A weakly supervised approach for recycling code recognition

A Weakly Supervised Approach for Recycling Code Recognition

Lorenzo Pellegrini[a,*], Davide Maltoni[a], Gabriele Graffieti[a], Vincenzo Lomonaco[b], Lisa Mazzini[c], Marco Mondardini[c] and Milena Zappoli[c]

[a]Department of Computer Science - Science and Engineering (DISI), University of Bologna, Via dell'Università, 50, Cesena (FC), 47521, Italy
[b]Department of Computer Science, University of Pisa, Largo B. Pontecorvo, 3, Pisa (PI), 56127, Italy
[c]Data Analytics and Intelligent Automation, Hera Spa, Viale Carlo Berti Pichat, 2/A, Bologna (BO), 40127, Italy

ARTICLE INFO

Keywords:
Recycling code recognition
Recycling symbols recognition
Waste recognition
Weakly supervised classification

ABSTRACT

Waste sorting at the household level is a virtuous process that can greatly increase material recycling and boost the circular economy. To this purpose, waste must be differentiated by material (e.g., PVC, Polyethylene, Paper, Glass, Aluminum, etc.), a task that can be simplified by printing a recycling code on the product case. Unfortunately, the large number of recycling codes printed on products makes this process unfriendly for many users. In this work, we propose a vision-based mobile application to support users in recognizing recycling codes for proper waste sorting. The proposed system combines a dual-head CNN with an image processing pipeline (based on domain knowledge) in order to improve: i) the reliability of symbol detection/classification and ii) the weakly-supervised labeling of new samples during iterative training. Our experimental results prove the feasibility of developing effective applications with minimum effort in terms of data collection and labeling, which is one of the main obstacles to successfully applying deep-learning techniques to real-world problems.

1. Introduction

Recycling codes are used to identify the materials from which products are made, to facilitate their recycling. Although the symbols used to depict recycling codes are not many and their shape can be easily identified (e.g., chasing arrows in Figure 1), the total number of codes is high[1] and some of them are rarely used. Hence, users cannot easily remember the codes and the corresponding recycling rules. The aim of this work is the development of a mobile application that can support the users in recognizing recycling codes for proper waste sorting.

In the proposed approach, as a user points his smartphone camera to the zone where the code is printed, the application must promptly recognize the symbol and display recycling instructions. The development of such an application is not trivial because: (i) the symbols and the associated labels (number/text) are usually small in size and properly framing them with a smartphone camera is critical because of the close distance necessary for the required zoom and the risk of capturing out-of-focus or blurred images; (ii) taking a single shot and sending it to a remote server for recognition requires a stable internet connection and would often lead to rejecting the chosen sample thus requiring multiple attempts. Performing online recognition from the camera video stream is definitely better, but it requires on-device processing at a proper frame rate.

As of today, the most obvious approach seems to be selecting a state-of-the-art Convolutional Neural Network (CNN) which is efficient enough for real-time inference on a smartphone and training it as a classifier on a sufficiently wide labeled dataset. This approach has some drawbacks:

- recycling codes are numerous and a dataset containing a sufficient number of examples per class is not available. Collecting from scratch such a dataset is time-consuming and finding examples of some "rare" classes can be an issue not easy to address with data augmentation techniques;

*Corresponding author
l.pellegrini@unibo.it (L. Pellegrini); davide.maltoni@unibo.it (D. Maltoni); gabriele.graffieti@unibo.it (G. Graffieti); vincenzo.lomonaco@unipi.it (V. Lomonaco); lisa.mazzini@gruppohera.it (L. Mazzini); marco.mondardini@gruppohera.it (M. Mondardini); milena.zappoli@gruppohera.it (M. Zappoli)
ORCID(s): 0000-0002-7680-8116 (L. Pellegrini); 0000-0002-6329-6756 (D. Maltoni); 0000-0002-1649-6018 (G. Graffieti); 0000-0001-8308-6599 (V. Lomonaco)

[1]See https://en.wikipedia.org/wiki/Recycling_codes

Figure 1: Some examples of recycling symbols in use in the European Union (European Commission, 1997). Minor graphical differences exist in the symbols introduced by other regulatory bodies. For instance, in the United States of America (ASTM International, 2021), the chasing arrows symbol is represented as a solid triangle.

- while on-screen feedback can guide the user in properly framing the symbols (see Figure 3), it is not realistic to impose that the searched pattern is well centered and size normalized. Therefore, we must move from classification to a detection task with the consequence that: i) the model complexity increases; ii) the symbol bounding boxes need to be labeled in the training dataset.

To solve the aforementioned critical aspects, we propose a two phases approach that combines a deep learning architecture with an image processing pipeline. Training the proposed system only requires a small labeled dataset since weak supervision is leveraged to improve the accuracy iteratively.

1.1. Motivation and contribution

The motivation of this work is to prove that a smartphone application can be developed to support users in recycling code recognition. An important design constraint is to keep data collection/labeling efforts as low as possible.

The main contributions are:

- the combination of deep models with image processing pipelines (based on domain knowledge) to overcome the limitations of data-driven approaches when training data is limited and does not cover all the cases;
- the definition and validation of a weakly supervised iterative training scheme, where the new data labeling is validated through the proposed image processing pipeline. Such an approach, which was scarcely investigated in the literature, can be very useful to solve real-world problems in presence of limited labeled data. While this is demonstrated in the context of recycling code recognition, the approach is general enough to be applied to other real-world scenarios;
- the demonstration that recycling code recognition can be performed on-device with good accuracy. Even if some implementations of recycling code recognition already exist (e.g., on the Kaggle code page associated with the Plastic Recycling Codes (SevenPlastics) dataset; Ya (2019)), we are not aware of accompanying scientific papers or technical descriptions and therefore we cannot make a direct comparison with the proposed approach. However, in our experiments, we evaluate different architectures and provide an ablation study where the most important contributions of our proposal are isolated and benchmarked against baseline methods;
- we release the novel dataset used in our experiments. The dataset is composed of short video sessions gathered by users depicting recycling codes in various conditions[2]: different smartphone devices, different illumination

[2]Dataset available here: *link to be disclosed.*

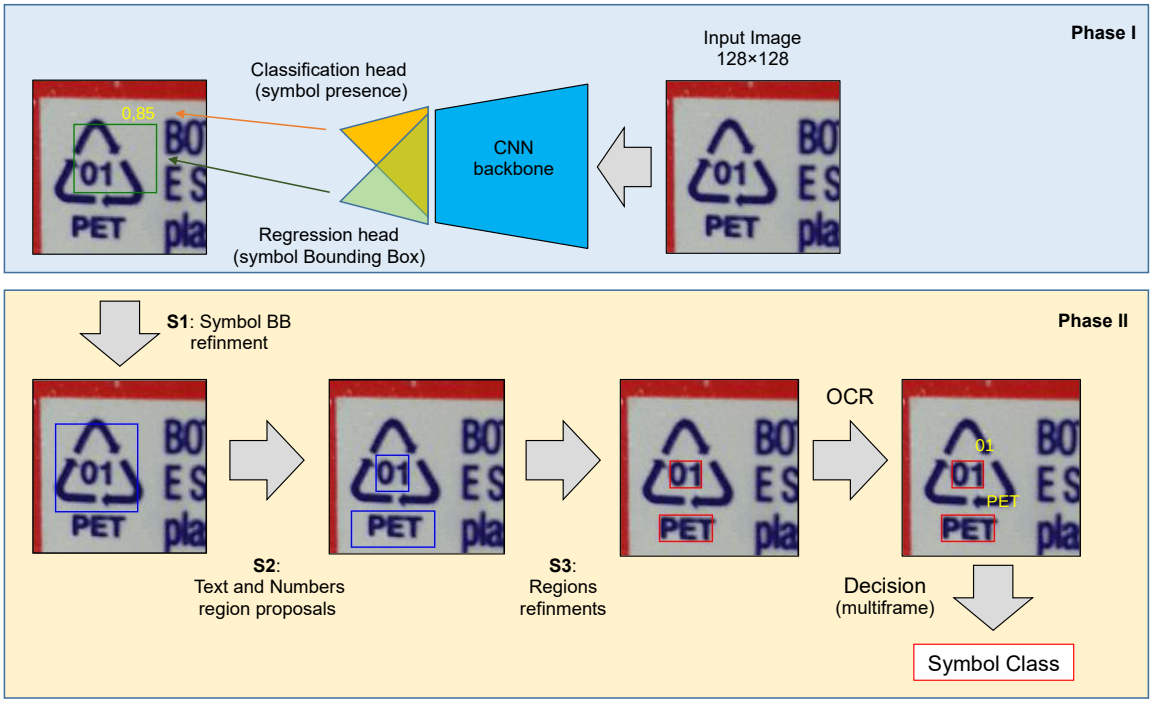

Figure 2: Overall schema of the proposed two-phase approach. In the first phase, the system uses a CNN to detect the presence of a recycling symbol (classification head) and its location in the image (regression head). The symbol location is then refined according to S1 (Section 4.1). Given the refined symbol location, two coarse text region proposals are obtained using fixed ratios and relative positioning (S2; Section 4.2) and then refined using the regions refinement procedure S3 (Section 4.3). Finally, the refined text regions are passed to the OCR module to obtain the numerical and/or textual code. A multi-frame decision mechanism is employed to fuse the results of multiple frames (Section 4.4).

conditions (both indoor and outdoor), and uncontrolled pose/distance. The release of a new dataset is very important to study recycling symbol recognition, since the only available dataset (i.e., the aforementioned SevenPlastics) only contains plastic products.

The rest of this paper is organized as follows: in Section 2, we introduce the basis of our approach and in Section 3 we discuss the related literature. Section 3 presents the deep architecture used and its initial training, and Section 4 details the image processing pipeline designed to refine the deep model output to properly classify the symbol code. The weakly supervised iterative training is introduced in Section 5, and experimental results are provided and commented on in Section 6. Finally, in Section 7 we draw some conclusions.

2. The approach

A graphical scheme of the proposed two-phase symbol recognition approach is depicted in Figure 2. An efficient classification CNN (MobileNet v2; Sandler et al. (2018)) is extended with a regression head for the prediction of the symbol position (bounding box). This model is much simpler and more efficient than a YOLO (Redmon et al., 2016) or SSD (Liu et al., 2016) since we assume that a single object of interest is present in the image. The model is trained to localize a recycling symbol (chasing arrows) in a single image: the model output is the confidence in the symbol presence (classification) and the bounding box coordinates (regression). It is worth noting that recycling codes of the different classes (associated with different numbers and texts) are not discerned at this level, and therefore the classifier makes a binary decision (symbol found vs no-symbol found). Extending the application to work with multiple recycling symbols (e.g., circle, trashcan, etc.) is not complex and some hints are provided in the following sections. The frames where a symbol is found are passed to a second processing phase where an OCR algorithm is used to extract the numbers and text identifying the recycling category.

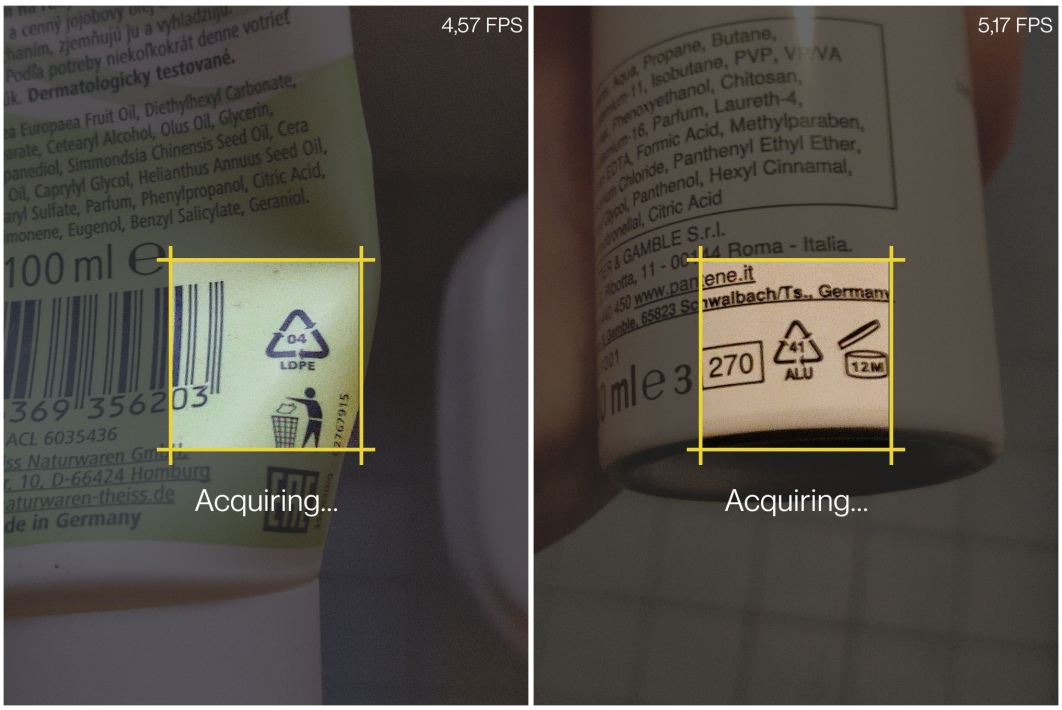

Figure 3: The acquisition interface of the proposed application. The user is asked to frame the symbol inside the central area (with a yellow border). The external greyed-out area is ignored during recognition, but is useful to help the user center the symbol.

Although the above two phases decomposition allows training of the system without a comprehensive dataset (in terms of real-world combinations of symbols, numbers, and text), some problems persist.

In fact, the OCR algorithms tested (even in combination with state-of-the-art text detectors such as CRAFT; Baek et al. (2019)), are often disturbed by the presence of graphics (e.g., lines, drawings) in the proximity of the text and their accuracy significantly degrades when the text is not well isolated from the background (see Figure 4). To solve this problem, instead of retraining an OCR to work in the specific context of our application[3], we introduced an image post-processing pipeline that, starting from the bounding box returned by the model, identifies and refines the text regions to be processed by a general-purpose OCR.

Furthermore, training a regression head requires a sufficient number of bounding box labeled examples in the training dataset, whose markup is a boring and time-consuming task. To overcome this problem we propose a weakly supervised (iterative) training whose flow is illustrated in Figure 5, where:

- the model is initially trained on a small recycling code dataset: the Plastic Recycling Codes (SevenPlastics) dataset available on Kaggle (Ya, 2019);
- the initial model has been used to collect new examples that are labeled (both the class and the bounding box) with self-supervision and weak user engagement. In particular, the image processing pipeline allows for validating the bounding box labels provided by self-supervision thus minimizing the risk of drifts;
- the model is then (iteratively) trained on the union of the initial data (full supervision) and the successive examples (weak supervision), achieving a relevant accuracy improvement.

[3] Again, this would require a representative training dataset, which is difficult to collect.

Figure 4: Some of the results obtained by applying the Tesseract Optical Character Recognition (OCR) (Smith (2007)) to the text regions returned by the CRAFT text detector (Baek et al. (2019)). In the first row, the predicted text positions are highlighted in red and the OCR results are in blue. The second row shows the heatmap used by the text detector. The number and text in the first two columns are properly extracted, while the OCR failed in the last two columns because of poor text region detection.

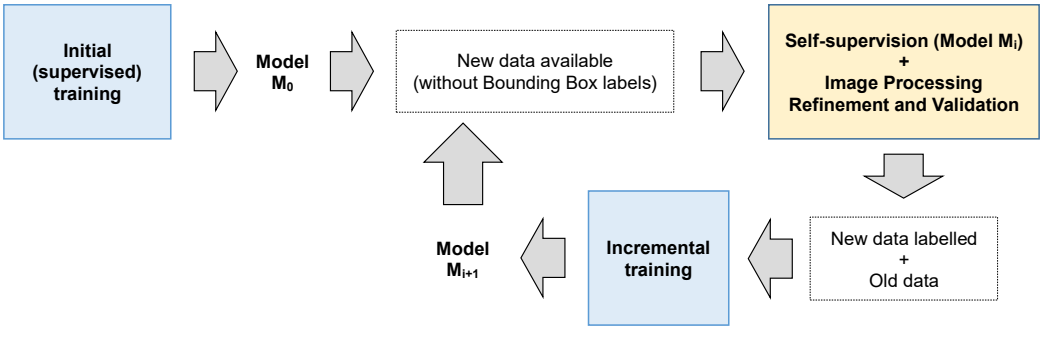

Figure 5: The proposed weakly supervised iterative training pipeline. The CNN model M_0 is firstly obtained by training on the Kaggle Plastic Recycling Codes (SevenPlastics) dataset. New videos are then collected by the users, who label each video with the correct symbol. Users do not need to annotate images with bounding boxes since the pipeline introduced in Section 5 is used to provide bounding boxes via self-supervision. The union of the previous training dataset and the newly self-labeled data is then used to incrementally train the model to obtain M_{i+1}.

2.1. Related literature

Waste sorting

Ya (2019) introduced a dataset of Plastic Recycling Codes (SevenPlastics) with an associated recognition approach . Unfortunately, no accompanying scientific papers or technical descriptions were released. We are using this dataset for the initial training of our model.

While we are not aware of other specific studies on recycling code recognition, some researchers proposed to infer the material directly from the product visual aspect (Yang and Thung, 2016; Meeradevi, 2020; Bobulski and Kubanek, 2021). This is a quite challenging task due to the huge amount of existing products and packages, their aspect variations, and the continuous launch of new items. For this reason, most of the approaches proposed in the literature operate in restricted settings (e.g., a limited number of materials, uniform background, etc). Furthermore, in some cases, the visual discrimination of materials is not possible, and even humans need to rely on the associated recycling symbols.

Single-frame and Video detection

Object detection is a deeply studied problem and several solutions exist in the literature. Object detection on single frames can be approached with high-accuracy multi-stage models such as Faster R-CNN (Ren et al., 2015) or with more efficient single-stage models such as YOLO (Redmon et al., 2016) or SSD (Liu et al., 2016). Object detection from videos is an attractive alternative because, by exploiting temporal smoothness, the bounding box accuracy can be improved across the frames. The model used in the first stage of our approach works at the single-frame level. We initially excluded video detection techniques because most of them are computationally demanding and our aim is not to detect the symbol across all frames but to locate it accurately on one or few frames. However, efficient implementations that can also be deployed on mobile CPUs (e.g., Zhu and Liu (2018); Liu et al. (2019); Sabater et al. (2020)) have been proposed in the recent literature. Therefore, in a future extension of this work, it could be interesting to evaluate these techniques in conjunction with the proposed image processing pipeline.

Weakly Supervised Object Detection

A comprehensive survey on weakly supervised object detection was proposed in Zhang et al. (2021). In this scenario, the samples of the training dataset are labeled with the object class, but bounding boxes are not available. Several methods use an initial annotation stage for the detector start-up and then rely on some form of self-supervision to iteratively annotate the bounding boxes and improve the detector.

Differently from the above cited techniques, our approach exploits domain knowledge (through an image processing pipeline) to mitigate the risk of drifts which are difficult to avoid in self-training approaches. In our ablation study, we show that using a basic self-training approach significantly decreases the symbol recognition accuracy.

3. Phase I: frame classification and bounding box regression

The model used in Phase I is a MobileNet v2 (128x128 input, depth multiplier = 1.0) pre-trained on ImageNet (Deng et al., 2009), where:

- the output classification layer is replaced with a new layer with just 2 classes (symbol vs no-symbols);
- a new fully-connected layer with 4 neurons is attached to the second last layer to be trained as a regressor to predict the symbol position (bounding box). The coordinates of the bounding box are normalized in $[0, 1]$.

The choice of the MobileNet v2 as the CNN backbone is mainly due to its inference efficiency. However, in Section 6.1 we show that using more powerful (and computationally demanding) architectures for initial symbol detection would have no practical advantages.

The loss function used is a weighted sum of cross-entropy (for the classification output) and mean-square error (for the regression output). The weights of the two components are 1 for the classification loss and 0.2 for the regression loss.

This model is trained on a subset of the Kaggle recycling code dataset consisting of 454 frames whose bounding boxes have been manually marked by us. An equal number of negative examples (i.e., images belonging to the "no-symbol" class) are added by randomly cropping ImageNet (Deng et al., 2009) samples[4]. The training is carried out for 100 epochs with the Adam optimizer. This training session is a tuning for all the levels except the two output heads whose weights are randomly initialized. As shown in Section 6 even a small dataset of recycling codes allows the model to reach a good binary classification accuracy. However, the symbol locations returned by the regressor are often imprecise and do not allow for accurately cropping the text regions.

4. Phase II: box refinements and OCR

Figure 2 highlights the sequence of processing steps performed for the classification of a single frame. The CNN model of Phase I outputs a confidence score on the presence of a recycling symbol (0.85 in the example) and a bounding box denoting his position in the frame. If the confidence is lower than a threshold the frame is immediately discarded, otherwise the bounding box position is refined (step S1). The regions where we can expect to find numbers and text can be then hypothesized (step S2) based on domain knowledge (i.e., the geometric shape of the symbols and the average position/size of its textural attributes). In the case of the chasing arrows symbol, the internal region should contain a

[4]Since Kaggle images are cropped around the symbol we cannot use out-of-center portions to create negative examples.

number and the base region a text. However, this is not always the case and more than two region proposals can be passed-on if necessary. Again, since the region proposals are not accurate enough for reliable OCR, a refinement step (S3) is necessary for segmentation. Each refined region is then processed through Tesseract OCR and the results are passed to a multi-frame decision-maker for final classification.

Hereafter we provide further details on steps S1, S2, S3, and the final OCR and multi-frame decision. Intermediate results are graphically shown in Figure 6. Further examples are reported in Figure 7. Since in this paper we focus on a chasing arrow symbol, the image processing steps are tailored to this symbol. However, we believe that simple variants of proposed pipelines can be designed for other common recycling symbols. The parameters of the processing steps have been tuned by extracting geometric statistics from the Kaggle labeled dataset.

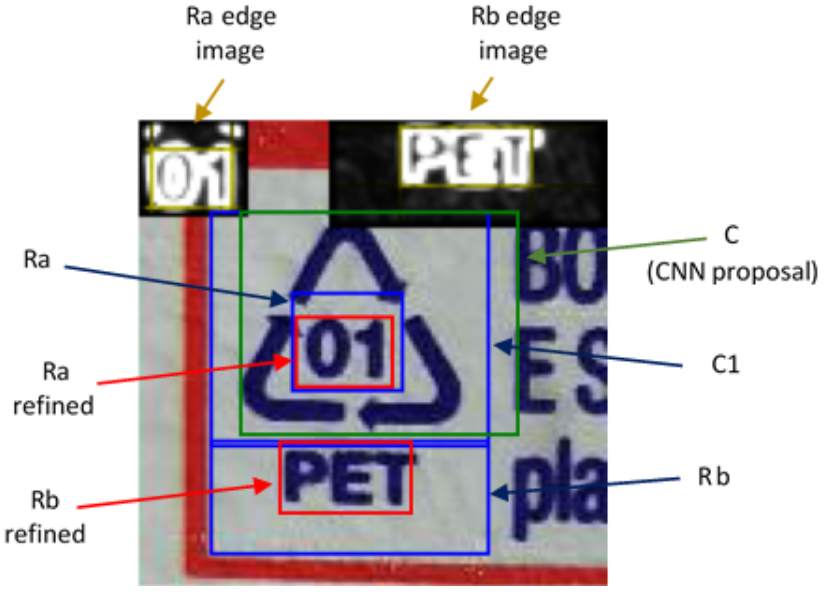

Figure 6: Graphical visualization of intermediate steps on an example image.

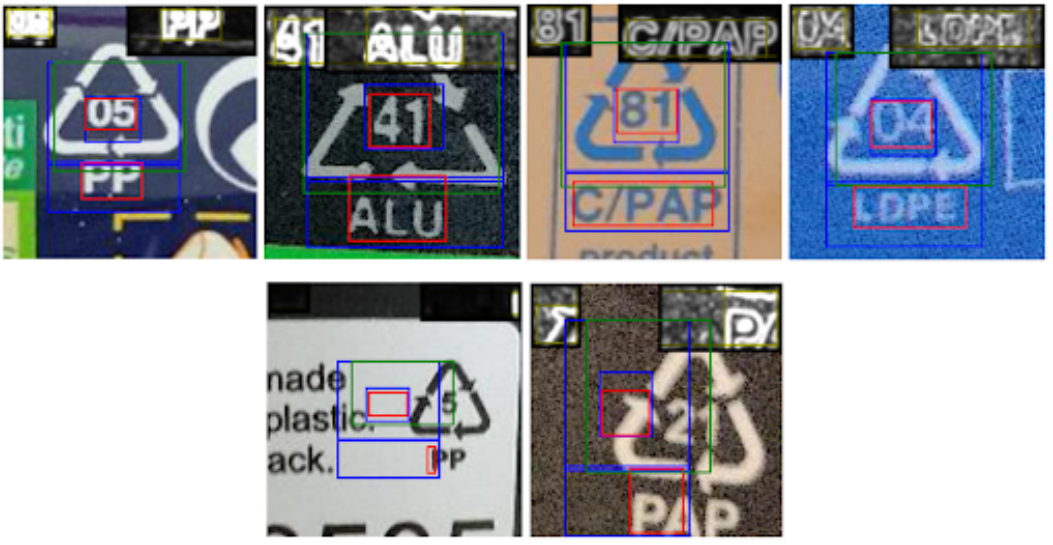

Figure 7: The first row shows some examples where the proposed processing is successful. In the second row, some errors are shown; errors are typically due to a bad initial estimation of C by the CNN.

4.1. S1: Symbol bounding box refinement

The rectangle returned by the CNN is used to crop a sub-image C = $[x_{left}, y_{top}, x_{right}, y_{bottom}]$, which is first converted to grayscale and then processed as follows.

1. Compute the vertical axis of symmetry x_{sym} of C by minimizing the mean pixel intensity abs-difference between two small regions at the right and left of each hypothesized vertical axis. Since the considered symbol is nearly symmetric (with respect to y) this simple step allows a better horizontal centering.
2. Select a bottom region B as
   $[x_{sym} - w/2, y_{bottom} - o, x_{sym} + w/2, y_{bottom} + o]$
   where w is the initial crop width, ($w = x_{right} - x_{left}$), and o is a 20 pixel offset.
3. Compute the magnitude of horizontal edges (abs of Sobel vertical filter 3x3) in B and apply a Gaussian blur (3x3) to reduce noise.
4. Integrate the edge magnitude along the rows in B and select y_{new_bottom} as the row with the highest magnitude. The rationale behind this step is that the chaining arrows symbol has a triangular shape and the triangle base produces strong horizontal edges.
5. The region
   $C1 = [x_{sym} - w/2, y_{top}, x_{sym} + w/2, y_{new\_bottom} + o2]$
   is returned, where $o2$ is a small constant offset (6 pixels).

4.2. S2: Region proposals

Two region proposals Ra and Rb can be defined at fixed positions relative to C1. Ra is coarsely centered in the triangle center and its (width, height) are proportional to the triangle sides. Rb is positioned below the triangle base and is as large as the triangle. Due to the empirical nature of the above rules, Ra and Rb are enlarged (with an extra offset) to avoid losing portions of the searched regions. Even if Ra should contain a number (material code) and Rb a text (material acronym), this is not always the case and a more flexible decision is necessary after OCR. Furthermore, for increased robustness more than two proposals can be generated in this step, since the multi-frame decision adopted is general enough to tolerate this.

4.3. S3: Regions refinement

Ra and Rb sub-images are firstly converted to grayscale. Then, each of them is shrunk by moving its 4 sides toward the center to precisely frame the text.

1. Compute the magnitude of the gradient (Sobel filters 3x3) in B and apply a Gaussian blur (3x3) to reduce noise. Because of the enlargement in the previous step we expect an external offset region where the gradient magnitude is small.
2. Integrate the magnitude over rows. Iteratively move the region's top(bottom) side toward the center until the integrated magnitude over the current row is higher than a threshold.
3. Integrate the magnitude over columns. Iteratively move the region's left(right) side toward the center until the integrated magnitude over the current column is higher than a threshold.

4.4. OCR and multi-frame decision

The refined Ra and Rb are passed to Tesseract OCR (Smith, 2007), version 4.1.1[5]. Their grayscale intensities can be optionally inverted: in fact, we noted that Tesseract works better with dark text on bright backgrounds. To decide whether to apply the inversion, the average value of pixel intensity in an external boundary region (likely belonging to the background) is compared with the average intensity of the remaining pixels.

Tesseract can be configured to work with a restricted list of allowed characters. The OCR process is executed the first time by allowing only digits [0...9]. If a valid numerical symbol code is not returned, then a second attempt is made by passing a set of alphabetical characters obtained as the union of the characters used in all the symbol acronyms. Considering that number and text may appear in all proposed crops, the OCR engine is run a maximum of 4 times (2 on the refined Ra and 2 on the refined Rb) for each frame.

Even if the designed two-phase approach greatly improves single frame classification (see Table 2) some frames remain critical mainly due to out-of-focus, symbols too close/far from the camera, excessive rotation or skew, and

[5]Tesseract 4.1.1: https://github.com/tesseract-ocr/tesseract/releases/tag/4.1.1

flares. Therefore a multi-frame fusion, based on the sum rule and a fixed threshold, is adopted to improve the overall accuracy. The pseudocode is reported in Algorithm 1. It is worth noting that a symbol receives a score of 0.5 if either the number or the text matches (lines 9 and 10), and a full score (1.0) is assigned only in case of a simultaneous match. The threshold used in our experiments (line 11) is 1.5: this means that 2 fully matching frames need to be associated with a given symbol before recognition or, alternatively, a higher number of partially matching frames is necessary.

The scoring mechanism could be further improved by assigning lower scores (but not zero) when a partial match is detected: for instance, a score of 0.25 could be assigned to "HDPE" if the OCR returns "HOPE" which is not in the dictionary. A normalized Levenshtein distance (Levenshtein et al., 1966) can be used to grade scores. A further enhancement could be achieved by adopting an OCR working with a dictionary at word levels instead of at character levels.

Algorithm 1: Pseudo-code of the proposed approach. D is a many-to-many dictionary that considers equivalences such as different numerical symbols tied to a common text label (and vice versa). For instance, "PAP" is linked to numerical codes {20, 21, 22} while numerical code 02 is linked to text codes {"PEHD", "HDPE"}. The score fusion mechanism keeps these equivalences in consideration (omitted for simplicity in the pseudo-code).

Input: A sequence of frames F_i in the current recognition session; D: A dictionary of recognizable symbols; *model*: the CNN model with classification and regression heads; S: the scores of elements in D. Initialized to 0.

1 **for** *for each frame X in* F_i **do**
2 has_symbol, C = model(X)
3 **if** *not has_symbol* **then** skip frame
4 C1 = S1(X, C)
5 Ra, Rb = S2(C1)
6 $Ra_{refined}$, $Rb_{refined}$ = S3(Ra, Rb)
7 $sym_{ra} = \text{OCR}(Ra_{refined})$
8 $sym_{rb} = \text{OCR}(Rb_{refined})$
9 **if** sym_{ra} *in* D **then** $S[sym_{ra}]$ += 0.5
10 **if** sym_{rb} *in* D **then** then $S[sym_{rb}]$ += 0.5
11 **if** *(exists a* sym_f *such that* $S[sym_f] > threshold$*)* ***and*** *(*$S[sym_f] > S[sym_i]$ *for each* sym_i *!=* sym_f*)* **then**
12 **return** sym_f
13 **end**
14 **end**
15 **return** "unknown"

5. Iterative training

As explained in Section 3, the CNN (classifier + regressor) is first trained on a subsample of the Kaggle Plastic Recycling Codes (SevenPlastic) dataset (Ya (2019), hereafter DB_0), obtaining the model M_0 which is embedded in the first version of the application. A small group of users was involved in a testing stage. When they note that the pointed symbol code is correctly classified no further action is required; otherwise, they provide the correct symbol code through a simplified graphical interface[6]; in the latter case the video frames of the current session are stored in the device and later transferred to a remote workstation. At the end of the testing stage, we collected a total of 173 video sessions corresponding to error cases. While the ground truth code labels are available, the bounding boxes are missing and our aim is to avoid their tedious and expensive manual labeling. Therefore, for each frame of each session, we used model M_0 to estimate the symbol position C and applied steps S1, S2, S3, and the OCR (multistep fusion is not necessary here). If after OCR both the number and text are coherent with the ground truth symbol code, we assume that the refined box C1 (produced by S1) can be reliably used as a ground truth bounding box: therefore, we add this frame to a new dataset P_1. A new training DB_1 dataset can be then assembled as the union of DB_0, P_1, and N_1, where

[6]The application allows to select the symbol from a list and associate it to all the frames in the current video.

Table 1
Composition of the used datasets.

Dataset	Video sessions	Positive examples	Positive examples origin	Manually annotated triangle boxes	Self-supervised triangle boxes	Negative examples	Negative examples origin
DB0	0[7]	454	SevenPlastics	454	0	454	ImageNet-2012
DB1	36	454 + 1699	SevenPlastic + Our videos	454	896	2153	ImageNet-2012
DB2	36[8]	454 + 1699	SevenPlastics + Our videos	454	1001	2153	ImageNet-2012
T0	137	6719	Our videos	4672	-	6719	Ours + ImageNet-2012[9]

N_1 is an extra set of negative examples (of the same size as P_1) randomly sampled from ImageNet. Finally, a new model M_1 can be obtained by retraining the CNN on DB_1. The whole procedure can be iterated more times to improve the application accuracy. As the application accuracy increases with the iterations, the amount of user supervision is progressively reduced.

6. Experimental results

As explained in Section 5, the proposed approach has been trained incrementally. The initial training dataset DB_0 is composed of 454 frames extracted from the Kaggle SevenPlastic dataset. Bounding boxes have been manually provided for DB_0 and T_0: for each instance, we marked three bounding boxes corresponding to the ground truth position of C, Ra, and Rb. Both DB_0 and T_0 have been extended with an equal number of negative examples by randomly cropping ImageNet data.

The test set T_0 used across all the evaluations has been initially collected by a group of 9 users (each with their smartphone). T_0 consists of 137 videos including 6719 frames and 14 different recycling symbols. The symbol class and the bounding boxes (C, Ra, and Rb) have been manually marked.

At training iteration 1, DB_1 is the union of DB_0 with P_1 and N_1 where: (i) P_1 contains 1699 frames extracted from 36 videos collected by beta users; as explained in Section 5 only the frames properly classified are selected and their symbol bounding boxes have been self-generated; (ii) N_1 is a set of negative examples of the same size of P_1.

Finally, for training iteration 2, DB_2 is obtained without further user involvement, but using model M_1 to reclassify P_1 samples. This resulted in an increased number of frames selected (from 896 to 1001) and more accurate self-generation of the ground truth bounding boxes. Table 1 summarizes the composition of the training and test datasets.

Table 2 reports the results obtained after training the system on DB_0, DB_1, and DB_2. Accuracy is reported for single frames or videos (session); as expected, the multi-frame fusion boosts the accuracy. In general, the proposed weakly supervised iterative training proved to be very effective: at the frame level, the classification accuracy increased from 54.76% to 65.70% to 67.54% and, at the video level, from 81.20% to 86.31% to 88.13%.

The symbol binary classification of the CNN (i.e., the accuracy of) also significantly increased after training on DB_1, because of the new (disjoint) set of symbols introduced. Analogous improvements can be noted in the Intersection over Union (IoU) metrics of the bounding boxes.

Visual error analysis revealed that the 15 videos that were not correctly classified after the second training iteration (see Figure 8) are characterized by: the lack of numerical codes, the presence of textual codes inside the symbol far from the expected position, and very low quality (e.g., out of focus) images. Providing multiple proposals for both the text and number regions could help solve many of these cases.

[7] The SevenPlastics dataset consists of uncorrelated images.
[8] No new sessions were inserted in DB2. The content of DB2 is DB1 plus the newly self-supervised triangle bounding boxes.
[9] Only images with annotated triangle boxes are used to generate negative samples from our videos by taking crops far from the triangle.
[10] IoU is defined as the intersection between the true and predicted bounding boxes divided by their union.

Table 2
Results obtained after training the system on DB_0, DB_1, and DB_2 with a MobileNetV2 model. The whole training procedure (iterations 0, 1, and 2) was repeated 4 times with different initial seeds. Results are reported as average values and standard deviations (within brackets). The *video classification accuracy* refers to the ability to correctly predict the code depicted in the 137 test video sessions by using Algorithm 1. Two failure cases exist: i) *wrong*, if the multi-frame score threshold was surpassed for the wrong code; ii) *unknown*, if the multi-frame score threshold has not been exceeded by any code. The *frame classification accuracy* refers to the binary problem of inferring if the image depicts a recycling symbol and is computed based on the accuracy on the CNN (classification head). The *IoU* metrics[10]were computed on the boxes obtained during the intermediate processing steps.

Training iteration	Video classication accuracy	Frame classification accuracy	Symbol binary classification (CNN)	C - IoU (CNN)	C1 - IoU	Ra refined and Rb refined - IoU
Training iter: 0 Train DB: DB_0 Test DB: T_0	81.20% (3.34) 111.25 (4.57) - correct 6.75 (2.22) - unknown 19 (3.65) - wrong	54.76% (4.96)	92.61% (2.37)	68.65% (4.90)	73.55% (3.98)	Upper: 60.52% (2.52) Lower: 65.71% (2.04) Avg: 61.86% (2.33)
Training iter: 1 Train DB: DB_1 Test DB: T_0	86.31% (2.26) 118.25 (3.10) - correct 3.25 (1.71) - unknown 15.5 (2.08) - wrong	65.70% (0.60)	99.21% (0.17)	80.46% (1.91)	82.58% (1.95)	Upper: 65.53% (1.38) Lower: 70.34% (0.36) Avg: 66.66% (0.77)
Training iter: 2 Train DB: DB_2 Test DB: T_0	88.14% (1.62) 120.75 (2.22) - correct 2 (0.82) - unknown 14.25 (2.99) - wrong	67.54% (1.08)	99.50% (0.18)	83.43% (1.15)	84.30% (0.85)	Upper: 66.92% (2.39) Lower: 71.56% (0.69) Avg: 67.96% (1.76)

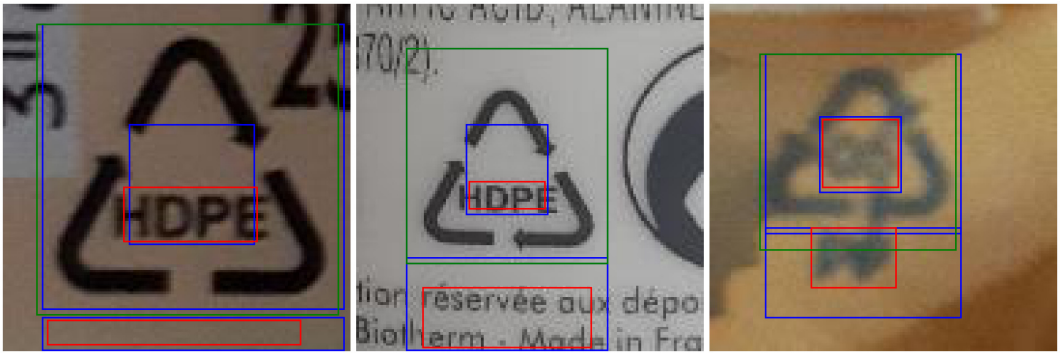

Figure 8: Examples of residual errors after the second training iteration. In the first and second columns, a textural code is printed inside the symbol instead of a number, and its width extends beyond the maximum expected one, so the refined region Ra (red rectangle) cannot include the text border. In the last example, the image is too blurred.

In Table 3, we report the time required by the different stages/steps measured on a Samsung S10+ running Android 11. The most time demanding step is OCR; however, this step is performed only when a symbol is detected and the overall user experience remains quite fluid.

6.1. Frame detection with other architectures

The experiments reported in this section aim to understand whether the use of more powerful deep models would make the symbol initial detection more reliable and the subsequent image processing useless. We first replaced the MobileNet v2 CNN backbone with a ResNet18 backbone (He et al., 2016); in this experiment, we still use the double-head approach of Figure 2 as well as all the successive steps in the pipeline. Results in Table 4 show a slight improvement of accuracy after iteration 0 (e.g., ResNet18 video detection accuracy 81.75% vs 81.20% of the MobileNet), but the advantage is then lost during successive steps (probably due to higher overfitting of the larger model) leading the MobileNet to perform better at the end of the training (e.g., MobileNet video detection accuracy

Table 3
Time required by the different phases/steps, obtained as average on 10 examples.

Step name	Average time per image (ms)
Phase I: Image classification and symbol detection (CNN inference)	24.35
Phase II: S1	1.09
Phase II: S3 (both boxes)	0.48
Phase II: OCR (both boxes)	143.49

Table 4
Results obtained when using a ResNet-18 model as the CNN backbone. The same metrics of Table 2 are here reported.

Training iteration	Video classication accuracy	Frame classification accuracy	Symbol binary classification (CNN)	C - IoU (CNN)	C1 - IoU	Ra Refined and Rb refined - IoU
Training iter: 0 Train DB: DB_0 Test DB: T_0	81.75% 112/137 - correct 6/137 - unknown 19/137 - wrong	57.57%	90.56%	72.43%	75.91%	Upper box: 58.52% Lower box: 66.71% Average: 60.96%
Training iter: 1 Train DB: DB_1 Test DB: T_0	84.67% 116/137 - correct 2/137 - unknown 19/137 - wrong	65.69%	99.16%	81.66%	82.66%	Upper box: 66.63% Lower box: 70.45% Average: 67.40%
Training iter: 2 Train DB: DB_2 Test DB: T_0	86.13% 118/100 - correct 5/100 - unknown 14/137 - wrong	65.56%	96.64%	79.00%	82.50%	Upper box: 67.81% Lower box: 71.41% Average: 68.43%

88.14% vs 86.13% of the ResNet18). Furthermore, the inference time of the ResNet18 (computed on a workstation CPU) is 1.42× with respect to the MobileNet v2.

A second experiment was designed to replace the double-head CNN component with an integrated state-of-the-art detector: Faster-RCNN with Resnet50-FPN backbone. As expected (see Table 5) the initial detection accuracy of this model is still higher, but successive iterations do not bring advantages. At the end of the training, the video classification accuracy is nearly the same as the version using the MobileNet (89.05% vs 88.14% of the MobileNet) and a moderate gain can be appreciated only on frame classification accuracy (73.18% vs 67.54% of the MobileNet). When comparing the inference time (on a workstation CPU), we measure a 467× factor, which clearly makes such a complex model unsuitable for mobile deployment.

6.2. Ablation study

To understand the contribution of the domain-knowledge steps we performed an ablation study consisting of two further experiments:

1. in the former, we removed steps S1 (symbol bounding box refinement) and S3 (text regions bounding boxes refinement). In this case, Ra and Rb are obtained by S2 directly applied to C. Results in Table 6 prove that S1 and S3 have a very relevant role. For example, the frame classification accuracy drops from 54.76% to 21.27%;
2. in the latter, we focused on training iteration 1 where for the examples introduced in P_1 the ground truth of the symbol bounding box is given by C (and not C1); furthermore P_1 contains all the negative examples collected during beta testing and not only those whose text and number are validated after OCR (see Section 5). We denote this scheme as basic self-supervision. Results in Table 7 show that the exploitation of domain knowledge leads to improved accuracy.

6.3. Experimental setup

Experiments were carried out on a server powered by an NVIDIA Titan X (Pascal). PyTorch (Paszke et al., 2019) was used for server-side experiments. For the mobile application, the model was trained and optimized using TensorFlow Lite (Abadi et al., 2016). The custom refinement pipeline is implemented in Android using OpenCV (Java).

Table 5
Results obtained after training the system with a Faster-RCNN (ResNet50-FPN backbone). For this experiment, the triangle box refinement step (S1) is skipped since it was not increasing the bounding box accuracy.

Training iteration	Video classication accuracy	Frame classification accuracy	Symbol binary classification (CNN)	C - IoU (CNN)	Ra and Rb - IoU
Training iter: 0 Train DB: DB0 Test DB: T0	89.05% 122/137 - correct 3/137 - unknown 12/137 - wrong	75.20%	86.67%	92.60%	Upper box: 71.13% Lower box: 76.99% Average: 72.82%
Training iter: 1 Train DB: DB1 Test DB: T0	88.32% 121/137 - correct 3/137 - unknown 13/137 - wrong	74.09%	92.29%	92.44%	Upper box: 70.48% Lower box: 76.89% Average: 72.48%
Training iter: 2 Train DB: DB2 Test DB: T0	89.05% 122/137 - correct 3/137 - unknown 12/137 - wrong	73.18%	91.44%	92.00%	Upper box: 69.58% Lower box: 76.94% Average: 72.00%

Table 6
Accuracy after training iteration 0, in the case of a full system (row 2) and removal of S1 and S3 (row 1).

	Video classification accuracy	Frame classification accuracy	Ra and Rb - IoU
Training Iter: 0 *Without S1 and S3*	67.88% 93/137 - correct 16/137 - unknown 28/137 - wrong	21.27%	Upper box: 25.56% Lower box: 27.59% Average: 26.59%
Training Iter 0 *Full system*	81.20% 111.25/137 - correct 6.75/137 - unknown 19/137 - wrong	54.76%	Upper box: 60.52% Lower box: 65.71% Average: 61.86%

Table 7
Accuracy after training iteration 1, in the case of a full system (row 2) and a basic self-supervision approach (row 1).

	Video classification accuracy	Frame classification accuracy	C - IoU (CNN)	C - IoU	Ra and Rb - IoU
Training Iter: 1 *Basic self-supervision*	82.48% 113/137 - correct 10/137 - unknown 14/137 - wrong	59.34%	74.33%	78.40%	Upper box: 60.95% Lower box: 67.13% Average: 62.45%
Training Iter 1 *Full system*	86.31% 118.25/137 - correct 3.25/137 - unknown 15.5/137 - wrong	65.70%	80.46%	82.58%	Upper box: 65.53% Lower box: 70.34% Average: 66.66%

The models are trained using Adam (Kingma and Ba, 2015) for 100 epochs using a learning rate of $lr = 0.0005$, and betas $\beta_1 = 0.9, \beta_2 = 0.999$. In general, we avoided a tight hyperparameter tuning to mitigate the risk of overfitting; default values have been used when available (e.g., CNN training) and reasonable working values have been set during the initial development for custom steps (e.g. image processing pipeline).

6.4. Efficiency and complexity

From the empirical analysis shown in Table 3, it is evident that most of the time taken by the entire inference pipeline is spent in the CNN inference and OCR steps. When it comes to the choice of architecture to use for the

symbol detector, mobile-oriented architectures are preferable as they are tailored to the hardware (smartphones) of the end users. Examples of such architectures include MobileNets (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019) and EfficientNets (Tan and Le, 2019). As it is shown in Table 3, when properly optimized[11], an architecture like the MobileNet v2 can operate at more than 30 frames per second using single-image batches.

Most of the time is taken by the OCR step. For the OCR engine, we opted for Tesseract (Smith, 2007), which is a popular open-source library. Discussions regarding Tesseract efficiency can be found in the relevant documentation. However, regardless of the choice of the OCR engine, the OCR step is executed only if a symbol is detected, thus increasing the efficiency when no symbols are present in the images.

As shown in Table 3 the time taken for image processing steps S1 and S3 is negligible with respect to the other steps. In general, S1 and S3 require image filtering (with small 3x3 filters) and intensity integration along rows/columns that can be done with $O(K \cdot N^2)$ operations where N is the side of the small image crops considered and K is a small constant.

7. Conclusions

Final users' engagement in waste sorting constitutes a fundamental asset toward greener and more sustainable processes for material recycling. However, product heterogeneity and varying regulations (e.g., depending on geographical location) may pose serious questions about the feasibility and accuracy levels of such classification. While the complete automation of waste sorting is far from being possible in industrial and particularly in household settings, we proposed a vision-based mobile application that may help final users in the waste sorting process by automatically recognizing product recycling classification with high degrees of accuracy.

The developed methodology, based on the simple idea of focusing on recycling codes rather than object visual aspects, blends together state-of-the-art deep learning detection techniques as well as ordinary image processing pipelines and tools. This approach reflects a well-known recent trend in mixing machine learning models with specific domain knowledge to maximize task performance and reduce the overall amount of labeled data needed by the system (von Rueden et al., 2021). Furthermore, such a solution has been designed to run efficiently on embedded systems (e.g., ordinary smartphones) with highly constrained memory and computational budgets.

Finally, we proposed an interactive training scheme to address realistic scenarios where end-users contribute to the overall system prediction improvement only through weak supervision. We believe continual learning procedures (Parisi et al., 2019; Pellegrini et al., 2021) may allow in the future not only for incremental improvements in the recognition of currently considered recycling codes but may also result in the efficient adaptation to new recycling codes being introduced over time. Adaptation capabilities may be leveraged for "personalized learning", with the simple yet effective idea of customizing prediction models to personal, user-specific conditions (e.g., camera settings, more frequent products to recycle, etc.) which may result in additional performance gains.

In future work, we would like to explore such ideas towards the development of continually learning predictive models running completely at the edge, mostly unsupervised or through weak supervision.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X., 2016. Tensorflow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

ASTM International, 2021. Standard practice for coding plastic manufactured articles for resin identification. https://www.astm.org/d7611_d7611m-21.html. doi:10.1520/D7611_D7611M-21.

Baek, Y., Lee, B., Han, D., Yun, S., Lee, H., 2019. Character region awareness for text detection, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9357–9366. doi:10.1109/CVPR.2019.00959.

Bobulski, J., Kubanek, M., 2021. Deep learning for plastic waste classification system. Applied Computational Intelligence and Soft Computing 2021, 6626948. URL: https://doi.org/10.1155/2021/6626948, doi:10.1155/2021/6626948.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

European Commission, 1997. Commission decision of 28 january 1997 establishing the identification system for packaging materials pursuant to european parliament and council directive 94/62/ec on packaging and packaging waste. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31997D0129.

[11] The model was optimized for mobile and quantized (uint8) using TensorFlow Lite.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q., 2019. Searching for mobilenetv3, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324. doi:10.1109/ICCV.2019.00140.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. URL: https://arxiv.org/abs/1704.04861, doi:10.48550/ARXIV.1704.04861.

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

Levenshtein, V.I., et al., 1966. Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, Soviet Union. pp. 707–710.

Liu, M., Zhu, M., White, M., Li, Y., Kalenichenko, D., 2019. Looking fast and slow: Memory-guided mobile video object detection. URL: https://arxiv.org/abs/1903.10172, doi:10.48550/ARXIV.1903.10172.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham. pp. 21–37.

Meeradevi, Sharavana Raju K., V., 2020. Automatic plastic waste segregation and sorting using deep learning model. International Journal of Scientific & Technology Research 9, 5773–5777.

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Networks 113, 54–71. URL: https://www.sciencedirect.com/science/article/pii/S0893608019300231, doi:https://doi.org/10.1016/j.neunet.2019.01.012.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Curran Associates Inc., Red Hook, NY, USA.

Pellegrini, L., Lomonaco, V., Graffieti, G., Maltoni, D., 2021. Continual learning at the edge: Real-time training on smartphone devices, in: 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2021, Bruges, Belgium, October 6-8, 2021, pp. 23–28. URL: https://www.esann.org/sites/default/files/proceedings/2021/ES2021-136.pdf, doi:10.14428/esann/2021.ES2021-136.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788. doi:10.1109/CVPR.2016.91.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, MIT Press, Cambridge, MA, USA. p. 91–99.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Walczak, M., Pfrommer, J., Pick, A., Ramamurthy, R., Garcke, J., Bauckhage, C., Schuecker, J., 2021. Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Transactions on Knowledge and Data Engineering , 1–1doi:10.1109/TKDE.2021.3079836.

Sabater, A., Montesano, L., Murillo, A.C., 2020. Robust and efficient post-processing for video object detection, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 10536–10542. doi:10.1109/IROS45743.2020.9341600.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

Smith, R., 2007. An overview of the tesseract ocr engine, in: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), pp. 629–633. doi:10.1109/ICDAR.2007.4376991.

Tan, M., Le, Q.V., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks, in: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, PMLR. pp. 6105–6114. URL: http://proceedings.mlr.press/v97/tan19a.html.

Ya, P., 2019. Plastic recycling codes dataset. https://www.kaggle.com/piaoya/plastic-recycling-codes.

Yang, M., Thung, G., 2016. Classification of trash for recyclability status. CS229 Project Report 2016.

Zhang, D., Han, J., Cheng, G., Yang, M.H., 2021. Weakly supervised object localization and detection: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence , 1–1doi:10.1109/TPAMI.2021.3074313.

Zhu, M., Liu, M., 2018. Mobile video object detection with temporally-aware feature maps, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5686–5695. doi:10.1109/CVPR.2018.00596.