ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

MLCA Mini-Project: Identifying recyclable/organic waste

MSc in Business Analytics

Machine Learning & Content Analytics

Spring Quarter 2021-2022

Professor: Haris Papageorgiou

Students: Michail Kalligas 2822103

Panagiotis Lolos 2822105

Athens, 2022

Table of Contents

1. Introduction

In this mini-project, we attempt to create a classifier neural network using Keras' MobilenetV2 which can effectively sort through colorized images of items and waste by classifying them either as organic or as recyclable. After thorough research, we identify the above as a much-needed solution for the recycling industry and attempt to finetune it to the best of our abilities, knowledge, and equipment. We use a dataset comprising of approximately 25.000 colored images of waste, labeled O for organic or R for recyclable and augment it accordingly, in an attempt to increase the training data size and the overall model accuracy, while reaching satisfactory results.

1.1. The business problem

Over the past few years, recycling has become ever more important from both an ecological and an economic standpoint. The recycling industry is projected to increase by over 5% on a yearly basis until 2028 (Grand View Research, 2022), mostly due to consumer awareness, the COVID-19 pandemic, and the global energy crisis. Through the evolution of technology, recycling can now be applied in a diverse list of waste categories, reducing production costs and overall waste, improving material efficiency and effectively countering waste pollution (Thøgersen, 1996). One of the largest recycling activities involves sorting organic waste from recyclable waste. An important issue is that citizens mix up recyclable waste with organic waste, which must then be sorted properly before the process begins, to protect the machinery and ensure the materials' preservation. For many materials, such as wood, glass or aluminum, the sorting process is done automatically (Singapore, 2022), for organic waste, however, the process is still done manually through a sorting line. According to *Figure 1*, organic waste comprises nearly half of the total waste generated annually, which proves the necessity of automatizing the sorting process.

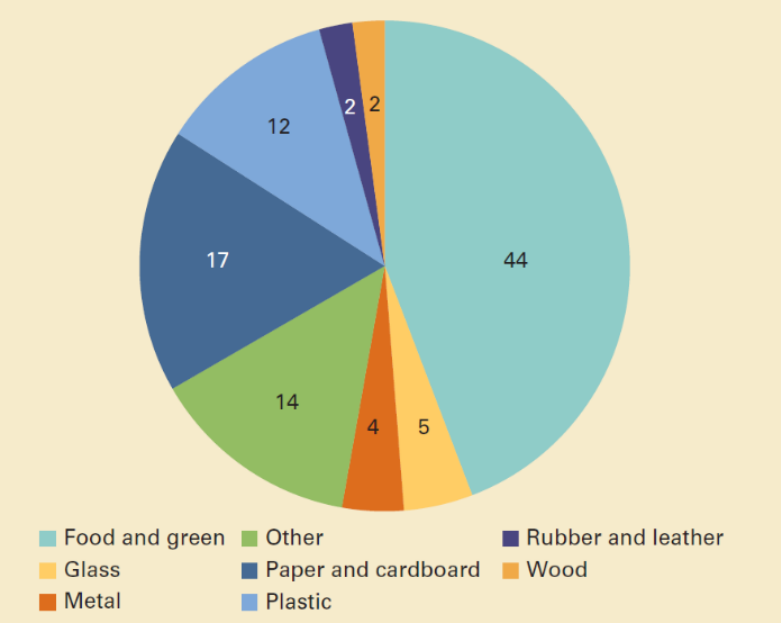

Figure 1: Percentage of waste types, according to "What a Waste 2.0" (The World Bank (IBRD, 2022)

Organic waste is the main component of landfills and a principal factor in land and water pollution (United States Environmental Protection Agency, 2018), even though it is presumably one of the easiest types of waste to properly recycle through composting, aerobic digesting and other physical or chemical procedures (Ni Business Info, 2022). *Figure 2* shows the percentage of solid waste management for the past 58 years.

It quickly becomes obvious that, although recycling, composting and combustion are seeing an increase in usage, the total amount of waste reaching landfills has remained relatively the same over the years, partially due to an overall increase in waste generation.

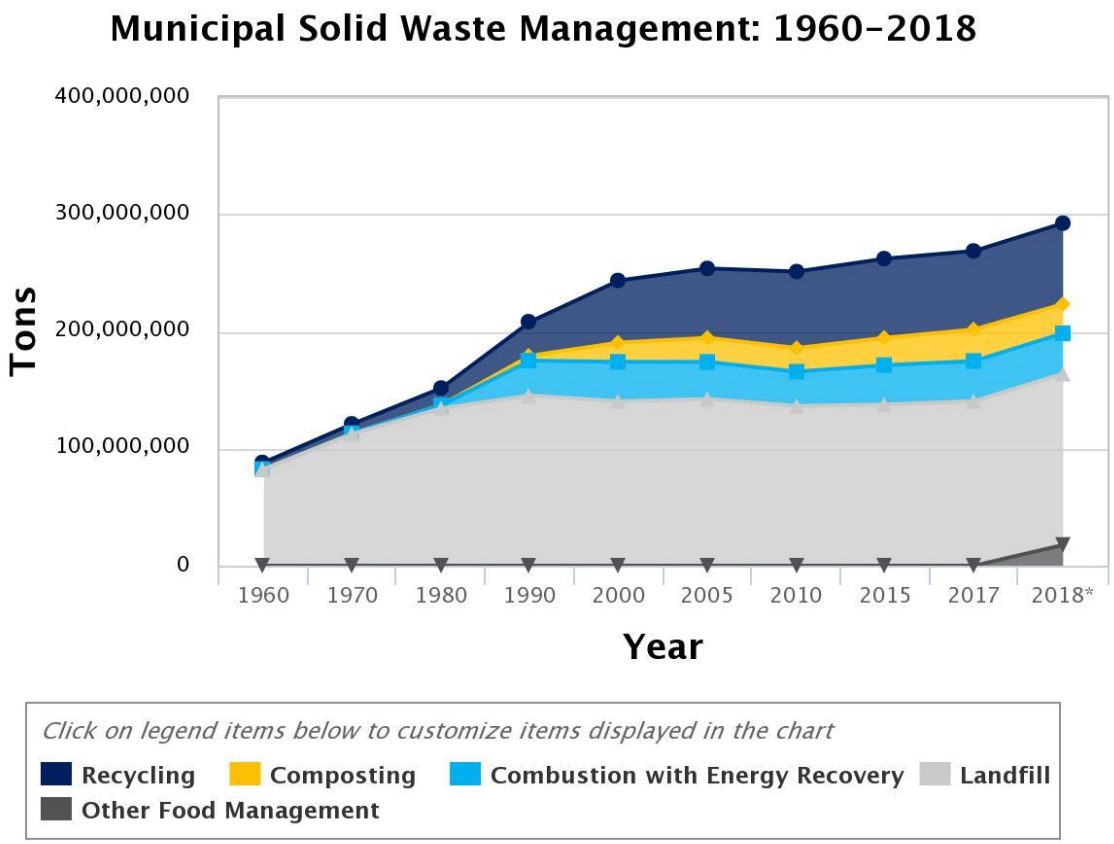

Figure 2: Chart taken from (United States Environmental Protection Agency, 2018)

In other words, correctly identifying, sorting and handling organic waste while filtering other types of solid waste offers great potential for reducing landfills size and producing a multitude of benefits (Rethink Waste TM, 2015) such as:

- Reduction in water, air and earth pollution risks around existing or extinguished landfills.
- Reduction in fertilizer costs by enhancing production through recycled organic fertilizer.
- Improvement of soil fertility and health.
- Significant reduction in methane emissions.
- Increase in gardening, farming and landscaping products which are environmentally friendly.

1.2. Proposed Solution

We propose a theoretical solution for the above problem which requires a multitude of steps:

1. Identifying organic waste during the filtering process of recyclable waste.
2. Instead of just removing organic waste from the production line, re-sorting it into compostable, recyclable, combustible and unusable types.
3. Delivering each type of properly identified organic waste to an appropriate plant.

In this mini-project we will only focus in the first step, as we believe it can become fully automated. The same may also apply to step 2, although we lacked the chemical and biological knowledge as well as the data to implement it computationally. Step 3 is obviously a business process issue, better solved and handled by the recycling companies themselves.

We expect that through steps 1 and 2, the throwaway waste of recycling plants will be reduced up to 75% ideally, while increasing the input of organic waste recycling plants for the same amount.

To go into further detail for step 1, current state of the art recycling plants for non-organic solid waste employ large production lines, where sorting happens mainly automatically, supervised by trained personnel. Our idea is that the same method can be used in plants which receive a mixture of organic and recyclable products, such as product packaging, food scraps, appliances etc. (Sapkota, 2017). Our goal is to create a classifier that could work with a regular image of an item regardless of its' position, rotation or size, since empirically we can identify differences between organic and non-organic waste in color, texture and shape. Our main focus is in identifying and excluding organic waste from the rest of the items, as a way of filtering them out.

1.3. Implementation

Our implementation revolves around creating a convolutional neural network that can identify an item in a picture either as organic or as recyclable. We use a pretrained model from the Keras library, MobileNetV2 (TensorFlow, 2015) by retraining a portion of its layers and adding a simple logistic regression layer at the end which classifies the input as O (=organic) or R (=recyclable). For the training we use the only satisfactory dataset we could find, augmented accordingly to increase training accuracy and reduce overfitting.

Through rigorous re-evaluation and finetuning, we reach a model with an accuracy of nearly 94% and a cross-entropy loss of 0.1. Our notebook, linked at the end of this report contains the entire implementation, as well as a function which can be manually used to inspect the model's results using images outside of the dataset.

2. Methodology

We will now go into details on our implementation, methodology, data used and the theory behind them.

2.1. Data Collection

Due to the problem of sorting recyclable from organic waste being currently handled manually, there is a visible scarcity in available training datasets. In fact, we could only find one satisfactory dataset. There is however a webpage called recycleye.com containing a plethora of filtered and raw datasets which could potentially be immensely useful for training an even better model than the one we have. Unfortunately, our request for access was not granted.

Additionally, we attempted to fetch our own images to create a more compact dataset using different APIs such as OpenCV and CV2. Since we could not find an adequate workaround for removing watermarks from the fetched images and for accurately filtering out background details (which amounted to a lot of noise for image embeddings), and since we could not efficiently solve the above issues in the time we had available, we decided to stick to the previously mentioned dataset, which we will review next.

2.2. Dataset Overview

The dataset we found is the Waste Classification Data by Sashaank Sekar. Henceforth O-R dataset consists of two directories, Train and Test, each containing several images representing household waste items. Each directory contains two subdirectories, one for organic waste (labeled O) and one for Recyclable waste (labeled R). The Train directory contains 22564 images (85% of the total) and the Test directory contains 2513 images (15% of the total).

2.3. Data Processing/Augmentation/Normalization

The total of 25000 images was deemed too small for properly and adequately training our model, thus we used a variety of augmentation techniques from the TensorFlow library to enhance the dataset, such as flipping, rotating and rescaling randomly.

It should be noted that the images in the original dataset have variable scale, pixel size and quality, which posed a problem when creating the embeddings initially. Also, the vast majority of images are colored with blurry (or not) backgrounds and some even contain source labels. We estimate that a cleaner dataset would be better for training, although we could not afford to filter through it considering our need for data quantity. Subsequently, the results of our implementation prove that the pretrained convolutional layers effectively overcame the aforementioned obstacles.

Figure 3 shows a small example of the images contained by the resulting dataset.

Figure 3: Example images from the dataset used for training/testing the data before augmentation

Figures 4 and 5 showcase some of our data augmentation techniques used to enhance the existing dataset. We specifically used TensorFlow's built-in functions:

- RandomFlip(): horizontal and vertical
- RandomRotation(): with a factor of 0.2
- RandomContrast(): with a factor of 0.5
- RandomZoom(): with a factor of 0.3

The above values were determined after multiple trials with the focus on creating a larger diverse dataset which retains the core shapes, colors and textures of the items.

Figure 4: Random rotation augmentation

Figure 5: Random contrast augmentation

Compared to using the original unprocessed dataset, we noticed an accuracy increase of over 5% in our resulting model, which further proves our need for more diverse data. Finally, we rescaled every image to 160x160 pixels, to normalize the input data for training and testing.

2.4. Algorithms and architectures

For our model we used TensorFlow's **MobileNetV2** (Mark Sandler, 2018). MobileNetV2 (henceforth MN2) is a convolutional neural network pretrained on the ImageNet Dataset, a dataset containing over 1.4 million images spread among 1000 classes (Deng, 2009). Before explaining our reasoning behind this choice, it is useful to first describe the model's architecture and specifications.

MN2 is an improvement over MN1 with the goals of object detection, pattern recognition and image classification on a variety of tasks and benchmarks while staying lightweight and available for mobile use. The model implements depthwise separable convolutions comprised of two layer types:

- **Depthwise convolution** which performs simple convolutional filtering on each input.
- **Pointwise convolution** which combines inputs linearly to extract features in 1x1 convolution.

According to the authors, depthwise convolution works just as well as regular convolution but is more cost efficient "*by almost a factor of* k^2*"*, where k equals the convolution size. MN2 uses 3 x 3 depthwise separable convolutions which thus reduces the cost from regular convolutional models by at least 8 to 9 times, while sacrificing a nearly negligible level of accuracy.

To further improve upon cost efficiency, the model also includes linear bottlenecks to reduce dimensionality of the computational space by embedding each input to a low-dimensional sub-space. *Figure 6* shows the different blocks of the separable linear bottleneck and the expansion layer of the convolution, each containing a set of layers with their respective activation functions. The diagonally textured parts only contain linear transformation functions, thus creating a linear bottleneck, which the model's creators prove to be an optimization over MN1 by preventing non-linearities from corrupting significant information. The thickness of each part is used as an indicator of their size in-memory. The two blocks produce equivalent results when stacked, although the bottleneck convolution is obviously much more efficient in terms of memory usage.

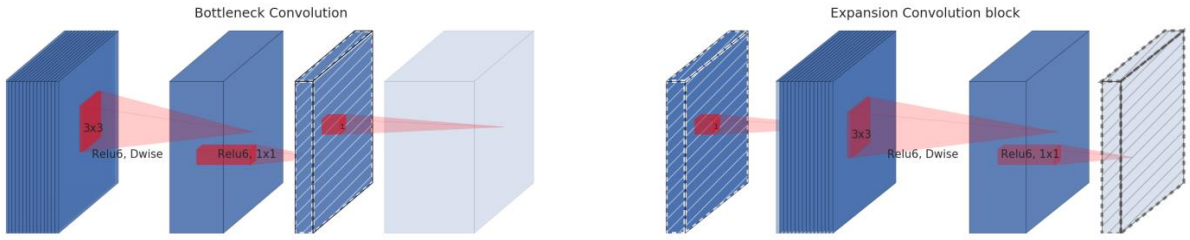

Figure 6: Separable linear bottleneck (left) and expansion layer of the convolution block (right) directly taken from (Mark Sandler, 2018)

Another improvement can be viewed in *Figure 7*, representing a variation of the residual block, the inverted residual block, as well as visible differences between them. The reasoning is the same as with the bottleneck, applied for the expansion layer. The creators use this method as shortcuts between the bottlenecks, which reduce runtime and slightly increase accuracy.

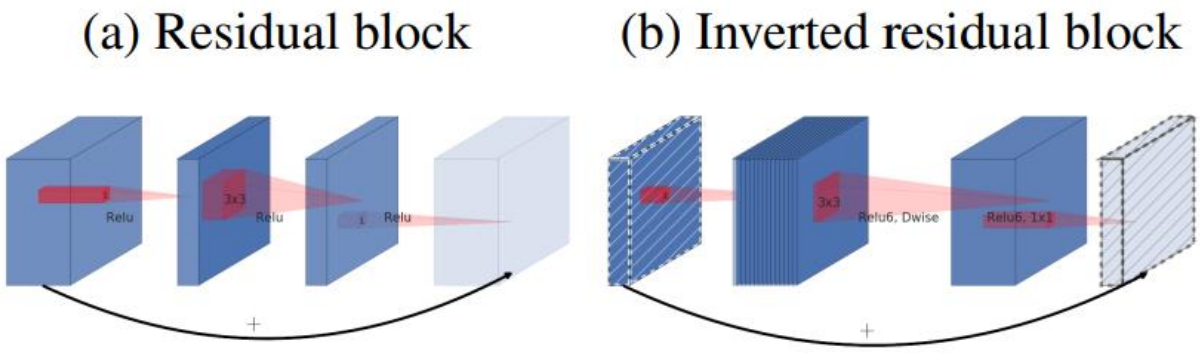

Figure 7: Residual block (left) and inverted residual block (right) showing differences in their size directly taken from (Mark Sandler, 2018)

Conclusively, the model's basic building block is in fact a bottleneck depth-separable convolutional block which incorporates all of the above methods into one highly cost-efficient variation of traditional convolution.

The model also uses a constant expansion rate throughout at a factor of 6, which the authors found experimentally to be the most efficient. *Figure 8* shows the architecture of the building blocks of the model in whole. It allows for creating directed acyclic graphs as part of a much more efficient inferential model than MN1, while still retaining the same capabilities for high accuracy and precision.

Finally, the model has an implementation on the trade-off hyper parameters concerning smaller models, such as our own, which performs better than traditional networks.

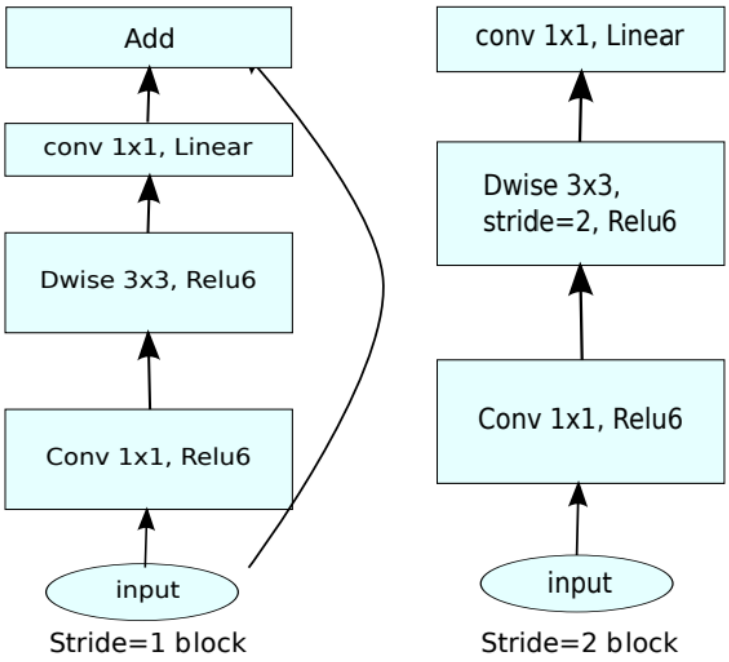

Figure 8: MN2 architecture

The model has been pre-trained into:

- **image classification** using the previously mentioned ImageNet dataset (Deng, 2009),
- **object detection** using the COCO dataset (Tsung-Yi Lin) and implementing the model into a variant of regular SSD: SSDLite,
- **semantic segmentation** using the PASCAL VOC 2012 dataset (Mark Everingham, 2014) and implementing the model into DeepLabv3 as a feature extractor (Jonathan Huang, 2017).

Our choice of MN2 was made strictly based on the above information. To be precise, this model satisfies our need for:

- Accurate pretrained image classification regardless of texture, shape, or color
- Object detection in large and detailed environments or backgrounds
- Lightweight, memory and computationally efficient model capable of being mounted on mobile devices
- Pretrained model on the ImageNet dataset, which contains explicit categories of recyclable and organic waste or other household items such as “food”,” syringe”,” cardboard”, “hand-held computer”, “handkerchief” etc.
- Strong capabilities for accurate inference using smaller training datasets.
- Implementation finetuned for smaller models.
- Ability for semantic segmentation in case of future implementations for further dividing recyclable materials into categories and retraining the model accordingly.

2.5. Our implementation and fine-tuning

We followed the authors' documentation and tutorials when implementing the model and tuning it to cater to our needs. We created a pipeline using the Keras Functional API, which included the following:

- Input reshaping layer
- Data augmentation layer
- TensorFlow's preprocessing input layer
- MobileNetV2 base pretrained model (which includes batch normalization layers by default)
- A global averaging layer
- A dropout layer
- A dense prediction layer functioning as a logit regression function

We first began with feature extraction to use the representations learned by MobileNetV2. The initial model contained 2,259,265 parameters, of which we retrained 1,281 for our purposes. Since we were satisfied with the pretrained weights of the model, we froze the convolutional base for training and only adjusted our own added layers containing global average weights and the predictions/classifications. We trained the model for 20 epochs and compiled the total result with a learning rate of 0.0001. For loss function, since the classification is binary, we used TensorFlow's built-in Binary Cross-entropy. Considering the fact that we were developing our code using cloud tools, we also found it extremely useful to add TensorFlow's callback class to the model, as it helped us keep track of the training process, save the best weights as checkpoints, and prevent several potential overfitting instances. They allowed us to train the model for multiple epochs and to save the optimal version based on the loss function without having to readjust the training procedure midway every time.

We also used a few of Keras' saving-and-reloading functions when developing and debugging the code on the cloud IDE.

For fine tuning we unfroze the last 56 layers of the model according to the model creators' suggestion, increasing the number of trainable parameters to 1,862,721. Since the parameters and weights were now drastically more in number, we also reduced the learning rate to 0.00001 as a counter to overfitting. We retrained the model for another 10 epochs.

The entire implementation was tuned and debugged after constantly inspecting results and consulting MN2's documentation and tutorials. Throughout the entire process, we used current state of the art methods to avoid overfitting, such as a constant dropout rate of 0.2, a rescaling (normalization) of each image to (*160x160x3*), the usage of TensorFlow's preprocess_input() function for transforming each pixel's value from the [0,255] RGB system to the [-1,1] value space acceptable by MN2 and a global average layer summing up the weights before the logit classification.

The output label was decided through a softmax classifier layer, translating each image's probability to belong to one or the other class to the according label depending on an adjusted threshold. For example, for a default threshold of 0.5, any image with a probability of 0.51 or higher to be organic would be labeled organic.

As an optimizer we preferred Keras' RMSprop without momentum, which is a simpler gradient decent method than the widely used Adam, and significantly reduced training times without severely impacting learning rate and results. For our choice we consulted (Jiang, 2020) and experimented with a sample dataset.

3. Experiments – Results

In this section we present our model's results and performance, its shortcomings and how it fares compared to regular convolutional models.

3.1. Quantitative Analysis (Visualizations)

For accuracy, we count the percentage of true positive and negative predictions against the total. We believe this metric to be robust in this case, as the original dataset is fairly balanced (55-45 ratio to organic-recyclable images respectively). Given our limited equipment, dataset and field knowledge, our initial goal was for an accuracy rate of 90-95%. For loss function, we use cross-entropy, which is the most common loss function used for binary classifier models in current state of the art literature. Cross-entropy shows the difference between two distributions, in this case that of the predictions and that of the actual data, much in the same way entropy shows the difference between two random events. Since the prediction is eventually conducted by a softmax algorithm, this function allows us to see how "decidedly" the model classifies each item, as it considers the probabilities before the labeling. Our goal, as with all loss functions, is to minimize cross-entropy as much as possible; however, a value of 0 would almost certainly mean extreme overfitting. Several authors such as (Brownlee, 2020) consider in general any value under 0.2 to be satisfactory, with the ideal ranging in the [0.02-0.05] track and anything below to be highly risky of overfitting. We set our own threshold to accept any loss value lower than 0.2.

Table 1 shows our final measurements in core points of the development, training, and testing process. Our measurements represent the instance closer to the average after multiple reruns of the entire code to present accurate results. We also tested the model with different numbers of training epochs ranging from 5 to 40, to better detect the global minimum for the loss function.

The initial stage refers just to the pretrained model, with the added classification layers untrained. Accuracy and loss were expectedly lower than pure luck since the model attempted to classify only images it had previously seen from the ImageNet dataset.

The trained stage refers to the model after training the added layers to the original O-R dataset. Through callbacks we determined in almost every training session that the optimal number of training epochs was 17, resulting in a loss of almost 0.2 and an accuracy of nearly 90%, which was very close to our minimum expectations.

The fine-tuned stage refers to our final model after training both the added classification layers and the last 50 layers of MN2 using an augmented version of the O-R dataset. The training epochs for just the classification layers remained 17, although we added another 3 epochs of training the entire 56 layers out of 156. The results by far exceeded our expectations with a validation loss of almost always around 0.1 and a validation accuracy of over 94%.

Model Stage	Training Loss	Training Accuracy	Val. Loss	Val. Accuracy	Epochs trained
Initial	-	-	0.783	60.75%	0
Trained	0.227	90.81%	0.186	91.10%	17
Fine-tuned	0.213	91.47%	0.144	94.69%	17 + 3

Table 1: Quantitative measurements of the best model[*]

[*] These numbers were taken from the best run, meaning that the numbers in the deliverable jupyter notebook will be slightly different. The best weights saved to "finetuning_cp1", and the training/val. measurements were retrieved from revision history.

Figures 9 and 10 show the model's evolution in terms of accuracy and loss for both the training and validation sets during a representative iteration of the training process. The green vertical line indicates where the training ends and the fine tuning begins. To avoid misconceptions, we clarify that after the start of the fine-tuning, the model used is the one with the highest validation accuracy and lowest validation loss up until the green vertical indicator, which is on epoch 17, not epoch 19. During fine-tuning, we notice a spike in accuracy after 3 epochs which leads to a global minimum for the validation loss and then both values diverge from their respective training values, in a clear sign of overfitting.

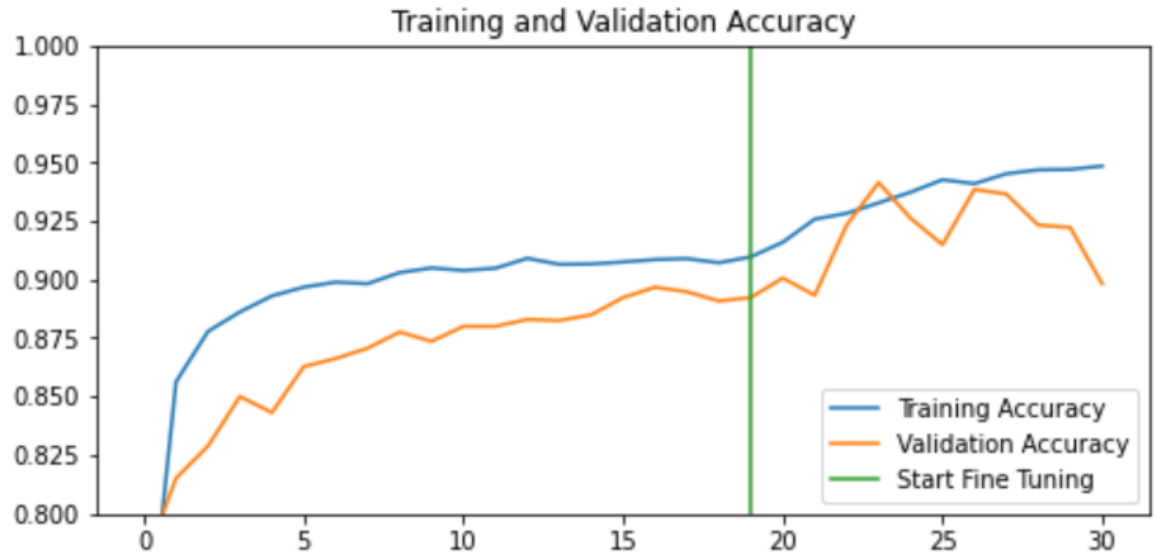

Figure 9: Training and validation accuracy through the training and fine-tuning process

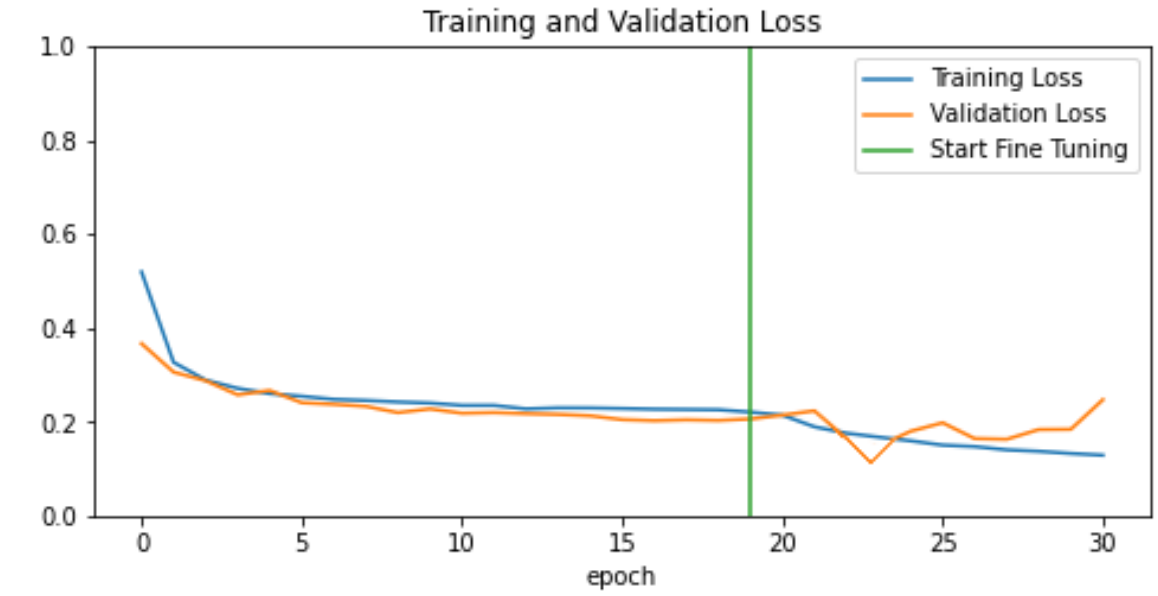

Figure 10: Training and validation loss through the training and fine-tuning process

3.2. Qualitative & Error Analysis

We found it quite insightful to analyze the model's misclassifications and to attempt to confuse it with our own images. *Figure 11* shows an example of the model's predictions. Even though it successfully identifies 8 out of the 9 images correctly and it correctly classifies the bottom left and middle images as recyclable straws, it misclassifies the middle right image as organic, although it contains straws as well. Empirically through multiple iterations we noticed that the model focuses mainly on color, contrast, texture and shape for the classifications. Indicatively we noted the following behaviors:

- Vivid colors with high contrast (those more recognizable to the human eye) are more probably identified to be recyclable.
- Complex and grain textures are more probably identified as organic, whereas untextured items are more probably identified as recyclable.
- Easily defined shapes containing straight lines are more probably identified as recyclable.

Figure 11: Example of the model's classifications

Interestingly enough, the model misidentifies organic images more often than recyclable images. Since our goal is to create a detector that excludes organic materials from the bulk and not the other way around, this is an acceptable evolvement. However, we noticed that the model fails to identify recyclable items with stains or signs of deterioration which would otherwise deem them unrecyclable. In *Figure 12* we showcase one such example. The top left image, probably due to its color is identified as paper and classified as recyclable, which is not the case for used handkerchiefs. The top right image is classified as organic, ignoring recyclable items on the background, which is a point to consider for a detector-filter. The bottom right image is incorrectly classified as recyclable because the dataset did not contain dirty or deteriorated items, which is a point to improve upon. The bottom left image was simply our attempt to see if the model can classify items it has never seen before as recyclables, which it did successfully.

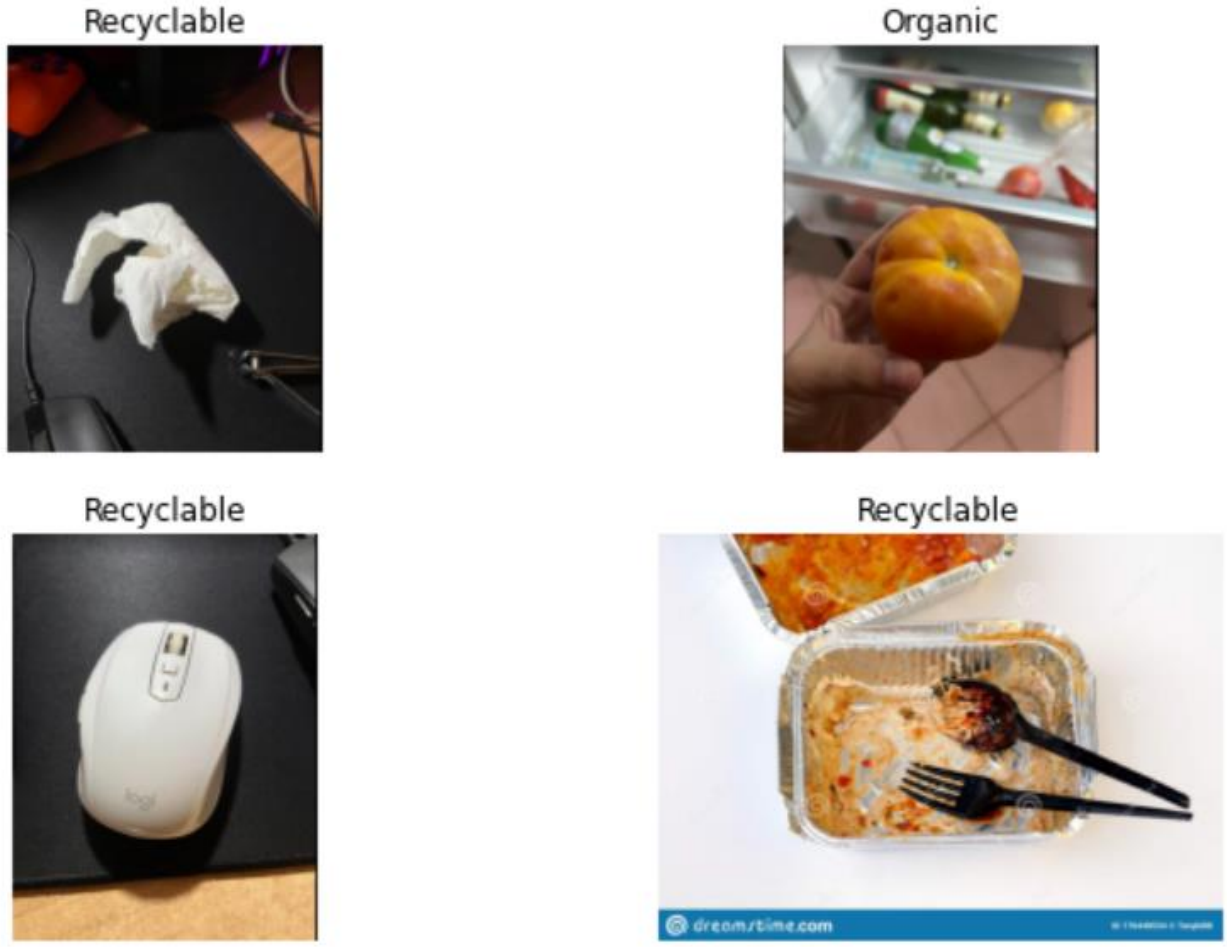

Figure 12: Example of misclassifications made by the model

Another cause for concern is the fact that the model classifies packaging regardless of the content even when it's visible, as shown in *Figure 13.* This is an expected result, but we deemed it noteworthy in case of a practical application of the algorithm, for which we will propose a potential solution later on.

Figure 13: Packaging correctly classified regardless of content, even if it's visible

Figure 14 shows our attempt to prove that the model is biased towards recyclable images. We fed it with images completely irrelevant and probably impossible to be seen by the model in a real-world application. The model classified them all as recyclable due to their shapes, textures and colors being clearer.

Figure 14: Strong indication of the model being biased towards recyclable images when encountering an unknown image.

The above can be problematic when implementing a recycling plant line because not all items reaching it will be as easily distinguishable or as clean and uncorrupted as they were in our training dataset.

3.3. Comparison with regular convolutional models

In this section we will compare our model to that of (Nakkas, 2022), who attempted to solve the same issue by using a regular convolutional neural network and the same original dataset as our own. This section is in no way criticism of their work, rather than an objective comparison example between existing technologies and methodologies.

Their model has a far simpler architecture than ours and requires significantly less training time and resources. It contains 3 convolutional blocks followed by their respective pooling layers, 3 flatten and dense layers with an output layer using the sigmoid activation function. The model also uses a dropout rate of 0.5 on its' dense layers, in an attempt to reduce overfitting. No further fine tuning was conducted, allowing for a very simplistic yet efficient convolutional architecture.

The model is evaluated using the same measurements as we did: accuracy and cross-entropy loss function. *Figures 15 and 16* show the model's training/validation accuracy and loss evolution. After only 4 epochs, the model overfits on a steadily increasing rate.

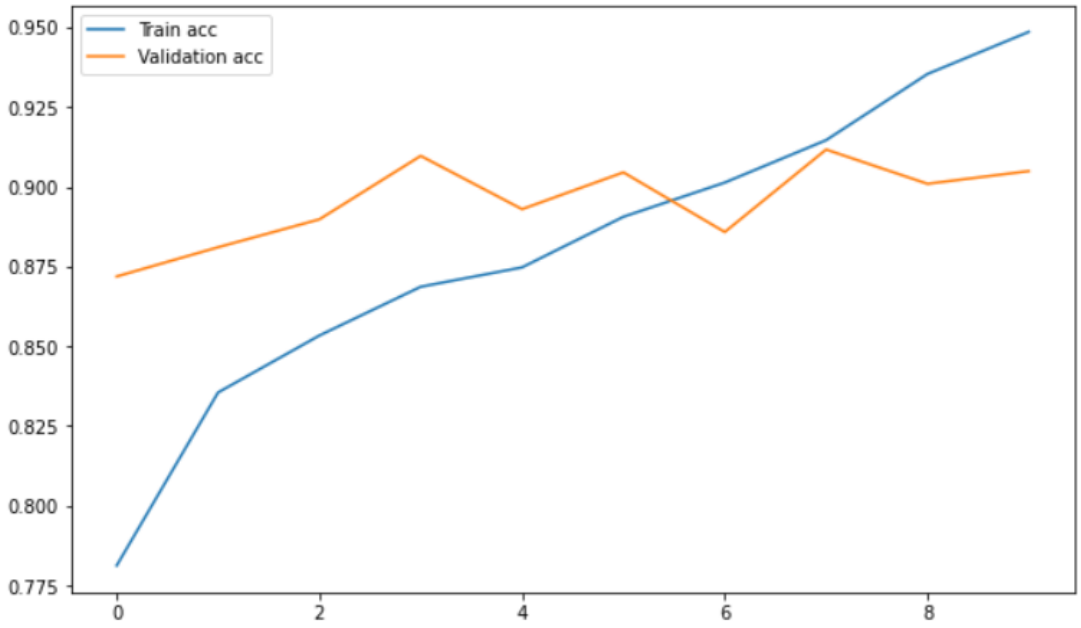

Figure 15: Training and validation accuracy evolution of a regular convolutional neural network

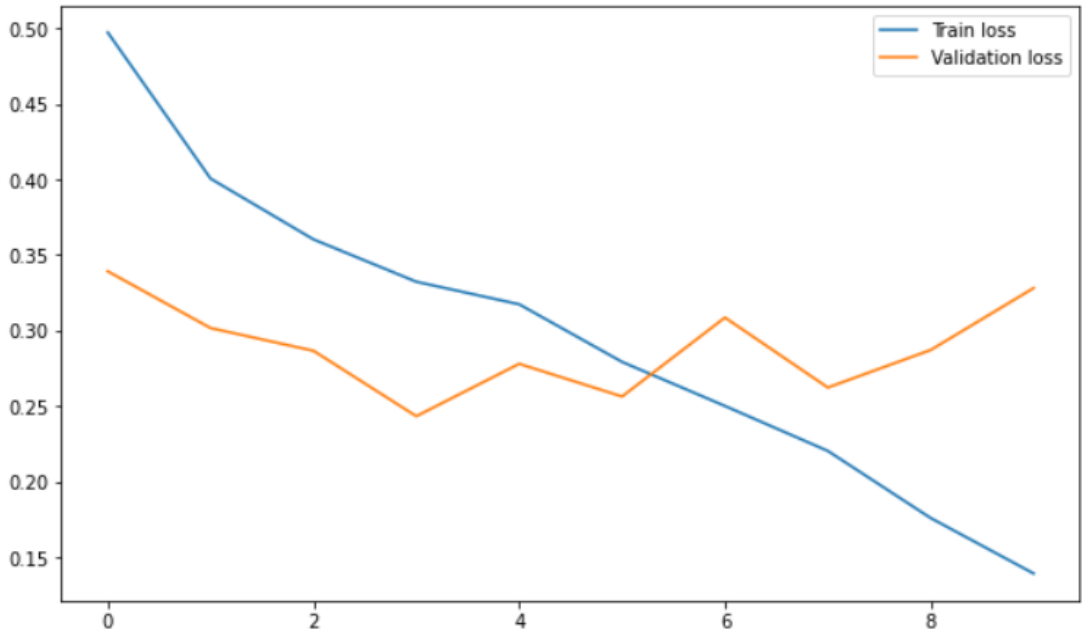

Figure 16: Training and validation loss evolution of a regular convolutional neural network

Table 2 is a direct comparison between our method and the regular convolutional model using an un-augmented dataset to showcase the difference in both training times, overall accuracy, loss and efficiency over the data.

Model	*Training Time*	*Accuracy*	*Loss*
Enhanced MN2	176 mins	94.7 %	0.1445
Regular CNN	17 mins	90.7%	0.249

Table 2: Differences between training and results of the two compared models

Some factors to consider making this comparison wholistic:

- After training, both models offer instantaneous singular predictions.
- Our model is significantly more memory consuming but remains easily mountable on mobile devices given current hardware technology.
- The regular CNN can be improved upon by up to a 2% increase in accuracy and a potential reduction in loss using the augmented dataset for training.
- Both models would benefit greatly by a more diverse dataset, although our model's complexity allows it to evolve faster and more effectively.

3.4. Project overview, comments

In this mini-project we successfully implemented a classifier that filters through images by identifying recyclable materials over organic ones. We researched a variety of potential implementations including regular neural networks, convolutional networks, image classification transformers, image segmentation models and object detection models and concluded that MobileNetV2 is the most suitable to solve our business problem. For our needs we could only find one dataset which contains exclusively household items and organic materials. Through augmentation techniques and rigorous fine-tuning we managed to train our

implemented model to reach an satisfactory accuracy of 96.2% and a loss value within acceptable margins. Finally, we compared our model to a simpler variation and found it superior in both its current state and potential.

3.5. Method shortcomings & Future Work

As mentioned in the previous sections, our method was hindered by a multitude of obstacles:

- Our limited knowledge in chemistry and biology concerning the recycling industry made it impossible for us to create our own large-scale dataset.
- Our business need for a lightweight, easily mountable system made it imperative that we sacrifice accuracy in favor of efficiency and cost reduction.
- The training dataset contained a limited variety of items and waste, which caused our model to possibly overfit into certain item types based on their color, shape, or texture. For the same reason, the model performed poorly in identifying completely different items than the ones it had seen before.

Considering our initial goals, the business problem as a whole and the obstacles we encountered in this project, we therefore propose a few future actions:

1. Retraining or fine-tuning the implemented model using a larger, more diverse and more balanced dataset, with an expected reduction in cross-entropy loss to 0.05 for out-of-sample classifications.
2. A transformation of the model from binary classifier to a classifier of multiple types of organic waste (combustible, compostible, etc), as well as a more detailed classification of recyclable waste (paper, glass, batteries, appliances, etc), with a specific focus on maintaining the system's lightweight nature.
3. Deploying and experimenting with the model in a real sorting line of a recycling plant to identify unknown shortcomings, biases and issues that may arise which are impossible to detect theoretically.
4. Ignoring completely the need for a cost effective system, an attempt with transformers or more complex architectures such as Microsoft's Swin Transformer (Ze Liu, 2021) or Google's ViT (Bichen Wu, 2020).

We remain positive that our model can adequately solve the presented problem provided with the right equipment, data and experimentation.

4. Project Log

4.1. Members/Roles

Both members were equally involved in all aspects of this team project, with each one assuming the lead role on one aspect while still cooperating on the other.

Michail Kalligas led the code development and compilation.

Panagiotis Lolos led the reporting and research.

We would also like to acknowledge Mr. George Perakis for his continued guidance and the provision of teaching materials which proved extremely helpful to our mini-project.

4.2. Time Plan

Early June: conception of the idea

Early-Mid July: research of the business need, current state of the industry, existing solutions

Early August: model development and fine-tuning

Mid August: Report and project conclusion

5. Bibliography

Bichen Wu, C. X. (2020). *Visual Transformers: Token-based Image Representation and Processing for Computer Vision.* Retrieved from arXiv/2006.03677: https://arxiv.org/abs/2006.03677

Brownlee, J. (2020, 12 20). *Cross-entropy for machine learning.* Retrieved from machine learning mastery: https://machinelearningmastery.com/cross-entropy-for-machine-learning/

Deng, J. D.-J.-F. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255). IEEE.

Grand View Research. (2022, 8 22). *Waste Recycling Services Market Size, Share & Trends Analysis Report.* Retrieved from Grand View Research: https://www.grandviewresearch.com/industry-analysis/waste-recycling-services-market

Jiang, L. (2020, June 7). *A Visual Explanation of Gradient Descent Methods.* Retrieved from Towards Data Science: https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Jonathan Huang, V. R. (2017). *Tensorflow object detection api*. Retrieved from https://github.com/tensorflow/models/tree/master/research/object_detection

Mark Everingham, S. M. (2014, June 25). The PASCAL Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, pp. 98–136.

Mark Sandler, A. H.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 4510-4520). doi:https://doi.org/10.48550/arXiv.1801.04381

Nakkas, B. N. (2022, 6 1). *Waste classification with CNN.* Retrieved from kaggle.com: https://www.kaggle.com/code/beyzanks/waste-classification-with-cnn/notebook

Ni Business Info. (2022). *How to Recycle your Organic Waste.* Retrieved from nibusinessinfo.co.uk: https://www.nibusinessinfo.co.uk/content/how-recycle-your-organic-waste

Rethink Waste TM. (2015, May). *The Importance of Recycling and Composting.* Retrieved from rethinkwaste.org: https://rethinkwaste.org/2020/06/30/the-importance-of-recycling-and-composting/

Sapkota, A. (2017). *Organic Waste Recycling (Methods, Steps, Significance, Barriers).* Retrieved from microbenotes.com: https://microbenotes.com/organic-waste-recycling/

Singapore, N. E. (2022, 8 22). Types of Recyclables and Recycling Processes. Retrieved from https://www.nea.gov.sg/our-services/waste-management/3r-programmes-and-resources/types-of-recyclables-and-recycling-processes

TensorFlow. (2015). *MobileNet, MobileNetV2, and MobileNetV3*. Retrieved from Keras.io: https://github.com/keras-team/keras/blob/v2.9.0/keras/applications/mobilenet.py#L80

The World Bank (IBRD, I. (2022, 8 22). *Trends in Solid Waste Management.* Retrieved from What a Waste 2.0: https://datatopics.worldbank.org/what-a-waste/trends_in_solid_waste_management.html

Thøgersen, J. (1996, July 1). Recycling and Morality: A Critical Review of the Literature. *Environment & Behavior, 28*(4), pp. 536-558. doi:10.1177/0013916596284006

Tsung-Yi Lin, M. M. (n.d.). Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science* (Vol. 8693). doi:https://doi.org/10.48550/arXiv.1405.0312

United States Environmental Protection Agency. (2018). *National Overview: Facts and Figures on Materials, Wastes and Recycling.* Retrieved from epa.gov: https://www.epa.gov/facts-and-figures-about-materials-waste-and-recycling/national-overview-facts-and-figures-materials

Ze Liu, Y. L. (2021, 4). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *CoRR, abs/2103.14030*. Retrieved from https://arxiv.org/abs/2103.14030

6. Appendix

6.1. Jupyter Notebook

This pdf contains the outputs of our latest Jupyter Notebook. As mentioned earlier, some numbers may differ because of multiple reruns, but the core remains the same.

Jnpdf_F2822103_F2822105.pdf