Sustainable Waste Classification using Deep Convolutional Networks

Sarah Ciresi
Department of Computer Science
Stanford University
sciresi@stanford.edu

Anand Natu
Department of Mechanical Engineering
Stanford University
anatu@stanford.edu

Abstract

One of the principal challenges of municipal waste sorting originates with consumers improperly disposing of waste; this leads to waste streams that are more difficult to sort effectively at recycling plants. In turn, this paper explores the task of automated waste classification into one of five recyclables (cardboard, glass, metal, paper, plastic), compost, or landfill-bound trash. Two different convolutional architectures (ResNet101 and MobileNetV2) serve as the basis for experimentation. Three distinct transfer learning settings are tested: (1) fixed feature extraction; (2) partial finetuning of the network; and (3) full finetuning. We find that full finetuning is the most effective transfer learning treatment for this task. Moreover, while the ResNet101 model achieves superior accuracy (91.6%), the lightweight, resource-efficient MobileNetV2 achieves comparable results (90.1%), suggesting that waste classification can be deployed on mobile hardware to directly address improper recycling practices.

1 Introduction

Despite increased efforts in recent years to educate consumers on best practices for recycling, municipal waste stream contamination remains a serious problem. The ability of consumers to properly differentiate between different waste types and dispose of them appropriately is a key component in ensuring that recyclable and compostable waste is in fact processed separately and not redirected to landfill due to over-contamination.

In turn, we seek to investigate the task of waste classification, i.e. differentiating between waste that is recyclable, compostable, and landfill-bound. We are motivated by the long-term goal of creating a mobile application that allows users to use their phone camera to determine which items can be recycled or composted instead of sent to landfill. Fundamentally, our problem is a multilabel supervised image classification problem. Given an input image of a piece of waste, we classify the input into one of seven categories: trash, compost, or one of five recyclables (cardboard, glass, metal, paper, plastic) and output the appropriate predicted label.

Little work has been done to develop robust assistive tools for consumers, and as such, we explore three separate transfer learning methods utilizing pretrained models to solve this task: fixed feature extraction; partial fineuning; and full finetuning. For model selection, we first consider the highly representative ResNet101 model, in order to assess task feasibility using top-level predictive power [3]. After achieving promising results, we then examine the MobileNetV2 model, a lightweight model specifically tailored for mobile applications[4].

2 Related Work

The problem of recycling-based waste classification is a fairly novel and subsequently understudied task. In our survey of prior work, the work of Yang and Thung (2016) was found to be among the most relevant to this task [11]. The authors implemented a CNN based on the AlexNet architecture to classify images as either landfill-bound trash or into one of five recyclable categories (paper, glass, plastic, metal, cardboard). However, their final CNN model result was ultimately unsuccessful (22% accuracy compared to their SVM baseline accuracy of 63%). The authors attributed their model performance in part to the scarcity of their data, which had mostly been collected by hand, despite their use of data augmentation. The work is relevant in that it highlights primary areas of concern for our own work (data sufficiency), and is a useful starting point for characterizing the classification problem and model architecture.

The work of Bircanoglu et al [2] represents a more recent and successful result building off of Yang and Thung's TrashNet dataset, using DenseNet to achieve high test set accuracies (75%-95%). Chu et al [5] developed a multilayer hybrid model architecture with a CNN based on AlexNet, measured against a pure CNN baseline, achieving test accuracies of >90% on their own waste dataset. These findings are relevant as they demonstrate high performance on TrashNet and other datasets with the use of more advanced convolutional models.

3 Dataset and Data Preprocessing

We use the TrashNet dataset collected by Yang and Thung [11] as the base of our dataset. The dataset contains colored images of waste labelled as one of six classes; five categories of recyclables: paper, plastic, metal, cardboard, and glass, and one 'trash' category. Each recyclable category contains about 400-500 images while the trash class contains 138 images, totaling 2,527 images. We augment the TrashNet dataset with an additional 926 images across the six categories as well as 300 new images of pictures of waste in the novel compost class. We hand-collect the data by taking photographs of waste using cell phone cameras, such that the images resemble input from the intended mobile use case. This results in a full dataset size of 3,753 images.

Our own data is preprocessed to match the TrashNet dataset, such that our images are first resized to the same resolution: HxWxC = 384x512x3. We adapt the resizing code written by Thung and Yang in [11] to accomplish this preprocessing step. We then resize all 3,753 images to dimension 224x224x3 (standard input size for ResNet101 and MobileNetV2 networks). We divide the full dataset into training, validation and test sets, split at 70/15/15 respectively. The combined dataset is still small and we choose to use data augmentation techniques on the training set to help remedy this problem. The transforms include random crops and random flips of the training set images. We also performed manual re-labeling of some of the orignial TrashNet images to comport with our class formulation (e.g. images labelled as plastic which are in fact non-recyclable).

4 Methods

As mentioned above, this task is framed as a multilabel image classification problem in which an input image is classified into one of seven waste classes. The use of transfer learning is a natural choice given the novelty of the problem and the small size of our dataset. However, the exact transfer learning technique that should be utilized is less obvious. Typically, transfer learning is applied via one of three mechanisms: (1) using the pretrained model as a fixed feature extractor, freezing all pretrained layers and only retraining the final linear classifier; (2) keeping a specified number of earlier layers fixed and finetuning only the higher-level portion of the network; or (3) finetuning all layers of the network. Deciding which mechanism to employ for a new task depends on several factors, most notably the size of the new dataset and its similarity to the original dataset. Moreover, the three techniques differ in their computational impact (increased computational load as additional layers are tuned) and have different implications when considering resource-constrained environments.

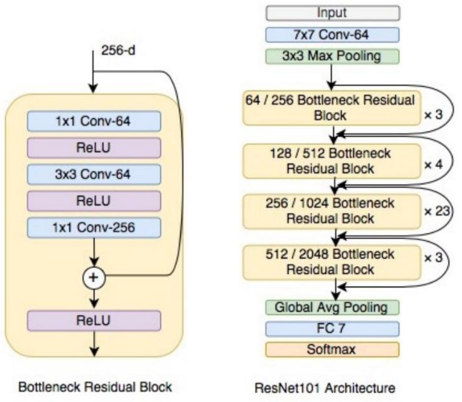
As such, we experiment with the three aforementioned learning techniques and evaluate performance of each applied to two distinct convolutional architectures, both of which are illustrated in Figure 1. We compare these models to a baseline support vector machine model. For the deep models, we use standard cross entropy loss:

$$L_i = -\log \frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}} \qquad L_{CE} = \sum_{i=1}^{N} L_i \tag{1}$$

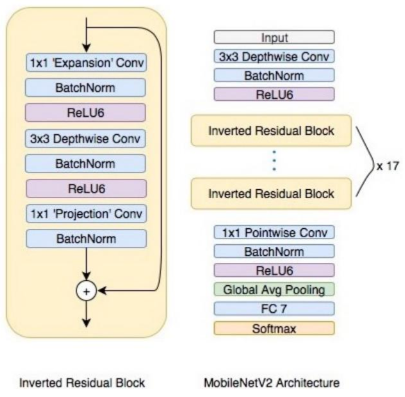
Overall classification accuracy is used as an evaluation metric and confusion matrices are computed for analysis.

4.1 Support Vector Machine with Featurization

Our baseline model for this task is a support vector machine (SVM) classifier with a radial basis function (RBF), with hyperparameter values $\gamma = 1e - 4$ and $C = 1e4$. An RBF SVM baseline was chosen due to the simplicity of its implementation as well as its demonstrable efficacy in recycling image classification tasks [11], thereby making it a suitable lower performance bound for our model. Furthermore, we use a histogram of oriented gradients (HOG) featurization approach for our baseline model after evaluating several different feature representations and observing that the HOG featurization realized the best classification performance.


256-d
1x1 Conv-64
ReLU
3x3 Conv-64
ReLU
1x1 Conv-256
ReLU
Bottleneck Residual Block
Input
7x7 Conv-64
3x3 Max Pooling
64 / 256 Bottleneck Residual Block
× 3
128 / 512 Bottleneck Residual Block
× 4
256 / 1024 Bottleneck Residual Block
× 23
512 / 2048 Bottleneck Residual Block
× 3
Global Avg Pooling
FC 7
Softmax
ResNet101 Architecture

(a) ResNet101 Network Architecture



(b) MobileNetV2 Network Architecture

Figure 1: Convolutional Model Architectures adapted for our task.

4.2 ResNet101

We base our first model on the 101-layer ResNet model proposed by He et al [3], due to its enhanced complexity and greater representative power compared to other architectures, such as AlexNet, previously explored for this task [11],[5]. The ResNet101 model introduces residual connections that enable the construction of much deeper networks while offering lower computational complexity than other deep architectures. Specifically, residual connections reformulate layers as residual blocks which not only feed into the subsequent layer, but also into layers several steps downstream through ("skip connections"). After establishing task feasibility with the complex and compute-expensive ResNet101, we examine the much more lightweight, efficiency-focused MobileNetV2 architecture, which is more well-suited to our task.

4.3 MobileNetV2

Given that our intended target use case is a mobile application, we test the lightweight MobileNetV2 model proposed by Sandler et al [4]. MobileNetV2 is an efficient deep neural architecture that uses depth-wise separable convolutions to build lightweight, low latency models that are well-suited for mobile and embedded vision applications. The depthwise separable convolution factorizes a convolution into separate depthwise and pointwise convolutions. This factorized architecture reduces computational cost by a factor of almost k^2 (i.e. 8-9x cost-savings for 3x3 convolutions). Other techniques such as linear bottlenecks (inserting linear layers to reduce dimensionality) and inverted residuals (“shortcut” connections between bottleneck layers to facilitate gradient flow) are employed to further streamline model memory- and compute-efficiency.

5 Experiments/Results/Discussion

5.1 Transfer Learning Setup

All of our models and experiments are implemented using the PyTorch deep learning framework [9]. For all three transfer learning techniques, we pretrain the convolutional models on the ImageNet dataset prior to applying them to our task. We use [10] to set up the pretrained model for the ResNet101 networks, and slightly adapt the implementation of MobileNetV2 provided by Lin in [8]. We also make use of [6] and [7] for additional guidance. To utilize the pretrained models, we replace the last fully connected layer of each model with a seven-node fully connected layer, corresponding to the seven output classes. We then apply one of the three transfer learning treatments: (1) fixed feature extraction; (2) partial finetuning; and (3) full finetuning.

The deep models are trained using Adam optimization for a duration of 30 epochs and a batch size of 64, as these hyperparameters produced the optimal validation-set results overall across all models. Hyperparameters specific to each model are also tuned on the validation set, and include learning rate, learning rate decay schedule, and number of layers kept fixed (for learning method (2)). For each architecture, we choose the method that performs best on the validation set to use during test-time evaluation.

Model	$\mathcal{L}$	α	γ	η
ResNet101	(1)	1.5e4	0.1	10
ResNet101	(2)	1e4	0.1	7
ResNet101	(3)	1e4	0.1	7
MobileNetV2	(1)	1e2	0.1	7
MobileNetV2	(2)	5e4	0.1	7
MobileNetV2	(3)	8e4	0.1	7

(a) Summary of hyperparameter settings for each deep model, by transfer learning method, $\mathcal{L}$. α = learning rate, γ = learning rate decay factor , η = learning rate decay step size (in epochs).

Model	$\mathcal{L}$	Train Acc.	Val Acc.	Test Acc.
SVM	–	**0.998**	**0.565**	**0.546**
ResNet101	(1)	0.739	0.786	–
ResNet101	(2)	0.976	0.937	–
ResNet101	**(3)**	**0.977**	**0.944**	**0.916**
MobileNetV2	(1)	0.771	0.817	–
MobileNetV2	(2)	0.955	0.911	–
MobileNetV2	**(3)**	**0.952**	**0.921**	**0.901**

(b) Classification performance of all candidate models on the train and validation sets, using each learning method, $\mathcal{L}$. Classification performance of the final models on the test set.

Table 1: Classification performance with the given hyperparameter settings for the candidate models.

5.2 Candidate Model Comparison

Table 1b summarizes the classification performance of the baseline model compared with both deep models under the various learning methods, as evaluated on the training and validation sets, as well as test-set performance of the final three models. Table 1a summarizes the hyperparameter settings for the deep models.

As discussed above, our baseline model is an RBF SVM with HOG featurization. Previous work has indicated that SVM classifiers exhibit enhanced performance using custom features for more complex data representations [1], and exploration of these candidate featurizations revealed that the HOG featurization performed best for our task. Although the very high training accuracy (99.8%) indicates overfitting, performing granular parameter tuning to remedy this issue was practically infeasible due to the very long training times for the RBF SVM.

For the ResNet101 model, we find that the optimal finetuning strategy for learning method (2) is to freeze layers up to and including the four 128/512 residual blocks, and finetune all subsequent layers. As Table 1b shows, the ResNet101 model used as a fixed feature extractor significantly underperforms the other ResNet101 models, but realizes a notable performance gain after finetuning, and obtains the best training and validation accuracy of any model when fully finetuned (validation accuracy of 94.4% and final train loss of 0.1239). Accordingly, we choose the fully finetuned version as the ResNet101 model used for test-set evaluation.

For the MobileNetV2 model, the number of layers kept fixed in the partially finetuned setting is set to 10 after exploring best performance on the validation set. As with ResNet101, learning method (1) fares the worst, with a significant performance jump after finetuning. We observe that the partial and fully finetuned MobileNetV2 models attain training and validation accuracies comparable to ResNet101. We select the fully finetuned model for test-time evaluation given its superior validation performance (92.1%, with a final train loss of 0.1543).

5.3 Final Model Comparison

As shown above, the full-finetuning setting proved to be the most effective transfer learning method regardless of model architecture, and was selected for test-time evaluation in both cases. Further inspection of the test-set performance of the final models illustrates additional significant results. As Table 1b shows, both convolutional models outperform the baseline by a comparable and significant margin. The exact classification accuracies of the deep models on the test set fall in line with the training and validation set results discussed above. Here, we find that ResNet101 model achieves the highest accuracy, at 91.6%, while the MobileNetV2 model is impressively close, reaching 90.1%. The consistency and strength of these results suggest that our strategies for fully finetuning the convolutional models correctly balanced the trade-off between additional finetuning of the model to the task of interest, and the benefits of transfer learning in addressing the limitations of our dataset.

5.4 Error Analysis

Inspecting the confusion matrices in Figure 2 illustrates the superior performance of the deep models over the baseline, in terms of both accuracy (globally across all classes) as well as precision (spread of misclassifications for a given class prediction). This verifies that our convolutional models exhibit better predictive power for all waste classes. Yet, certain misclassification trends are still evident. We see poor performance for the trash category both in terms of precision (many non-trash items incorrectly categorized as trash) and recall (many

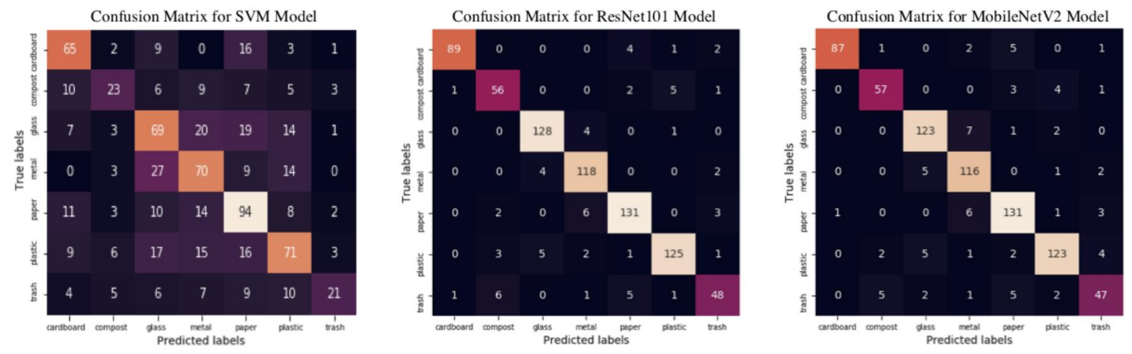


Figure 2: Confusion matrices for the final SVM, ResNet101, and MobileNetV2 models.

trash items incorrectly categorized as non-trash). This may be explained in part by a slight class imbalance; the trash class suffers the lowest number of images. We thus plan to increase this number to remedy the problem.

Figure 3 displays some concrete examples of images misclassified by MobileNetV2 on our dataset. Visual inspection reveals that many of these errors are in fact associated with ground-truth visual ambiguity between two different class types. For instance, certain types of compostable items appear very similar to plastic (e.g. image 4 in Figure 3) and cannot be readily differentiated as compost by a human. We also see cases of misclassification between paper, cardboard, and compost, since both paper and cardboard are typically also compostable (as seen in image 6). Other misclassifications arise from noisy data – images which are blurry, highly zoomed, or poorly cropped (images 1, 2, and 3). This exercise indicates that even for our most lightweight model, typical errors comport with edge cases for human-level recognition, which is an indicator that the model performance is conceptually aligned with the nature of this classification task.

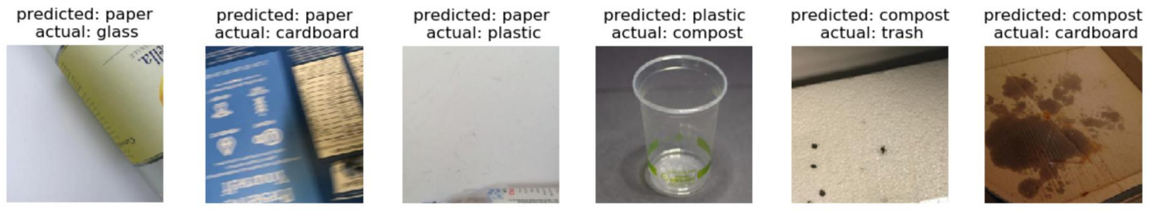


Figure 3: Examples of misclassified images with their ground-truth labels for MobileNetV2 with full finetuning.

6 Conclusion and Future Work

In this paper, we explore three different transfer learning settings applied to ResNet101 and MobileNetV2 for classifying images of waste into seven different categories. We find that all variants of convolutional models outperform our SVM baseline by large margins. While the fully finetuned ResNet101 model achieves the highest test set accuracy among all candidate models, we emphasize the performance of the lightest and most computationally and memory efficient model, MobileNetV2, given our ultimate intended use case. Under full finetuning, the MobileNetV2 architecture achieves an impressive 90.1% accuracy on the test set, demonstrating the potential for this model to be deployed as part of a mobile application.

Based on the preliminary results achieved thus far, as well as our envisioned end use-case for our model, there are several key avenues for further work. Firstly, we would like to continue expanding and enriching our dataset, especially in the 'trash' and 'compost' classes, since data limitations were identified early on as a key challenge for this task, and one which has significant implications on model performance and generalizability. Moreover, as we refine our model in preparation for its intended end use-case of mobile deployment, we plan to consider additional factors beyond simple predictive performance, including: the ability of the model to run on mobile hardware; the model's real-time classification performance on mobile hardware; and incorporating additional instructive guidance along with the classifications in order to aid users (e.g. special disposal instructions or considerations that may apply for certain types of recycling or specific edge cases).

7 Contributions & Acknowledgements

The authors (SC and AN) formulated the waste classification problem to be examined, and SC developed the target use case towards which our experimentation was to be oriented. SC and AN conducted initial literature review and dataset collection, as well as subsequent data augmentation. AN implemented and refined the baseline model for our experiments. SC implemented and refined the convolutional models for our experiments. SC conducted transfer learning experiments for the convolutional models and performed all learning/model hyperparameter tuning. SC and AN contributed equally to the production of the report.

References

[1] Bryan House et. al. "Towards Real-time Sorting of Recyclable Goods Using Support Vector Machines". In: IEEE. 2011.

[2] Cenk Bircanoglu et. al. "RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks". In: IEEE. 2018.

[3] Kaiming He et. al. "Deep Residual Learning for Image Recognition". In: (2015).

[4] Mark Sandler et. al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: (2019).

[5] Yinghao Chu et. al. "Multilayer Hybrid Deep-Learning Method for Waste Classification and Recycling". In: *Computational Intelligence and Neuroscience* (2018).

[6] Sasank Chilamkurthy. *Transfer Learning Tutorial*. Last accessed 3 June 2019. 2017. URL: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`.

[7] Nathan Inkawhich. *Finetuning Torchvision Models*. Last accessed 3 June 2019. 2017. URL: `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html`.

[8] Ji Lin. *A PyTorch implementation of MobileNetV2*. `https://github.com/tonylins/pytorch-mobilenet-v2`. 2019.

[9] Pytorch. *Pytorch, Deep Learning Framework*. URL: `https://github.com/pytorch`.

[10] Pytorch. *Torchvision.models*. Last accessed 3 June 2019. 2018. URL: `https://pytorch.org/docs/stable/torchvision/models.html`.

[11] Mindy Yang and Gary Thung. "Classification of Trash for Recyclability Status". In: (2016).