Athens University of Economics and Business

Msc in Data Science

Master's Thesis

"Product Categorization and Matching Using Deep Learning Techniques"

Ioannis Daskalopoulos

supervised by

Prodromos Malakasiotis

Athens, November 2019

Abstract

In the field of e-business consulting, a great number of products is in need to be analyzed daily in order for insights to be extracted. Such a task is heavily reliant on the categorization and matching of the incoming products. Manual labeling is becoming costly for the firms as the influx of data seems to only increase. Automating this process is of high value and importance.

In this thesis, we first tackle product categorization using deep learning techniques to classify products according to their brand, type, and product line, based exclusively on their images. Taking into consideration the limited available resources we employ EfficientNet a lightweight Convolutional Neural Network which was found to achieve state-of-the-art results in image classification. Subsequently, we try to match new images with the corresponding products based solely on the images' embeddings extracted by the trained models of the previous task (product classification). The achieved results in both tasks are very promising showing that product categorization and matching based exclusively on images is feasible.

Περίληψη

Στον τομέα της παροχής συμβουλών σε ηλεκτρονικές επιχειρήσεις, απαιτείται καθημερινά η ανάλυση ενός μεγάλου αριθμού προϊόντων με σκοπό την εξαγωγή των χρήσιμων πληροφοριών που τα διέπουν. Αυτή η διαδικασία εξαρτάται σε μεγάλο βαθμό από την κατηγοριοποίηση και την αντιστοίχιση των εισερχόμενων προϊόντων. Η χειρωνακτική επισημείωση των κατηγοριών στις οποίες ανοίκει κάθε προϊόν καθίσταται δαπανηρή για τις επιχειρήσεις, καθώς ο όγκος των εισερχόμενων δεδομένων συνεχώς αυξάνεται. Η αυτοματοποίηση της διαδικασίας αυτής έχει μεγάλη αξία και σημασία.

Σε αυτή τη διπλωματική εργασία, εξετάζουμε αρχικά μεθόδους ταξινόμησης προϊόντων, χρησιμοποιώντας τεχνικές βαθιάς μάθησης, έτσι ώστε να ταξινομήσουμε τα προϊόντα ανάλογα με το εμπορικό σήμα, τον τύπο και την προϊοντική γραμμή τους, βασισμένοι αποκλειστικά στις εικόνες τους. Λαμβάνοντας υπόψη τους περιορισμένους διαθέσιμους πόρους για τη διεξαγωγή της διπλωματικής, χρησιμοποιούμε το EfficientNet, ένα ελαφρύ υπολογιστικά Συνελληκτικό Νευρωνικό Δίκτυο το οποίο επιτυγχάνει υπερσύγχρονα αποτελέσματα στην ταξινόμηση των εικόνων. Στη συνέχεια, προσπαθούμε να αντιστοιχήσουμε νέες εισερχόμενες εικόνες με υπάρχοντα προϊόντα κάνοντας χρήση των ενσωματωματώσεων των εικόνων οι οποίες εξάγονται από τα μοντέλα που έχουν εκπαιδευτεί στην προηγούμενη διαδικασία (ταξινόμηση προϊόντων). Τα επιτευχθέντα αποτελέσματα είναι εξαιρετικά, δείχνοντας ότι η ταξινόμηση προϊόντων και η αντιστοίχιση τους με βάση αποκλειστικά τις εικόνες είναι εφικτή.

Acknowledgements

I would like to express my gratitude to my supervisor Prodromos Malakasiotis for providing me with a great deal of support and assistance through the learning process of this master's thesis. His expertise and directions throughout this endeavor were invaluable. Furthermore, I would like to thank Konstantinos Pechlivanis and Ilias Antonopoulos for introducing me to the topic as well for extended discussions and valuable suggestions which have contributed greatly to the improvement of the thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Product Categorization and Matching

In the past few years, there has been a surge of new retailers being involved in e-business. Consequently, they brought forth an abundance of data rich in information but poor in structure, rendering the distillation of insights difficult. The aforementioned products are displayed with a plethora of different identifiers. This absence of global identifiers has complicated the task of searching and comparing products for customers. It also imposes difficulties in the extraction of meaningful statistics for analysts. As a result, the need to rearrange and aggregate this information into one unified structure has surfaced. To form such a structure one needs to attach labels to the products in order to organize them in groups that are meaningful to humans. Achieving such a feat is no easy task, as it is reliant upon the successful completion of two different tasks, product categorization and matching.

Categorizing the products according to business criteria is a critical task of utmost importance for e-consulting firms. If the data are to be consumed by analysts, a lot of business knowledge

is used in order to create non redundant labels that describe the product. In the past, such firms would go in great lengths to accomplish such a task, usually dedicating plenty of human resources. With the rise of machine and deep learning a lot of effort has been directed into the automation of such processes. In this task, we will start with the unstructured data and attach labels to them, portraying the fundamental properties of each product. Such labels can be the brand, product type, price range etc. The more labels available, the better the description of the product. Figure 1.1 illustrates the categorization process.

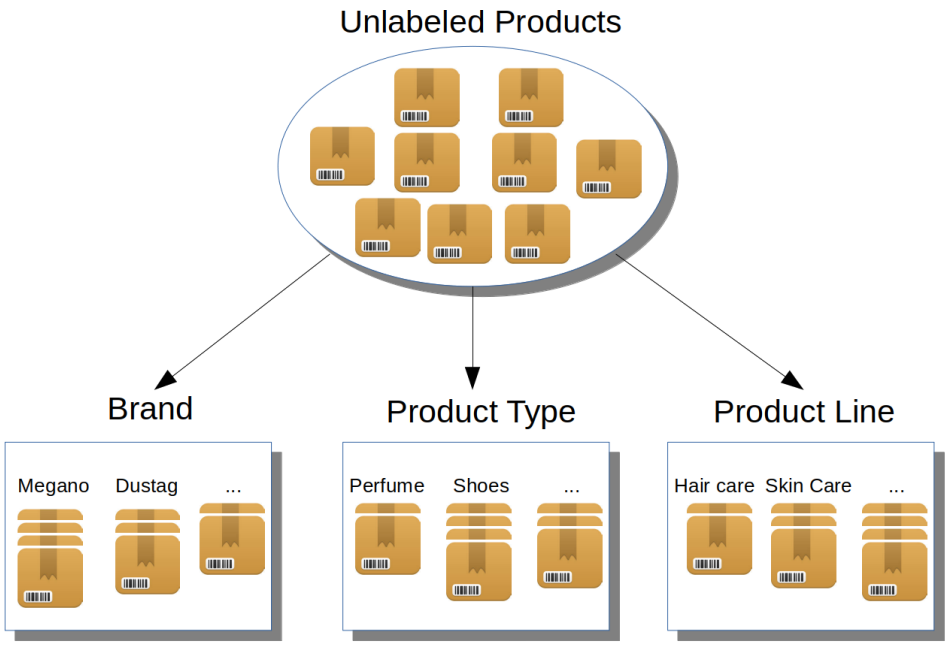


Figure 1.1: The same products are categorized according to brand, category and product line.

Product categorization on its own does not suffice for refining the data into a proper structure. Typically, an e-consulting firm possesses a plethora of products originating from different stores. As a consequence, a lot of duplicate products may exist, rendering the categorization process redundant in many cases. Thus, we need to be able to discern if a set of products originating from different stores, is actually the same product or not. The task described above is known as product matching and is portrayed in figure 1.2.

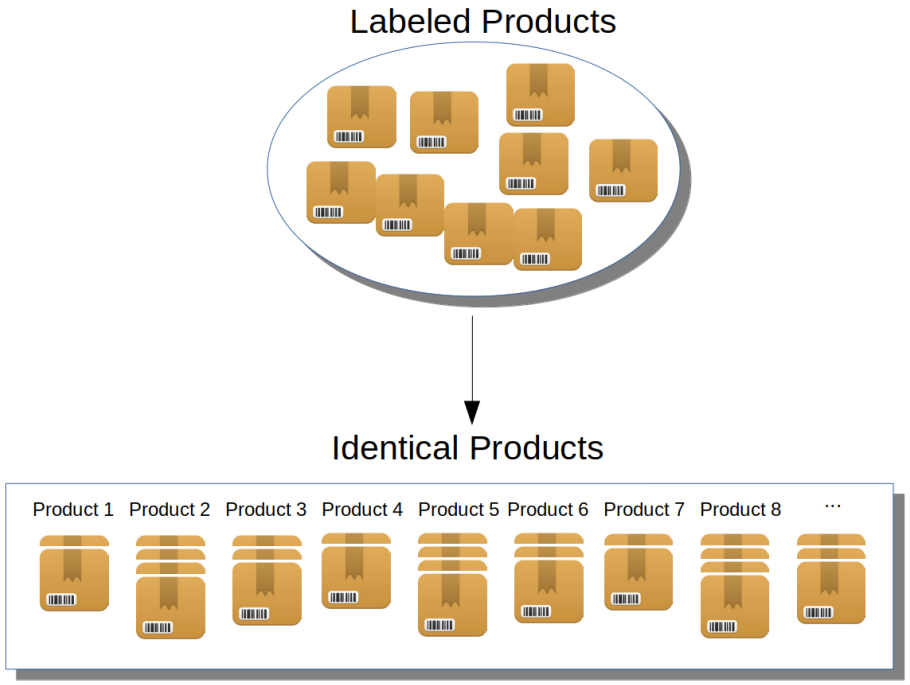


Figure 1.2: The products which were previously categorized are now split into classes holding identical products.

1.2 Goal of the thesis

As of yet, there has been no definitive solution to the problem of product matching. The fact that the problem is still open, has allowed for the development of a variety of approaches towards its solution. A popular approach among them is the use of text embeddings which encapsulate distinct characteristics of the textual information of the product. This approach, albeit successful, does not utilize the complete information available for each product. Utilizing the images could create a more complete representation of the product.

This thesis is conducted in collaboration with a company that operates in the field of e-business consulting. We are given a dataset extracted from the company's database, containing information about products sold by retailers in Greece. Each entry contains some textual information about the product, URLs pointing to images of the product in the retailers' site, and the preassigned labels brand id, category id, product line id and barcoded product id. The first 3 labels are to be used for product categorization and the last for product matching.

To cope with the problem of real time product categorization and matching, they have developed a system exclusively relying on textual information. As such it is unable to handle cases with poor or no textual information. Our task is to explore the application of deep convolutional neural network architectures in the context of image embeddings, in an endeavor to support the systems they have already implemented. We start by extracting images from their respective websites. Then, we refine the dataset into a proper format to train three neural networks, each one predicting predict brand, product type and product line for the products. We then use the trained models to ex-

tract image embeddings which in turn are given to a nearest neighbor algorithm which is responsible for product matching. As we show later, with the recent advancements in computer vision, the information obtained from the images is not only useful, but can also rival the performance obtained from text embeddings. Concluding, the integration of the work provided by this thesis can greatly enhance their existing systems.

1.3 Outline

The rest of the thesis is organized as follows:

- Chapter 2 discusses related work.
- Chapter 3 describes the dataflow of the system from image extraction to model architectures.
- Chapter 4 presents the methodology and experiments.
- Chapter 5 draws conclusions and proposes new ideas for future work.

Chapter 2

Related Work

The work of this thesis is inspired by Ristoski et al. [7] who experimented on an existing database containing structured information about a set of products. They identified matching products in unstructured product descriptions which were extracted from the Web and described with Microdata annotation, using the existing database as supervision. Then, they enriched the existing database with the matched product data and categorized them according to a set of new universal labels. They achieve the aforementioned tasks by making use of Conditional Random Fields for the extraction of attributes from textual descriptions and Convolutional Neural Networks to produce the embeddings from the products' images. The labels in which the data were categorized were a superset of the ones used in our thesis. Also, the process of extracting the embeddings from images in order to perform product matching, in spite of using simplistic Convolutional Neural Network architectures, is similar to the process that will be followed by this thesis. By making use of both text and image embeddings, they produce superior results than they do by using only one source of data, leading us to the conclusion that by combining the existing system with

the work of this thesis, we will also get better performance from the integrated system.

An earlier work conducted by Anitha Kannan et al. [3] proposes that in the field of e-commerce, using images as a weak classifier can enhance the performance of existing systems that rely solely on textual information to classify products. The dataset they experiment on is extracted from 17 categories related to computing from the Bing Shopping catalog. The authors use the images to solve problems that are encountered in textual information i.e. overlapping text across categories, short and undescriptive text, discrepancies in the vocabulary usage. They manage to improve a text-only classifier with their new integrated system. Finally, they conclude that such integrated systems must have the ability to recognize which signal can better discriminate the product, use the weaker image (i.e. images) selectively and adapt to changes in the vocabulary.

More recently, Tom Zahavy et al. [14] experiments on a dataset with 1.2 million collected from Walmart.com. The team manages to create a multi-modal system, which is a system that utilizes both text and images, in order to perform large-scale item categorization. The system uses two different CNN architectures for image and text classification and a Deep Policy network that decides which network to trust more for each decision. The results suggest that the text classifier is often more informative than the image classifier but there is a portion of the dataset that the image classifier finds correctly while the text classifier does not. By managing to determine which of the image classifier's predictions are correct, a multi-modal system can exceed state-of-the-art-systems that do not utilize both text and images.

Chapter 3

Methods

3.1 Product Categorization and Matching Pipeline

Figure 3.1 shows our system's pipeline for categorizing products. The initial step is to get the images from the given URLs. For this reason, a very basic crawler has been implemented. The images are stored locally and a new dataset pointing to their locations in the local disk is created. Afterwards, a generator transforms and pushes the images into neural network. Finally, the neural network produces a prediction for each one of the three labels.

Figure 3.2 shows our system's pipeline for matching products. We begin by pushing the images again into the neural network which was trained for categorization. This time by removing the last layers from it we extract the images' embeddings. Then, we perform KNN in order to find the embeddings that are closest to each other as they are likely to represent the same product.

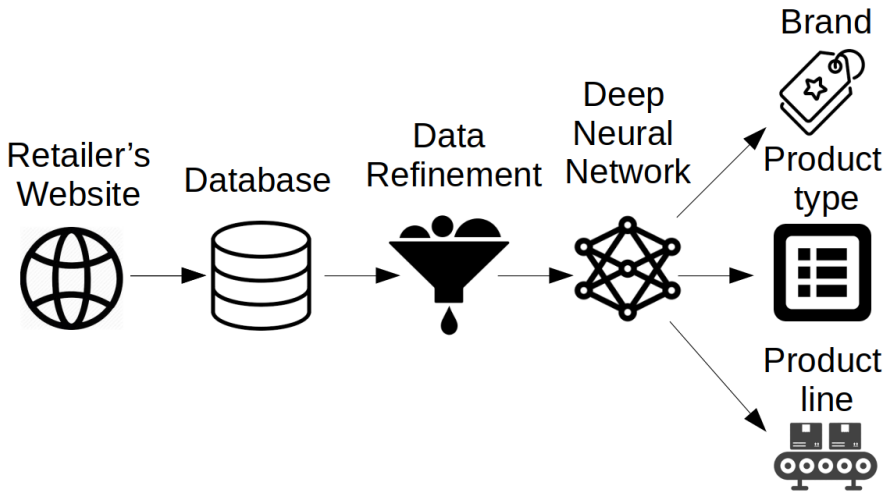


Figure 3.1: Product information crawled from the web is prepossessed and fed into a network to produce probabilities

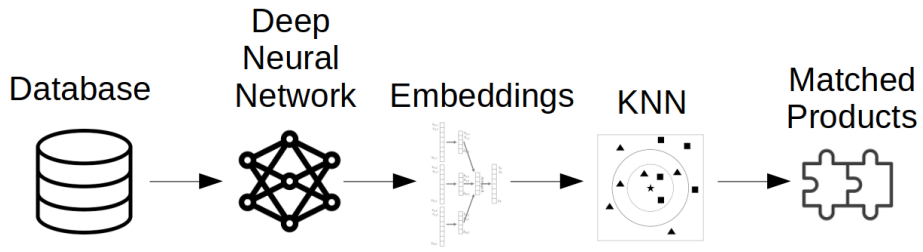


Figure 3.2: Product information crawled from the web is prepossessed and fed into a network to produce probabilities

3.2 Data Collection

For the collection of the data we use a crawler which is an internet bot that systematically browses the internet in order to interact with it, usually updating or downloading content from it. Crawling consumes resources from the targeted website and thus, is not always welcomed by it. Websites often impose re-

strictions on crawlers such as allowing only certain parts of the website to be crawled, or blocking the crawler if it exceeds a certain number of requests per minute.

The crawler implemented is a very basic one with its sole purpose of downloading images. A basic delay is applied between each request (1s) to ensure that our IP will not be blocked by any website resulting in lost images. All the images will be stored locally in a dynamically created folder structure with product ID as folder name and the respective images inside. Then, a new dataset containing the locations of the stored images and their labels is created. The reason such a folder structure is important, is that it can be easily updated in the influx of new data without compromising the old data. In other words, it ensures that duplicates will not be downloaded and only new products will be requested from the websites. To improve the efficiency of the crawler a multi-threading mechanism was employed, with each thread being responsible for making requests to different websites.

3.3 The Generator

Sometimes, even the most state-of-the-art configurations have not enough memory to process a large enough dataset. Datasets that consist of images are a prime example of this description. To deal with this problem we use a generator which will constantly load images from the local storage, transform them and push them into the neural network.

The generator will start by reading the images' locations in the local storage from the dataset and load the corresponding images. Afterwards, it will convert images with any number of

channels (i.e. Grayscale, RGB, RGBA) to RGB and resize them to a specified square resolution (i.e. 112x112 or 224x224). Also, the values contained in each channel will be normalized to [0, 1] to improve neural network performance. Optionally, more transformations can occur. In our case we used horizontal shift.

3.4 Models

3.4.1 Architectures

Convolutional Neural Networks (CNN) have been constantly producing state-of-the-art results for a lot of computer vision tasks. Their success lies in the constant research for novel architectures in order to improve the existing benchmarks. Due to the sheer number of options available, a multitude of architectures was examined before choosing the best one for the problem. DenseNet [2] was used as the baseline model as it produced decent results while costing relatively cheap in resources. Then, we used the more complex Convolutional Neural Network architectures Nasnet [15] and InceptionResnetV2 [9] increasing the system's performance in the process. All the aforementioned models were trained firstly with the ImageNet weights and a few last layers unfrozen and then with random initialized weights. In all the experiments the models with the random initialized weights produced superior results. Nevertheless, the next architecture that will be discussed outperformed the previous ones by a large margin. The comparison will be discussed thoroughly in the experiments section. Figure 3.5 showcases a comparison between EfficientNet family and other popular architectures. All models in the figure were trained in the popular Imagenet dataset.

3.4.2 EfficientNet

Tan et al. [10] approaches the process of finding better architectures by developing a basic low cost model which then can be scaled up in order to achieve better performance. Scaling up in this case means increasing the width and depth of the network and the resolution of the given images. Tuning those 3 parameters for the best performance usually requires a lot of manual tuning and resources. In this thesis we experiment with EfficientNet B5 which allows the scaling process to be done in a more principled manner by using a compound coefficient to scale up the network. As a consequence, by tuning only one hyperparameter, the model can scale-up its width, depth and resolution uniformly by choosing from a set of fixed values for the previously mentioned compound coefficient. This allows for more efficient tuning while also producing models that can achieve state-of-the-art performance with 10 times more efficiency. The basic EfficientNet architecture is called EfficientNet B0 and its architecture is depicted in figure 3.3.

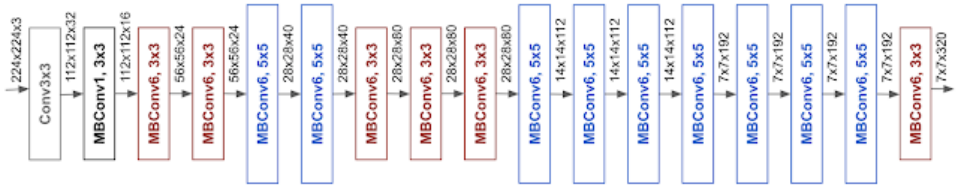


Figure 3.3: Architecture of Efficientnet-B0

3.4.3 Compound Scaling

Compound scaling is based on the idea that properly balancing width, depth and resolution according to the available resources

will provide superior performance overall. To find the compound scaling coefficient one must perform a grid search to determine the relationship between those 3 hyperparameters always under the restriction imposed by the resources. Once the relationship is found, the coefficients must scale up until they take up all the available resources. The authors propose the following formula that describes the relationship between the three hyperparameters:

$$\textbf{depth: } d = \alpha^{\phi} \tag{3.1}$$

$$\textbf{width: } w = \beta^{\phi} \tag{3.2}$$

$$\textbf{resolution: } r = \gamma^{\phi} \tag{3.3}$$

$$\textbf{s.t. } \alpha * \beta^2 * \gamma^2 \approx 2 \tag{3.4}$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{3.5}$$

ϕ is a user-specified coefficient that controls resources in terms of FLOPs (Floating Point Operations) and α, β, γ distribute the resources to depth, width, and resolution respectively. FLOPS of a regular convolution op is almost proportional to d, w^2, r^2, hence doubling the depth will double the FLOPS while doubling width or resolution increases FLOPS almost by four times. Hence, in order to make sure that the total FLOPS don't exceed 2^{ϕ}, the constraint applied is that $(\alpha * \beta^2 * \gamma^2) \approx 2$. Figure 3.4 visually expresses how compound scaling affects the network.

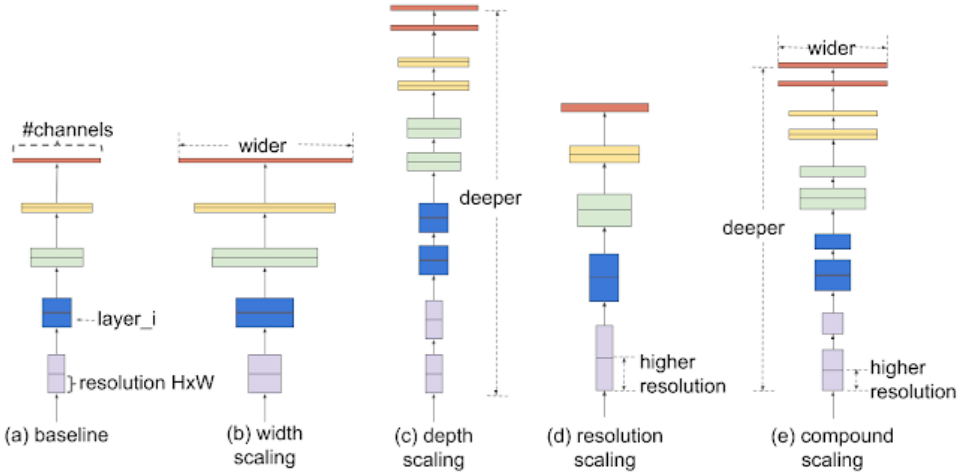
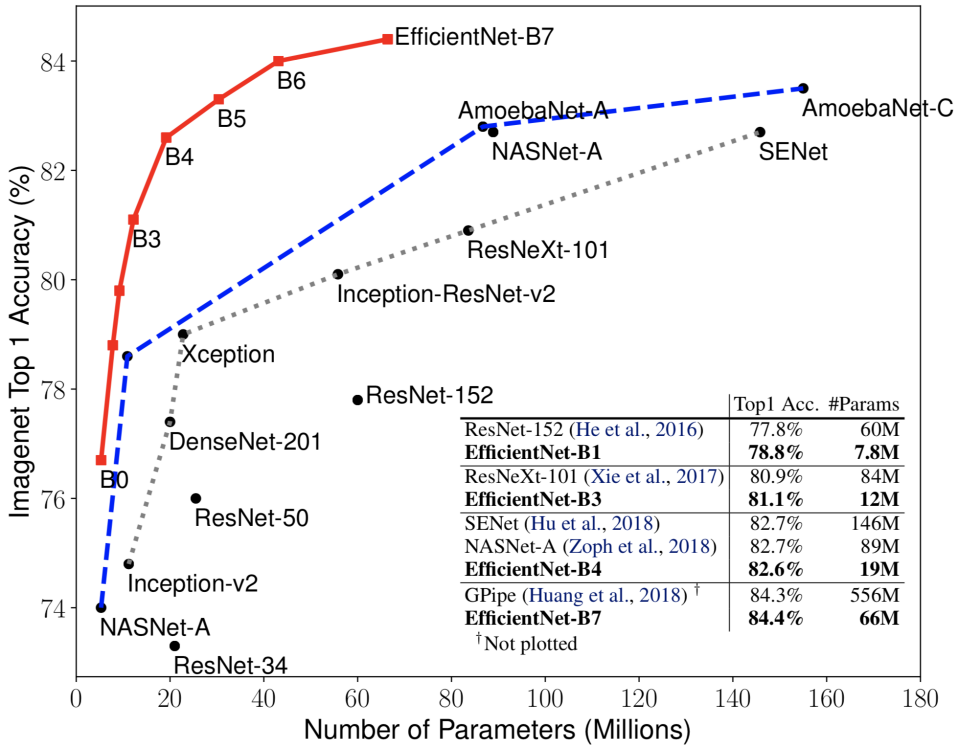


Figure 3.4: Scaling width, depth, resolution and Compound Scaling

This process continuously improves performance with more resources available. EfficientNet takes this idea a step further by providing a list of fixed coeffcients which have been trained in the imagenet dataset. Each set of coefficients has got a distinct name starting from efficient-B0 (base network). The largest network is the efficientnet-B7. As shown in the image 3.5, scaling the model further produces diminishing returns in performance.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Figure 3.5: Comparison between the EfficientNet and other state-of-the-art models in terms of performance

3.4.4 Advanced Features

EfficientNet uses a variety of advanced techniques in order to achieve state-of-the-art efficiency. The most notable of them is model compression, which ensures the model remains lightweight even when scaled. Model compression is first and foremost achieved through pruning, where the parameters which do not help improve the performance are removed. Another part of model compression is quantization, where the parameters which are usually stored in 32-bit numbers, are converted to lesser precision if deemed necessary.

3.4.5 The Adabound Optimizer

A very important factor that defines the success of a neural network model is the optimizer being used. Algorithms like Adam [4] achieve fast convergence with an element-wise scaling term on learning rates. Despite their success, they have been observed to generalize poorly compared to SGD as suggested by Pedro Savarese et al. [8]. Adabound [5] is a variant of Adam which employs dynamic bounds to smoothly transition from Adam like behaviour to SGD. This hybrid approach retains the speed of its predecessor while also resulting in better generalization capacity. Another benefit of this optimizer is that as it states, the neural network becomes less sensitive to hyperparameter changes. This is very useful in our case study as the resources that are available for training are limited, restricting us from tuning.

3.4.6 Product Matching using KNN

During the product matching process, the architectures we discussed are used to extract image embeddings. Having obtained the image embeddings, we can apply an algorithm that detects which embeddings actually represent the same product. Traditionaly, KNN variants are widely used to detect image similarity [12] [6] [1]. Thus we will adopt a similar approach and use KNN while keeping the number of neighbors low. The reasoning behind this is that there are a lot of products in the dataset and only small sets of them are identical. Consequently, the representations that are similar to each other will be a few and only a small number to none of the closest neighbors could actually be identical products.

Chapter 4

Experiments

4.1 Datasets

There is a plethora of datasets containing a variety of products available in the web. In this thesis, we are dealing with real world data, derived from an industry environment and contain products that can be found in Greek pharmacies and supermarkets. Two datasets were given in total. The first dataset contains 270k products while the second is a superset of the first with 70k additional products. Each product is accompanied by the following information: a unique id, textual information (i.e., product name, last sale), a URL pointing to the retailer's site a URL pointing to the image in the retailer's site, a site id (unique for each retailer) and 4 labels, i.e., brand id, product type, product line, and barcode. From the 4 tasks, the first three are used for categorization and the last for product matching. The images attached to each unique product ID could be more than one. The company encourages novel practices on the dataset as they have not yet implemented any system that makes use of the images as a baseline. Examples of images in the dataset can be seen in Figures 4.1, 4.2, and 4.3. Many of these instances

may contain mistakes. For instance, many retailers like to attach their store's logo to compensate for the lack of images for the product. An additional problem apart from the multitude of classes we are dealing with, is the unbalanced distribution of the products to the corresponding classes for each label as shown in figures 4.4, 4.5 and 4.6.

Figure 4.1: Product images of the same brand

Figure 4.2: Product images of the same category

Figure 4.3: Product images of the same product line

Figure 4.4: Class distribution for brand

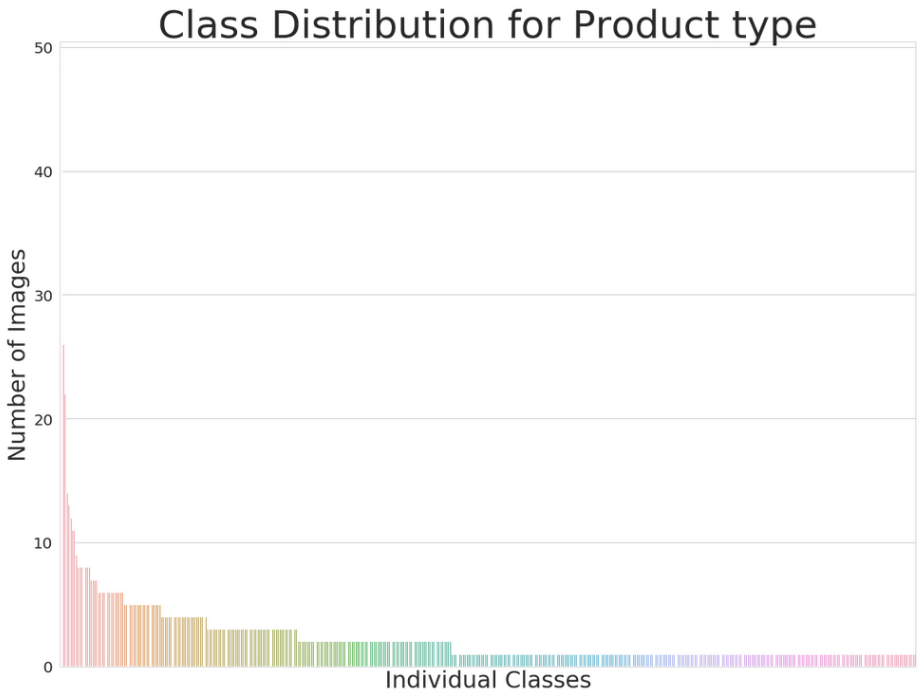


Figure 4.5: Class distribution for category

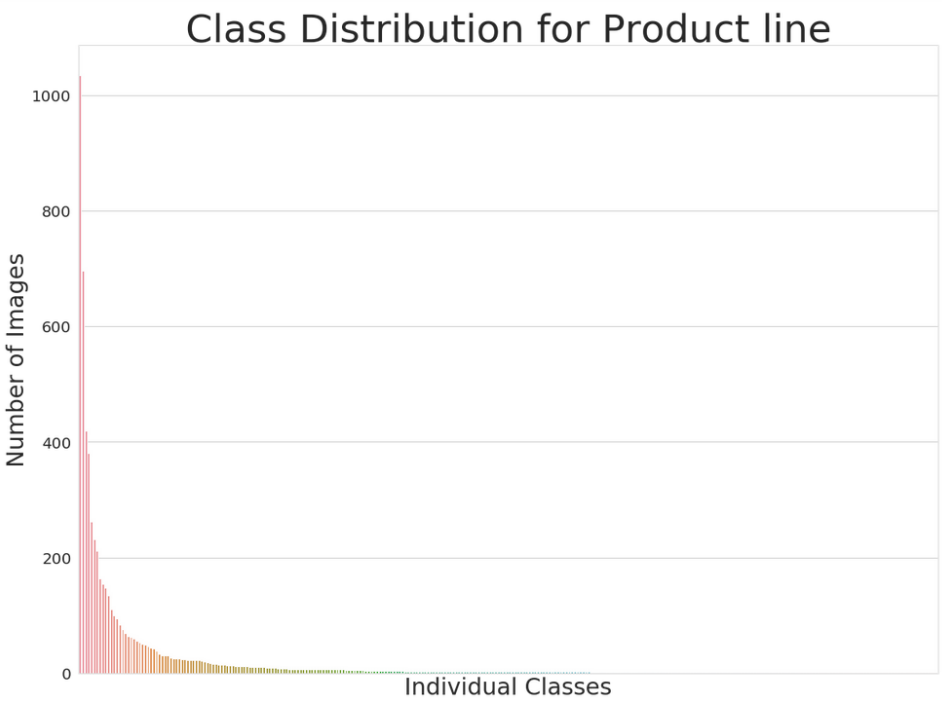


Figure 4.6: Class distribution for product line

4.1.1 Product Categorization

The EfficientNet architecture that we exhibited in the models section will be used for both versions of the dataset. The rest of the architectures mentioned in this thesis are trained in a small subset of the data available, consisting of the 20 most popular classes only. Those experiments were done exclusively for comparison purposes, in order to find the best architecture to tackle with the larger datasets. As this small dataset differs vastly from the ones discussed above and the hyperparameters used are different, the results are placed in the appendix in table A.1. From each version of the dataset we will extract a subset of the products available according to the following restrictions: each label must contain at least 10, 50 and 100 images in each class. Thus, each version will spawn three new datasets. Tables 4.1 and 4.2 show how many classes are contained in each label for each version of the dataset. The images in the first version are resized to 112x112 and are slightly shifted horizontally. The images in the second version are resized to 224x224 and no transformations are applied to them. The six datasets that have resulted are trained against three single task models for each one of the labels and a multitask model for the three labels simultaneously.

4.1.2 Product Matching

Product matching can only occur after the categorization task is finished and the models are trained. We begin by feeding the now trained models with the datasets once again, this time extracting the image embeddings produced before the softmax layer. Results from the single task models were omitted as they

Labels	At least 10	At least 50	At least 100
Brand	1779	648	387
Product type	950	728	574
Product line	2650	787	372

Table 4.1: Classes contained in each label in the first version of the dataset.

Labels	At least 10	At least 50	At least 100
Brand	1995	725	441
Product type	980	753	619
Product line	3081	998	497

Table 4.2: Classes contained in each label in the second version of the dataset.

produced results bellow 0.1 indicating that this way to represent the images is wrong. Nevertheless, the multitask model seems to encapsulate the meaning of each image adequately. The embeddings were then matched using KNN to determine which representations were close to each other. It is worth noting that at this point the restrictions of at least 10, 50 or 100 images per class only indicate the size of the dataset and not how many barcodes per class exist.

4.2 Experimental Setup

4.2.1 Product Categorization

Table 4.3 lists the hyperparameters used for the first experiment. Width and Depth coefficients are set to indicate an EfficientNet of B5 complexity. The default resolution for the specified coefficients is 224x224 but due to restrictions in computational resources during the first iteration of the experiments it was halved to 112x112. For this exact reason no early stopping mechanism was implemented and the epochs were set to a fixed 15. In figures 4.7, 4.8 and 4.9 we can observe the training process for the first version of the dataset with at least 10 images per class which show clear signs of convergence beyond the 9th epoch. The training process for the rest of the datasets in all labels can be found in the index. The batch size of choice was the largest possible that fit the GPU which was 32.

Resolution	112x112
Width coeff	1.6
Depth coeff	2.2
epochs	15
Batch size	32

Table 4.3: Hyperparameter setup for the experiments in the first version of the dataset.

Table 4.4 lists the hyperparameters used for the first experiment. Width and Depth coefficients are the same used in the previous iteration of experiments. The default resolution for the

specified coefficients (i.e. 224x224) was kept as more resources became available. The number of epochs was kept at 15 as the experiments followed a similar pattern to the previous ones and had already converged by then. The training process for the rest of the datasets can be found in the index. The batch size of choice was the largest possible that fit the GPU which was 8.

Resolution	224x224
Width coeff	1.6
Depth coeff	2.2
epochs	15
Batch size	8

Table 4.4: Hyperparameter setup for the experiments in the second version of the dataset.

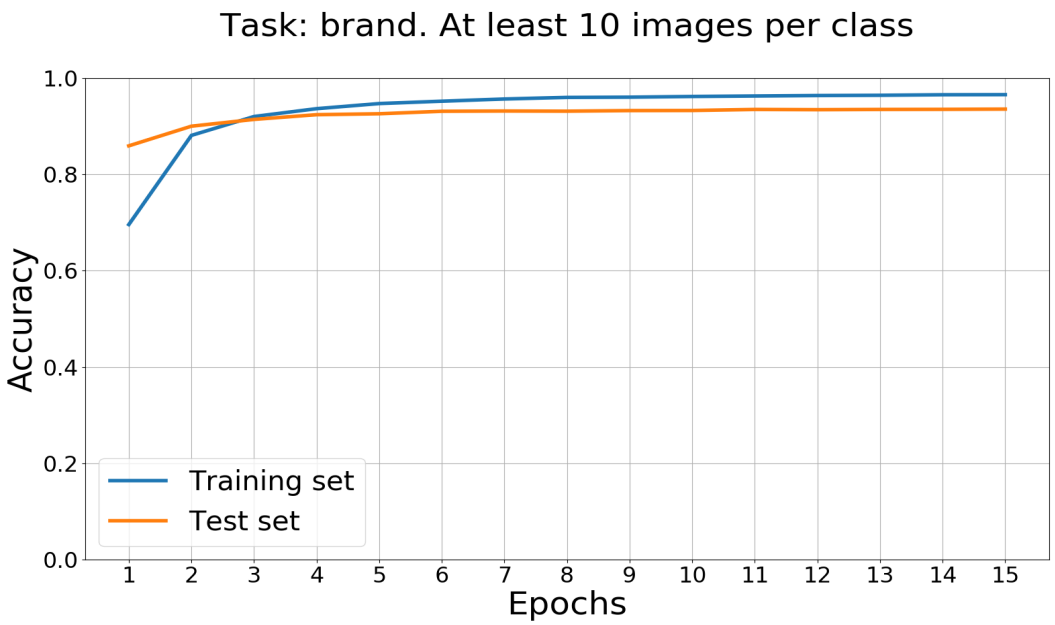


Figure 4.7: Brand, dataset 1: at least 10 elements per class

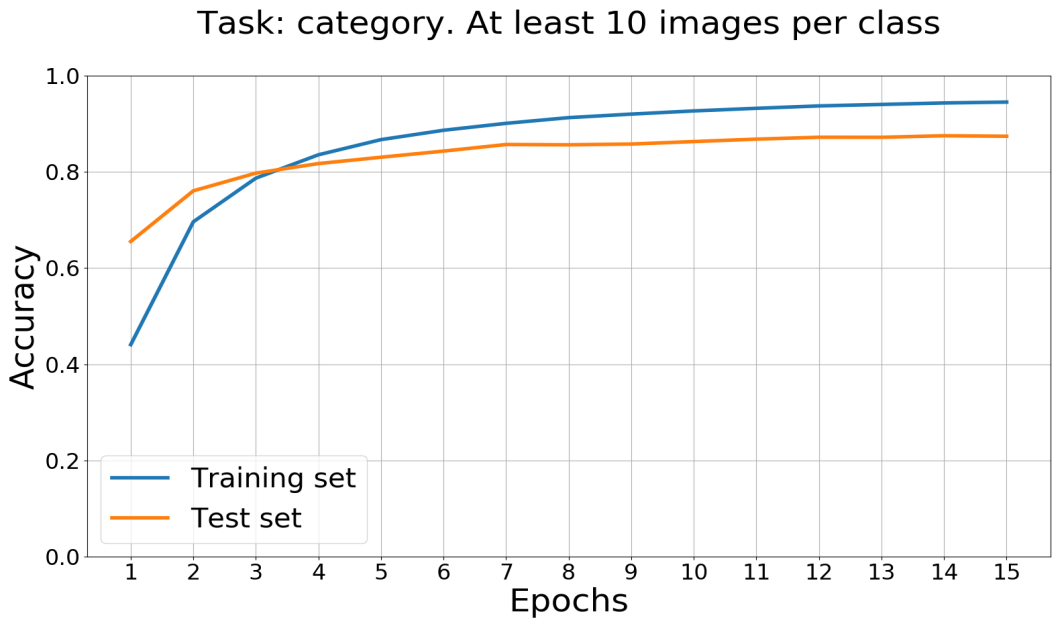


Figure 4.8: Product type, dataset 1: at least 10 elements per class

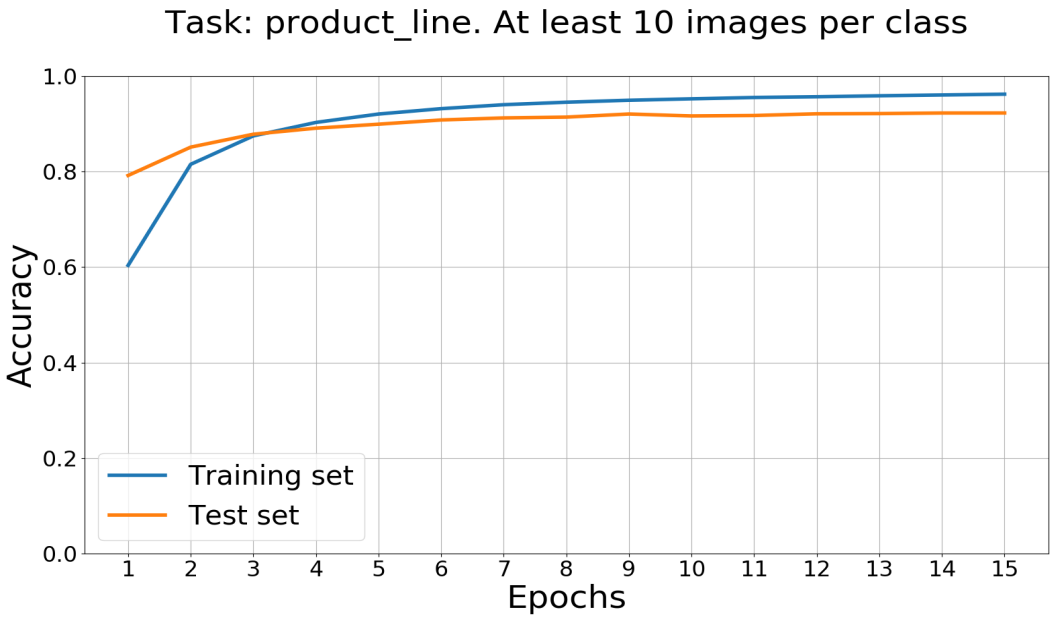


Figure 4.9: Category, dataset 1: at least 10 elements per class

4.2.2 Product Matching

Table 4.3 showcases the hyperparameters used for the experiments in product matching. The number of neighbors used was 1 and 2. Increasing the number of neighbors further does not only increase computational costs exponentially which is prohibited given the resources available but also yields worse performance as we will see later in the results section.

Number of Neighbors	1, 2
Distance Metric	Euclidean

Table 4.5: Hyperparameter setup for the experiments in product matching.

4.3 Results

4.3.1 Product Categorization

Tables 4.6 and 4.7 contain the results for the two datasets in the categorization task. Similar results produced from the multitask model can be found in the appendix B.1 with table B.1 containing the results for the first dataset and table B.2 containing the results for the second. As we restrict more the number of images per class, the performance in all labels is increasing. Brand and Product line outperform Product type. This is to be expected as the images in the product type classes vary drastically from each other in color mostly and less in shape, making the categorization task difficult. The second dataset outperforms the first as there are slightly more classes in each label but also more images per class. Results from models other than EfficientNet will not be displayed as they are produced from a far smaller subset and are not directly comparable to this. Concluding, more images per class will help the model perform better until a certain threshold. Afterwards, other factors such as damaged images come into play prohibiting better performance without changing the preprocessing policy.

Labels	At least 10	At least 50	At least 100
Brand	0.9378	0.9448	0.9541
Product type	0.8792	0.9210	0.9366
Product line	0.9250	0.9456	0.9531

Table 4.6: Results for the categorization task in the first version of the dataset.

Labels	At least 10	At least 50	At least 100
Brand	0.9462	0.9568	0.9648
Product type	0.8969	0.9337	0.9511
Product line	0.9323	0.9535	0.9596

Table 4.7: Results for the categorization task in the second version of the dataset.

4.3.2 Product Matching

Tables 4.8 and 4.9 display the results for the two datasets in the product matching task. In both datasets, a dramatic decrease in performance occurs as we move from 1-NN to 2-NN. This result does not seem unnatural as the majority of the products in the datasets contain zero or one identical images. So, in most cases, trying to obtain information from multiple neighbors can lead the algorithm astray. In the end, a simple distance metric like Euclidean can suffice as the representations of the images of identical products will likely be very close to each other. Of course, the three labels describe the images to a degree but also

have their own limitations. As shown in the image 4.10 there are products that possess the same labels but different barcodes. Such images will not be matched properly. Nevertheless the results are deemed as good given that we did not make use of text for the difficult task of product matching.

Neighbors	At least 10	At least 50	At least 100
1-NN	0.7521	0.7422	0.7456
2-NN	0.4952	0.4898	0.4913

Table 4.8: Results for the matching task in the first version of the dataset.

Neighbors	At least 10	At least 50	At least 100
1-NN	0.7516	0.7440	0.7458
2-NN	0.5038	0.4990	0.5002

Table 4.9: Results for the matching task in the second version of the dataset.

Chapter 5

Conclusions and future work

5.1 Conclusions

In this thesis we proposed a system that employs state of the art techniques in order to achieve good results for product categorization and matching. In order to reach our end goal which is product matching, we must first categorize our data in a set of labels that best represent the products. The architecture of choice for the categorization task was EfficientNet B5. The categorization procedure, produced great results in all labels with brand and product line producing higher accuracy than product type.

The models used for product categorization were then reused to extract embedding from the images. Some variations of the KNN were tried in order to find the matching products given their representations and the 1-NN with euclidean distance vastly outperformed the rest. Certain restrictions in the representational capacity of the product images, imposed by the labels, led to worse results with regards to product matching. This is to be expected but we still managed to surpass by a large margin the baseline set by the cooperating company.

5.2 Future work

5.2.1 Data Enrichment

An assumption done at the very start of the thesis is that the 3 labels (brand, category, product line) are describing the products adequately. However, we later found out that there are a lot of instances where the 3 labels are the same between 2 products but the barcode is different. Thus, those models will produce similar embeddings from those product images without them actually being similar. If we add more labels to the products and manage to achieve a richer representation for each image, product matching performance is likely to increase.

5.2.2 Zero shot learning

Zero-Shot learning [13] aims to classify objects to classes not previously seen during training. In an industry environment, new products are always added to the dataset, and some of them are never seen before. This principle is also reflected in the given dataset. Zero-shot learning allows us to recognise objects that are not seen before. This process is using image embeddings, which are the embeddings obtained from the images to compare them with class embeddings. Class embedding is used to represent a general class and is usually extracted from the set of training images existing in each class. If an image of the test set, differs greatly from every class embedding, then the model will deem it as a new entry.

5.2.3 One shot learning and Siamese Networks

Modern Image categorization processes require a very large amount of data in order to predict accurately, One shot learning [11] tries to tone down this need for more data. More specifically, it is an object categorization problem where the model tries to learn information about object categories while having one or a few training images. The implementation of one shot learning is done through a Siamese Network. This network has got two identical fully connected CNNs with same weights and accepting two different images. Both networks produce a 128 dimensional embedding of the image. Those embeddings then pass from the last layer that connects the two networks and calculates a distance metric between the two. The result is, that instead of the usual probabilities obtained from a softmax, Siamese Networks produce a number between 0 and 1 that describes the similarity between the two images being given. This approach fully utilizes the low number of images per barcode and tackles the problem from a different perspective.

5.2.4 Tuning

As stated before, the project was ran in limited resources and as a result certain compromises were made mostly in terms of performance. First of all, there was not a possibility for proper tuning due to the deadline of the thesis. Thus, further exhaustive search can be done in the hyperparameter space. During the experiments we found out that the model performs better when the images are not subject to heavy transformations. However, there is still an increase in performance to be found by tuning the image generator using techniques such as AutoAugment. As

for the models used, efficientnet B5 was prefered. While it produces the best performance for the FLOPs needed, there are far more complex models which can further increase the accuracy in the dataset. Concluding, this is only possible with the acquisition of more resources and the current setup provides a very good performance for its computational complexity.

Appendices

Appendix A

Architecture comparison summary

The following table summarizes the comparisons done in order to determine the best performing neural network architecture for our categorization problem. It is important to note that the experiments were done in a very small portion of the dataset with only the 20 most popular classes for each label. Also, the hyperparameters used are different from the ones in the main experiments.

Architectures	Brand	Product Type	Product Line
DenseNet	0.833	0.702	0.808
Inception-ResNet-v2	0.841	0.732	0.836
NasNet	0.823	0.730	0.811
EfficientNet B3	0.840	0.724	0.825
EfficientNet B5	0.865	0.738	0.862

Table A.1: Comparison summary.

Appendix B

Training figures for both datasets

Bellow are displayed all the figures that resulted from the training process in the categorization task.

Dataset version 1

Training process for Brand

Figure B.1: Brand, dataset 1: at least 50 elements per class

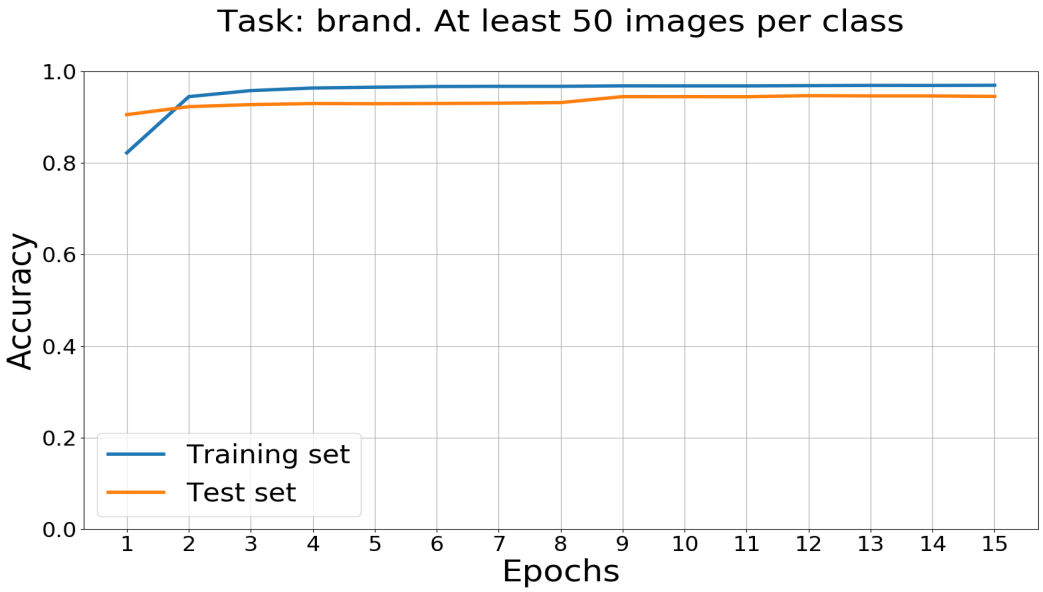


Figure B.2: Brand, dataset 1: at least 50 elements per class

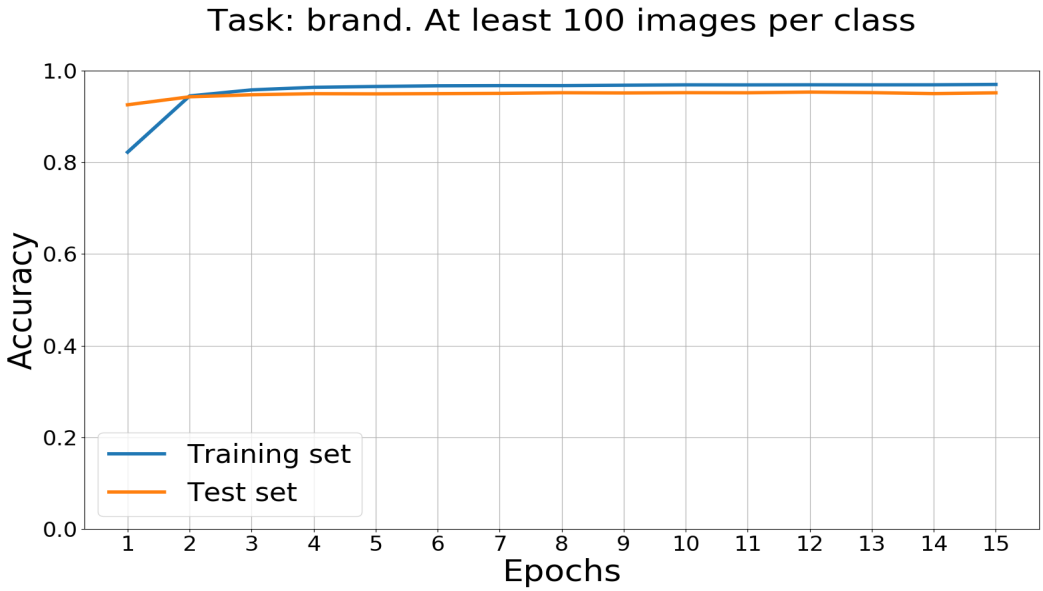


Figure B.3: Brand, dataset 1: at least 100 elements per class

Training process for Product type

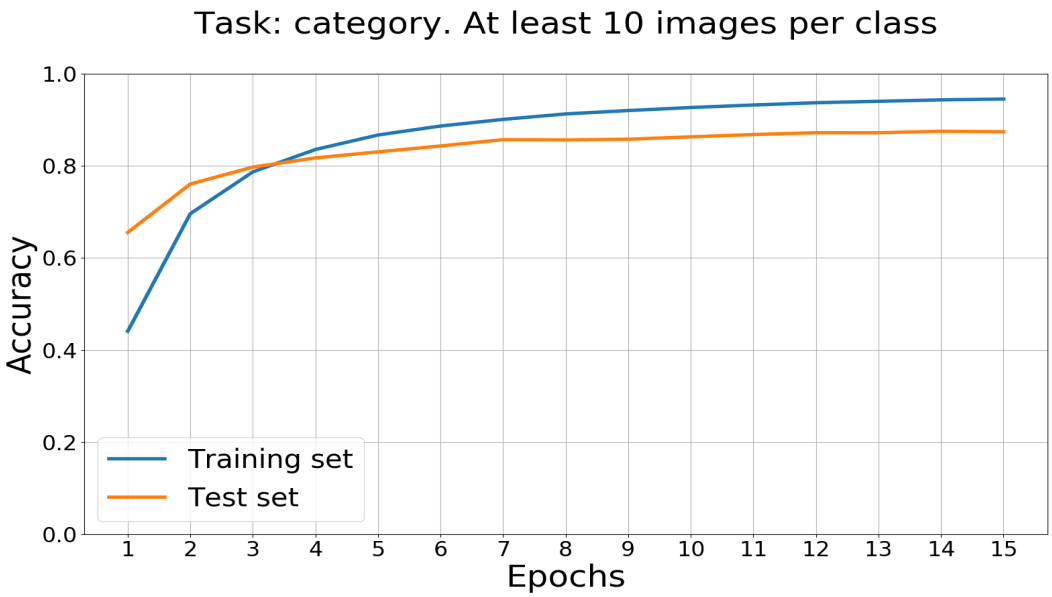


Figure B.4: Product Type, dataset 1: at least 50 elements per class

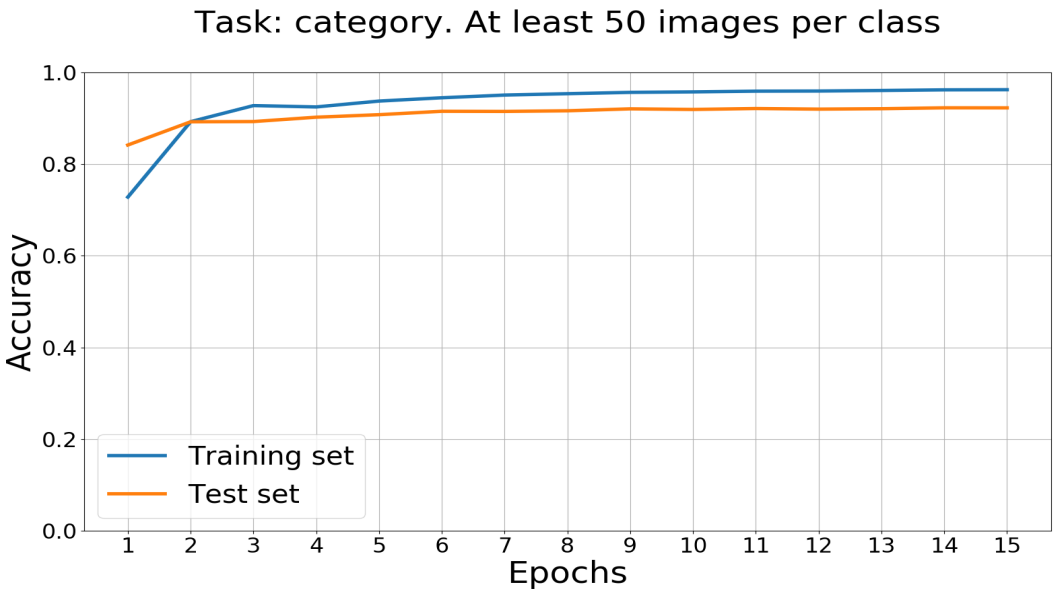


Figure B.5: Product Type, dataset 1: at least 50 elements per class

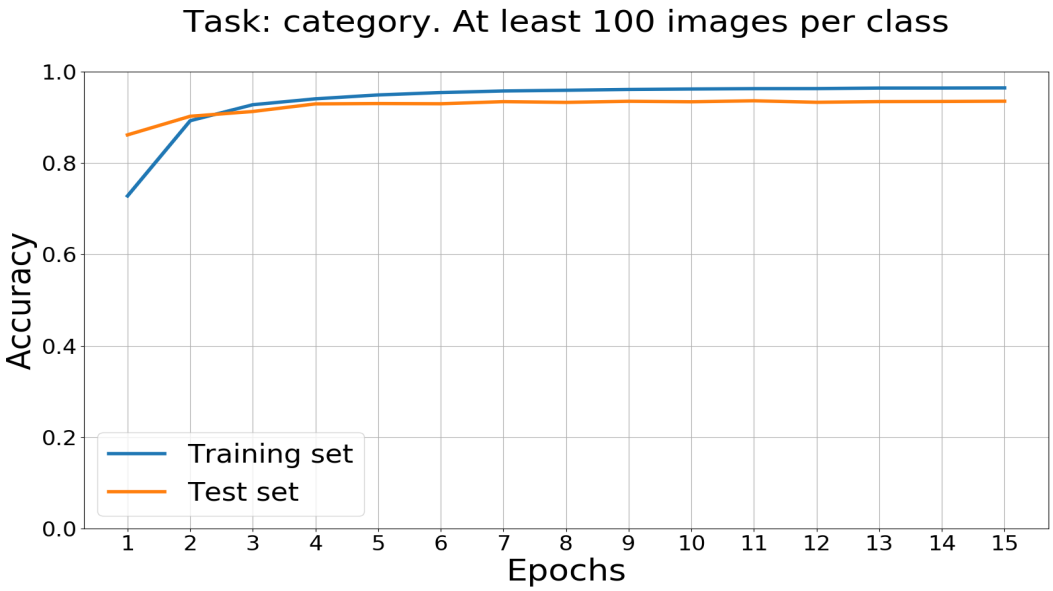


Figure B.6: Product Type, dataset 1: at least 100 elements per class

Training process for Product Line

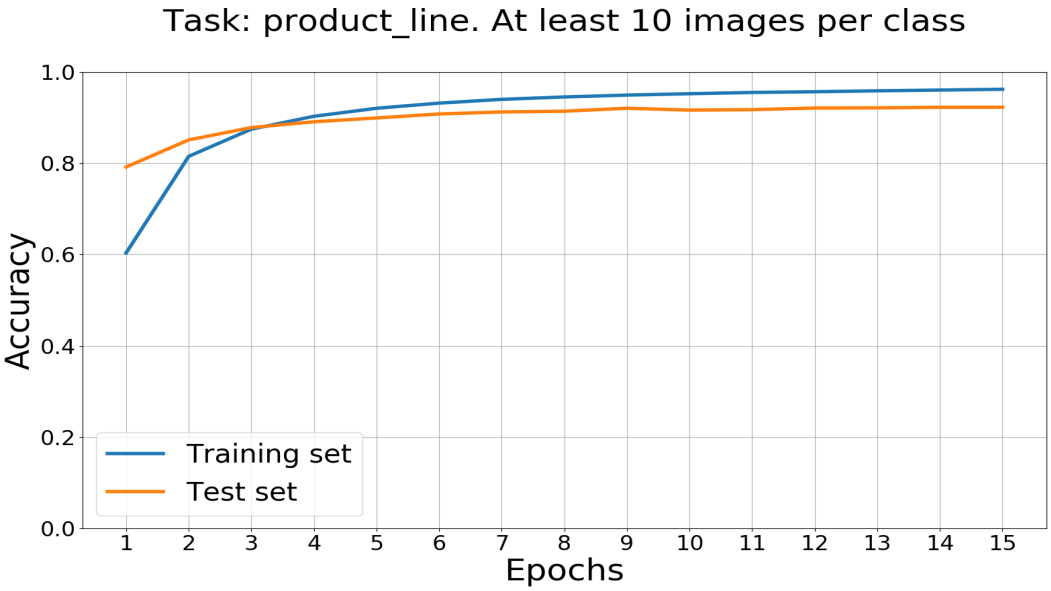


Figure B.7: Product Line, dataset 1: at least 10 elements per class

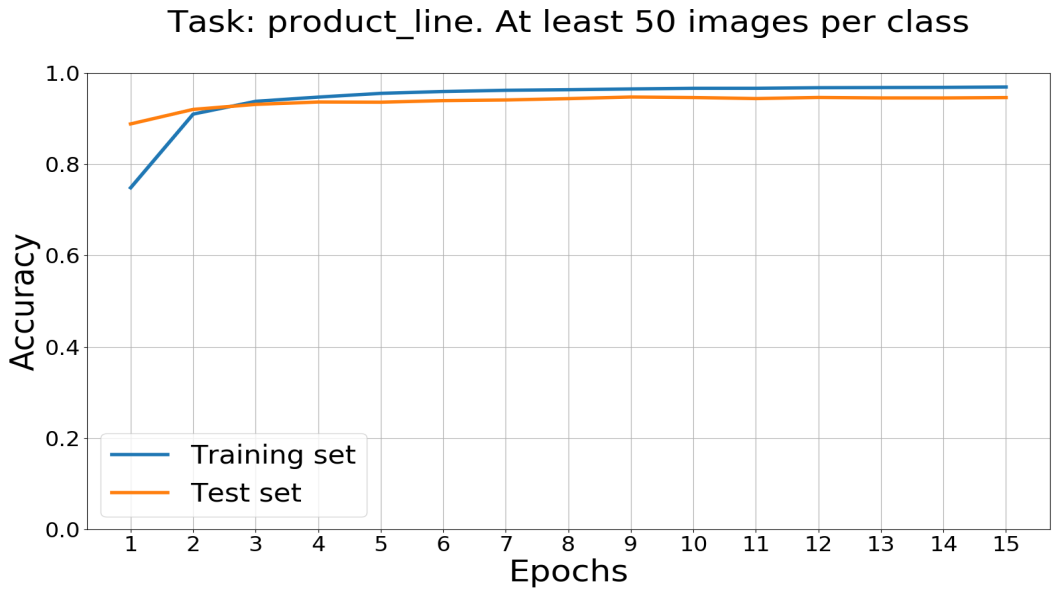


Figure B.8: Product Line, dataset 1: at least 50 elements per class

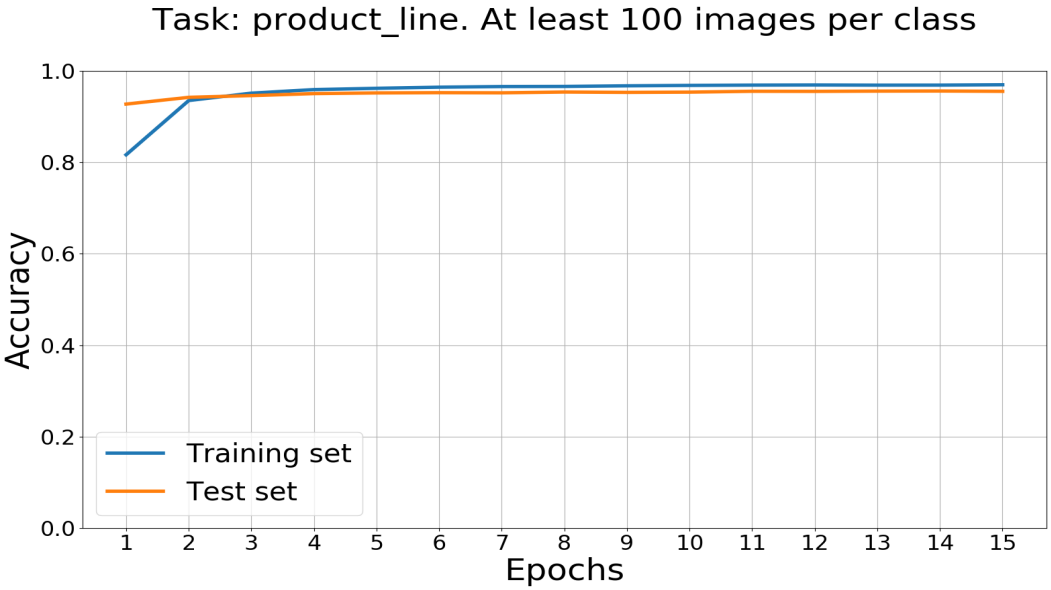


Figure B.9: Product Line, dataset 1: at least 100 elements per class

Dataset version 2

Training process for Brand

Figure B.10: Brand, dataset 2: at least 10 elements per class

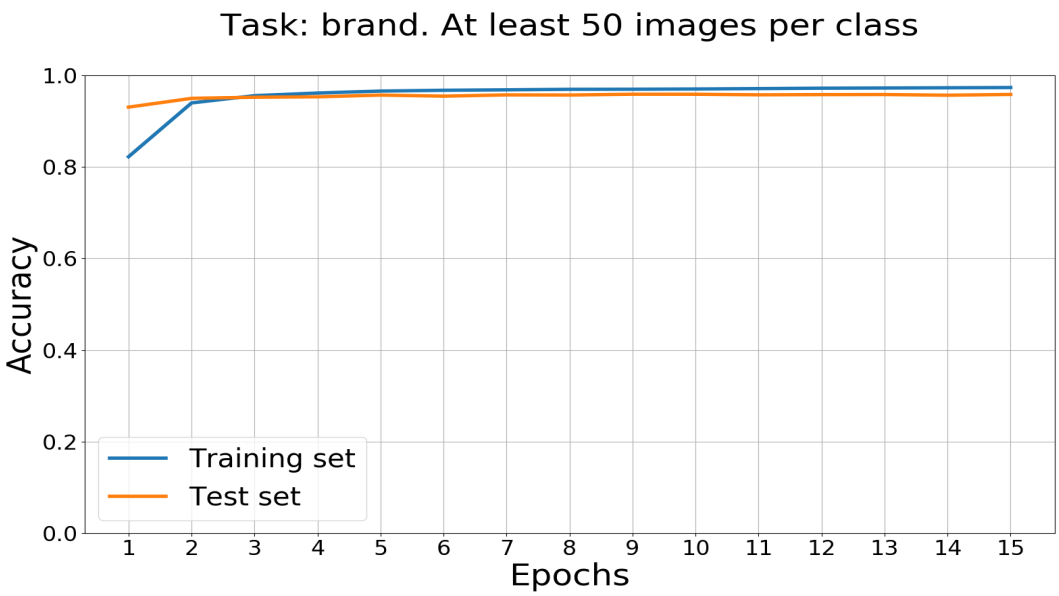


Figure B.11: Brand, dataset 2: at least 50 elements per class

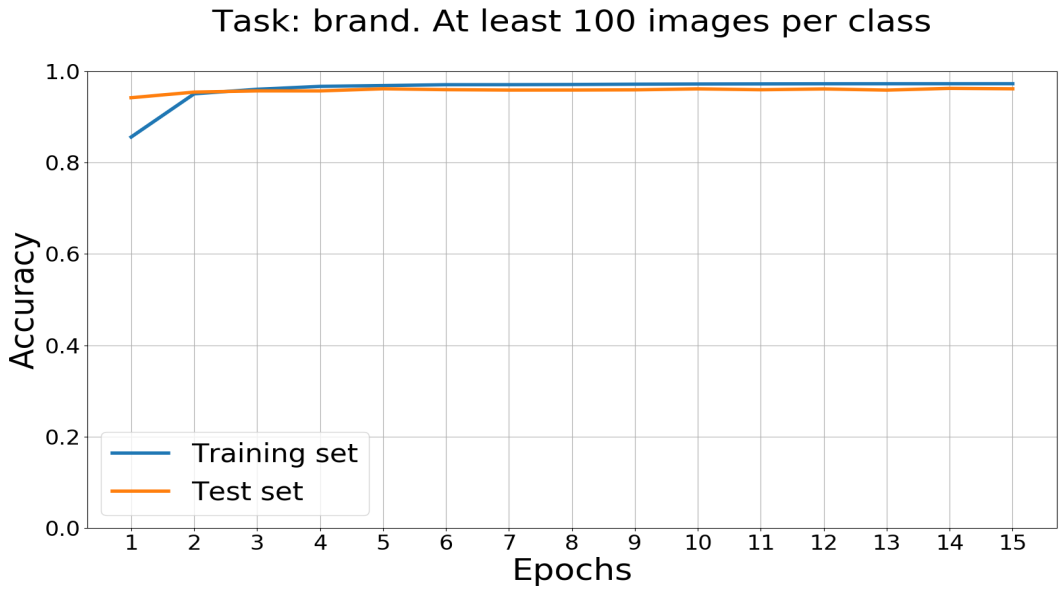


Figure B.12: Brand, dataset 2: at least 100 elements per class

Training process for Product type

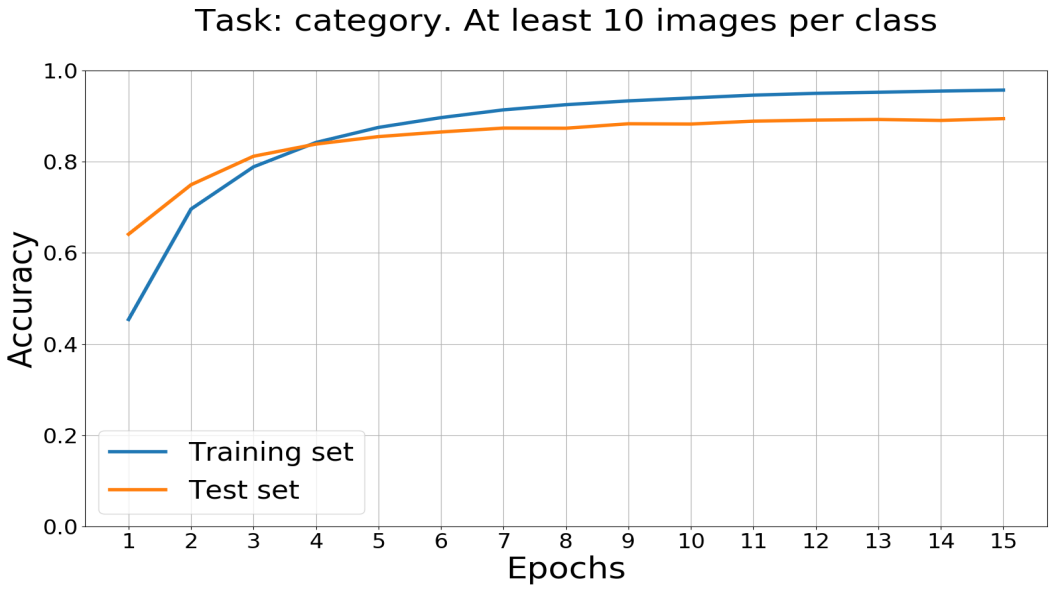


Figure B.13: Product Type, dataset 2: at least 10 elements per class

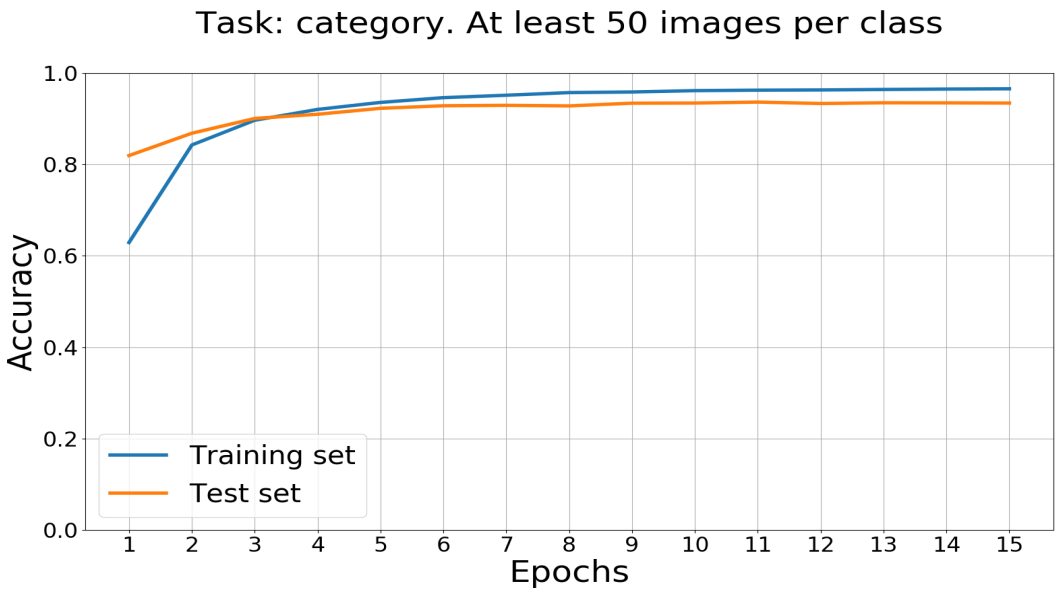


Figure B.14: Product Type, dataset 2: at least 50 elements per class

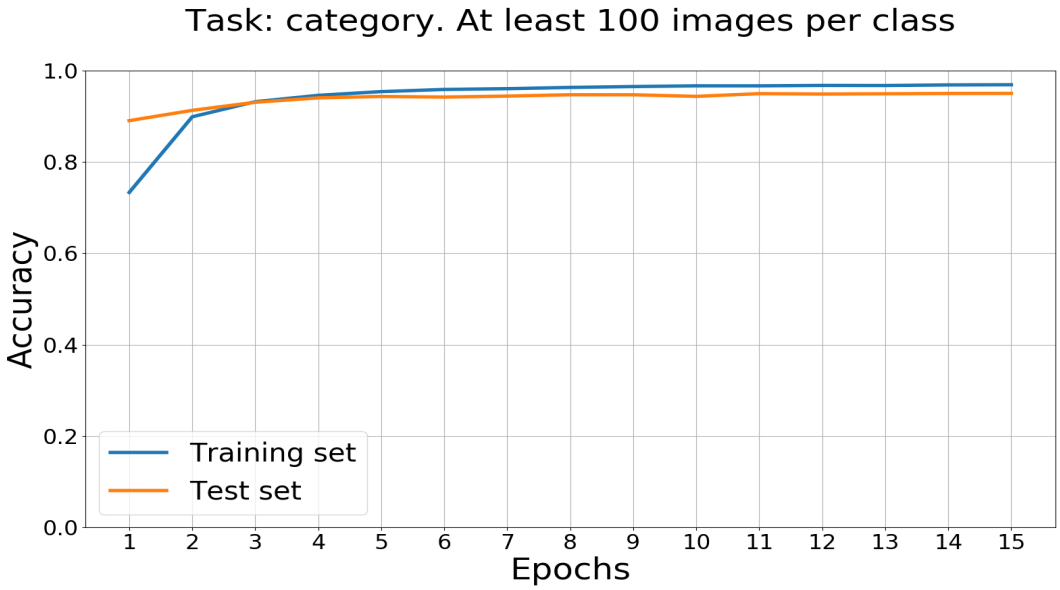


Figure B.15: Product Type, dataset 2: at least 100 elements per class

Training process for Product Line

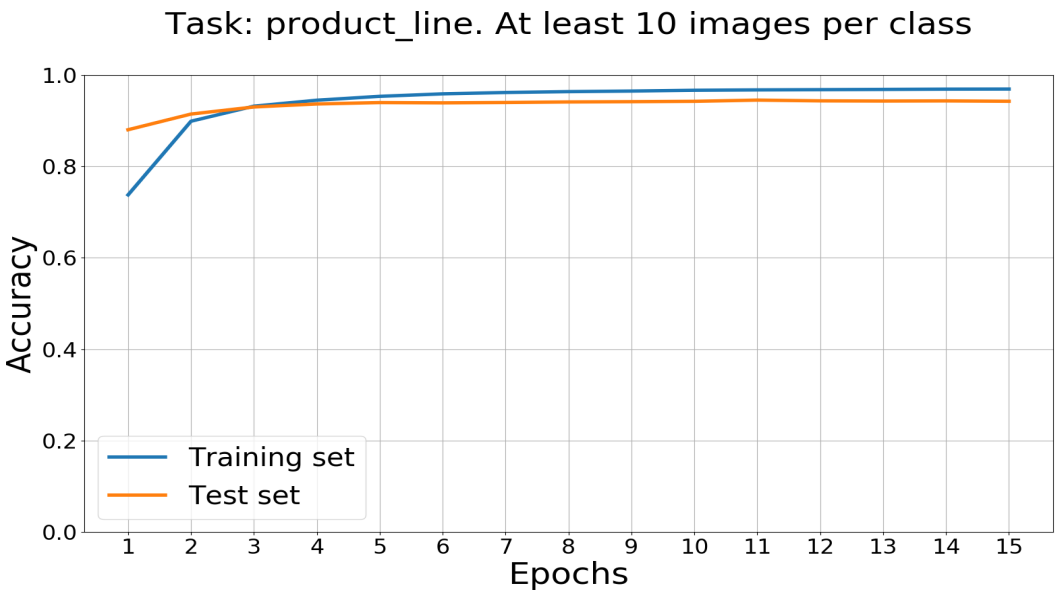


Figure B.16: Product Line, dataset 2: at least 10 elements per class

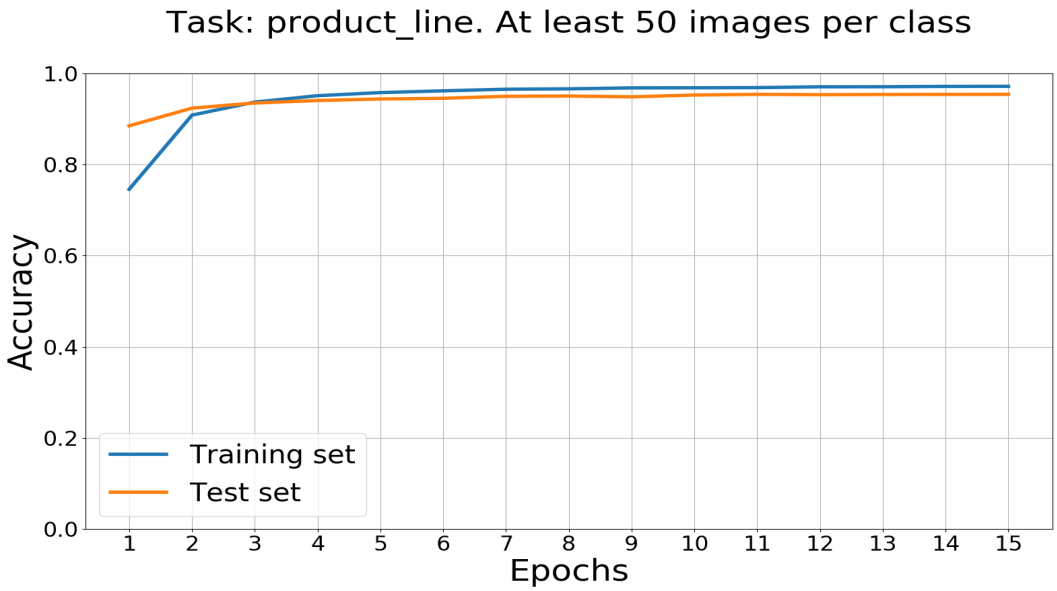


Figure B.17: Product Line, dataset 2: at least 50 elements per class

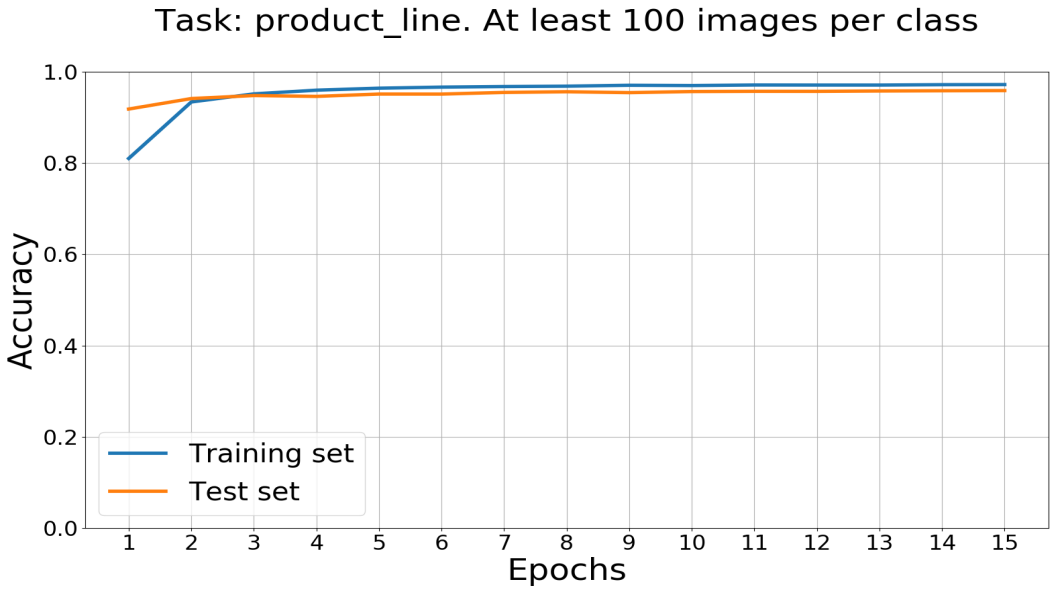


Figure B.18: Product Line, dataset 2: at least 100 elements per class

B.1 Results for the categorization process of the multitask model

The tables bellow display the performance of the multitask model for both versions of the dataset.

Labels	At least 10	At least 50	At least 100
Brand	0.9255	0.9328	0.9398
Product type	0.8850	0.9190	0.9301
Product line	0.9238	0.9342	0.9434

Table B.1: Results for the categorization task in the first version of the dataset for the multitask model.

Labels	At least 10	At least 50	At least 100
Brand	0.9405	0.9460	0.9576
Product type	0.9112	0.9294	0.9481
Product line	0.9303	0.9384	0.9447

Table B.2: Results for the categorization task in the second version of the dataset for the multitask model.

Bibliography

[1] Giuseppe Amato and Fabrizio Falchi. "kNN based image classification relying on local feature similarity". In: *Proceedings of the Third International Conference on SImilarity Search and APplications*. ACM. 2010, pp. 101–108.

[2] Forrest Iandola et al. "Densenet: Implementing efficient convnet descriptor pyramids". In: *arXiv preprint arXiv:1404.1869* (2014).

[3] Anitha Kannan et al. "Improving product classification using images". In: *2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, pp. 310–319.

[4] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[5] Liangchen Luo et al. "Adaptive gradient methods with dynamic bound of learning rate". In: *arXiv preprint arXiv:1902.09843* (2019).

[6] Xiaoning Qian et al. "Optimal embedding for shape indexing in medical image databases". In: *Medical image analysis* 14.3 (2010), pp. 243–254.

[7] Petar Ristoski et al. "A machine learning approach for product matching and categorization". In: *Semantic web* Preprint (2018), pp. 1–22.

[8] Pedro Savarese. "On the Convergence of AdaBound and its Connection to SGD". In: *arXiv preprint arXiv:1908.04457* (2019).

[9] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[10] Mingxing Tan and Quoc V Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *arXiv preprint arXiv:1905.11946* (2019).

[11] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems*. 2016, pp. 3630–3638.

[12] Fang Wang et al. “Improving image distance metric learning by embedding semantic relations”. In: *Pacific-Rim Conference on Multimedia*. Springer. 2012, pp. 424–434.

[13] Yongqin Xian et al. “Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly”. In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[14] Tom Zahavy et al. “Is a picture worth a thousand words? A deep multi-modal architecture for product classification in e-commerce”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[15] Barret Zoph et al. “Learning transferable architectures for scalable image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8697–8710.