**Utilizing Deep-Feature-Based Method to Classify Marine Debris**

**Abstract:**

The rising quantity of contaminants entering seas and other bodies of water has been identified as a serious environmental, economic, and social concern by the worldwide community.

Aside from prevention, one of the most important steps in tackling marine pollution is the removal of trash currently existing in marine habitats. Machine learning (ML) and deep learning (DL) approaches may be used to automate marine trash removal and make the cleanup process more efficient. This study compares the performance of six well-known deep convolutional neural networks (CNNs), namely VGG19, InceptionV3, ResNet50, Inception-ResNetV2, DenseNet121, and MobileNetV2, when used as feature extractors for the identification and classification of underwater marine debris using three different extraction schemes. The performance of a neural network (NN) classifier built on top of deep CNN feature extractors is compared when the feature extractor is (1) fixed; (2) fine-tuned on the given task; and (3) fixed during the first phase of training and fine-tuned thereafter. Fine-tuning produced better-performing models in general, but it was considerably more computationally costly. The fine-tuned Inception-ResNetV2 feature extractor had the greatest overall NN performance, with an accuracy of 91.40 percent and an F1-score of 92.08 percent, followed by the fine-tuned InceptionV3 extractor. We also look at how well traditional machine learning classifiers perform when trained on deep CNN features. Finally, we show that substituting NN with a traditional ML classifier like support vector machine (SVM) or logistic regression (LR) may improve classification performance on fresh data even further.

1. **Introduction**

In today's culture, rising levels of marine pollution are becoming a significant environmental issue. Various trash products contaminate our oceans, seas, and other water bodies, endangering coastal animals, habitat, human safety, and the economic health of coastal towns [1]. Fishing gear that has been discarded continues to capture and destroy marine creatures. Plastic trash is frequently mistaken for food by animals such as seabirds and turtles due to its similar look and odor, resulting in malnutrition and hunger [2,3]. Furthermore, seafood polluted with microplastics, for example, tiny plastic particles less than 5 mm in size, is potentially hazardous to people [4–6]. In addition, marine debris has a significant economic impact in businesses including tourism, aquaculture, and fisheries [7]. Detailed knowledge of the magnitudes, sources, and impacts of marine debris, general behavioral change, enhancement of the circular economy, prevention of waste items entering the marine environment, waste generation reduction, and removal of debris already present in the marine environment are all key measures for solving marine pollution [8]. Marine trash identification and cleaning, particularly that which is deep beneath the water's surface, is difficult and costly. As a result, automated detection and removal of marine trash is sought to make the cleanup procedure more efficient. Autonomous underwater vehicles (AUVs) and deep-learning-based visual detection of underwater garbage can help with the latter.

Object recognition and localization [9,10], classification [11,12], and semantic segmentation [13,14] are only a few of the computer vision applications where deep learning approaches have proven effective. Deep learning approaches automatically discover hidden data representations from raw data [15] and provide end-to-end learning processes, unlike traditional machine learning techniques, which need domain-specific hand-engineered feature extraction. When it comes to picture and video data, convolutional neural networks (CNNs) [16] are one of the major approaches contributing to the success of deep learning. CNNs are meant to learn features hierarchically by combining lower-level characteristics into higher-level ones, and its architectural design is influenced by receptive field structures in the animal visual cortex [17,18]. Convolutional, pooling, and fully linked layers are the three fundamental components of CNN architecture. Convolutional kernels are slid through all spatial locations using a predetermined stride, and dot-products between kernel weights and tiny local patches in the input volume are computed to produce feature maps. Each convolutional kernel generates a unique feature map. Lower convolutional layers extract general characteristics like lines and edges, whereas higher convolutional layers encode more sophisticated, higher-level information. The element-wise application of a nonlinear activation function, such as the rectified linear unit (ReLU) [19], to generated feature maps introduces nonlinearity, which is useful for identifying nonlinear features [20]. To provide invariance to minor shifts and distortions, pooling layers minimize the input's spatial size by replacing local patches in input feature maps with their maximum or mean value. To conduct high-level reasoning, some CNN architectures include extra fully connected layers on top of the stacked convolutional and pooling layers before the final softmax layer [21,22]. This study uses characteristics derived from existing state-of-the-art deep convolutional architectures to solve the challenge of automated, image-based maritime debris classification and identification.

Several studies have used deep-learning-based approaches to solve the challenge of detecting, classifying, and quantifying marine trash in recent years. [23] uses convolutional neural network architecture to automatically detect maritime trash in forward-looking sonar images. To recognise garbage objects in aquatic environments, a single-stage RetinaNet detector trained on non-aquatic waste pictures is employed in [24]. Kylili et al. use the VGG16 convolutional model architecture to classify floating macro-plastic marine debris images into three categories (bottle, bucket, and straw) [25] and differentiate between six types of plastic debris, one type of marine life, and other items encountered at the shoreline or in the seawater [26]. In a realistic underwater setting, Fulton et al. [27] test four alternative neural network topologies for visual identification of plastic trash. Plastic, all man-made things purposefully placed in the environment, and all biological stuff were all trained to identify in underwater recordings by models. Politikos et al. [28] employ a region-based CNN to identify seabed marine trash on an imagined acquisition in Ermoupolis bay, Syros Island, Greece. Musi c et al. [29] conducted an initial study on the performance of neural networks for the detection and classification of underwater sea litter images. The dataset was built using images from the Internet and hybrid images generated using a Blender environment from given background images and 3D litter models, and the neural networks were trained and tested on it. [30] uses machine learning methods, including CNNs, to automatically categorise pictures of five different kinds of microplastic particles found on Canary Islands beaches. [31,32] discusses the use of unmanned aerial vehicles and deep learning algorithms to quantify marine trash on beaches.

Automated identification of maritime trash is fraught with difficulties. To begin, there are several different forms of marine debris, each having substantial differences within each class. Plastic bottles, bags, and cups, for example, come in a variety of forms yet all belong to the same category of marine trash plastics. Second, regardless of the recording angle, water turbidity, or lighting, waste items should be identified in various deterioration phases. Furthermore, deep neural network training necessitates a significant volume of labelled data. **It is expensive and not always possible to get a sufficiently big and varied dataset for image-based marine debris categorization. Transfer learning, or knowledge transfer from a source to a desired target domain, is a popular and effective technique for overcoming a lack of labelled training data [33]. On a new job of interest, the aim is to reuse characteristics acquired by the network on a very large dataset, such as the ImageNet [34] dataset, which comprises 1.2 million training pictures from 1000 classes.**

**Transfer learning is usually done in one of two ways: (1) a fixed feature extractor using a pre-trained network [35,36]; (2) fine-tuning the weights of a pre-trained network on a target dataset using a pre-trained network [37,38]. The last, typically completely linked, component of the network is eliminated in the first method, while the rest of the network (convolutional basis) remains fixed during training. A new classifier, either a new neural network or a traditional machine learning classifier like a support vector machine or a logistic regression classifier, is added on top of the fixed base used for feature extraction to learn underlying patterns from extracted features and discriminate data based on them.** The pre-trained network weights are used as a weight initialization technique in the fine-tuning approach. To enhance the final generalization performance on a new task of interest, these weights are changed during training by backpropagating mistakes from the target task into the base network [37]. Transfer learning has been used successfully in a variety of computer vision applications, including medical image analysis [39–42], agriculture [43,44], remote sensing [45,46], and the textile sector [47]. Several studies on the detection and categorization of maritime debris have used transfer learning. In [25,26], the VGG16 network, which was pre-trained on the ImageNet dataset, is utilized as a fixed feature extractor for the categorization of plastic marine trash. The suggested RetinaNet model with ResNet50 backbone is used in [24] to detect trash objects in aquatic environments using object-detection-based transfer learning. [27] fine-tunes successful object detection networks, including YOLOv2, Tiny-YOLO, Faster RCNN, and Single Shot MultiBox Detector (SSD), for underwater identification of three kinds of items, including plastic marine waste. Politikos et al. [28] employ MobileNetV1 architecture pre-trained on the COCO detection dataset as a backbone for their Mask R-CNN architecture for seabed litter detection.

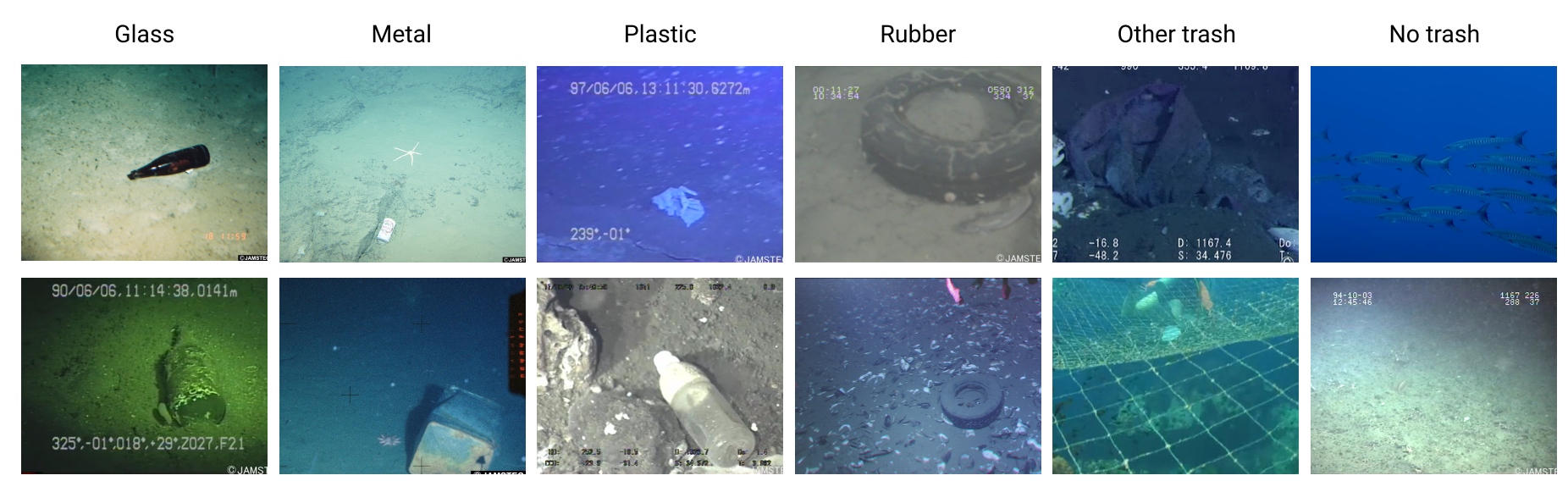
The main goals of this study were to: (1) develop the model for autonomous marine debris identification and classification of different types of marine debris; (2) compare the performance of prominent deep convolutional architectures, including VGG19, InceptionV3, ResNet50, Inception-ResNetV2, DenseNet121, and MobileNetV2, on the task of marine debris classification; (3) investigate different schemes to utilize transfer learning for marine debris classification: fixed extraction of features, fine-tuning, and combination of both; (4) compare the performance of conventional machine learning classifiers trained on feature vectors extracted by deep convolutional architectures.

The remainder of the paper is organised in the following manner. The dataset (Section 2.1), developed deep convolutional architectures (Section 2.2), and machine learning methods are all described in Section 2 of this paper (Section 2.3). The implementation details and experimental settings are presented in Section 3. Section 4 presents the experimental data, which are then addressed in Section 5. Finally, in Section 6, there are some closing observations and suggestions for further work.

1. **Materials and Methods**

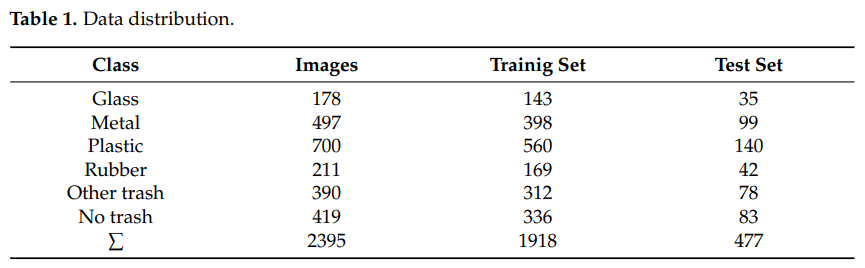
2.1 Dataset

A large annotated dataset of underwater trash is needed to utilize a deep-learning approach for marine debris detection and classification. The Japan Agency for Marine-Earth Science and Technology (JAMSTEC) has made their Deep-sea Debris Database, which contains numerous marine debris videos and photos, available online to the public [48]. In our work, we used data from the Deep-sea Debris Database complemented with Google Images. Images were manually labeled and validated by one of the researchers. Each image was visually inspected prior to being added to the dataset. The final dataset contains 2395 images from six different classes: glass, metal, plastic, rubber, other trash, and no trash. Figure 1 provides a sample of images from the dataset.



**Figure 1**. Sample of images from the dataset

Available data is split into a training set and a test set as follows to guarantee that training and test sets have the same distribution of considered classes: The final model evaluation was done with 20% of each class's pictures, while the other 80% were used for training. The data distribution in the original dataset, as well as the training and test subsets, is shown in Table 1.

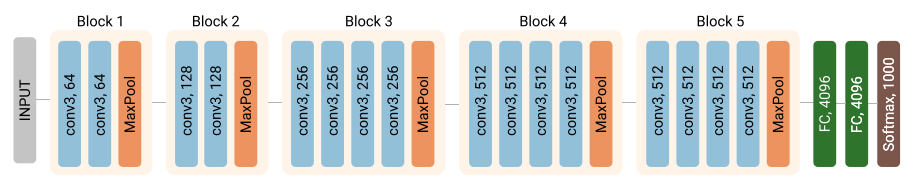


* 1. Deep Convolutional Architectures

Deep convolutional networks do better in picture classification problems than standard computer vision approaches. They also provide additional flexibility since they can be retrained with unique datasets and need less human expert analysis and fine-tuning [49]. This section discusses the six deep CNN architectures used in this study to classify marine trash and extract dense picture representations. Convolutional and spatial pooling layers are repeated in the following architectures: max pooling, average pooling, and global average pooling. To produce feature maps, convolutional layers convolve the input using a kernel shared across all input spatial regions. The feature map value at position (i, j) obtained with kth kernel (filter) is calculated as zi,j,k = , where wk and bk denote kth kernel’s weight vector and bias term, while xi,j denotes the input patch centered at (i, j). The nonlinear activation function g : R → R is applied element-wise on obtained feature maps to obtain activations ai,j,k = g(zi,j,k) [20]. The pooling layers aggregate information within feature map ak by replacing local pooling regions Ri,j in ak with the maximum element from Ri,j in the case of max pooling, and with arithmetic mean of elements in Ri,j in the case of average pooling. On the other hand, global average pooling averages all values in ak.

2.2.1. VGG19

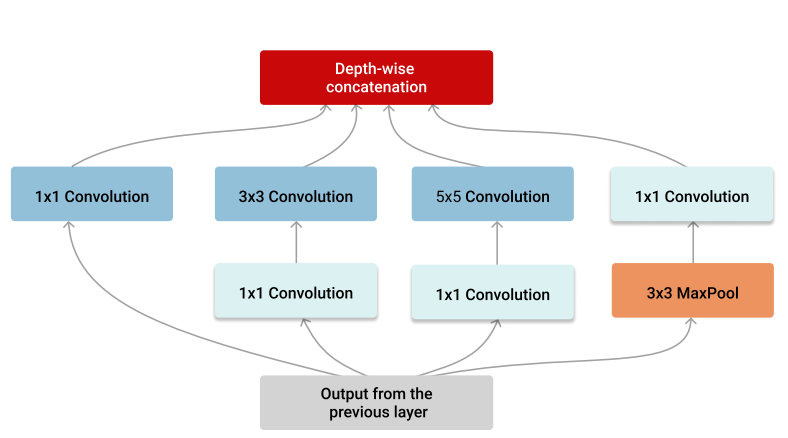
Simonyan et al. [22] presented a number of deep CNN architectures with varying numbers of weight layers. VGGNets are designs that stack convolutional layers with tiny 3 x 3 receptive fields in blocks before adding a max-pooling layer. They use 1 x 1 convolutions to increase the nonlinearity of the decision function without altering the receptive fields of convolutional layers in one network configuration. In this work, we employ VGG19 architecture that has 19 weight layers. The VGG19 architecture with ≈ 143.47M trainable parameters, as defined in [22], is illustrated in Figure 2.

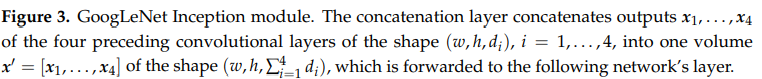


**Figure 2.** VGG19 network architecture. Input to the VGG19 network is 224 × 224 RGB image. Convolutional layers are labeled as “conv s, n”, where s denotes the filter size and n number of filters in a given layer. All layers, besides the last Softmax layer, use the ReLU activation function. MaxPooling is applied on 2 × 2 patches with a stride of 2. The feature maps produced by the max-pooling layer in Block 5 are flattened into a single 25088-dimensional vector and fed into fully connected layers.

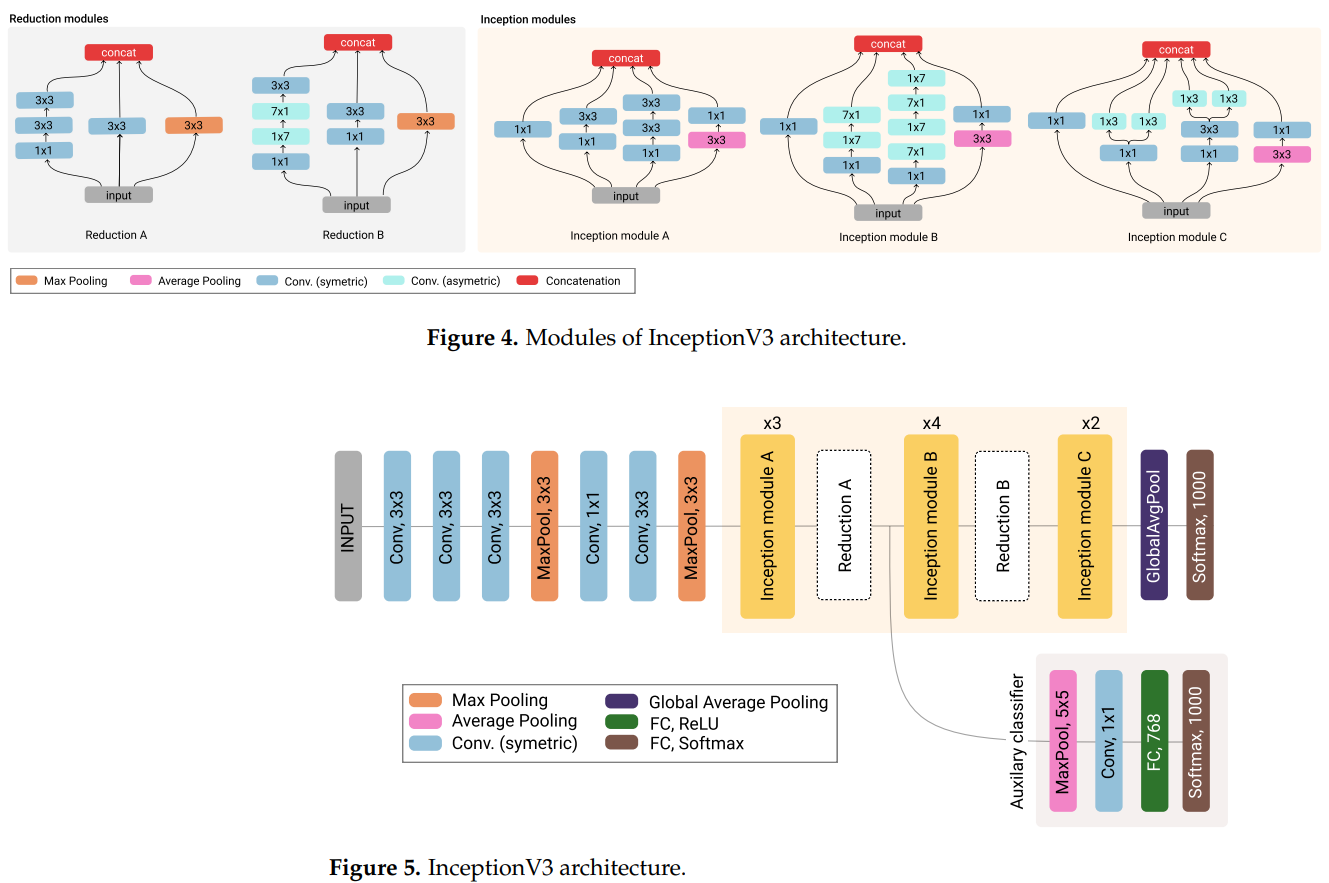
2.2.2. InceptionV3

Although compellingly architecturally simple and with good generalization performance, VGGNets come with high computational costs. To obtain an efficient deep model with reduced computational costs and high generalization ability, Szegedy et al. stack Inception modules in a 22 weight layer deep GoogLeNet architecture [50]. In contrast to conventional architectures where either convolution (with a single filter size) or pooling operation is performed, Inception modules perform convolution with different filter sizes (1 × 1, 3 × 3 and 5 × 5) in parallel along with the max-pooling operation and pass the concatenated results forward throughout the network. Varying filter sizes enable the model to capture spatial information at different scales at the same level in the network. The computational efficiency is preserved by adding an extra 1 × 1 convolution before expensive 3 × 3 and 5 × 5 convolutions and after the pooling layer, as illustrated in Figure 3. Two auxiliary classifiers are connected to intermediate Inception modules to propagate gradients through all network layers effectively.





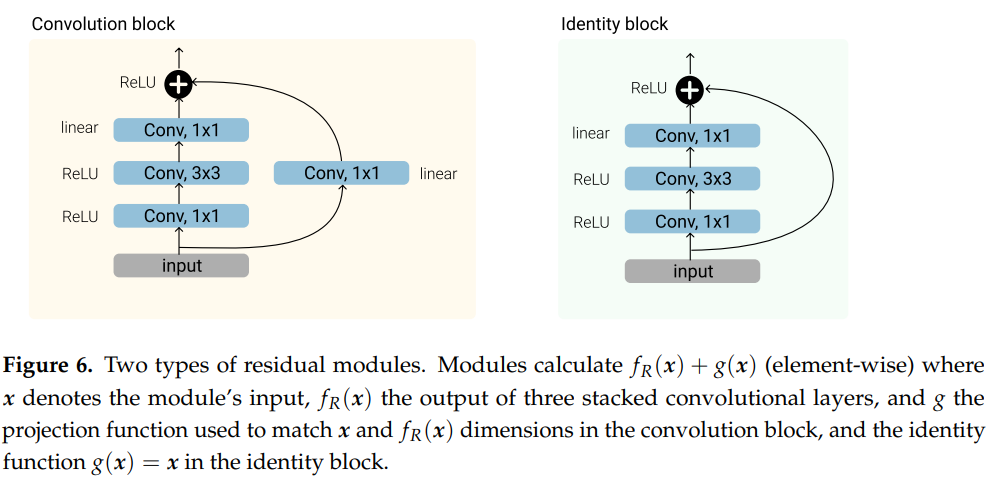
A later variant of the GoogLeNet architecture employed in this study, InceptionV3 architecture [51], modifies the original one in the following ways: (1) larger convolutions in the Inception modules are factorized into smaller ones, i.e., 5 × 5 convolution is replaced by two 3 × 3 convolutions in the Inception module A; (2) Inception module B factorizes symmetric 7 × 7 convolutions into asymmetric 1 × 7 and 7 × 1 convolutions; (3) the Inception module C, which is introduced for promotion of high dimensional representations, replaces 3 × 3 convolution with parallel asymmetric 1 × 3 and 3 × 1 convolutions; (4) one auxiliary classifier with employed batch normalization is used as a regularizer together with the label smoothing technique; (5) efficient size reduction with parallel convolutional and pooling blocks with a stride two is employed in the reduction modules. Figures 4 and 5 show modules and schematic representation of the InceptionV3 architecture. All convolutional layers employ batch normalization and use the ReLU activation function.



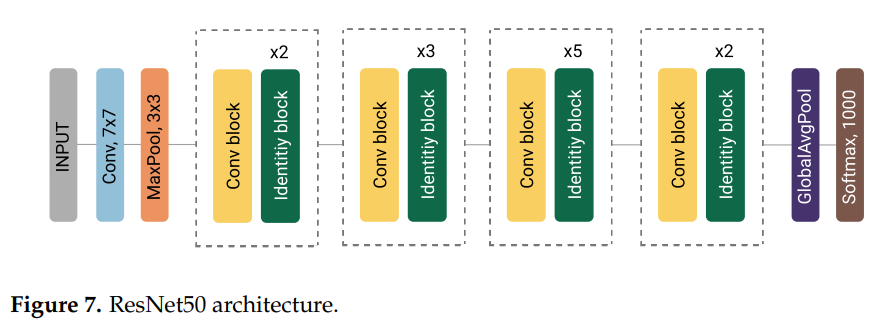
2.2.3. ResNet50

In [52], He et al. address the training accuracy degradation problem in deep network architectures by introducing the residual learning framework. Let f(x) be the underlying mapping to be learned with several network layers, where x denotes the input to the first of these layers. In the residual learning framework, stacked network layers learn the residual mapping fR(x) := f(x) − x, which is easier to optimize than underlying f(x). Original mapping f(x) now corresponds to fR(x) + x, which is realized with shortcut connections and element-wise additions in a feedforward convolutional network.

Residual Networks (ReNets) employ shortcut connections together with the batch normalization technique (after each convolution and before activation) to ease the training of deep network architectures and enjoy accuracy gains from an increase in network depth. Figure 6 shows two main residual modules, Convolution and Identity blocks, which comprise the 50/101/152 ResNet framework.

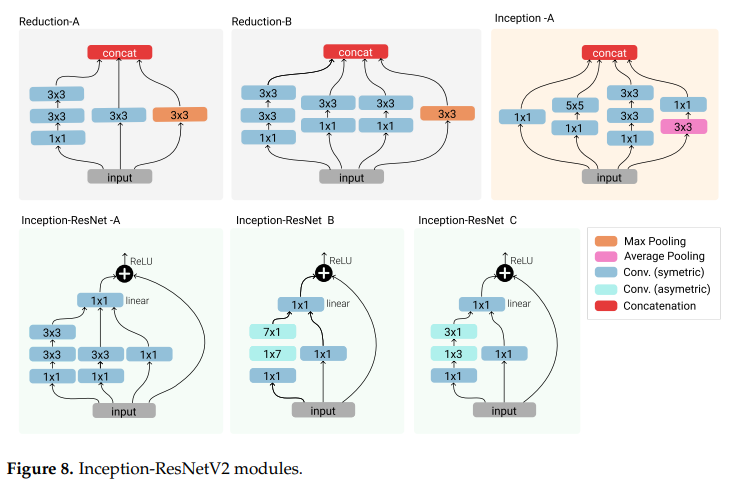


Identity shortcuts in identity blocks can only be used when input and output have the same dimension. Otherwise, a projection shortcut with 1 × 1 convolutions, i.e., convolution block from Figure 6, is used to match the input and output dimensions. The 50-layers deep ResNet50 architecture employed in this study is illustrated in Figure 7.

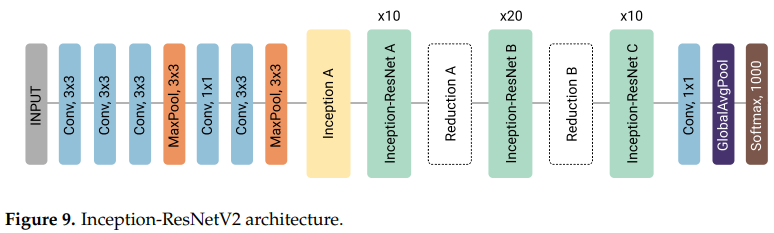


2.2.4. Inception-ResNetV2

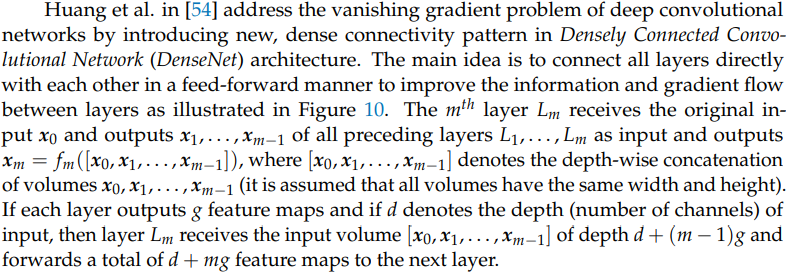
The Inception-ResNet [53] architecture combines good performing Inception architecture [51] with residual learning framework [52] by replacing the filter concatenation stage in Inception modules with residual connections. In residual versions of Inception modules (Inception-ResNet A, B and C modules in Figure 8), 1 × 1 convolution without activation, i.e., with linear activation g(x) = x, is added before the summation to scale up the dimensionality of a given volume to match the input depth.

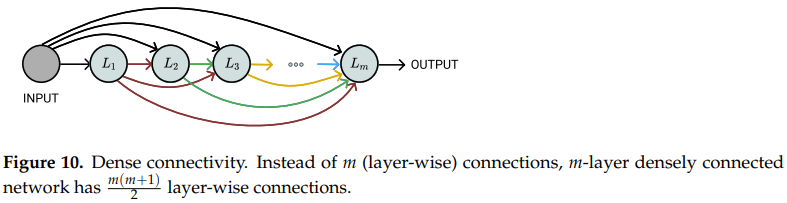


The batch normalization is utilized only on the top of traditional layers, not on summations. Introduction of residual connections into Inception architecture significantly accelerates the training of the Inception networks with an increased depth. However, the residual version of the Inception network is prone to instabilities during the training when a number of filters exceed 1000. To stabilize the training procedure, Inception-ResNets scale down the residuals before addition. Schematic representation of the Inception-ResNetV2 network architecture is shown in Figure 9.

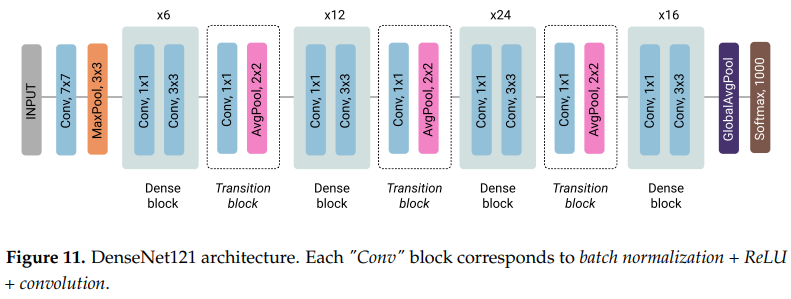


2.2.5. DenseNet121

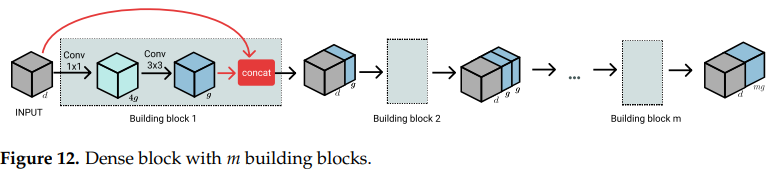




The architecture of densely connected network with 121 weight layers (excluding batch normalization layers), DenseNet121, is shown in Figure 11.



Four dense blocks are comprised of 6, 12, 24, and 16 smaller building blocks each comprised of 1 × 1 convolution, which reduces the number of the input features (added for computational efficiency), and 3 × 3 convolution, which produces g = 32 feature maps and concatenates it to the original input volume as illustrated in Figure 12. Transition layers following dense blocks aim to reduce the depth and spatial size of the input volume.

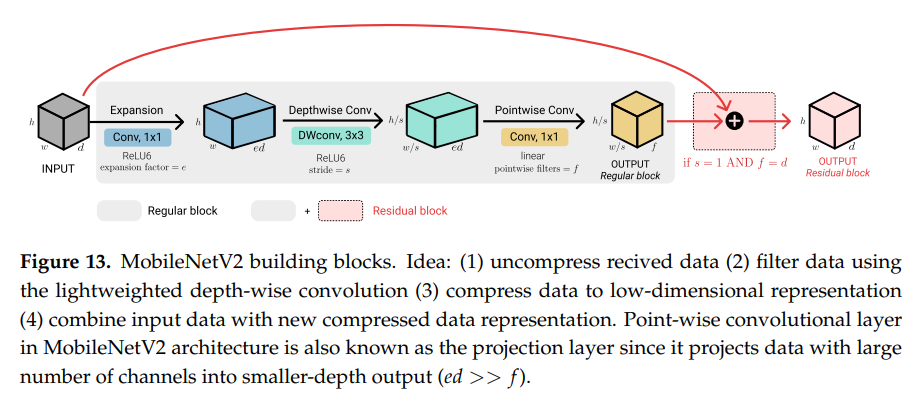


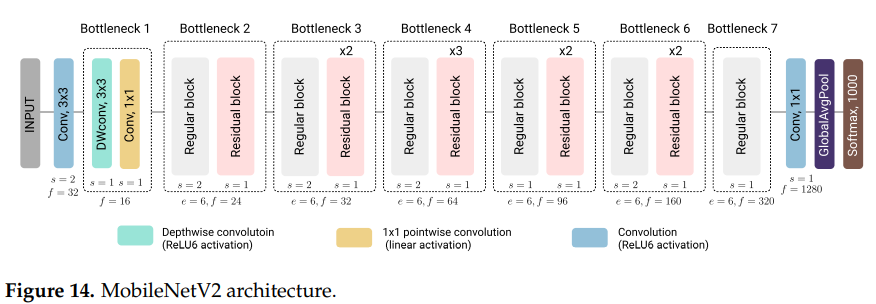
Reusing features learnt in previous layers allows the classifier to employ features of various levels of complexity, eliminating the need to train duplicate features and resulting in a narrower architecture with fewer parameters.

2.2.6 MobileNetV2

Lightweight MobileNet architectures are intended for mobile and embedded vision applications. The primary building block of MobileNetV1 [55] architecture is depth-wise separable convolution, which, unlike standard convolution, separates filtering and combining of input features into two distinct stages: (1) depth-wise convolution, which filters input features by applying a single convolution kernel per input channel; (2) point-wise 1 × 1 convolution used to linearly combine depth-wise convolution output channels into new features. Implemented 3 × 3 depth-wise separable convolutions require 8 to 9 times less computation compared to standard convolutions at the cost of a small reduction in accuracy [55].

MobileNetV2 [56] architecture upgrades ideas from its predecessor MobileNetV1. It retains depth-wise separable convolution as an efficient building block and introduces linear bottlenecks and shortcut connections into architecture. The MobileNet architecture utilizes the ReLU6 activation function, ReLU6(x) = min{max{x, 0}, 6} on all layers except on the linear 1 × 1 point-wise convolution layers colored in yellow in Figures 13 and 14, and on the final softmax layer. Inverted residual block characteristic for MobileNetV2 architecture, illustrated in Figure 13, adds narrow layers instead of traditional residuals that use expanded data representations.

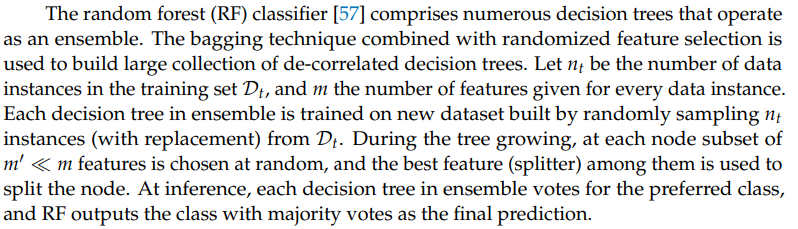




* 1. Machine Learning Classifiers

Traditional machine learning classifiers were employed to categorize marine debris pictures based on retrieved feature vectors in this part.

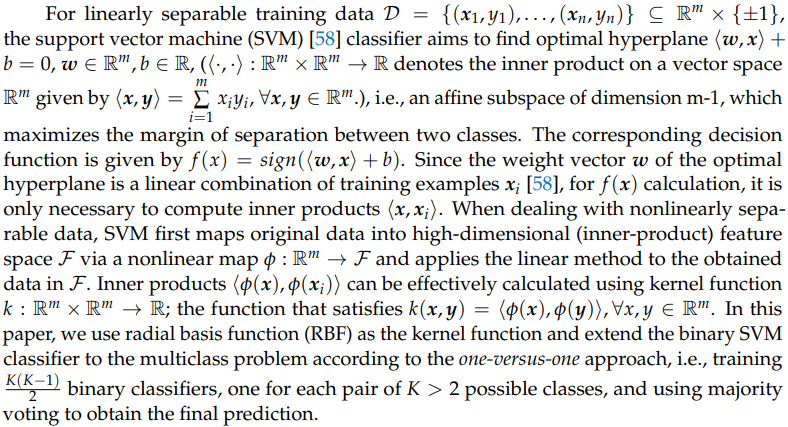
2.3.1 Random Forests



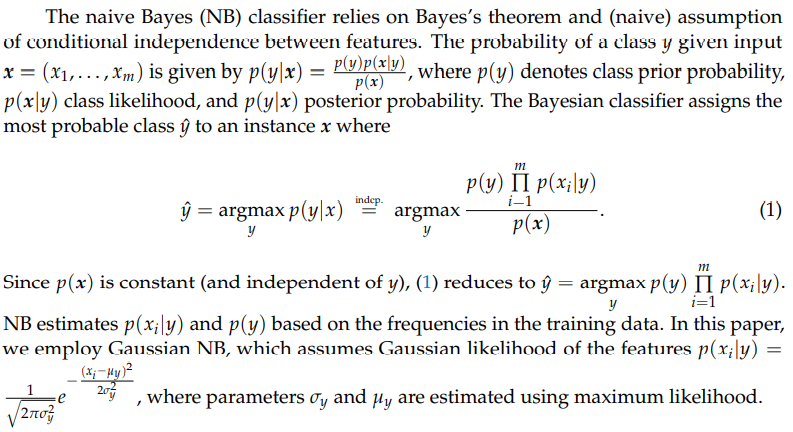
2.3.2. k-Nearest Neighbors

The k-Nearest Neighbors (kNN) method is a basic example of a lazy learning algorithm. It saves all of the training data and defers execution until a fresh data instance requires classification. The technique creates a new data point for the majority class of k training data instances that are closest to the received data point in terms of distance. Euclidean distance is the most often used distance measurement. Any other measure, on the other hand, may be used to calculate distance.

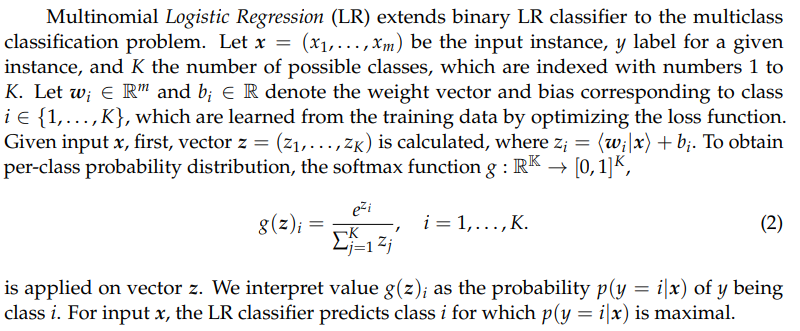
2.3.3. Support Vector Machines



2.3.4. Naïve Bayes



2.3.5. Logistic Regression



**3. Experiments**

3.1 Experimental Setup

In order to develop the deep model, Python 3.7.6 programming language combined with TensorFlow 2.1.0 [59] machine learning framework and also Tensorflow.KerasAPI [60]. Scikitlearn [61] machine learning library is also used by us to implement machine learning classifiers for the Python programming language.

3.2 Extraction of Features

In conventional CNNs, the feature maps from the last convolutional layer are flattened

into a feature vector. It is then pass on to completely linked layers and final softmax classification

layer [21,22]. Lin et al. [62] offer a substitute with Global Average Pooling layers, which dimensionally compresses the data exists in feature maps into a vector. Dense feature vectors for sea debris pictures are draw out using the state-of-the-art convolutional model architectures described in Section 2.2 and was illustrated in Figure 15. Every model’s last layer with 3D output is chosen as the feature extraction layer. Global Average Pooling is implemented to the output of that layer for a feature vector from a 3D output volume. Let w x h x n as the feature extraction layer’s output’s shape and let f1, f2, . . . , fn denote n feature maps in the output volume. By applying a Global Average Pooling on the given volume, we obtain a vector ( f 1, f 2, . . . , f n) where fi, represents the mean value of w x h values in the feature map fi, i = 1, . . . , n. Therefore, the feature vector’s size matches the depth of the output volume of the feature extraction layer.

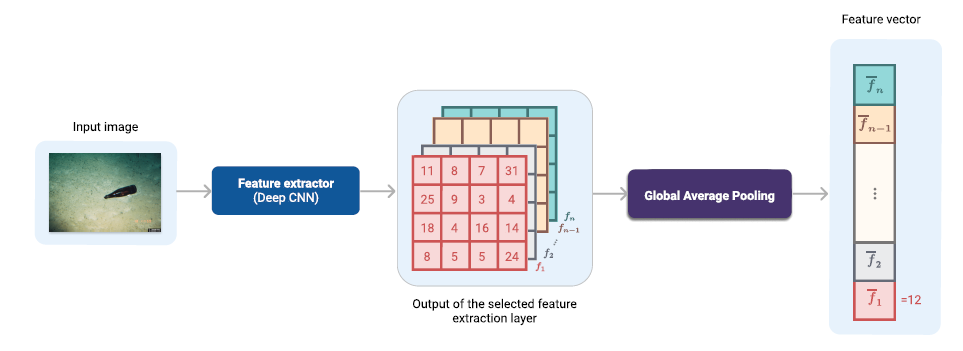


Figure 15. Extraction of dense feature vectors

All of the feature extraction deep models are filled with weights pre-trained on the ImageNet [34] dataset. All layers which follow the feature extraction layers are released and replaced by a Global Average Pooling for feature extraction in every deep CNN architecture. Table 2 shows all deep architectures involved in the process for feature extraction together with the information about the total parameters (the dropped layers after the feature extraction layer) and the size of the extracted vectors. Two fully connected (FC) layers consist of 256 and 128 neurons and the Softmax layer with six neurons in a new neural network (NN) classifier is added above the pooling layer, as illustrated in Figure 16. The batch normalization [63] technique is applied on newly added FC layers.

Table 2. Deep CNN feature extractors

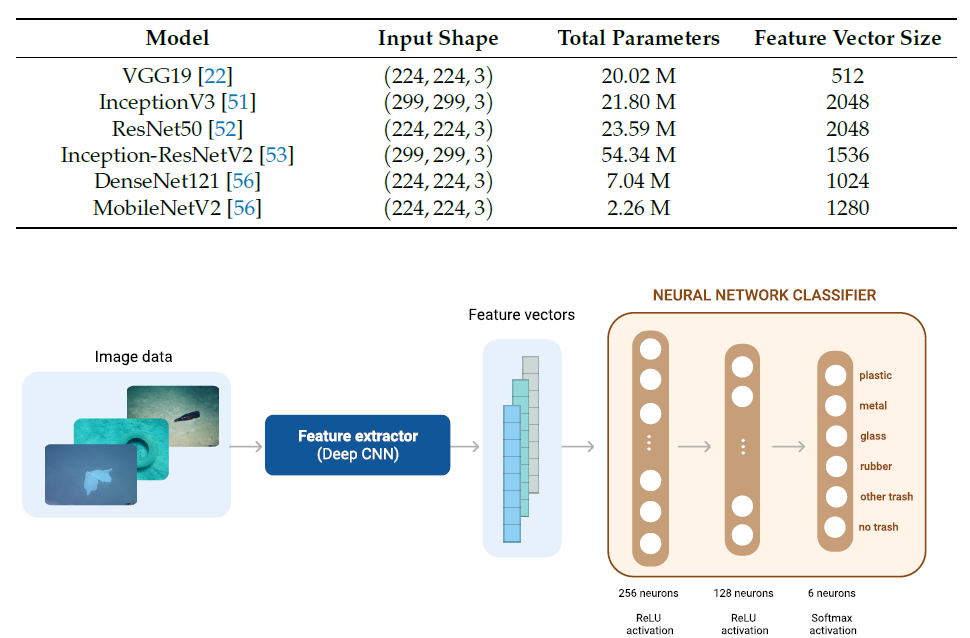
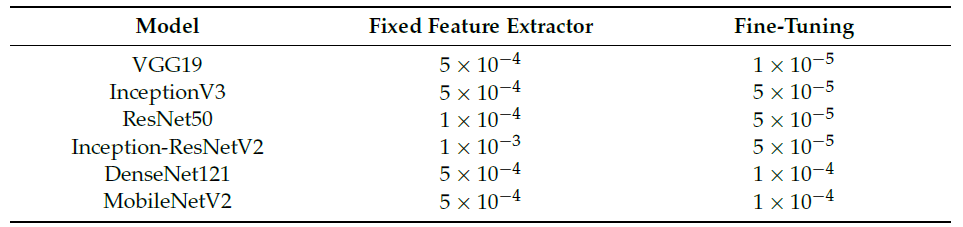


Figure 16. Classification of marine debris images using a neural network classifier, which receives extracted feature vectors as inputs and outputs class-wise probability distribution.

In our experiment, there are 3 different methods we use for deep CNN feature. Firstly, its layers were all freeze and we trained only the newly added NN classifier. Secondly, fine-tune all weights of the deep CNN are used and we trained it together with the NN classifier. For example, we use the loaded Image-Net weights for the beginning training of the model. Thirdly, deep CNN were freeze and train the top NN classifier. Then, we unfreeze the weights of deep CNN in order to fine-tune them. The first way is denoted as fixed feature extractor (FFE). The second way is denoted as fine-tuning (FT). The third way is denoted as FFE+FT. We use Adam [64] optimizer with learning rates as in Table 3, b1 = 0.9, b2 = 0.999 and e = 10􀀀7 to train all models. The learning rate of each model is chosen from the set of predefined values on a logarithmic scale using the 5-fold cross-validation.

Table 3. Learning rates

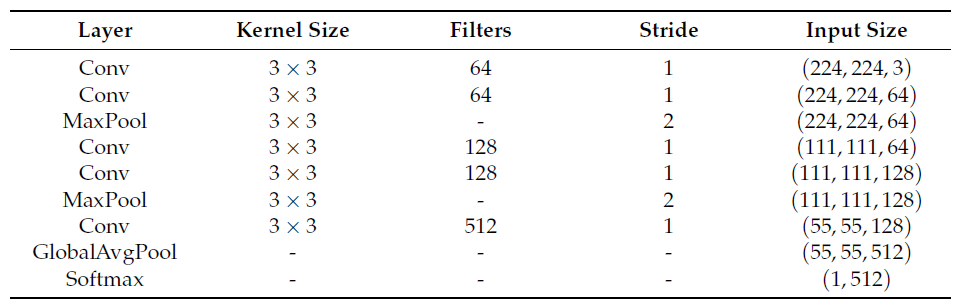
All models were trained for a total of 100 epochs. In the FFE+FT example, we only train the NN classifier on top for the first 25 epochs and keep the deep CNN layers frozen. We fine-tune the weights of the deep CNN and the NN classifier in the remaining 75 epochs. We chose small mini-batches of size 16, which have a smaller memory footprint than bigger ones, for all models. In addition, small batch sizes improve generalization and optimization convergence [65,66]. We utilize data augmentation to artificially enlarge the training set because we only have a restricted amount of data for the training. During the training, we add random rotations, width and height shifts, shearing and horizontal flipping to the images.

For each model architecture, images are processed in an adequate format by utilizing the corresponding preprocess\_input function from the tf.keras.applications (https://www.tensorflow.org/api\_docs/python/tf/keras/applications, accessed on 10 June 2021)module. More precisely, for VGG19 and ResNet50 models, the images are converted from RGB to BGR image format, and each color channel is zero-centered with respect to the unscaled ImageNet data. Pixel values of input images are scaled between 􀀀1 and 1 for the InceptionV3, Inception-ResNetV2, and MobileNetV2 architectures. Finally, for the DenseNet121 architecture, pixel values are scaled between 0 and 1, and then each color channel is normalized with respect to the ImageNet data.

3.3 Simple Neural Network Architecture

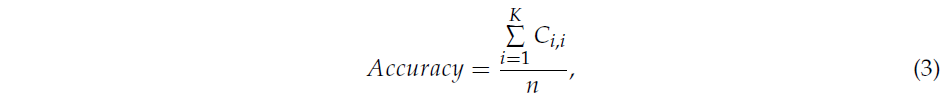
We built a basic neural network with the architecture given in Table 4 to compare the performance of typical pre-trained deep model designs with the performance of smaller neural networks. Batch normalization [63] is used to all convolutional layers, and ReLU nonlinearity is used. This neural network was trained for 150 epochs using marine debris data with 16-epoch mini-batches. With a learning rate of 104, b1 = 0.9, b2 = 0.999, and e = 107, we employed the Adam optimization algorithm. The image pixel values are scaled from 0 to 1. During training, the same augmentation techniques as with the pre-trained model were applied to artificially expand the training dataset.

Table 4. Simple neural network model architecture with approximately 855K trainable parameters.



3.4 Evaluation Metrics

We employ four quantitative metrics frequently used for multiclass classification problems to evaluate the performance of marine debris classifiers: accuracy, precision, recall, and F1-score [67]. Assume that a data collection D comprises K > 2 different classes, each encoded with the numbers 1, 2,..., K. Let Ci, j denote the number of samples classified as class j but belonging to class i. The confusion matrix is a K x K matrix with the formula C = [Ci, j]. The fraction of correctly identified data points is determined as the overall accuracy of a model and it is calculated as



where n denotes the cardinality of D. Since accuracy weighs more highly populated classes and thus strong errors on classes with just a few examples are hard to identify, we complement accuracy scores with precision, recall, and F1 metrics scores. We introduce the following notation:

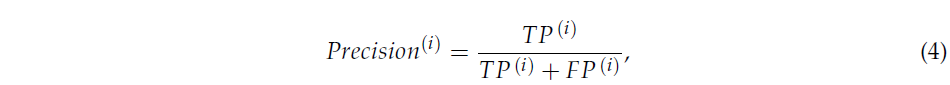
• TP (i) (True Positive): number of correctly classified instances of class i, i.e., Ci,i;

• FP (i) (False Positive): number of instances falsely classified as class i, FP ^ (i) =

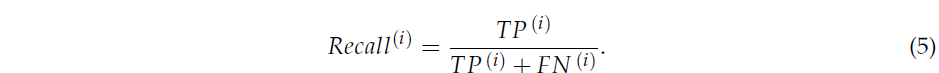


• FN (i) (False Negative): number of instances classified as j 6= i actually belonging to class i, FN (i) =

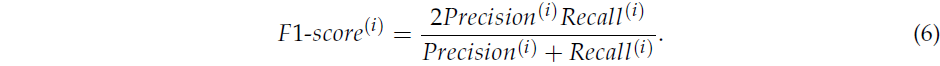
Precision corresponding to the class i is calculated as



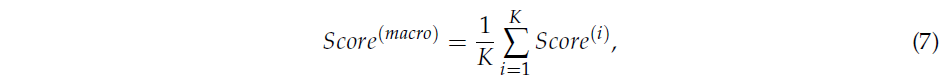
While corresponding recall is given by



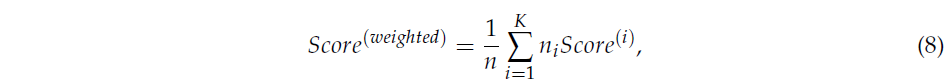
Precision measures the ability of a model to return only relevant instances, while recall expresses the model’s ability to find all relevant instances in a dataset. F1-Score of the i-th class combines corresponding precision and recall scores by calculating their harmonic mean resulting in



Obtained per class metrics are aggregated in overall macro scores computed as simple arithmetic means in the following way



where Score is either Precision, Recall, or F1-score. Sometimes F1-score^(macro) is calculated as  [67,68]. However, we use F1-score^(macro)rather than F1-score ^(macro2), since it is more robust toward the error type distribution [68]and it is also implemented in Python’s sklearn library [61]. Macro scores do not take the class imbalance into account. Additionally, weighted scores are computed to address the uneven data distribution. Weighted scores find the average of class-wise scores weighted by the number of class instances as follows:



where ni denotes the support for class i.

In addition to previously mentioned performance measures, we use Cohen’s Kappa coefficient [69], which expresses the level of agreement between classifier predictions and actual class labels. It is defined as



where po denotes the observational probability of agreement and pc probability of agreement by chance. k < 0 indicates poor agreement, k < 0, 0.2] slight, k < 0.2, 0.4] fair, k <0.4, 0.6] moderate, k <0.6, 0.8] substantial, and k <0.8, 1] almost perfect agreement [70]. The Kappa coefficient shows how much better the given classifier performs than the random classifier that predicts class labels based on the class frequencies.

Not Complete.

**References**

1. Sheavly, S.; Register, K. Marine debris & plastics: Environmental concerns, sources, impacts and

solutions. J. Polym. Environ. 2007, 15, 301–305.

1. Savoca, M.S.; Wohlfeil, M.E.; Ebeler, S.E.; Nevitt, G.A. Marine plastic debris emits a keystone

infochemical for olfactory foraging seabirds. Sci. Adv. 2016, 2, e1600395.

1. Pfaller, J.B.; Goforth, K.M.; Gil, M.A.; Savoca, M.S.; Lohmann, K.J. Odors from marine plastic debris elicit foraging behavior in sea turtles. Curr. Biol. 2020, 30, R213–R214.
2. Lusher, A.; Hollman, P.; Mendoza-Hill, J. Microplastics in Fisheries and Aquaculture: Status of Knowledge on Their Occurrence and Implications for Aquatic Organisms and Food Safety; FAO: Rome, Italy, 2017.
3. Smith, M.; Love, D.C.; Rochman, C.M.; Neff, R.A. Microplastics in seafood and the implications for

human health. Curr. Environ. Health Rep. 2018, 5, 375–386. doi:10.1007/s40572-018-0206-z

1. Meeker, J.D.; Sathyanarayana, S.; Swan, S.H. Phthalates and other additives in plastics: Human exposure and associated health outcomes. Philos. Trans. R. Soc. B: Biol. Sci. 2009, 364, 2097–2113.
2. Newman, S.; Watkins, E.; Farmer, A.; Ten Brink, P.; Schweitzer, J.P. The economics of marine litter. In Marine Anthropogenic Litter; Springer: Chaam, The Netherlands, 2015; pp. 367–394.
3. Williams, A.; Rangel-Buitrago, N. Marine Litter: Solutions for a Major Environmental Problem. J. Coast. Res. 2019, 35, 648–663.
4. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv 2015, arXiv:1506.01497.
5. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection.

arXiv 2020, arXiv:2004.10934.

1. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In

Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019; pp. 6105–6114.

1. Brock, A.; De, S.; Smith, S.L.; Simonyan, K. High-Performance Large-Scale Image Recognition Without Normalization. arXiv 2021, arXiv:2102.06171.
2. Tao, A.; Sapra, K.; Catanzaro, B. Hierarchical multi-scale attention for semantic segmentation. arXiv 2020, arXiv:2005.10821.
3. Zhu, Y.; Sapra, K.; Reda, F.A.; Shih, K.J.; Newsam, S.; Tao, A.; Catanzaro, B. Improving semantic

segmentation via video propagation and label relaxation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 8856–8865.

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444.
2. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989, 1, 541–551.
3. Hubel, D.H.; Wiesel, T.N. Receptive fields and functional architecture of monkey striate cortex. J. Physiol. 1968, 195, 215–243.
4. Fukushima, K. Neocognitron. Scholarpedia 2007, 2, 1717.
5. Nair, V.; Hinton, G.E. Rectified Linear units Improve Restricted Boltzmann Machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
6. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. Pattern Recognit. 2018, 77, 354–377.
7. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 2012, 25, 1097–1105.
8. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv 2014, arXiv:1409.1556
9. Valdenegro-Toro, M. Submerged marine debris detection with autonomous underwater vehicles. In Proceedings of the 2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA), Amritapuri, India, 18–20 December 2016; pp. 1–7.
10. Panwar, H.; Gupta, P.; Siddiqui, M.K.; Morales-Menendez, R.; Bhardwaj, P.; Sharma, S.; Sarker, I.H. AquaVision: Automating the detection of waste in water bodies using deep transfer learning. Case Stud. Chem. Environ. Eng. 2020, 2, 100026.
11. Kylili, K.; Kyriakides, I.; Artusi, A.; Hadjistassou, C. Identifying floating plastic marine debris using a deep learning approach. Environ. Sci. Pollut. Res. 2019, 26, 17091–17099.
12. Kylili, K.; Hadjistassou, C.; Artusi, A. An intelligent way for discerning plastics at the shorelines and the seas. Environ. Sci. Pollut. Res. 2020, 27, 42631–42643.
13. Fulton, M.; Hong, J.; Islam, M.J.; Sattar, J. Robotic detection of marine litter using deep visual detection models. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 5752–5758.
14. 28. Politikos, D.V.; Fakiris, E.; Davvetas, A.; Klampanos, I.A.; Papatheodorou, G. Automatic detection of seafloor marine litter using towed camera images and deep learning. Mar. Pollut. Bull. 2021, 164, 111974.
15. Musi´c, J.; Kruži´c, S.; Stanˇci´c, I.; Alexandrou, F. Detecting Underwater Sea Litter Using Deep Neural Networks: An Initial Study. In Proceedings of the 2020 5th International Conference on Smart and Sustainable Technologies (SpliTech), Split, Croatia, 23–26 September 2020; pp. 1–6.
16. Lorenzo-Navarro, J.; Castrillón-Santana, M.; Santesarti, E.; De Marsico, M.; Martínez, I.; Raymond, E.; Gómez, M.; Herrera, A. SMACC: A System for Microplastics Automatic Counting and Classification. IEEE Access 2020, 8, 25249–25261.
17. Fallati, L.; Polidori, A.; Salvatore, C.; Saponari, L.; Savini, A.; Galli, P. Anthropogenic Marine Debris assessment with Unmanned Aerial Vehicle imagery and deep learning: A case study along the beaches of the Republic of Maldives. Sci. Total Environ. 2019, 693, 133581.
18. Kako, S.; Morita, S.; Taneda, T. Estimation of plastic marine debris volumes on beaches using unmanned aerial vehicles and image processing based on deep learning. Mar. Pollut. Bull. 2020, 155, 111127.
19. 33. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. IEEE Trans. Knowl. Data Eng. 2010, 22, 1345–1359.
20. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. CVPR09, 2009. Available online: http://www.image-net.org/ (accessed on 2 October 2021).
21. Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 806–813.
22. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In Proceedings of the International Conference on Machine Learning, PMLR, Bejing, China, 22–24 June 2014; pp. 647–655.
23. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? arXiv 2014, arXiv:1411.1792.
24. Azizpour, H.; Sharif Razavian, A.; Sullivan, J.; Maki, A.; Carlsson, S. From generic to specific deep representations for visual recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Boston, MA, USA, 7–12 June 2015; pp. 36–45.
25. Ben Jabra, M.; Koubaa, A.; Benjdira, B.; Ammar, A.; Hamam, H. COVID-19 Diagnosis in Chest X-rays Using Deep Learning and Majority Voting. Appl. Sci. 2021, 11, 2884.
26. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. Nature 2017, 542, 115–118.
27. Wang, X.; Peng, Y.; Lu, L.; Lu, Z.; Bagheri, M.; Summers, R.M. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2097–2106.
28. Karri, S.P.K.; Chakraborty, D.; Chatterjee, J. Transfer learning-based classification of optical coherence tomography images with diabetic macular edema and dry age-related macular degeneration. Biomed. Opt. Express 2017, 8, 579–592.
29. Chen, J.; Chen, J.; Zhang, D.; Sun, Y.; Nanehkaran, Y. Using deep transfer learning for image-based plant disease identification. Comput. Electron. Agric. 2020, 173, 105393.
30. Qi, H.; Liang, Y.; Ding, Q.; Zou, J. Automatic Identification of Peanut-Leaf Diseases Based on Stack Ensemble. Appl. Sci. 2021, 11, 1950.
31. Jeon, H.K.; Kim, S.; Edwin, J.; Yang, C.S. Sea Fog Identification from GOCI Images Using CNN Transfer Learning Models. Electronics 2020, 9, 311.
32. Mahdianpari, M.; Salehi, B.; Rezaee, M.; Mohammadimanesh, F.; Zhang, Y. Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. Remote Sens. 2018, 10, 1119.
33. Iqbal Hussain, M.A.; Khan, B.; Wang, Z.; Ding, S. Woven Fabric Pattern Recognition and Classification Based on Deep Convolutional Neural Networks. Electronics 2020, 9, 1048.
34. Japan Agency for Marine Earth Science and Technology, Deep-sea Debris Database. Available online: http://www.godac.jamstec. go.jp/catalog/dsdebris/metadataList?lang=en (accessed on 2 October 2021).
35. O’Mahony, N.; Campbell, S.; Carvalho, A.; Harapanahalli, S.; Hernandez, G.V.; Krpalkova, L.; Riordan, D.; Walsh, J. Deep Learning vs. Traditional Computer Vision. In Advances in Computer Vision; Arai, K., Kapoor, S., Eds.; Springer International Publishing: Chaam, The Netherlands, 2020; pp. 128–144.
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
37. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
39. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
40. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
41. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv 2017, arXiv:1704.04861.
42. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
43. Breiman, L. Random forests. Mach. Learn. 2001, 45, 5–32.
44. Vapnik, V.N. The Nature of Statistical Learning Theory; Springer: New York, NY, USA, 1995.
45. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://www.tensorflow.org/ (accessed on October 2021).
46. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 2 October 2021).
47. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2011, 12, 2825–2830.
48. Lin, M.; Chen, Q.; Yan, S. Network in network. arXiv 2013, arXiv:1312.4400.
49. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 7–9 July 2015; Volume 37, pp. 448–456.
50. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. arXiv 2014, arXiv:1412.6980.
51. Wilson, D.R.; Martinez, T.R. The general inefficiency of batch training for gradient descent learning. Neural Netw. 2003, 16, 1429–1451.
52. Masters, D.; Luschi, C. Revisiting small batch training for deep neural networks. arXiv 2018, arXiv:1804.07612.
53. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. 2009, 45, 427–437.
54. Opitz, J.; Burst, S. Macro f1 and macro f1. arXiv 2019, arXiv:1911.03347.
55. Cohen, J. A coefficient of agreement for nominal scales. Educ. Psychol. Meas. 1960, 20, 37–46.
56. Landis, J.R.; Koch, G.G. The measurement of observer agreement for categorical data. Biometrics 1977, 33, 159–174.
57. Anscombe, F.J. Graphs in Statistical Analysis. Am. Stat. 1973, 27, 17–21.
58. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. J. Mach. Learn. Res. 2008, 9, 2580–2605.
59. Zhu, L.; Spachos, P. Towards Image Classification with Machine Learning Methodologies for Smartphones. Mach. Learn. Knowl. Extr. 2019, 1, 59.
60. Sykora, P.; Kamencay, P.; Hudec, R.; Benco, M.; Sinko, M. Comparison of Feature Extraction Methods and Deep Learning Framework for Depth Map Recognition. In Proceedings of the 2018 New Trends in Signal Processing (NTSP), Liptovsky Mikulas, Slovakia, 10–12 October 2018; pp. 1–7.
61. Postorino, M.N.; Versaci, M. A geometric fuzzy-based approach for airport clustering. Adv. Fuzzy Syst. 2014, 2014.
62. Mahmoudi, M.R.; Baleanu, D.; Qasem, S.N.; Mosavi, A.; Band, S.S. Fuzzy clustering to classify several time series models with fractional Brownian motion errors. Alex. Eng. J. 2021, 60, 1137–1145.
63. Xu, K.; Pedrycz, W.; Li, Z.; Nie, W. Optimizing the prototypes with a novel data weighting algorithm for enhancing the classification performance of fuzzy clustering. Fuzzy Sets Syst. 2021, 413, 29–41.