U. PORTO

FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

Visual Similarity with Deep Triplet Quantization: Application to the fashion industry

by

Pedro Miguel Vieira Esmeriz Pereira

Dissertation for achieving the degree of

Master in Data Analytics

Dissertation supervisor: Professor João Manuel Portela da Gama

September 2020

“ O que a consciência sabe, o inconsciente não sente”

Resumo

A pesquisa visual tem sido um assunto crescente no comércio eletrónico, onde as plataformas líderes têm vindo a fornecer vários mecanismos para pesquisar um item por meio de uma imagem enviada pelo utilizador. Através de modelos de *deep learning*, as empresas têm vindo a fornecer recomendações precisas para imagens de catálogo, mas apenas algumas foram capazes de lidar com imagens de consulta com ruído de fundo e padrões de qualidade mais baixos, tipicamente obtidas via smartphone. No entanto, para uma base de dados na escala de milhões, isso envolve o uso de infraestrutura distribuída e dispendiosa, frequentemente suportada por unidades de processamento gráfico para acelerar a pesquisa e fornecer resultados abaixo de um segundo. Isso implica um forte investimento das empresas, sendo apenas acessível a grandes empresas.

Estas duas questões têm sido estudadas há muito tempo por investigadores, dando origem a estruturas híbridas que combinam redes neuronais e técnicas de hashing. Esses métodos têm mostrado o potencial de agregar funções de discretização com os procedimentos de treino naturais de uma rede neural num modelo eficiente, mas eficaz, capaz de fornecer recomendações baseadas numa base de dados de grande escala.

Nesta linha de trabalho, o seguinte projeto, realizado em parceria com a empresa ShopAI, tem como foco o estudo de redes neuronais convolucionais com treino induzido por *triplets* com camadas de quantização adicionais treinadas com o objetivo de criar um motor de pesquisa visual para a indústria da moda. Neste trabalho, um novo mecanismo de rotulagem de hierarquia foi criado para promover uma criação automatizada, porém detalhada, de *triplets*. No que diz respeito à implementação da rede neural, este projeto foi construído sobre versão da rede Xception do ImageNet, combinada com as técnicas de quantização definidas por B. Liu et al. (2019) em DTQ.

O modelo treinado foi então analisado por meio (i) da sua precisão média com base no método de rotulagem de hierarquia, (ii) de um caso de uso prático com equipas de marcas de moda para análise de similaridade e (iii) testes A / B com um modelo Xception não quantizado e um sistema de recomendação baseado em regras de associação. Em todos eles, mostra-se que o modelo criado foi capaz de produzir resultados de qualidade com erro marginal de precisão nos seus resultados, mantendo um alto grau de eficiência.

Abstract

Visual search has been a growing subject in the ecommerce business where high leading platforms are providing several mechanisms to search an item via an user submitted image. Through deep learning models, companies have been able to provide accurate recommendations for In Shop images, but only a few have been able to address query images with background noise and lower quality standards, typically taken via smartphone. Nonetheless, for a million scale database, this involves the usage of costly distributed infrastructure, often supported by graphical processing units to accelerate the search and provide results in sub second time. This implies a heavy investment from companies, proving to be a feature accessible only to large scale players in the fashion industry, capable of supporting such investment.

These two issues have long been studied by researchers, giving the rise of end-to-end deep neural hashing networks, combining both deep neural networks and hashing techniques. These methods have shown the potential of merging learning to hash functions with the training procedures of a neural network into an efficient, yet effective, model capable of delivering recommendations on a large scale database.

In this line of work, the following project, performed in partnership with the company ShopAI, focuses on the study of convolutional neural networks with triplet induced training with additional quantization layers trained for the purpose of creating a visual search engine for the apparel fashion industry. In this work, a new hierarchy labelling mechanism was created to promote an automated yet detailed creation of triplets. Concerning the implementation of the neural network, this project was built upon the ImageNet pre-trained version of Xception, merged with the quantization techniques defined by B. Liu et al. (2019) in DTQ.

The model trained was then analysed via (i) its mean average precision based on the hierarchy labelling method, (ii) a practical use case with fashion retailers teams for similarity analysis and (iii) A/B Testing with a non quantized Xception model and a association rule-based recommender system. In all of these, it is shown that the model created was able to produce high quality results with marginal accuracy error on its results, while maintaining a high degree of efficiency.

Acknowledgements

First, this project would not have been possible without the people at ShopAI. Every person on our team provided an immense will to motivate and support me in this endeavour, thus I express my gratitude to all of them as a family.

I would like to thank my supervisor, Professor João Manuel Portela da Gama for all the counselling, comprehension and patience regarding the elaboration of this dissertation. To all my friends, a huge thank you for all the experiences and moments that we have lived and for their friendship in the good times and in the bad times. A particular note to my friend in arms Milka, for the constant and unwavering support and presence during highs and lows in my life.

My gratitude to Raquel, the Outlier, for her heartfelt companionship in this journey and whose wholesome and unwavering altruism and dedication has no match.

Lastly, all of what I am today I owe it to my family. To my mother, the most passionate and persistence person I know, who never knows how to quit. To my father, for teaching me the value of respect, honesty and humbleness. To my sister, the truest companion in my life and the person that has shared the most with me. My deepest gratitude to all of you for everything you have given me.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This section contains the introductory segments of the proposed dissertation, specifically the motivation for the theme, the main objectives for the construction of the visual search engine, the methodology to be used and the problem description to be solved.

1.1 Motivation

Over the years there has been an accelerated emergence of e-commerce companies due to the democratization and easier access to the internet, allowing users to quickly browse and search their needs regardless of their location and time of day. While this emphasizes the opportunity of creating a business reachable through the internet, it also enables other competitors to easily compete in the same market due to low barriers to entry. In order to prevail in this space, companies are forced to develop strategies and user interface tools that enhance the customer experience, aiding the end consumer to find the desired products or services and thus creating more revenue (Goswami et al., 2011).

On the fashion e-commerce segment this issue is of major concern (i) since the products are aesthetical in nature, thus must be appealing, (ii) the shorter attention span of the customer due to high supply in this field, and (iii) the retention of attention by the customer is mainly visually driven. Given the visual focus inherent to this environment, several image-based solutions have emerged where the focus relies on providing recommendations to the user based on visual similarity (Jing et al., 2015a). Moreover, companies are currently implementing image search tools where the user can submit a photo and look for items similar to it (Shankar et al., 2017; Y. Zhang et al., 2018).

In these approaches, the concept being the functionality involves the usage of deep learning models to translate an image into a numerical array and posteriorly determine the most similar products to that image. These models tend to be computationally intensive due to (i) the high complexity intrinsic of neural network and (ii) the high dimensional embeddings required for this calculation. Therefore, the current focus of research in this area has been on the conception of a model that can accurately provide visually similar results with minimal infrastructure costs involved (Zhai et al., 2019).

1.2 Objectives

Having convolutional neural networks as the cornerstone, the primary objective of this work is to create a deep learning model capable of producing accurate representation embeddings from images with a low memory footprint at the real-time response. The approach employed involves (i) the selection of a convolutional neural network architecture for feature extraction, (ii) defining a similarity strategy based on that model and (iii) implement a hashing procedure within the model to produce compact binary codes. The business implication of this work is the possibility of creating an image-based search tool scalable in practice regardless of the catalog size of an e-retailer. This model will be trained and tested against the fashion items data of a Portuguese company.

As a secondary objective, this work will provide an ablation study where several convolutional neural networks, similarity strategies and hashing procedures will be benchmarked against each other.

1.3 Questions of study

This work will combine multiple deep learning architectures for visual feature encoding that will later be used for visual similarity calculation. This structure will allow answering the following questions:

- What are the most appropriate CNN architectures for image feature extraction?
- What is the most effective strategy for similarity search?
- What is the most effective dimensionality reduction process to apply?

2 Literature review

This chapter begins with an exposition on the evolution of deep learning technologies over the years and the main components that a convolutional neural network employs in contrast with other artificial neural networks. Moreover, the most commonly used architectures on image classification are also mentioned. In this section, the reader can find the definition of each of these architectures in addition to their main differences.

The following section discloses the main structures used to tackle the visual similarity search problem. These include the use of classification-based neural networks, siamese neural networks and triplet neural networks - these last two involving the parallel use of more than one common CNN.

Lastly, there is a review of several methods that can be applied for dimensionality reduction. These can be divided into three groups: (i) the traditional and most applied methods, (ii) the rise of quantization hashing procedures in efficient similarity search and (iii) the creation and training of quantization functions as an integral part of a convolutional neural network.

2.1 Convolutional Neural Networks

Convolutional neural networks, being a branch of deep learning, use a deep graph structure with several processing layers responsible for multiple linear and non-linear transformations. These, among other deep architectures, have been achieving state-of-the-art results on fields ranging from visual recognition and classification to object detection on a variety of fields (Krizhevsky et al., 2017). The image processing in a CNN is performed in a sequential process where the first layers learn the low-level features like edges and the deeper layers learn higher-level ones, like patterns (LeCun et al., 2015).

2.1.1 Components

The difference of CNNs in comparison with other artificial neural networks is that the former makes use of specific layers that enhance the analysis of an image. In these layers, operations like convolution, pooling and several activation processes are used to better adjust the parameters during the optimization-oriented learning of the neural network in order to perform some visually related tasks (LeCun et al., 2015).

Convolution layer

The convolution layer is the main component of a CNN and its role is to reduce the images into a form that enables a faster and easier process without losing essential features. A convolution consists of calculating the element-wise multiplication between the image matrix and a filter matrix, also denominated as the kernel, and sum the results to get the value that forms a single element of the output matrix. This is performed iteratively, where the filter will be applied to several submatrices of the input matrix. This procedure has as parameters the size of the filter and the stride, this last responsible for determining how many positions should the filter be shifted after each iteration.

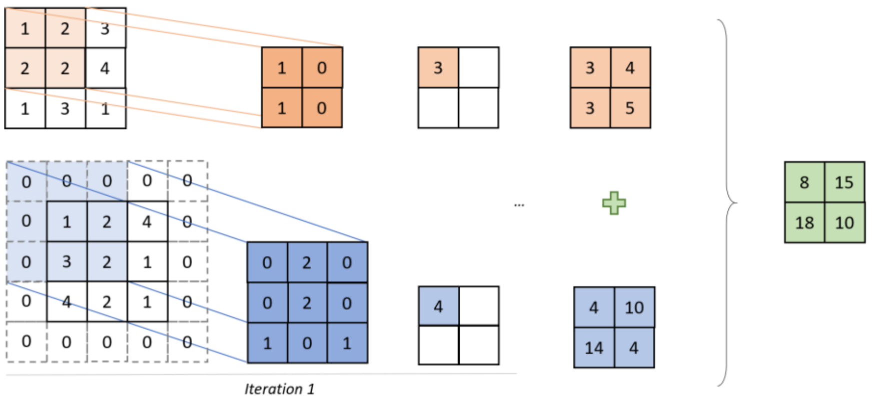

Figure 1 – Convolution with different strides and padding

In Figure 1 there are two different examples of convolutions being presented. The orange example shows a matrix of 3x3 where a convolution was performed with a kernel of size 2x2 with stride 1x1, meaning that the filter will shift only 1 cell at a time, both horizontally and vertically. The blue example shows a filter with size 3x3 and stride 2x2. Moreover, in this example, there is also padding applied to the matrix, in which case the input matrix is considered as a 5x5 matrix with the additional cells having value 0. This is mainly used to maintain the original matrix size after convolution and should consider the stride size used. A convolution could also be applied to several matrices in conjunction as, for example, when processing the RGB representation of an image. In this case, convolution is performed on all 3 matrixes and the output matrices are summed into one, resulting in the green example. The values of the filters are learned across the training of the neural network in order to best detect dimensions as color, texture, and shape, among others, by optimizing the values through the backpropagation method (Wan et al., 2014).

Pooling layer

The pooling layer is similar to the convolutional layer but, in this case, the purpose is solely to reduce the spatial size of the convolved features to decrease the computational requirements to further process the data. The concept behind pooling is to extract the dominant features from the previous layers, while acting a noise suppressant of low importance values (LeCun et al., 2015). In order to perform this operation, there is also a kernel that is applied to an input matrix with a predefined size, stride and padding just like the convolution layer. In this case, the kernel is not a matrix of values but rather a selection matrix where a specific strategy is employed to extract the dominant features.

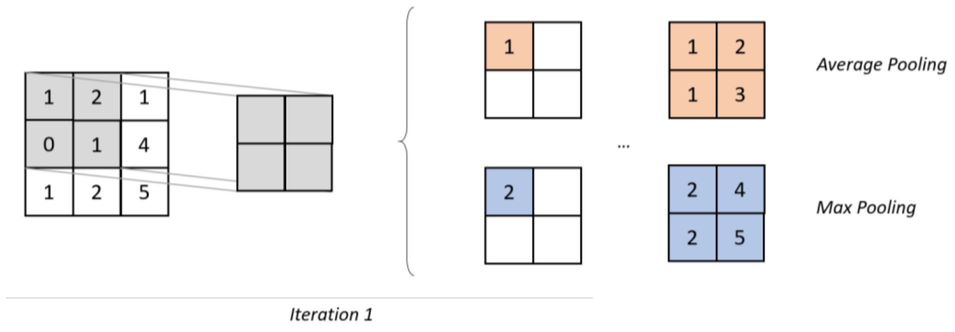

Figure 2 – Max and avg pooling on the same input matrix

In Figure 2 there are two examples of different pooling mechanisms, while both use a kernel with size 2x2, stride 1x1 and no padding. The orange example represents the case of average pooling where, for every slide the kernel performs over the input matrix, an average value is calculated. This exemplifies the case of max pooling since, instead of the mean value, only the maximum one is selected. Though there are other functions that can be applied to pooling, such as the selection of the minimum value, these two functions are the most commonly used. Moreover, depending on the task at hand, different pooling methods should be applied, mainly because each of them has a different impact on image analysis and variance reduction (Boureau et al., 2010). Max pooling is characteristically used to extract the most important and visually intense features like edges, whereas average pooling extracts features in a smoother approach. Depending on how deep the pooling layer is within the network, different pooling mechanisms could be applied to better fit the task being performed. Additionally, pooling is a static method where no values are subject to change, thus no training is performed.

Activation functions

An activation function, also known as transfer functions, is used to determine the output of a node of a neural network, mapping the results between -1 and 1 or even 0 and 1 depending on the function. This mapping occurs by summing the products of inputs and their weights and applying a function to get the output of that layer.

Despite several linear and nonlinear functions can be used, the rectified linear unit – ReLU - is one of the most commonly used techniques, in which if the input value is negative then the corresponding output is zero, and if it is positive, then the output is equal to the input (Wan et al., 2014). This is an example of the concept of activation in the context of an artificial neural network - since for negative values the output is nullified or deactivated, and for positive values, the output is activated, in this case with the same value as the input.

Fully Connected layer

The fully connected layers are the last components, except for the output layers, of a convolutional neural network and its objective is to perform the computations for the class scores. This is the equivalent to have a normal artificial neural network at the end of the convolutional or pooling steps, so that each neuron in this layer will be connected to all the neurons of the previous layer (Wan et al., 2014).

These layers, being the end of the neural network, finalize the end-to-end structure, right before the classification layer where, through softmax or other functions, a final class is returned as the end result. During the training of these networks, the process is executed both forward and backwards. First, an image is fed forwarded through the convnet, suffering several convolutions and pooling processes until the last layer, where the final score is obtained and compared with the label given for that same image. In this comparison, the minimization of the error between the expected result and the obtained one is achieved via gradient descent or other methods - through which the learning parameters of that layer are adjusted. Consequently, by way of the backpropagation method, these modifications are propagated to the previous layer in order to also optimize its parameters in an iterative manner until the primary layers are reached (Wan et al., 2014).

2.1.2 Architectures

Over the years, several architectures have been designed, especially concerning image recognition, employing different implementations of convolution and pooling modules as well as distinct activation functions to achieve greater accuracy. This section covers the main CNN modules, that have been created and that enable the creation of several investigation courses to create more capable networks. In this field, ImageNet has been considered the main benchmarking baseline through which several algorithms have been tested due to its size and data variety (Deng et al., 2009).

VGG networks

In the last decade, several deep learning algorithms have been setting progressively higher results. Some algorithms that first showed the advantage of the depth of the network as a factor of importance towards accuracy were the VGG networks, one implemented with 16 weight layers, as seen on Figure 3, and the other with 19, created by the University of Oxford (Simonyan & Zisserman, 2014).

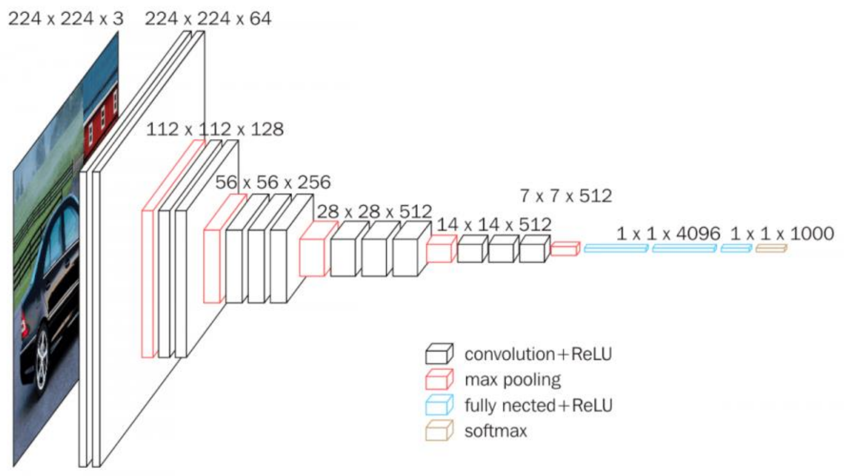

Figure 3 – VGG16 network design

Being still only a deep sequential feed-forward neural network and to achieve its level of accuracy, the VGG networks are characterized as having a high number of parameters due to large convolution operations, which affects training times and thus its applicability in most modern-day use cases. Different modules have since been created that enhance the ability to go wider and deeper, as well as making it more accurate and efficient, both from a memory and performance stance.

Inception Networks

The inception module, first introduced in Inception v1, also referred to as GoogLeNet, is a unit where the goal is to obtain wider networks instead of deeper ones (Szegedy et al., 2014). The starting point towards the inception branch of networks concerns the variability of the salient parts of an image, where the same object can occupy all the area within an image or just a small portion of it. Therefore, the proposal of the inception model is that the analysis of the image should take into consideration this variability, which means that the convolution should not only consider the learning of the kernel values but also which kernel size is the best for each convolutional stage.

In practice, and as seen in Figure 4, this proposal defends the implementation of three convolutional layers, one of 1x1, another of 3x3 and another of 5x5, in parallel with different kernel sizes, as well as a pooling procedure and then pass the concatenated result of these operations to the proceeding layer. However, since deepening the network further extends the computational effort, a dimension reduction of the inputs is performed with an extra 1x1 convolution being added before each convolution of size higher than 1.

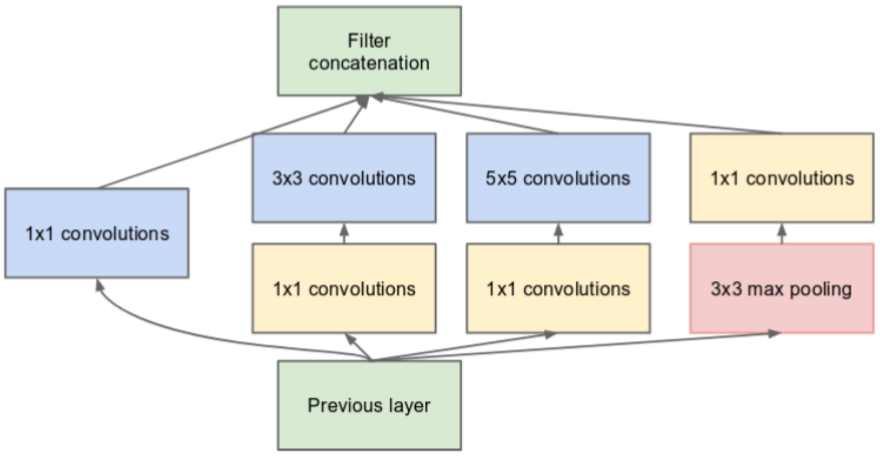

Figure 4 – Inception v1 module

During the training of these networks, not only the kernel values are adjusted, but also the weight of each convolutional layer of the inception module. This means that, on each stage of the network, there is the possibility to highlight the most appropriate convolutions, in order to better represent the relevant features at each stage, granting more flexibility regarding the processing of an image.

Nonetheless, this inception module would later be improved with Inception v2, in which it would be simplified on several stances to reduce representational bottlenecks (Szegedy et al., 2015). The hypothesis was that neural networks perform better when the convolutions involved do not drastically change the dimensions of the input. The first simplification would be the factorization of the 5x5 convolutional layer to 2 sequential 3x3. The second step was to redefine every nxn convolution to a stacked combination of 1xn and nx1. Both these modifications significantly improved the performance of the inception module. The third and last modification concerns the stacking of these 1xn and nx1 layers on a parallel disposition instead of the sequential one, using the same premise as the one defined on Inception v1 where different convolutional processes could have more importance than others on the convolutional process. These processes can be seen in Figure 5, left to right.

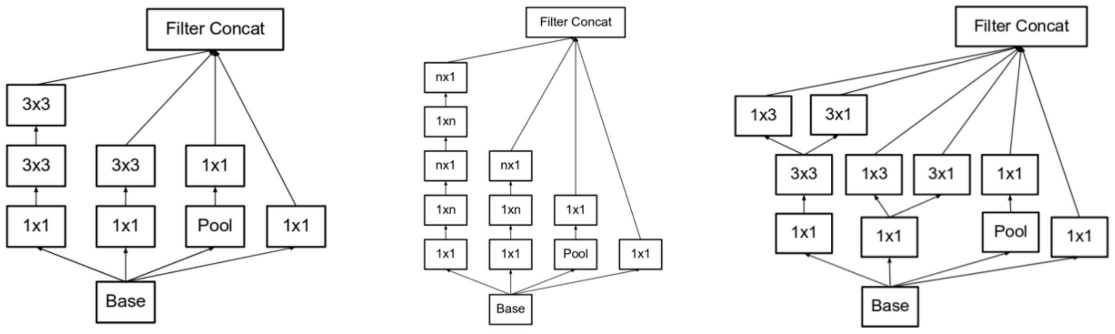

Figure 5 – (a) Simplification of 5x5, (b) 1xn nx1 reduction, (c) filter widening

After the creation of Inception v2, several other inception networks were also created, albeit not with such significant changes to the inception module as in these two previous works. Inception v3 focused on the optimization of the network as a whole towards the classification problem by factorizing 7x7 convolutions, changing the optimization process and introducing batch normalization, to name a few (Szegedy et al., 2015). Inception v4 would focus on the uniformization of the network, specifically on the initial set of operations performed before the inception blocks (Szegedy et al., 2016). The Xception network, instead, performs the modified depthwise separable convolution first with a 1x1 convolution and then channel-wise spatial convolution (Chollet, 2016). Other works like Inception-ResNets (Szegedy et al., 2016) exploit the usage of inception modules and residual learning, a subject that was first introduced by the ResNet networks.

Residual Networks

The start of residual learning emerged with the family of the ResNet networks, mainly ResNet50, where the number of layers was pushed to 50 and even higher in other configurations (He et al., 2015). When the hypothesis of high layer number networks started gaining interest with the VGGs and Inception networks, an issue that arose was that the deeper a network went, the training of the networks becomes more difficult since the accuracy started to saturate and degrade. Initially thought to be the occurrence of overfitting, this issue was later identified as the problem of vanishing gradient. This problem, present in gradient-based learning methods, occurs when the calculated gradient is so small that prevents the weight from changing its value (Hochreiter, 1998). In order to tackle this, ResNets make use of residual units to skip the training of some layers and feed that result in deeper ones. Another distinct module was the introduction of batch normalization layers, boosting the values of the weights and thus higher learning rates can be used to minimize this issue and help accelerate the training (He et al., 2015).

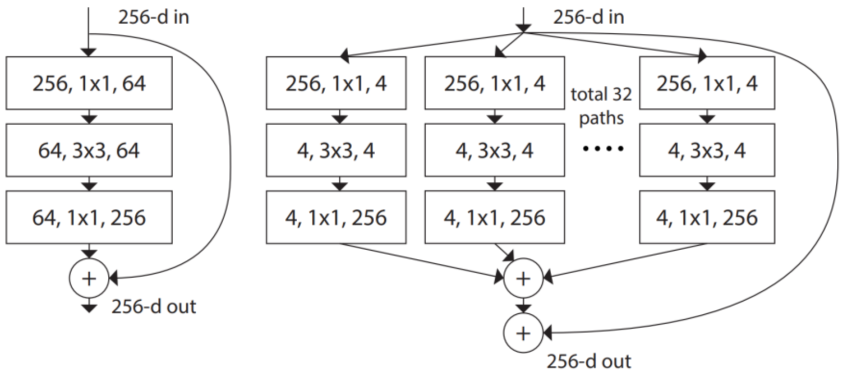

Figure 6 – (a) Residual Learning (b) Aggregated Res. Learning

In Figure 5a) the process of residual learning is illustrated where the input is not only fed at the start of the layer, but also after a couple of iterations to a deeper one. Thus, the values forwarded to the last layer of this module are the first iteration of the input and the input itself. Several other architectures have emerged that take this procedure as a base to other developments. The DenseNets apply the concept of residual learning to the extreme, since the output of a layer is fed forwarded to all the posterior layers (Huang et al., 2016). Although, the ResNeXts also use residual units, a parameter of cardinality is introduced where this unit is repeated several times – Figure 5b) (Xie et al., 2016).

2.2 Similarity Search

Several studies have shown the impact of neural networks regarding visual tasks like object detection and classification. Based on these technologies, several companies have been trying to develop solutions to create visual search engines. With these engines, the concept is to allow an end-user to search for an item by just submitting a photo and the engine returns the most visually similar items. Therefore, this section provides an overview of the main approaches to tackle the similarity search problem.

2.2.1 Category Based

The first and most common approach regarding visual similarity with deep learning was to train a convolutional neural network towards image classification. The principle was to establish a dense detail on the classification tree of training data such that the network would specialize in performing specific classification tasks.

To achieve this, publicly available models pre-trained on the ImageNet dataset are typically used, upon which a training refinement process takes place (Deng et al., 2009). Being thoroughly optimized to extract important features over a variety of different classes, these models are considered good base models to avoid the training of a neural network from scratch. The process of transfer learning can then be applied to these models, in which the main objective is to fine-tune the parameters to another task (Zhuang et al., 2019). The process of transfer learning can also be partial or complete. In the first case, only some layers are subject to modifications, while on the latter all the layers are refined. After achieving a satisfactory classification accuracy, the feature array of the last fully connected layers would then be retrieved, being regarded as the numeric representation of the image (Machado, 2017). The idea was that if the network was able to successfully discriminate and classify different images, then the embeddings preceding that classification would depict the most important features of the image, in order to compare them through a distance metric and retrieve the most similar ones. *Pinterest*, among others, developed a framework to obtain a numeric representation of every image. With these arrays stored in a distributed database, the retrieval process was performed through a KNN algorithm whose distance metric - Euclidean, Minkowski or Hamming, to name a few – would represent the similarity between images (Jing et al., 2015b).

2.2.2 Siamese Networks

The similarity approach based on classification has several assumptions and requirements that could be prejudicial both from an application and accuracy perspective. It implies the processing of large amounts of image and labels to train a neural network with a purpose that is not the end goal, withholding several inaccuracies since the training process does not consider the subsequent actions.

To tackle this issue, one approach was to oppose convolutional neural networks with the same composition and weights, in a twin format, also known as the siamese network (Koch et al., 2015). The objective of this structure is to feed two images at the same time to the network, one being labeled the anchor and the other being either the positive or the negative of the anchor. At the end of this structure lies a merging distance layer that will apply a distance function to the extracted features of these two images to perform pairwise similarity assessment between features. The concept, as seen in figure 6, is that the neural network training will already evolve towards the approximation between positive-anchor pairs of embeddings and the separation of negative-anchor ones.

In this manner, the neural network is effectively trained regarding the similarity of features and the demand for large volumes of data is diminished, having been used in several domains. Koch et al. (2015) and Rao et al. (2017) have applied it for one-shot learning, exceeding other state-of-the-art method. Maheshwary & Misra (2018) also used them to determine the semantic similarity between job postings and resumes and (Dey et al. (2017) proposed one to detect similar signatures and even identify the forged ones.

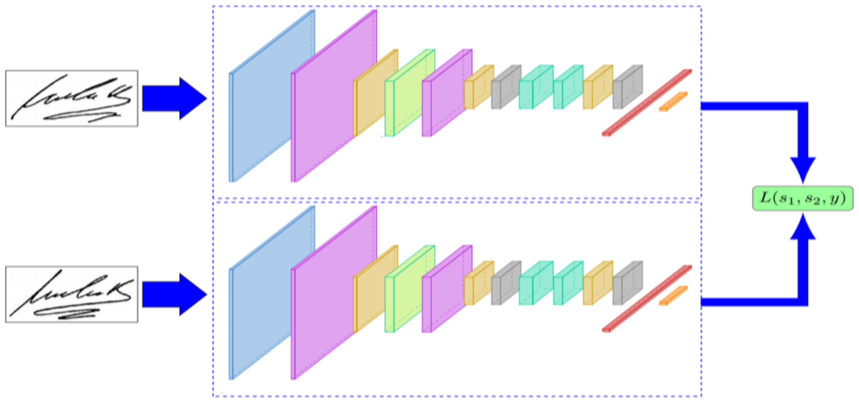

Figure 7 – Siamese network

2.2.3 Triplet Networks

The last multi-CNN structure to arise in the similarity search field was the triplet network, as seen in Figure 7, and it considers examples with an anchor, a positive and a negative image in the same input and enables relative similarity (Jiang Wang et al., 2014). In a siamese network, the relationships created are either (i) related or (ii) non-related pairs of images, making it difficult to structure a ranking method between several images (Hoffer & Ailon, 2014). In this network, the concept of "the positive (A) is more similar to the anchor (X) than the negative (B)" allows another input as "the positive (B), previously a negative, is more similar to the anchor (X) then (C)". With this structure, one could create a ranking A>B>C relationship, allowing a more detailed structure regarding similarity and thus achieving better results than the previous methods (Hoffer & Ailon, 2014; Kumar B G et al., 2016). Concerning training datasets, for N images, the category-based methods have up to N examples, the siamese has N(N-1) distinct pairs and the triplet ones have N(N-1)(N-2) examples, generating a greater dataset out of all.

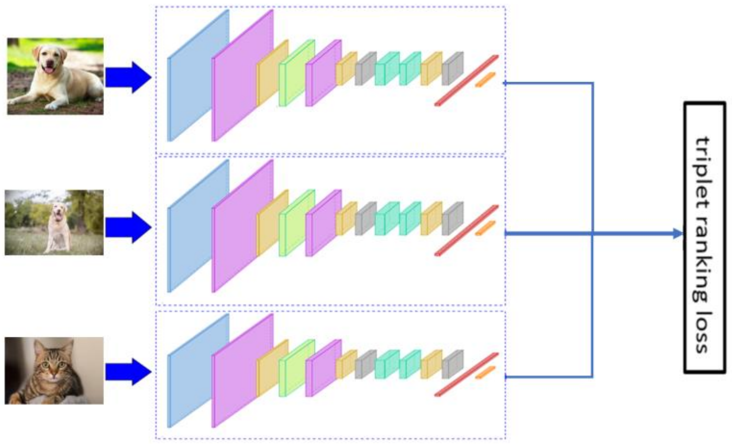

Figure 8 – Triplet network

This method has been successfully implemented in several use cases achieving state-of-the-art results, such as face recognition (Schroff et al., 2015) or, as the case of this work, applied to the fashion industry to provide accurate recommendations of same style items (Shankar et al., 2017). Nonetheless, and also concluded by Shankar et al. (2017), though accurate image similarity is already possible for normal applications, the computational efforts of providing triplet based solutions still require access and maintenance of distributed and high-performance infrastructure.

2.3 Dimensionality Reduction

Due to the exponential increase in data collection experienced over the years, companies and researchers have no access to greater datasets to empower the training of learning algorithms by providing more data. While this leads to more opportunities for improvement, it also leads to the hazard of having excess variables, having an impact on both the performance and accuracy of a model. This is particularly present in the KNN problem, in which a higher the number of variables leads to a higher need for more data, commonly known as the curse of dimensionality (Indyk & Motwani, 1998). Thus, the process of dimensionality reduction has gained relevance as an effort to reduce the high-dimensionality of data into a lower one, while preserving the maximum information possible (Van Der Maaten et al., 2009).

In this section, a description of the most traditional and commonly used methods is performed concerning the major advantages and drawbacks, as well as an analysis of the evolution of hashing functions.

2.3.1 Traditional Methods

The application of main dimensionality reduction methods, specifically on the similarity search context, has progressed in several directions. Among the first, there is the principal component analysis (Wold et al., 1987) where an orthogonal transformation is performed to convert the observations into linearly uncorrelated variables. This procedure, and resulting variations (McNames, 2001), ensures a synthetization oriented towards preserving most variance, but has the drawback of losing context of the variables.

Other approximative methods, more focused on efficient KNN searches, such as the K-DTree (Bentley, 1975) and Ball Tree (T. Liu et al., 2006) use space partitioning data structure to organize points in multidimensional space to expedite the nearest neighbor search queries. The concept of both algorithms is to use the tree properties to avoid computing large portions of the search space, thus making the search more efficient.

Locality Sensitive Hashing functions (Indyk & Motwani, 1998) have also been implemented to accelerate the search, where data points are hashed into buckets so that points that are close are located in the same buckets with higher probability. Nevertheless, this method has been proven to lack for real-world applications (Shankar et al., 2017).

2.3.2 Learning Hash Functions

All the previous methods took into consideration a generic function that, once applied to a dataset, would produce a partitioning scheme that would speed the search query on the nearest neighbor problem. Another approach concerns the learning of personalized hashing functions to generate a compact code, aiming to obtain a result of the nearest neighbor search as close as possible to the original result (Jingdong Wang et al., 2014).

Concept and Families

Since these functions cannot be applied in a static manner like other methods previously mentioned, a framework must be followed to guarantee the creation and improvement of the function, thus becoming an optimization process. Therefore, the methodology of learning functions involves (i) the creation of a hash function to be adapted, (ii) the similarity measure to apply to the codes (iii) the definition of the loss function and (iv) the optimization technique adopted (Jingdong Wang et al., 2016).

This area of research currently holds four different families of learning-to-hash functions: pairwise, multiwise, implicit and quantization. The pairwise functions progress through the minimization of distances obtained via pairwise input mechanism. In multiwise functions, each observation possesses several labels and the evolution of the function is dictated by the maximization of the number of correct labels given to the observations. The implicit one focuses on learning the hash function without the explicit evaluation of distances at the input level or at the code space. The quantization process constrains a set of values, such as the real number, to a discrete set where the minimization of the distance difference - also denominated as quantization error - is performed both at the input and code level (Jingdong Wang et al., 2016).

Of these families, this study will focus on the quantization processes. The formulation of a quantization hashing process differs from the remaining processes in its approach, in which distance measures can be applied just like in a normal nearest neighbor search. Additionally, it could also be regarded as an integral part of pairwise and multiwise functions, thought capable of better results, since the input mechanism of a quantization process can also be performed considering pairs or triplets of data (Li et al., 2015; Jingdong Wang et al., 2016; X. Wang et al., 2016).

Quantization

The first model to successfully implement a quantization process for an approximative nearest neighbor search was the Product Quantization (PQ). This process involves using a set of codewords to divide the space into a cartesian product of lower-dimensional subspaces and the respective quantization of each of them separately. A vector is then represented by a compact code that features its subspace quantization indices (Jégou et al., 2011). This offers a lot of improvement over other ANN methods, since, both performance and memory wise, the data structure of the index is lower than other conventional methods. With a different approach, Gong et al. (2012) introduced the iterative quantization process (ITQ) in which, firstly, a dimension reducing mechanism takes place, like PCA, followed by an iterative minimization of the quantization error. This minimization is achieved by iteratively rotating the axes in order to find the rotation that explains most of the variance, thus minimizing the quantization error. Norouzi & Fleet (2013) would later develop two models, the Orthogonal k-means and Cartesian k-means, being both extensions of k-means while closely related with ITQ and PQ and performing even better than these. Ge et al. (2014) would also improve the work of PQ by using the quantization distortion as an objective function to analyze the optimality of the product quantizer. In this, both the parametric solution - assuming the data follows a Gaussian distribution - and the non-parametric solution were able to achieve better results than PQ. T. Zhang et al. (2014) proposed a variation of PQ, the composite quantization (CQ) where the orthogonality constraint of the space partitioning is relaxed, allowing a more flexible structure to form, performing better than all above since the previous ones are restrained versions of CQ. On Figure 8 it is observable how the partioning of data points is performed on several methods, highlighting the flexibility of the CQ.

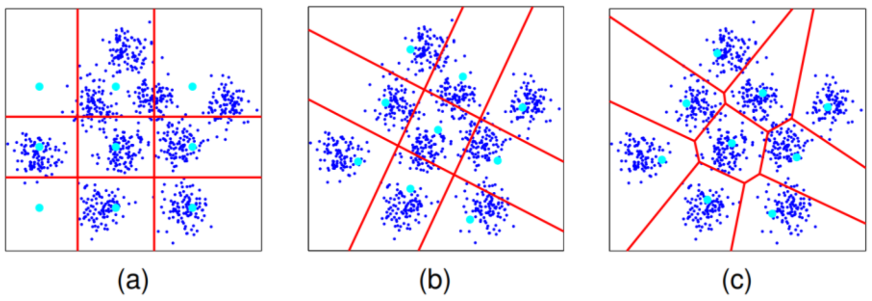

Figure 9 – (a) PC, (b) Cartesian k-means, (c) CQ

2.3.3 Neural Hashing

All the previous methods, static and learning ones, take into account representation of a vector into a smaller code that enables faster and accurate nearest neighbors but does not consider the quality of the initial vector by itself before the hashing process. Therefore, the maximum accuracy of these methods is bounded by the quality of the initial vector representation, whose computation involves a distinct process.

To address this issue, several studies have been made on how to merge (i) the feature extraction component of a convolutional neural network and (ii) a hashing mechanism, all integrated into a unique training process from image input to compact code generation. To address this, Guo & Li (2015) proposed a two-stage convolutional neural network (CNNH) where the codes are learned from pairs of labels and then the hash function and feature extraction process are learned based on the codes. This, however, does not allow the simultaneous learning of both processes.

Lai et al. (2015) offer a deep hashing method (NINH) based on triplet labeling, successfully achieving a fully integrated learning process from end-to-end. Li et al. (2015) would also propose an end-to-end learning method (DPSH), but whose training method was based on pairwise labels, beating other state-of-the-art methods at that time. Bringing the quantization methods into the deep neural hashing learning process, Cao et al. (2016) present a siamese network (DQN), with similar results to DPSH, with a hashing function based on quantization of the codes, achieving similar performance at the prior.

Contrary to both DPSH and DQN siamese structure, X. Wang et al. (2016) make use of supervised triplet training and following hashing (DTSH), achieving yet better results than these two methods. Combining the quantization nature of DQN and the triplet structure of DTSH, Yu et al. (2018) PQnet, beating all previous state-of-the-art and further enforcing the importance of relative similarity achieved through triplet based learning. Still regarding deep triplet quantization learning, Song et al. (2019) present a structure (DRQ) where the learning of the codebooks used for product quantization is performed in a recurrent manner. By making use of this recurrent process, the same codebook used in product quantization can be used iteratively to achieve different code lengths, reducing the training time when evaluating different code lengths. Moreover, due to this iterative process over the same module, the number of parameters of the network does not increase linearly with the code sizes, contrary to the previous networks. When comparing with PQnet, DRQ loses in the mean average precision score on CIFAR-10 dataset by 0.3% to 0.5% but wins on the NUS-WIDE dataset on all but the 12 bits code size results. However, while being one of the best models on CIFAR-10 and NUS-WIDE, DRQ loses to DTQ (B. Liu et al., 2019) on the ImageNet dataset on all code size, by 9.8% to 14% (Song et al., 2019).

The DTQ structure can be seen as an improvement over DQN since the quantization process being applied relaxes the orthogonality constraint, and thus can be viewed as a variation of both product quantization and composite quantization (B. Liu et al., 2019; T. Zhang et al., 2014). By transposing the orthogonality constraint to the objective function, the model evolves towards a flexible partitioning structure that allows the best representation of the vector into a compact code while also trying to maintain the degree of orthogonality. When opposing DRQ to DTQ regarding the number of parameters of the hashing process, the DRQ possesses an inferior number. However, this is does not include the number of parameters involved on the convolutional neural network, which on most neural networks is 2 to 3 orders of magnitude higher than the number of parameters in the hashing layers (He et al., 2015; Simonyan & Zisserman, 2014; Song et al., 2019).

2.4 Critic Analysis

After the analysis of (i) convolutional neural networks, (ii) similarity search approaches and (iii) dimensionality reduction, this section offers the evaluation of the author concerning the desired areas of study and the subjects through which the following work focuses.

Concerning convolutional neural networks per se, it is clear that it is a field where research has already produced algorithms capable of thoroughly analyzing an image and identify several classes of images better than a human being since 2015 (He et al., 2015). Therefore, little focus will be given regarding the formulation of a novel convolutional neural network, but rather an analysis and selection of the currently available ones. This should take into account (i) accuracy results on publicly available datasets, (ii) number of parameters and floating-point operations, affecting performance, (iii) publicly available networks that have already been pre-trained on ImageNet. The reason for the third aspect is driven by time and computational restraint, since performing a parameter optimization of these networks in these datasets involves strong computational power and time to do so. The first two guidelines are due to the objective of creating a solution whose feasibility on real-world applications is a must, especially considering the goal of providing a real-time response. The following chapters will include the benchmark of available CNNs (Bianco et al., 2018) according to these guidelines and selection of the most appropriate.

Furthermore, on similarity search approaches, it has been shown that the triplet method is the most capable of producing accurate results concerning similar products. This has even been proven in real scenarios as on the fashion e-commerce industry, where robust and accurate, though computationally expensive, results are already being produced by several companies (Shankar et al., 2017; Zhai et al., 2019; Y. Zhang et al., 2018). Therefore, the approach that will be further explored is the triplet based one.

Finally, on dimensionality reduction, several benchmarks indicate that both traditional and isolated learning hash functions perform worse than integrating a learning hash function in a convolutional neural network to produce compact codes. Thus, the focus will rely on the construction of a model following the work on DTQ and comparison with a non-hashing triplet network.

3 Methodology

The following chapter defines the methodology employed to create a deep learning model based on existing convolutional neural networks architectures tested on several public datasets of image recognition. From this basis, how to adapt these networks into a feature extractor module capable of producing compact binary codes, a development workflow is defined that will be the guideline for the sections of chapter 4 and 5. Given the objective of reproducibility of this model towards a practical business solution for ecommerce retailers, a high level architecture is presented, both for online and offline image similarity recommendations. As a final section, a summary of the technology stack used in this project, both software and hardware wise, is also provided due to its relevant impact for real world replication of the system developed.

3.1 Development Workflow

As previously referred, several image analysis tools currently developed have been using deep learning models and transforming the classical classification of images into a similarity approach. On the similarity challenge, the evolution from classification tasks, to siamese networks and finally triplet based similarity has provided increasingly better results, hence why this work focuses on the implementation of a triplet network.

To address the goal of achieving a deep triplet neural network capable of producing high quality and compat binary representation of images, there are four main stages that must be defined and carefully adjusted. As seen in figure 9, there are four components regarding the creation and training process of the model, all of them with the main topics included.

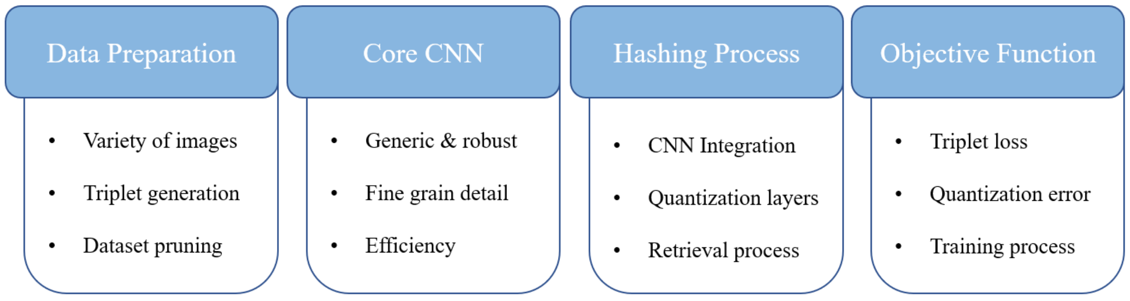

Figure 10 - Main stages for the implementation of the model

The first concerns the creation of the training dataset. On one hand, this dataset must include several images across several dimensions to ensure that the model is robust enough to be able to react to both high-end catalog produced images but also user submitted images. On the other hand, given the exponential generation of triplet combinations, a selection strategy is of paramount importance to balance this trade-off between quality and efficiency of the training.

The second concerns the base convolutional neural network being used as the core module of the neural network. Though the objective is to train a similarity network, this is only possible if the model is also capable of distinguishably analise broad and fine grained details in images. On fashion items, this ranges from assessing which items are shown in an image to the patterns in the item itself. Moreover, even within the pattern, the model should be able to recognize distinct elements – e.g. a t-shirt bearing a skate or a general view as a beach.

The third component regards the hashing process, in this case quantization. The integration of a CNN with a quantization method is not a direct process, given the distinct nature of the two – one improves by continuous differentiation, while the other concerns a discrete optimization problem. Therefore, the layers to introduce must consider the core steps of a quantization process, whilst adequating it to the training framework of a neural network. To do so, the objective function is a minimization of both the similarity function between the triplets and the quantization error induced by the quantization layers created. The final step within these layers is to define the retrieval of nearest neighbors after having the database populated with the catalog images processed and quantized.

The fourth and last component concerns the objective function and how the model will iterate to improve while training. For the objective function, the two main components of analysis are the specification of the triplet loss function and of the quantization error. The first is the main driver towards better similarity searches and the second towards a more effective representation of the binary codes to be produced. Though explained in a separate section, an orthogonality constraint is also applied to the code generating codebook with the goal of generating codes with less redundancy on the codewords.

3.2 Solution Architecture

The design for a scalable and responsive architecture from a *Software-as-a-Service* (SaaS) point of view should consider both online and offline recommendations. First, to address these two use cases, the retailer's catalog is fully analysed by the proposed model and the codes that are produced will be stored in an external database. Moreover, and according to a specified frequency, the retailer will also have readily available a set of *N* neighbors for each of the items in its own catalog. This will support the offline recommendations and increases the frequency at which these pre-computed neighbors should be updated, alongside with the degree of new items present in the catalog. In this manner, the proposed architecture is able to respond to both online and offline recommendations, as seen in figure 10. The distinction between online and offline is, for the former, the computation of the nearet neighbors is performed in a synchronous real-time request, where every picture submitted is processed and compared with the binary code database. On the offline, the objective is to provide similar products between items in the same catalog – thus the input image is known at the start and can be precomputed and saved to accelerate the supply of recommendations.

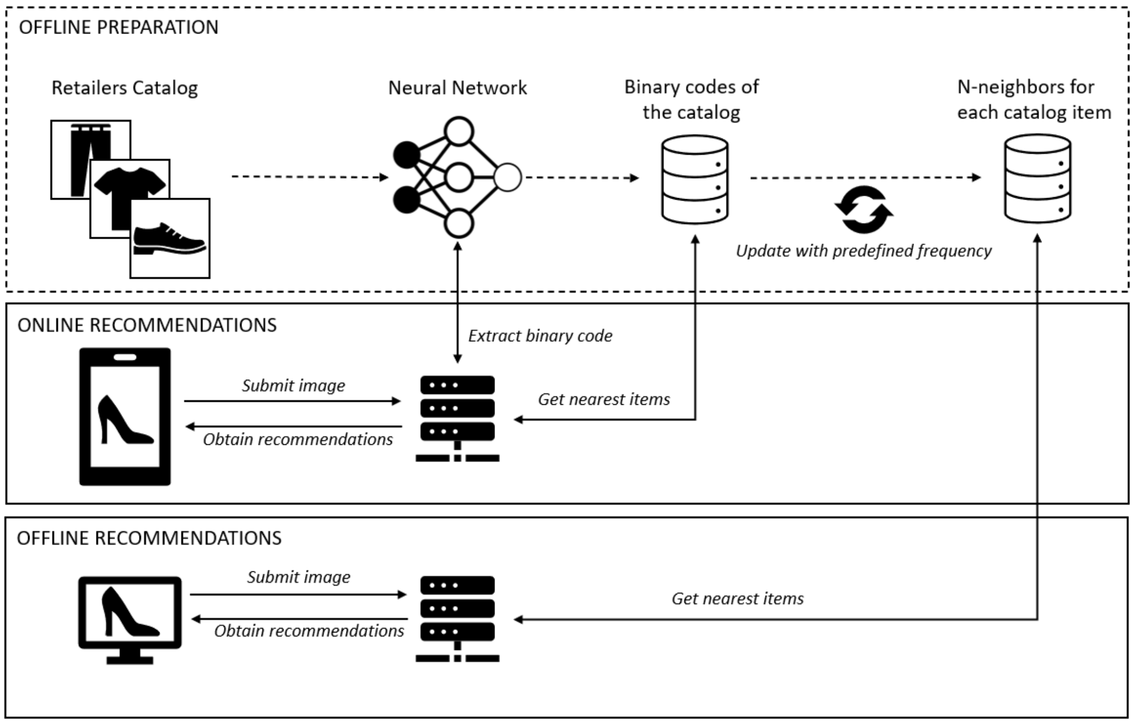

Figure 11 - Overall product architecture and use cases

3.3 Use Cases

For online recommendations, the typical use case is of a person submitting an image in search of a particular item. This can be done from the photography collection in the smartphone or directly from the camera. On this picture, the user has the option to crop the image and then submit it, finally receiving the most similar items – figure 11.

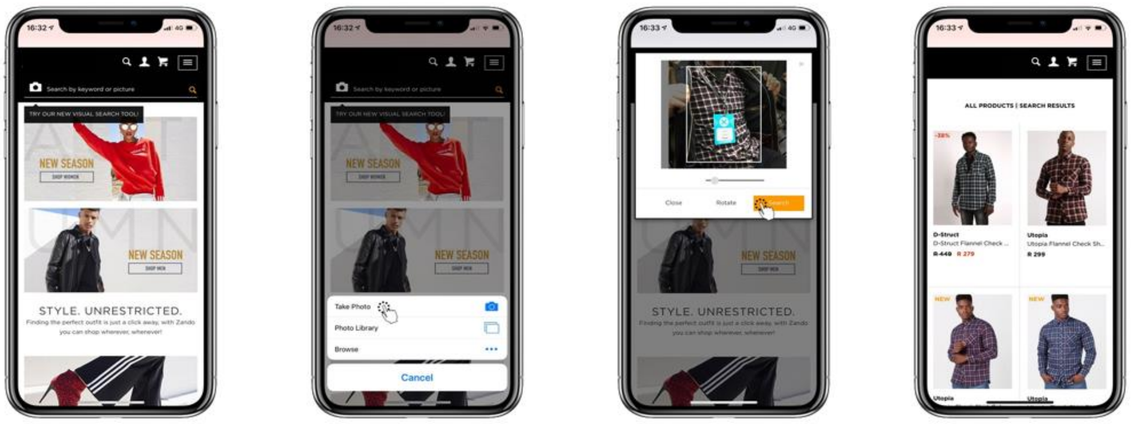

Figure 12– Online case: image submission process left to right

However, the most common case is of a user just browsing a retailers ecommerce website looking for items. For this case, when the user clicks in a product, the product page should load with a set of similar items - Figure 12, the recommendations in red. The objective here is to capitalize on the attention the user gave to a specific product and show an array of slightly different items that could be better fitting to its taste. The number of requests in this type of recommendations is higher than the previous, hence why the precomputation of results is essential to scale the service to several users at the same time.

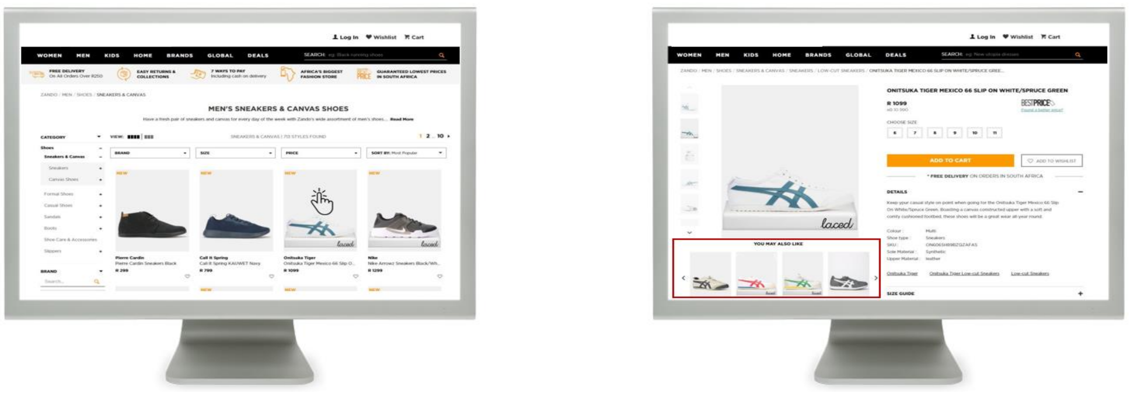

Figure 13 - Offline use case: webpage on the left, product page on the right

3.4 Technology Stack

The main programming language to be used in this report will be Python since this is the main language being used on most projects. Within this programming language, the following libraries will also be the main ones to be used:

- Tensorflow for neural network design and training,
- Pandas and NumPy for data treatment and KNN implementation,
- SQL on BigQuery and PostgreSQL databases for data storage/retrieval,
- FlaskAPI for real-world scenario evaluation.

Still concerning real-world scenarios, the benchmark of the final solution will involve testing how the solution will perform regarding (i) latency of each request and (ii) the maximum number of requests per second. To do so, both Gunicorn and Nginx will be used to deploy the solution from a development environment into a fully operational production HTTP server.

The training of the models, and to make use of the advantages of graphic computer units, will involve the implementation and configuration of Nvidia drivers, along with the installation of the Cuda and cuDNN drivers to maximize the utilization of resources for deep learning training. This computation will be performed in a server with the following characteristics:

- Intel Core i7-9700K Octa-Core 3.6GHz, Turbo 4.9GHz 12 MB Skt1151
- Nvidia GeForce RTX2080Ti Triple Fan 11GB GDDR6
- SSD M.2 2280 Samsung 970 Evo Plus 500GB MLC V-NAND NVMe
- RAM 32 GB DDR4-3000MHz CL15
- Motherboard ATX MSI Z390-A Pro

The data for this project will be provided by ShopAI, a company operating in the field of image similarity for fashion e-commerce, with data obtained from in conjunction with its customers – both store and street photos of items – previously labeled. These labels include the category established for each item. The process of mining a triplet based dataset will be performed during the data preparation process of this work.

4 Implementation

The following sections will describe the creation and development of the similarity model, starting from the data preparation, moving to the core CNN selection and ending with the implementation of the quantization layers and the training process of the model.

4.1 Data Preparation

In this regard, the dataset preparation involves the gathering of several images on different conditions, in order to maximize the accuracy of the model across different kinds of apparel items. Afterwards, a triplet definition and selection takes place, in which a set of criteria were defined to allow a hierarchy of similarity to be drawn. Based on this hierarchy, only the best fitting triplets will be used during the training of the model.

4.1.1 Data gathering

As with any machine learning algorithm, the most essential part concerns the gathering of data, promoting the balance between the amount of samples from each kind of examples. In this case, the images must consider different scenarios and conditions to assure that the model is robust yet variable enough to provide good recommendations.

To do so, the first challenge concerns the differentiation between studio produced images from retailers' catalog and normal pictures taken by the average person. While the first pertains high resolution pictures of apparel items in the most distinguishable angles in perfect lighting conditions, the second is typically affected by inferior quality and thus noisier images. Additionally, the number of items in the fashion and apparel business grows from clothing and footwear to accessories and other wearables - not to mention the distinction between genders and adult versus child items. Lastly, and on top of the previous one, for each set of these categories, the variety of patterns present in a piece of clothing is virtually unlimited. Anything from simple stripes and squares to complex images as beaches or skylines must be considered. Consequently, a wide variety must be included for each category.

For this project, the data gathering was enabled by ShopAI, a company operating on the visual search industry for ecommerce fashion, where several images from different retailers have been collected meeting the standards mentioned above.

4.1.2 Triplet Generation

One of the major issues regarding deep neural network training is the access to high volume of data to effectively train the model. However, when creating triplet based data, the generation of new samples suffers from a combinatorial explosion. For a set of *N* images, the total amount of distinct triplets capable of being generated is approximately N^3. In this case, the challenge is to select only the best triplets combinations in order to maintain high quality samples and avoid lower quality ones.

Previous works on triplet based systems have defined a multi category dataset for the images being used: distinction between shirts, shoes, jackets, among others. While it is true that a shirt is different from a jacket, the same can be said between two jackets or even that a shirt and a jacket are similar, because they possess the same pattern on the front. This highlights the relativity that is inherent to similarity and the shortcomings of just classifying an item into just one specific category.

B. Liu et al. (2019) propose a technique defined as *Group Hard* where, for every anchor-positive pair concerning same category images, a negative is selected out of a different category, considered to be the hardest one to distinguish from the positive. This is performed by computing a distance metric between them and only a fraction of the most similar negative images is selected. This approach, nonetheless, contemplates assumptions that could undermine the accuracy of the model. If the model is going to be trained using these triplets to produce more accurate embeddings, then what embeddings are going to be used at the start and are they good enough to perform this selection? Moreover, this study performs relatively well when tackling wide category challenges, such as the ImageNet competition, but how well is it certain to perform in a narrower universe of analysis such as online apparel images as in this project? Finally, as stated, two items might be from different categories and yet be similar enough through a very specific pattern or belong to the same category but with completely different style and thus not very similar. The assumption that same category items are always the most similar may restrict the generalization of the algorithm. Thus, though based on this strategy, a new approach was developed where the selection of triplets is performed considering a hierarchy of similarity between images.

Starting from the *Group Hard* strategy, and represented in figure 13, the first division to be performed is to distinguish the different categories among the initial image dataset. This involves categorizing the items into common fashion categories such as shoes and trousers, since this is the most obvious and direct differentiation between items a person can make. From this, another characteristic of the fashion industry is the launching of products in sets/collections – those that typically possesses the same design or pattern. Taking advantage of this, the items are then grouped into same collection/pattern sets where the similarity among them is expected to be higher than between same category items but belonging to different collections. It is in this second level that the differentiation of items within the same category is enforced and where the inner-category similarity analysis is further pushed to allow a finer separability of positive/negative cases to occur. Finally, a third level of classification is performed – green in figure 13 - where the goal is to associate different images of the same item. The focus is to collect (i) catalog images but in different perspectives and (ii) traditional in street photographs from the same product and index them all in the same ID. By introducing noisier street images and indexing it to the images from studio conditions, the objective is to allow the model to process both high and low quality images as different representations of the same product. Given the entropy of street photos, the matching with in shop images of different perspectives aims to fortify the matching of a user submitted image to the correct item.

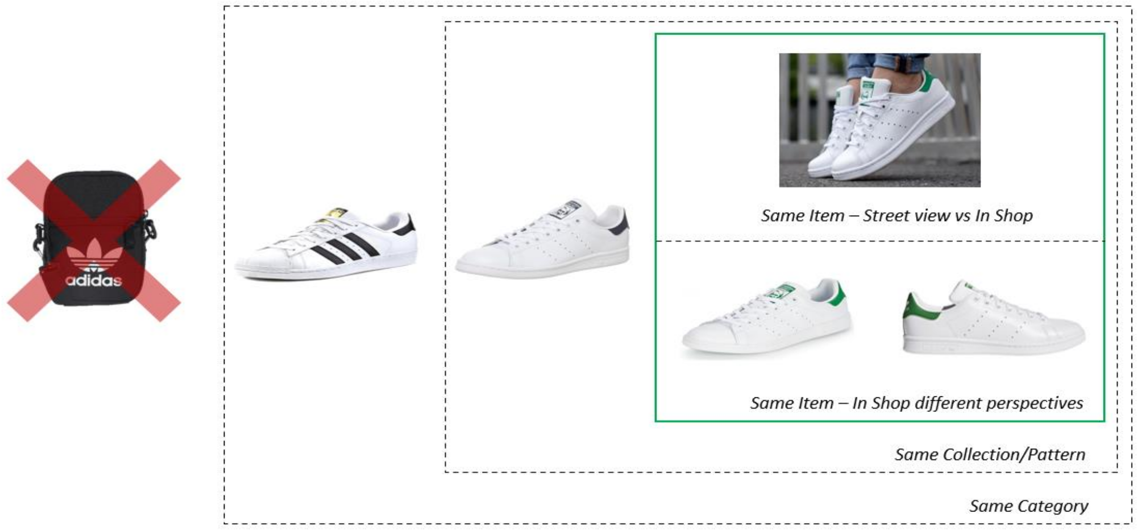
Same Item – Street view vs In Shop
Same Item – In Shop different perspectives
Same Collection/Pattern
Same Category

Figure 14 – Triplet selection strategy

4.2 CNN selection

The core component of this model involves creating a feature extractor network upon which the hashing layers will be built upon. However, to train a neural network from scratch involves heavy commitment of resources, both computational and time wise. Such task, in the current day and age, could be surpassed by performing transfer learning on pre-trained models and consulting several benchmarks to select the best choices. In this sense, this chapter shows the walkthrough procedure to select the core architecture.

4.2.1 Keras Applications

In this case, the pre-trained models that will be analysed to integrate the core extractor module of the network were developed for the ImageNet challenge, given its wide category variety and data volume, and are available on the Keras library. On Table 1 there is a representation of the benchmarks and main characteristics of the neural networks available on Keras at the day of this writing. Nonetheless, to aid in the selection of the model, additional benchmarks have been considered to expand the analysis of the models.

Table 1- Keras pre-trained models on ImageNet

Model	**Size**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

4.2.2 Benchmark & Analysis

While the Keras library provides several state-of-the-art pre-trained convolutional neural networks, to choose between them is not only a matter of comparing top accuracies. They must have a good accuracy but should also provide a good response time and low memory footprint, particularly for the online recommendations part where the mobile factor takes the center stage. Therefore, and using the benchmark provided by Bianco et al. (2018) on figure 14, one can see that NASNet-A-Large achieves the highest accuracy but with a heavy toll concerning G-FLOPs and number of parameters. Other networks that have a good performance, with low G-FLOPs and number of parameters, are the SE-RexNeXts, who take the residual learning with cardinality to another front by adding a Squeeze and Excitation modules. However, these networks are not covered by the Keras library and therefore are not subject of study in this work.

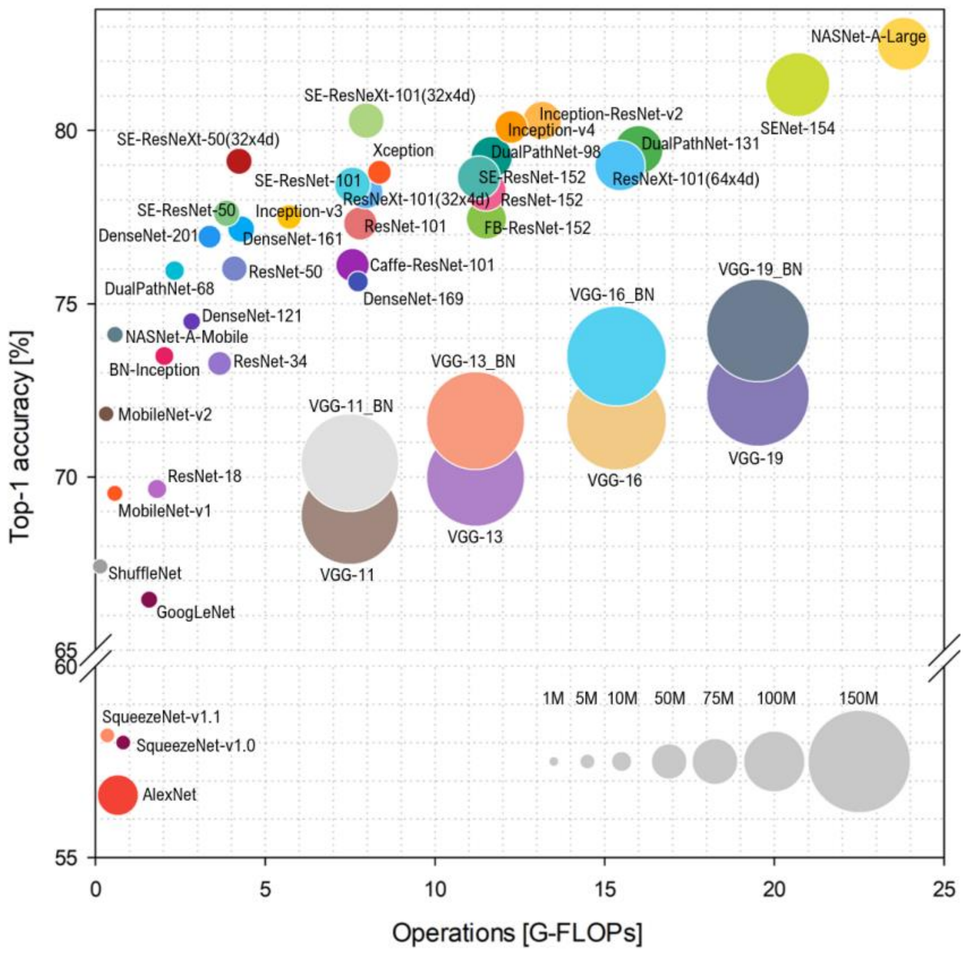

Figure 15 – Top1 Accuracy vs G-FLOPs vs Number of Parameters

Apart from these networks, others that also provide good results are the Xception, the Inception-ResNet-v2 and the Inception-v4, among others that are absent from Keras. In fact, when considering the networks that are present, Inception-v4 is still not fully supported by Keras at the moment of this writing. Narrowing the response times provided by Bianco et al. (2018) into the list available from Keras one can analyse the different response times for a Nvidia Jetson TX1 on Table 2. The focus of comparison is mainly on the batch size 1 because the use case where speed is paramount is the online recommendations since the request and delivery of the recommendations is performed in a synchronous method – the neighbors are computed after the input image is received. At the start, one should discard the VGGs due to their low accuracy levels and high number of parameters and G-FLOPs. The DenseNets and the Resnets, though with sub 100ms results, provide inferior results when compared with Xception and Inception-ResNet-v2. Out of these two, the lastest has double the number of parameters, G-FLOPs and response time than the former, while only slightly improving the overall accuracy. Thus, and being its author the creator of Keras, this network was considered the best fitting case in the trade-off of Accuracy vs Response Time vs G-FLOPS and Parameters. Another network that has been performing well across all these dimensions is the EfficientNet (Tan & Le, 2019), and although is still not fully supported by Keras at this moment, it is an architecture that will be further analysed and tested in future works.

Table 2- Benchmark times for different batch sizes on Nvidia Jetson TX1 (ms)

DNN	**1**	**2**	**4**	**8**	**16**	**32**	**64**
DenseNet-169	137.96	130.27	110.82	100.56	92.97	88.94	
DenseNet-201	84.57	61.71	62.62	53.73	49.28	46.26	
Inception-ResNet-v2	198.95	141.29	127.97	130.25	117.99	116.47	
Inception-v3	79.39	59.04	56.46	51.79	47.6	46.85	
MobileNet-v1	15.06	11.94	11.34	11.03	10.82	10.58	10.55
MobileNet-v2	20.51	14.58	13.67	13.56	13.18	13.1	12.72
NASNet-A-Large	437.2	399.99	385.75	383.55	389.67		
NASNet-A-Mobile	133.87	62.91	33.72	30.62	29.72	28.92	28.55
ResNet-101	84.52	77.9	71.23	67.14	58.11		
ResNet-152	124.67	113.65	101.41	96.76	82.35		
ResNet-50	53.09	44.84	41.2	38.79	35.72		
Xception	98.96	93.4	90.49	87.65	86.89		

4.3 Quantization layers

As aforementioned, the inclusion of hash functions within the training of the neural network as a whole produces better results when opposed to separating them into two different stages. The reason is that integrating both the improvement of the quantization error and the triplet error within the same unified end-to-end framework enables the propagation of the error across the whole structure. With this, both the quantization layers and the feature extractor ones are aligned and will be having adjustments every step of the training, since one change on one of the layers could affect the others.

In this sense, the creation of additional layers were also included, following the work of B. Liu et al.(2019) on how to implement product quantization layers on top of neural networks. The proceding sections will describe the main layers and concepts used to effectively perform this process, as well as its definition through which binary codes are created.

4.3.1 Codebook, Subspaces and Subcenters

The main idea of product quantization is to split a vector of size D into smaller sub-vectors, each of them being quantized into a reference of the nearest codeword. This division occurs by specifying both the number of subspaces *M* onto which divide the vector into *M* parcels of size *D/M* and the number of codewords *K* each of these subspaces has. It should be noted that the parameter *M* is used to control the trade-off between accuracy and memory-cost, as higher values of *M* can achieve a finer and more accurate quantization, despite the higher memory usage. For *K*, each of the codewords is a vector of size D/M and a common value to be defined is 256 since the binary representation of this value only requires 8 bites (= 1 byte). The resulting vector, the binary code, as a result of the decomposition and subsequent quantization of each *m* subvectors will have a memory footprint of just $M * \log_2 K$ bits.

To produce this binary codes, however, the number of subspaces and subcenters is used to create what is known as the codebook *C*. This is essentially a matrix composed of all the codewords *K* for all the subspaces *M*. Figure 15 shows a practical example of how this process is achieved.

At the start of the process it is shown an input vector whose embedding is of a continuous nature, just like the one obtained from the feature extractor layers, with size D=8. In this case, the number of subspaces considered was 4 and the number of codewords is 2, consequently the embedding is then divided into 4 smaller components, each of them highlighted in different color. Afterwards, there is the codebook, which in this case, and also in the model itself, the matrix was compacted into a single wider matrix to shorten the number of operations to perform. Each subspace is restricted to a dimension of 2x2 since *K=2* and the size of each codeword is D/M = 2, being each column of the highlighted areas a codeword of 2 centers.

When comparing each of the subvectors with the specific subspace in the codebook, the subcenters with the nearest values to the ones in the subvector are then indexed in the binary code. As an example, for each of the subspaces, the corresponding codeword selected as the nearest centers is specified by a red box. Considering the embedding as *z*, the codebook as *C* and the binary code as *b*, then for the example shown is it seen that this binary code is the one where the vectorial product of *C* and *b* is the most similar to *z*. The difference between *z* and *C*b* is denominated as the quantization error and this is one of the components subject to minimization in the objective function – described in detail in chapter 4.4. Another form of storing the binary codes is to convert them into the integer format, shown as well as integer code in figure 15.

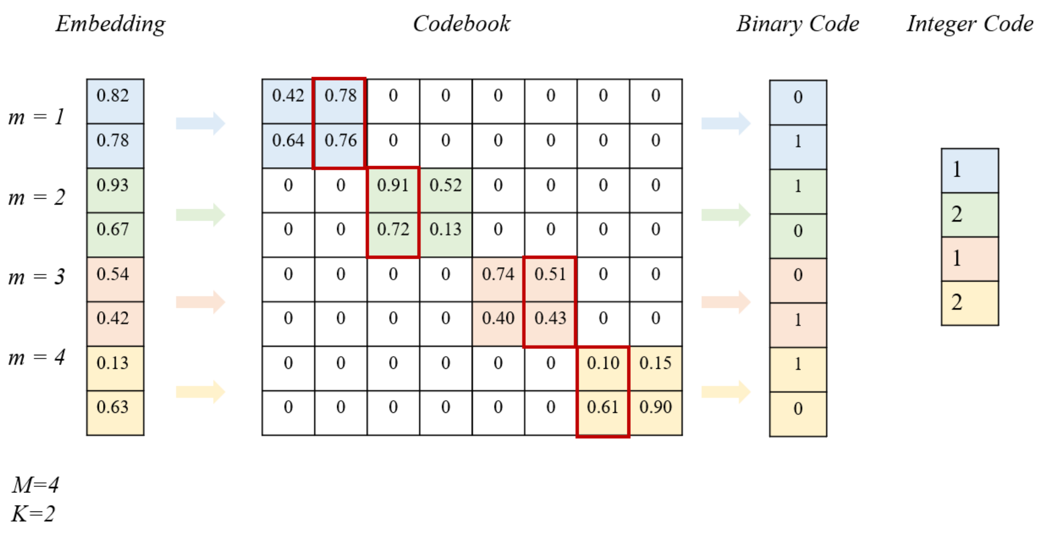

Figure 16 – Product Quantization of Embedding

4.3.2 Retrieval of Neighbors

One subject that is primarily focused on the productization of the model is the manner of retrieval of neighbors after the input image has been provided. Since the catalog of a retailer is going to be fully processed beforehand, the codes of every image for each product will be stored in a database.

The idea of compacting into a binary code is advantageous on the nearest neighbor search due to its low memory footprint when compared with the embeddings and the speed of computing the nearest neighbors. Jégou et al. (2011) propose two different approaches based on the distance computation between the query vector *x* and the quantized codes on the database *q(y)* for the computation of the nearest neighbors.

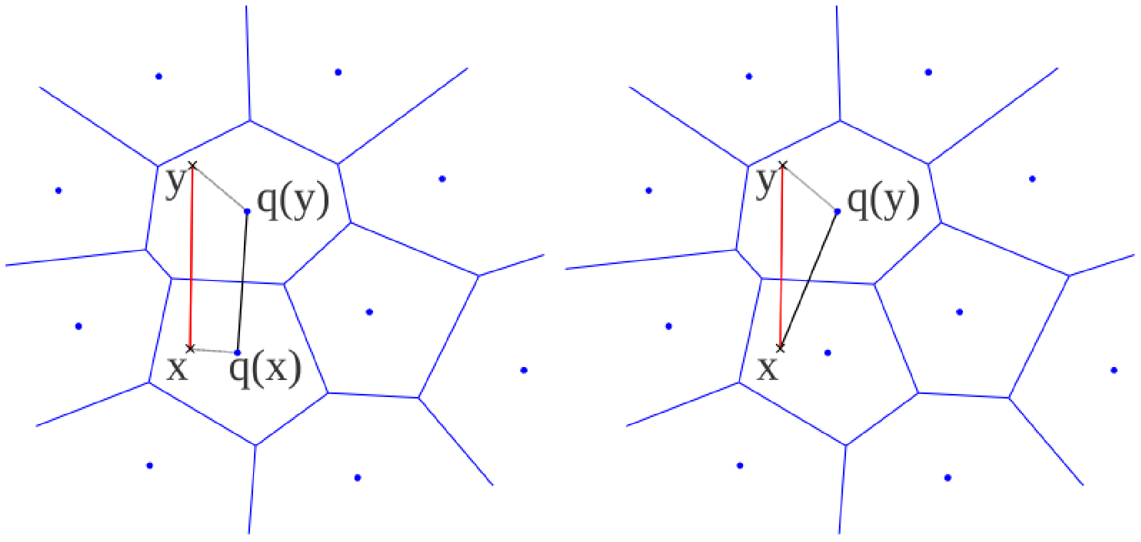

Figure 17 – Symmetric and asymmetric distance computation

The symmetric distance computation performs the quantization of the query vector *q(x)* and calculates the similarity between the quantized query and the quantized codes – represented in the left image of Figure 16. The Asymmetric distance computation computes the same without quantizing the query vector. The authors argue that, while slightly more costly than the symmetric computation, the asymmetric one achieves lower quantization distortion – reason why the main focus for retrieval results on chapter 5 will go for the asymmetric one. For both cases, the similarity calculation is executed by applying the inner-product between the query vector, quantized or not depending on the metric, and the quantized database codes.

4.4 Objective Function & Training

The last step of the implementation concerns the definition of the objective function of the model, where the network will iteratively train in order to minimize the function. For this case, the objective function can be divided into *triplet loss* - which translates the separability between positive and negative embeddings from the anchor - the *quantization error* - where the error associated with the synthetization of the embeddings is reduced - and the *orthogonality constraint*. After the definition of this function, the training stages and process is defined. In this section, the same notation as the one present in DTQ (B. Liu et al., 2019) is used since this formulation was based on this work.

4.4.1 Triplet loss

At the preparation stage, numerous triplet samples, N_t, were created, where each sample is composed of a positive, a negative and an anchor. Considering each sample from i to N_t, and each of the three components, each sample is then respectively represented as z_i^p, z_i^n and z_i^a, where z refers to the embeddings of the image to be processed obtained at the last stage before the quantization layers.

For this project, the relative similarity measure is defined by computing the Euclidean distance between the positive and negative images and the anchor one, where positive-anchor (p-a) distance should be smaller than the negative-anchor (n-a) one. To avoid an absolute similarity relation between samples, a parameter δ is also introduced as a relaxation, where the positive example is only required to be more similar than the negative by a specific margin, enforcing the case of a relative similarity process instead of an absolute similarity relation.

The formulation described is present in Equation 1, this being the first component of the overall objective function, represented by L. The application of the difference margin is applied by retaining the maximum value between 0 and the overall triplet distances, where if the value is below 0 then it is not considered.

$$L = \sum_{i=1}^{N_t} L_i = \sum_{i=1}^{N_t} \max\left(0, \delta - \|z_i^a - z_i^n\|^2 + \left\|z_i^a - z_i^p\right\|^2\right) \quad (1)$$

4.4.2 Quantization error

When performing an hashing process, the main trade-off is to reduce the size of the data while losing the least information possible from data. In quantization, this is described as the quantization error, being this the part approached by the second term of the objective function here described on Equation 2.

$$Q_1 = \sum_{i=1}^{N_t} \sum_{*\in\{a,p,n\}} \left\| z_i^* - \sum_{m=1}^{M} C_m b_{mi}^* \right\|^2 \tag{2}$$

As seen in chapter 4.3, the generation of binary codes is performed by performing the Euclidean distance on the difference between the embeddings from the last layer of the core CNN, z_i^*, and the vectorial product between the codebook and the code to be generated, $C_m b_{mi}^*$. This is transposed into the objective function, where the minimization of the norm grants the convergence of the generation of binary codes into more fitting representation of the extracted embeddings.

4.4.3 Weak-Orthogonality Constraint

While the triplet strategy is performed in an effort to perform effective similarity searches, the efficiency of the network to generate binary codes is dependent on the codebook used to generate them. As seen, several authors have directly imposed an orthogonality constraint to avoid code redundancy and improve the code compactability.

To enable a more flexible structure to be created, B. Liu et al. (2019) modify this constraint into a quantifiable component of the objective function by transforming the orthogonality relationship where $\mathrm{A}^{\mathrm{T}}\mathrm{A} = \mathrm{I}$, being A a general matrix and I the identity matrix. In this transformation, if A is orthogonal then $||\mathrm{A}^T\mathrm{A} - \mathrm{I}||^2$ is added into the objective function. The minimization process will converge this component into the lowest possible value, but in this case, one can control the degree of orthogonality, with the parameter γ to impose rather than being a binary decision. Therefore, the model evolves in the direction of maximizing the orthogonality of the codebook and thus its efficiency, without restricting it to be entirely orthogonal – Equation 3.

$$Q_2 = \gamma \sum_{m=1}^{M} \sum_{m'=1}^{N} ||C_m^T C_{m'} - I||^2 \tag{3}$$

4.4.4 Training process

Combining all the components of the objective function, the function to be minimized, in Equation 4, is then composed by the triplet loss and the quantization loss which, by itself, includes the quantization error and the orthogonality constraint and is affected by an hyper-parameter λ. Please note that Q = Q_1 + Q_2.

$$\min_{\theta, C, B^*} \quad L + \lambda Q \tag{4}$$

From this, the training of the model is then divided into three main sets of variables that will be updated iteratively - during the update process of each of them, the remaining ones are static. The first concerns the deep convolutional parameters, $\theta,$ and it will be optimized via back-propagation. The second involves updating the codebook, C, through the adoption of the gradient descent. In this case, $C = C - \eta \frac{\partial Q(C)}{\partial C}$, where η is a learning rate. The third and last component regards the update of the binary codes B^*. However, this process is not straightforward to implement on a gradient descent method, because the generation of binary codes implies a discrete optimization to be performed, instead of a continuous one. Therefore, given that each code of each image is independent of the remaining ones, the problem is analysed as a high-order Markov Random Field problem. As these problems tend to be an NP-hard one, this is addressed by applying the Iterated Conditional Modes algorithm that solves each of the subspaces in M alternatively, guaranteeing the convergence to local optimum and thus an effective binary code to be created.

5 Results & Analysis

The following chapter describes the results after training the model and the corresponding analysis between the triplet quantization method, named *TQ*, and the triplet based without quantization, named as *TNQ* – using Xception as the same common feature extractor. Both these models were created under the same resources, training datasets and common parameters to best reflect the loss in information from the quantization process. For TQ, the nearest neighbors search is performed via ADC and SDC as described in chapter 4.3.2. For the TNQ, given that the embeddings are used directly has obtained by the second to last layer, the computation of neighbors is performed via Euclidean distance using FAISS (Johnson et al., 2019) enabling fast, GPU based, nearest neighbor search for large scale data retrieval.

The settings of the parameters referred along this document are the same as the ones specified in DTQ (B. Liu et al., 2019) with the exception of the number of epochs – set to 20 due to time constraints – and the number of subspaces – here tested for *M=4,6,8*.

Due to the subjective nature of similarity and that the challenge at hand concerns a more specific niche of analysis - the fashion apparel industry - the scope of analysis of this project was divided into three main subjects. The first, the most objective yet restrict one, concerns the mean average precision on the recommendations provided by the algorithm for the first result obtained by the nearest neighbor search. The second refers to the creation of a practical use case with a retailer, where a human sample and recommendations provided by the algorithm were both subject to external analysis by other people regarding the quality of the recommendations. The third comparison is the A/B testing the algorithm in ecommerce platforms of retailers to assess (i) the variation of results versus the non-quantized model and (ii) how both of these compare with a different kind of recommendations system.

Apart from the first benchmark, this analysis was performed in association with ShopAI, the company that provided the data for the training of this model, and with several retailers that are customers of it. Therefore, the last two components will only reflect the variations in comparison with a baseline in order to maintain the anonymity and data protection of business metrics both from ShopAI and its customers.

5.1 Accuracy Analysis

In order for TQ to be evaluated, the conditions of analysis must contemplate the scenarios of online and offline recommendations. For the first, the metric can be divided into recommendations (i) having the same category at the input query, (ii) having the same category and pattern of the input query and (iii) finding the exact product in store. For the second, the same conditions apply with the exception of the same product analysis, since the offline case is not supposed to recommend the same product.

Analysing Table 3, it can be concluded that the SDC performs worse than ADC for all the conditions defined, following the behaviour described by Jégou et al. (2011), where the quantization distortion is higher in the SDC. Additionally, it is also observable that TNQ is better than TQ – 64 bits by an average of 3.6 and 7.3 p.p for ADC and SDC respectively, as a result of the quantization distortion produced by the additional layers.

From another perspective, it is also seen that, for both TNQ and TQ the precision is lower on the Street query images than on the In Shop – explainable by the fact that street images have more noise. Nonetheless, it is seen that higher number of subspaces increases the precision across all conditions, though with a maximum difference of just 0.8 and 0.2 p.p. for ADC for both Street and In Shop conditions respectively, from 32 bits to 64 bits.

Since the remaining tests were performed on real life conditions with human teams and ecommerce platforms from retailers working closely with ShopAI, only the ADC will be continued to be analysed since it performed best in this benchmark.

Table 3- SDC mAP of TQ vs TNQ (%)

Conditions	Metric	TQ - 32 bits	TQ - 48 bits	TQ - 64 bits	TNQ
In Shop - Category	SDC	90.5	90.9	91.6	96.5
In Shop - Pattern	SDC	80.7	82.8	84.1	87.5
Street - Category	SDC	69.4	73.1	75.5	82.5
Street - Pattern	SDC	58.9	61.4	63.6	72.1
Street - Exact product	SDC	45.7	47.3	49.5	62.3
In Shop - Category	ADC	93.0	93.1	93.2	96.5
In Shop - Pattern	ADC	85.5	85.6	85.8	87.5
Street - Category	ADC	78.5	78.9	79.3	82.5
Street - Pattern	ADC	68.1	68.3	68.6	72.1
Street - Exact product	ADC	55.2	55.5	56.0	62.3

5.2 Practical Use Case

To better reflect human perspective, 4 samples of 1000 images each - 750 from In Shop images and 250 from Street view - was gathered, identified and subject to the nearest neighbors search by the 3 configurations of TQ and TNQ. With the same distribution, 1000 manually curated recommendations were also produced by retailers belonging to ShopAI customer base. Afterwards, these sets of recommendations were evaluated by an additional team, whom it was asked to classify the quality of the recommendations from *Very Poor* to *Very Good*, as shown in Table 4. It can be concluded that no significant deviation occurs between the different configurations of TQ, improving only from *Medium* cases to *Good*. Additionally, TQ -64 bits for *Very Good* achieves 93.8%, 2.9 p.p behind TNQ and 4.6 p.p behind Human Team.

Table 4- Human evaluation of TQ vs TNQ vs Human Team (%)

Classification	TQ - 32 bits	TQ - 48 bits	TQ - 64 bits	TNQ	Human Team
Very Poor	0.2	0.2	0.2	0.0	0.0
Poor	0.0	0.0	0.0	0.0	0.0
Medium	2.4	1.9	1.6	0	0.1
Good	3.7	4.2	4.4	3.3	1.5
Very Good	93.7	93.8	93.8	96.7	98.4

While the results support the quality of the model at the level of human perception, the overall value of 96.7% of Very Good for TNQ is intriguing, since TNQ achieved only 96.5% for In Shop and 62.3% for Street images of the same category – the minimum baseline considered for similarity. Upon further examination on the dataset, several examples emerged where images belonging to different categories were classified as Very Good – Figure 16 (a) as example. This occurred because many images were cropped, cutting essential details on the images that would otherwise seem more different if analysed as a whole. This enforces the case of relative similarity and is in fact desired to strengthen the robustness of the model regarding lower quality or defective query images.

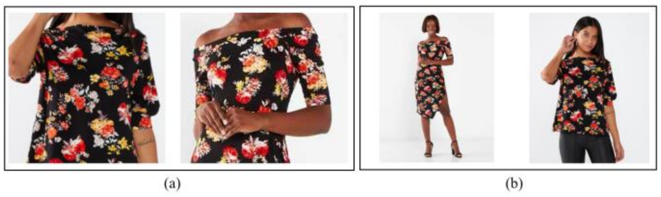

Figure 18 – (a) Cropped images, (b) Unedited images

5.3 A/B Testing

Another benchmark performed to assess the results of TQ concerned the direct application of it in a business ecommerce platform and test how it performed on the level of (i) seeking the customer’s attention and (ii) converting the attention to a sale, generating revenue. For the first, the metric created was the *Session to Click* and it is the ratio between the number of times a customer clicks on an item from the recommendations provided by the model after landing on a product page and the number of times a user lands on said page – this is shown chapter 3.3, Figure 13. For the second, the metric defined was *Click to Purchase* and it is the ratio between the number of times a sale is made after clicking in a recommendation and the number of times a customer clicked on a recommendation. The product of these two metrics translates the number of sales a customer buys an item generated by the model after landing in a product page.

The test was performed by randomly generating two sets of recommendations, one provided by either TNQ or TQ – 64 bits and another provided by a typical recommender system as a Baseline Recommender. For this system, the principle is to show items that people bought together with the item present in the product page. These two sets are then displayed in the product page in a specific order – either TQ or TNQ first and the Baseline Recommender second or vice-versa, as seen in Figure 19. This reversion in the order is due to the fact that customer attention is highly subjective and volatile, therefore the place of exposure of the recommendations highly influences its success. By mixing the order of the recommendations, the goal is to maintain a fair comparison of the metrics described.

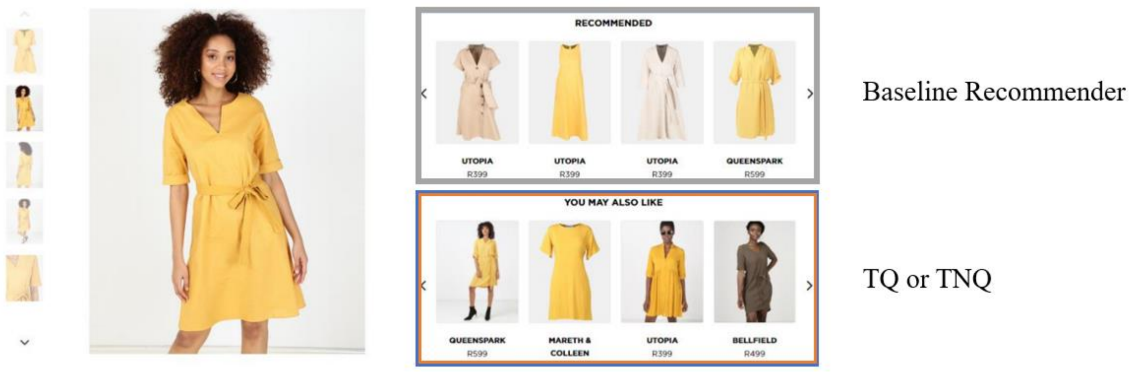

Figure 19 – Product Page with different recommender systems

For this A/B testing, four different fashion retailers were considered, each of them with an apparel style different from the others, as well as the respective geographical markets – for example, smart-casual retailer in South Africa vs. boutique fashion in England. Consequently, the benchmark analysis is broader and more robust, since several retailers are considered with different cultural and sociological standards.

These tests were executed directly on the production website of these retailers, hence the belowmentioned results report actual business results from said brands over a period of 3 months. As such, for the sake of anonymity and disclosure protection, the results were modified into relative terms - the Baseline Recommender will be regarded and the standard and results of TNQ and TQ are a multiplicative over the baseline.

It can be seen on Figure 20 that both TNQ and TQ gather more attention from the end user when compared with the baseline. Retailer C is where this difference is more pronounced, with both TQ and TNQ performing 45% and 49% better than the baseline and D is the least significant case where the increase is only 4% and 5% respectively. Nonetheless, A and B are the larger size catalog retailers with a more distinct number of products in their range as opposed to C and D, suggesting that a conservative average increase of +17% is a more realistic conclusion to be drawn concerning the performance of the model on the *Session to Click*. Between TQ and TNQ, it can also be observed that the maximum difference is 4 p.p. for case C, but only 2-3 p.p for the higher sized catalog A and B - averaging 2.5 p.p. - suggesting that the addition of the quantization process only marginally decreases the results obtained by the model.

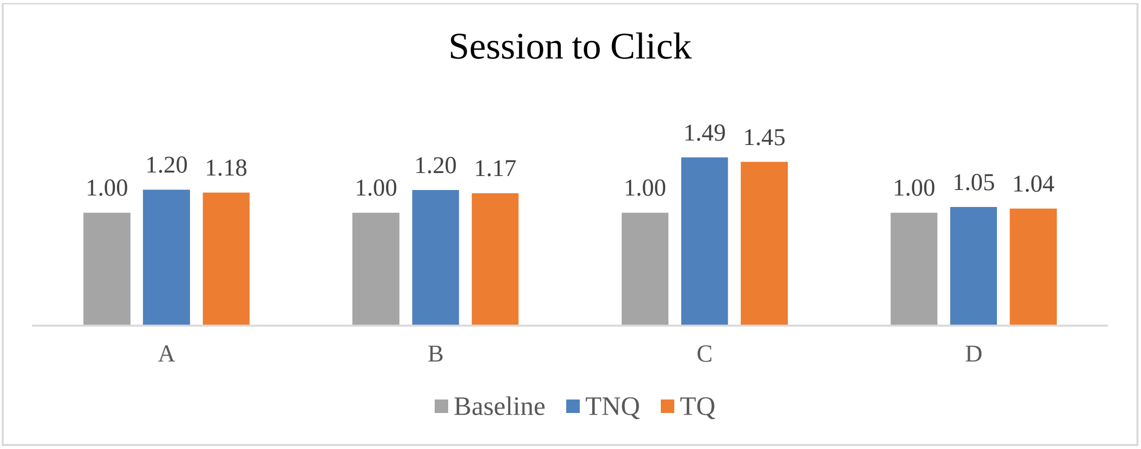

Figure 20 – A/B Testing – Session to Click results

Concerning the Click to Purchase metric displayed in Figure 21, the results again point towards similarity based recommendations having a better performance than shopping pattern association rules, with TQ and TNQ performing, at the minimum, 10 p.p. better than the baseline. Brand C continues to demonstrate the highest variation but on brand D, previously the least prone towards similarity recommendations, the increase is of 33% for TNQ and 36% for TQ model. An interesting observation is that brand A, the largest retailer, now stands out at 31% and 32% increase for TNQ and TQ respectively.

Contrary to the previous results, this testing shows that TQ only performs worse than TNQ for brand B, with an average variation of +0.5p.p. over TNQ across all brands. Given that the addition of quantization layers should not improve the overall similarity accuracy of the model and that TNQ is basically the core extractor of TQ, the hypothesis is that this slight edge by the TQ should not be regarded as an advantage over TNQ, but rather as a consequence of particular observations on the buying pattern on the users. Therefore, additional A/B testing for longer periods could strengthen the evidence shown in the results, since the variation is so small it was not deemed necessary.

Concerning the notoriously higher results of TNQ and TQ over the baseline, the hypothesis discussed with the retailers is that while browsing a specific product page, the user is still focusing on which item to buy instead of searching for additional items to add to the shopping bag. Therefore, delivering similar results would prove useful on this search while providing other bought together items could add entropy to the decision process of the user and thus producing lower results than similarity based methods.

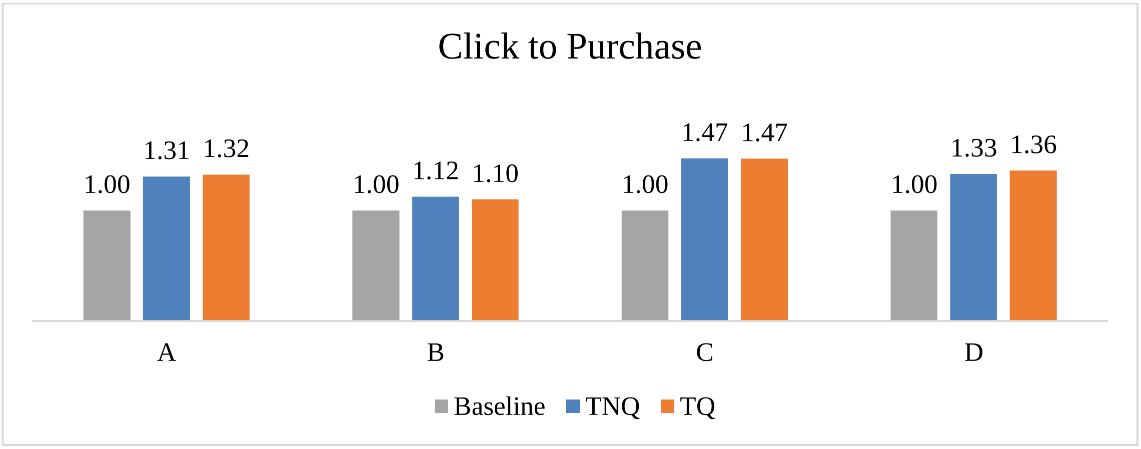

Figure 21 – A/B Testing – Click to Purchase results

6 Conclusions & Future Works

At the start of this project it was mentioned that the subject of visual search has been improving to the point that several ecommerce platforms already provide some kind of interface to allow their users to submit a photo and retrieve similar results to the ones provided. It was also seen that the quality of these results has already been developed to a level where, even for images with noisy background, the similarity engines are robust enough to provide useful recommendations. This has been achieved thanks to deep neural networks whose image representation onto an array of values through which a distance metric can be applied to effectively find the best matching images. In this sense, the challenge is to be able to maintain or improve the quality of these engines, while minimizing the memory footprint of the embeddings and, consequently, reduce computational necessities and its associated high infrastructural costs.

In this sense, three main questions were defined as the main guidelines for this report, all of which were approached and analysed in detail. Concerning the selection of the most appropriate CNN architecture for image feature extraction, the selection relied on Xception. The reasoning behind this decision was the overall satisfactory results of this network regarding mean average precision on ImageNet, number of parametes, floating point operations, memory consumption and response time. However, the main driver of this decision was its inclusion in the Keras library due to the objective of capitalizing on pre-existing models to perform transfer learning for similarity purposes. This enabled a faster development and testing at the expense of not being able to use other models who perform better, such as de Squeeze and Excitation networks and the ResNeXts.

Regarding the most effective strategy for similarity search, no further analysis was performed in this project since several authors have already thoroughly investigated that triplet based training currently enables the best results on relative similarity across several fields. Nonetheless, a triplet based model, with and without quantization, were analysed in this report concerning the human evaluation of its results – instead of just comparing numerical metrics as mean average precision – and the overall status is *Very Good* as seen in chapter 5.2. This was further highlighted with the business analysis of applying these models in production environments of real retailers to analyse its impact regarding customer's retention and conversion. This could be considered the paramount analysis since the main objective of visual search engine in ecommerce is to enable another channel of revenue. If a customer proactively searchs for an item, obtains a set of similar products and ends up buying one of the recommended items - though it is not proof of being the most similar item in all cases - it can be considered as a successful outcome. In this comparison, it was also observable that triplet similarity based recommendations achieved better results both on customer retention and conversion rates.

On a analogous approach, the analysis of the most effective dimensionality reduction process was also performed by studying several techniques in literature. Several studies have shown that learning to hash functions perform better than offline reduction methods. This methodology is further strengthen by the implementation of learning to hash functions directly on a neural network to combine the optimization of both the quantization error and the similarity/classification task. In this line of thought, and due to the reported state-of-the-art results concerning quantization processes, the model described in this document included a relaxed quantization process with weak orthogonality constraint. Though not targeted in this work, other works concerning triplet similarity in fashion (Shankar et al., 2017) have implemented other methods such as LSH and observed an accuracy drop of over 10 p.p., much worse than the results obtained. Here again, the results observed in chapter 5.1 for asymmetric distance calculation show that the overall loss in accuracy due to quantization distortion averages only 2.5 p.p. for offline recommendations and 4.3 p.p. for online ones - for a code size of 64 bits. Nonetheless, on chapter 5.2, it was seen that the variation of the binary code size has marginal effect on the overall similarity evaluation. Moreover, considering an embedding size D = 1024 - each element being a 32 bit representation - the memory footprint of the quantized codes is 512 times lower than the embedding for a 64 bit representation and 1024 for 32 bit representation – the latter only marginally worse than the first. Therefore, the model managed to significant lower the memory consumption of data by several orders of magnitude while minimizing the similarity quality of the recommendations.

This being stated, it can be concluded that a successful implementation of a similarity based model was executed, capable of producing high quality and compact image codes.

Lastly, regarding fututre developments, there are two main areas to address. The first, already mentioned in this chapter, concerns the application of better pre-trained networks for the core of the model. Both Squeeze and Excitation networks and ResNeXts have shown to perform better than other models both on accuracy and efficiency terms. Moreover, the latest results obtained from EfficientNet – achieving state-of-the-art results both performance and accuracy wise – have drawn the attention of several researchers concerning its process of iteratively analysing its depth, width, and resolution. This goes against the commonly developed approaches where the architectures are defined at the start and then trained to obtain better results via parameter tuning. Additionally, this model is currently being implemented in Keras - easing its testing in the framework described in this work - thus sparkling interest in using it in future works.

The second concerns the dimensionality reduction analysis of this report. The main goal of achieving a more efficient method with minimal distortion in the results was successfully achieved but only one neural quantization process was effectively created for this project. While traditional methods have shown to produce inferior results, and thus the importance of comparing them in this dissertation is minimal, other neural quantization methods are available that could also produce satisfactory results. Moreover, the model had to be trained for every subspace configuration, which greatly occupied the computational and time resources available for this project. In conclusion, more neural hashing architectures could be implemented and compared in the future, with particular focus on the DRQ, in which several subspace configurations can be performed in the same training stage.

References

Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, *18*(9), 509–517. https://doi.org/10.1145/361002.361007

Bianco, S., Cadène, R., Celona, L., & Napoletano, P. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. *CoRR*, *abs/1810.0*. http://arxiv.org/abs/1810.00736

Boureau, Y.-L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 111–118.

Cao, Y., Long, M., Wang, J., Zhu, H., & Wen, Q. (2016). Deep quantization network for efficient image retrieval. *Thirtieth AAAI Conference on Artificial Intelligence*.

Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. *ArXiv E-Prints*, arXiv:1610.02357.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Li, F. F. (2009). ImageNet: a Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Llados, J., & Pal, U. (2017). SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification. *ArXiv E-Prints*, arXiv:1707.02131.

Ge, T., He, K., Ke, Q., & Sun, J. (2014). Optimized Product Quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(4), 744–755. https://doi.org/10.1109/TPAMI.2013.240

Gong, Y., Lazebnik, S., Gordo, A., & Perronnin, F. (2012). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(12), 2916–2929.

Goswami, A., Chittar, N., & Sung, C. H. (2011). A study on the impact of product images on user clicks for online shopping. *Proceedings of the 20th International Conference Companion on World Wide Web*, 45–46.

Guo, J., & Li, J. (2015). CNN Based Hashing for Image Retrieval. *ArXiv E-Prints*, arXiv:1509.01354.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv E-Prints*, arXiv:1512.03385.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hoffer, E., & Ailon, N. (2014). Deep metric learning using Triplet network. *ArXiv E-Prints*, arXiv:1412.6622.

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *ArXiv E-Prints*, arXiv:1608.06993.

Indyk, P., & Motwani, R. (1998). Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 604–613. https://doi.org/10.1145/276698.276876

Jégou, H., Douze, M., & Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 117–128. https://doi.org/10.1109/TPAMI.2010.57

Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J., & Tavel, S. (2015a). Visual search at pinterest. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2783258.2788621

Jing, Y., Liu, D., Kislyuk, D., Zhai, A., Xu, J., Donahue, J., & Tavel, S. (2015b). Visual search at pinterest. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1889–1898.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*.

Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *ICML Deep Learning Workshop*, *2*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. https://doi.org/10.1145/3065386

Kumar B G, V., Carneiro, G., & Reid, I. (2016, June). Learning Local Image

Descriptors With Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lai, H., Pan, Y., Liu, Y., & Yan, S. (2015). Simultaneous feature learning and hash coding with deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3270–3278.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Li, W.-J., Wang, S., & Kang, W.-C. (2015). Feature Learning based Deep Supervised Hashing with Pairwise Labels. *CoRR*, *abs/1511.0*. http://arxiv.org/abs/1511.03855

Liu, B., Cao, Y., Long, M., Wang, J., & Wang, J. (2019). Deep Triplet Quantization. *CoRR*, *abs/1902.0*. http://arxiv.org/abs/1902.00153

Liu, T., Moore, A. W., & Gray, A. (2006). New Algorithms for Efficient High-Dimensional Nonparametric Classification. *J. Mach. Learn. Res.*, *7*, 1135–1158.

Machado, R. P. da S. R. (2017). *Image visual similarity with deep learning: application to a fashion ecommerce company*.

Maheshwary, S., & Misra, H. (2018). Matching Resumes to Jobs via Deep Siamese Network. *Companion Proceedings of the The Web Conference 2018*, 87–88. https://doi.org/10.1145/3184558.3186942

McNames, J. (2001). A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*, 964–976.

Norouzi, M., & Fleet, D. J. (2013). Cartesian k-means. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3017–3024.

Rao, D. J., Mittal, S., & Ritika, S. (2017). Siamese Neural Networks for One-shot detection of Railway Track Switches. *ArXiv E-Prints*, arXiv:1712.08036.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–823.

Shankar, D., Narumanchi, S., Ananya, H. A., Kompalli, P., & Chaudhury, K. (2017). Deep Learning based Large Scale Visual Recommendation and Search for E-

Commerce. *ArXiv E-Prints*, arXiv:1703.02344.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv E-Prints*, arXiv:1409.1556.

Song, J., Zhu, X., Gao, L., Xu, X.-S., Liu, W., & Shen, H. T. (2019). Deep Recurrent Quantization for Generating Sequential Binary Codes. *CoRR*, *abs/1906.0*. http://arxiv.org/abs/1906.06699

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *ArXiv E-Prints*, arXiv:1602.07261.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *ArXiv E-Prints*, arXiv:1409.4842.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *ArXiv E-Prints*, arXiv:1512.00567.

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR*, *abs/1905.1*. http://arxiv.org/abs/1905.11946

Van Der Maaten, L., Postma, E., & den Herik, J. (2009). *Dimensionality reduction: a comparative*.

Wan, J., Wang, D., Hoi, S. C. H., Wu, P., Zhu, J., Zhang, Y., & Li, J. (2014). Deep learning for content-based image retrieval: A comprehensive study. *Proceedings of the 22nd ACM International Conference on Multimedia*, 157–166.

Wang, Jiang, song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., & Wu, Y. (2014). Learning Fine-grained Image Similarity with Deep Ranking. *ArXiv E-Prints*, arXiv:1404.4661.

Wang, Jingdong, Shen, H. T., Song, J., & Ji, J. (2014). Hashing for Similarity Search: A Survey. *ArXiv E-Prints*, arXiv:1408.2927.

Wang, Jingdong, Zhang, T., Song, J., Sebe, N., & Shen, H. T. (2016). A Survey on Learning to Hash. *CoRR*, *abs/1606.0*. http://arxiv.org/abs/1606.00185

Wang, X., Shi, Y., & Kitani, K. M. (2016). Deep Supervised Hashing with Triplet Labels. *ArXiv E-Prints*, arXiv:1612.03900.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1), 37–52. https://doi.org/https://doi.org/10.1016/0169-7439(87)80084-9

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2016). Aggregated Residual Transformations for Deep Neural Networks. In *arXiv e-prints*. https://ui.adsabs.harvard.edu/abs/2016arXiv161105431X

Yu, T., Yuan, J., Fang, C., & Jin, H. (2018). Product quantization network for fast image retrieval. *Proceedings of the European Conference on Computer Vision (ECCV)*, 186–201.

Zhai, A., Wu, H.-Y., Tzeng, E., Huk Park, D., & Rosenberg, C. (2019). Learning a Unified Embedding for Visual Search at Pinterest. *ArXiv E-Prints*, arXiv:1908.01707.

Zhang, T., Du, C., & Wang, J. (2014). Composite Quantization for Approximate Nearest Neighbor Search. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, II–838–II–846.

Zhang, Y., Pan, P., Zheng, Y., Zhao, K., Zhang, Y., Ren, X., & Jin, R. (2018). Visual search at alibaba. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 993–1001.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2019). A Comprehensive Survey on Transfer Learning. *ArXiv E-Prints*, arXiv:1911.02685.