ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Machine Learning and Content Analytics

Supermarket products image classification

Team members: Athanasiou Antonis (p2822102)
Atzami Stavroula (p2822105)
Gkourioti Panagiota (p2822109)
Koletsi Thaleia (p2822120)

Professor: Haris Papageorgiou
Assistant responsible for this assignment: George Perakis

Athens, 4/9/2022

Table of Contents

1. Introduction

1.1. Challenge Description

In recent years, technological advancements are significantly influencing the ways consumers interact with retailers and retailers communicate with their customers. To enhance customer experience and reduce costs, retailers make use of various technologies that facilitate and speed up the shopping process.

Currently, the checkout process at retail stores usually involves barcode scanning, which is used for product identification. Barcode scanning facilitates, on the one hand, the management of retail items, but, on the other hand, poses some issues. In particular, the barcode on the product might sometimes be misplaced, which requires time for the cashier to manually locate and enter the barcode. Moreover, this checkout method also poses environmental risks. Namely, the tags that are glued on the products generate significant plastic waste and can also be harmful to customers' health, since they leave glue residue on edible items, such as fruits or vegetables. Finally, the barcodes are very labor intensive, as they must be scanned individually and could also easily be ripped or damaged.

These challenges indicate that a more reliable product identification system would improve the retail store operations and increase overall customer satisfaction.

1.2. Project Mission

The need for a time-saving, cost-reducing and efficient checkout process has led an increasing number of supermarkets to adopt self-checkout systems. These systems allow customers to complete the purchase process without the help of retailer personnel, thereby reducing business and labor costs. With the progress of artificial intelligence and computer vision, many advancements can be made to the self-checkout systems that dramatically alter the shopping experience for the consumers and make the retailers competitive in the globalized markets.

By making use of electronic devices for photographing and by implementing automatic product recognition software, the checkout process could be performed exclusively by the customer. The replacement of traditional checkout counters with smart baskets that scan each product inserted into them could make shopping much easier and faster, without having to wait in long queues.

The aim of this project is to build a deep learning model that can efficiently identify supermarket products through images. More specifically, the camera integrated into the basket will scan each product and effectively analyze and process the image sequence, in order to identify and classify it. Then, the product will be added to the checkout along with its details and price.

In the next paragraphs, we are describing the process we have followed in order to achieve our mission. Firstly, we have collected and processed the data we needed, i.e. images from supermarket products, in order to be able to create the appropriate predicting CNN models. After training, and evaluating each one of them, we have selected the model with the best performance, which meets our goal to the greatest extent. At the end, we are proposing new ways on how automatic product recognition in retail stores can be useful and be adopted.

2. Data

2.1. Data collection

The beginning of the project is the data. The collection of the necessary input data is a crucial step to start creating various models and testing their performance and results.

In this project, the dataset collected consists of a variety of images from different kinds of consumer products. Initially, the idea was to work with the pool of images stored in supermarkets' e-shops but after experimenting with this, the results were not the desired ones. The transparent or white background in photos and the small number of images for each of the available products could raise issues when the camera will photograph the product placed in the basket in real-time. Produced images will have varied backgrounds and the products will be captured from different sides.

For that reason, we came up with the idea to take real pictures of the products in a supermarket environment. The purpose was to work with specific categories of consumer goods, therefore, the products used are various types of beverages, preserves, pasta, legumes, dairy products, detergents and mouth health. The algorithms will initially be trained and tested on these products, and at a next level, they will also be applied to other types of goods.

2.2. Dataset overview

The dataset includes 5427 images of 135 consumer products in jpg format. Each product has approximately ~60 images that depict the product from all angles and at different positions and backgrounds. There is a separate folder for every product containing all of its corresponding photos.

The images of each product have been named according to the respective folder name, which contains the product description and size. This renaming process was executed via a python script[1], which sequentially reads each folder and names the photos included in it using the folder name (meaning the SKU name) and a counter.

2.3. Data Processing/Annotation/Normalization

Taking into account that the size of the majority of images collected, either directly or through videos which have been converted to image sequences, was excessively large, we decided to compress them by reducing their size to ¼ according to their initial size before uploading them to Google Drive[2]. Additionally, in order to speed up the time needed to load the images into Google Collab every time, given that this process would be repeated many times, we created a compressed NumPy array in Google Drive which contains all our images[3]. Therefore, the time needed to load the photos in the notebook each time has been eliminated as we are reading directly the aforementioned compressed NumPy array. X Variables (X_TRAIN, X_TEST) represent the images' arrays and Y Variables (Y_TRAIN, Y_TEST) represent the labels of the products that should be predicted from the machine learning algorithms.

Concerning the image size, we concluded after testing that the most appropriate size for our models to be trained is 200x200. Subsequently, we split our dataset into train and test. The test dataset contains 25% of the whole images, and more specifically, 1357 images. The validation dataset has been created during the training phase, and contains the 25% of the training dataset.

In the last step, before starting to train and test our models, we have proceeded with the normalization of our data, by dividing both the train and test datasets by 255. Each pixel of an image has

[1] See "photo_renamer.ipynb"
[2] See "Compress Images.py"
[3] See "read_images.ipynb"

a number from 0 to 255 where 0 is the total white and 255 represents the total black. Therefore, the division of an image by 255 transforms any number in this image between 0 and 1.

3. Methodology

Given that the vast amount of image data produced by cameras is unstructured, it is imperative to find models and techniques to analyze the images efficiently. Deep learning, which uses artificial neural networks to perform sophisticated computations on large amounts of data, has gained significant popularity and has become the core solution for image classification and object detection.

Image classification is a fundamental problem in computer vision. It refers to a process that can classify an image according to its visual content and extract information classes from a multiband raster image. Image classification techniques are mainly divided into two categories: supervised and unsupervised.

The unsupervised classification algorithms analyze and cluster unlabeled datasets by finding common characteristics or hidden patterns within the data. The computer uses various methods to determine which pixels are related and groups them into classes based on their characteristics.

Supervised image classification, on the other hand, uses previously classified reference samples in order to train the classifier and subsequently classify new, unknown data. Specifically, the supervised image classification techniques involve visually choosing samples of training data within the image and allocating them to pre-chosen categories. This is done to create statistical measures to be applied to the overall image.

CNNs are one of the best learning algorithms for understanding image content and have shown exemplary performance in computer vision tasks, especially in image classification. They are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal pre-processing.

3.1. Custom CNN Model

Our first attempt at creating a product classification model was training from scratch a Convolutional Neural Network (CNN)[4]. CNNs differ from conventional Multi-Layered Perceptrons (MLPs) in architecture in respect to the layers they employ. Specifically, CNNs first pass the input through a series of Convolutional Layers.

Convolutional Layers are very similar to filters in the way they extract information looking at a limited input region, unlike MLPs and Fully Connected Layers, where each neuron is affected by every single Input. Due to this, MLPs have shown in practice poor object classification performance, which comes to no surprise if we think about how humans recognize objects in real life. For example, in order to classify a car, we are looking for very specific features such as 4 wheels, a specific car shape, car doors etc. Every other feature, such as a tree or a road, when faced with the specific problem of recognizing a car, is irrelevant and should be disregarded. Guided by this intuition, CNN utilizes Convolutional Layers to extract object features such as edges, shapes etc.

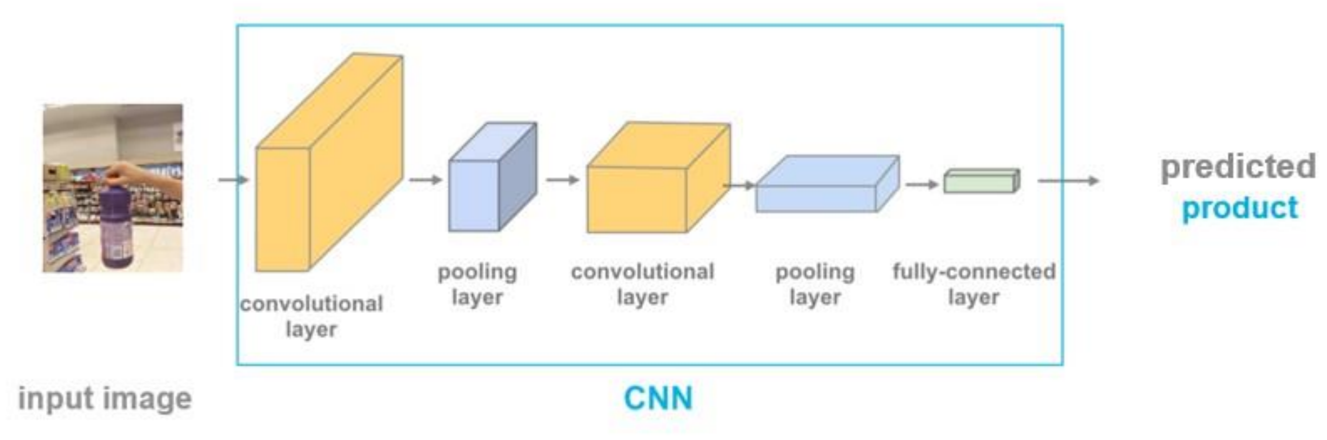


Figure 1: Convolutional Neural Network Architecture

A Convolutional Layer consists of several filters (called kernels), each looking at a specific Input Region (i.e. kernel size). A kernel can also have a stride, meaning how many pixels to skip when it reads the input image. Convolutional Layers also have an activation function, usually ReLu. All the above,

[4] See "Model_Building.ipynb"

number of kernels, kernel size, stride and activation function are the hyperparameters of the Convolutional Layer.

A Pooling Layer is used after the Convolutional Layer in order to extract the most prominent feature out of a filter and downsampling our input at the same time. This is done by inspecting a region and returning the max value of said region. As such, a Max Pooling Layer of a kernel size 2 by 2 effectively cuts in half the input that the next layer needs to process, while keeping the most important features computed by the convolutions.

After several passes of Convolutional Layers and Pooling Layers, which can be considered as the feature extraction phase, the final prediction is done by a fully connected layer part.

For our Custom CNN, we have trained:

- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 1
- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 1
- Fully Connected Layer, 4096 Neurons, ReLu
- Batch Normalization Layer
- Output Layer, SoftMax

3.2. VGG16

VGG16 is the next network we have used for product classification. VGG16 is very similar in terms of architecture with our previous Custom CNN Model. The difference lies between the number of layers and learnable parameters that they both have, with VGG16 being a much larger model (16 learnable layers vs 5 learnable layers). We have left the feature extraction part (i.e. the convolutions) as is, and we only trained the fully connected part to fit our data.

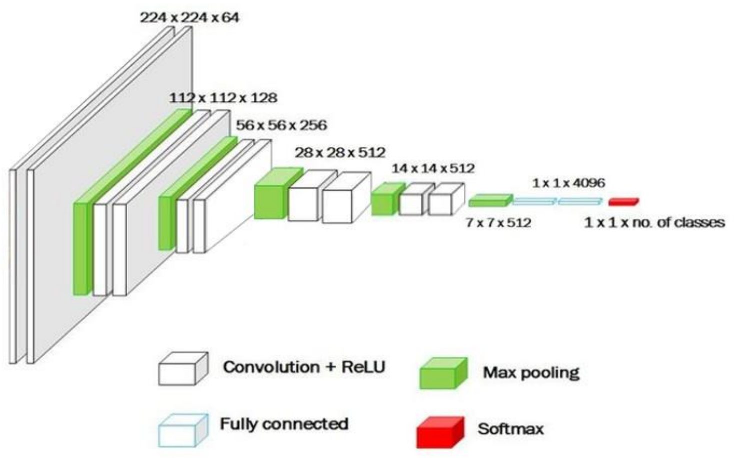


Figure 2: VGG-16 Architecture

VGG16 model's architecture is as follows:

- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 2
- Convolutional Layer, 128 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 128 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 2
- Convolutional Layer, 256 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 256 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 2
- Convolutional Layer, 512 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Convolutional Layer, 512 Filters, Kernel Size = 3x3, Stride = 1, ReLu
- Max Pooling, Kernel Size = 2x2, Stride = 2
- Fully Connected Layer, 4096 Neurons, ReLu
- Fully Connected Layer, 4096 Neurons, ReLu
- Batch Normalization Layer
- Output Layer, Softmax

3.3. RESNET50

ResNet, short for Residual Networks, is a classic neural network used as a backbone for many computer vision tasks. ResNet-50 is a pre-trained Deep Learning model for image classification of the Convolutional Neural Network, which is a class of deep neural networks, and it's most commonly applied to analyzing visual imagery. ResNet-50 is 50 layers deep and is trained on a million images of 1000 categories from the ImageNet database. Furthermore, the model has over 23 million trainable parameters, which indicates a deep architecture that makes it better for image recognition.

What characterizes a residual network is its identity connections (Figure 3). Identity connections take the input directly to the end of each residual block, as shown below with the curved arrow:

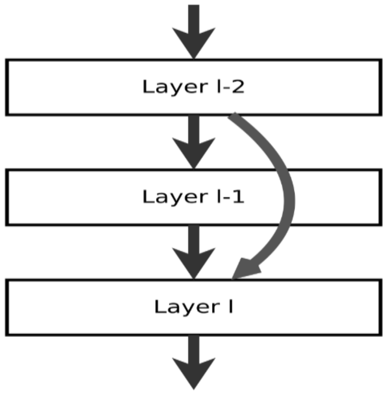


Figure 3: Identity connections

The ResNet-50 model consists of 5 stages, each with a residual block. Each residual block has 3 layers with both 1*1 and 3*3 convolutions. ResNet-50 model architecture contains the following elements (*Figure 4*):

- A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us **1 layer**.
- Max Pooling with also a stride size of 2.
- In the next convolution there is a 1 * 1, 64 kernel following this a 3 * 3, 64 kernel and at last a 1 * 1, 256 kernel. These three layers are repeated 3 times, giving us **9 layers** in this step.
- Nextly, there is a kernel of 1 * 1, 128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512. This step is repeated 4 times, giving us **12 layers** in this step.
- After that, there is a kernel of 1 * 1, 256 and two more kernels with 3 * 3, 256 and 1 * 1, 1024. This is repeated 6 times, giving us a total of **18 layers**.

- And then, there is a 1 * 1, 512 kernel with two more of 3 * 3, 512 and 1 * 1, 2048 and this is repeated 3 times, giving us a total of **9 layers**.
- After that, there is a max pooling and a Fully Connected Layer containing 4096 neurons with ReLu, a Batch Normalization Layer and at the end a Fully Connected Layer with 135 neurons, with a Softmax function, giving us **2 layers**.

We haven't count the activation functions and the max pooling layers.

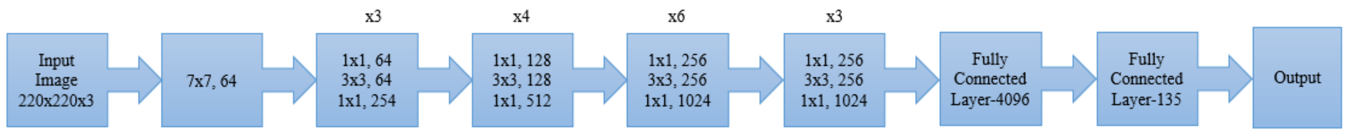


Figure 4: ResNet-50 Architecture

In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away, called identity connections. Residual connections between layers help reduce loss, preserve knowledge gain, and boost performance during the training phase. They allow the model to learn an identity function which ensures that the higher layer will perform at least as well as the lower layer, and not worse. A residual connection in a layer means that the output of a layer is a convolution of its input plus its input.

3.4. INCEPTIONV3

The final pre-trained model deployed in the project is InceptionV3. It is an advanced version of InceptionV1, a convolutional neural network model, which is used for image recognition and detection problems and was introduced in 2014 as GoogleNet. The model follows a sparsely connected network architecture which replaces fully connected network architectures and has 42 layers in total.

The main characteristic of the algorithm is that it contains an Inception module, meaning multiple filters of different sizes on the same level, in order to achieve parallel layering instead of having deep layers. This would prevent the loss of information that is caused when we make the structure deeper. Additionally, the use of Global Average Pooling 2D (GAP) replaces the Flatten and Dropout methods and contributes to avoid overfitting by doing high reductions. So, if we have a HxWxD tensor, GAP averages the HxW features and decreases the tensor to 1x1xD.

The architecture of the model (*Figure 5*) includes *factorized convolutions*, which contribute to the computational efficiency of the model by reducing the number of parameters, *small convolutions*, that replace the bigger ones and contribute to faster training, *asymmetric convolutions*, which amend the dimensions of a convolution without increasing the number of parameters used (*Figure 6*), as well as an *auxiliary classifier*, meaning a small CNN existing between layers, and supports regularizing. Finally, it includes a grid size reduction which often implements pooling operations.

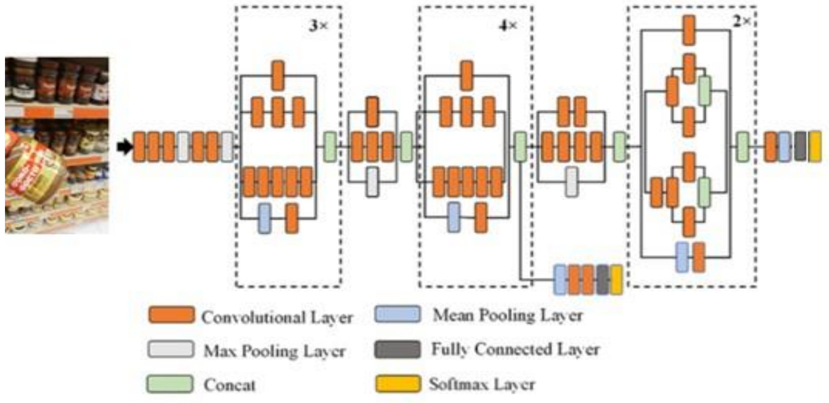


Figure 5: InceptionV3 Architecture

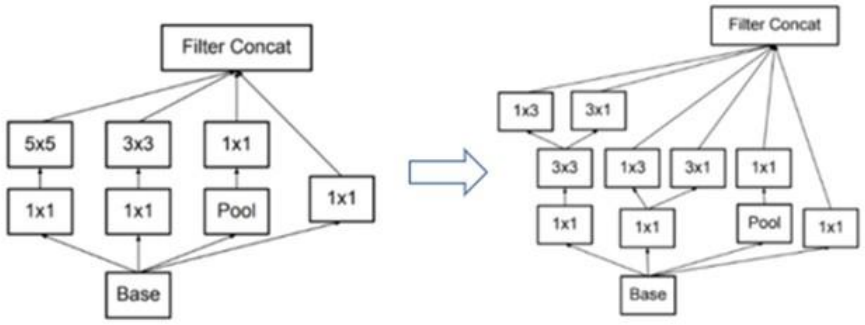


Figure 6: Asymmetric convolution rationale

InceptionV3 architecture is structured as follows:

- Convolutional Layer, 32 Filters, Kernel Size = 3x3, Stride = 1
- Convolutional padded Layer, 32 Filters, Kernel Size = 3x3, Stride = 1
- Max Pooling, Kernel Size = 3x3, Stride = 2

- Convolutional Layer, 64 Filters, Kernel Size = 3x3, Stride = 1
- Convolutional Layer, 80 Filters, Kernel Size = 3x3, Stride =2
- Convolutional Layer, 192 Filters, Kernel Size = 3x3, Stride =1
- 3x Inception (*Figure 7*)
- 5x Inception (*Figure 8)*
- 2x Inception (*Figure 9)*
- Average Pooling, Kernel Size = 8x8, Stride = 2
- Output Layer, Softmax

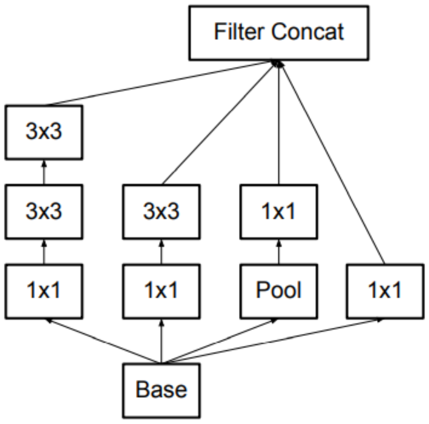


Figure 7: Inception module 1

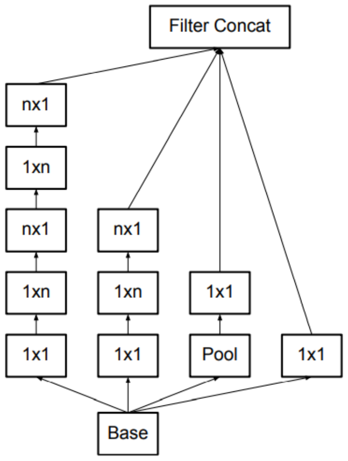


Figure 8: Inception module 2

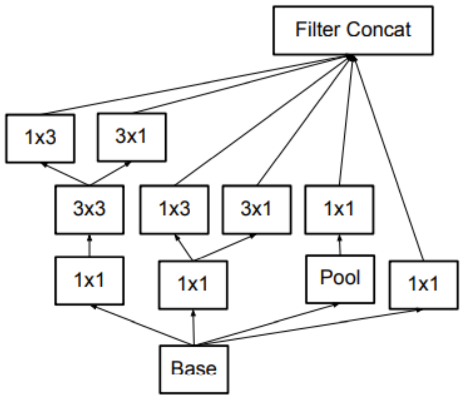


Figure 9: Inception module 3

4. Results & Quantitative Analysis (incl. visualizations)

After having discussed and analyzed the architectures and the hyperparameters for each of the models, the results and findings will be presented in this section. Specifically, the performance of each model will be assessed by creating classification reports, in order to compare the metrics of each class, and by plotting the learning curves regarding the training and validation accuracies as well as losses in order to evaluate the ability of each model to make accurate predictions[5].

The report shows the main classification metrics, namely precision, recall and f1-score, on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. **Precision** is a measure of a classifier's exactness, also called positive predictive value, and is defined as the ratio of true positives to the sum of true and false positives. **Recall** is a measure of the classifier's completeness and stands for the ability of a classifier to correctly find all positive instances. It is defined as the ratio of true positives to the sum of true positives and false negatives. The **f1-score** is a weighted harmonic mean of the previous two metrics. F1-scores are lower than accuracy measures as they embed precision and recall into their computation. **Support** is the number of actual occurrences of the class in the specified dataset. In our case of multi-class classification, we use macro and weighted averaging methods in the classification report in order to get a general idea of the metrics. The macro-averaged scores are computed by taking the arithmetic mean (unweighted mean) of all the per-class metrics. This method treats all classes equally regardless of their support values. The weighted-averaged scores are computed by taking the mean of all per-class metrics while considering each class's support. Support refers to the number of actual occurrences of the class in the dataset.

[5] See "Model_Evaluation.ipynb"

CNN

Custom Convolutional Neural Net converged to weights after 13 Epochs, computing in total 579,754,055 weights. After just 3 epochs, both validation loss and accuracy follow closely with training loss and accuracy with no significant variation. During training, a validation accuracy of 84% was achieved, while training data managed to fit the data perfectly (100% accuracy in training). This gap of 16% between training and validation indicates some sort of overfitting which we couldn't manage to overcome with regularization.

Each model was put to the test using a holdout test sample of images. The total number of Test Images was 1357. These images are novel to the network (never seen before in training) and contain products from all 135 classes. Out of 1357 images, the custom Convolutional Neural Network predicted 1157 of them correctly, achieving an 85,26% accuracy score.

The following figures show classification reports produced by the predictions of CNN model, containing the best and worst performing classes by f1-score.

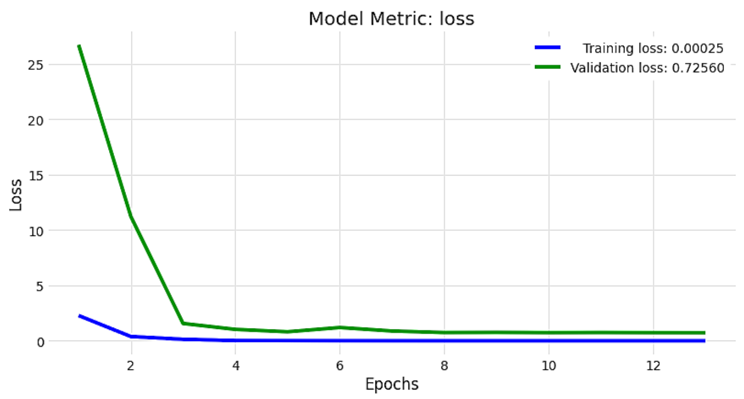
	labels	*f1_score*	*precision*	*recall*	*support*
0	**AFOI CHAITOGLOU CRUNCHY PEANUT BUTTER 350gr**	1.0	1.0	1.0	5
78	**LOUX PORTOKALADA ANTHR 330ML**	1.0	1.0	1.0	13
45	**DROLIO ME AROMA LOYLOYDIA**	1.0	1.0	1.0	5
49	**FAKES 3A PSILES**	1.0	1.0	1.0	4
52	**FANTA PORTOKALI MPLE 330ML**	1.0	1.0	1.0	13
55	**FASOLIA METRIA SKLAVENITIS**	1.0	1.0	1.0	1
65	**KARPOS ALMOND PROTEIN**	1.0	1.0	1.0	4
70	**KRIKRI YOGURT 2x140g**	1.0	1.0	1.0	4
71	**LACTA CREAM 360gr**	1.0	1.0	1.0	7
76	**LIPTON ICE TEA LEMON 500ML**	1.0	1.0	1.0	13

Table 1: CNN classification report – best class predictions by f1-score

	labels	*f1-score*	*precision*	*recall*	*support*
116	**SKIP KAPSOULES 20 MEZOURES SPRING FRESH**	0.00	0.00	0.00	4
118	**SKIP POWER CLEAN 70 MEZOYRES**	0.00	0.00	0.00	3
68	**KLINEX XLORINI ULTRA LEVANTE**	0.22	0.20	0.25	4
115	**SKIP KAPSOULES 20 MEZOURES POWER CLEAN**	0.25	0.50	0.17	6
4	**AIM WHITE SYSTEM ENAMEL 75ml**	0.29	0.50	0.20	5
44	**DODONI YOGURT STRAINED 2**	0.33	1.00	0.20	5
27	**COLGATE MAX WHITE CHARCOAL 75ml**	0.36	0.50	0.29	7
95	**OMO 45 MEZOURES TROPIKA LOYLOYDIA**	0.40	0.50	0.33	3
29	**COLGATE SENSITIVE INSTANT RELIEF 75ml**	0.48	0.42	0.56	9
101	**PARONDONTAX COMPLETE PROTECTION WHITE 75ml**	0.48	0.42	0.56	9

Table 2: CNN classification report – worst class predictions by f1-score

Visualizing the training loss against validation loss and training accuracy against validation accuracy over the number of epochs is a good way to determine if the model has been sufficiently trained.



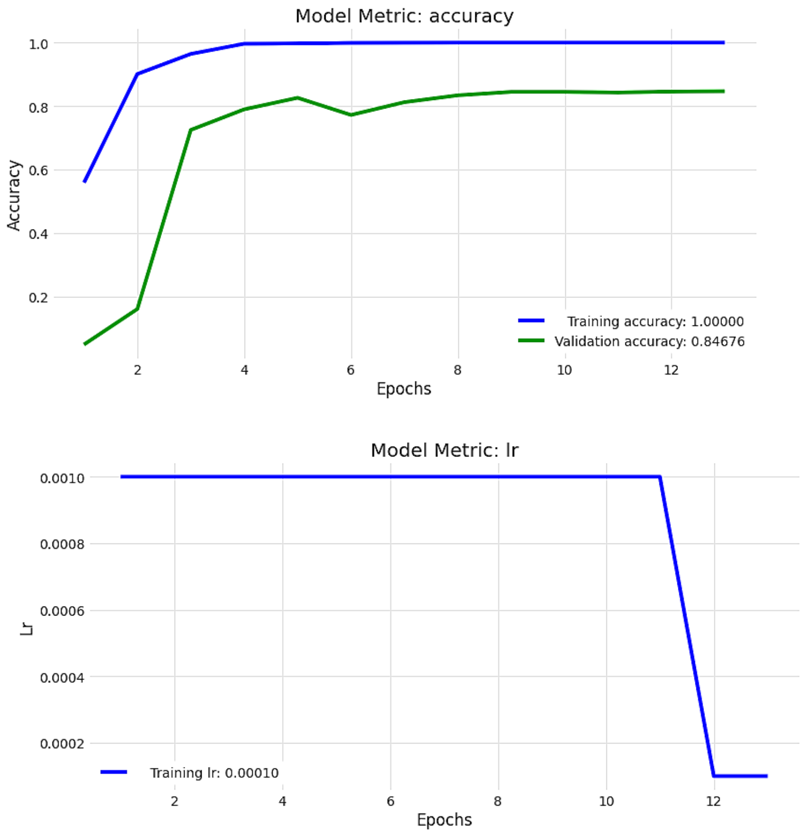


Figure 10: CNN Learning curves

The above plots of loss and accuracy depict the history of the model fit. It appears that the loss for training and validation is flattened after the third epoch. The parallel accuracy plots have a consistent gap between training and validation, which could indicate overfitting and, therefore, the training stops at an earlier epoch. The learning rate remains constant at 0.001 during 11 epochs, before declining to 0.0001 after reaching a plateau.

VGG16

VGG16 was the first pre-trained model we employed to tackle our product classification problem. The feature extraction was taken "as is" (wasn't trained). In total, 92,844,167 weights were fit to our data, which converged after 15 epochs.

Interestingly though, considerable variation is observed during the training phase, especially between epochs 5 and 9, where the accuracy score on validation fluctuates between roughly 80% and 25%[6]. Furthermore, the learning rate declined twice, firstly to 0.0001 and then to 0.00001, since there was no significant improvement in the accuracy of the model.

VGG16 achieved an accuracy score of 91.75% on the held-out test dataset, meaning that out of 1357 images, the VGG16 pre-trained model predicted 1245 of them correctly.

RESNET50

The next pre-trained model we applied to our image classification problem was RESNET50. RESNET50 converged to weights after 26 Epochs, significantly more than the other two, computing in total 8,953,991 weights.

It is notable that during the first 15 epochs of the training phase, the accuracy scores fluctuate greatly. For instance, the accuracy scores on validation, between epochs 10 and 11, vary between roughly 10% and 72% [7]. The learning rate follows the same trend as VGG16, as mentioned before.

RESNET50 achieved an accuracy score of 81.95% on the held-out test dataset, meaning that it classified accurately 1112 out of 1357 images.

INCEPTIONV3

Finally, the pre-trained model INCEPTIONV3 converged after 9 Epochs computing 280,711 weights. This was the fastest model to train since it took only 9 Epochs to converge, while having the fewest trainable parameters by a large margin.

With regard to the loss and accuracy learning curves, it is observed that after two Epochs, there is barely any improvement on the validation accuracy. It is also interesting to note that the validation accuracy is very high from the first Epoch of the model training[8].

[6] See Appendix Figure 12
[7] See Appendix Figure 13
[8] See Appendix Figure 14

Overall, INCEPTIONV3 achieved an accuracy score of 89.31% on the held-out test dataset, meaning that it classified accurately 1112 out of 1212 images.

To sum up, all models seem to have common products where they fail, as depicted also in relevant tables (*2*, *6*, *8*, *10*). Products like toothpaste and detergents are high in the scales with the worst class predictions. In addition to that, four models do not have significant differences with regard to the following metrics. All metrics are between 80% - 90% except from ResNet-50 macro average which is 77%, thus, overall, they have approximately similar classification ability, with VGG16 showing the best performance (overall accuracy 92%).

model		***f1_score***	***precision***	***recall***	***support***
	accuracy			0.85	1357
CNN	***macro avg***	0.84	0.8	0.8	1357
	weighted avg	0.86	0.85	0.85	1357
	accuracy			0.92	1357
VGG16	***macro avg***	0.9	0.87	0.87	1357
	weighted avg	0.93	0.92	0.92	1357
	accuracy			0.82	1357
RESNET	***macro avg***	0.8	0.76	0.77	1357
	weighted avg	0.82	0.82	0.81	1357
	accuracy			0.89	1357
INCEPTIONV3	***macro avg***	0.87	0.83	0.83	1357
	weighted avg	0.9	0.89	0.89	1357

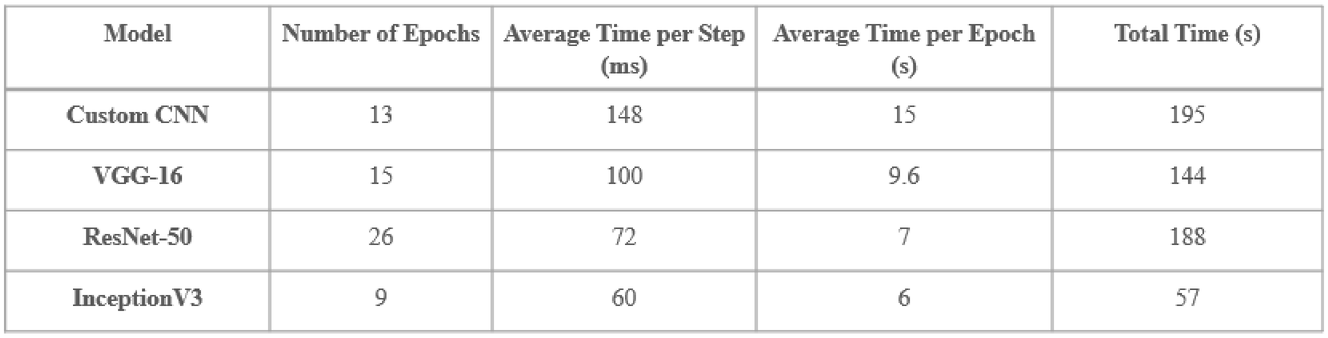
Table 3: 4 models - classification metrics comparison

5. Qualitative & Error Analysis

5.1. Qualitative Analysis

In terms of qualitative results, we would like to mention and analyze the time needed for each model to be trained. Our training dataset, after excluding the validation dataset, contains 3.053 images. As the batch size has been set to 32, which is the default batch size, each epoch is consisted of 96 steps. On the Custom CNN model each step from every epoch took around 148ms while the model took around 195s to be trained after 13 epochs. On the VGG16 model each step from every epoch took around 100ms while the model took around 144s to be trained after 15 epochs. On the ResNet-50 model each step from every epoch took around 72ms while the model took around 188s to be trained after 26 epochs. Lastly, on the InceptionV3 model each step from every epoch took around 60ms while the model took around 57s to be trained after 9 epochs.

We can clearly note that InceptionV3 took the least time to be trained while ResNet-50 took the most time. Also, we can note that the number of epochs that ResNet-50 needed to be trained is approximately two times greater than the other models. Furthermore, given that ResNet-50 has the worst accuracy in comparison with all the other models (*Table 4*), we have tried to increase our dataset by creating new images through augmentation[9], as a way to improve the performance of this model. Unfortunately, the results weren't the expected ones, as the accuracy of the model has decreased around 15-20%. Therefore, we didn't proceed with the augmented data but tried other ways to improve its performance, such as increasing the number of neurons in the last layer.



Model	Number of Epochs	Average Time per Step (ms)	Average Time per Epoch (s)	Total Time (s)
Custom CNN	13	148	15	195
VGG-16	15	100	9.6	144
ResNet-50	26	72	7	188
InceptionV3	9	60	6	57

Table 4: Time Performance of each model

5.2. Error Analysis

All the above models have been tested and developed based on single or aggregate metrics like accuracy, precision and recall that cover the model performance on the entire dataset. Error analysis helps to create Responsible ML models by identifying if the model is behaving more erroneously for certain classes. Thus, an in-depth review of the error cases can help on the improvement of the models' performance.

A few examples of wrongly predicted classes are presented below, for each model:

i) On the Custom CNN model, one of the products that the most of its images have been wrongly classified, as the recall is 55%, is "COLGATE SENSITIVE INSTANT RELIEF 75ml". 4 out of 9 images

[9] See "Augmentation script.py"

included in the test dataset have been wrongly predicted. The interesting thing here is that 3 out of 4 predictions are related to the same product, and more specifically to “PARONDOTAX COMPLETE PROTECTION WHITE 75 ml”. These two products are indeed very similar to each other, thus, the need of more images from both products would be helpful on the training phase of the model.

ii) On VGG16 model, 2 images of the product “SKIP POWER CLEAN 70 MEZOYRES” have been wrongly classified as “ARIEL GIGA PACK” and “KLINEX XLORINI ULTRA LEVANTE”. Both predictions are classified to detergents, thus, although the model identifies the category of the product, it doesn’t correctly predict the exact label.

iii) On ResNet-50 model, 3 out of 4 images of the product “KLINEX XLORINI ULTRA LEVANTE” have been wrongly classified as “KLINEX XLORINI ULTRA PINK POWER” or “KLINEX XLORINI ULTRA FRESH”. As also mentioned above, the model identifies the category of the product, but not the correct label.

iv) On the InceptionNetV3 model, 4 out of 5 images of the product “DODONI YOGURT STRAINED 2%” have been wrongly predicted as “DODONI YOGURT STRAINED”, “FAGE TOTAL 5%”, “FAGE TOTAL 2%” and “FUZE TEA GREEN 500ML”. Once more, the model seems to identify the category of the product, but not the exact product.

Although all the above models have gained an accuracy between 82% - 92%, there are many cases where they fail to predict the correct label. As we can conclude from the error analysis, the most times, the models identify the correct product category but not the correct products. As the most products of the same categories are very similar to each other, it is inevitable that the process of identifying the exact product will be difficult for the model.

A few approaches that might help on error resolution would be the collection of more images with higher quality from the products where the most errors are happening, data augmentation on those products and tuning hyperparameters differently.

6. Final Comments and Future Work

The implementation of AI and automatic product recognition would help solve many problems in the retail industry as well as improve store operations. It has been observed that auditing shelves manually and keeping track of inventory is very time-consuming as well as prone to errors. Product detection could help deal with this issue since the smart basket will keep a record of the products that are entered and bought, while automatically updating the stock levels. Thus, the staff can be informed in real-time about which products are being removed from the shelves, so that they can be replaced if needed. Another way that self-checkout with product recognition can be further exploited is by encouraging upselling. For instance, the basket could recommend similar items with lower prices or complementary items. Finally, these technologies could also assist the visually impaired people, by integrating software that reads the labels of the products out loud, so that they are able to shop independently.

7. Time Plan

As depicted below (*Figure 11*) the time plan was set in weeks 22 and 24 (*01/06 -15/06*). During the whole period, the project team was communicating through web calls in order to align on the processes, update on tasks' status and agree on the deadlines for each of the assigned tasks.

All along the first weeks, the team was searching on sites or ML/DL previous papers in order to find an interesting and eventually business useful idea for the project. Having found the topic, then the main work was focused behind data, thus, data collection and processing was the starting point and, as a next step, the model investigation (which models were appropriate for image classification, had better performance and results).

During the second half of the period, the team's effort was on code creation and implementation, models' evaluation based on the results and the writing of the report.

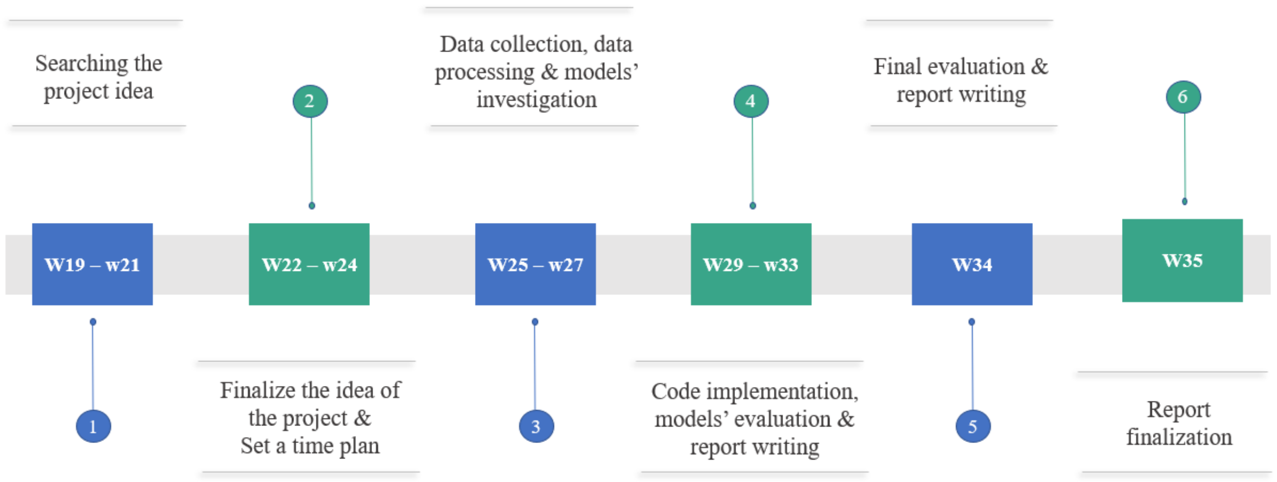

Searching the project idea
2
Data collection, data processing & models' investigation
4
Final evaluation & report writing
6
W19 – w21
W22 – w24
W25 – w27
W29 – w33
W34
W35
1
Finalize the idea of the project & Set a time plan
3
Code implementation, models' evaluation & report writing
5
Report finalization

Figure 11: Project timeline

8. Members & Roles

The project team consists of four members Athanasiou Antonis, Atzami Stavroula, Gkourioti Panagiota and Koletsi Thaleia. All team members were involved in each of the six stages described in the “time plan” section.

Specifically, for the data collection, all of the members were assigned separately with specific categories of products to take photos, label and process them for uploading. For the models’ creations, Athanasiou Antonis was responsible for CNN and VGG-16 code implementation and reporting, Atzami Stavroula for RESNET50 code implementation and reporting and Gkourioti Panagiota & Koletsi Thaleia for InceptionV3 code implementation and reporting. Finally, all members equally contributed to the creation of the final report.

9. Bibliography

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). *Rethinking the Inception Architecture for Computer Vision* (arXiv:1512.00567). arXiv. http://arxiv.org/abs/1512.00567

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions* (arXiv:1409.4842). arXiv. http://arxiv.org/abs/1409.4842

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. http://arxiv.org/abs/1512.03385

https://en.wikipedia.org/wiki/Residual_neural_network

Ali, L., Alnajjar, F., Jassmi, H. A., Gocho, M., Khan, W., & Serhani, M. A. (2021). Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures. *Sensors*, *21*(5), 1688. https://doi.org/10.3390/s21051688

https://medium.com/@nina95dan/simple-image-classification-with-resnet-50-334366e7311a#:~:text=What%20is%20ResNet%2D50%20and,applied%20to%20analyzing%20visual%20imagery

https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33#:~:text=The%20ResNet%2D50%20model%20consists,over%2023%20million%20trainable%20parameters

https://adventuresinmachinelearning.com/global-average-pooling-convolutional-neural-networks/

https://paperswithcode.com/method/global-average-pooling

10. Appendix

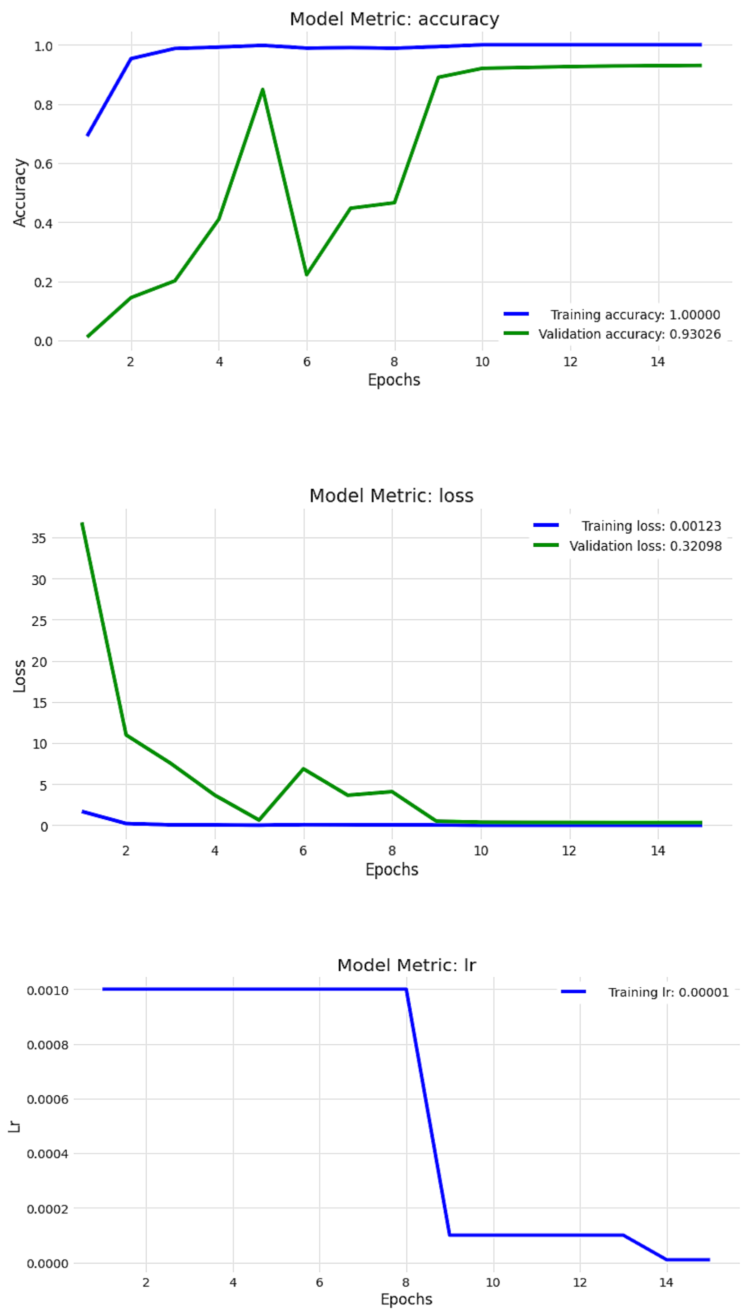


Figure 12: VGG16 learning curves

	labels	f1_score	precision	recall	support
46	FAGE AGELADITSA 2	1.0	1.0	1.0	6
54	FASOLIA 3A METRIA	1.0	1.0	1.0	5
81	MERENDA PAULIDIS PRALINA FOUNTOUKIOU 360gr KAI...	1.0	1.0	1.0	7
24	COCA COLA ZERO LEMON 330ML	1.0	1.0	1.0	13
65	KARPOS ALMOND PROTEIN	1.0	1.0	1.0	4
84	MISKO LIGOUINI LAZANIA	1.0	1.0	1.0	3
41	DELTA MILK XWRIS LAKTOZI	1.0	1.0	1.0	4
55	FASOLIA METRIA SKLAVENITIS	1.0	1.0	1.0	1
90	NIKOLAS REPANIS ERYTHROS OINOS AGIORGITIKO 750ML	1.0	1.0	1.0	23
17	BARILLA PENNE COLLEZIONE	1.0	1.0	1.0	5

Table 5: VGG16 classification report – best class predictions by f1-score

	labels	f1_score	precision	recall	support
55	FASOLIA METRIA SKLAVENITIS	0.00	0.00	0.00	1
118	SKIP POWER CLEAN 70 MEZOYRES	0.00	0.00	0.00	3
68	KLINEX XLORINI ULTRA LEVANTE	0.33	0.50	0.25	4
84	MISKO LIGOUINI LAZANIA	0.50	1.00	0.33	3
44	DODONI YOGURT STRAINED 2	0.50	0.67	0.40	5
95	OMO 45 MEZOURES TROPIKA LOYLOYDIA	0.50	1.00	0.33	3
32	COLGATE TOTAL WHITENING 75ml	0.59	0.56	0.63	8
30	COLGATE STOMATIKO DIALIMA PLAX ORIGINAL 250ml	0.60	1.00	0.43	7
77	LISTERINE SMART RINSE 250ml	0.60	0.75	0.50	6
116	SKIP KAPSOULES 20 MEZOURES SPRING FRESH	0.60	0.50	0.75	4

Table 6: VGG16 classification report – worst class predictions by f1-score

	labels	f1_score	precision	recall	support
46	FAGE AGELADITSA 2	1.0	1.0	1.0	6
54	FASOLIA 3A METRIA	1.0	1.0	1.0	5
81	MERENDA PAULIDIS PRALINA FOUNTOUKIOU 360gr KAI...	1.0	1.0	1.0	7
24	COCA COLA ZERO LEMON 330ML	1.0	1.0	1.0	13
65	KARPOS ALMOND PROTEIN	1.0	1.0	1.0	4
84	MISKO LIGOUINI LAZANIA	1.0	1.0	1.0	3
41	DELTA MILK XWRIS LAKTOZI	1.0	1.0	1.0	4
55	FASOLIA METRIA SKLAVENITIS	1.0	1.0	1.0	1
90	NIKOLAS REPANIS ERYTHROS OINOS AGIORGITIKO 750ML	1.0	1.0	1.0	23
17	BARILLA PENNE COLLEZIONE	1.0	1.0	1.0	5

Table 7: RESNET50 classification report – best class predictions by f1-score

	labels	f1_score	precision	recall	support
26	COLGATE GUM INVIGORATE REVITALIZE 75ml	0.00	0.00	0.00	4
116	SKIP KAPSOULES 20 MEZOURES SPRING FRESH	0.00	0.00	0.00	4
95	OMO 45 MEZOURES TROPIKA LOYLOYDIA	0.00	0.00	0.00	3
27	COLGATE MAX WHITE CHARCOAL 75ml	0.00	0.00	0.00	7
4	AIM WHITE SYSTEM ENAMEL 75ml	0.20	0.20	0.20	5
68	KLINEX XLORINI ULTRA LEVANTE	0.22	0.20	0.25	4
117	SKIP POWER CLEAN 45 MEZOURES	0.29	0.50	0.20	5
44	DODONI YOGURT STRAINED 2	0.29	0.50	0.20	5
1	AIM WHITE NOW FOREVER WHITE 75ml	0.31	0.29	0.33	6
3	AIM WHITE NOW SMILE DETOX 75ml	0.36	0.33	0.40	5

Table 8: RESNET50 classification report – worst class predictions by f1-score

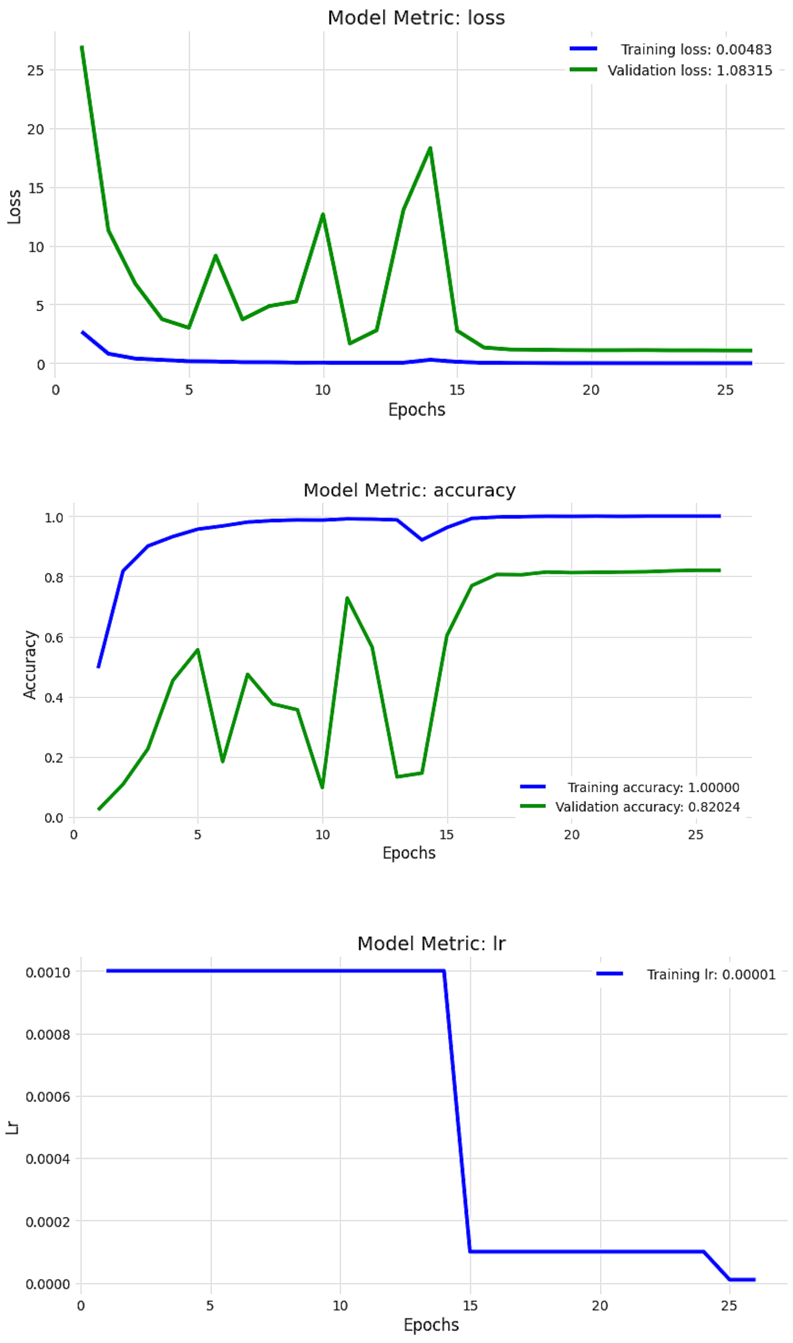


Figure 13: RESNET50 learning curves

	labels	*f1_score*	*precision*	*recall*	*support*
0	**AFOI CHAITOGLOU CRUNCHY PEANUT BUTTER 350gr**	1.0	1.0	1.0	5
85	**MISKO SPAGGETI N7**	1.0	1.0	1.0	5
41	**DELTA MILK XWRIS LAKTOZI**	1.0	1.0	1.0	4
54	**FASOLIA 3A METRIA**	1.0	1.0	1.0	5
63	**KANENAS TSANTALI ERYTHROS OINOS 750ML**	1.0	1.0	1.0	23
65	**KARPOS ALMOND PROTEIN**	1.0	1.0	1.0	4
66	**KLINEX KATHARISTIKO GENIKIS SPRAY**	1.0	1.0	1.0	5
70	**KRIKRI YOGURT 2x140g**	1.0	1.0	1.0	4
72	**LEMONADA EPSA 232 ML**	1.0	1.0	1.0	13
74	**LIKER BAILEYS 350 ML**	1.0	1.0	1.0	21

Table 9: INCEPTIONV3 classification report – best class predictions by f1-score

	labels	*f1_score*	*precision*	*recall*	*support*
21	**BARILLA SPAGGETI N5**	0.00	0.00	0.00	5
1	**AIM WHITE NOW FOREVER WHITE 75ml**	0.00	0.00	0.00	6
55	**FASOLIA METRIA SKLAVENITIS**	0.00	0.00	0.00	1
44	**DODONI YOGURT STRAINED 2**	0.29	0.50	0.20	5
16	**BARILLA FARFALE N65 500G**	0.40	1.00	0.25	4
57	**GIOTIS SIROPI CARAMEL 350gr**	0.46	0.50	0.43	7
77	**LISTERINE SMART RINSE 250ml**	0.50	1.00	0.33	6
118	**SKIP POWER CLEAN 70 MEZOYRES**	0.50	1.00	0.33	3
32	**COLGATE TOTAL WHITENING 75ml**	0.50	0.75	0.38	8
31	**COLGATE TOTAL ORIGINAL 75ml**	0.50	0.44	0.57	7

Table 10: INCEPTIONV3 classification report – worst class predictions by f1-score

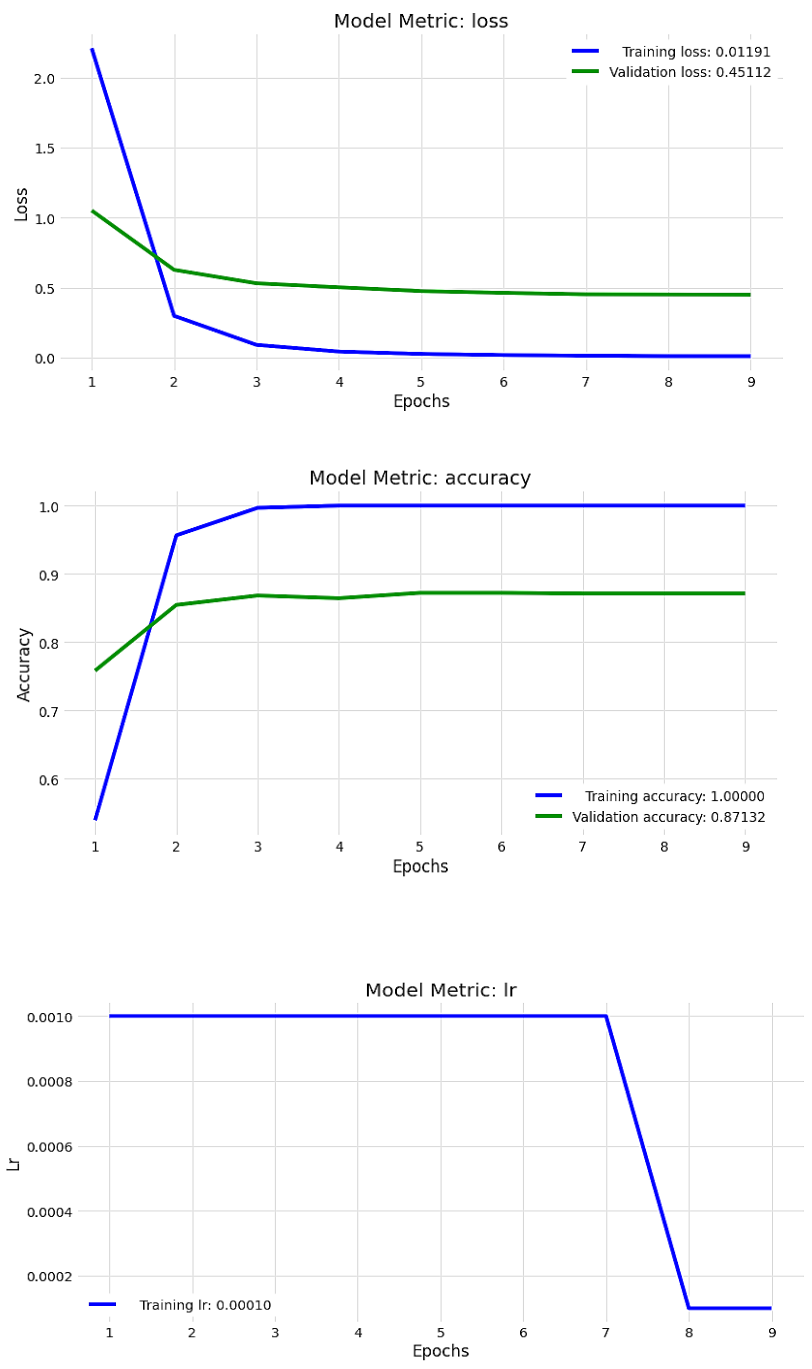


Figure 14: INCEPTIONV3 learning curves