class, searching over specified values of model parameters (learning rate $\in$ {adaptive, constant $\in$ [0.005, 0.015, 0.05]}, solver $\in$ {stochastic gradient descent, stochastic gradient-based optimizer (adam) (Kingma and Ba, 2015)}, activation $\in$ {rectified linear unit function (ReLU), hyperbolic tan function (tanh)}, and a hidden layer size $\in$ [5, 100]). Other hyperparameters were kept in their default settings.

Table 3.3: Summary of hyperparameters for the MLP classifier implemented in Scikit-Learn v.1.1.3 (*neural_network.MLPClassifier* model class).

Input parameter	Keyword argument	Value	Note
Random state	random_state	0	Fixing the random state ensures deterministic behavior during fitting
Learning rate	learning_rate_init	0.015	Controls the step-size in updating neuron weights
Maximum number of iterations	max_iter	800	Number of epochs (how many times each data point is used)
Learning rate schedule	learning_rate	'constant'	Selected constant learning rate
Solver	solver	'adam'	Weight optimization using the Adam algorithm (Kingma and Ba, 2015)
Neuron activation function	activation	'tanh'	Activation function for the hidden layer
Hidden layer size	hidden_layer_sizes	(20,)	Single hidden layer with 20 neurons

3.2.3 Convolutional neural network

CNN is a type of artificial neural network that is designed for the analysis of data with a grid-like topology (e.g., images) (Zhou et al., 2012; LeCun et al., 2015; Krizhevsky et al., 2017). It consists of several layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layer applies a convolution operation to the input image, where the convolution kernel slides over the image and computes the dot product between the kernel and the local patch of the image to extract features. The convolutional layer is followed by an activation function that applies non-linear transformations to the output of the convolution. The pooling layer reduces the spatial dimensions of the output of the convolutional layer by applying a pooling operation, such as max pooling, that takes the maximum value of a local patch. The fully connected layer combines the features learned by the convolutional and pooling layers and makes the final prediction. The weights and biases of the convolutional and the fully connected layers are adjusted during the network training through backpropagation, exploiting the gradient descent algorithm.

Unlike GB and MLP classifiers, CNN takes the whole image as input. Since the model in our study was trained on 200×200 px 3-channel (RGB) images, images for classification having a different size were rescaled to 200×200 px using an interpolation function. The CNN classifier was implemented in the Tensorflow Keras v.2.10.0 Python package, provided by the *models.Sequential* class.

Different architectures of CNNs with various number of filters for the convolutional layers have been tested. The selected model includes three convolutional layers, each followed by a max pooling layer, a flatten layer, and two dense layers. The first and third convolutional layers have 32 3×3 filters, a stride of 1, and a ReLU activation function. The second convolutional layer has 64 3×3 filters and the same activation function. The max pooling layers downsample the feature maps by a factor of two to make the model more efficient. The flatten layer converts the 2D feature maps into a 1D vector. The two dense layers consist of 256 units with a ReLU activation function, followed by an output layer with four neurons corresponding to the individual CDW classes.

The selected model architecture is described in detail in Table 3.4. During the training process, the model achieved 100% accuracy on the training data (α_{train}) after 30 epochs, but the maximum accuracy on the testing data ($\alpha_{\text{test}} = 80\%$) was reached after 11 epochs, suggesting potential overfitting (Figure 3.4). The model trained after 11 epochs was adopted for the future CDW classification.

3.3 Model Evaluation Metrics

To evaluate the performance of our multi-class classification models, we primarily utilize accuracy and the weighted F-score.

Let P_c^{true} be the number of true positives for class c, N_c^{true} the true negatives, P_c^{false} the false positives, and N_c^{false} the false negatives. Accuracy, denoted by α, measures the proportion of all correct predictions across all four classes:

$$\alpha = \frac{\sum_{c=1}^{4} P_c^{\text{true}} + N_c^{\text{true}}}{\sum_{c=1}^{4} P_c^{\text{true}} + N_c^{\text{true}} + P_c^{\text{false}} + N_c^{\text{false}}}. \tag{3.6}$$

The precision P_c and recall R_c for each class are respectively defined as:

$$P_c = \frac{P_c^{\text{true}}}{P_c^{\text{true}} + P_c^{\text{false}}} \quad \text{and} \quad R_c = \frac{P_c^{\text{true}}}{P_c^{\text{true}} + N_c^{\text{false}}} \tag{3.7}$$

Table 3.4: Architecture of the CNN models; the individual layers were implemented in the Tensorflow Keras v.2.10.0 Python package, the *layers* class.

Layer	Keras class	Purpose
Convolutional layer (32 filters, size 3×3)	*Conv2D(32, (3, 3), 1, activation='relu', input_shape=(200, 200, 3))*	Extract features from the input images
Maximum pooling layer (2×2 pool)	*MaxPooling2D()*	Downsample the feature maps from the previous layer
Convolutional layer (64 filters, size 3×3)	*Conv2D(64, (3, 3), 1, activation='relu')*	Extract features from the previous layer
Maximum pooling layer (2×2 pool)	*MaxPooling2D()*	Downsample the feature maps from the previous layer
Convolutional layer (32 filters, size 3×3)	*Conv2D(32, (3, 3), 1, activation='relu')*	Extract features from the previous layer
Flattening layer	*Flatten()*	Flattens the 2D feature map into a 1D array
Fully connected layer (256 neurons)	*Dense(256, activation='relu')*	Take the flattened vector from the previous layer as input
Output layer (4 neurons)	*Dense(4)*	Values of individual neurons represent probabilities that the input belongs to each of the possible classes

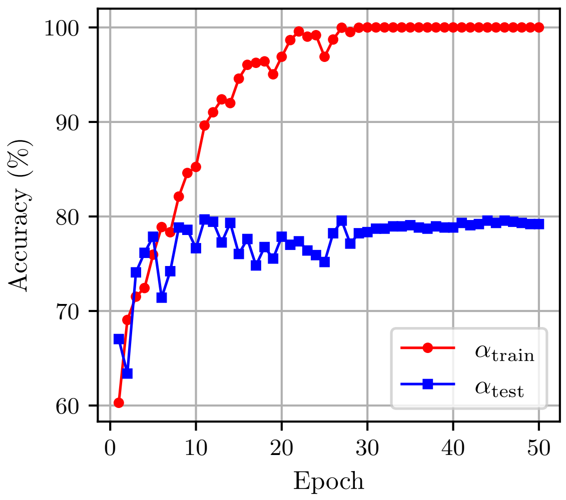

Figure 3.4: Training and testing accuracy as a function of epoch recorded during CNN training.

The F-score for class c, denoted as F_c, offers a balance between P_c and R_c. It is described as the harmonic mean of P_c and R_c:

$$F_c = \frac{2P_cR_c}{P_c + R_c} \tag{3.8}$$

For our multi-class problem, the weighted F-score, F_{weighted}, is calculated by averaging the F-score of each class, weighted by the proportion of samples from that class:

$$F_{\text{weighted}} = \sum_{c=1}^{4} w_c F_c \tag{3.9}$$

where w_c denotes the weight (proportion of samples) for the c^{th} class.

We employ both α and F_{weighted} in this study to evaluate the performance of our classifiers, providing a comprehensive view of their efficacy, especially in light of the minor class imbalance present in our dataset.

3.4 Results and discussion

The performance of individual classifiers is represented through confusion matrices (Figure 3.5), alongside the results of "manual" classification. This manual classification was accomplished using an online survey[1] by five experts on building materials from the Faculty of Civil Engineering, Czech Technical University in Prague.

While accuracy provides a general measure of correctness, the weighted F-score offers a more balanced measure between precision (how many selected items are relevant) and recall (how many relevant items are selected). For instance, GB and MLP classifiers achieved an accuracy of 82.5% with F-scores of 82.4%, indicating a harmonious balance between precision and recall. The CNN classifier achieved an accuracy of 82.1% and an F-score of 82.3%, further demonstrating the model's consistent performance. In comparison, human experts achieved an accuracy of 87.2% and an F-score of 87.5%, outperforming the machine classifiers slightly[2].

Both machine-learning classifiers and human experts had difficulties distinguishing between image samples of AAC, asphalt, and concrete. This demonstrates the inherent difficulty in differentiating these materials visually, particularly when they share similar characteristics like a grayish color and texture. In contrast, ceramics (bricks, roof tiles, etc.) were recognized with an impressive accuracy of over 96% by both groups. A potential enhancement to the classification process could be the integration of a basic weight measurement device. Given the significant differences in density between the grayish materials, weight can be a distinguishing factor. Moreover, if a dual-camera setup were employed, the segmentation technique would permit volume estimation from visual data, further refining the differentiation process.

[1] https://rm.fsv.cvut.cz/cdw/

[2] As shown next, evaluating more image subsets increased the accuracy of ML models, surpassing the classification accuracy of human experts.

Despite the commendable performance of human experts, there are inherent limitations to relying on manual sorting. Prolonged concentration can lead to lapses in attention, impacting the consistency of the sorting process (Firestone, 2020). Furthermore, machine classifiers, especially when deployed on standard office computers, can process samples at a rate that outpaces human capability by orders of magnitude.

In recent literature, Davis et al. (2021) reported accuracy levels between 80% and 97% for the CNN-based classification of general waste. Their categories included paper, glass, plastic, metal, cardboard, and non-recyclables. Although their work achieved an accuracy of up to 95.7% for CDW, it's crucial to note that the objects they classified had more distinct shapes than the CDW fragments. Xiao et al. (2019) reported a perfect accuracy of 100% in their classification of CDW on a conveyor belt. They employed a high-cost near-infrared hyperspectral camera and a dataset with distinct categories like foam, plastic, brick, concrete, and wood. Introducing more challenging materials such as asphalt conglomerates or AAC, often found in CDW, could potentially reduce this high accuracy even with advanced hardware.

A study on the performance of the individual classifiers in terms of speed and accuracy is presented as a function of subset size in Figure 3.6. As larger subsets contained more information, the accuracy of models increased. This phenomenon was most significant in the case of CNN, for which the image subsets had to be rescaled to 200×200 px to have the same size as images used for training. Similar findings were reported by Dimitrov and Golparvar-Fard (2014), who developed a system for vision-based material recognition and monitoring of construction progress, employing the SVM classifier (Cortes and Vapnik, 1995).

In our study, the GB and MLP models that utilized feature extraction, exhibited similar speed and accuracy, both superior to CNN, especially for small subsets. Unlike CNN, both models approached their maximum accuracies at approximately 150×150 px subset size. The classification speed of GB and MLP classifiers, including feature extraction, was about 15× higher compared to CNN.

The practical demonstration of the image subset classification is provided in Figure 3.7. Here, the randomly selected CDW fragments from the testing dataset were localized using the Rembg[1] Python package based on the U^2-Net deep neural network (Qin et al., 2020). An auxiliary script was designed to extract image subsets from the unmasked regions. The accuracy of the CNN classifier was compromised by the small size (135×135 px) subsets placed over the region of interest; however, even despite this shortcoming, even the CNN classified the fragments correctly with high confidence. Nearly 100% confidence was reached by the GB and MLP classifiers.

This demonstration shows that the accuracy reached for individual subsets is improved by placing a higher number of these subsets over the samples. The accuracy was tested on a comprehensive dataset containing 2664 images of CDW fragments (Nežerka et al., 2023); the summary of reached accuracies for individual classifiers is provided in Table 3.5. Classification of several samples per a CDW fragment led to overall accuracy ranging between 85.9% (CNN) and 92.3% (GB), reaching the accuracy reported by other authors dealing with the classification of clean building mate-

[1]https://github.com/danielgatis/rembg

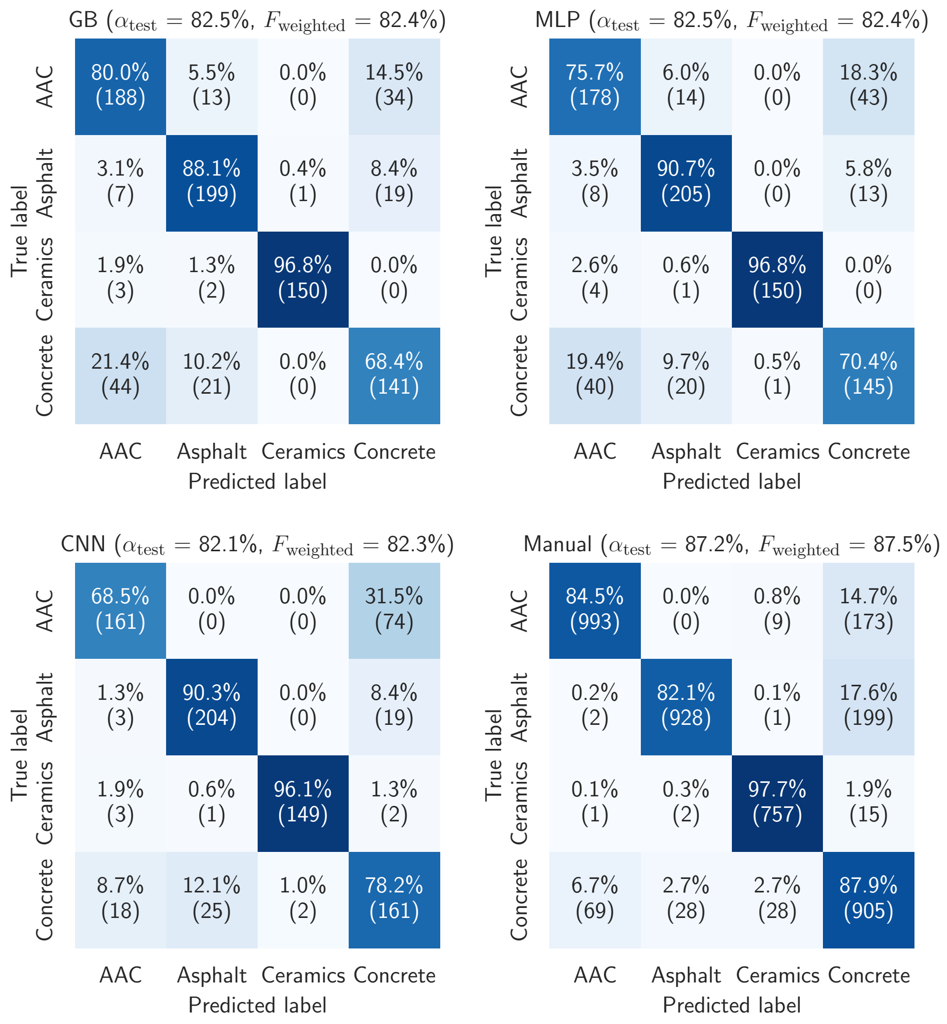

Figure 3.5: Confusion matrices for different classifiers and comparison of their performance with manual classification done by five experts on building materials from the FCE CTU in Prague.

rials (not contaminated by other materials or dust). In a study by Mahami et al. (2020), the authors managed to classify eleven construction materials using CNN (VGG16 network (Simonyan and Zisserman, 2014)) and reached up to 97.35% accuracy, yet, their dataset did not contain contaminated materials having similar textures, such as fragments of AAC and concrete in our study.

Our models, especially the Gradient Boosting and Multi-Layer Perceptron classifiers, demonstrated competitive performance when compared to previous studies, as summarized in Table 3.6. Notably, while our dataset size was comprehensive, the nature of our CDW images, which included contaminated materials with similar textures, made the classification task more challenging.

It should be noted that all the images for both training and testing datasets were taken using the same camera and similar conditions, which can compromise the robustness of the classification models. The goal of this proof-of-the-concept study is

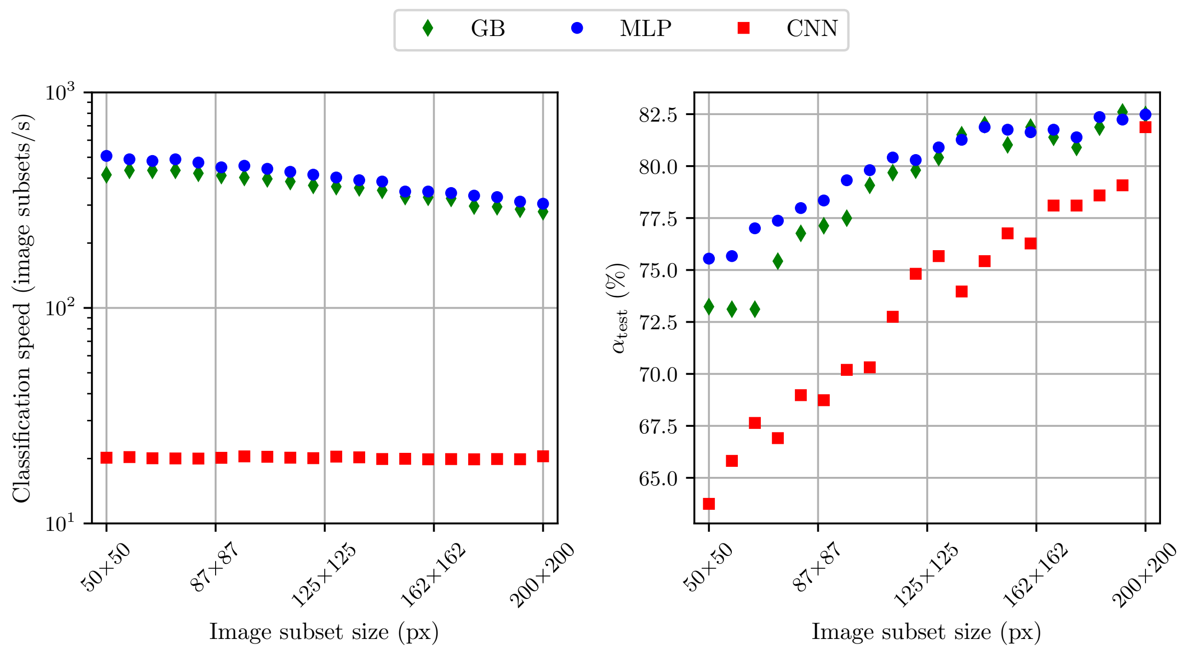

Figure 3.6: Speed (left) and accuracy (right) reached by individual classifiers on the validation (testing) datasets for different sizes of image subsets that were extracted by cropping the redundant portion of the images.

Table 3.5: Accuracy of different classifiers when recognizing whole CDW fragments by classifying several (>4) 200×200 image subsets with a 70 px overlap (Figure 3.8).

Classifier	AAC (582 images)	Asphalt (741 images)	Ceramics (572 images)	Concrete (769 images)	Complete dataset (2664 images)
GB	86.9%	93.9%	99.1%	89.7%	92.3%
MLP	89.4%	93.8%	98.4%	85.2%	91.3%
CNN	56.7%	97.2%	99.0%	87.5%	85.9%

to demonstrate the capabilities of the proposed low-cost lightweight procedures that could be implemented in CDW sorting and recycling plants for CDW recognition on conveyor belts. For particular industrial applications, new site-specific training datasets should be acquired, optimally involving auxiliary data (weight, acoustic emissions, etc.) from other sensors. Fusion of RGB cameras with different sensors could significantly increase the accuracy, especially in the case of lightweight AAC which is often confused with fragments of concrete that also have a fine texture and grayish color.

3.5 Application procedure

Our developed machine-learning-assisted method for CDW fragment recognition is designed for easy integration into existing CDW sorting systems. Here, we outline the

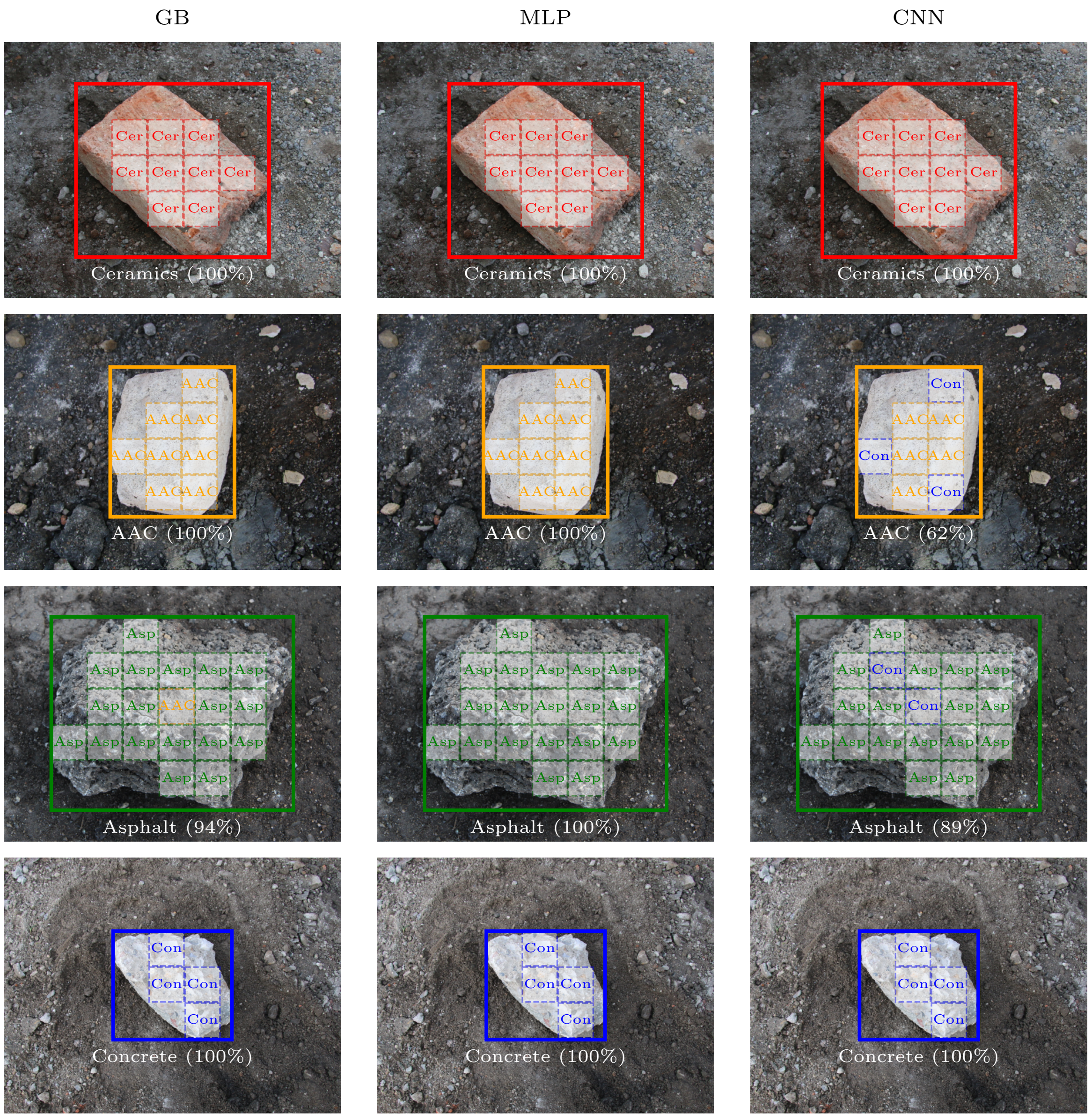

Figure 3.7: Localization of whole CDW fragments and their classification based on texture recognition using different classifiers; the size of image subsets 135×135 px.

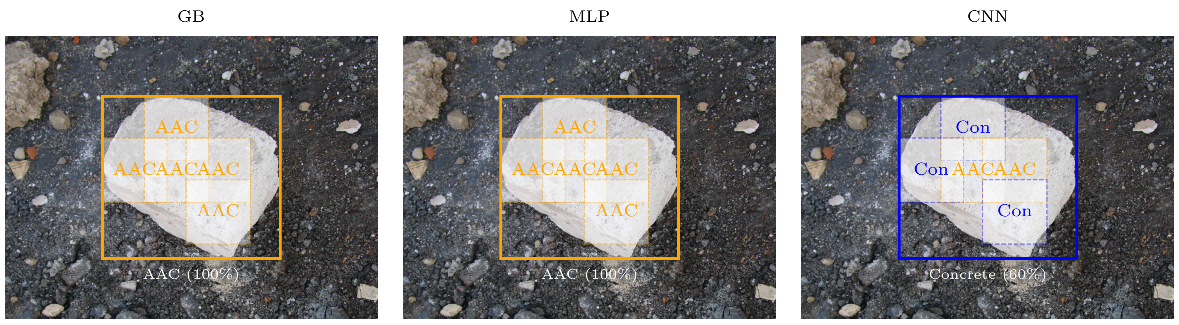

Figure 3.8: A typical misclassification of AAC fragments by CNN during a comprehensive validation of the classification algorithms; size of image subsets 200×200 px with a 70 px overlap.

Table 3.6: Comparison of the current study with previous significant works focused on machine-learning-based recognition of construction materials in terms of model performance, data type, and dataset size.

Reference	Model	Accuracy	Dataset type and size	Dataset size
This study (GB)	GB	92.3%	CDW images	2664
This study (MLP)	MLP	91.3%	CDW images	2664
This study (CNN)	CNN	85.9%	CDW images	2664
Davis et al. (2021)	CNN	80-97%	Images of conatiners with bulk CDW	2283
Xiao et al. (2019)	CNN	100%	Hyperspectral images of very diverse materials	250
Dimitrov and Golparvar-Fard (2014)	SVM	Up to 97.1%	Point cloud patches (images of construction surfaces)	3740
Mahami et al. (2020)	CNN (VGG16)	97.35%	Images of clean very diverse materials	1231
Yuan et al. (2021)	BD-P model	90.2%	Bulk density (truck loads)	4.27 mil.
Hoong et al. (2020)	CNN (Custom ResNet34)	97%	Images of recycled aggregates	36000
Lin et al. (2022)	CNN (CVGGNet-16)	76.6%	Images of diverse clean bulk materials	2836 (bofore augmentation)

potential application procedure:

1. **Image Acquisition**: Using high-resolution cameras, images of CDW fragments on conveyor belts or sorting platforms are captured. Ideally, this would be integrated into a continuous flow system where CDW moves along a conveyor.

2. **Preprocessing**: The captured images undergo preprocessing, which may include cleaning using air-flow or other mechanisms to enhance clarity, and then they are fed into the model.

3. **Density Estimation**: For individual fragments on the conveyor belt, a weight measurement system can be integrated to estimate the density of each fragment. This can assist in further refining the classification, especially for fragments with similar appearances but different densities (e.g., AAC and concrete).

4. **Classification**: The preprocessed images are classified in real-time using a trained model. The model identifies the type of CDW fragment and can potentially direct its sorting into appropriate bins or sections.

5. **Post-processing**: Based on classifications, automated mechanisms or manual laborers can be directed to ensure correct sorting or further refinement.

6. **Feedback Loop**: The system can be designed to continuously learn from any misclassifications through a feedback mechanism, enhancing accuracy over time.

This proposed application procedure is modular and can be customized based on the specific requirements of the CDW sorting facility, available resources, and desired accuracy levels.

4 Deep neural networks for segmentation and classification

Multiple deep neural networks have been introduced for image segmentation tasks over the course of the last few years. U-Net (Ronneberger et al., 2015) and fully convolutional networks (FCN) (Long et al., 2015) are two of the most common, with U-Net being used as a keyword over 14,000 times and FCN being used as a keyword over 20,000 times based on data from web of science. (Ozturk et al., 2020) found that U-Net gave significantly more accurate results than FCN when using smaller resolution images (256$\times$256 px). U-Net was the number one choice for the segmentation of construction and demolition waste (CDW) as we aim to perform this task in real time, and higher resolutions may result in slower segmentation.

ResNet (He et al., 2016), winner of the 2015 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), appeared to be one of the best choices for the classification task as it is highly accurate and fast to make predictions. Another idea to improve classification accuracy was to combine deep neural networks with decision trees. Multiple studies have shown the potential of this idea to increase the accuracy of deep neural networks (Kontschieder et al., 2015; Arifuzzaman et al., 2023; Crockett et al., 2018).

4.1 Data preparation

Data preparation is a critical step in the training of deep neural networks, significantly impacting the model's performance, efficiency, and generalization capabilities. Effective data preparation involves several key processes, including data cleaning, augmentation, and splitting, each contributing to the overall quality and robustness of the model.

4.1.1 Data cleaning

The original dataset presented in Chapter 2 was inappropriate for automatic segmentation as some images were depreciated for various reasons: they were taken from a pile of multiple pieces of material, and some contained structures, feet, or other objects that were redundant in an image. Data cleaning ensured the reliable and automatic creation of the ground truth masks that served as an important input into segmentation

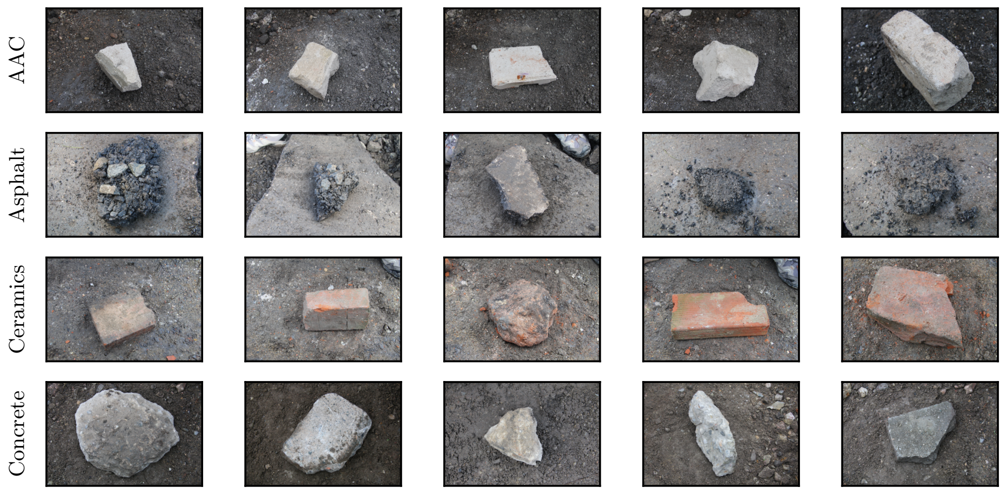

Figure 4.1: Examples of image datasets for the examined CDW (Nežerka et al., 2024).

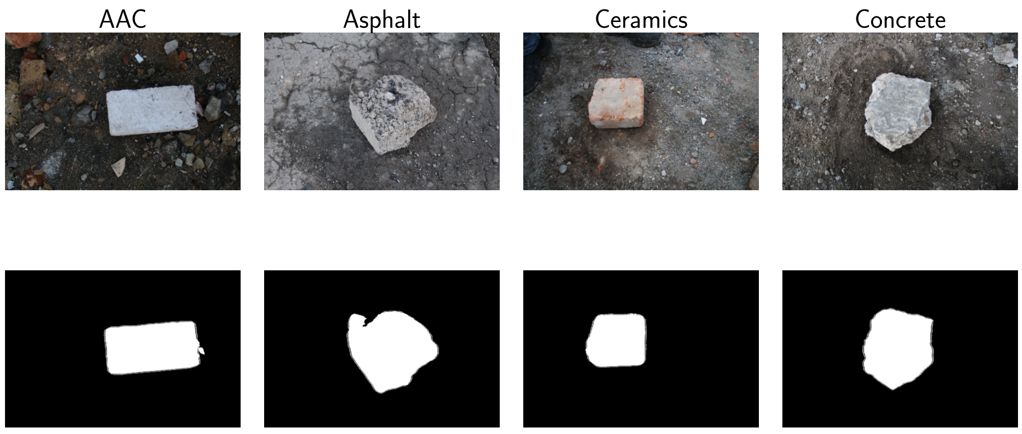

Figure 4.2: Images with corresponding masks for each material.

models. From the original dataset of 2,664 images, there were 2,133 images left, with 538 images of AAC, 315 images of asphalt, 622 images of ceramics, and 658 images of concrete (Figure 4.1).

4.1.2 Ground truth masks

For supervised learning segmentation tasks, it is necessary to create ground truth masks with highlighted regions of interest (Figure 4.2). For this purpose, we used rembg library, which is a powerful and easy-to-use tool for background removal over a wide range of images. The *remove* function defined in rembg implements multiple background removal models such as U-2-Net (Qin et al., 2020), IsNet (Qin et al., 2022), and Sam (Kirillov et al., 2023), making the process reliable, but computationally expensive. The ground truth masks created using *rembg.remove* have been visually reviewed to ensure maximal suitability.

4.1.3 Data augmentation

Data augmentation is a crucial machine learning technique that significantly improves the generalization of a model, especially when dealing with limited datasets. It involves generating new samples by applying various transformations to the existing data. These transformations include operations like rotation, flipping, translation, scaling, and adding noise (for example, blur or brightness changes). The primary benefits are (i) improved generalization, (ii) increased data diversity, and (iii) enhanced robustness.

Generalization of a model helps a model prevent overfitting, where the model performs well on the training data but poorly on the validation data. Data diversity is very important for implementing neural networks in the real world. In various fields, including CDW sorting, it is difficult or time-consuming to collect diverse and large datasets for training neural networks. Applying data augmentations increases the size of the dataset significantly. The model robustness is crucial for applications such as image recognition in diverse and unpredictable environments.

In our project, we exploited the albumentations library (Buslaev et al., 2020) designed to perform fast and flexible data augmentation for deep learning applications. It provides a comprehensive set of augmentation techniques specifically tailored for image data, making it a popular choice among practitioners and researchers. The library offers a vast array of augmentation techniques, including basic transformations (like flips, rotations, and scaling), color transformations, advanced augmentations (such as cutout, grid distortions, and elastic transformations), and more. The library supports the composition of complex augmentation pipelines and can be easily integrated with popular deep learning frameworks such as pytorch (Paszke et al., 2019), which has been used in this project. The pipeline for data augmentation is presented in the following code snippet:

```python
def augment_image(image, mask):
    # Define the augmentation pipeline for both image and mask
    transform = A.Compose([
        A.Rotate(limit=25, p=1),
        A.HorizontalFlip(p=0.5),
        A.VerticalFlip(p=0.5),
        A.RandomBrightnessContrast(p=0.3),
        A.OneOf([
            A.Blur(blur_limit=5, p=0.2),
            A.GaussianBlur(blur_limit=5, p=0.2),
        ], p=0.2),
        ToTensorV2(),
    ])
```

The cleaned and augmented dataset from Chapter 2 used for training of the U-Net and ResNet models consisted of 4266 images, half of them being augmented.

4.1.4 Data splitting

In this project, we structured the dataset into three sets: training, validation, and test. The training set is crucial for developing the neural network, while the validation set aids in fine-tuning hyperparameters and selecting the optimal model architecture. Adjustments based on the validation set's feedback after each epoch helped us enhance the model's generalization and avoid overfitting. The test set allowed for an unbiased evaluation of the model's performance in conditions mimicking real-world scenarios.

We adopted a 70/15/15 split ratio for both segmentation and classification tasks, effectively balancing the need for extensive training data with adequate validation and testing to ensure the model's robustness and reliability in practical applications. This tailored approach supports our objective of developing deep-learning models that perform well across diverse real-world environments.

4.2 Segmentation

Despite the versatility of the existing rembg model, training our custom U-Net (Ronneberger et al., 2015) model was crucial to achieve real-time segmentation speeds. We conducted runtime tests on both models using a set of 100 images. The rembg model proved inadequate for real-time applications, with a total runtime of 840 seconds, averaging 8.4 seconds per image. In contrast, our tailored U-Net model, detailed further in this section, markedly outperformed rembg, delivering a significantly faster runtime of 0.37 seconds per image, which is 22.7× quicker. This speed enhancement underscores the necessity of developing a specialized model to meet the demands of real-time processing.

4.2.1 U-Net architecture

U-Net architecture was first proposed in 2015 and it is described in Figure 4.3, reproduced from the original paper (Ronneberger et al., 2015). In our model, we used batch norm, not used in the original U-Net model; the architecture of our U-Net model can be seen in a code snippet below.

```python
class DoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DoubleConv, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3,
             ↪  stride=1, padding=1, bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3,
             ↪  stride=1, padding=1, bias=False),
            nn.BatchNorm2d(out_channels),
                nn.ReLU(inplace=True),
            )

    def forward(self, x):
        return self.conv(x)


class Unet(nn.Module):
    def __init__(self, in_channels=3, out_channels=1, features=[64, 128, 256, 512]):
        super(Unet, self).__init__()
        self.downs = nn.ModuleList()
        self.ups = nn.ModuleList()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)

        # Down part of Unet
        for feature in features:
            self.downs.append(DoubleConv(in_channels, feature))
            in_channels = feature

        # Up part of Unet
        for feature in reversed(features):
            self.ups.append(
                nn.ConvTranspose2d(
                feature*2, feature, kernel_size=2, stride=2, padding=0
                )
            )
            self.ups.append(DoubleConv(feature*2, feature))

        self.bottleneck = DoubleConv(features[-1], features[-1]*2)
        self.final_conv = nn.Conv2d(features[0], out_channels, kernel_size=1)

    def forward(self, x):
        skip_connections = []
        for down in self.downs:
            x = down(x)
            skip_connections.append(x)
            x = self.pool(x)

        x = self.bottleneck(x)
        # Now we need to go up
        skip_connections = skip_connections[::-1]

        for idx in range(0, len(self.ups), 2):
            x = self.ups[idx](x)
            skip_connection = skip_connections[idx//2]

            if x.shape != skip_connection.shape:
                x = TF.resize(x, size=skip_connection.shape[2:])

            concat_skip = torch.cat((skip_connection, x), dim=1)
            x = self.ups[idx+1](concat_skip)

        return self.final_conv(x)
```

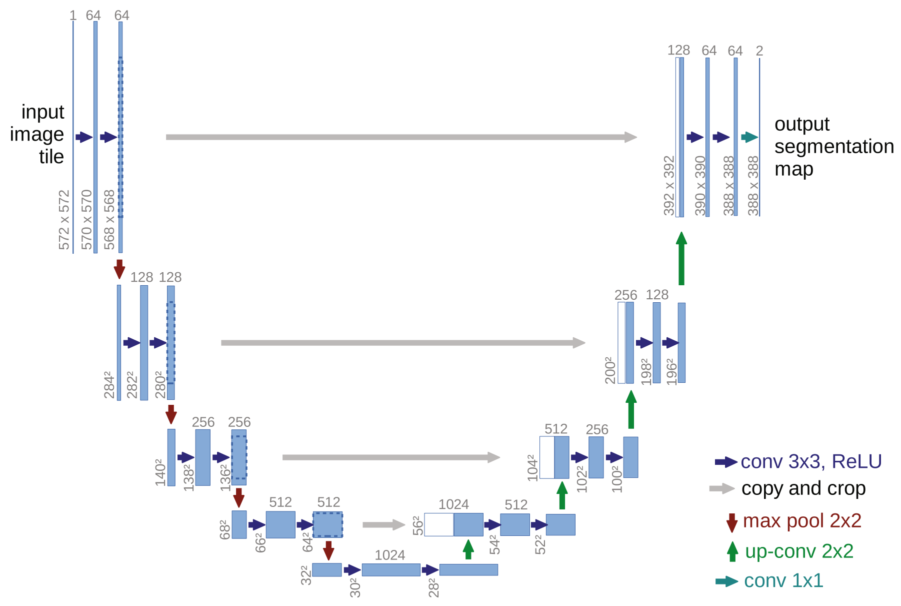

Figure 4.3: U-net architecture (example for 32×32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations (Ronneberger et al., 2015).

4.2.2 Segmentation accuracy metrics

Segmentation accuracy is critical in assessing the performance of image segmentation models. This chapter focuses on two widely used metrics for evaluating segmentation performance: the F1 score ($F1$) and Intersection over Union (IoU). Evaluating segmentation accuracy using $F1$ and IoU provides a balanced and comprehensive understanding of a model's performance.

$F1$ is the harmonic mean of precision and recall, defined as

$$F1 = 2\frac{PR}{P+R}, \tag{4.1}$$

where P represents precision defined as $P = \frac{P^{\text{true}}}{P^{\text{true}}+P^{\text{false}}}$, where P^{true} represents true positives and P^{false} represents false positives and R represents recall defined as $R = \frac{P^{\text{true}}}{P^{\text{true}}+N^{\text{false}}}$, where N^{false} represents false negatives.

The IoU metric, also known as the Jaccard index, is the most intuitive way to measure how good the prediction of segmentation is. It measures the overlap between the predicted region and the ground truth region, and it is defined as

$$IoU = \frac{|A \cap B|}{|A \cup B|}, \tag{4.2}$$

where A is the set of pixels of the predicted region, and B is the set of the ground truth region. The numerator $|A \cap B|$ of the fraction represents the intersection, which is the common area between the predicted and the ground truth regions. The denominator $|A \cup B|$ represents the union, or the total area covered by both the predicted and the ground truth regions, without subtracting any parts of either.

4.2.3 U-Net model

After optimization, the U-Net model used the sigmoid activation function, binary cross entropy with logits loss as a loss function, and Adam optimizer with a learning rate of 0.001. The model was been trained during 35 epochs. As depicted in Figure 4.4, the validation loss values exhibit stabilization around epoch 15, whereas the accuracy metrics for the validation data show stabilization around epoch 20, with optimal performance observed at epoch 32 as illustrated in Figures 4.5 and 4.6. The highest IoU achieved on the validation dataset was 0.902, and $F1$ reached 0.928. Notably, the model from epoch 32 achieved the second-highest performance on the test dataset, with IoU of 0.863 and $F1$ of 0.941. Table 4.1 presents the variability of IoU across different materials, suggesting variations in capture conditions. Although an IoU of 0.867 might initially appear low, examination of Figure 4.7 indicates that even an average IoU of 90% is sufficient for generating bounding boxes necessary for the classification tasks described in Section 4.3.

Class	Mean IoU
AAC	0.943
Asphalt	0.865
Ceramics	0.935
Concrete	0.767

Table 4.1: Mean IoU based on material calculated for the test data.

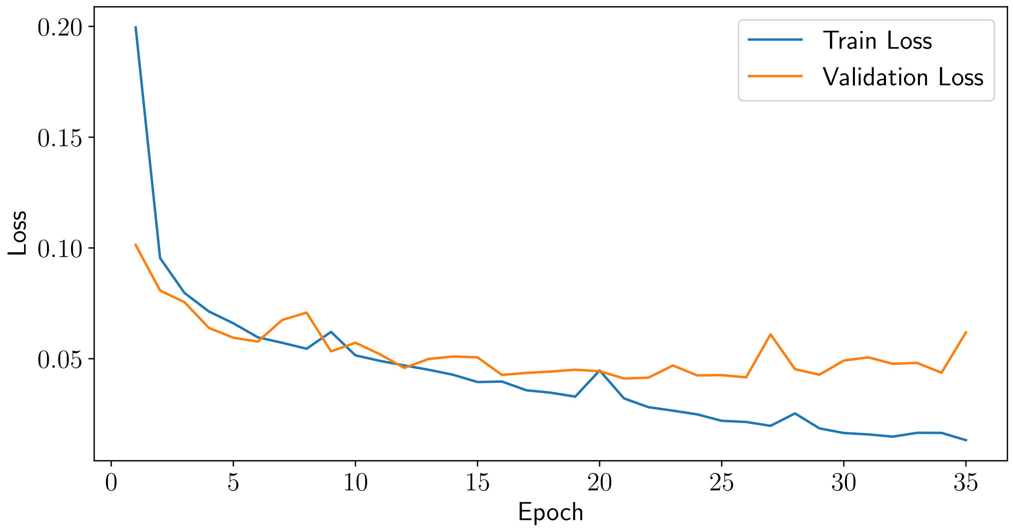

Figure 4.4: Training and validation loss as a function of epoch reported during the training of the U-Net model.

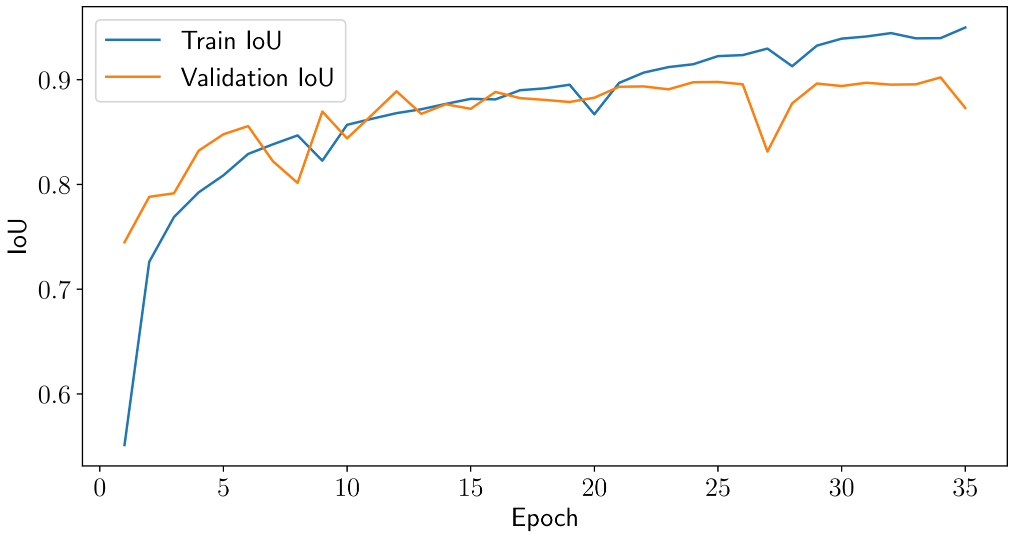

Figure 4.5: Training and validation IoU as a function of epoch reported during the training of the U-Net model.

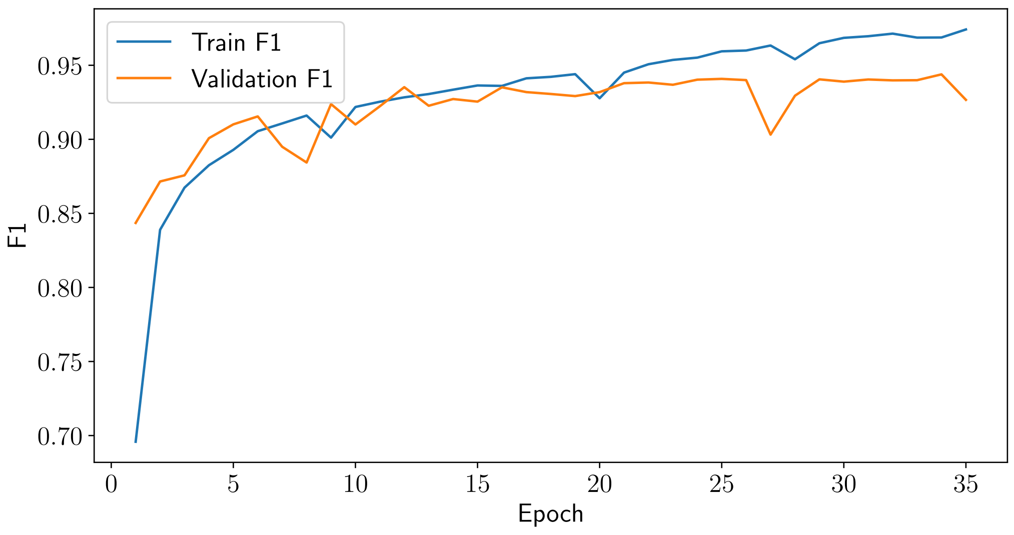

Figure 4.6: Training and validation $F1$ as a function of epoch reported during the training of the U-Net model.

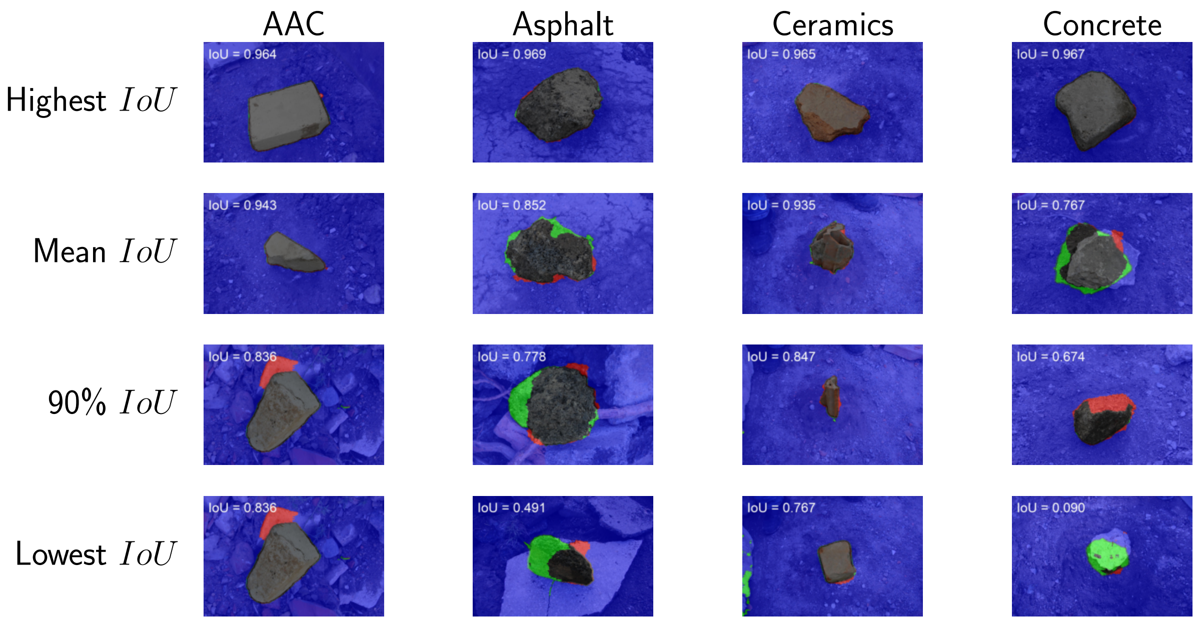

Figure 4.7: Comparison of ground truth and predicted mask. The blue overlay displays true negatives, the red overlay displays false positives, the green overlay displays false negatives, and true positives are transparent.

4.3 Classification

It is imperative to employ the most effective architectures available for classification tasks to achieve optimal results in terms of both accuracy and speed. The ResNet (residual network) architecture (He et al., 2016) was identified as a particularly promising approach for this application. ResNet addresses the problem of training very deep networks, which often suffer from vanishing gradients, making it difficult for the network to learn and converge. Our objective is to surpass the benchmark accuracy of 92.3% established by Nežerka et al. (2024), striving for the highest achievable performance level in CDW classification.

Residual learning: The core idea of ResNet is the use of residual learning to ease the training of networks that are substantially deeper than those previously used. Residual learning involves shortcut connections that skip one or more layers. These shortcuts add the input of a layer directly to the output of a subsequent layer, effectively creating a residual mapping. Mathematically, instead of learning a direct mapping $H(x)$, ResNet learns the residual function $F(x) = H(x) - x$, which is easier to optimize.

Shortcut connections: The shortcut connections perform identity mapping, meaning they do not introduce additional parameters or computational complexity. These connections help mitigate the vanishing gradient problem by allowing gradients to flow more easily through the network.

Building Blocks:

- **Basic block**: Used in ResNet architectures with fewer layers (e.g., ResNet-18 and ResNet-34). It consists of two convolutional layers with batch normalization and ReLU activation, followed by an addition operation from the shortcut connection.
- **Bottleneck block**: Used in deeper architectures (e.g., ResNet-50, ResNet-101, ResNet-152). It consists of three convolutional layers: a 1$\times$1 layer for reducing dimensions, a 3$\times$3 layer, and another 1$\times$1 layer for restoring dimensions, along with batch normalization and ReLU activation.

Depth and performance: ResNet architectures come in various depths, commonly referred to by the number of layers: ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. Deeper networks like ResNet-50 and beyond use bottleneck blocks to reduce the computational load while maintaining performance.

Impact on image recognition: ResNet has significantly improved the accuracy of image recognition tasks. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015 with a top-5 error rate of 3.57%. Its architecture has been influential, leading to the development of other advanced models such as DenseNet, Inception-ResNet, and others.

4.3.1 Model architecture

ResNet-18/34 (Basic block):

- Consists of multiple layers of basic blocks.
- Each basic block has two 3×3 convolutional layers.
- Shortcut connections skip over each block.

ResNet-50/101/152 (Bottleneck nlock):

- Consists of multiple layers of bottleneck blocks.
- Each bottleneck block has a 1×1, 3×3, and another 1×1 convolutional layer.
- Shortcut connections skip over each block.

Implementation:

- **Layers**: A typical ResNet model includes an initial convolutional layer followed by multiple stages of residual blocks, each stage reducing the spatial dimensions while increasing the number of filters.
- **Pooling**: A global average pooling layer at the end aggregates the feature maps before passing them to a fully connected layer for classification.

The following code snippet illustrates the model's implementation:

```python
class BasicBlock(nn.Module):
def __init__(self, in_channels, out_channels, stride=1):
    super(BasicBlock, self).__init__()
    self.conv1 = nn.Conv2d(in_channels, out_channels,
     ↪  kernel_size=3, stride=stride, padding=1, bias=False)
    self.bn1 = nn.BatchNorm2d(out_channels)
    self.conv2 = nn.Conv2d(out_channels, out_channels,
     ↪  kernel_size=3, stride=1, padding=1, bias=False)
    self.bn2 = nn.BatchNorm2d(out_channels)

    self.shortcut = nn.Sequential()
    if stride != 1 or in_channels != out_channels:
        self.shortcut = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=1,
             ↪  stride=stride, bias=False),
            nn.BatchNorm2d(out_channels)
        )

def forward(self, x):
    out = F.relu(self.bn1(self.conv1(x)))
    out = self.bn2(self.conv2(out))
    out += self.shortcut(x)
    out = F.relu(out)
    return out


class ResNet(nn.Module):
def __init__(self, block, num_blocks, num_classes=4):
    super(ResNet, self).__init__()
    self.in_channels = 64
    self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3, bias=False)
    self.bn1 = nn.BatchNorm2d(64)
    self.layer1 = self._make_layer(block, 64, num_blocks[0], stride=1)
    self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
    self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
    self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)
    self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))
    self.linear = nn.Linear(512, num_classes)

def _make_layer(self, block, out_channels, num_blocks, stride):
    strides = [stride] + [1] * (num_blocks - 1)
    layers = []
    for stride in strides:
        layers.append(block(self.in_channels, out_channels, stride))
        self.in_channels = out_channels
    return nn.Sequential(*layers)

def forward(self, x):
    out = F.relu(self.bn1(self.conv1(x)))
    out = F.max_pool2d(out, kernel_size=3, stride=2, padding=1)
    out = self.layer1(out)
    out = self.layer2(out)
    out = self.layer3(out)
    out = self.layer4(out)
    out = self.avg_pool(out)
    out = out.view(out.size(0), -1)
    out = self.linear(out)
    return out


def ResNet18(num_classes):
    return ResNet(BasicBlock, [2, 2, 2, 2], num_classes)
```

4.3.2 Data preparation for classification

In the study conducted by Nežerka et al. (2024), the classification algorithms for construction and demolition waste were trained using images cropped to 200×200 pixels from larger originals. The classification process entailed the placement of as many 200×200 pixel squares within the region of interest (ROI) as feasible, with the final classification being determined by the most frequently occurring prediction across all squares within the ROI. This method, while systematic, potentially omits crucial information by not considering the entire ROI in the neural network's input, which could lead to less effective classification outcomes.

For our model, we opted to enhance the input methodology by first creating a bounding box based on predicted masks, then cropping the original image to this bounding box. Subsequently, these cropped and masked images were resized to 256×256 pixels to align with the predetermined input dimensions for our ResNet model. The visual representation of the ResNet input data, post-resizing, is depicted in Figure 4.8.

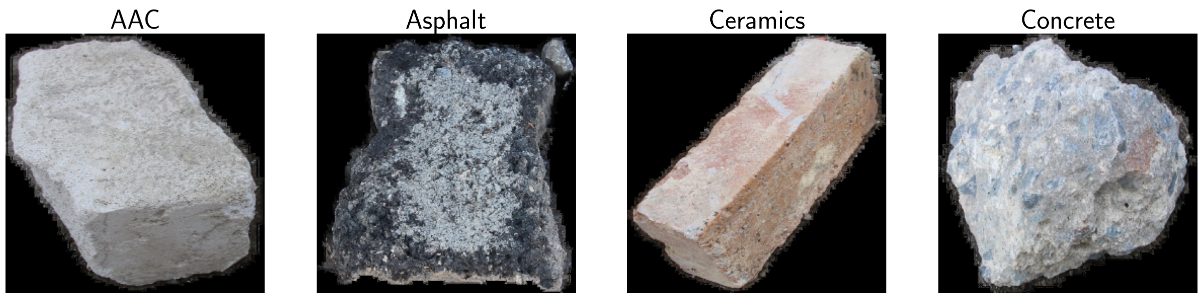

Figure 4.8: Input data for ResNet model.

4.3.3 Performance of the ResNet model

Similar to the U-Net model, the dataset for the ResNet model comprised 4266 images, with half subjected to augmentation to enhance model robustness. We utilized a ResNet-18 architecture due to its balance of depth and computational efficiency. The model training parameters were configured with a batch size of 16 and four parallel data loaders ('num_workers=4'). We employed the cross-entropy loss function coupled with the Adam optimizer, setting the learning rate to 0.001. The training regimen spanned 90 epochs to ensure sufficient model convergence.

As depicted in Figures 4.9 and 4.10, the ResNet model initially exhibited instability, which stabilized by epoch 55. Optimal validation performance was achieved at epoch 71, with an accuracy of 97.2%. Additionally, this epoch registered the second-highest performance on the test dataset with an accuracy of 94.9%, while the highest test accuracy occurred at epoch 61 with 95.3%. Analysis of the confusion matrix from epoch one, shown in Figure 4.12, indicates that the ResNet model accurately classifies asphalt and ceramics with minimal error. The primary confusion occurs between AAC

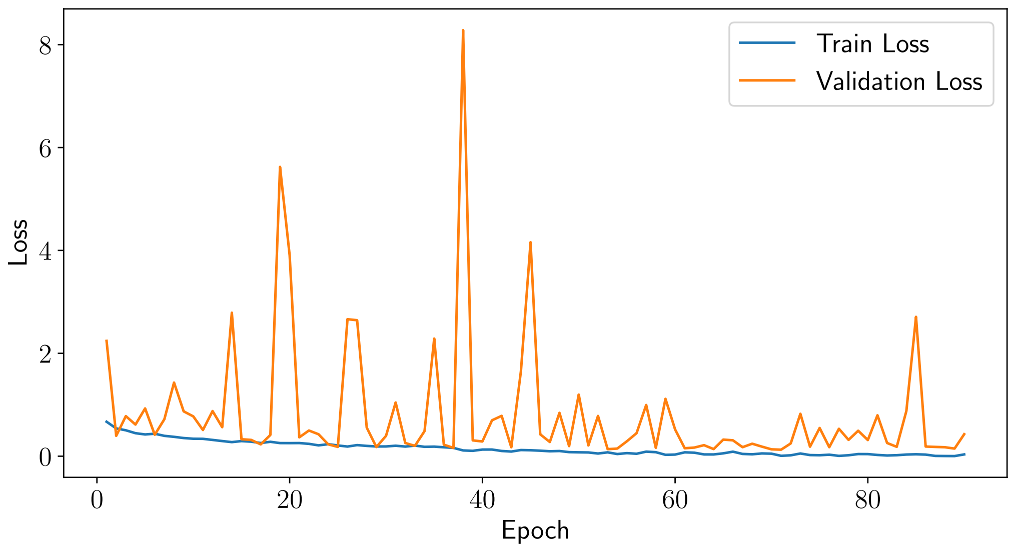

Figure 4.9: Training and validation loss as a function of epoch reported during the training of the ResNet model.

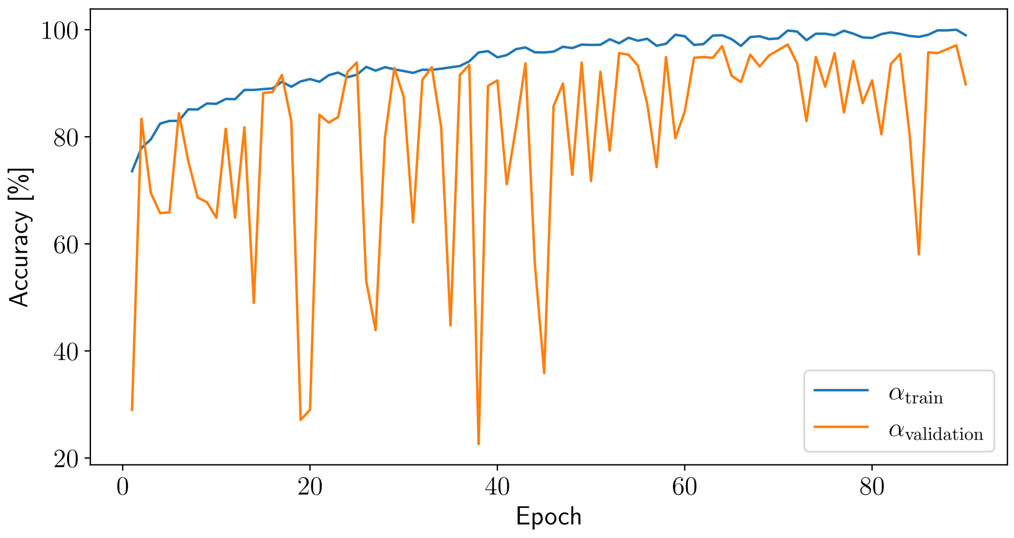

Figure 4.10: Training and validation accuracy as a function of epoch reported during the training of the ResNet model.

and concrete, marking an improvement over previous studies. Significant enhancements in overall accuracy are also documented in Table 4.2.

Model	Accuracy (%)
CNN	85.9%
GB	92.3%
MLP	91.3%
ResNet	**95.3%**

Table 4.2: Accuracy comparison for models introduced in previous study by Nežerka et al. (2024) and ResNet model.

Correct and incorrect prediction for AAC is visualized in figure 4.11. Higher probabilities were assigned to the correct predictions than to the incorrect ones, allowing to identify predictions that are suspicious to be incorrect ones and reclassify them using a DT model based on feature extraction.

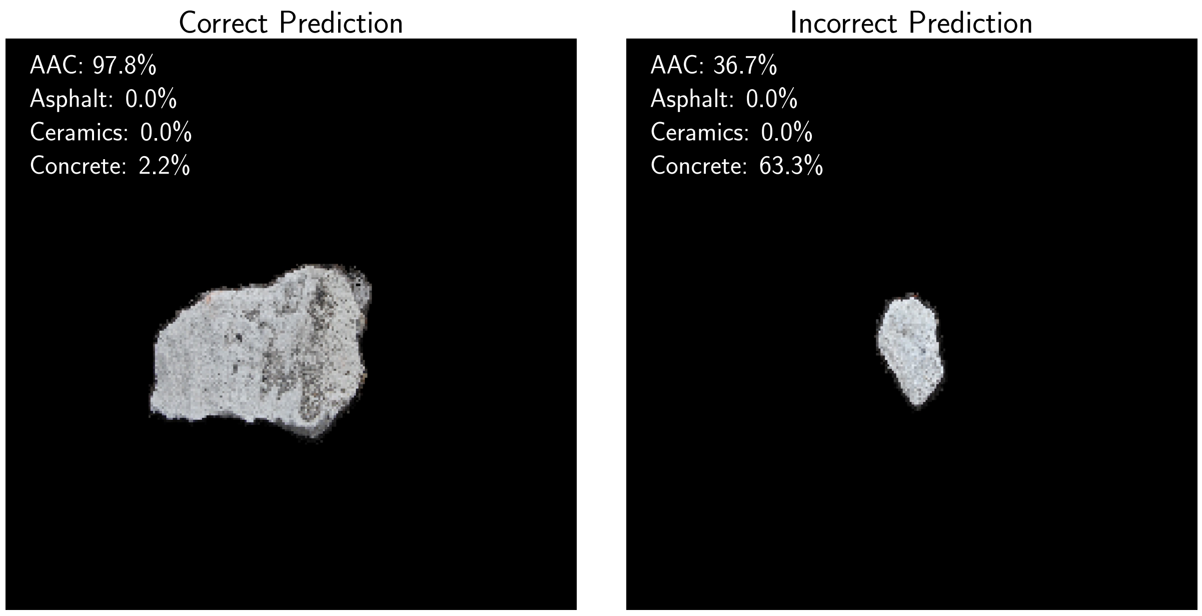

Figure 4.11: Predicted probabilities for correct and incorrect predictions for AAC.

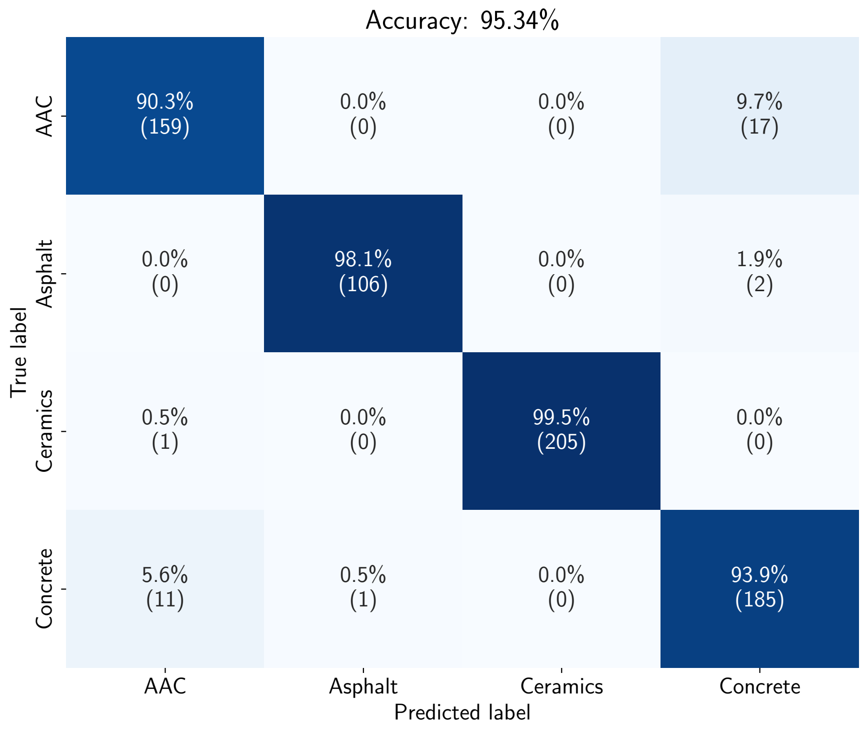

Figure 4.12: Confusion matrix for model in epoch 61.

References

J. Antoš, V. Nežerka, and M. Somr. Assessment of 2D-DIC stochastic patterns. *Acta Polytechnica CTU Proceedings*, 13:1–10, 2017. doi: 10.14311/app.2017.13.0001.

M. Arifuzzaman, M. R. Hasan, T. J. Toma, S. B. Hassan, and A. K. Paul. An advanced decision tree-based deep neural network in nonlinear data classification. *Technologies*, 11:24, 2023. doi: 10.3390/technologies11010024.

F. Bosché, M. Ahmed, Y. Turkan, C. T. Haas, and R. Haas. The value of integrating scan-to-BIM and scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components. *Automation in Construction*, 49:201–213, 2015. doi: 10.1016/j.autcon.2014.05.014.

A. Braun and A. Borrmann. Combining inverse photogrammetry and BIM for automated labeling of construction site images for machine learning. *Automation in Construction*, 106:102879, 2019. doi: 10.1016/j.autcon.2019.102879.

A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11, 2020. doi: 10.3390/info11020125.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. doi: 10.1007/bf00994018.

K. Crockett, J. O'Shea, W. Khan, and Z. Bandar. A hybrid model combining neural networks and decision tree for comprehension detection. In *Neural Networks (IJCNN)*. IEEE, 2018. doi: 10.1109/ijcnn.2018.8489621.

P. Davis, F. Aziz, M. T. Newaz, W. Sher, and L. Simon. The classification of construction waste material using a deep convolutional neural network. *Automation in Construction*, 122:103481, 2021. doi: 10.1016/j.autcon.2020.103481.

F. N. M. de Sousa Filho, V. G. Pereira de Sá, and E. Brigatti. Entropy estimation in bidimensional sequences. *Physical Review E*, 105:054116, 2022. doi: 10.1103/physreve.105.054116.

A. Dimitrov and M. Golparvar-Fard. Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections. *Advanced Engineering Informatics*, 28: 37–49, 2014. doi: 10.1016/j.aei.2013.11.002.

Z. Dong, J. Chen, and W. Lu. Computer vision to recognize construction waste compositions: A novel boundary-aware transformer (BAT) model. *Journal of Environmental Management*, 305:114405, 2022. doi: 10.1016/j.jenvman.2021.114405.

C. Firestone. Performance vs. competence in human-machine comparisons. *Proceedings of the National Academy of Sciences*, 117:26562–26571, 2020. doi: 10.1073/pnas.1905334117.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001. doi: 10.1214/aos/1013203451.

J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 2002. doi: 10.1016/s0167-9473(01)00065-2.

A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2022.

S. P. Gundupalli, S. Hait, and A. Thakur. A review on automated sorting of source-separated municipal solid waste for recycling. *Waste Management*, 60: 56–74, 2017. doi: 10.1016/j.wasman.2016.09.015.

K. K. Han and M. Golparvar-Fard. Appearance-based material classification for monitoring of operation-level construction progress using 4d BIM and site photologs. *Automation in Construction*, 53:44–57, 2015. doi: 10.1016/j.autcon.2015.02.007.

T. Hastie, R. Tibshirani, and J. Friedman. Boosting and additive trees. In *The Elements of Statistical Learning*, pages 337–387. 2008. doi: 10.1007/978-0-387-84858-7_10.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. doi: 10.1109/cvpr.2016.90.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. doi: 10.1162/neco.2006.18.7.1527.

R. Hlůžek, J. Trejbal, V. Nežerka, P. Demo, Z. Prošek, and P. Tesárek. Improvement of bonding between synthetic fibers and a cementitious matrix using recycled concrete powder and plasma treatment: from a single fiber to FRC. *European Journal of Environmental and Civil Engineering*, 26:3880–3897, 2020. doi: 10.1080/19648189.2020.1824821.

S. Y.-C. Ho, T.-W. Chien, M.-L. Lin, and K.-T. Tsai. An app for predicting patient dementia classes using convolutional neural networks (CNN) and artificial neural networks (ANN): Comparison of prediction accuracy in microsoft excel. *Medicine*, 102:e32670, 2023. doi: 10.1097/md.0000000000032670.

J. D. L. H. Hoong, J. Lux, P.-Y. Mahieux, P. Turcry, and A. Aït-Mokhtar. Determination of the composition of recycled aggregates using a deep learning-based image analysis. *Automation in Construction*, 116:103204, 2020. doi: 10.1016/j.autcon.2020.103204.

T. Joensuu, H. Edelman, and A. Saari. Circular economy practices in the built environment. *Journal of Cleaner Production*, 276:124215, 2020. doi: 10.1016/j.jclepro.2020.124215.

Z. N. Khudhair, A. N. Khdiar, N. K. El Abbadi, F. Mohamed, T. Saba, F. S. Alamri, and A. Rehman. Color to grayscale image conversion based on singular value decomposition. *IEEE Access*, 11:54629–54638, 2023. doi: 10.1109/access.2023.3279734.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. doi: 10.1109/iccv51070.2023.00371.

P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015. doi: 10.1109/iccv.2015.172.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90, 2017. doi: 10.1145/3065386.

Y. Ku, J. Yang, H. Fang, W. Xiao, and J. Zhuang. Deep learning of grasping detection for a robot used in sorting construction and demolition waste. *Journal of Material Cycles and Waste Management*, 23:84–95, 2020. doi: 10.1007/s10163-020-01098-z.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. doi: 10.1038/nature14539.

S. Liang and Y. Gu. A deep convolutional neural network to simultaneously localize and recognize waste types in images. *Waste Management*, 126:247–257, 2021. doi: 10.1016/j.wasman.2021.03.017.

K. Lin, T. Zhou, X. Gao, Z. Li, H. Duan, H. Wu, G. Lu, and Y. Zhao. Deep convolutional neural networks for construction and demolition waste classification: VGGNet structures, cyclical learning rate, and knowledge transfer. *Journal of Environmental Management*, 318:115501, 2022. doi: 10.1016/j.jenvman.2022.115501.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. doi: 10.1109/cvpr.2015.7298965.

W. Lu and J. Chen. Computer vision for solid waste sorting: A critical review of academic research. *Waste Management*, 142:29–43, 2022. doi: 10.1016/j.wasman.2022.02.009.

H. Mahami, N. Ghassemi, M. T. Darbandy, A. Shoeibi, S. Hussain, F. Nasirzadeh, R. Alizadehsani, D. Nahavandi, A. Khosravi, and S. Nahavandi. Material recognition for automated progress monitoring using deep learning methods, 2020.

K. P. Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

V. Nežerka, J. Trejbal, and T. Zbíral. Dataset of construction and demolition waste images: aerated autoclaved concrete (AAC), asphalt, ceramics, and concrete, Feb. 2023.

V. Nežerka and J. Trejbal. Assessment of aggregate-bitumen coverage using entropy-based image segmentation. *Road Materials and Pavement Design*, pages 1–12, 2019. doi: 10.1080/14680629.2019.1605304.

V. Nežerka, Z. Prošek, J. Trejbal, J. Pešta, J. Ferriz-Papi, and P. Tesárek. Recycling of fines from waste concrete: Development of lightweight masonry blocks and assessment of their environmental benefits. *Journal of Cleaner Production*, 385: 135711, 2023. doi: 10.1016/j.jclepro.2022.135711.

V. Nežerka, T. Zbíral, and J. Trejbal. Machine-learning-assisted classification of construction and demolition waste fragments using computer vision: Convolution versus extraction of selected features. *Expert Systems with Applications*, 238: 121568, 2024. doi: 10.1016/j.eswa.2023.121568.

M. S. Nixon and A. S. Aguado. Image processing. In *Feature Extraction and Image Processing for Computer Vision*, pages 83–139. Elsevier, 2020. doi: 10.1016/b978-0-12-814976-8.00003-8.

B. I. Oluleye, D. W. Chan, A. B. Saka, and T. O. Olawumi. Circular economy research on building construction and demolition waste: A review of current trends and future research directions. *Journal of Cleaner Production*, 357:131927, 2022. doi: 10.1016/j.jclepro.2022.131927.

K. Özkan, S. Ergin, Ş. Işık, and İ. Işıklı. A new classification scheme of plastic wastes based upon recycling labels. *Waste Management*, 35:29–35, 2015. doi: 10.1016/j.wasman.2014.09.030.

O. Ozturk, B. Sarıtürk, and D. Z. Seker. Comparison of fully convolutional networks (FCN) and U-Net for road segmentation from high resolution imageries. *International journal of environment and geoinformatics*, 7:272–279, 2020. doi: 10.30897/ijegeo.737993.

B. Pan, Z. Lu, and H. Xie. Mean intensity gradient: An effective global parameter for quality assessment of the speckle patterns used in digital image correlation. *Optics and Lasers in Engineering*, 48:469–477, 2010. doi: 10.1016/j.optlaseng.2009.08.010.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

S. M. Piryonesi and T. E. El-Diraby. Using machine learning to examine impact of type of performance indicator on flexible pavement deterioration modeling. *Journal of Infrastructure Systems*, 27:04021005, 2021. doi: 10.1061/(asce)is.1943-555x.0000602.

Z. Prošek, J. Trejbal, V. Nežerka, V. Goliáš, M. Faltus, and P. Tesárek. Recovery of residual anhydrous clinker in finely ground recycled concrete. *Resources, Conservation and Recycling*, 155:104640, 2020. doi: 10.1016/j.resconrec.2019.104640.

X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. Zaiane, and M. Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.

X. Qin, H. Dai, X. Hu, D.-P. Fan, L. Shao, and L. V. Gool. Highly accurate dichotomous image segmentation. In *Lecture Notes in Computer Science*, 2022. doi: 10.1007/978-3-031-19797-0_3.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241, 2015. doi: 10.1007/978-3-319-24574-4_28.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/323533a0.

S. L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16:235–240, 1994. doi: 10.1007/bf00993309.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:623–656, 1948. doi: 10.1002/j.1538-7305.1948.tb00917.x.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

H. Son, C. Kim, and C. Kim. Automated color model-based concrete detection in construction-site images by using machine learning algorithms. *Journal of Computing in Civil Engineering*, 26:421–433, 2012. doi: 10.1061/(asce)cp.1943-5487.0000141.

Y. Su. Multi-agent evolutionary game in the recycling utilization of construction waste. *Science of The Total Environment*, 738:139826, 2020. doi: 10.1016/j.scitotenv.2020.139826.

J. Valentin, J. Trejbal, V. Nežerka, T. Valentová, and M. Faltus. Characterization of quarry dusts and industrial by-products as potential substitutes for traditional fillers and their impact on water susceptibility of asphalt concrete. *Construction and Building Materials*, 301:124294, 2021. doi: 10.1016/j.conbuildmat.2021.124294.

T. Vincent, M. Guy, P. Louis-César, B. Jean-François, and M. Richard. Physical process to sort construction and demolition waste (c&dw) fines components using process water. *Waste Management*, 143:125–134, 2022. doi: 10.1016/j.wasman.2022.02.012.

Z. Wang, H. Li, and X. Zhang. Construction waste recycling robot for nails and screws: Computer vision technology and neural network approach. *Automation in Construction*, 97:220–228, 2019a. doi: 10.1016/j.autcon.2018.11.009.

Z. Wang, B. Peng, Y. Huang, and G. Sun. Classification for plastic bottles recycling based on image recognition. *Waste Management*, 88:170–181, 2019b. doi: 10.1016/j.wasman.2019.03.032.

Z. Wang, H. Li, and X. Yang. Vision-based robotic system for on-site construction and demolition waste sorting and recycling. *Journal of Building Engineering*, 32: 101769, 2020. doi: 10.1016/j.jobe.2020.101769.

Y. Wu, J. P. Noonan, and S. Agaian. A novel information entropy based randomness test for image encryption. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2011. doi: 10.1109/icsmc.2011.6084076.

Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan. Local shannon entropy measure with statistical tests for image randomness. *Information Sciences*, 222:323–342, 2013. doi: 10.1016/j.ins.2012.07.049.

W. Xiao, J. Yang, H. Fang, J. Zhuang, and Y. Ku. Development of online classification system for construction waste based on industrial camera and hyperspectral camera. *PLOS ONE*, 14:e0208706, 2019. doi: 10.1371/journal.pone.0208706.

W. Xiao, J. Yang, H. Fang, J. Zhuang, and Y. Ku. Classifying construction and demolition waste by combining spatial and spectral features. *Proceedings of the Institution of Civil Engineers—Waste and Resource Management*, 173:79–90, 2020. doi: 10.1680/jwarm.20.00008.

J. Yang, Z. Zeng, K. Wang, H. Zou, and L. Xie. GarbageNet: A unified learning framework for robust garbage classification. *IEEE Transactions on Artificial Intelligence*, 2:372–380, 2021. doi: 10.1109/tai.2021.3081055.

L. Yuan, J. Guo, and Q. Wang. Automatic classification of common building materials from 3d terrestrial laser scan data. *Automation in Construction*, 110:103017, 2020. doi: 10.1016/j.autcon.2019.103017.

L. Yuan, W. Lu, and F. Xue. Estimation of construction waste composition based on bulk density: A big data-probability (BD-p) model. *Journal of Environmental Management*, 292:112822, 2021. doi: 10.1016/j.jenvman.2021.112822.

T. Zbíral and V. Nežerka. Python codes for machine-learning-based classification of construction and demolition waste fragments, Feb. 2023.

L. Zheng, H. Wu, H. Zhang, H. Duan, J. Wang, W. Jiang, B. Dong, G. Liu, J. Zuo, and Q. Song. Characterizing the generation and flows of construction and demolition waste in China. *Construction and Building Materials*, 136:405–413, 2017. doi: 10.1016/j.conbuildmat.2017.01.055.

S. Zhou, Q. Chen, and X. Wang. Convolutional deep networks for visual data classification. *Neural Processing Letters*, 38:17–27, 2012. doi: 10.1007/s11063-012-9260-y.

Z.-H. Zhou. Decision trees. In *Machine Learning*, pages 79–102. Springer Singapore, 2021. doi: 10.1007/978-981-15-1967-3_4.

Z. Zhu and I. Brilakis. Parameter optimization for automated concrete detection in image data. *Automation in Construction*, 19:944–953, 2010. doi: 10.1016/j.autcon.2010.06.008.