Banana state estimation using Convolutional Neural Networks for waste reduction

Agostino Sorbo
agostino.sorbo@gmail.com

Supervisor: Bob Borsboom

7 July 2023

Abstract. This paper explores the application of Convolutional Neural Networks to the task of perishable food state classification. Fresh fruit and vegetables contribute to almost half of household food waste. Furthermore, perishable food spoilage and quality directly impact the processed perishable food industry. Currently, the state of fruits and vegetables is mostly done by visual assessment, both domestically and in industrial settings. Introducing an automated system able to classify perishable foods according to their state may help reduce food waste. This can be done domestically by integrating such solutions in smart home appliances such as fridges and pantries and in industrial settings by enabling automated, highly accurate quality control. Current research mainly focuses on the binary classification of perishable foods as either fresh or rotten. However, to allow perishable foods to be consumed or processed before they reach the rotting stage and avoid food waste, it may be helpful to move beyond this binary classification. This thesis aims to show how Convolutional Neural Networks can be used to perform an external analysis of perishable foods to estimate their state. In particular, this paper illustrates how to train multi-class, single-label classifiers for perishable food state classification. To do so a dataset is adapted for the task. Following, two approaches are experimented with: developing a custom, shallow CNN architecture, randomly initialized, and training a Deep Residual CNN leveraging transfer learning. For both architectures, several models were trained to perform a grid search and select the best hyperparameters. As suggested by previous research, models trained with the ResNet50 architecture were successful and especially well-performing in the task of classifying bananas as either: unripe, ripe, overripe, or rotten. The procedure shown in this paper may be replicated and applied to other perishable foods to further limit perishable food waste both for domestic and industrial applications.

Keywords: Convolutional Neural Network · Perishable goods freshness · Food-waste reduction · Image classification · Residual Network · Transfer learning.

1 Introduction

1.1 The issue

Almost 59 million tonnes of food waste amounting to 131kg per person are generated yearly in the EU according to the Eurostat report on food waste released in 2022 [1]. This amounts to roughly 10% of the food in retail, food service, and households in the Union. When observing global data this amount grows to 14% of all the produced food which is lost after the harvest stage, before reaching the retail stage [2]. In the EU 53% of the total food waste is generated by households and an additional 15% is generated by food service and retail combined [1]. [3] suggests that fresh fruit and vegetables contribute to almost 50% of food wasted by households. A behavior analysis conducted on Polish customers reports that the vast majority of respondents attributed the cause of the waste of the food due to it being spoiled or past the due date [4].
Food waste has a strongly negative impact on the environment causing an estimated 7% of the overall EU Greenhouse Gas emission [1]. Furthermore, when narrowing the scope to fruits, spoilage is estimated to account for almost a third of their cost [5]. Fruit quality directly impacts the food processing industry [6] as unripe or rotten fruits may disrupt the quality of whole batches of processed fruit products (e.g. fruit preserves, canned fruit, fruit juices, and syrups) [7]. Degraded fruits may also have a negative effect on people's health as a result of nutrient loss [8] and of microbial growth that occurs as the fruit spoils [7]. Currently, the ripeness and eventual signs of decay are often manually assessed based on fruit appearance. This approach, however, is subjective, time-consuming, and expensive (in industrial settings) [8]. In this thesis, the aim is to create a model to estimate the ripeness and decay stage of fruits, specifically bananas. The primary motivation for this is to reduce domestic and industrial perishable food waste.

1.2 Motivation and scope of the study

With the goal to reduce food waste, it may be useful to go beyond this binary classification to allow products to be consumed before they reach the overripe or rotten stage.
This thesis aims to show how Convolutional Neural Networks (CNNs) can be used to perform an external analysis of bananas to estimate their state. In particular, the goal is to describe how to train a multi-class, single-label classifier for perishable food.

Such a model exists and can be found on the internet. However, no information regarding the authors or how the research was carried out is available [9]. In this paper, a CNN model is trained to classify images of bananas as one of the following classes: unripe, ripe, overripe, and rotten.

2 Current state of research

Recent studies have applied Artificial Intelligence techniques to classify fruits as either fresh or rotten. In particular, Convolutional Neural Networks (CNN) have been employed and were found to be a viable option. The authors of [6] compared the performance of several CNN architectures in industrial settings (namely to identify and exclude rotten fruits on a conveyor belt prior to food processing). In particular, this study focused on the application of transfer learning to CNNs to classify three different types of fruits according to their freshness. The study experimented with AlexNet, ResNet50, and VGG-16 architectures. All architectures were found to be well-performing, however, ResNet50 was found to outperform the other architectures. This was imputed to the ability of residual networks of reducing the vanishing gradient as well as the exploding gradient problems even with a considerably elevated number of layers. The authors noted that the added performance of the ResNet architecture did come with the price of substantially higher complexity. Finally, the authors of [6] also concluded that, although this was not done in their experimentation, models could be trained to classify different levels of "rottenness". Such a conclusion suggests that the present study may produce models able to successfully classify fruits on a scale according to their state. Studies on perishable food state classification were also conducted beyond the industrial target. With domestic use in mind, [10] focused on the development of smart home appliances (smart fridges) to assess the presence and state (i.e. fresh or rotten) of foods [10]. In this paper, the authors propose a more general model trained to identify, count, categorize, and finally assess the state of products in a fridge. Differently from the previously discussed paper, the researchers applied a machine learning technique known as Compositional Attention Networks [11]. This approach focuses on an iterative reasoning process that enables interpretability. The resulting model successfully recognized, counted, and classified generated images of several products including perishable items.
As reviewed later in this section, the results of [6] suggest that transfer learning applies to the task of perishable food classification. In particular, deep residual convolutional networks such as ResNet50 may be a well-performing architecture for the binary as well as multi-class freshness classification of perishable foods.

2.1 Transfer learning

Transfer learning is a machine learning (ML) technique in which knowledge learned from training on a particular task is transferred to improve the performances of models trained on a different task [12]. The idea behind this is that a ML model can learn a new task by transferring abstract knowledge learned while training on a different but similar task. For example, a model trained to classify bananas according to their state (freshness/rottenness) may be used as a starting point to train a model that classifies apples according to their state. During the training of such a new model, knowledge relative to the state of bananas may be transferred onto the apple domain (e.g. knowledge regarding

browning indicating bruising or ripening and black/gray patches indicating rotting or molding may transfer from one task to the other). Such practice may considerably boost performance and shorten training times as the training of new models may leverage knowledge already acquired on other tasks. Moreover, machine learning often requires large amounts of data for training. Such large datasets are often not available for the task at hand. Using transfer learning, very capable models trained on large, high-quality datasets can be trained on new tasks for which only limited training data is available [12].

2.2 Deep residual learning

Often, the deeper a network is, the more powerful models it can generate, increasing its performance and its ability to generalize. However, prior to the publishing of [13] the maximum depth of a network was severely constrained. Then, some of the best-performing architectures were VGG-19 and AlexNet. Deep networks would incur issues with the training error that after an initial reduction would begin to climb once again. This was proven not to be due to overfitting [13]. Furthermore, the phenomenon would worsen with the addition of extra layers [14]. In [13], it is claimed that if there exists a shallow network capable of being trained for a particular function, there must be a deeper architecture that can learn the same function. Such architecture can be obtained by simply copying the layers of the shallower one and having the remaining layers to learn the identity function. For this reason, it should be possible to train the deeper network to achieve at least the same accuracy. Such a finding entails that the issues with the training of deep networks were due to the ease of optimization and not overfitting. For a network to learn the identity function is fairly computationally taxing. The authors of [13] noted this and experimented with the initialization of such networks: Instead of a random initialization to learn the identity function, networks would be initialized with the identity function itself. Layers would thus be taught to reproduce the identity function while introducing only minor variations. Such variations are referred to as *residuals*.
In residual learning, direct connections are made between a series of conventional weighted layers. Such connections do not attempt to learn the function of such weighted layers but seek to learn the difference the weight layers add to their inputs. The default function of such a so-called *residual block* would then be the identity function. When trained, this layer will learn how to deviate from the identity function to obtain a certain change. Chaining a sequence of such blocks interspersed by ReLU functions (to render it non-linear) constitutes a residual network.

3 Methods

3.1 Convolutional neural networks and their application to image data

Convolutional Neural Networks (CNNs) belong to the area of deep learning that specializes in pattern recognition. As in every artificial neural network, CNNs

are composed of a series of layers. Each layer takes some inputs, transforms them, and passes its outputs to the next layer. Convolutional neural networks, or CNNs, are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs [15]. These pattern recognition abilities arise from some particular layers: the convolutional layers. Convolutional layers require three components: input data, a filter (to transform the input), and a feature map (also known as activation map or convolved feature) as the output [15]. Filters (also known as kernels) can be seen as *windows* that take into consideration subsections of the input image. Every filter has a feature detector: a two-dimensional array of weights that represent a part of the image. Iteratively, a filter will scan portions of the input trying to detect a specific feature, then shift this window by a certain amount of pixels (called stride). At every iteration, the dot product of the filter and the current input pixels is computed and inserted in the feature map. When the kernel has considered the whole image, the feature map containing the result of the convolution is complete. Finally, an activation function called Rectified Linear Unit (ReLU) is applied to the feature map to introduce non-linearity. The introduction of the ReLU transformation helps to reduce the vanishing gradient and exploding gradient issues that may arise during the backpropagation phase of the training [16][13]. In particular, these issues become progressively more serious the deeper (meaning having many layers) a network is. Usually, the convolutional layers are followed by pooling layers: special layers used for dimensionality reduction [15]. Similarly to convolutional layers, pooling layers run a filter through the input to produce an output. However, such filters do not have a feature detector but an aggregation function instead. The two most common pooling strategies are max pooling (which outputs the value of the pixel with the highest value in its field of view) and average pooling (which outputs the average of the pixels in its field of view) [15]. The pooling layer causes a great deal of information loss. However, this loss often carries the benefits of improved efficiency and reduced complexity and overfitting risk. Usually, at the end of a CNN, one or more fully connected (FC) layers are put added. This is done to use the results of the previous layer to finally perform the classification task. As suggested by its name, a fully connected layer directly connects each node from the input layer to each node in the output. The number of outputs in the last FC layer is set to be equal to the number of classes. Alternatively, global average pooling may be used with the same goal with several added benefits including a reduced risk of overfitting [17]. As a final step, the outputs of a such layers are usually passed through a SoftMax activation function. This function ultimately produces a probability distribution for the classes thus assigning to each input the probability of it belonging to the given classes.

3.2 Data preparation

Several datasets are available for the binary classification of fruits and vegetables [18][19][20]. However, as previously mentioned, this paper aims to perform a multi-class, single-label classification. For this task, a dataset was found on the

roboflow platform [9]. This dataset was created for object detection and multi-label classification of bananas. The data is split into three parts: 16k training images, 1.5k validation images, and 760 test images thus respectively an 87% / 8% / 4% ratio. Each data point has one or more labels out of the following: fresh-ripe, fresh-unripe, unripe, ripe, overripe, and rotten. The images were subject to some minor pre-processing: Auto-orient was used to display the images in the correct orientation according to the metadata on camera positioning (landscape or portrait). Stretching is used to bring all images to a 1:1 ratio with a 416x416 px resolution. Several data augmentation techniques are employed to increase the dataset size by three times: Horizontal and vertical flipping, random cropping (between 0% and 20% zoom levels), brightness adjustment (between -25% and 25%), random noise (up to 5% of the pixels), random box cutouts (4 boxes collectively covering 15% of the pixels). Several manipulations are operated to adapt this database to a single-label classification task. First of all, the three splits are unified. Upon inspection, some classes appear to be equivalent. Namely, the fresh-ripe and fresh-unripe classes are respectively equivalent to the ripe and unripe classes. For this reason, these classes are merged together. Subsequently, all images with more than one label are removed. There is now a significant class imbalance. As suggested in [21] no further augmentation was performed to balance the classes as excessive augmentation can cause underfitting or overfitting. To fix the imbalance, the ripe class is downsampled and the overripe and rotten classes are upsampled. This results in the final dataset having 3177 images per class. Finally, the dataset is split with a 70% / 15% / 15%

Fig. 1: Extract of six images from the test split

ratio yielding ∼9k images for the training set, ∼2k images for the validation set, and ∼2k images for the test set. This ratio was initially chosen according to the Pareto law [22]: 80% for training and 20% for validation and testing. However,

as suggested in [23] a greater size of the validation set can be useful to reduce noise, which is especially convenient for hyperparameter optimization tasks. The sizes of the validation and test set are chosen to be equal, however, it may not be necessary to destine such a large portion of the dataset to the test split. The reason for first unifying and then performing the split once again is to ensure that the validation and test sets are representative of the data. To achieve this and reduce noise in the splits as much as possible, random sampling was used.

NB: During the manipulation of the dataset, the three splits of the original dataset are unified, manipulations are carried out, and finally the dataset is partitioned once again using random sampling. In this process, we can expect to find a fairly homogenous distribution of original images, their augmentations, and their copies resulting from up-sampling across all three splits (training, validation, and test). In this paper, it was decided to include augmented data in the validation and test splits in line with other studies on similar tasks [24]. However, it may be argued that including augmentations of the same image in both training and validation/test splits may introduce false correlations that could result in poor generalization to *real-world* data. To assess if these unwanted effects are present further experiments may be conducted.

3.3 Hyper-parameters Optimization

Several parameters affect the performance of the trained classifiers. To determine the optimal configuration a grid search was conducted. Following are the optimized parameters and the relative values: Architecture (ResNet50 / Custom-CNN), Batch Size (5 / 10 / 20), and Learning rate (0.001 / 0.0005 / 0.0001). The image resolution was reduced to 64x64 px to reduce the training times during the grid search. However, increased resolution is generally associated with better CNN performances across different fields of application [25][26][27]. It should also be noted that increasing the resolution often requires a reduction in batch size [26][27].

The amount of epochs also has a considerable impact on performance. This value is not fixed as it was found to be strongly dependent on the other parameters. Furthermore, the number of epochs in the grid search is excluded as this would result in severely elongated optimization times. However, early stopping is implemented to select the best epoch number for any given hyper-parameter configuration. Early stopping was found to be an effective means to halt the training at the best epoch [28]. The best epoch was chosen to be the one at which the model performance on the validation set stops increasing before the model overfits the training data **Fig. 2**. To obtain this, the implemented early stopping halts the training if performances (measured in terms of accuracy)do not increase by at least 1% (cumulative) in the last 3 epochs. During training a copy of the model is saved at each epoch. When the training is halted the best performing model in the training history is chosen.

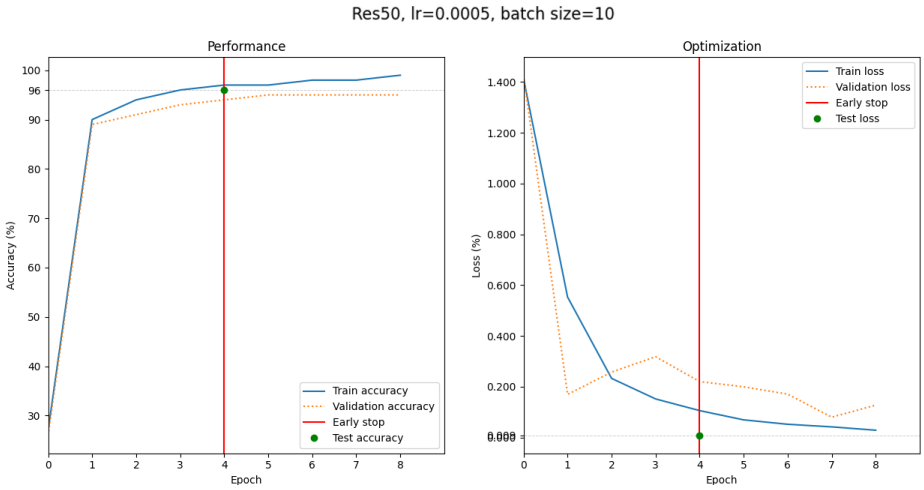


Fig. 2: Training history of the best-performing ResNet50 model

3.4 Cifar10 and CustomCNN

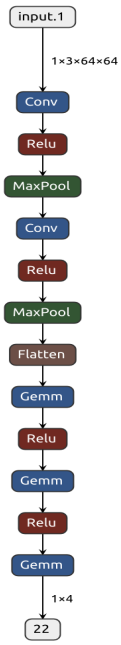


Fig. 3: Illustration of the CustomCNN architechture

As previously mentioned, the vast majority of prior research on the topic of perishable food classification focused on binary classification. However, with the goal of classifying bananas as either unripe, ripe, overripe, or rotten, a multi-class classifier needs to be built. To explore image classification in multiple classes, the PyTorch guide [29] on how to train such classifiers on the Cifar10 dataset [30] was followed. This guide instructs on the use of PyTorch to load the Cifar10 dataset, define a CNN architecture, define a loss function, and finally train and test a model. The first step in adapting this code base to the task of perishable food classification is to swap the Cifar10 dataset with the one obtained in the data-preparation phase. To enable the loading of a custom dataset, a custom data loader was implemented according to the relative official PyTorch guide [31]. Finally, the last fully connected layer was modified to have its output size match the number of desired classes (4). The obtained custom CNN architecture, inspired by the guide on CNNs over the Cifar10 dataset [29] has a total of 7 layers and over 500k parameters [32].

3.5 ResNet50 and its application to perishable food state classification

To increase the affordability of training deep residual networks, the authors of [13] introduced *bottleneck blocks* to reduce computational costs. Such layers save computations by *compressing*

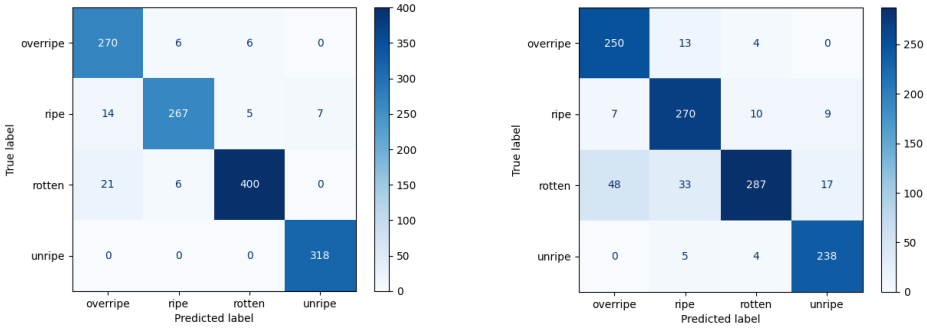
the feature map, applying the convolution, and finally restoring the feature map to its original size [13]. By combining residual and bottleneck layers the VGG-19 architecture was improved and deepened to obtain the ResNet50 architecture (see Appendix J). Recently, the authors of [6] experimented with the application of classical CNNs and residual networks to the task of perishable food state classification. More specifically, [6] focused on finding the best architecture for the binary classification of three types of fruit as either fresh or rotten. The researchers found ResNet50, AlexNet, and VGG-16 to be viable options for such classification tasks. When comparing these models, the ones based on the ResNet50 architecture performed the best. In this paper, a model is trained using the ResNet50 architecture. Given the contained dataset size, transfer learning is used to improve performances and reduce the time needed to train such a large network. The network is thus initialized with weights pre-trained on Imagenet, a large, high-quality dataset [13]. Analogously to the aforementioned Custom CNN architecture, the last average pool layer was modified to have its output size match the number of desired classes (4). The final Residual Network has a total of 50 layers and over 23,9 million parameters [33]. Finally, although the use of recurrent blocks is associated with some regularization effects [13], occasional spikes in the validation loss could be observed. This could be due to noise in the validation split. To reduce these unwanted effects L2 regularization was applied (with weight decay=0.003) [**?**].

4 Results

	Learning rate	0.001	0.0005	0.0001	0.001	0.0005	0.0001	0.001	0.0005	0.0001
	Batch size	5	5	5	10	10	10	20	20	20
	Architecture	**CustomCNN**								
Class accuracy	**Unripe**	94.6%	96.3%	96.5%	91.1%	95.9%	96.1%	97.4%	94.6%	97.4%
Class accuracy	**Ripe**	89.5%	80.5%	93.0%	97.5%	98.2%	93.6%	83.6%	96.1%	86.3%
Class accuracy	**Overripe**	89.6%	76.5%	88.6%	89.6%	91.9%	89.4%	61.4%	82.6%	82.6%
Class accuracy	**Rotten**	84.0%	91.2%	68.5%	60.3%	54.9%	59.7%	81.9%	62.8%	59.9%
	Accuracy	89%	86%	86%	84%	85%	84%	81%	83%	81%
	Class accuracy range	10.6%	18.8%	28.0%	37.3%	43.2%	36.4%	36.0%	33.3%	37.5%
	Loss	0.0002	0.00005	0.0001	0.0659	0.0316	0.1543	0.1919	0.1378	0.2129
	Epochs	6	6	7	5	7	23	5	8	27
						Res50				
Class accuracy	**Unripe**	97.8%	97.6%	97.8%	95.9%	97.4%	96.1%	98.0%	97.8%	98.0%
Class accuracy	**Ripe**	95.5%	98.0%	95.1%	95.9%	97.7%	92.8%	95.7%	97.1%	95.7%
Class accuracy	**Overripe**	96.6%	97.5%	93.2%	95.3%	95.6%	93.0%	95.6%	94.5%	95.6%
Class accuracy	**Rotten**	94.0%	90.7%	90.1%	94.0%	93.4%	87.7%	92.6%	91.2%	92.6%
	Accuracy	95%	95%	94%	95%	96%	92%	95%	95%	95%
	Class accuracy range	3.8%	7.2%	7.7%	1.9%	4.3%	8.4%	5.5%	6.7%	5.5%
	Loss	0.0014	0.0045	0.0321	0.0024	0.0086	0.1751	0.0667	0.0523	0.0667
	Epochs	4	5	8	3	4	9	4	5	4

Table 1: Results of the grid-search optimization

The performed grid search found the best combination of hyperparameters for this classification task among the proposed ones: Learning rate: 0.0005, Batch size: 10, Architecture: ResNet50. The optimal configuration of the Learning rate and batch size was found to differ among the custom CNN architecture and the ResNet50. As expected, the number of epochs needed to reach the optimal test accuracy on equal terms (parameter setting) is lower for the much deeper ResNet50 architecture. This is dramatically accentuated by strongly reduced learning rates (e.g. 0.0001) for which the difference in epochs truly becomes apparent. Under all configurations, the ResNet50 architecture seems to outperform the shallower CustomCNN. In particular, the accuracy of all ResNet50 models is considerably higher than the ones of the Custom CNN models. With the optimized parameters the custom CNN stops at 89% accuracy while the ResNet50 reaches 95% accuracy. Furthermore, the accuracy of ResNet50 models was found to be rather consistent across classes. In contrast, The custom CNN models display significant dishomogenous accuracy across classes. , This shallower architecture appears to perform particularly poorly in classifying rotten bananas. Also, the ResNet50-based models struggle the most in identifying rotten bananas. This becomes even more noticeable when using sub-optimal hyperparameter configurations. The best-performing models of both the ResNet50 and



(a) Best ResNet50 model

(b) Best CustomCNN model

Fig. 4: Confusion matrices for the classification on the test set

the custom CNN architectures (highlighted in blue in **Table 1**) were employed to classify the images of the test set. The best models were chosen according to their accuracy and accuracy homogeneity across classes, namely: [lr:0.0005, batch size: 10, architecture: ResNet50] and [lr:0.001, batch size: 5, architecture: CustomCNN]. The accuracy homogeneity across classes is measured in terms of the range of class accuracies. A tighter range indicates higher homogeneity (lower values are preferrable). The best-performing ResNet50 model made the majority of misclassifications in the rotten and ripe classes **Fig. 4a**. Specifically, 21 images of rotten bananas **Appendix F** were misclassified as overripe, and 14 images of ripe bananas as overripe **Appendix G**. As mentioned previously, the

best-performing model using the custom CNN architecture made the majority of misclassifications in the rotten class **Fig. 4b**. In particular, 48 images were misclassified as overripe instead of rotten **Appendix H** and 33 as ripe instead of rotten **Appendix I**.

5 Discussion

The best model was selected according to the overall accuracy and the accuracy homogeneity among different classes. However, no general formula for choosing the best-performing model can be given. In industrial applications, the best performance is highly dependent on the specific circumstances and the model must be tailored to the particular requirements. For example, a jam-producing factory may prefer a model with low homogeneity among class precision but with a very high recall for rotten fruits to ensure that the final product is not compromised. The optimal hyperparameter values found via the grid search for the CustomCNN are on the extreme ends of the given ranges. Furthermore, only one iteration of the grid search was performed. This may entail that further optimization is achievable. Reaching the maximum optimization is beyond the scope of this thesis. However, when training similar models for production settings it may be advisable to perform multiple iterations of the grid search. A viable approach may be to iteratively select some values around the currently most optimal hyperparameters and perform progressively finer grounded searches. This process may be halted when the performance increase or the parameter variations become negligible. The cause at the root of the best ResNet model's misclassifications of the 21 images of the rotten class **Appendix F** may be ascribed to the resolution of the images and zoom levels. When visually examining them with the same resolution fed to the model (64x64 px) the correct class membership is not evident: Some bananas display dubious coloring but no clear signs of rotting while, in others, the zoom level is so strong that a considerable portion of the banana does not fit in the view. Observing the visible part of the banana it is unclear what the correct labeling would be. Observing the 14 images of ripe bananas classified as overripe **Appendix G** two images and their augmentations are found. In one of the images, a ripe banana is laid next to a large, dark orange object. This object, its shape, and its color may confuse the classifier and yield an *overripe* label. For the other image and its augmentations, no clear cause of the misclassification can be identified. The causes of the best CustomCNN-based model's misclassifications of the 48 images **Appendix H** may be diverse. For some of them, an argument analogous to the one made with the ResNet50 misclassifications could be made. In fact, some of the images observed with 64x64 px resolution could be misclassified even by a human annotator. In other cases, the rotting is not in an advanced stage thus the model may identify bananas as overripe instead of rotten. Observing the 33 images labeled as rotten but classified as ripe **Appendix I** a common denominator seems to emerge. All the misclassified images show bananas with clear signs of advanced decay although circumscribed to only a reduced area. This

may have led the classifier to lessen their importance and judge based on the majority of the area of the bananas. Additionally, it can be noted that both best models tend to misclassify an image and its relative augmentations and eventual up-sampled points. In this paper accuracy is the main metric used to evaluate the models' performances. However, this is only suitable for balanced datasets. When formulating this study it was chosen to balance the dataset and employ accuracy to enable the comparison with previous research on binary classification of perishable foods. However, replicating this study on different perishable foods, balancing may be inconvenient or even unfeasible. In such cases, other metrics such as precision, recall, and f1 scores could be used instead.

6 Conclusion

This thesis showed how to employ Convolutional Neural Networks (CNNs) for the external analysis of bananas with the goal of reducing perishable food waste. More specifically, it was shown how to pre-process a dataset of perishable food images, how to use it to train a multi-class, single-label CNN classifier, and finally how to optimize the model hyperparameters. In line with what was suggested in previous research, the ResNet50 architecture was found to be a viable option for the task of state classification of perishable foods. Moreover, CNN-based models were found capable of being extended further than binary classification tasks. As a matter of fact, two models were proposed to classify bananas as either *unripe, ripe, overripe, or rotten*. The ResNet50 architecture was compared with a custom build, shallower architecture. In terms of performance, the ResNet50 architecture consistently outperformed the shallower custom CNN architecture. The accuracy of all ResNet50 models was considerably higher than that of the custom CNN models. When using the optimized parameters, the custom CNN achieved an accuracy of 89%, whereas the ResNet50 reached an accuracy of 96%. Moreover, the accuracy of the ResNet50 models demonstrated a high level of consistency across all classes, in contrast with the custom CNN models exhibiting significant variations in accuracy among different classes. Nevertheless, the best model trained using the Custom CNN architecture still exhibits decent accuracy. This is particularly notable when taking into account the considerably lower complexity in comparison with the ResNet50 architecture. Finally, it was shown how to examine misclassification for the trained models, and a guideline on how to select the best model was discussed. In conclusion, this thesis proposed a viable solution applicable to reducing perishable food waste. Limitations and future work are discussed in the *reflections* section following.

7 Reflections

7.1 Limitations

Reflecting on the performed study several limitations were found. Randomness may be affecting results: Each experiment could be reiterated multiple times

and the results averaged, in particular for Custom CNN as the initial weights are randomly initialized.
Further optimization is likely possible: In particular, for the best custom CNN model, all the selected hyperparameters were found to be on the extremes of the proposed range of values. The suggested grid search should be reiterated and progressively finer-grained around the selected values.
The obtained model is very specific to the dataset: Previous studies on CNNs [6] suggest that several factors, including the background of the images, impact performances. For this reason, models trained on a specific dataset may not perform well on samples from different datasets.
An external analysis may yield inaccurate estimates. Life sciences researchers suggest that external analysis of perishable foods may not always accurately reflect the objective quality and freshness of the food

7.2 Future work

To account for randomness inherent to the stochastic nature of the proposed models, the grid search experiments may be run multiple times and their results averaged. This would help reduce the stochastic element and improve the results' reliability.
The approach proposed in this thesis was found to be successful for the given dataset. However, the process of creating a dataset, processing the data, and selecting the optimal model can be tedious and time-consuming. Such complexity may restrict the use of such solutions to larger companies that can afford to invest in the development of such customized models. To enable smaller companies to reduce their perishable food waste, future work may include the development of a robust, more generic model. Such a generic model could then be used as a transfer learning base for similar tasks on different perishables. Constructing a more varied dataset and experimenting with techniques (e.g. background reduction) may yield a more general model. Such models would be more affordable for smaller companies that could introduce similar systems as an aid to human operators.

7.3 Replicability

As the motivation at the base of this study is to reduce food waste, particular focus was put on making it easy to replicate. This was done to enable others to train similar models on different perishable foods and facilitate further research on this task. The codebase necessary to train similar models is thus made publicly available on GitHub at github.com/asorbo/BananaComputerVision.

References

1. Eurostat, "Food waste and food waste prevention - estimates." [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Food$_{w}aste_{a}nd_{f}ood_{w}aste_{p}revention_{-}estimates$
2. J. Gustavsson, Ed., *Global Food Losses and Food Waste: Extent, Causes and Prevention; Study Conducted for the International Congress Save Food! At Interpack 2011, [16 - 17 May], Düsseldorf, Germany.* Food and Agriculture Organization of the United Nations.
3. V. De Laurentiis, S. Corrado, and S. Sala, "Quantifying household waste of fresh fruit and vegetables in the EU," vol. 77, pp. 238–251. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0956053X18301946
4. B. Bilska, M. Tomaszewska, and D. Kołożyn-Krajewska, "Analysis of the Behaviors of Polish Consumers in Relation to Food Waste," vol. 12, no. 1, p. 304. [Online]. Available: https://www.mdpi.com/2071-1050/12/1/304
5. Y. Fu, M. Nguyen, and W. Q. Yan, "Grading Methods for Fruit Freshness Based on Deep Learning," vol. 3, no. 4, p. 264. [Online]. Available: https://doi.org/10.1007/s42979-022-01152-7
6. A. Kazi and S. P. Panda, "Determining the freshness of fruits in the food industry by image classification using transfer learning," vol. 81, no. 6, pp. 7611–7624. [Online]. Available: https://doi.org/10.1007/s11042-022-12150-5
7. A. Vantarakis, M. Affifi, P. Kokkinos, M. Tsibouxi, and M. Papapetropoulou, "Occurrence of microorganisms of public health and spoilage significance in fruit juices sold in retail markets in Greece," vol. 17, no. 6, pp. 288–291. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1075996411000564
8. Determination of Fruit Freshness Using Near-Infrared Spectroscopy and Machine Learning Techniques — SpringerLink. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-19-3394-3$_{5}$2
9. "Fruit ripening process dataset." [Online]. Available: https://universe.roboflow.com/fruit-ripening/fruit-ripening-process
10. D. Gudovskiy, G. Han, T. Yamaguchi, and S. Tsukizawa. Smart Home Appliances: Chat with Your Fridge. [Online]. Available: http://arxiv.org/abs/1912.09589
11. D. A. Hudson and C. D. Manning. Compositional Attention Networks for Machine Reasoning. [Online]. Available: http://arxiv.org/abs/1803.03067
12. K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," vol. 3, no. 1, p. 9. [Online]. Available: https://doi.org/10.1186/s40537-016-0043-6
13. K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. [Online]. Available: http://arxiv.org/abs/1512.03385
14. K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5353–5360.
15. M. Keen. What are Convolutional Neural Networks? — IBM. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks
16. S. S. Talathi and A. Vartak. Improving performance of recurrent neural network with relu nonlinearity. [Online]. Available: http://arxiv.org/abs/1511.03771
17. M. Lin, Q. Chen, and S. Yan. Network In Network. [Online]. Available: http://arxiv.org/abs/1312.4400
18. Fruits fresh and rotten for classification. [Online]. Available: https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

19. Fruits and Vegetables dataset. [Online]. Available: https://www.kaggle.com/datasets/muhriddinmuxiddinov/fruits-and-vegetables-dataset
20. N. Sultana, "Fresh and Rotten Fruits Dataset for Machine-Based Evaluation of Fruit Quality." [Online]. Available: https://data.mendeley.com/datasets/bdd69gyhv8/1
21. M. Sikka. Balancing the Regularization Effect of Data Augmentation. Medium. [Online]. Available: https://towardsdatascience.com/balancing-the-regularization-effect-of-data-augmentation-eb551be48374
22. T. D. Detective. Finally: Why We Use an 80/20 Split for Training and Test Data Plus an Alternative Method (Oh Yes...). Medium. [Online]. Available: https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d
23. M. Alhamid. What is Cross-Validation? Medium. [Online]. Available: https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75
24. Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved YOLO-V3 model," vol. 157, pp. 417–426. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991831528X
25. M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html
26. V. Thambawita, I. Strümke, S. A. Hicks, P. Halvorsen, S. Parasa, and M. A. Riegler, "Impact of Image Resolution on Deep Learning Performance in Endoscopy Image Classification: An Experimental Study Using a Large Dataset of Endoscopic Images," vol. 11, no. 12, p. 2183. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/
27. C. F. Sabottke and B. M. Spieler, "The Effect of Image Resolution on Deep Learning in Radiography," vol. 2, no. 1, p. e190015. [Online]. Available: http://pubs.rsna.org/doi/10.1148/ryai.2019190015
28. A. M. Ghosh. Early Stopping with PyTorch to Restrain your Model from Overfitting. Analytics Vidhya. [Online]. Available: https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5
29. Training a Classifier — PyTorch Tutorials 2.0.1+cu117 documentation. [Online]. Available: https://pytorch.org/tutorials/beginner/blitz/cifar10$_t$*utorial.html*
30. A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images."
31. Writing Custom Datasets, DataLoaders and Transforms — PyTorch Tutorials 2.0.1+cu117 documentation. [Online]. Available: https://pytorch.org/tutorials/beginner/data$_l$*oading*$_t$*utorial.html*
32. Sorbo. BananaComputerVision/BananaStateClassification.py at main · asorbo/BananaComputerVision. GitHub. [Online]. Available: https://github.com/asorbo/BananaComputerVision
33. Number of training parameters in millions(M) for VGG, ResNet and... ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Number-of-training-parameters-in-millionsM-for-VGG-ResNet-and-DenseNet-models$_t$*bl*1$_3$38552250

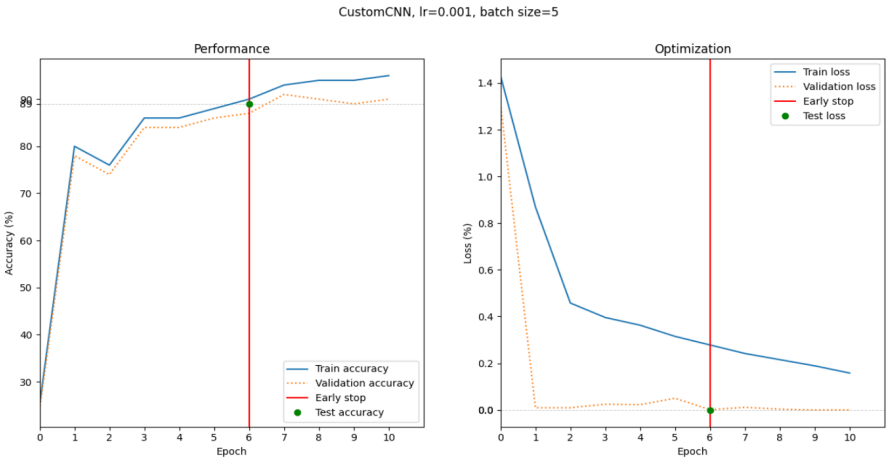


Fig. E: Training history of the best-performing custom CNN model

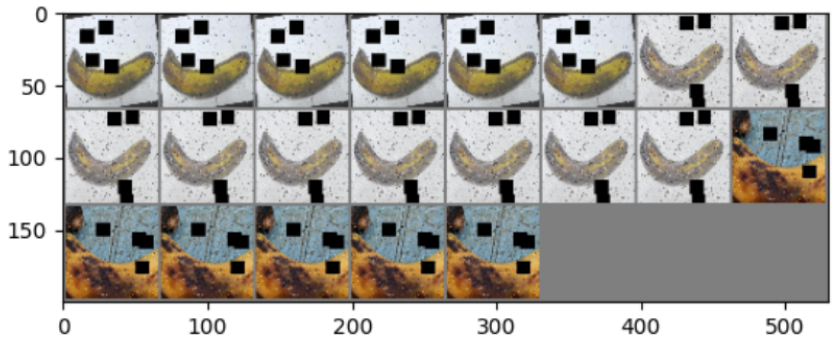


Fig. F: Rotten bananas misclassified as overripe by ResNet50

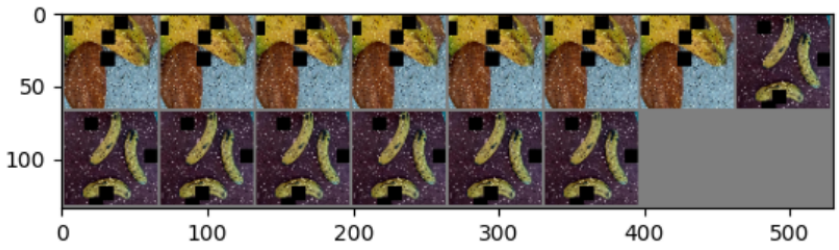


Fig. G: Ripe bananas misclassified as overripe by ResNet50

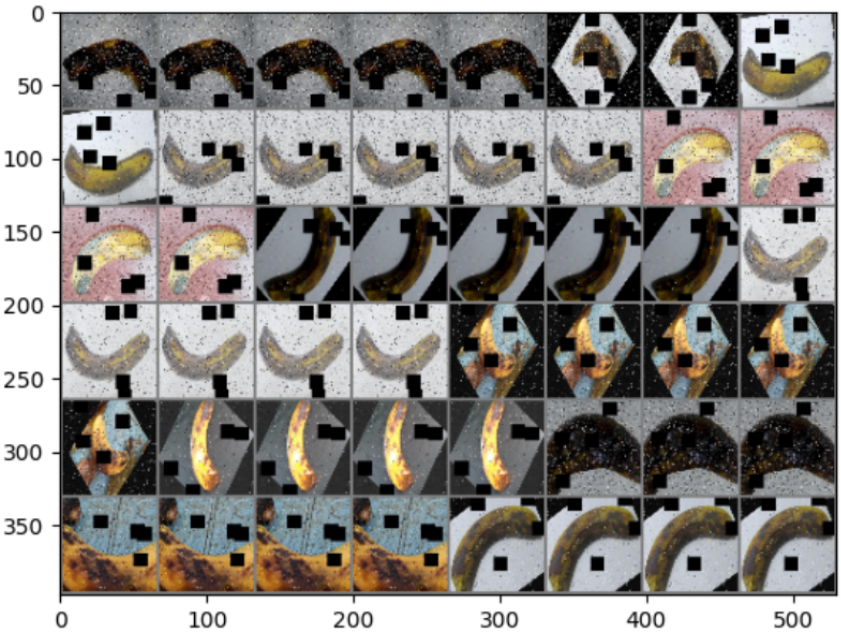


Fig. H: Rotten bananas misclassified as overripe by CustomCNN

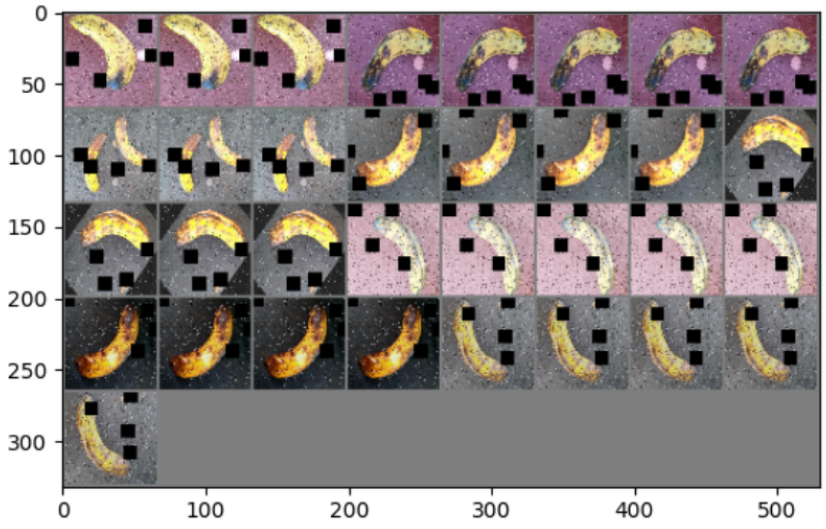


Fig. I: Rotten bananas misclassified as ripe by CustomCNN

Fig. J: Illustration of the ResNet50 architechture