BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE ENGLISH

DIPLOMA THESIS

Mobile Application for Identifying Plastic Recycling Symbols using Image Classification and OCR

Supervisor,

Lect. PhD. Arthur Molnar

Author,

Foidaș Andrei-Ștefan

2022

Abstract

The main purpose of my thesis is to make information about recycling plastic easier to understand and very quickly accessible through the "Recycle Buddy" application. Recycling Buddy is an Android application developed in Kotlin that allows the user to take a photo (or select one from the phone's gallery) of a recycling sign and providing information about the respective sign, such as if it is reusable or not and if it is recyclable or not.

The thesis is structured into several main chapters. The first chapter studies the theoretical part of the image classification models used in the application, the third one presents the challenges I encounter while implementing this idea and the results of my studies and experiments, while the last one presents the mobile application: the theory behind it, the implementation and the user manual.

When I first started, the accuracy of the image classification models was only about 30 – 40%, but I managed to increase it to well over 75% by several means: collecting more images, (editing the set of images), image augmentation, and most importantly, k-fold cross-validation.

- Complete Abstract TODO: talk about experiments and results, talk about app

Table of Contents

1. Introduction

A study published in the journal Science Advances shows that out of 8.3 billion metric tons of produces plastic, 6.3 billion metric tons becomes waste, meaning that only 9% is being recycled. But recycling is not an easy task; it can be very tedious and complicated. There are over 8 plastic recycling symbols, each of them specifying how one should recycle the plastic or even if it should be recycled, and if that plastic can be reused or it can become toxic for your health. Moreover, all signs are really similar and it’s really hard to remember what they all mean.

The main purpose of my thesis is to make information about recycling plastic easier to understand and very quickly accessible through the “Recycle Buddy” application. Recycling Buddy is an Android application developed in Kotlin that allows the user to take a photo (or select one from the phone’s gallery) of a recycling sign and providing information about the respective sign, such as if it is reusable or not and if it is recyclable or not.

The classification of the photo is done using Image Classification combined with OCR for more accurate results. The classification poses a great challenge since the recycling sign are very similar and there are no large datasets of the signs available. Throughout the thesis I will study different Image Classification models, comparing their results, as well as gathering as much data and photos of plastic recycling sign and monitoring the evolution and performance of the models.

- Introduction TODO
- Chapter about plastic waste problem with statistics + motivation TODO

2. Image Classification of Plastic Recycling Signs

2.1. Theoretical Foundation

The human brain can easily distinguish and recognize in an image different shapes, patterns, colors and so on. For example, given an image two images, one with an apple and one with a banana, we immediately differentiate them without issues. However, when a machine manages to do it, we enter the subject of Artificial Intelligence. Precisely, we are talking about Computer Vision [8].

2.1.1. Computer Vision and Image Classification

Computer Vision wants to mimic the human vision, except the human sight has the advantage of thousands of lifetimes when it comes to training.

Its field covers numerous distinct applications including image classification, object detection, image segmentation and localization. However, Image Classification might be regarded a fundamental problem that serves as the foundation for all other Computer Vision applications.

Image Classification [4] has the task of labeling a set of photos or a fragment of a photo based on different categories that are separated by a set of rules. Its applications vary from the medical imaging like cell classification and tumor detection, to object identification in satellite images, to animal and plant monitoring and so on. The Image Classification techniques can be divided into two categories: unsupervised and supervised image classification.

The ability of the algorithm to analyze and cluster unlabeled datasets by locating patterns in data without the need for human involvement is known as unsupervised learning. This method does not require training data, thus being fully automated. Supervised learning on the other hand, utilizes previously classified samples in order to train and be able to classify new, unknown data.

2.1.2. Convolutional Neural Network

The convolutional neural network (ConvNet or CNN) is a type of deep neural network (DNN) that has shown remarkable performance in computer vision applications, particularly image classification. The Convolutional Neural Network is a special form of multi-layer neural network that is inspired by the human optical and neural systems. The usage and development of CNNs such as DenseNet, ResNet, GoogleNet, VGGNet, and many others began in 2012 when AlexNet, a large deep convolutional neural network, performed exceptionally well on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Convolution layer, Pooling layer, ReLU layer, and Fully connected layer make up the fundamental CNN architecture [1], [4].

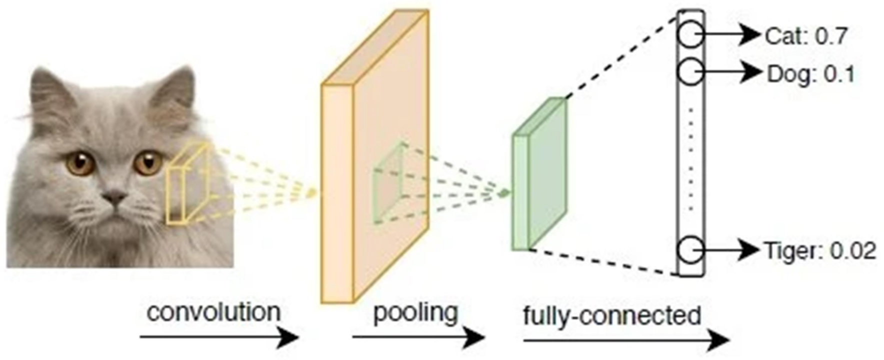


Fig 1. Convolution Neural Network, Source: [4]

Convolutional Layer, primarily for image recognition, provide the true potential of deep learning. It is the first and most important layer. This layer uses a convolution neural network to convolve the whole images as well as the intermediate feature maps, generating multiple feature maps. A feature map is a map that connects input layers to hidden layers.

Rectified Linear unit Layers, or **ReLu Layers**, are activation functions used to reduce overfitting and improve the accuracy and efficacy of CNNs. These layers make it easier to train models and deliver more accurate results. This is a layer of neurons that uses the loss function or non-saturating non-linearity function:

$$f(x) = max(0, x)$$

We can also have the tanh function:

$$f(x) = tanh(x)$$

Or the logistic sigmoid function

$$f(x) = \frac{1}{(1 + e^{-x})}$$

The **Pooling Layer**'s job is to minimize or simplify the spatial of feature map information. It also gathers and analyzes the results of all neurons in the layer before it. We have three types of pooling: average pooling, L2-norm pooling, and max pooling, which is the most used because of its speed and improved convergence.

The Fully Connection Layer receives a vector of integers as input. The concept "fully connected" refers to the fact that it is connected to all of the outputs. It is the last layer in the CNN process, generally after the last pooling layer. Fully connected layers mimic the behavior of a typical neural network and include around 90% of the parameters found in the convolution neural network. This layer takes the previous pooling layer's output as input and returns an N-dimensional vector, where N is the number of classes from which the program must decide. It enables us to send forward the neural network into a preset length vector. We might input the vector into specific number categories for image classification. The output is a vector of numbers as well.

The optional **Loss Layer** computes a number that assesses the model's performance using functions that take in the model's target and input. It can have two main functions:

- Forward, with two parameters, input and target: computes the loss value depending on the input and target values.
- Backward, with two parameters, input and target: returns the result after calculating the gradient of the loss function linked with the criterion.

Backpropagation is a concept employed in CNNs to determine the gradient of the loss function (which determines the cost associated with a given state).

2.1.3. Transfer Learning

Humans are born with the ability to transfer information from one activity to others by applying the knowledge gained from doing one specific activity to address similar problems in a related manner. By the word “transfer” we refer to the cognitive process in which a learner’s mastery of knowledge or ability in one context allows them to reapply those same knowledge or abilities in another. There are a few drawbacks of working with the traditional deep learning or machine learning algorithms like the fact that the training step requires plenty of data and collecting sufficient information can be a very tedious and time consuming task. Moreover, those algorithms are usually built to function with a predefined feature-space distribution and the model must be completely rebuilt from the ground up if the feature-space distribution changes [2].

When we are trying to create a model whilst having insufficient data, it would be very challenging to do it using a machine learning-based algorithm and this is where transfer learning can provide a considerable boost when it comes to the learning performance. Transfer’s learning core concept is to use knowledge or labeled data gathered from related domains to aid a new algorithm in achieving better performance in the area of interest. Nowadays, it is preferred to begin with a pre-trained model that has learned various features like shapes or edges in images and that can already classify objects.

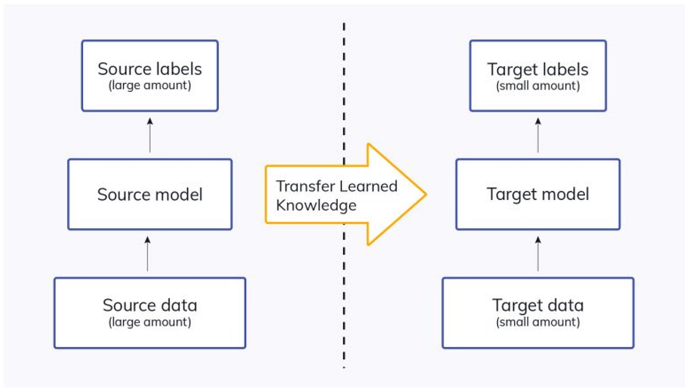


Fig 2. Transfer Learning, Source: [6]

Deep transfer learning [6] is referred to as the process of using pre-trained models or neural networks as a foundation for transfer learning with regards to deep learning. As stated in the previous subchapters, deep learning networks are organized in layers and learn various features at different layers. Higher-level features are gathered in the first layers, which narrow down to very detailed features as we go deeper into the system.

In order to achieve the final output, the layers in the network are connected to a final layer (in case of supervised learning, usually a fully connected layer). This allows the usage of popular pre-trained models like Google’s EfficientNet or Inception models, Microsoft’s ResNet models and Oxford’s VGG models for various tasks, without the last layer acting as a fixed feature extractor. The main aspect here is using the weighted layers of the pre-trained network to extract features rather than updating the model's weights during training with new data for the new task. The pre-trained networks will essentially act like a generic model of the visual world since they were developed on a generic and wide enough dataset.

Some pre-trained models that proved to fit the recycling signs model the best were Oxford’s VGG19 model and Google’s EfficientNet-B7 model.

- Freezing vs finetuning [7] TODO

2.2. Pre-Trained Models

2.2.1. The VGG19 Model

With Convolution Networks becoming increasingly common in computer vision, a number of attempts have been made to modify Alex Krizhevsky's initial model design [10] in order to increase accuracy. The best entries to the ILSVRC2013 (ImageNet Large-Scale Visual Recognition Challenge), for example, had a smaller receptive window size and a smaller first convolutional layer stride [11]. Another series of developments aimed on testing the network and densely training throughout the whole image and over several scales [12].

Another significant feature of the convolution neuronal network's architecture design is its depth. In order to improve the design based on depth, the other parameters of the architecture need to be fixed and the network's depth to be gradually increased. This is done by adding additional convolutional layers, which is made possible because of the usage of a relatively small (3x3) convolution filters in all layers.

This is how *Oxford's VGG* family of models were developed [9], having significantly more accurate ConvNet architectures that are not only applicable to other image recognition datasets, but also perform well even when used as part of a relatively simple pipeline, achieving state-of-the-art accuracy on ILSVRC classification and localization tasks.

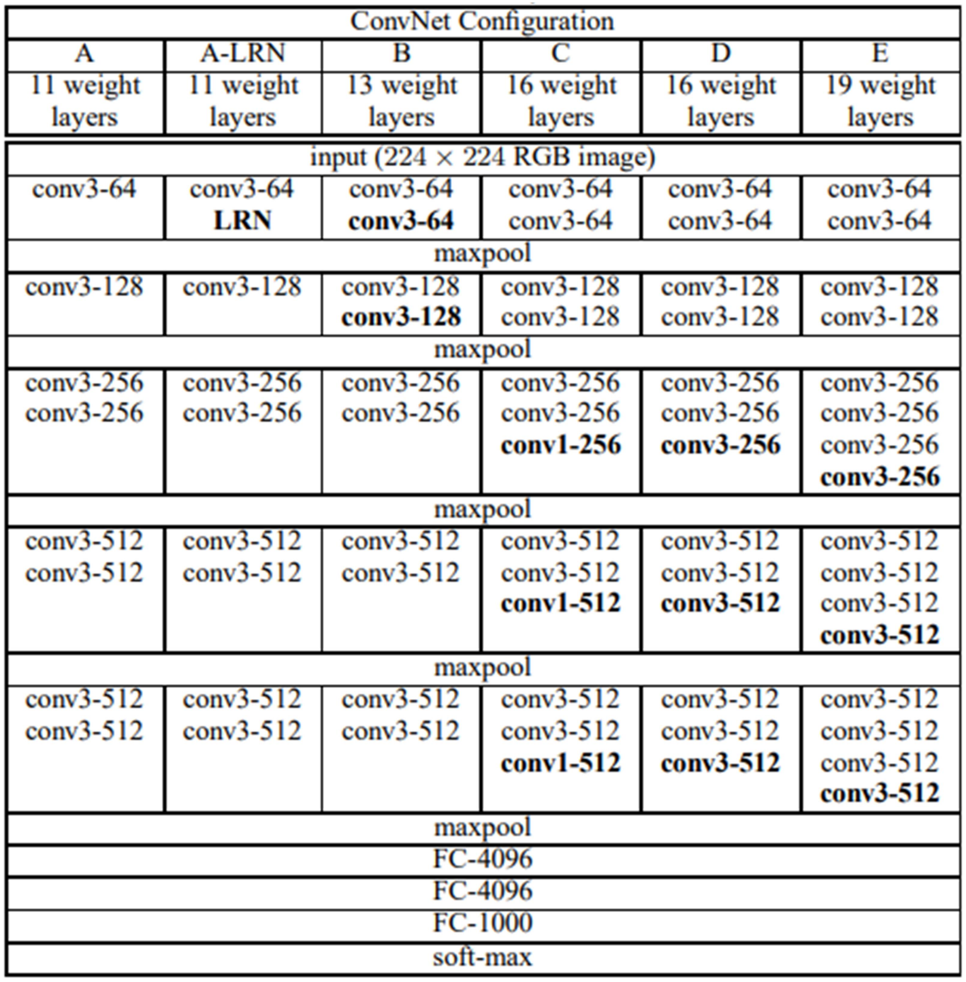
In terms of the models' architecture, the network was given a fixed-size (224x224) RGB image as input, implying that the matrix is of shape (224, 224, 3). The only preprocessing done was subtracting each pixel's mean RGB value, which was computed throughout the whole training set. To transmit an image through a stack of convolutional layers, filters with a very small receptive field (3x3) are used. In order to retain spatial resolution, the convolution stride is set to 1 pixel and the spatial padding of the convolution layer input is applied. Max-pooling is done with a stride of 2 pixels cross a 2x2 pixel frame.

Following a stack of convolutional layers are three Fully-Connected (FC) layers: the first two have 4096 channels each, while the third has 1000 channels since it performs 1000-way ILSVRC classification. A soft-max layer is the final layer.

In order to increase non-linearity and improve processing performance, all hidden layers are followed by a Rectified linear unit (ReLu). Previous models used tanh or sigmoid functions, but this approach outperformed them significantly.

The VGG configurations differ significantly from the top-performing entries from the 2012 and 2013 ILSVRC competitions [10, 11]: extremely small 3x3 receptive fields are employed throughout the net, which are convolved with the input at every pixel, rather than utilizing comparatively big receptive fields in the initial convolution layers. It's clear that a stack of two 3x3 convolution layers (without spatial pooling in between) has an effective receptive field of 5x5 and three similar layers has a receptive field of 7x7.

The configuration of the convolution neural network is showed in the Figure 2.2.1.1. As more layers are added, the depth of the configurations grows from left (A) to right (E); the added layers are shown in bold. **VGG19** is shown in column; other columns represent various VGG model variants.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 2.2.1.1. VGG Convolution Network configurations

2.2.2. The EfficientNet-B7 Model

- Add references TODO

In order to achieve accuracy, Convolution Neuronal Networks are frequently scaled up in different ways. Some examples are: ResNet that can be scaled up from ResNet-18 to ResNet-200 by adding more layers and GPipe which achieved in 2019 Top 1 with an accuracy of 84.3% ImageNet by scaling, four times bigger, a baseline model. There are now several approaches to scaling up CNNs because the procedure isn't very well understood. The common approach when it comes to scaling CNNs is to increase the width or the depth. Scaling up models by image resolution is a less common strategy but is growing in popular recently. In past work, it was usual to scale just one out of the three dimensions – image size, width, depth. While still two or three dimensions can be scaled freely, it involves time-consuming manual adjustments and frequently leads to sub-optimal results in efficiency and accuracy.

Mingxing Tan and Quoc V. Le's research [3] revelaed how balancing all dimensions of the netwrok – resolution, width, depth – is crucial and can easily be achieved by simply scaling each of them with a constant ratio. They proposed a simple yet efficient *compound scaling strategy* based on that result. This strategy implies scaling the models resolution, width and depth evenly, using a set of predfined scaling coefficients. As an example, if we wish to utilise 2^N times more computing resources we simply do the following: make a small grid search on the initial small model that generated several constant coefficients – α, β, γ – we increase the image size by α^N, width by β^N, and depth by γ^N.

This was the first apporach to objectively measure the link between all three dimensions of network: resolution, width and depth. Because the CNN requires additional layers to extend the receptive field and several channels to detect a lot of fine-grained features as the input image is larger, the compound scaling technique makes sense.

The difference between the scaling approach proposed by Mingxing Tan and Quoc V. Le and traditional methods is seen in Figure 3 below.

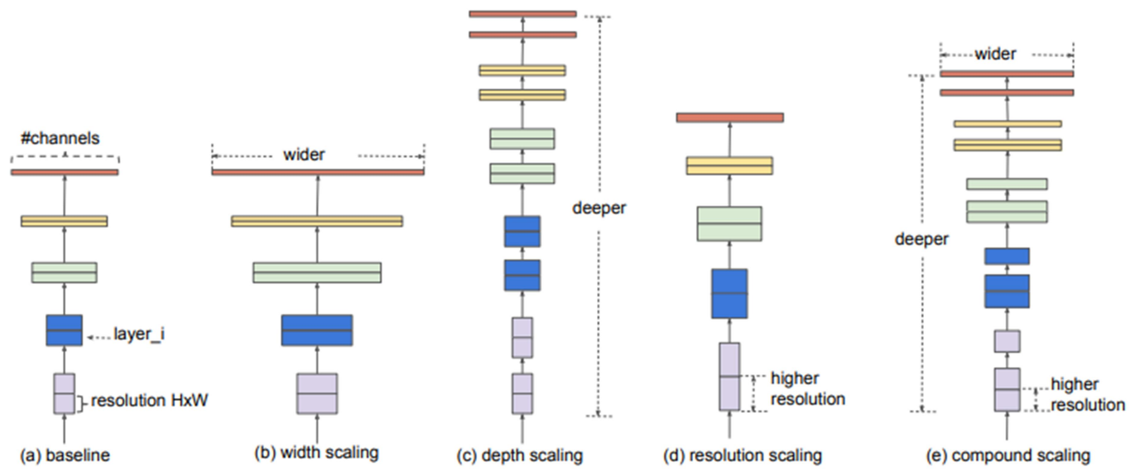


Fig 3. Model scaling, Source: [3]

Since the efficacy of model scaling is highly dependent on the baseline network, the *EfficientNets* family of models was created by scaling up a newly developed baseline network. This baseline network was created using a neural architecture search.

Some achievements of the EfficientNet family of models are: EfficientNet-B7 outperforms the highest available GPipe accuracy while using 8.4 times less parameters and operating 6.1 times quicker on inference. With identical FLOPS (floating-point operations per second); when compared to the frequently used ResNet-50, EfficientNet-B4 improves top-1 accuracy from 76.3 percent to 83.0 percent. In addition to ImageNet, EfficientNets transfer well and achieve state-of-the-art accuracy on 5 of the 8 most commonly used datasets while reducing parameters by up to 21 times when compared to traditional ConvNets.

The architecture of the baseline network - called EfficientNet-B0 - is comparable to MnasNet since the same search space is used, with the exception that the EfficientNet-B0 is significantly larger owing to the increased FLOPS target (the FLOPS target is 400M). Its key building block is the MBConv (mobile inverted bottleneck convolution, similar to MobileNetv2), to which squeeze-and-excitation optimization is applied.

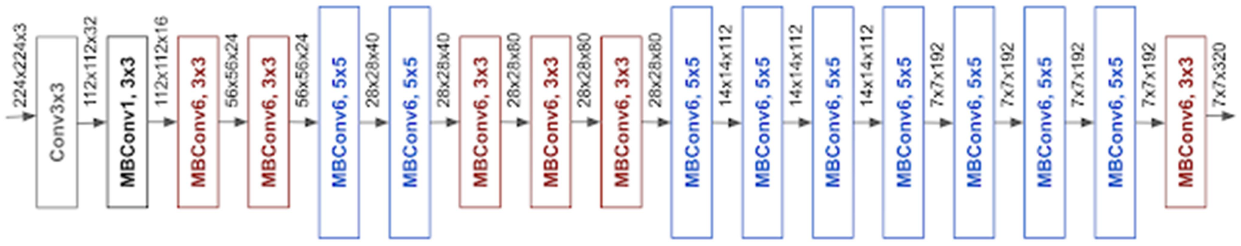


Fig 4. EfficientNet-B0, Baseline model, Source: [3]

Starting with EfficientNet-B0 as a baseline, the compound scaling method is used to scale it up in two steps:

- Fixing φ = 1 and cconsidering that there are twice as many resources accessible, and applying a small grid search of α, β, γ based on the equation below, it was discovered that, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, the optimum values for EfficientNet-B0 are α = 1.2, β = 1.1, γ = 1.15.
- Using the equation below, EfficientNet-B1 to B7 are obtained by scaleing up the baseline network with various φ, while α, β and are γ fixed.

$$\begin{aligned}
\text{depth: } & d = \alpha^{\phi} \\
\text{width: } & w = \beta^{\phi} \\
\text{resolution: } & r = \gamma^{\phi} \\
\text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\
& \alpha \geq 1, \beta \geq 1, \gamma \geq 1
\end{aligned}$$

3. Case Study/Original Contribution

3.1. Challenges

Even a great concept for a machine learning model can perform well below expectations and deliver far too little accuracy to be considered a decent model, mostly because of its complexity and limited dataset. To say the least, collecting a large volume of data before starting a new project might be difficult. Moreover, getting a decent model performance with a small amount of data for training is really difficult, if not impossible.

This was one of the first challenges I encountered when I started this project, and it quickly escalated to be one of the toughest. There is very little to no data of plastic recycling symbols available, and since plastic recycling symbols are organized into 8 categories, the amount of data needed for a model to be efficient and accurate is well above 8000 images. The fact that all plastic recycling symbols are very similar, in the sense that most of the time they differ by just the number inside the triangle or the name of the plastic below the triangle, poses another challenge for the image classification algorithm.

In order to overcome these issues presented above, the solution I chose to go with was transfer learning. Why transfer learning? As stated in the previous chapters, the concept behind it is very simple: a model can be trained with a small amount of data and yet perform well. I decided to go with two structurally distinct state-of-the-art pre-trained models that are also frequently used in the industry: Oxford's VGG19 model and Google's EfficientNet-B7 model.

3.2. Implementation of the Models

The implementation of the models is done in Python using the tensorflow.keras API. The application works with three models simultaneously: a simple model implemented by me, a VGG19 model, and an EfficientNet-B7 model from the tensorflow API. The final output for the application will be a weighted sum of the best performing models.

All three models work with the same dataset of images that is loaded into the application using Python's cv2 module (pre-built CPU-only OpenCV packages for Python) and, after all images are resized to a specific size, they are stored in a Numpy array. Numpy arrays allow us to easily store the data needed for the models and to easily feed the data to the models.

For the pre-trained models, we proceed to freeze the pre-trained layers and add our custom final layers with 8 nodes for classifying the output. However, since we are dealing with a limited quantity of training data, image augmentation is a viable option. That is, we create a collection of duplicates of existing images with slight modifications. These modifications might include things like slight rotations, zooming, horizontally flipping pictures, and so on. Naturally, we created the model to include image augmentation as a means to artificially increase the amount of training data. During model fitting, we employ an early stop callback in order to avoid overfitting our models and a checkpoint callback to save the models locally in order to load them quickly when we need them.

- Own model

Having the checkpoint for the models saved, we can quickly load them and receive a prediction for any image without having to re-train the whole model. The prediction output for every model is a vector with 8 values in the interval [0, 1]. Each value represents the probability that the image will be classified as one of the eight different types of plastic.

3.3. Prediction Results

The initial dataset is composed of around 300 images of plastic recycling symbols, but they are not evenly distributed among the 8 plastic categories, which has a negative impact on the prediction of new photos.

For the initial dataset, the EfficientNet-B7 had the best result with a validation accuracy of close to 55% (Fig 3.3.3) and a loss of 2 (Fig 3.3.4), followed by VGG19 with almost 50% (Fig 3.3.1) and a loss of 1.75 (Fig 3.3.2). But during training, VGG19's model outperformed EfficientNet-B7's model reaching over 90% accuracy, while EfficientNet-B7 peaked at 60%.

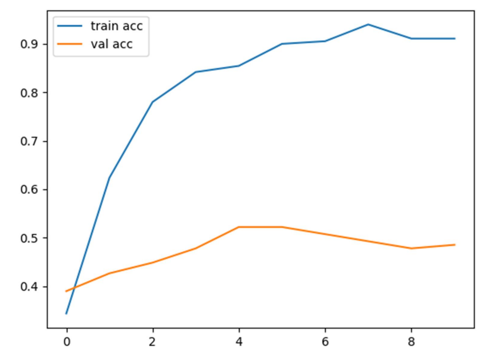


Fig 3.3.1. VGG19 Accuracy

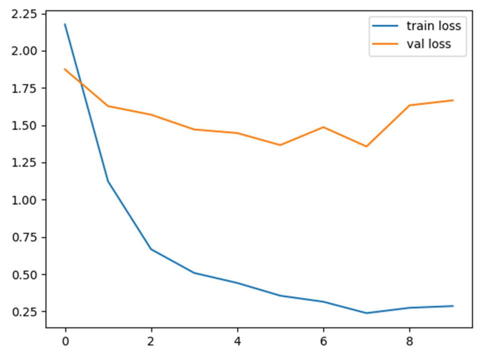


Fig 3.3.2. VGG19 Loss

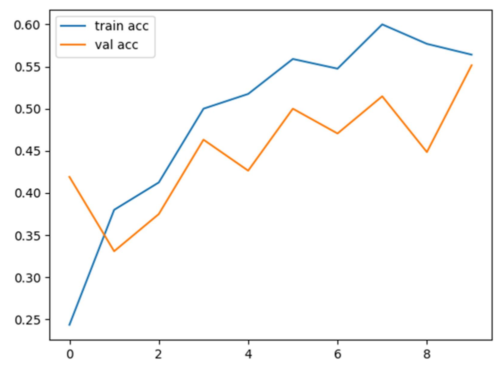


Fig 3.3.3. EfficientNet-B7 Accuracy

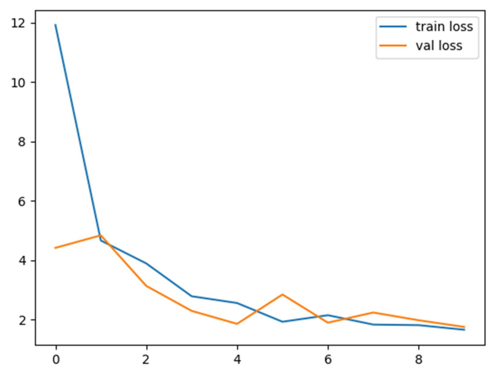


Fig 3.3.4. EfficientNet-B7 Loss

Testing multiple configurations for the small model created by me, I could not match the accuracy of the pre-trained models, achieving a maximum validation accuracy of around 33% with a loss of 1.8.

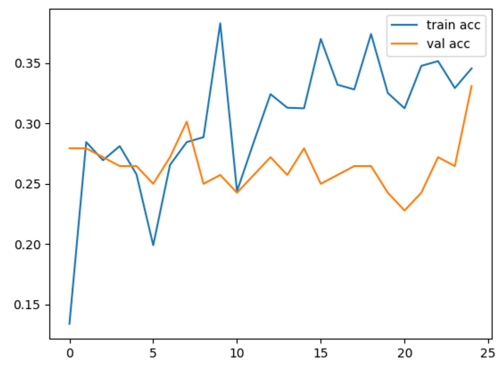


Fig 3.3.5. Initial Model Accuracy

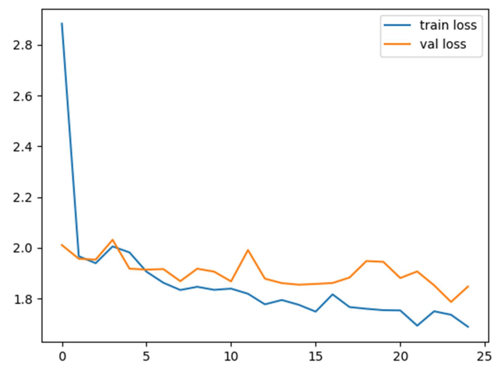


Fig 3.3.6. Initial Model Loss

- a lot TODO here

- refactoring a bit maybe?

-challenging to find data/photos for my app: solutions transfer learning + my app

-talk about transfer learning + ocr to help at the beginning

-present small model and compare result to transfer learning

-chapter about why python, tensorflow and keras

-images of code in chapter above? – data augmentation part + maybe model + k-cross validation

4. Recycle Buddy Application

One of my main goals throughout this thesis was to make information about plastic recycling symbols as easily accessible as possible because it is difficult to remember all of the meanings and processes behind plastic recycling signs and their recycling process. In order to ensure that a user will be able to access that information anytime, anywhere and in the simplest way, I decided to go with a smartphone application.

4.1. Mobile Applications

A recent study [13] shows that almost 84% of the Earth's population owns a smartphone as of 2022, a considerable difference in contrast with 2016 when not even 50% of that year's global population had a smartphone. This means that there are approximately 6.64 billion smartphones users in the world today, ensuring that the smartphone is a great choice when taking accessibility into account. Because it is such a small device, the user is very likely to carry it with them when they need information about plastic recycling symbols, making a smartphone app extremely convenient in that scenario.

But what is a smartphone application? [16] A smartphone (or mobile) application, usually referred to as an "app", is a type of software application that runs on a mobile device such tablets and smartphones. Mobile applications typically provide users with services that are comparable to those available on PCs. Usually, each app has just one limited feature, such as a mobile we browsing, calculator, or a game. Although the limited hardware resources of early mobile devices stopped apps from multitasking, their distinctiveness is now part of their attractiveness since it allows users to opt what their devices can do.

Methods for developing mobile applications are continually changing and expanding. There are several techniques to designing a mobile application, but the ideal approach will depend on the developer's needs. Based on the devices operating system, apps are divided into two broad categories: native applications and cross-platform applications [14].

The term "native app development" refers to the process of creating mobile applications that is suited to run on a single platform. Platform-specific programming languages and tools are used to develop the app. You can make native and iOS applications with Objective-C or Swift, and Android applications with Java or Kotlin, for example. Because of their fast performance, native apps are considered to provide better user experience. The aesthetics are also in line with the platform's UX, enhancing the user experience.

Cross-platform development is the process of creating an app that operates on multiple platforms. Some of the tools available for achieving this are Xamarin, React Native, and Flutter, and those applications may be used on both Android and iOS devices.

If the developer wants to release the program on all platforms, cross-platform development saves time and money, but it comes at the expense of quality. It's difficult to make an app that works efficiently across several platforms, and the software will need to run through an additional abstraction layer, resulting in slower performance.

For this project, I decided to go with a native application since it offers several advantages: since the app is designed and optimized for a certain platform, the app performs far better and at a higher level than before; native mobile apps are secure and reliable; they have better mobile hardware integration; and they inherit their devices' OS interface, making them look like an integrated part of the device, providing a better user experience and making the app more interactive and intuitive.

4.2. Android

In order to decide on which OS to launch the application, I had to study and compare the available options: Android, iOS, Windows Phone, and Nokia are the most well-known mobile OSs. The market share [15] ratios of those OSs as of April 2022 are: Android with 71.59%, iOS with 27.68% and other mobile OSs with 0.73% such as Samsung, Nokia and others (Figure 4.1.1).

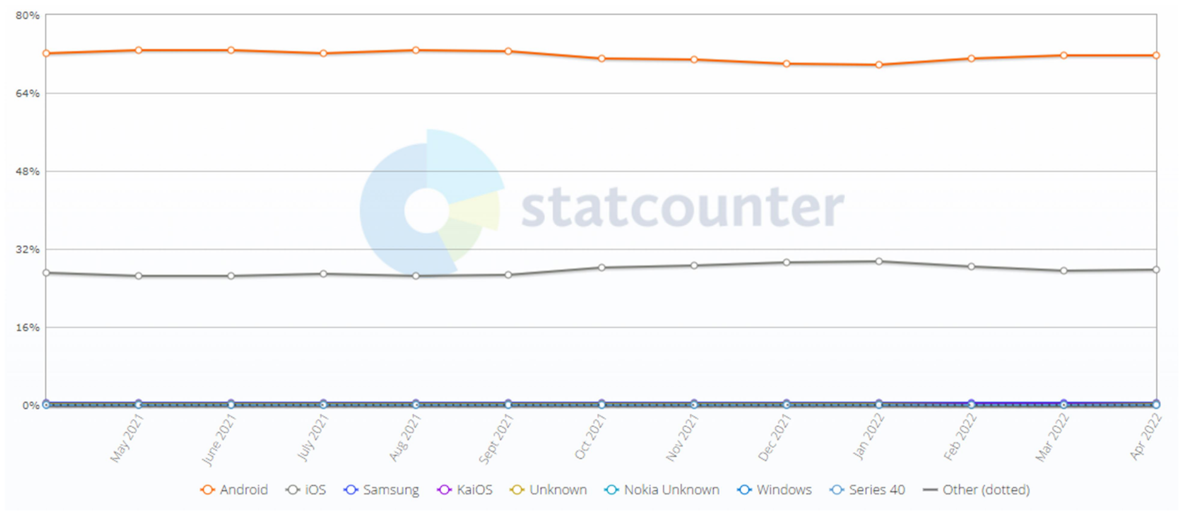


Fig. 4.1.1. Mobile Operating System Market Share Worldwide – Source [15]

For various reasons, I decided to go with Android as the operating system: as presented above, the market share worldwide is completely dominated by the Android Market guaranteeing a wider range of potential users; there are low barriers of entry to developing an Android application, meaning it can be created on Mac, Windows and Linux, unlike iOS where a designer must be using a Mac; and since Android is an open-source platform, it provides better customization features, greater flexibility when developing features, and easy availability of resources, making it easier and quicker to develop applications.

The Android operating system [17-18] was originally designed by Android Inc., which was eventually bought by Google and released in 2008. As previously stated, it is an open-source project, maintained by the Open Handset Alliance. Android is built on a customized Linux kernel that uses Linux 2.6 to provide memory management, security, process network stack, management, and a driver model. It includes a wide range of libraries that enable app developers to construct a wide range of applications. The Java programming language is frequently used to create Android apps.

4.3. Android Studio and Kotlin

The Android Software Development Kit (SDK) provides a collection of platforms, tools, sample code, and documentation for creating Android apps.

Android Studio [19-20], which is based on IntelliJ IDEA, is the official Integrated Development Environment (IDE) for Android application development. Android Studio extends IntelliJ's powerful code editor and development tools to help developers build Android applications quicker. The SDK (Software Development Kit) is a subset of the Android Studio program that contains only the tools needed to interface with a device through the command line. It contains emulators, APIs, software libraries, reference materials, and a range of other tools that help not only with the development of Android apps, but also with the legal analysis of Android devices.

Kotlin was my programming language of choice when it came to the implementation of the mobile application. Although Kotlin was initially released in 2011, its first stable release was released in February 2016. Kotlin is a modern programming language that can coexist with Java in a smooth way.

The literature shows that Kotlin is becoming increasingly popular among Android software developers. Around one-fifth of Android apps hosted on the F-Droid marketplace contain Kotlin code, with two-thirds of those projects including more Kotlin than Java code [22].

Kotlin [21] is a statically typed programming language that runs natively on the JVM (Java Virtual Machine; this is a virtual machine implementation that runs a Java application) and completely interoperates with Java: you can find in an application both Java and Kotlin, and Java code can be called from Kotlin code and vice versa. One of the main reasons driving Android developers to switch to Kotlin is this possibility of coexistence between the two languages.

A better handling of null values was one of the key design factors that led to the creation of the Kotlin programming language. NullPointerExceptions (NPE) can occur when dealing with null values in Java without the use of particular checks. NullPointerExceptions are a common cause of Android application crashes, according to several researches in the literature. According to R. Coelho [23], NPEs were the underlying cause of over 30% of all stack traces gathered after an Android app crash. The authors also point out how difficult it is to safeguard code from exceptions, particularly when the application does not allow third-party source code access.

In addition to classes, Kotlin allows functions to be first-level constructs, something which Java does not support. In Kotlin, everything is an object, including numeric values, which are primitive types in Java. Kotlin allows you to extend a class with additional functionality without having to inherit from it or utilize a design pattern like Decorator by using special declarations called Extension Functions and Extension Properties.

However, several Java features, such as checked static members, exceptions, ternary operator, and non-private fields are not available in Kotlin.

Even though, in terms of maintainability, there are no substantial differences between Java and Kotlin, I chose Kotlin since there is a considerable difference in the amount of code written, demonstrating Kotlin's superior conciseness and since the frequency of NullPointerExceptions is much lower while developing in Kotlin in terms of ease of development. On the other hand, there was no discernible change in the recurrence of other typical Java issues and Java's IDE support was rated greater to Kotlin's.

4.4. Implementation

The Recycle Buddy application's main goal is to teach its users how to recycle plastic correctly by providing information about all the plastic recycling symbols. The app allows consumers to either take photos or upload photos of a plastic recycling symbol, and using the Image Classification models and the OCR algorithm presented in the previous chapters, it will generate a response to the user's uploaded picture with one of the 8 plastic types. By giving feedback on this response, the end-user can, in turn, help the Image Classification algorithm learn from its mistakes.

The implementation of the application was done using Kotlin, for the IDE I used Android Studio, and for testing and debugging I used Android Studio's Google Pixel 2 emulator together with two physical Android phones: a Samsung Galaxy S21 Ultra and a Huawei P20 Pro.

The mobile application is structured into two separate pages: the "Homepage" and the "More Information Page"; additionally, the result generated from the server will show in a popup over the Homepage screen.

The applications graphical user-interface (GUI) was designed using the Android pre-built UI components (structured layouts and UI controls). Layouts were used to define the structures for the user interface in the application, more precise, each page, including the popup, has its own layout. A hierarchy of View and ViewGroup objects is used to create all of the layout's components. A View depicts anything that the user can view and interact with while a ViewGroup is a container that defines the layout structure of the View. A View has objects such as Buttons or TextViews or Spinner, while the ViewGroup objects are of type LinearLayout or ConstraintLayout. Android has a simple XML vocabulary that corresponds to View classes and subclasses like widgets and layouts. By declaring the UI in an XML format, the code that controls its functionality gets isolated from the design of the application. It's also simple to create multiple layouts for different screen sizes and orientations when using XML files. When the app is compiled, each XML layout file is compiled into a View.

The Homepage is implemented as an AppCompactActivity and it manages various tasks: checking for camera permissions, taking photos and choosing photos from the gallery, and interacting with the GUI. When taking a photo of the recycling symbol, the application uses the phone's main camera application since the user is already familiar with its interface, and on most devices, the pre-built camera app is very powerful and contains a lot of functionalities for capturing better images. The communication with the camera app is done through an Intent (MediaStore.ACTION_IMAGE_CAPTURE) and, in order to ensure better quality, the photo that is taken is stored temporarily on the user's device by adding some extra options to the Intent (MediaStore.EXTRA_OUTPUT) and by using a file provider. Opening the device's gallery application is also done through an Intent (Intent.ACTION_PICK). The taken or chosen photo will be displayed on the homepage and if it happens to be rotated, a function rotates it to the correct position. The interaction with the GUI is done as follows: the setContentView function loads the layout resource from your app code, in the activity. From here, the activity has access to all the properties of the views from the layout.

The popup is implemented as an AlertDialog inside the Homepage activity. Its job is to display the server's response, and to provide a means to offer feedback on that response. When the user considers that the response is wrong, a dropdown list implemented as a Spinner will appear with all plastic types where they can select what they think the correct answer is.

After the consumer receives the response from the server and generates feedback, they can press a button that will take them to a new screen, the "More Information Page". This page is also implemented as an AppCompactActivity but its only task is to display additional information about a specific plastic type. The information is updated dynamically using the same interaction between the activity and the layout.

The communication between the application and the server is done through the OkHttp HTTP client, or to be more precise, the okhttp3 package. The app and sever communicate back and forth several times. First when the application sends the image to the server (the request body is created as a MultipartBody, where the photo is stored) and the server responds with the result generated from the Image Classification models and the OCR. And second, when the app sends to the server the feedback generated by the user, using a normal FormBody that holds a couple of Strings.

The images and descriptions of the plastic types displayed to the users are stored locally within the application in order to provide a better UI experience. This also ensures that the program will run more smoothly since it will not have to download all of the data each time it is needed, and the tradeoff is minimal because they take up very little space.

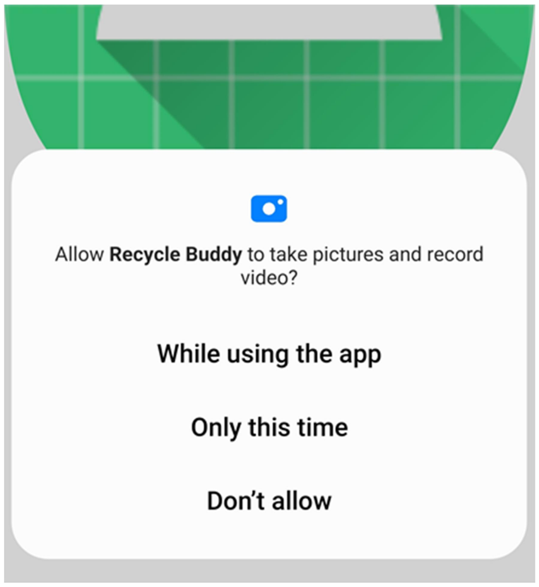
- Diagrams TODO

4.5. User Manual

Recycle Buddy is a free Android application and anyone who owns an Android smartphone (with an Android 8.0 Oreo version or newer) can access it. This subchapter will provide a guide on how to use the application to its full potential while presenting all its features.

- Write about light theme and dark theme (and would be a great idea to fix them too oops) TODO

The main purpose of the app is to help the user identify the meaning behind a plastic recycle symbol, so permissions to the camera and access the phones storage need to be granted in order to take and respectively upload images of plastic recycling symbols. A prompt asking for permissions will be shown when the app is first launched (Figure 4.3.1) and the user needs to allow the app to use the camera function and storage functions of the phone in order to utilize the application.


Allow **Recycle Buddy** to take pictures and record video?

While using the app

Only this time

Don't allow

Fig 4.3.1. Camera and Storage Permissions

Upon entering the application we have the homepage like in the Figure 4.3.2, where the user is presented with two buttons, “Take Picture” and “Open Gallery”, a placeholder image and a disabled button “Upload Picture”.

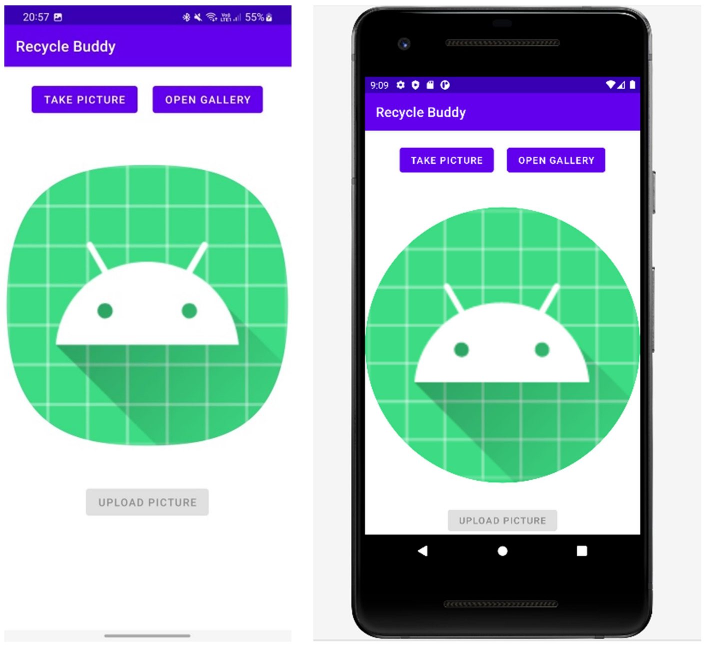


Fig 4.3.2. Recycle Buddy Homepage

If the user wishes to take a picture of a plastic recycling symbol, they should press “Take Picture” button which will take them to the phones built-in camera application, allowing them to take a picture of the symbol having all the features of the phones camera at hand (Figure 4.3.3). If the user already has an image of the recycling symbol in the phones storage, they can press the “Open Gallery” button which will open the phones storage system, allowing the user to select the image they want to upload (Figure 4.3.3).

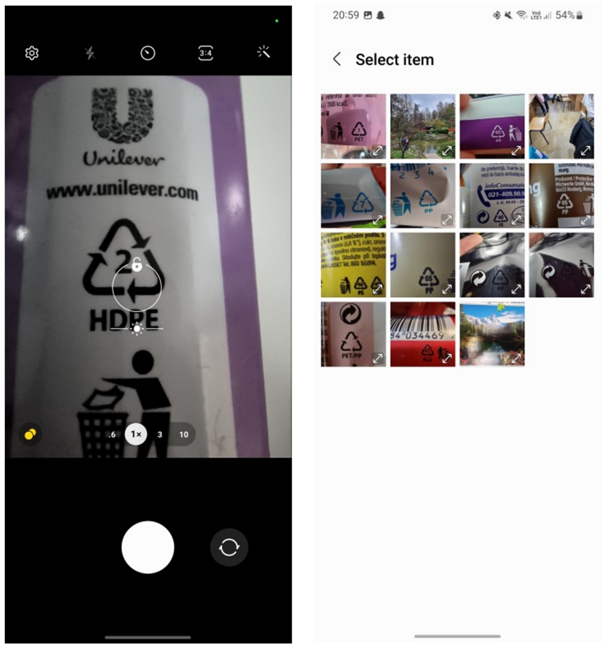


Fig 4.3.3. Camera and Gallery

After a photo was taken or selected, it will appear instead of the placeholder image at the center of the homepage like in the Figure 4.3.4. Also, the “Upload Picture” button will be enabled, allowing the user to upload that image to the server. The result may take a few moments to arrive since the server needs to run the image through several Image Classification models.

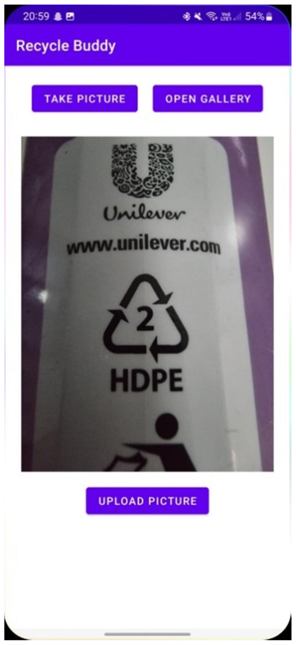


Fig 4.3.4. Homepage with selected image

Upon receiving a response from the server, a popup including information about the result will appear (Figure 4.3.5). The popup will contain a picture (to make it easier for the user to tell if that is the plastic recycling sign they are looking for) and the name of the plastic recycle symbol that the server generated as a result. Underneath them, there are two buttons: “Not Right?” and “Correct!”. When the user thinks the response is right, they should press the “Correct!” button, signaling the app that the server made a right decision. If the user sees that the response may not be right, they can press the “Not Right?” button which will make a dropdown appear with all the types of plastic listed on it; choosing the right answer from that dropdown is crucial for the application because, in this way, the Image Classification models can learn from their mistakes and provide more accurate results in the future.

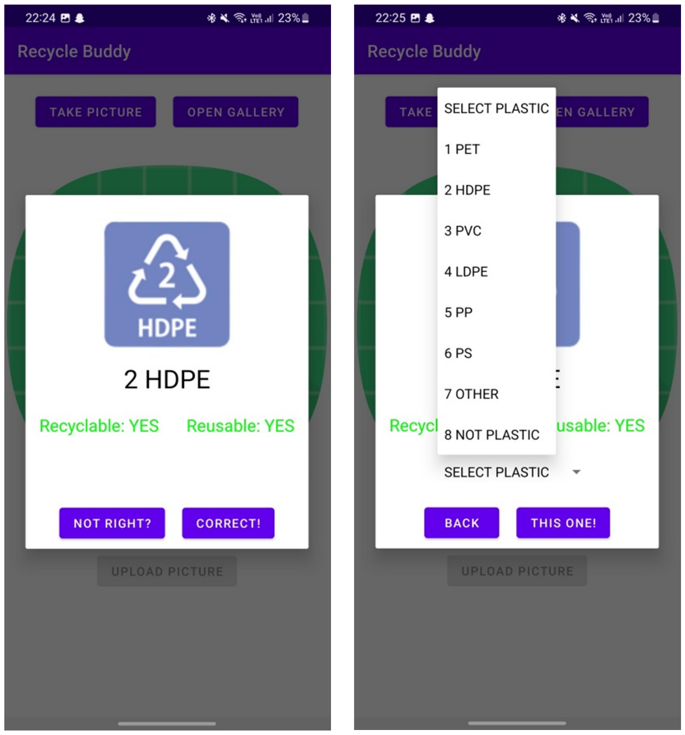
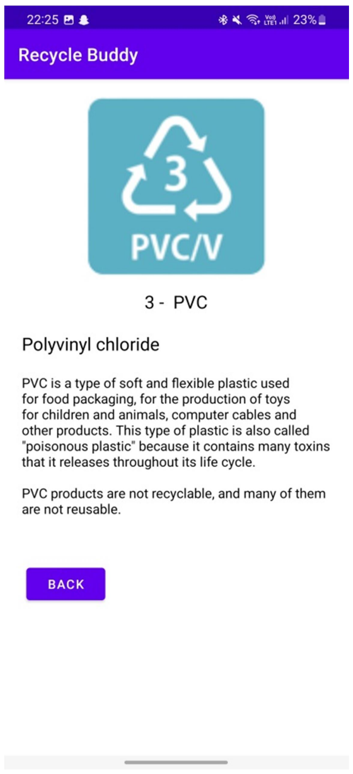


Fig 4.3.5. Popup with result from server

After closing the popup, a new button will appear – “More Info” – which when pressed will take the user to a new window that will display additional information about the plastic type that the server generated as a response, or in the case where the server made a mistake, the plastic the user selected as being the correct type in the popup dropdown. The abbreviation of the plastic will be shown near the button to ensure a more user friendly interface. (- insert figure here)

This new page (Figure 4.3.7) brings forth more information about a type of plastic such as its full scientific name, for which purpose it is mainly used and in what objects it is usually found, how it should be recycled (if it can be recycled) and if it is safe to reuse and under which circumstances. The button labeled “Back” allows the user to return to the homepage, as well as the back function of the phone.



3 - PVC

Polyvinyl chloride

PVC is a type of soft and flexible plastic used for food packaging, for the production of toys for children and animals, computer cables and other products. This type of plastic is also called "poisonous plastic" because it contains many toxins that it releases throughout its life cycle.

PVC products are not recyclable, and many of them are not reusable.

Fig. 4.3.7. Additional information page

5. Conclusions

Bibliography

[1] Boukaye Boubacar Traoré, Bernard Kamsu-Foguem, Fana Tangara - Deep convolution neural network for image recognition, Ecological Informatics, 2018 – Elsevier (2019)

[2] Srikanth Tammina - Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images, International Journal of Scientific and Research Publications (2019)

[3] Mingxing Tan, Quoc V. Le - EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks - International Conference on Machine Learning, 2019 (2019)

[4] Gaudenz Boesch - A Complete Guide to Image Classification in 2022, https://viso.ai/computer-vision/image-classification/

[5] purva91 - Top 4 Pre-Trained Models for Image Classification with Python Code, https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

[6] Pragati Baheti - A Newbie-Friendly Guide to Transfer Learning, https://www.v7labs.com/blog/transfer-learning-guide

[7] Derrick Mwiti - Transfer Learning Guide: A Practical Tutorial With Examples for Images and Text in Keras, https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

[8] What is computer vision?, https://www.ibm.com/topics/computer-vision

[9] Karen Simonyan, Andrew Zisserman - Very Deep Convolution Networks for Large-Scale Image Recognition (2015)

[10] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton - ImageNet Classification with Deep Convolutional Neural Networks - Part of Advances in Neural Information Processing Systems 25 (2012)

[11] Matthew D. Zailer, Fergus Rob - Visualizing and understanding convolutional networks (2014)

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun - OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks (2014)

[13] Ash Turner - How Many Smartphones are in the World?, https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[14] Anastasiya Marchuk - Native Vs Cross-Platform Development: Pros & Cons Revealed, https://www.uptech.team/blog/native-vs-cross-platform-app-development

[15] Mobile Operating System Market Share Worldwide, https://gs.statcounter.com/os-market-share/mobile/worldwide

[16] Mobile Application (Mobile App), https://www.techopedia.com/definition/2953/mobile-application-mobile-app

[17] T. Hared, R. Dara, S. C. Kremer - Computer and Information Security Handbook (Third Edition), Chapter 6 - Intrusion Detection in Contemporary Environments (2017)

[18] V. Garousi, R. Kotchorek, M. Smith – Advances in Computers, Chapter 5 - Test Cost-Effectiveness and Defect Density: A Case Study on the Android Platform (2013)

[19] R. Tamma, D. Tindall and O. Skulkin, Learning Android Forensics – Second Edition (2018)

[20] Android Studio – Meet Android Studio, https://developer.android.com/studio/intro

[21] L. Ardito, R. Coppola, G. Malnati, M. Torchiano - Effectiveness of Kotlin vs. Java in Android App Development Tasks, Information and Software Technology (2022)

[22] R. Coppola, L. Ardito, M. Torchiano, Characterizing the Transition to Kotlin of Android Apps: a Study on F-Droid, Play Store, and Github, Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics (2019)

[23] R. Coelho, L. Almeida, G. Gousios, A. V. Deursen, C. Treude - Exceptionhandling bug hazards in Android, Empirical Software Engineering 22 (2017)