Jewellery Recommendation System for e-Commerce using CBIR

02466 Project Work for
Artificial Intelligence and Data

Summer 2021

Technical University of Denmark
DTU Compute

Submitted June 21, 2021
Version 2. June 25, 2021

Authors
Alma Fazlagic s194271
Andreas Lau Hansen s194235
Anton Thestrup Jørgensen s194268
Natasha Graae Norsker s194270

Supervisors
Morten Mørup mmor@dtu.dk
Laura Perge lp@ecapacity.dk
Nikolaj Juul Madsen njm@ecapacity.dk

Contents

Foreword

This project was done as a 4th semester bachelor course in the spring of 2021.

All code is available at our Github:

https://github.com/natashanorsker/fagprojekt

Acknowledgements

Thank you to Laura, Nikolaj and Morten for helping us through the project. Thank you to Niels Aske Lundtorp Olsen for help regarding statistical analysis.

Changes for version 2

Table 7, 8, Figure 25, and Figure 26. Were corrected since the random method was incorrectly reported to perform much better than it actually was. Added appendices for convergence of model training and examples of triplet pairings during training. Small mistakes about the trace in proof 7 haved been fixed.

Glossary

HSV — Hue, Saturation, Value. A colour space to represent images in. Colours are represented as a cylinder with hue as an angle to change the colour, saturation is the intensity of the colour and value is the brightness. This colour space is often used for its superior ability to differentiate between objects.

CBIR — Content based image retrieval. The act of retrieving images from a database based on the content in the image.

TRN — Triplet Ranking Network. A Deep Neural Network implemented with a triplet loss.

Wild Images — Images taken by the end user. They are likely to contain noisy backgrounds, bad lighting, and a partially obscured object out of focus.

Catalogue images — Professionally photographed images from the Pandora product catalogue. The jewellery is either photographed on a uniform white background or on a model. For many purposes in this paper, model images are treated as wild images.

Embedding — also called latent representation, a compact vector representation of a high dimensional data object e.g. an image. Embeddings are typically used in conjunction with an embedding space where each axis is a "feature".

VAE — Variational Autoencoder.

Abstract

In the world of e-commerce, providing accurate product recommendations which align with customer needs can create a serious advantage over competitors. This report investigates how instead of searching with keywords when trying to retrieve a wanted product, the consumer can search with an image instead and be presented with the most similar products. This is the idea behind Content-Based Image Retrieval (CBIR) systems, which help solve the consumers problem of ambiguity when searching only with key-words.

CBIR systems have already been implemented within certain fashion sites, however a CBIR for jewellery poses several other challenges due to the fact that different pieces of jewellery are usually very visually similar to the untrained eye.

This imposes a need for measuring similarity that goes deeper than the category level separation, which is achieved by others in the field via a host of metric learning techniques, where images are grouped in structures along with information on whether they are similar or not. This approach tends to group images that are semantically similar, which translates to accurate recommendations.

To be able to compare different approaches, we collected a dataset of e-catalog images from several Pandora Jewellery sites, which mainly consists of well lit pieces of jewellery on a plain background.

We propose two models for achieving these recommendations. Firstly, a Variational Autoencoder which seeks to learn a suitable representation by looking at how well it can reconstruct images from the latent space. Secondly, a Triplet Ranking Network, which pulls and pushes positive and negative images from an anchor image defined in a large number of image triplets, which should be sampled such that they approximately represent the similarity demands of the user.

To handle challenging input images, we incorporate an image segmentation model to crop and rescale images, while replacing the background with an off-white one resembling that of the e-catalog training data.

The systems were scored using the mean average precision, cumulative matching coefficient, and recall at rank K metrics. With these metrics we found that the Triplet Ranking Network was superior to a Variational Autoencoder. The opposite was observed in a qualitative evaluation through human experiments, where we found the Variational Autoencoder to perform better than the deep-ranking method with statistical significance. Nevertheless they both return useful recommendations that significantly outperform a random recommendation baseline.

As a complete system, the methods we present has the potential to provide a significant value increase to the searching and discovery process when shopping for jewellery.

1 Introduction

Business to consumer e-commerce is an immense economic entity with a projected share of 23.3% of the total global retail sale in 2024 [Tug21] corresponding to approximately 6.5 trillion US dollars.

Without the need for physical storage to display products, e-commerce sites can have substantially more products in their store. This creates a problem when the customer seeks to locate a specific product due to the sheer amount of items to search through.

Humans primarily use visual cues when determining what product they want to buy when entering a brick-and-mortar store. However on the majority of e-commerce sites you are forced to use a key-word matching system to retrieve the wanted products. The output of such retrieval systems are not always ideal, since there can be large semantic gaps between the text description and the more abstract visual features of a product. Furthermore, it is not always clear what keywords will work well on a specific site.

A more intuitive way of searching through products would be to upload an image of something similar to what you are searching for. This query image could then be compared to the product images on the e-commerce site and return the top 'X' most similar products. When the recommendations are based on the content of the images, it is most commonly referred to as a content-based image retrieval system (CBIR system).

In recent years, there has been a rapid development in the creation of these CBIR-systems. CBIR is a technique that extracts the visual features such as colour, texture and shape from the images themselves and uses these features to find the most similar images. It has two major components: the image embedding, or the representation of the image in a latent space, and the similarity measure used for database search.

A well-performing CBIR must be robust to the variations coming from orientation, scaling and differing backgrounds that can be expected from real user captured images. This makes an inexpensive procedure such as pixel-wise matching ineffective for use in CBIR. Furthermore a major challenge of retrievals based on query images is the large distinction between the low-level visual features and the high-level meaning of the image.

There is a growing need for effective CBIR systems to diminish the gap between locating products in a regular store and on an e-commerce site. In this report, we will target CBIR for jewellery specifically. Although several leading e-commerce sites in the fashion industry are starting to implement such image retrieval systems, there has not been a lot of movement in the jewellery industry alone. Notable mentions of well-implemented CBIR include ASOS' Style Match [Nik17], Alibaba Cloud image search [Zha+21] and Pinterest's Visual Search Tool [Jin+15]. One of the main differences from the sites mentioned prior and jewellery specifically lies in the similarity between 'in-category' and 'out-of-category' images.

1.1 Statement of Problem

In this report, the following problem statement will be investigated:

How can we create an intelligent system for ranking jewellery products from an e-catalogue based on their visual similarity with a user-generated query image such that the ranking reflects the user's needs?

Within this problem statement, we will examine the following research questions:

- *Does a Deep Ranking model produce more semantically meaningful embeddings than a Variational Autoencoder model?*
- *Does differentiating the method for sampling image triplets have an effect on the results?*
- *Is it necessary to crop jewellery out of a query image to retrieve meaningful recommendations and is there an improvement of doing so?*

2 Literature review

There are two general schools of CBIR; one which uses a craftsmanship approach utilising a bag of features (BoF) method and one which relies on deep metric learning (DML). The BoF approach has been around for much longer than the DML approach. DML seems to be increasingly popular and well-performing, but the traditional methods still perform quite well and new papers are still being published with the traditional approach. Therefore we think it is fitting to mention both approaches.

2.1 Craftsmanship approach

The BoF approach is based on the heuristics of the features that should be retrieved. In the most recent paper on this approach, the authors of [SA21] compute a feature vector using heuristics about the colour, texture, and shape, which are then be used with an euclidean distance metric to find the nearest 'X' images. The feature coding for the colour of the image can be retrieved using uniform quantization[1] of a HSV transformed image. Textures can be retrieved using a Gabor filter[2] applied with a convolution operation on the image. For the shapes of objects, a Zernike Moment[3] based calculation is used, which relates regions of interest in an image to their roundness [pyi20]. The regions of interest have to be extracted and selected by some image analysis process using for example a BLOB analysis. Most importantly, they use the SURF (Speeded-Up Robust Features) feature descriptor. The function of this step is to bring out local features, for example if applied to the image of a butterfly it will be able to detect the patterns of rings on its wings. The algorithm starts off with interest points which are automatically selected at key points in the image (e.g. by points with high contrast), from these points the neighbourhood is represented by a feature vector [Bay+08]. When all these methods are used in conjunction with each other it is called a bag of features.

2.2 Metric Learning

Several state-of-the-art (SOTA) approaches utilises distance metric learning [Sik+20; Pas+20]. Distance metric learning attempts to learn an embedding function $f : \mathbb{R}^m \to \mathbb{R}^n$, where $m >> n$, that maps samples from the data space $\mathbb{R}^m$ to the embedding space $\mathbb{R}^n$ in such a way that the similar data points are kept close in the latent space, while the dissimilar points are pushed farther away. This is especially useful when solving a CBIR-problem, since the recommendations can then easily be retrieved based on a similarity measure between the different embedded vectors. Metric Learning is a weakly supervised approach, because it uses supervision only at a tuple level (this can be input pairs, triplets or quadruplets for instance). The tricky part of DML is that the similarity in the input space can be very complex and it might therefore be difficult to create even the weak supervision.

The methods described in the following sections are based on an iteration of the same idea.

[1] Uniform quantification is a binning / compression of the colours of the image

[2] A Gabor filter is a specially crafted image filter that is especially good at bringing out shapes

[3] In general, moments for pictures have somewhat the same meaning as they do in probability theory, in the way that the first moment is the centre of the image much like the first moment of a probability distribution is the mean.

They all define a loss function over pairs, triplets, N-pairs, or other linked structures of points using information on whether the points are similar, either via a similarity label or categorical partitioning of the training set. The methods then apply 'forces' to the points, pushing similar points closer and dissimilar ones apart. All methods described in this section minimise their loss using Stochastic Gradient Descent (SGD). In the case of the Lifted Structured Feature Embedding, it is just 1-2% behind the best method described in [Pas+20] (with the Fashion-MNIST and 'Animals with attributes 2'[4] datasets).

2.2.1 Contrastive Loss

The Contrastive Loss considers the set of all training points and demands a binary indicator of similarity between all pairs. The loss function minimises the distance between similar points and maximises that of dissimilar ones by using a distance measurement between the query image and another data point in the latent space. Because of this quality, contrastive loss has been used in CBIR to produce a latent space with sensible properties. The loss function furthermore introduces a margin parameter α that prevents solutions where the distance between dissimilar points goes to infinity.

Hadsell *et al.* [HCL06] introduce this loss function and provides an excellent analogy of how contrastive loss emulates a physical system where training points are connected with springs of varying strengths.

Öztürk [Özt21] tests the contrastive loss across five different distance methods to compare which one yields the more effective feature representation. He uses a siamese network structure, that consists of two convolutional sub-networks that are identical and share the same weights and hyperparameters. The experiment is performed on the well-known datasets MNIST and CIFAR-10 where they all perform very well and the euclidian distance metric is found to perform the best.

Deepak *et al.* [DA20] utilise the Siamese Network structure with Contrastive Loss for CBIR of MRI images with brain tumors. They achieve state-of-the-art results within this domain.

2.2.2 Triplet Network

The basic idea of a Triplet Ranking Network (TRN) is that it receives the data points in triplets which contains a query, positive (similar) and negative (dissimilar) image, such that it receives information on the relative similarity between different data points. A fully detailed explanation of the triplet loss is described in section 4.2.1.

In 2014, Wang *et al.* [Wan+14] proposed the use of triplet loss in a deep ranking network (named so because it uses deep learning) as a novel way of learning fine-grained image similarity in a CBIR system. They found that when trying to find similar images to a query image, *category-level* image similarity was not sufficient - instead the distinction of differences in the *same* category was needed, which is what the deep ranking models are able to do. Their deep

[4] Dataset with approximately 40k images of animals across 50 classes labeled with 85 numeric attributes.

ranking model achieved much better results than the SOTA deep classification models at that time.

In the following years after the findings of Wang *et al.*, the use of triplet loss was greatly investigated in the domain of Person Re-identification (ReID) with notable mentions such as [SKP15] from Google and [HBL17], where the latter researched deeper into different ways of mining the triplets in mini-batches. In 2017 the authors of [HBL17] set the SOTA on the ReID datasets with their TriNet, which employs the ResNet-50 architecture and the so-called Batch Hard sampling strategy, which we describe in section 4.2.1. The different batching strategies for triplet mining are also described and researched in full detail by the authors of [Sik+20].

More similarly to our use of CBIR for e-commerce purposes, Shankar *et al.* [Sha+17] present a large scale visual search and recommendation system using a Deep CNN architecture implemented with a triplet loss. On the Exact Street2Shop dataset[5] their model, VisNet, achieves SOTA results. They discuss the overall impact their model had on a e-commerce website and reported a CVR[6] of 26%, where the CVR on average was about 8-10%.

The findings of these reports motivate our use of implementing a deep ranking model with a triplet loss for a jewellery recommendation system. Furthermore, several reports such from Zeng *et al.* [Zen+20] and Passalis *et al.* [Pas+20] found that Deep Neural Networks implemented with a triplet loss performed better than those with a constrastive loss.

2.2.3 Quadruplet Network

Chen *et al.* [Che+17] propose the use of a quadruplet network instead of contrastive or triplet networks. The Quadruplet Network is based on the same idea as the Triplet Network, but incorporates a semi-relevant image as well as a positive and a negative one, resulting in quadruplet pairings as input. They suggest that triplet networks suffer from a weaker generalisation capability from training to testing than a quadruplet network would.

The quadruplet loss suggested by Chen *et al.* is a modified version of triplet loss Eq. (13) in the aspect that the first term of the loss equation is identical to the triplet loss and focuses on the relative distances between the positive and the negative images. It then adds an additional constraint to the model, that the 'semi'-negative images should be closer to the anchor image than the negative are.

The results highlight that the network produces larger inter-class variations along with smaller intra-class variation, which is shown to result in embeddings that perform better on their testing set than the embeddings from the triplet network. This approach produces *strong* as well as *weak* push strategies, since the semi-relevant images are not pushed away as strongly as the non-relevant or negative images. This differs from the triplet based approach that only considers a *strong* push strategy. The addition of the semi-relevant images also allows the designer of

[5] Dataset containing about 400,000 images from e-commerce websites alongside about 20,000 street photos of clothing items.

[6] Conversion rate - the percentage of consumers who added a similar product from the recommendations to their cart.

the network to choose them in a way that reflects the desired latent space and thus gives more freedom of adding information to the model. For our use-case, it would make sense to use images belonging to the same colour and type of jewellery as the semi-relevant images and using images belonging to the same product (augmented images) as the positive images, since it would make sure that products of the same type and colour as the query is closer to it in the latent space than other products are. Chen *et al.* furthermore shows that their quadruplet loss performs better than contrastive loss and triplet loss on three different datasets.

2.2.4 Lifted Structured Feature Embedding

As described in [Son+16], Lifted Structured Feature Embedding is a method that implements a structured loss based on all positive and negative pairs of samples in a batch such that it takes into account all pairwise edges in a batch. It is a clever way of taking advantage of mini-batching[7] to get a faster training time. The method lifts the vector of pairwise distances in the batch to a matrix of pairwise distances. A positive and an anchor is used as a pair and is found within each mini-batch. Each node in the pair independently interact with all other nodes in the mini-batch.

The lifted embedding fixes some problems that can arise from the contrastive and triplet methods. The contrastive method can fail when the randomly selected negative is collinear with examples from another class. The triplet method can fail if the negative sample is within the margin of the positive sample and the anchor. In this case the positive image is pushed towards the negative. Figure 1a shows the conceptual comparison in the way that different data-points are used as a positive or negative reference with the different types of embedding methods. Computing all the extra edges might at first glance seem to require a lot more extra computation time, however it is not as expensive as adding extra images to the input tuples as is done when going from triplets to quadruplets.

2.2.5 Alternatives to Euclidean Loss

Additional loss metrics worth noting includes Angular loss and Divergence Loss.

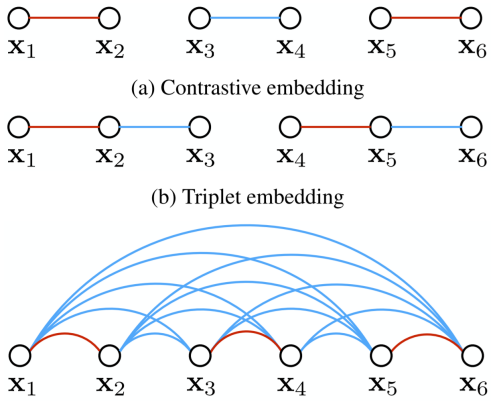
Angular Loss is introduced by Wang *et al.* [Wan+17] as an alternative to relative distances such as triplet loss or lifted structure loss. The angular loss still uses a triplet but encodes a relation with the angle at the negative vertex (image). It is scale and rotation invariant due to the use of the angular cosine distance, making it more robust to local variations. Furthermore, it can also be combined with N-pair loss.

Divergence loss [Kim+18] have several 'learners'. Each learner is encouraged to focus on different aspects of an input image through a regularising term. For example it could be that one learner focuses on pattern, another on materials, and another at the object class etc. The main benefit of this is that the embedding space will be more diverse.

[7] See [HNM19, sec. 15.4.1] for an explanation and advantages of mini-batching

2.2.6 Comparison of different method performances

Figure 1 taken from [Son+16] does a good job at illustrating the different embedding structures reviewed in the section. The main difference between contrastive loss and some of the other losses is that the contrastive loss utilises a binary set of classes (a *good* vs a *bad* image), while the other methods use a relative supervision (the positive image is *better* than the negative image). As can be seen in Figure 1 (b), the lifted structure embedding method requires just one CNN, which optimises the training time of the model significantly.

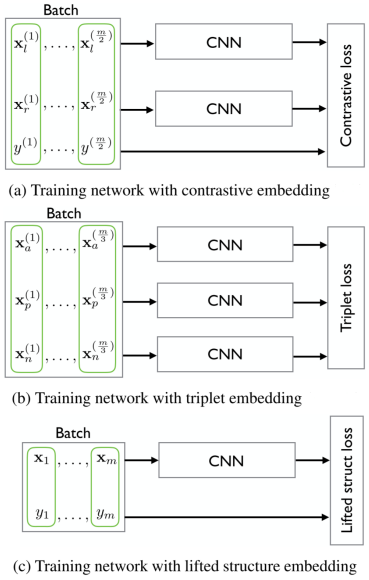


(a) Contrastive embedding

(b) Triplet embedding

(c) Lifted structured embedding

(a) Illustration of the evolving idea of contrastive embeddings compared. Red edges are are positive pairs, the same class, blue edges are negative pairs, different class. Here shown with a batch of size 6.



(a) Training network with contrastive embedding

(b) Training network with triplet embedding

(c) Training network with lifted structure embedding

(b) Network architectures

Figure 1: Comparison of the different self-supervised methods. Fig. 1a and 1b reproduced from [Son+16]

Passalis *et al.* [Pas+20] have done a comparison across Contrastive Loss, Triplet Loss, Lifted Embedding Loss and a Variational autoencoder (VAE) as well. They found the VAE performed the worst and the triplet loss performed better than the contrastive loss. Passalis *et al.* furthermore propose a method based on the idea of retaining the in-class variance in order to retain as much information as possible for a representation. Furthermore, this also allows the retrieval of objects from novel classes not seen during training. The reason for the reduction in variance stems from the fact that in the discriminative methods just mentioned a contrastive loss is used, which essentially has the effect of pulling together points of the same class.

Chen *et al.* [Che+17] discuss that in the BoF approach, the feature extractions and the similarity measurements are independent from each other, while the CNN approach that utilises deep metric learning can be globally optimised by back-propagating through the network. They highlight that since CNN's can be globally optimised it generally leads to a better performance of the end-to-end system.

Although Lifted Structured Feature Embeddings and Quadruplet Networks have been shown to outperform the Triplet Network, it can be difficult to implement when working with a

small dataset, because the positive, semi-relevant and negative images have to be sampled and defined in a way where there is enough samples in each 'category' for it to still be balanced. In this research paper, we compare triplet networks that uses different sampling strategies to a Variational Autoencoder (VAE).

3 Data

The dataset consists of catalogue images acquired from several Pandora websites[8]. Across the sites, 1716 different products were accumulated with 4 images per product on average. The images display closeups of the jewellery with consistently white-grey backgrounds. For an example of product images see Figure 2. The dataset was collected using a web scraper along with additional information about the products such as product ID's and image URLs. The images were then cropped and resized from (1000, 1000, 3) to (96, 96, 3).

The majority of product images are close-ups of the jewellery, although for some products there are images of human models wearing the jewellery as well. For the model images, the product is oftentimes worn along with other jewellery, making it hard to identify what to focus on in the image. For this reason, the model images are only used as 'wild' images for testing and have not been included in the training data. Wild images are otherwise taken by the end user and are likely to contain noisy backgrounds, bad lighting, and a partially obscured object out of focus.

Figure 2: Assembly of different images randomly sampled from the training data

Although there are 4 images per product on average, 72 products have a '360 degree spin feature' meaning that the user is able to see several angles of the products. This feature consists of 36 frames resulting in around 40 images in total for those products. This adds a skewness to the data set since the products with the 360 feature are severely over represented. To remedy this, and to produce embeddings that represent the product over multiple angles and scales, product images where augmented until we had acquired 40 images per product. This was done by rotating, zooming in and changing the width and height of the images. The data augmentation allowed us to collect 68.640 images in total. Figure 3 shows the image augmentation process.

Figure 3: From left to right; the original image as seen on Pandoras website, the cropped and resized image and an augmented image

[8]From the following country sites: (CN, HK, JP, NZ, SQ, DK, DE, FR, IT, UK, NL, PL, SE, AT, AU)

The data set is furthermore unbalanced by the amount of products in each of the larger, semantically similar categories like ('Bracelets', 'Charms', 'Earrings', Miscellaneous', 'Necklaces & Pendants', 'Rings') as can be seen in Figure 4.

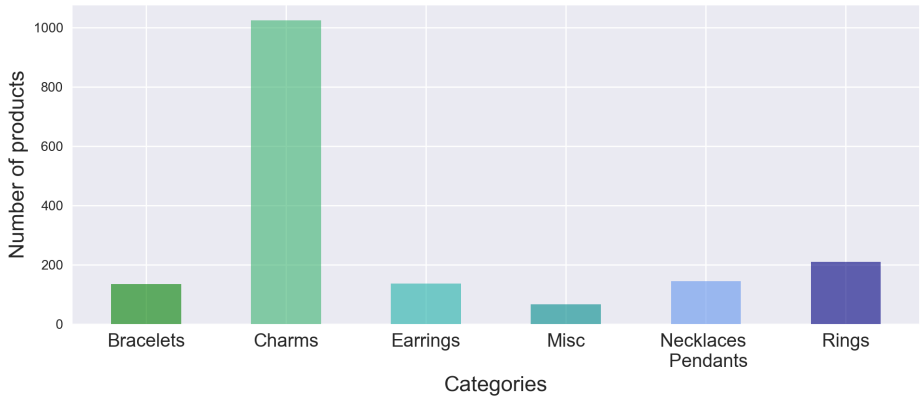


Figure 4: The number of products per category

In Table 1 the exact number of each product based on type and colours are displayed. The imbalance might make the model favour weights and features related to differentiating the larger categories such as 'charms'.

Type	Colour	No. of products	% of Total
Charms	Gold	67	3,89%
(1025 products)	Rose	113	6,57%
	Silver	763	44,33%
	Oxidised Silver	4	0,23%
	Two Tone	78	4,53%
Rings	Gold	15	0,87%
(210 products)	Rose	59	3,43%
	Silver	134	7,79%
	Oxidised Silver	0	0,00%
	Two Tone	2	0,12%
Necklaces	Gold	23	1,34%
(145 products)	Rose	30	1,74%
	Silver	86	5,00%
	Oxidised Silver	0	0,00%
	Two Tone	6	0,35%
Earrings	Gold	9	0,52%
(136 products)	Rose	30	1,74%
	Silver	95	5,52%
	Oxidised Silver	0	0,00%
	Two Tone	2	0,12%
Bracelets	Gold	11	0,64%
(133 products)	Rose	26	1,51%
	Silver	77	4,47%
	Oxidised Silver	2	0,12%
	Two Tone	17	0,99%

Table 1: Number of products based on the type and colour. Miscellaneous, brooches and bundles are excluded from this table but is counted towards the total number of products and therefore represented in the percentages.

Figure 5 shows a principal component visualisation of the data set. The classes correspond to the main categories on the Pandora website. The figure shows that the images in this basic interpretation are inseparable; there is no clear way to draw a boundary between the classes or define any axes of features based on these main categories alone using only the two first principal components.

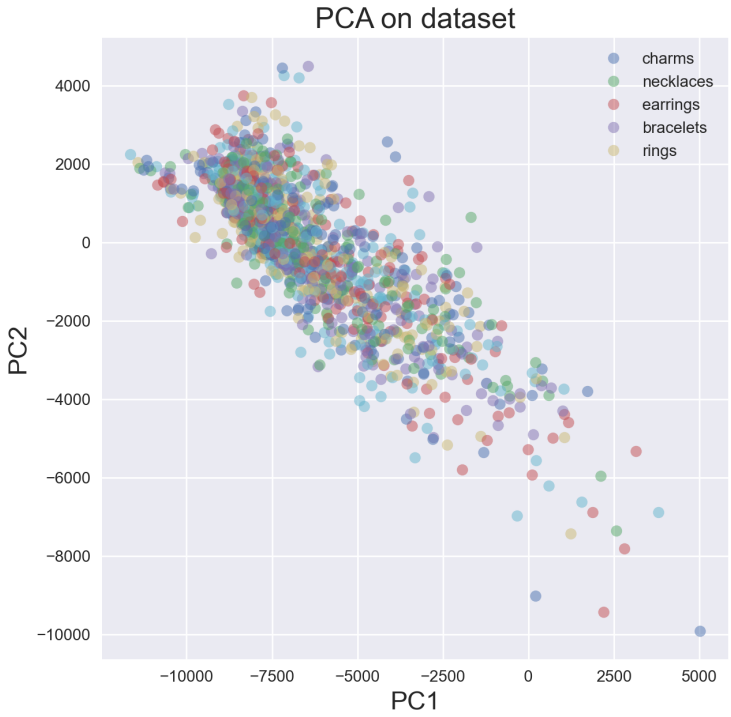


Figure 5: PCA projection off all images, no augmentation

3.1 Ethical considerations

This project does not contain many of the risks that other machine learning projects are typically prone to. We do not provide recommendations that are in any way critical to the safety and wellbeing of our users, and we do not handle sensitive personal data. However, we still have some considerations to ensure that our model is implemented ethically, such that we can build trust with the user.

Given that the model will be operating in EU countries, it will have to comply with the General Data Protection Regulation (GDPR). This requires that query images containing people are not stored in the system, or that consent is obtained during submission of the image. Since the system will be generating recommendations based on similarity scores between the feature vectors of images, this issue can be solved by calculating the feature vector of the query image immediately and never storing the image directly.

Regarding the diversity, unfair biases in our dataset should be avoided, such that unintended prejudice from our model will not become a problem [Hig19, p. 18]. Our machine learning model should be able to perform equally well on the query images of people of different race and genders. To ensure our model upholds this, the training data must not be biased towards race and gender, and this becomes especially relevant when the model is implemented and trained on 'wild' data.

For the purposes of this report, the 'wild' data is only used for testing the generalisation error of

our model. The data is augmented using image segmentation that crops out the jewellery from the image. It could potentially be the case that the image segmentation model crops the jewel differently if it is held in the hand by two different people, depending on the data it has been trained on. Perhaps it is better better at extracting the jewel if there is a darker background because a silver jewel for instance has a higher contrast towards a darker background, which makes it is easier for the system to crop out. A way to remedy this is to ensure that the image segmentation model is trained on a data-set that is as diverse as possible. No analysis of bias or discrimination in the basis-model that we used seem to exist.

4 Methods

In this section we will describe the two CBIR-methods that are being compared, namely the VAE and the TRN, along with the image segmentation model that is part of the system pipeline seen in Figure 6.

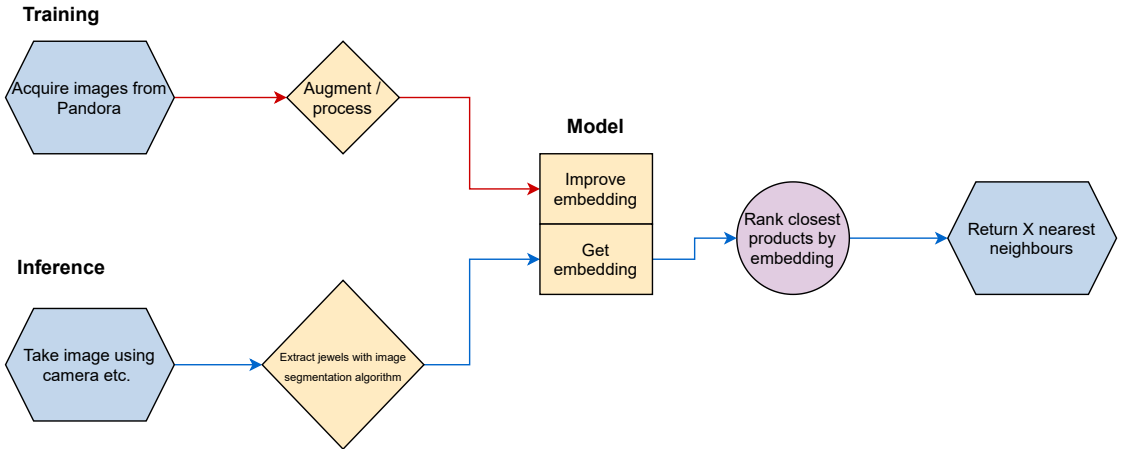


Figure 6: Overall system pipeline.

4.1 Variational Auto Encoders

4.1.1 Introduction

When it comes to measuring similarity between data points, it is often desirable to reduce the number of dimensions that the data is represented in, such that the only dimensions that are considered actually contribute valuable information about the data. This is especially evident when dealing with images, where even a standard Full-HD RGB image is represented in $1920 \times 1080 \times 3 = 6.220.800$ dimensions. This is obviously too many, as changing for example the red value of a single pixel has virtually no effect on the content of the image. The problem only gets worse as we consider image similarity within a single domain (e.g. jewellery), where a lot of attributes are implicitly given by the context, and even less dimensions should be sufficient to represent each object.

4.1.2 Autoencoders

One method of dimensionality reduction is autoencoders, which utilise an encoder-decoder structure to attempt to learn an optimal mapping from the input space to a lower dimensional latent space, such that the decoder can recreate the original input as closely as possible. When the dimensionality is reduced to a number that is low enough to be practically useful, the encoding-decoding process will almost always be lossy, meaning that the original image can not be perfectly reconstructed from the latent representation. This is not a problem however, as it is mostly the latent space and its properties that are of interest to this project.

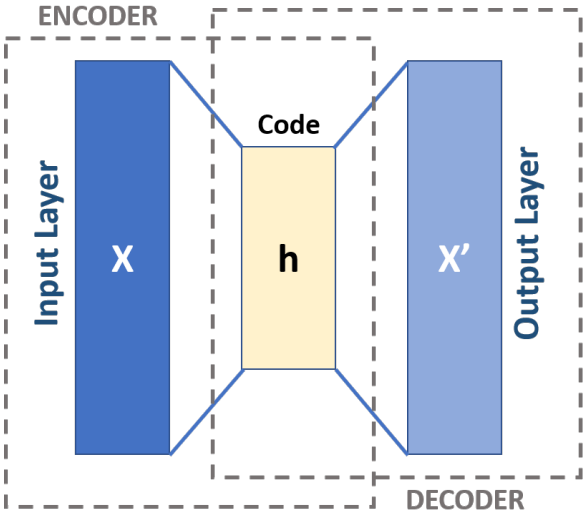


Figure 7: Architecture of an autoencoder. Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_schema.png

In Figure 7 the overall structure of the autoencoder is shown. An encoder $e : \mathbb{R}^M \to \mathbb{R}^m$ maps the input $\mathbf{x}$ to an encoding $\mathbf{h}$ in the latent space, such that $e(\mathbf{x}) = \mathbf{h}$. The decoder $d : \mathbb{R}^m \to \mathbb{R}^M$ then takes the encoding $\mathbf{h}$ and attempts to reconstruct $\mathbf{x}$. This reconstruction will be denoted $\mathbf{x}'$, such that $d(\mathbf{h}) = \mathbf{x}'$. The problem of finding the optimal encoder-decoder pair can simply be written as:

$$\begin{aligned}(e^*, d^*) &= \underset{(e,d)\in E\times D}{\arg\min}\ \mathcal{L}\left(\mathbf{x}, d(e(\mathbf{x}))\right) \\ &= \underset{(e,d)\in E\times D}{\arg\min}\ \mathcal{L}\left(\mathbf{x}, \mathbf{x}'\right)\end{aligned} \tag{1}$$

Where E and D are the sets of all admissible encoders and decoders of the architectures that are considered, and $\mathcal{L}$ denotes the loss function comparing the inputs $\mathbf{x}$ to their reconstructed counterparts $d(e(\mathbf{x}))$. A common loss function for comparing images is the cross-entropy:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = -\sum_i \sum_j \mathbf{x}_{i,j} \log \mathbf{x}'_{i,j} \tag{2}$$

Cross-entropy usually compares two discrete probability functions with the same support, and since the images have the same dimensions and are represented with numbers in the range $[0, 1]$, the abstraction from images to probability distributions is not that far-fetched.

An autoencoder of this type will often find a mapping that is suitable to reproduce the training images, but since the only performance measure is the similarity between specific images and their reconstructions, there is no guarantee that the latent space is in any way smooth or continuous. This means that points that are not exactly equal to encodings of the training set will often lead to nonsensical results when decoded. Recommendation systems are very interested in the latent properties of new unseen images, which means that a latent space with

these non-smooth properties is not desired.

4.1.3 Adding variation

To regularise the latent space and encourage properties like smoothness and continuity, the inputs are encoded as probability distributions over the latent space instead of individual points. Autoencoders using this technique are called Variational Autoencoders.

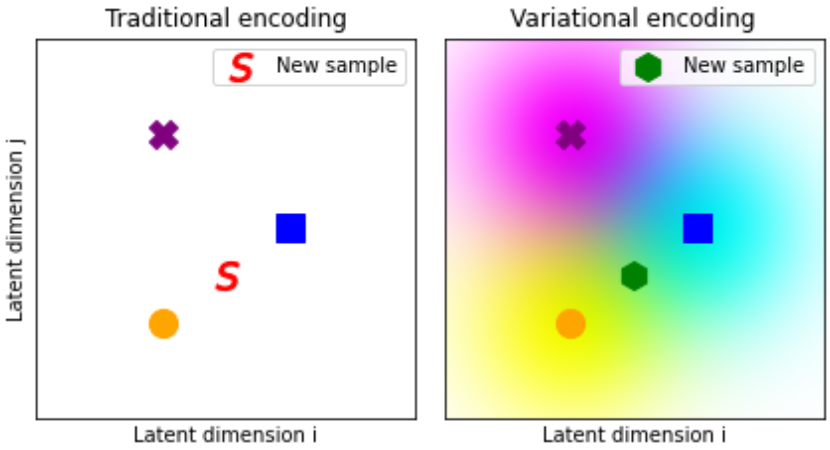


Figure 8: Visualization of the latent space. In the variational setting (right), points sampled far from the training points will still have sensible properties.

A common simplification is to limit these distributions to multivariate normals with diagonal covariance matrices. This places certain assumptions on the distribution of the data, but the ease of implementation makes it the solution of choice for this report. During training the model samples points from these distributions, and it is then these sampled points that are reconstructed and compared to the input:

$$\begin{aligned}&\text{Before:}\\&\mathbf{x} \to e(\mathbf{x}) \to d(e(\mathbf{x})) = \mathbf{x}'\\&\text{After:}\\&\mathbf{x} \to \mathbf{z} \sim p(\mathbf{z}|\mathbf{x}) \to d(\mathbf{z}) = \mathbf{x}'\end{aligned}$$

This forces the latent space around the input encodings to represent data that is similar to that of the input (see fig. 8). An issue with this approach is that the model might learn distributions with a high distance between their means compared to their variances, which also facilitates an irregular latent space in regions where none of the distributions have much probability mass. To combat this, a Kullback-Leibler (KL) term is added to the loss function:

$$\begin{aligned}L(\mathbf{x}, \mathbf{x}', \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= -\sum_i \sum_j \mathbf{x}_{i,j} \log \mathbf{x}'_{i,j} + D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I}))\\&= -H\left(\mathbf{x}, \mathbf{x}'\right) + D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,||\, \mathcal{N}(\mathbf{0}, \mathbf{I}))\end{aligned} \tag{3}$$

This KL term approaches zero when the two distributions given as arguments approach each other, and thereby forces the encoded distribution to be close to a given distribution, in this case the standard multivariate normal.

Calculating KL for two arbitrary distributions usually requires integrating over the entire domain, but since both distributions are Gaussian with diagonal covariance matrices, substantial simplifications can be made. Recall the general KL expression, and the probability density of the multivariate Gaussian:

$$D_{\mathrm{KL}}(p||q) = \int_x p(x) \log\left(\frac{p(x)}{q(x)}\right) \tag{4}$$

$$p\left(\mathbf{x}\right) = \frac{1}{\left(2\pi\right)^{k/2}\left|\boldsymbol{\Sigma}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{5}$$

Where k is the dimension of the Gaussian. The two distributions are in this case $p \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $q \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\boldsymbol{\Sigma}$ is diagonal. The first step is to realise that the KL expression (eq. 4) is an expectation over a function. Doing so, and expanding the fraction in the logarithm, yields the following:

$$D_{\mathrm{KL}}(p||q) = \mathbb{E}_p\left[\log p(\mathbf{x}) - \log q(\mathbf{x})\right] \tag{6}$$

Simplifying the logarithms in advance reveals several cancelling terms:

$$\begin{aligned}
\log p(\mathbf{x}) &= \log\left(\frac{1}{\left(2\pi\right)^{k/2}\left|\boldsymbol{\Sigma}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\right) \\
&= -\log\left(\left(2\pi\right)^{k/2}\right) - \frac{1}{2}\log\left(\left|\boldsymbol{\Sigma}\right|\right) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \\
\log q(\mathbf{x}) &= \log\left(\frac{1}{\left(2\pi\right)^{k/2}\left|\mathbf{I}\right|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{0})^T\mathbf{I}^{-1}(\mathbf{x}-\mathbf{0})\right)\right) \\
&= \log\left(\frac{1}{\left(2\pi\right)^{k/2}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right)\right) \\
&= -\log\left(\left(2\pi\right)^{k/2}\right) - \frac{1}{2}\mathbf{x}^T\mathbf{x}
\end{aligned}$$

Inserting in eq. 6:

$$
\begin{aligned}
D_{\mathrm{KL}}(p||q) &= \mathbb{E}_p\left[-\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) + \frac{1}{2}\mathbf{x}^T\mathbf{x}\right] \\
&= \frac{1}{2}\mathbb{E}_p\left[\mathbf{x}^T\mathbf{x}\right] + -\frac{1}{2}\mathbb{E}_p\left[\log|\boldsymbol{\Sigma}|\right] - \frac{1}{2}\mathbb{E}_p\left[(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \\
&\text{Simplifying the first term via [Bra+12, eq. 380, section 8.2]} \\
&= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\left\{\boldsymbol{\Sigma}\right\} - \log|\boldsymbol{\Sigma}| - \mathbb{E}_p\left[(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]\right) \\
&\text{The inside of the last expectation is scalar, and therefore equal to its own trace:} \\
&= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\left\{\boldsymbol{\Sigma}\right\} - \log|\boldsymbol{\Sigma}| - \mathbb{E}_p\left[tr\left\{(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}\right]\right) \\
&\text{Using linearity of expectation and } tr\{\mathbf{ABC}\} = tr\{\mathbf{BCA}\}: \\
&= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\left\{\boldsymbol{\Sigma}\right\} - \log|\boldsymbol{\Sigma}| - tr\left\{\mathbb{E}_p\left[(\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu})\right]\boldsymbol{\Sigma}^{-1}\right\}\right) \\
&= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\left\{\boldsymbol{\Sigma}\right\} - \log|\boldsymbol{\Sigma}| - tr\left\{\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}\right\}\right) \\
&= \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + tr\left\{\boldsymbol{\Sigma}\right\} - \log|\boldsymbol{\Sigma}| - k\right) \qquad (7)
\end{aligned}
$$

Given that $\boldsymbol{\Sigma}$ is diagonal, its determinant is simply the product of its diagonal elements, and as a consequence the logarithm of the determinant is the sum of the logarithms of the diagonal elements, simplifying the calculation of eq. 7 even further. A diagonal matrix can be more effectively stored as a vector, leading to the naming convention that $diag(\boldsymbol{\Sigma}) = \boldsymbol{\sigma}$. With this in place, the final expression becomes:

$$
D_{\mathrm{KL}}(\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \,||\, \mathcal{N}(0,\mathbf{I})) = \frac{1}{2}\left(\boldsymbol{\mu}^T\boldsymbol{\mu} + \sum_{\sigma\in\boldsymbol{\sigma}}(\sigma - \log\sigma) - k\right) \qquad (8)
$$

4.1.4 The reparameterization trick

When naively introducing random sampling to the training step, the model can no longer use backpropagation to perform gradient descent. This is remedied by re-parameterizing the sampling layer, so that instead of sampling directly from the distribution given by the mean $\boldsymbol{\mu}$ and the standard deviation $\boldsymbol{\sigma}$, it computes the sample as $\boldsymbol{\mu} + \boldsymbol{\sigma}\cdot\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a noise term of the form $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$:

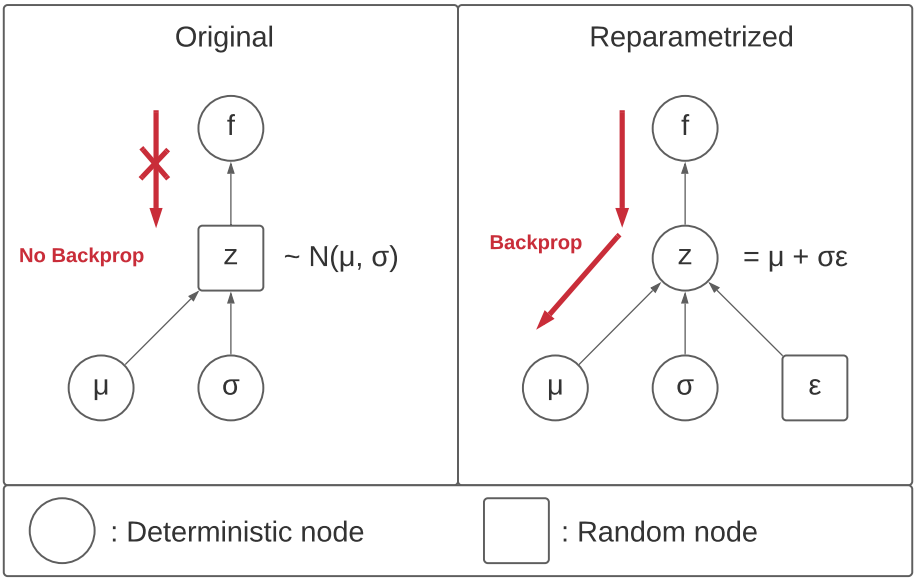


Figure 9: Reparameterization of the sampling layer. Graphics inspired by [KW-19].

When using the structure on Figure 9 (right), gradients can be computed for both $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, and backpropagation is therefore possible once again.

4.1.5 VAE implementation

The VAE is implemented using the neural network library Keras, which allows the creation of layers, models and submodels. This approach allows the encoder and decoder to be defined separately, but still be trained as one coherent model. To implement a model in Keras, three main components must be specified: the architecture, the loss function, and the data set.

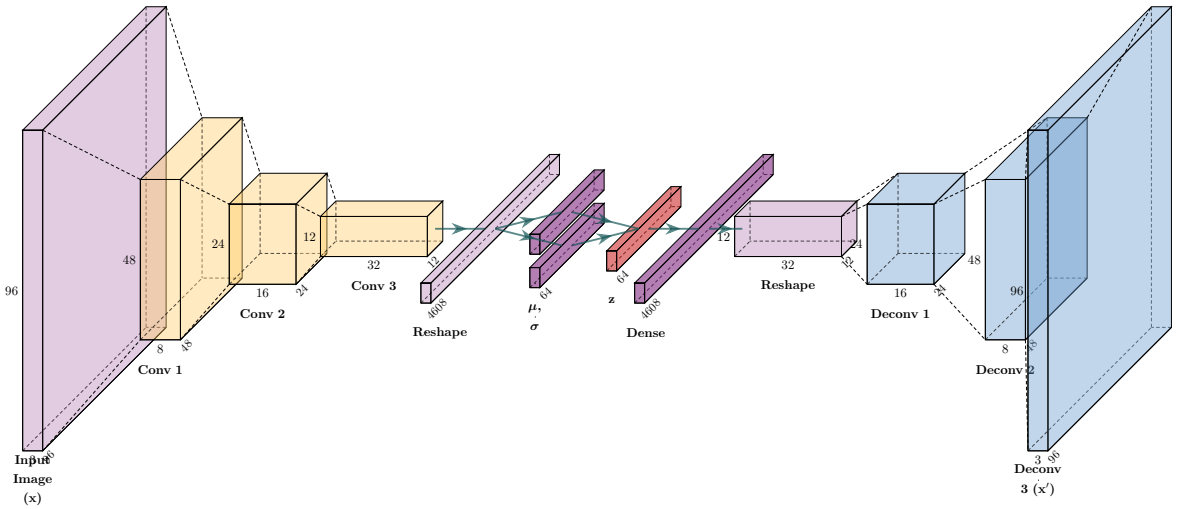


Figure 10: Architecture of the autoencoder. A larger version can be found in Appendix D

The overall architecture of the autoencoder consists of an encoder and a decoder as stated in section 4.1.2. In figure 10, the latent space representation is denoted by the red block labeled

$\mathbf{z}$, everything to the left of it is part of the encoder, and everything to the right is part of the decoder. The encoder takes an input image and feeds it through three convolutional layers, each with a filter stride of 2, effectively halving the height and width for each layer, while adding an increasing number of filter channels. It then flattens the final convolution and feeds it to two dense layers in parallel, encoding the mean ($\boldsymbol{\mu}$) and the variance ($\boldsymbol{\sigma}$) respectively. These two layers are then used to sample points from the encoded distribution in the final layer. A detailed description of each layer can be found in table 2.

Layer	Shape	Activation	Stride	Kernel Size
Input Image	$(96, 96, 3)$	n/a	n/a	n/a
Conv 1	$(48, 48, 8)$	ReLu	2	3
Conv 2	$(24, 24, 16)$	ReLu	2	3
Conv 3	$(12, 12, 32)$	ReLu	2	3
Reshape	$(4608,)$	n/a	n/a	n/a
Dense $\boldsymbol{\mu}$	$(64,)$	Linear	n/a	n/a
Dense $\boldsymbol{\sigma}$	$(64,)$	Linear	n/a	n/a
Latent Space ($\mathbf{z}$)	$(64,)$	$\boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$	n/a	n/a

Table 2: Layer by layer architecture of the encoder. All layers take the layer before it as input, except $\boldsymbol{\sigma}$ which takes Reshape as input, and $\mathbf{z}$ which is a function of the two previous layers.

The decoder takes a point sampled from the encoded distribution of an image, and feeds it into a fully connected dense layer. This dense layer is then reshaped, and used as input to three successive deconvolution layers. These deconvolutions operate with stride 2 like the convolutions in the encoder, and because of this symmetry the final deconvolution has the same shape as the inputs to the encoder, which is desired. A detailed description of each layer can be found in table 3.

Layer	Shape	Activation	Stride	Kernel Size
Input Encoding	$(64,)$	n/a	n/a	n/a
Dense	$(4608,)$	ReLu	n/a	n/a
Reshape	$(12, 12, 32)$	n/a	n/a	n/a
Deconv 1	$(24, 24, 16)$	ReLu	2	3
Deconv 2	$(48, 48, 8)$	ReLu	2	3
Deconv 3	$(96, 96, 3)$	Sigmoid	2	3

Table 3: Layer by layer architecture of the decoder. All layers take the layer before it as input.

In order to feed the large dataset into the VAE efficiently, a custom data generator was implemented. Instead of loading the entire dataset of size N at once into a NumPy array of size $(N, 96, 96, 3)$, the generator creates shuffled batches of NumPy arrays of size 512, which are fed into the model sequentially. This heavily reduces the memory needed to train the VAE, as only one batch needs to be loaded into RAM at once, as opposed to the entire dataset.

4.2 Triplet Ranking Network

Search-by-example is the process of finding images that are similar to a query image and in category-level image similarity two images are considered similar if they belong to the same category. The authors of [Wan+14] describe that in order to have a powerful image search engine, category-level image similarity is not sufficient for a search-by-example search application because the engine needs to distinguish between images that are in the same category - this defines fine-grained image similarity. For instance, two images are in the same *ring* category but the colour and design of the two rings differ greatly, see figure 11.

In order to incorporate fine-grained image information into the embeddings of the jewellery images, Wang et al. [Wan+14] propose the use of deep ranking in their model architectures. Deep ranking builds on the idea of giving the model access to information on the image similarity relationship between different images in order to incorporate these fine-grained similarities into the embeddings. Unlike classification models where the objective is to predict a label given an image for instance, the objective in ranking models and ranking losses is to predict the distance between two images and determine how similar they are.

There are several different deep ranking model architectures and losses as described in section 2.2, where one of the most commonly used is the Triplet Ranking Network.

4.2.1 Triplet loss

To give the TRN information on similarity relationships between data points, sets of three images (triplet pairings) are created and utilised as an input. A set, $\mathcal{P}$, consists of an anchor image, a positive image that is similar to the anchor, and a negative image, which is dissimilar. Figure 11 shows an example of an anchor image along with a positive image as well as a negative image.

This method of creating the triplet pairings puts the restraint on your data set that it must contain several images of the same object. This constraint is satisfied in our dataset.

In this section the following notations from [Sik+20] are used to describe the triplets: In the training dataset X, x^i denotes an instance of the i'th product. Using this, a single triplet pairing consisting of an anchor, positive and negative image is denoted by (x_a^i, x_p^i, x_n^j), where the anchor and positive image are of the same product i and the negative image is of product j.

Figure 11: From left to right: The anchor image x_a^i, the positive image x_p^i, and the negative image x_n^j.

The goal of incorporating triplet pairs as input to the embedding model is to teach the model that the positive images are semantically more similar to the anchor image than the negative images. This should translate into embeddings of anchor images being closer to embeddings of their positive images than the negative ones in the latent space. Doing this will decrease the intra-class and increase the inter-class variances for the different embeddings, since it pulls the anchor closer to the positive and pushes the negative embedding away in the latent space [Sik+20]. This goal is illustrated in Figure 12 from [SKP15].

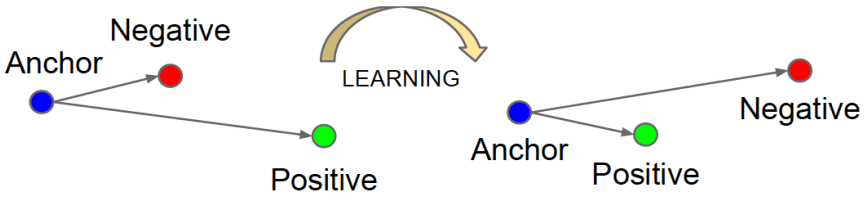


Figure 12: The triplet loss aims to minimise the distance in a anchor-positive pair, (x_a^i, x_p^i) and maximise the distance in a anchor-negative pair, (x_a^i, x_n^j).

This idea yields the following for the triplet loss:

$$\left\|f(x_a^i) - f(x_p^i)\right\|^2 < \left\|f(x_a^i) - f(x_n^j)\right\|^2 \tag{9}$$

Where $f(\cdot)$ indicates the deep embedding function. The terms on both sides indicates the distance from the embeddings of the positive and negative images to the anchor, measured using the squared euclidean distance. For simplicity, we will denote the deep embeddings, $y = f(x)$, of the anchor, positive and negative image by y_a^i, y_p^i, y_n^j respectively, so the equation above becomes:

$$\left\|y_a^i - y_p^i\right\|^2 < \left\|y_a^i - y_n^j\right\|^2 \tag{10}$$

Additionally, we can simplify the terms on both sides of eq. 9 with a distance function $\mathcal{D}(y^i, y^j)$, that outputs the squared euclidean distance between the embeddings of two images i, j. We rewrite from eq. 9:

$$\mathcal{D}(y_a^i, y_p^i) < \mathcal{D}(y_a^i, y_n^j) \tag{11}$$

In order to ensure that an anchor image is closer to its positive anchor than its negative anchor by more than an infinitesimal number, a margin is added to force some distance between y_p^i and y_n^j in the latent space:

$$\mathcal{D}(y_a^i, y_p^i) + \alpha < \mathcal{D}(y_a^i, y_n^j) \tag{12}$$

Therefrom, the loss function, which is also described in [HBL17], is:

$$\mathcal{L}_{\text{tri}}(y_a^i,\ y_p^i,\ y_n^j) = \sum_{\substack{a,p,n \\ i \neq j}} \left[\alpha + \mathcal{D}(y_a^i, y_p^i) - \mathcal{D}(y_a^i, y_n^j)\right]_+ \tag{13}$$

The triplet loss function makes sure that given the anchor image x_a^i, the embedding of the

positive image x_p^i will be closer to the anchors projections than that of the negative image x_n^j by at least a margin α. The notation '$[\cdot]_+$' denotes the standard Hinge loss $\max(\cdot, 0)$ which makes sure that only positive loss values is taken into account and the loss values will be 0 if the embedding satisfies the constraint in eq. 12. The Hinge loss ensures that the model is not rewarded for separating the images in the embedding space more than α. This ensures that the embedding space does not "explode".

4.2.2 Offline and online triplets

[Sik+20] distinguishes between an offline and online approach of triplet mining. When triplets are selected offline, a triplet dataset is created before the training session and all training data is taken into account. They are chosen randomly based on the labels, meaning that it is not considered whether the triplet pairing is objectively good. The network is then trained using these triplets. In an online triplet mining approach, the triplets are created and altered within each mini-batch of the training data and the embeddings are used to create what we will refer to as *easy*, *hard* and *semi-hard* triplets, described in the next section.

In offline triplet mining a few problems arise. The main issue lies with incorporating all possible pairs of triplet in the training set. This will be infeasible for a large, or even a medium sized, data set, since the number of triplet pairs, $N_{\mathcal{P}}$, grows with $O(N^3)$ as the number of data points, N, increases. This means that our fairly humble sized data set of 61.776 images yields approximately $61776^3 = 2.3575 \cdot 10^{14}$ number of pairings.

Additionally, selecting the triplets at random might become problematic. This is because choosing triplets $\mathcal{P}_i = (x_a^i, x_p^i, x_n^j)$ at random would lead to the constraint in eq. 12 being too easily satisfied, since there is a good chance that the negative anchor embedding y_n^j is naturally a lot different from the objective image than the positive anchor image is (Figure 11 is a good example of a too easily satisfied triplet pairing). If this is the case, the triplets will prevent the network from learning meaningful embeddings, since there is not significantly new information.

The authors of [SKP15] implement online triplet mining strategies as supposed to the offline approach and in [HBL17] their online approach outperforms their offline approach by a large margin. These previous and well-studied experiments along with the problems presented for the offline approach motivate our use of the online approach for selecting triplets.

4.2.3 Easy, hard and semi-hard triplet mining

We want to prevent generating all possible triplets that are easily satisfied by eq. 12, which we define to be *easy* triplets as the loss will become 0 and the parameters in the model will not be updated. [SKP15] describes that in order to avoid producing these easy triplets, it is crucial to select *hard* triplets which will contribute to improving the model. These hard triplets are the ones who violate the constraint given in eq. 12. We define *hard positives* as the positive images

in the anchor-positive pair (x_a^i, x_p^i) which satisfies:

$$\arg\max_{x_p^i} \mathcal{D}(f(x_a^i), f(x_p^i)) \tag{14}$$

i.e. we select the positive image to be the one with maximum distance to our anchor image in the embedding space. Similarly, the *hard negatives* are defined to be the negative images in the anchor-negative pair (x_a^i, x_n^j) which satisfies:

$$\arg\min_{x_n^j} \mathcal{D}(f(x_a^i), f(x_n^j)) \tag{15}$$

i.e. we select the negative image to be the one with minimum distance to our anchor image in the embedding space. In conclusion, the hard triplets are defined as those, where the negative image is closer to the anchor image than the positive image. An interpretation of this is explained well in [HBL17] - in terms of our jewellery setting, the model will not learn much by being told that jewellery pieces with different colours/design are indeed different jewellery pieces. However, if the model sees images of jewellery pieces that look very similar but are different (*hard negatives*) and images of jewellery pieces that are the same but the image augmentation causes the images of the same product to look very different (*hard positives*), the model will hopefully be better at understanding the concept of the same jewellery piece.

However, mining only hard triplets might become problematic for the model. If the model only is shown the hard triplets, the embedding function, $f(\cdot)$, will be unable to learn so-called 'normal' associations to the data. To combat this, the *semi-hard triplet* is introduced. For semi-hard triplets we select the positive and negative such that:

$$\mathcal{D}(f(x_a^i), f(x_p^i)) < \mathcal{D}(f(x_a^i), f(x_n^j)) < \mathcal{D}(f(x_a^i), f(x_p^i)) + \alpha \tag{16}$$

This implies that the negative images are farther away than the positive images but are still to be considered hard because the squared euclidean distance is close to the positive image and the loss will not become 0. Figure 13 highlights the type of negative images described.

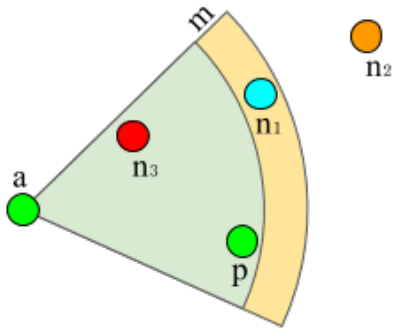


Figure 13: Types of negatives from [Tah20]. The hardest negative n_3 marked with red is closer to the anchor than the positive image, whereas the semi-hard negative n_1 marked with cyan is farther away than the positive but lies in the margin m. The easiest negative, n_2, is the one farthest away from the anchor.

4.2.4 Batch sampling strategies in online triplet mining

Once the type of triplet mining is selected, one can consider how to collect the triplets in mini-batches, which will be fed into the model. In the following, notations regarding the batch sampling from [Sik+20] are applied: we let b denote the size of the mini-batch and c denote the number of jewellery products in that batch. Additionally we let w defined by $w = \lfloor b/c \rfloor$ denote the sample triplet size per product in the mini-batch.

Regarding how to collect the triplets in the mini-batches, we look further into the so-called *Batch All*, *Batch Hard*, *Batch Semi-Hard* and *Random Hard* sampling strategies.

Batch All: The first strategy described in both [HBL17] and [Sik+20] simply uses *all* possible valid combinations of the triplets in the mini-batches, namely denoted the *Batch All* sampling strategy. Here we consider all possible positive and negative images in the mini-batch. The *Batch All* loss function is the regular triplet loss from eq. 13 summed over all triplets in the mini-batch:

$$\mathcal{L}_{\text{BA}} = \overbrace{\sum_{i=1}^{c}\sum_{a=1}^{w}}^{\text{all anchors}} \overbrace{\sum_{\substack{p=1 \\ p\neq a}}^{w}}^{\text{all pos.}} \overbrace{\sum_{j=1}^{c}\sum_{\substack{n=1 \\ j\neq i}}^{w}}^{\text{all neg.}} \left[\alpha + \mathcal{D}\left(y_a^i, y_p^i\right) - \mathcal{D}\left(y_a^i, y_n^j\right)\right]_+ \tag{17}$$

Batch Hard: Both papers also propose the use of the *Batch Hard*, where the approach for forming the triplet is that for each anchor image, we select both the *hardest positive* and *hardest negative* samples within the batch and these triplets will be used to compute our batch hard loss, which uses the form of the regular triplet loss seen in eq. 13:

$$\mathcal{L}_{\text{BH}} = \overbrace{\sum_{i=1}^{c}\sum_{a=1}^{w}}^{\text{all anchors}} \left[\alpha + \overbrace{\max_{\substack{p=1,\ldots,w \\ p\neq a}} \mathcal{D}\left(y_a^i, y_p^i\right)}^{\text{hardest positive}} - \overbrace{\min_{\substack{j=1,\ldots,c \\ n=1,\ldots,w \\ j\neq i}} \mathcal{D}\left(y_a^i, y_n^j\right)}^{\text{hardest negative}}\right]_+ \tag{18}$$

when applying the same notations as before.

Batch Semi-Hard: The middle ground between the *batch all* and *batch hard* is the *Batch Semi-Hard* strategy, which also is proposed in [Sik+20]:

$$\mathcal{L}_{\text{BSH}} = \overbrace{\sum_{i=1}^{c}\sum_{a=1}^{w}}^{\text{all anchors}} \overbrace{\sum_{\substack{p=1 \\ p\neq a}}^{w}}^{\text{all pos.}} \left[\alpha + \mathcal{D}\left(y_a^i, y_p^i\right) - \min_{\substack{j\in\{1,\ldots,c\}\setminus\{i\} \\ n\in\{1,\ldots,w\}}} \left\{\mathcal{D}\left(y_a^i, y_n^j\right) \mid \mathcal{D}\left(y_a^i, y_n^j\right) > \mathcal{D}\left(y_a^i, y_p^i\right)\right\}\right]_+ \tag{19}$$

The *Batch Semi-Hard* loss function will select triplets in such a manner that the distance between the anchor and negative image is minimised, but the computed distance must be greater than that between the anchor and positive image.

Random Hard: Lastly, the *Random Hard* batch strategy can be used when comparing the

different strategies. As the name suggests, instead of choosing the hardest negatives that will promote the highest loss values, the random hard strategy will choose randomly among the triplets that have a loss value above 0.

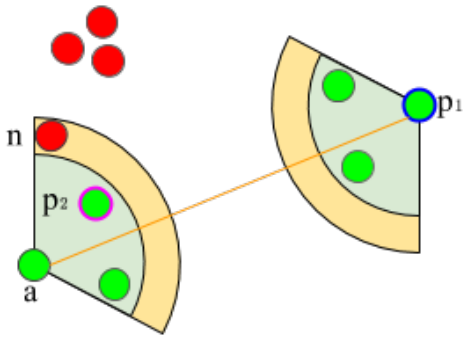


Figure 14: Hard and semi-hard batching illustration from [Tah20].

The *Batch Hard* sampling strategy will always pick the positive image that is the furthest away from the anchor and the negative which is the closest. This means that in the figure above Fig. 14, the sampled triplet will be (a, p_1, n). *Batch Semi-Hard* will however choose a positive image, here p_2, and ensure that the negative image selected is farther away than the positive but the distance to the anchor is still minimised as seen in eq. 19 and the sampled triplet will be (a, p_2, n)

We will focus on the *Batch Hard*, *Batch Semi-Hard* and *Random Hard* and compare the three sampling strategies. *Batch All* is not included as selecting all possible triplets will perhaps not contribute to training and cause slow convergence as the authors of [SKP15] argue. [HBL17] found that *Batch Hard* outperformed *Batch All* in the domain of Person Re-Identification.

Generally for all batch strategies, because the triplets are collected in mini-batches, they might not reflect the data neighbourhood entirely and they can result in in a sub-optimal solution, which [Sik+20] suggests. The online mining approach is still preferred as it outperforms the offline method as previously stated.

4.2.5 Margin values

The margin value ensures that there is some distance α in the embedding space between the positive and negative images as we see in eq. 12. [Che+17] argue the importance choosing the margin value in online triplet mining - a small margin would result in a few hard samples and feeding these into the model would cause slow convergence and and lead the model to a suboptimal solution. However, choosing a large margin would result in too many hard samples and cause over-fitting. [Che+17] propose to use an adaptive margin that aims to avoid the over- and under-sampling problem. The authors of [HBL17] use a set of four different margin values (0.1, 0.2, 0.5, 1.0) when comparing batch sampling strategies, which is the approach we will be experimenting with when choosing the optimal batching strategy.

4.2.6 Deep Triplet Ranking CNN Architecture

Each input to the deep ranking triplet model is a triplet of three RGB images (x_a, x_p, x_n) with dimensions (96, 96, 3) and the output is an embedding per input image of dimension 10.

We implemented three relatively simple identical CNNs with shared weights and hyperparameters. Each triplet image passes through one of the identical parallel sub-networks. The sub-networks share the following network architecture:

Layer	Shape	Activation	Stride	Kernel Size
Input Image	$(96, 96, 3)$	n/a	n/a	n/a
Conv 1	$(92, 92, 32)$	PReLU	n/a	5
Max-Pool 1	$(23, 23, 32)$	n/a	4	2
Conv 2	$(19, 19, 64)$	PReLU	n/a	5
Max-Pool 2	$(9, 9, 64)$	n/a	2	2
Linear	(256)	PReLU	n/a	n/a
Linear	(256)	PReLU	n/a	n/a
Linear	(10)	PReLU	n/a	n/a

Table 4: Layer by layer architecture of the decoder. All layers take the layer before it as input. PReLU is the parametric rectified linear unit.

The architecture for a single sub-network is illustrated below in Figure 17, where the convolutions are represented by yellow blocks, the pooling operations by red blocks and the fully connected layers are shown in purple.

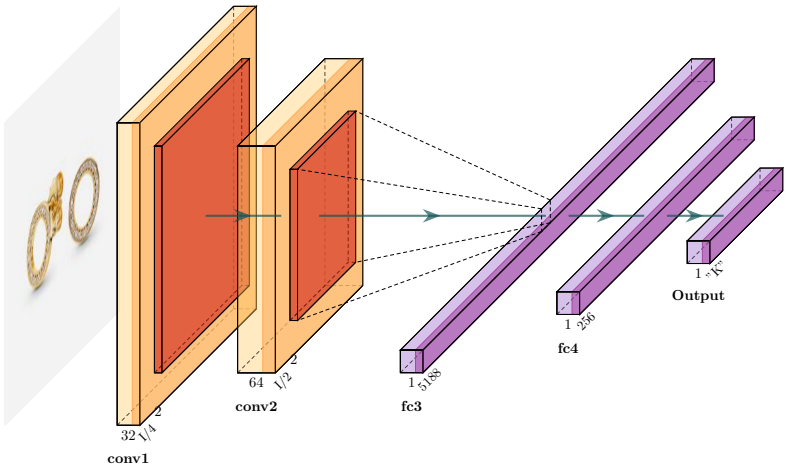


Figure 17: A subnetwork fo the Triplet Ranking Network. A larger version can be found in the appendix in section D.

Through grid-search and comparison of different models, the final hyper parameters are:

- Epochs: 15

- Learning Rate: 0.001
- Weight Decay: 1e-4
- Gamma: 0.1
- Margin: 0.1
- Batch sampling strategy: Semi-hard

The entire deep ranking model structure using the triplet loss is illustrated in Figure 18.

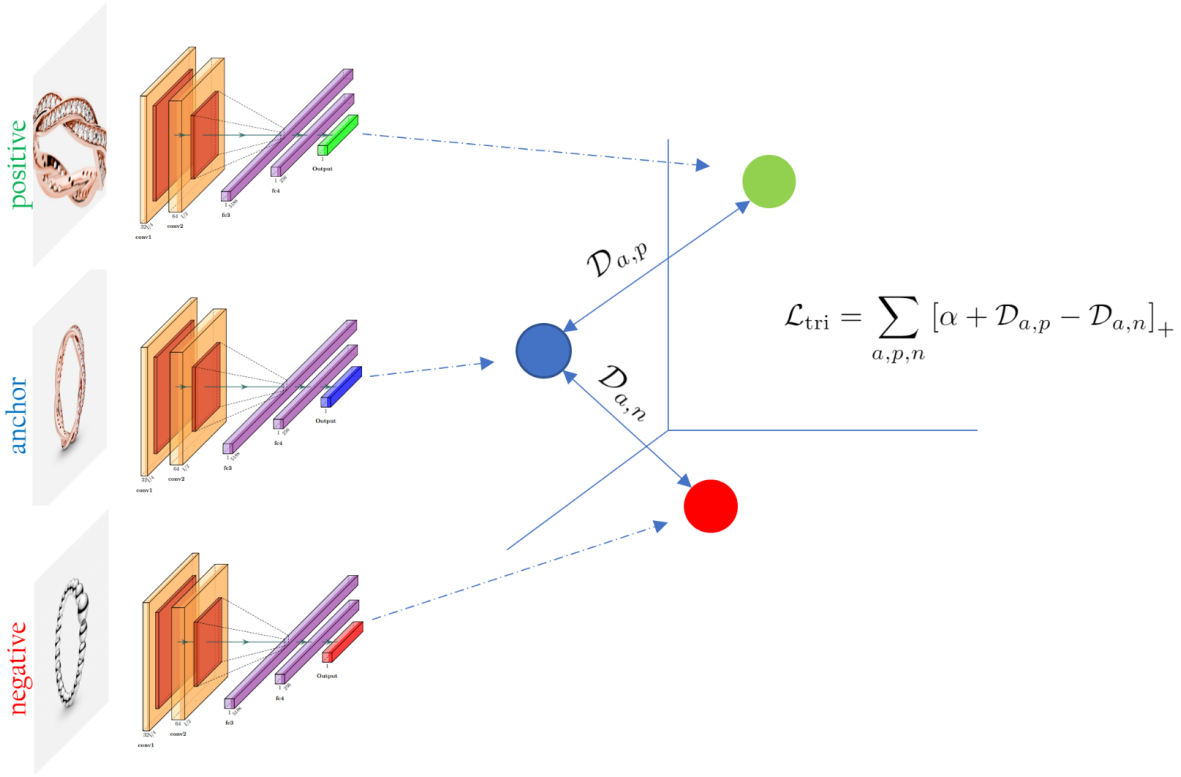


Figure 18: Triplet Ranking model architecture illustrated. $\mathcal{D}_{a,p}$ and $\mathcal{D}_{a,n}$ denotes the distance from the anchor to the positive and negative image respectively.

Figure 18 shows the triplet images being fed through three different networks that outputs embeddings in a latent space. Each output is represented by a circle and the loss function pushes the green circle closer to the blue to minimise the distance from anchor to positive and maximise the distance from anchor to negative.

4.3 Image segmentation

Images taken by a user contain a lot of other things than just the jewellery piece, for our use-case this is considered noise and should therefore be removed. The other objects in the image will make the model perform worse because it creates an embedding for the entire image. Ideally we would want the query image to be in same format as the training images, i.e. just the jewellery on a pristine white-grey background with nothing else. To accomplish this, the image

segmentation algorithm Detectron2 by Facebook AI research [Wu+19] is used. It implements state-of-the-art object detection algorithms and is able to perform semantic segmentation, which is a segmentation method that places a mask as well as a bounding box over the object it detects. We use this mask to select the area of the image which should be preserved. The regions outside the mask are coloured white-grey to match the catalogue images. By default, the model is not able to detect jewellery, so it was retrained with jewellery images. The process is as follows: firstly, a small new dataset with 44 images was created. The images in the dataset are manually segmented by drawing polygons around the jewellery, essentially constructing a hand drawn mask of the piece of jewellery. An example of this is shown in Figure 19.

Figure 19: A hand drawn mask of a piece of jewellery on an image from https://pixabay.com/photos/wedding-rings-rings-gold-rings-2252438/

Secondly, Transfer Learning is used: the pretrained Detectron2 model is retrained with the new data to create the desired jewellery class. It can then label images as seen in Figure 20. It is then simply a matter of extracting the jewellery, by using the mask and bounding box information. The final result looks is seen to the right in Figure 20.

Figure 20: Left: An example of a prediction from the image segmentation algorithm, Right: The necklace extracted from the image, note the same background colour as the training images has been applied. From https://unsplash.com/photos/WzxjTSGMoYA

5 Experiment

5.1 Evaluation metrics

In order to evaluate the performance of the Triplet Ranking Network and the Variational Autoencoder, standard precision ranking measurements such as mAP and recall at K are utilised as well as a human performance score. As a control, we define a third model, denoted Random Selection, which simply outputs five random images from the training dataset as recommendations for a query image.

5.1.1 Recall at K

The recall at K accuracy is utilised, since this is a commonly used accuracy metric for reports in the field [XSP20; Che+17; Soh16; Sik+20; Che+20]. For each evaluation image, x^i, all embeddings of images in the catalogue are ranked using k nearest neighbours with squared euclidean distance. This ensures that K recommendations are retrieved based on the highest similarity scores between the embeddings in the latent space and the embedded image, $f(x^i)$. The recall at K score for the specific test image is then 1 if at least one of the retrieved recommendations belong to the same category as x^i and 0 if this is not the case. This is averaged over all the test images to get a recall at top-K score for the model. Using several K-values provides a precision curve as a function of K, as also seen in [Che+20]. It can be defined as:

$$\text{recall} = \frac{|\{\text{relevant images}\} \cap \{\text{retrieved images}\}|}{|\{\text{relevant images}\}|} \tag{20}$$

i.e the fraction of actual matches among all matches.

5.1.2 Mean Average Precision

The *mean average precision measure*, mAP for short, is the most popular way of reporting the performance of a CBIR system. It is the mean of the average precision over all queries for a given model. Precision in this context is defined as

$$\text{precision} = \frac{|\{\text{relevant images}\} \cap \{\text{retrieved images}\}|}{|\{\text{retrieved images}\}|} \tag{21}$$

Immediately, we arrive at a problem: what is a relevant image? We calculate the metrics for two different definitions of a relevant image. The first definition is that an image is relevant if it belongs to the same sub-category such as 'dangling charm' or 'stackable ring'. The second definition is that a product is relevant if it has the same colour and type as the query image.

Average precision at k is the precision averaged over the all queries. k is the cutoff which precision is calculated at. The number of relevant queries is usually selected in the range $k \in [0, 50]$. The cutoff is done because there is a premonition that you likely are not using the retrieved images below a certain point. This is supported by [Jac14] that states that "according to a 2014 study from Advanced Web Rankings, more than 67% of all clicks on search engine

page results go to the top five listings"

$$\mathrm{AP}_n = \frac{1}{|P^+|} \sum_{k=1}^{n} P_k \cdot rel_k \tag{22}$$

Where $P(k)$ is the precision at k, $|P^+|$ is the total number of ground truth positives, $rel(k)$ is an indicator function, taking the value 1 if the image at rank k is relevant and 0 otherwise. The mean average precision is then the mean over all average precisions to get a single number describing the search effectiveness.

$$\mathrm{mAP} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i \tag{23}$$

5.1.3 CMC at rank K

The Cumulative matching characteristic / curve at rank K, here referred simply as rank-K, is a metric that describes whether or not the recommendations at at cutoff rank-K contain the same label. It is most commonly used in the face re-identification task but is also applicable here. One must consider that the scores are not comparable since for a face re-ID task there are usually many different labels (each person), in our case there are only a few labels, meaning that the for our use case the rank-K will easily be very high.

The CMC Curve plots the TPIR against ranks. The True Positive Identification Rate (TPIR) is the probability of observing the correct identity within the top K ranks.

$$Acc_{k:} = \begin{cases} 1 & \text{if top-}k \text{ ranked gallery samples contain the query identity} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

Where the index k : indicates that the recommendations after the first found k are set to one, since if the k'th image is the correct labels, all the rank-K higher also contain the k'th image. In the end the score is averaged over all queries. This is a much more search engine related metric than mAP, because the ranks are what matters.

5.2 Hyper parameter selection

A grid search is used to compare TRN models with different hyper parameters in order to choose the best one. The data is split into a training, validation and test data set, where the test proportion is 10%. Identical test data is used for all models. The hyper-parameters that are searched through are the number of epochs as well as the learning rate. The best hyper parameters are selected based on the models with the lowest validation loss.

5.3 Batch sampling strategy and margin selection

As mentioned in section 4.2.4, the *Batch Hard*, *Batch Semi-hard* and *Random Hard* strategies are three different sampling methods. Along with choosing the best sampling strategy, an appropriate margin value for each strategy is selected. The margin values (0.1, 0.5, 1.0) are

utilised and the experiment is set up similar to [HBL17, Table 1]. The strategies are evaluated and compared using their respective mAP, rank-1, rank-5 and recall scores rather than their loss values due to the fact that the margin value is a hyper-parameter for the triplet loss function in eq. 18 and 19. The losses will naturally differ because of the margin values and these cannot be compared against each other.

5.4 Human Evaluation

Due to the nature of our problem being visual similarity and not classification, it might be misleading to only use the accuracy measurements mentioned above. In order to more accurately estimate the actual semantic precision, we used human evaluations. Since this is a costly way of evaluating the models, we only use human performance rating for the comparison of our final models.

Evaluators were volunteers selected from friends and acquaintances that were from a large range of ages and did not necessarily have any experience with CBIR or machine vision problems in general.

5.4.1 Experimental setup

30 query images were selected across 15 different products. For each of the selected products, a catalogue and a wild image was presented to the human evaluator. The wild images were acquired from the UK Amazon website and are found in the review section of the Pandora product displayed on the website. An example of a product used in the setup can be seen in Figure 21.

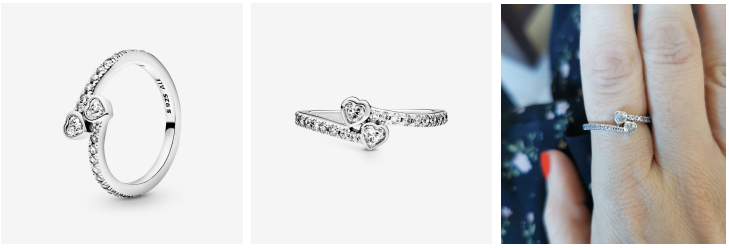


Figure 21: A product used in the setup: both the catalogue images and wild image are displayed to the human evaluator.

For every query image presented to the evaluator, we chose the 5 most relevant images outputted for each model: The Triplet Ranking Network with the Batch Semi-Hard sampling strategy, the Variational Autoencoder and Random Selection, and asked them to rank the recommendations from 1-3 according to how well they found the recommendations to be. A rank of 1 denotes the best recommendation of the the three and 3 denotes the worst. An example of the questionnaire setup is shown in Figure 22.

Figure 22: Recommendations for a query image (also seen in Figure 21) given by the three models Random (A), VAE (B) and TRN (C) respectively. The "radaraadr" string is used in the backend to identify the picture.

The participants will be ranking 30 of these sets of recommendations and they will not know which of the three models produced the recommendations. Therefore the letters A, B and C are shuffled across the different models across the questions. The recommendations within each letter were shuffled such that the leftmost one was not necessarily the most similar one. All recommendations used in the questionnaires can be seen in the appendix in section E.

5.4.2 Control of image segmentation

Furthermore, the significance of implementing image segmentation using Detectron2 for the 15 wild query images is investigated. As a control, two questionnaires for the human evaluators are set up; one where the wild images are not segmented before feeding them into the models, meaning the jewellery piece is not cropped out, and one where image segmentation is implemented. Otherwise the setup, described above, is the same for both questionnaires.

5.4.3 Statistical evaluation

In order to conclude which of the three models the human evaluators ranked the highest, the difference in the mean ranks from the questionnaires are investigated. This can be done using a Friedman's test, which is the non-parametric version of the Two-Way ANOVA test, which assumes that the dependent variable is normally distributed. The Friedman's test does not assume that the dependent variable comes from a normal distribution and the variable can be both ordinal and continuous [Lær18]. Since our rank score is ordinal and we cannot assume that these are normally distributed, the Friedman's test seems to be an appropriate fit to investigate the following hypotheses:

- H_0 = The three models have equal mean ranks: $\mu_{Tri} = \mu_{VAE} = \mu_{RS}$
- H_1 = The three models have different mean ranks: $\neg(\mu_{Tri} = \mu_{VAE} = \mu_{RS})$

In the Friedman's test the two independent variables are usually categorical, which together give rise to the either ordinal or continuous dependent variable. In [Gig19], the following example is presented in the Friedman' test setting: 10 different wines are to be examined and ranked by 20 wine connoisseurs and are given a rank between 1-10, where a rank of 1 and 10 denotes the best and worst wine respectively given by a connoisseur. The two categorical, dependent variables are the person ID and the kind of wine, and the dependent variable is ranking score.

In our case, the dependent variable is also the ranking score and one of the independent variables is the choice of model: TRN, VAE and Random Selection. The dependent variable denoting the person ID is defined slightly different because as we vary the 30 query images presented to the human evaluators, the ranking of the the three models will be different each time, unlike the 10 wines which should taste the same each time a connoisseur tastes them. To solve this problem, the person ID variable is merged together with the 30 query images, so that each time a human evaluator ranks a new query image, it would correspond to a human evaluator with another person ID. This is done so the effect of the model is independent of the person ID and the query image, meaning there are not any interaction effects present.

Three different statistical setups are defined for the Friedman's test, where the following samples from the human evaluation are used:

- Setup 1: All 30 query images
- Setup 2: The 15 catalogue images
- Setup 3: The 15 wild images - both with and without image segmentation, (control of image segmentation)

The rankings from the three setups are considered respectively in the Friedman's test. We investigate if a significant difference in the models is found in all three cases.

The statistical analyses are conducted in R with the built-in Friedman's test function, which outputs the value of the χ^2 test statistic and the p-value, which is evaluated on a 95% confidence level. These statistics will indicate whether a significant difference is found between all three models, however they will not provide insight on as to which models perform differently according to the human evaluation in each of the statistical setups. An appropriate post-hoc test used frequently for Friedman's tests is the Wilcoxon signed-rank test where all three models are compared against each other. Since we are doing multiple comparisons, the results are corrected using a Bonferroni adjustment. With a 95% confidence level, a significance level of 0.05 is used, but with the Bonferroni adjustment the new significance level with three comparisons becomes $\frac{0.05}{3} = 0.017$ [Lær18]. Hence, we will conclude that there is a significant difference between two models if the pairwise p-value is below 0.017.

6 Results

In the first section, the results of the gridsearch and sampling strategies for the triplet deep ranking models are shown. Thereafter the optimal triplet ranking model is compared with the Variational Autoencoder and the Random Selection model.

6.1 Triplet Ranking Network

6.1.1 Hyper parameter selection

The following figures display the average validation and training losses for different hyper parameters for the Semi Batch Hard selection strategy. Figure 24 is a zoomed in version of Figure 23 and the green line shows the hyper parameter selection used in the questionnaire. Table 15 in the appendix shows the top 20 best parameter selections.

Figure 23: The training and validation loss across different hyperparameter selections for the Semi Batch Hard selection

Figure 24: The training and validation loss across different hyperparameter selections for the Semi Batch Hard selection

Across all sampling strategies, a learning rate of 0.0001 seems to yield better results than higher learning rates. Additionally, increasing the number of epochs beyond 15 did not seem to improve the average validation loss. Therefore, when we compare the three different batch sampling strategies in the next section, all are trained with the 0.0001 learning rate and 15 epochs.

6.1.2 Batch sampling strategy

The following tables show mAP, rank-1 and rank-5 for the different negative image selection methods across several different margins. Since both the sampling method as well as the margin affects the calculation of the loss, they are compared based on these results instead of the validation loss. For all mAP rank - K calculations the train/test split was used such that all query images were from the test set but matches were allowed on the combined dataset.

The calculations have been done based on two different criteria. In table 5, the true positives are defined as products belonging to the same sub-category. A few examples of subcategories are: 'stackable ring' and 'dangling charm'. There are 79 of such categories.

Table 6 shows the calculation done for colour and metal type, meaning that a true positive is a recommendation with the same colour and metal type as the query image. A small amount of pre-processing was done such that different carats of gold were aggregated, as well as some other visually similar categories. In total there are 31 classes, a few examples of colours and metal types are: 'silver charm' and 'silver earring'.

Table 5: mAP and rank-at-k calculated based on subcategory of jewellery.

	margin 0.1			margin 0.5			margin 1.0		
Subcategories	mAP	rank-1	rank-5	mAP	rank-1	rank-5	mAP	rank-1	rank-5
Triplet Batch Hard	22.18	4	51	24.96	8	49	24.77	4	43
Triplet Batch Semi-hard	46.6	44	69	43.98	41	68	24.77	4	43
Triplet Batch Random Negative	50.07	46	76	49.28	47	78	51.66	50	79

Table 6: mAP and rank-at-k calculated based on colour and type of jewellery.

	margin 0.1			margin 0.5			margin 1.0		
Colour & Type	mAP	rank-1	rank-5	mAP	rank-1	rank-5	mAP	rank-1	rank-5
Triplet Batch Hard	34.1	1	39	33.07	4	40	21.83	16	48
Triplet Batch Semi-hard	58.39	55	81	53.01	48	78	54.09	52	77
Triplet Batch Random Negative	63.02	59	86	53.01	48	78	59.19	56	0.82

6.2 Comparison of the models using mAP and rank-K

Table 7 and 8 shows the mAP and rank-1 & 5 scores for the two models compared, as well as a method which randomly samples recommendations from the same subcategories as well as type and colour as defined above. The random sampler is very poor in both of the match types. The table shows that the TRN with semi-hard batching outperforms the VAE in both cases. In the case where subcategories are used, Table 7, the Deep Ranking fares better across all metrics.

Subcategories	mAP	rank-1	rank-5
Triplet Batch Semi-hard	46.6	44	69
VAE	51.11	51	80
Random	22.24	14	46

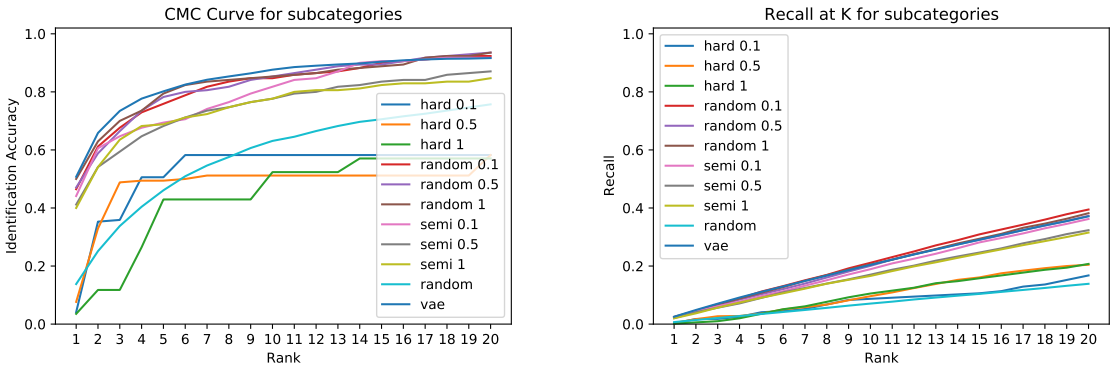
Table 7: Comparison of models using the mean average precision score at rank-20. using sub-categories

In the type and colour match case, Table 8, the VAE just slightly edges out the Random sampler in mAP and rank-1, while it is beat in rank-5. On the contrary, the Triplet method outperforms the VAE by a large margin close to 70% relative improvement in mAP. At rank-1 it is also over 3 times better at retrieving an item of same match type.

Colour & Type	mAP	rank-1	rank-5
Triplet Batch Semi-hard	58.39	55	81
VAE	47.08	43	78
Random	29.66	21	54

Table 8: Comparison of models using the mean average precision score at rank-20. Using categories and metal-type

A graphical comparison using subcategories, shown in Fig. 25, reveals that the Variational Autoencoder performs better than random at rank-K retrieval (Fig. 25a) but has recall similar to the worst deep-ranking method (Fig. 25b). In both metrics, the Deep Ranking methods outperforms the VAE and Random. The TRN with hard negative sampling does exhibit some weird behaviour, otherwise the other TRNs lie on top of each other.

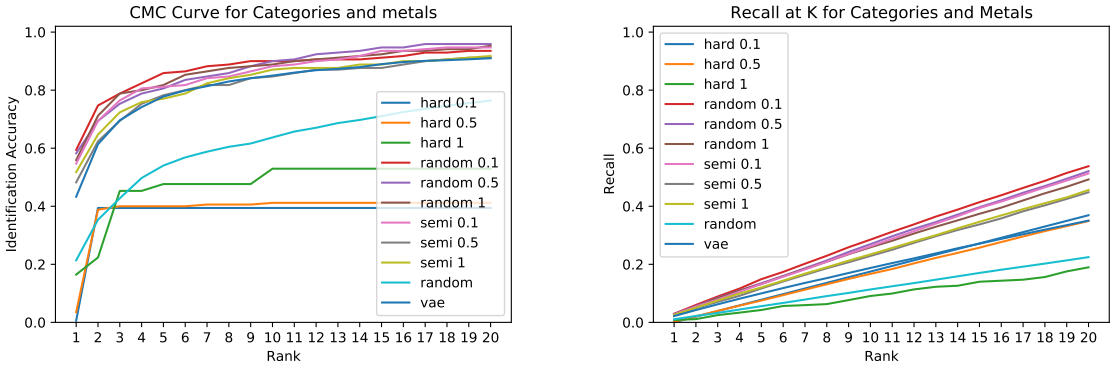


(a) Cumulative matching curve

(b) Recall at rank k

Figure 25: CMC and recall - K for subcategories. Numbers indicate margins. (The VAE is the top dark blue)

With categories and metals the same pattern is again observed, Figure 26. Here, the Deep Ranking methods is better at lower ranks but the VAE performs worse. At middle [5,10] ranks, the VAE overtakes the random method, but has similar recall.



(a) Cumulative matching curve

(b) Recall at rank k

Figure 26: CMC and recall - K for colour & type. Numbers indicate margins.

6.3 Statistical comparison of models

A total number of 45 people answered the questionnaire with cropped wild images, and 80 people answered the one without cropping. For each of the 30 query images in the questionnaire with cropping, 3 rankings were acquired, resulting in $45 \cdot 30 \cdot 3 = 4050$ number of rankings in total. In the no-cropping questionnaire, $80 \cdot 15 \cdot 3 = 3600$ rankings were acquired.

In the following section, the statistical analyses of the three different setups described in section 5.4.3 based on the human evaluation are presented. As stated in section 5.4.3, the p-values from the Wilcoxon signed-rank sum test will be evaluated on a significance level of 0.017 due to the Bonferroni adjustment.

6.3.1 Setup 1: All images

The mean ranks and the Friedman's test statistics for the three models when sampling all 30 query images are as follows:

	Mean Rank
Triplet	1.754
VAE	1.57
Random	2.676

	All models
χ^2-value	946.94
p-value	$< 2.2 \cdot 10^{-16}$

Table 9: Mean ranks and the Friedman's test statistics for setup 1

The mean ranks for each model along with their respective bootstrapped 95% confidence level are illustrated in Figure 27.

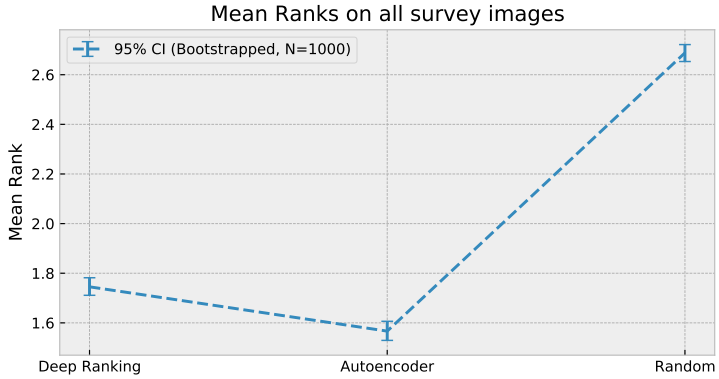


Figure 27: Mean ranks for each model including their 95% confidence level for setup 1

The very low p-value from the Friedman's test suggests that there is strong evidence against the null hypothesis, H_0, and the alternative hypothesis, H_1, is accepted. This illustrates that a significance difference is found between the performance in the three models.

The pairwise p-values from Wilcoxon signed-rank sum tests show which models differ in performance. The results are seen below in Table 10.

	p-value
Triplet vs VAE	$2.3 \cdot 10^{-14}$
Triplet vs Random	$< 2.2 \cdot 10^{-16}$
VAE vs Random	$< 2.2 \cdot 10^{-16}$

Table 10: Summary of the pairwise Wilcoxon Rank Sum Tests for setup 1

All p-values indicate a significant difference between the groups, which also can be seen in Figure 27, where the confidence interval of the mean ranks of the models do not overlap. This suggests that the Variational Autoencoder with the lowest rank of 1.57, performs best in the human evaluation when considering all 30 query images.

6.3.2 Setup 2: Catalogue images

Similarly to the results in setup 1, the mean ranks, Friedman's and Wilcoxon test results when only considering catalogue images in the analysis are shown below.

	Mean Rank
Triplet	1.698
VAE	1.473
Random	2.83

	All models
χ^2-value	714
p-value	$< 2.2 \cdot 10^{-16}$

Table 11: Mean ranks and the Friedman's test statistics for setup 2

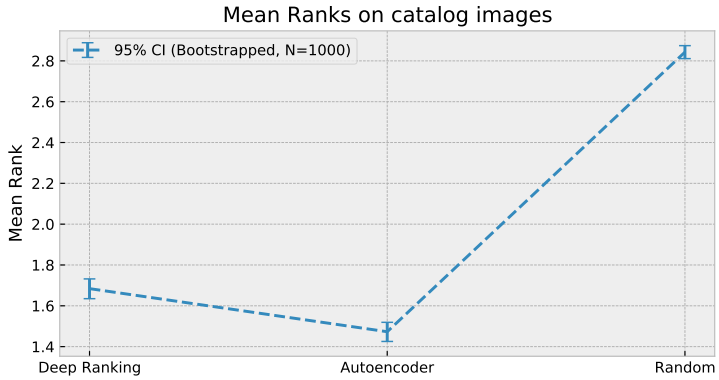


Figure 28: Mean ranks for each model including their 95% confidence level for setup 2

An equivalent conclusion to setup 1 is reached regarding the p-value from the Friedman's test - a significant difference is found between the models when only considering the catalogue images.

The results of the Wilcoxon signed-rank sum test are the following:

	p-value
Triplet vs VAE	$2 \cdot 10^{-11}$
Triplet vs Random	$< 2.2 \cdot 10^{-16}$
VAE vs Random	$< 2.2 \cdot 10^{-16}$

Table 12: Summary of the pairwise Wilcoxon signed-rank sum test for setup 2

In this setup, the p-value in all pairwise tests all indicate a significant difference between the models and the same conclusion is reached as before - the VAE outperforms both the TRN and Random model, which is illustrated in Figure 28.

6.3.3 Setup 3: Control of Wild Images

Lastly, the statistical results when sampling only the wild images in the questionnaire with cropping is shown below. Additionally, the mean ranks of the non-segmented wild images from the control questionnaire is shown alongside the mean ranks of the segmented wild images.

	Mean Rank
Triplet	1.81
VAE	1.668
Random	2.521

	All models
χ^2-value	282.17
p-value	$< 2.2 \cdot 10^{-16}$

Table 13: Mean ranks and the Friedman's test statistics for setup 3

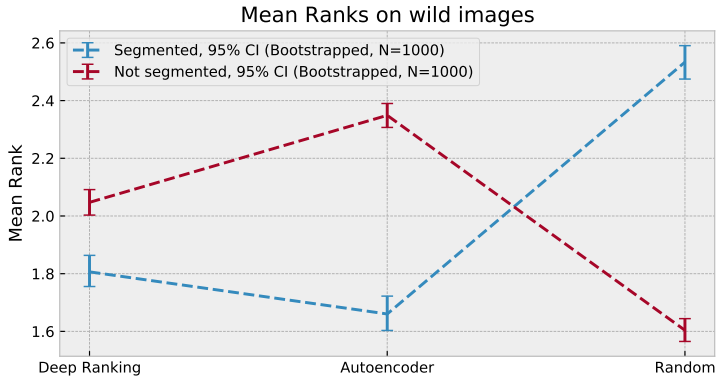


Figure 29: Setup 3: Mean ranks for each model including their 95% confidence level.

Again, the Friedman's test shows a significant difference is found between the models, and the p-values from the pairwise Wilcoxon tests from table 14 also suggest that a difference is found between all three models. In this last setup, both the TRN and Random model are outperformed by the VAE that has a mean ranking of 1.668.

Furthermore, the importance of segmenting the images can be argued with the results shown in Figure 29. When the jewellery pieces in the wild images are not cropped out, the Random Model outperforms both the TRN and the VAE, which performs the worst out of the three.

	p-value
Triplet vs VAE	$8.3 \cdot 10^{-5}$
Triplet vs Random	$< 2.2 \cdot 10^{-16}$
VAE vs Random	$< 2.2 \cdot 10^{-16}$

Table 14: Summary of the pairwise Wilcoxon Rank Sum Tests for setup 3

6.4 Detectron2 Performance

The cropping of the jewellery for 9 of the wild images used in the questionnaire are shown below.

(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 30: Top row: the wild images. Bottom row: the images after the Detectron2 image segmentation

Images (a), (g) and (h) are very well-cropped, while (c), (d) and (i) are extremely poorly cropped. Both (c) and (i) are croppings of something other than the jewellery.

Based on our own evaluation of the quality of 15 croppings from wild images, where the categories were 'good', 'okay' and 'bad', 60 % of the images were well-cropped, 20 % were okay and 20 % were bad. The croppings in the category 'bad' did not find the right thing in the image.

6.5 Showcase of model recommendations

Sets of recommendation of query images from the three models are displayed below. The query images consist of a catalogue and wild image of the same product. Additionally the models are presented with a product that is not a part of Pandora's e-catalogue.

6.5.1 Triplet Ranking Network

Figure 31: 10 nearest images to a catalogue (top), a wild (middle) and a product not sold on Pandoras website (bottom) computed by the Triplet Ranking network.

6.5.2 Variational Autoencoder

Figure 32: 10 nearest images to a catalogue (top), a wild (middle) and a product not sold on Pandoras website (bottom) computed by the Variational Autoencoder

6.5.3 Random Selection

The following figure shows a random selection of images from the data set, to provide a visual baseline for the recommendations above:

Figure 33: 12 randomly sampled images.

6.5.4 PCA from the Triplet Ranking Network

Figure 34 below shows the computed embeddings of catalogue images projected onto a 3D space using Principal Component Analysis. The Triplet model seems to be capable of separating the colours and shape of the products quite well. The different colours are separated front / back and grouping of fatter / thinner pieces of jewellery are present in both the rose gold section and the silver section. Patterns and textures also seem to be apart from each other. Note that only a subset of images in the dataset are shown.

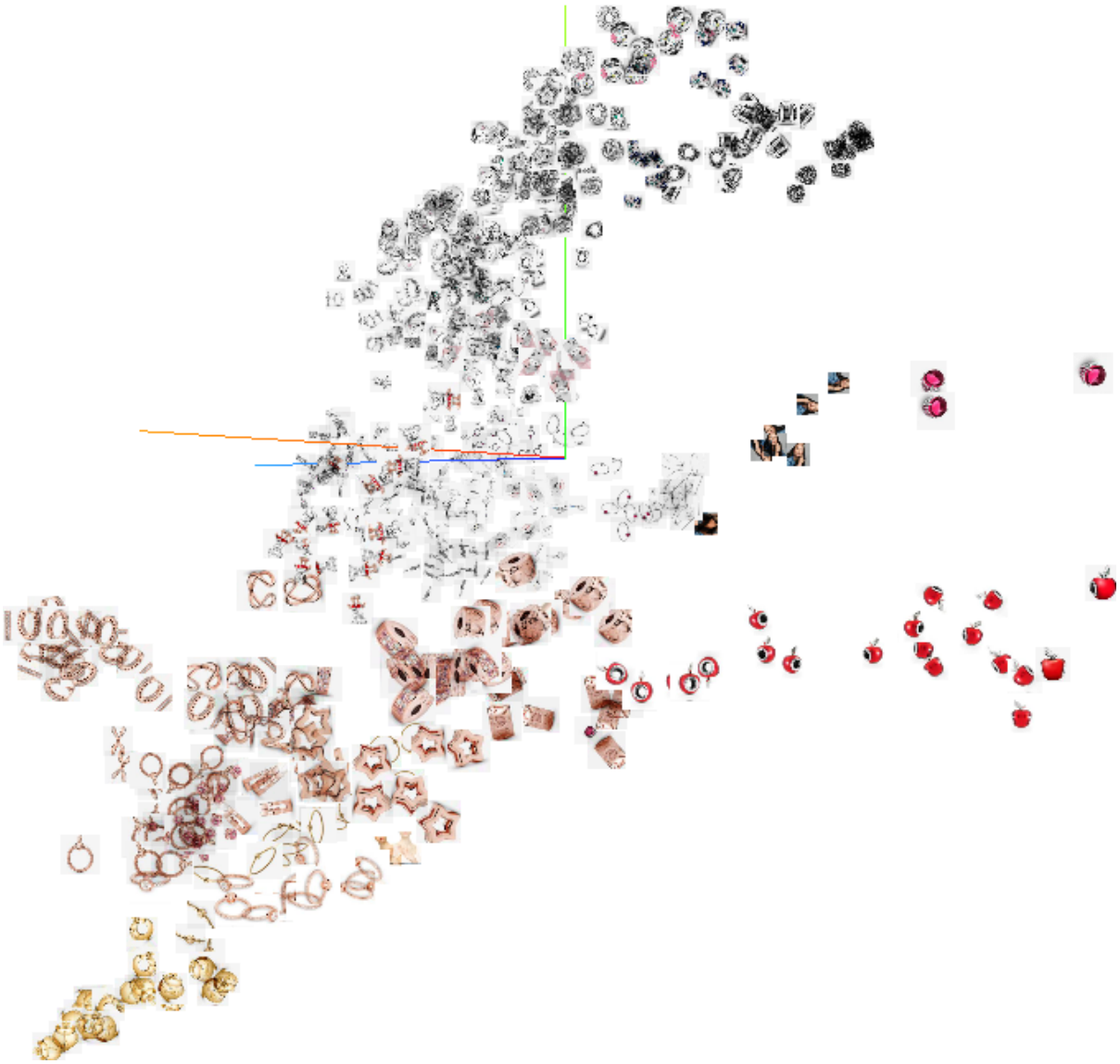


Figure 34: 3D PCA with image sprites from the catalogue.

7 Discussion

The main takeaway from the experiment results is that the TRN outperforms the VAE on all *quantitative* measures of performance, while the opposite is true for the survey results, which serve as a *qualitative* measure of performance.

7.1 Difference in model performance

7.1.1 Batching Sampling Strategies in the TRN

The *Batch Hard*, *Batch Semi-hard* and *Batch Random Negative* strategies are compared using the results from table 5 and table 6, where the mAP and rank-K scores are based on recommending products from the query images from the same subcategory and the same colour & type of the query images respectively.

Generally for the two cases, the evaluation metrics are higher when based on the colour & type of jewellery than the scores based on the subcategory alone, which is understandable since there are 31 and 79 classes respectively in the two cases.

The Batch Hard rank-5 scores are higher than both its own mAP and rank-1 scores as well as the rank-5 scores of the Batch Semi-hard and Random. Batch Hard is however outperformed by both the Semi-hard and Random batching in terms of the mAP and rank-1 scores - this is true for all margins and in both calculations based on subcategories and colour & type.

The hard sampling method uses the triplets where the embedding of the negative is closer to the anchor than the positive as training points for the model. Since positive images are defined as the ones belonging to the same product, it would be natural that the hard negatives are images of another product that looks similar in some way (metal, shape etc.). Since the model only trains on hard negatives, it pushes these further away, which might cause the bad performance in terms of our evaluation metric. The performance might simply be caused by the goal of the model and the evaluation metric being misaligned. The reason for keeping the evaluation metric in this format is that it seems to be more aligned with how the participants in the questionnaire value recommendations.

The Semi and Random hard strategies have almost equal performances across the three different evaluation metrics. This is also seen in Figure 25 and 26, where the CMC and recall scores for the two strategies are nearly the same. We choose our final model of the TRN to be implemented with the Batch Semi-hard sampling strategy and a margin of 0.1 yields overall better results than higher margins.

7.1.2 Comparison of TRN, VAE and Random

The mAP, rank-1, and rank-5 scores are compared with subcategories in Table 7 and color % type in Table 8 across models. Across all evaluation metrics the TRN outperforms the VAE and Random model. Figures 25 and 26 show equivalent results. The overall higher CMC and recall scores of Deep Ranking methods shows that a larger proportion of relevant items are returned as supposed to the VAE and Random model.

The VAE fails to perform any better than the random sampler as their respective mAP and rank-5 scores based on color & type are almost equal. The random sampler even achieves higher rank-1 scores than the VAE in both cases. However, the VAE performs better with higher mAP and rank-5 scores based on subcategories. The same tendency is also seen in Figures 25 and 26, where the scores of VAE and random sampling are usually the same except that the VAE achieves better subcategory recall results.

The reason why random sampling performs better or equal to the VAE could be that the random is just lucky to retrieve a relevant item in high ranks, but it is likely also due to the imbalance in the data set. The test data set consists of 10% of the total data set and is randomly sampled from the pool. Since aggregating the dataset by colour & type gives a very unbalanced set as shown in Table 1, this can distort the results shown in Table 8 in a way where the random sampling performs better than expected.

The recommendations from the VAE seen in both the results section and the questionnaire seem to suggest that the VAE favours the type of jewellery more than the colour - a further exploration into this idea is described below. Since the mAP and rank-K scores are calculated based on a true positive being a recommendation with the same colour & type as the query image, the VAE struggles to obtain high values in these evaluation metrics as seen in table 6.

7.1.3 Statistical comparison in the human evaluation experiment

Although we have shown that the Triplet Ranking model far outperforms the Variational Autoencoder (and [Pas+20] agrees), this is not what our human evaluation results show. There might be several reasons for this. Firstly, it might be due to the selection of positive images for the triplet pairings. Positive images are defined in our model as the images belonging to the same product as the query. This might not be entirely representative of how we wish the structures in the latent space to look. Since the push strategy of the triplet loss is to push all negative images away, this means that the products with the same colour and type will be pushed away from the query image in the embedding space with the same force as completely irrelevant images. This can be fixed by either redefining positive images as being the images with the same colour and type as the query image or utilising a quadruplet loss in order to create a *weak* and *strong* push strategy.

A possible explanation for why the autoencoder performs better on this task may be how wild images are treated. A successful cropping will preserve the shape of the target object very well, and anecdotal evidence from inspecting recommendations suggest that the autoencoder mainly considers shape when making recommendations. On the contrary, the colours of the object are still subject to bad lighting, blur etc. even when cropped, and colour seems to be the main factor when the triplet model makes recommendations.

7.2 What makes for a good recommendation?

Arguably one of the most important things to keep in mind when designing a recommendation system is, what a good recommendation looks like. Although this seems like a simple statement, answering the question has proven to be far from that. We found that the quality of a specific recommendation depended quite significantly on the person assessing it.

In order to get an understanding of what a good recommendation is, we asked more than 80 people to describe how the quality of a recommendation was assessed. There seemed to be six different features that people reported to use often in the assessment of the quality. The six features were: 'form', 'type of jewellery', 'colour', 'texture of product', 'context of product' as well as if the recommendation contained the exact same product as in the query. The context of the product refers to the more semantically abstract context such as the collection the product came from or the theme of the product (for example other Disney themed products if a Mickey Mouse charm was presented). Among these features, the colour and type of the jewellery recommended were the most important ones along with the precision (if the same product as the query was recommended).

Although the majority agreed that these three qualities were the most significant, they disagreed on the order of significance between them. Some believed that colour was more important than the type, while others had the opposite opinion. Some thought that the recommendation with the precise product in it, but other seemingly random products recommended as well, was still better than a recommendation filled with the same product types.

It makes it difficult to design a well-performing recommendation system, when there is no golden rule to what a good recommendation looks like, but it seems that getting the colour and type correct is crucial, while the other features such as texture and context are not weighted as high.

7.3 Differences in model recommendations

The VAE generally tended to focus on the structural shapes in the image, while the deep ranking network also seemed to take the colour of the jewellery into account. This might be due to the fact that the deep ranking model is rewarded when the embedding of an image is close to embeddings of other images of the same product in the latent space. Since other images of the same product always have the same colour, but might have a different orientation and shape in the image, the colour will be a very important feature for the deep ranking model. For the VAE, the shape and size of the jewellery is a very important feature. This is likely because the VAE is rewarded when the pixelwise difference between the outputted image are minimised, which it will be when the shape, orientation, size and position of the outputted image matches the original. To create embeddings that take both type, shape and colour into account, we suggest creating a Triplet Enhanced Variational Autoencoder and will elaborate on how to do it in the further research, section 8.3. Another approach to enforcing that the deep ranking model should take the type of jewellery into account could be to utilise a quadruplet loss, where the query should lie closer to different images of the same product than another product of the

same type in the latent space, while the image of the same type should lie closer to the query than that of a different type. This approach is also elaborated upon in the further research section.

Interestingly, we made an analysis of whether the people that said that colour was more important than type for a recommendation also preferred the Deep Ranking model over the VAE, but found no effect of this.

7.4 Performance of the Detectron 2 model

In order to evaluate the importance of using image segmentation to crop out the jewellery in the wild images, we made two surveys. One of the surveys used the Detectron 2 on the wild images before getting a prediction, while the other survey did not (control survey). The recipients of the questionnaire where not informed about the image segmentation process. In the control survey, randomly selecting products from the catalogue resulted in better recommendations than any of the models. The deep ranking model performed slightly better than the VAE on this survey. When using the Detectron 2 segmentation, both models where significantly better than the random selection, suggesting that the image segmentation of wild images is a very important step in the pipeline.

As seen in the results section in Figure 29, the performance of the Detectron 2 model varies drastically across the wild images. In order to get better recommendations, it would therefore be worthwhile to improve the Detectron 2. This has not been a priority in this project, although looking at the results from the two surveys, perhaps it should have been.

Detectron 2 is not trained for extracting jewellery, so in order to use it, we retrained it on images with carefully constructed hand drawn masks. Since the creation of the masks is a very time consuming and therefore expensive task, the data set we used for retraining it consisted of 44 images, which is an extremely small amount. Creating a larger jewellery data set for retraining the segmentation model is therefore highly recommended.

An important aspect to keep in mind when creating a data set for this purpose is to make sure that it is representative for different colours of skin, because an unbalanced data set in this aspect could lead to a difference in performance of the image segmentation, and thus the quality of the entire recommendation being based on racial factors.

7.5 Improvements to the dataset: Accounting for noisy data

The data augmentation provided us a data set with clean, cropped images of jewellery on white background and with the same size for both our training and testing set. However, when a user uploads an image to the recommendation system, the image will most likely contain disturbances like complex backgrounds, bad quality camera or lighting and weird angles, resulting in what has been referred to as a 'wild image'. Since our models were not trained on these wild images and they are drastically different from the catalogue images, they can all be considered out of distribution data points. The results highlight this, since the two models perform worse than random on completely wild images with no image segmentation.

Deep neural networks are generally trained and evaluated using a closed-world assumption[9]. However, using the trained models in real-world tasks where the data is oftentimes noisier and does not follow the same distribution oftentimes leads to a significant drop in the performance. Since the use-case of this CBIR-system is for users to upload their own image, it can be expected that these images are out-of-distribution from the catalogue images we have trained on. To get a grasp of how much the drop in performance is between in-distribution and out-of-distribution data, we used 15 wild images and 15 catalogue images as our test pool in the questionnaire. The results showed that the mean rank of the random selection differed from the catalogue images, with the rank 2.8 and the wild images, with the rank of 2.6. The difference between the performance of the models compared to random for the wild images was significantly worse than that of catalogue images.

In order to ensure a better generalisation error when utilising the CBIR-system on user-generated images, the model should be trained on wild images along with the catalogue images. One way to obtain these wild images could be scraping images from Instagram and Pinterest using the hashtag 'Pandora' in addition with another keyword like 'charm'. Since it is an extensive task building such a dataset, this was not done for this project.

Another way of ensuring an improvement in the generalisation error is to do more extensive data augmentation to make the distribution of the training images and the wild images closer together. The data augmentation done in this project has primarily focused on creating rotational invariance by rotating the content of the images randomly. Further data augmentation should focus on creating training images that resembles wild images. This can be done by adjusting the saturation of the images to mimic the changing light conditions and introducing different amounts of blur in the images. Generally the images in the catalogue appear *shinier* than the ones a typical user might submit. This is a problem because it will skew what product images are found by the model.

Another problem in the wild images happens when the jewellery is worn. Even with a perfect extraction of the jewellery, the shape could still be significantly changed from the catalogue image because of obstruction in the image. See Figure 30 from the results section for an example of this type of obstruction, where the cropping is very well done, but the obstruction from the finger changes the shape of the ring making it look like it is a charm instead. A solution to this problem could be to randomly obstruct some of the catalogue images used for training.

When we used a photograph taken by ourselves as a query image which we knew was in the dataset the model was not able to find the same image within a top 10 sorting. This was primarily due the fact that the image we fed into the model had significantly different lighting than the one in the catalogue, see Figure 31.

[9] A closed-world assumption is the assumption that the test data distribution is similar to the training data distribution

7.6 Exploration and exploitation

Currently we have set the model up to work in way such that the products it recommends are the one which are the most similar to the query, which means it is exclusively exploitative. Since the goal was to create a system which can give recommendations we might imagine that this might not be the perfect way for the system to function in, since some users might prefer more variation in their recommendations. A simple way to introduce variation would be to take a weighted sample from the similar images, such that more similar images are more likely to be sampled. Another slightly more complicated approach would be identifying axes in the embedding space that represent specific features, where we would be able to give the user the option to slide sliders to generate recommendations which could, for example, be skinnier or have the same pattern but a different colour or be a necklace which looks similar to bracelet. In this fashion the user could control the exploration of the system themselves.

8 Further Research

We have found several points of improvement that given the time would be interesting to do some further research into, here sorted by how many extra all-nighters they would take to implement.

8.1 Different similarity conditions for triplet sampling

If the positives in the triplet sampling algorithm had been chosen based on colour and category of the product instead of only choosing images of the same product, the TRN would possibly have made recommendations that were more aligned with the performance criteria of the end-users, or at least the ones who participated in the survey.

8.2 Incorporating triplet loss as regularisation in the VAE

As mentioned in the discussion, the VAE generally tended to focus on the structural shapes in the image, while the deep ranking network also seemed to take the colour of the jewellery into account.

Modifying the loss function of the VAE to incorporate semantic information of the triplets could potentially improve the learned latent embeddings. This approach is also suggested by [YCS19; IHR18], where the method is referred to as TVAE for *Triplet based Variational Autoencoder* and TEA for *Triplet Enhanced Autoencoder* respectively.

Figure 35 shows what the structure of the TVAE looks like.

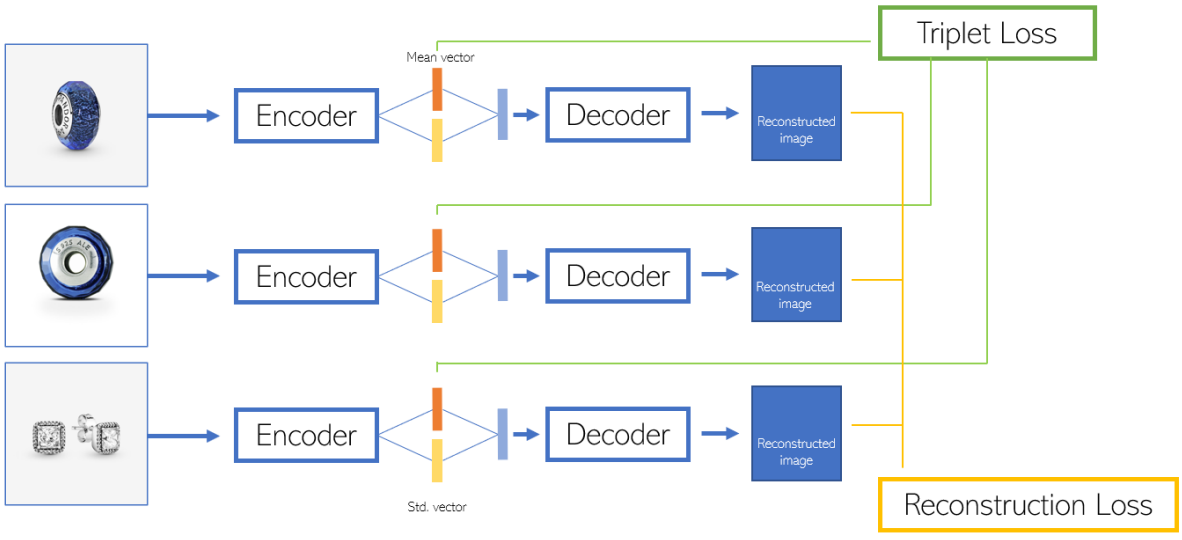


Figure 35: TVAE structure visualised. The illustration is heavily inspired from [IHR18]

Instead of using a single image as input as we have in the VAE, the model would be training using triplet sets of images as training instances, like the Triplet Ranking Network. In the TVAE there are three identical sub-networks that share parameters and weights in the same fashion as seen with the deep ranking. Given the input triplet (x_a^i, x_p^i, x_n^j), the encoder part

of the network outputs the embeddings (y_a^i, y_p^i, y_n^j) and the triplet loss is calculated. Lastly, the embeddings are fed through the decoder part of the network and the reconstruction loss is calculated. The loss per input instance is then a β-weighted loss between $\mathcal{L}_r$ and $\mathcal{L}_{tri}$. The network is then updated by gradients that are backpropagated through the network from the output layers.

The loss of the TVAE is:

$$\mathcal{L}_{TVAE} = \mathcal{L}_r + \beta \cdot \sum \mathcal{L}_{tri} \tag{25}$$

Where β is a variable that controls the importance of the triplet loss.

8.3 Using quadruplet loss to incorporate extra semantic information

In order to incorporate some semantic information into the deep ranking model, we chose to use triplet pairings as input as well as a triplet loss function. This gave the model information about what images belonged to the same product. This method is oftentimes utilised for ReID use-cases [Din+15], where images of the same person should lie close to each other in the latent space.

After conducting the experiment and asking people about what makes for a good recommendation, we found, that the most important thing is that a recommendation contains the specific product searched for or that the majority of the recommendations belongs to the same major category such as 'rings' or 'earrings' as well as have the same colour. The fine-grained details, like heart shapes in the jewellery, were less important.

Since the triplet loss only took the specific product ID into account, it was not taught to group rings together with other rings for instance. To improve upon this, we could have utilised a quadruplet network to use more foreknown information. In this case we could have chosen our quadruplets in a way where the positive image was of the same product, the semi-positive image was of the same type (ring, earring...) and the negative image was of a different product in a different category. Using a quadruplet loss function would then pull images of the same product closer to each other, while pulling them closer to other images in the same category as well.

This could be expanded to N-lets loss, where N represents the number of sub groupings. For this problem setting, it seems that Triplet Loss might be too few groupings, but since the data set is so small, more than 4-5 groupings is most likely not possible.

Due to the time constraints as well as the additional complexity, this was not feasible to implement, but we are very interested to see if this would make a significant change and hypothesise that it will.

8.4 Training on wild images

Training with images that are more similar to those that an end-user would submit might lead to a more robust model, but such a dataset specifically for jewellery does not exist.

9 Conclusion

Two different models, a Variational Autoencoder (VAE) and a Triplet Ranking Network (TRN), have been compared on their recommendations of jewellery products from an e-catalogue based on the visual similarity to a user-generated query image.

Based on human evaluation, the VAE performed significantly better on both catalogue and wild images than the TRN, while both models outperformed the random selection of products, Figure 10. On the contrary, the TRN far outperforms the VAE on both mAP, rank-1 and rank-5 (Figure 26), where the VAE performs roughly at random or a smudge better. The question is then which of these methods best quantify how semantically meaningful the embeddings are, and to that there is no clear answer. Seeing that the models have different strengths does however give credibility to the idea of a combined model like a Triplet Enhanced Variational Autoencoder (TVAE), which could unify the category and colour sensitivity from the TRN, and the shape and rotation preferences of the VAE.

When comparing the different triplet selection methods based on mAP and rank-K in Table 6, the Batch Semi-Hard strategy comes out on top, based on an overall impression of the three measures. In general there is no overwhelming difference between Semi-Hard and Random Negative sampling, but it seems that Hard sampling is directly counterproductive to performance on this problem.

A crucial step of the pipeline is the pre-processing, or segmentation, of images with challenging backgrounds. It was found, that no cropping of the wild images resulted in sub-random performance for both the TRN and the VAE on wild images (Figure 29). This clearly makes the segmentation model a necessity for this setup, but the classification models might be able to get by on their own if sufficient wild images were included in the training sets.

When considering all the possible model improvements presented, it would be wise to at least explore some of them before using any of the two models in a commercial setting. We hypothesise that utilising a quadruplet loss function or a TVAE could be better at grouping jewellery with the same colour, type, and shape together, ultimately resulting in better recommendations for the user, and higher turnover for the company.

References

[Bay+08] H Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014.

[Bra+12] Kaare Brandt Petersen Michael Syskind Pedersen et al. *The Matrix Cookbook*. Tech. rep. 2012.

[Che+17] Weihua Chen et al. "Beyond triplet loss: a deep quadruplet network for person re-identification". In: (Apr. 2017). URL: http://arxiv.org/abs/1704.01719.

[Che+20] Gal Chechik et al. *Large Scale Online Learning of Image Similarity Through Ranking*. Mar. 2020. URL: https://www.jmlr.org/papers/volume11/chechik10a/chechik10a.pdf.

[DA20] S. Deepak and P. M. Ameer. "Retrieval of brain MRI with tumor using contrastive loss based similarity on GoogLeNet encodings". In: *Computers in Biology and Medicine* 125 (Oct. 2020), p. 103993. ISSN: 18790534. DOI: 10.1016/j.compbiomed.2020.103993.

[Din+15] Shengyong Ding et al. *Deep Feature Learning with Relative Distance Comparison for Person Re-identification*. Tech. rep. 2015.

[Gig19] Gilles E. Gignac. *How2statsbook Chapter16*. 2019.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. "In Defense of the Triplet Loss for Person Re-Identification". In: (Mar. 2017). URL: http://arxiv.org/abs/1703.07737.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. IEEE Computer Society, June 2006, pp. 1735–1742. ISBN: 0769525970. DOI: 10.1109/CVPR.2006.100.

[Hig19] High-Level Independent Group on Artificial Intelligence (AI HLEG). "Ethics Guidelines for Trustworthy AI". In: *European Commission* (2019), pp. 1–39. URL: https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.

[HNM19] Tue Herlau, Mikkel N. Schmidt, and Morten Mørup. *Introduction to Machine Learning and Data Mining*. 1.4b. Dec. 2019.

[IHR18] Haque Ishfaq, Assaf Hoogi, and Daniel Rubin. *Workshop track-ICLR 2018 TVAE: triplet-based variational autoencoder using metric learning*. Tech. rep. Feb. 2018.

[Jac14] Madeline Jacobson. *How Far Down the Search Results Page Will Most People Go?* 2014. URL: https://www.theleverageway.com/blog/how-far-down-the-search-engine-results-page-will-most-people-go/.

[Jin+15] Yushi Jing et al. "Visual Search at Pinterest". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2015-August (Apr. 2015), pp. 1889–1898. URL: http://arxiv.org/abs/1505.07647.

[Kim+18] Wonsik Kim et al. "Attention-Based Ensemble for Deep Metric Learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11205 LNCS. Springer Ver-

lag, Sept. 2018, pp. 760–777. ISBN: 9783030012458. DOI: 10.1007/978-3-030-01246-5{_}45. URL: https://doi.org/10.1007/978-3-030-01246-5_45.

[KW-19] Diederik P Kingma Google, Max Welling, and Boston -Delft. "An Introduction to Variational Autoencoders". In: *Foundations and Trends R in Machine Learning* 12.4 (2019), pp. 1–18. DOI: 10.1561/2200000056.

[Lær18] Lærd Statistics. *Friedman Test in SPSS Statistics - How to run the procedure, understand the output using a relevant example — Laerd Statistics.* 2018. URL: https://statistics.laerd.com/spss-tutorials/friedman-test-using-spss-statistics.php.

[Nik17] Nikki Gilliland (Econsultancy). *ASOS visual search: Is it any good? — Econsultancy.* 2017. URL: https://econsultancy.com/asos-visual-search-is-it-any-good/.

[Özt21] Ş Öztürk. "Comparison of Pairwise Similarity Distance Methods for Effective Hashing". In: *IOP Conference Series: Materials Science and Engineering* 1099.1 (Mar. 2021), p. 012072. ISSN: 1757-8981. DOI: 10.1088/1757-899x/1099/1/012072. URL: https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012072%20https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012072/meta.

[Pas+20] Nikolaos Passalis et al. "Variance-preserving deep metric learning for content-based image retrieval". In: *Pattern Recognition Letters* 131 (Apr. 2020), pp. 8–14. ISSN: 01678655. DOI: 10.1016/j.patrec.2019.11.041.

[pyi20] pyimagesearch. *10.5 Zernike Moments Computer Vision.* June 2020. URL: https://cvexplained.wordpress.com/2020/07/21/10-5-zernike-moments/.

[SA21] Shalaw Faraj Salih and Alan Anwer Abdulla. "An Improved Content Based Image Retrieval Technique by Exploiting Bi-layer Concept". In: *UHD Journal of Science and Technology* 5.1 (Apr. 2021), p. 1. ISSN: 2521-4209. DOI: 10.21928/uhdjst.v5n1y2021.pp1-12. URL: http://journals.uhd.edu.iq/index.php/uhdjst/article/view/780.

[Sha+17] Devashish Shankar et al. "Deep Learning based Large Scale Visual Recommendation and Search for E-Commerce". In: *arXiv* (Apr. 2017). URL: http://arxiv.org/abs/1703.02344.

[Sik+20] Milad Sikaroudi et al. "Offline versus Online Triplet Mining based on Extreme Distances of Histopathology Patches". In: (July 2020). URL: https://arxiv.org/abs/2007.02200.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: (Mar. 2015). DOI: 10.1109/CVPR.2015.7298682. URL: http://arxiv.org/abs/1503.03832%20http://dx.doi.org/10.1109/CVPR.2015.7298682.

[Soh16] Kihyuk Sohn. *Improved Deep Metric Learning with Multi-class N-pair Loss Objective.* Tech. rep. 2016.

[Son+16] Hyun Oh Song et al. "Deep Metric Learning via Lifted Structured Feature Embedding". In: *Proceedings of the IEEE Computer Society Conference on Computer*

Vision and Pattern Recognition. Vol. 2016-December. IEEE Computer Society, Apr. 2016, pp. 4004–4012. ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.434.

[Tah20] Ahmed Taha. *Retrieval with Deep Learning: A Ranking loss Survey Part Medium*. 2020. URL: https://ahmdtaha.medium.com/retrieval-with-deep-learning-a-ranking-loss-survey-part-1-8e88a6f8e091.

[Tug21] Tugba Sabanoglu. *Retail sales by channel worldwide 2020 — Statista*. 2021. URL: https://www.statista.com/statistics/1095969/retail-sales-by-channel-worldwide/.

[Wan+14] Jiang Wang et al. "Learning Fine-grained Image Similarity with Deep Ranking". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Apr. 2014), pp. 1386–1393. URL: http://arxiv.org/abs/1404.4661.

[Wan+17] Jian Wang et al. "Deep Metric Learning with Angular Loss". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 2612–2620. ISBN: 9781538610329. DOI: 10.1109/ICCV.2017.283.

[Wu+19] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[XSP20] Hong Xuan, Abby Stylianou, and Robert Pless. *Improved Embeddings with Easy Positive Triplet Mining*. Tech. rep. 2020. URL: https://github.com/littleredxh/EasyPositiveHardNegative.

[YCS19] Yao Yang, Haoran Chen, and Junming Shao. *Triplet Enhanced AutoEncoder: Model-free Discriminative Network Embedding*. Tech. rep. 2019. URL: https://github.com/yybeta/TEA.

[Zen+20] Xianxian Zeng et al. "Fine-Grained Image Retrieval via Piecewise Cross Entropy loss". In: *Image and Vision Computing* 93 (Jan. 2020), p. 103820. ISSN: 02628856. DOI: 10.1016/j.imavis.2019.10.006.

[Zha+21] Yanhao Zhang et al. "Visual Search at Alibaba". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 18 (Apr. 2021), pp. 993–1001. DOI: 10.1145/3219819.3219820. URL: http://arxiv.org/abs/2102.04674%20http://dx.doi.org/10.1145/3219819.3219820.

10 Appendix

A Results

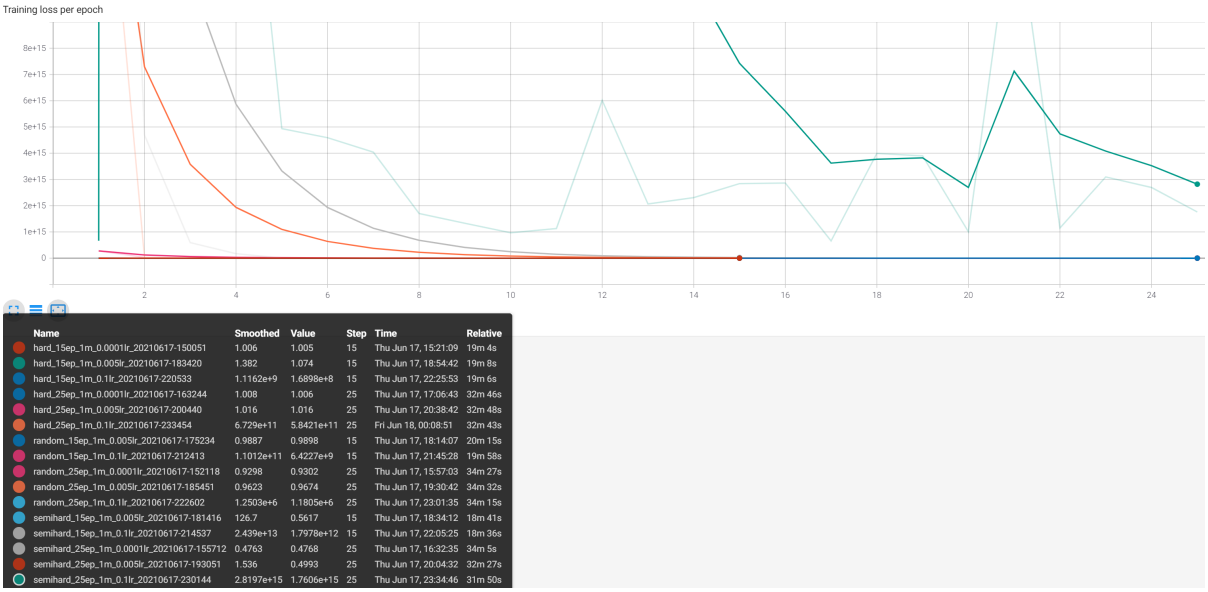
A.1 Gridsearch Results

The table below displays the 20 best gridsearch results from section 6.1.1 in terms of the average validation loss.

Num. Epochs	*Lr*	*Margin*	*Sampling Method*	*Avr. Training Loss*	*Avr. Val Loss*
1	0.005	1	random	2.19	1.48
10	0.005	1	hard	73.9	1.44
15	0.005	1	hard	64.9	1.22
25	0.005	1	hard	6.9	1.04
10	0.0001	1	hard	1.21	1.03
15	0.0001	1	hard	1.09	1.02
1	0.0001	1	hard	3.15	1.02
25	0.0001	1	hard	1.06	1.01
10	0.005	1	random	1.16	0.998
25	0.005	1	random	1.02	0.973
15	0.005	1	random	1.04	0.941
1	0.0001	1	random	0.973	0.932
10	0.0001	1	random	0.965	0.925
15	0.0001	1	random	0.951	0.895
25	0.0001	1	random	0.947	0.881
1	0.0001	1	semihard	0.513	0.497
10	0.0001	1	semihard	0.479	0.476
15	0.0001	1	semihard	0.478	0.475
25	0.0001	1	semihard	0.476	0.472

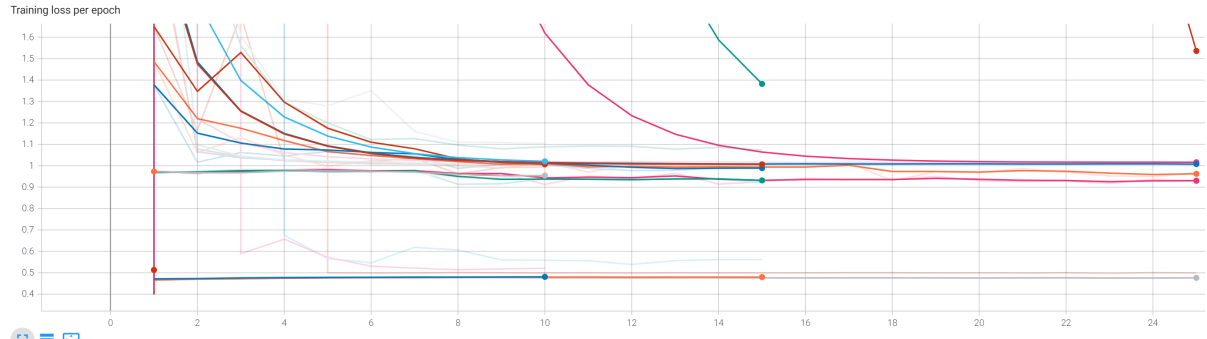
Table 15: Gridsearch results

B Training loss curves for triplet models



Name	Smoothed	Value	Step	Time	Relative
hard_15ep_1m_0.0001lr_20210617-150051	1.006	1.005	15	Thu Jun 17, 15:21:09	19m 4s
hard_15ep_1m_0.005lr_20210617-183420	1.382	1.074	15	Thu Jun 17, 18:54:42	19m 8s
hard_15ep_1m_0.1lr_20210617-220533	1.1162e+9	1.6898e+8	15	Thu Jun 17, 22:25:53	19m 6s
hard_25ep_1m_0.0001lr_20210617-163244	1.008	1.006	25	Thu Jun 17, 17:06:43	32m 46s
hard_25ep_1m_0.005lr_20210617-200440	1.016	1.016	25	Thu Jun 17, 20:38:42	32m 48s
hard_25ep_1m_0.1lr_20210617-233454	6.729e+11	5.8421e+11	25	Fri Jun 18, 00:08:51	32m 43s
random_15ep_1m_0.005lr_20210617-175234	0.9887	0.9898	15	Thu Jun 17, 18:14:07	20m 15s
random_15ep_1m_0.1lr_20210617-212413	1.1012e+11	6.4227e+9	15	Thu Jun 17, 21:45:28	19m 58s
random_25ep_1m_0.0001lr_20210617-152118	0.9298	0.9302	25	Thu Jun 17, 15:57:03	34m 27s
random_25ep_1m_0.005lr_20210617-185451	0.9623	0.9674	25	Thu Jun 17, 19:30:42	34m 32s
random_25ep_1m_0.1lr_20210617-222602	1.2503e+6	1.1805e+6	25	Thu Jun 17, 23:01:35	34m 15s
semihard_15ep_1m_0.005lr_20210617-181416	126.7	0.5617	15	Thu Jun 17, 18:34:12	18m 41s
semihard_15ep_1m_0.1lr_20210617-214537	2.439e+13	1.7978e+12	15	Thu Jun 17, 22:05:25	18m 36s
semihard_25ep_1m_0.0001lr_20210617-155712	0.4763	0.4768	25	Thu Jun 17, 16:32:35	34m 5s
semihard_25ep_1m_0.005lr_20210617-193051	1.536	0.4993	25	Thu Jun 17, 20:04:32	32m 27s
semihard_25ep_1m_0.1lr_20210617-230144	2.8197e+15	1.7606e+15	25	Thu Jun 17, 23:34:46	31m 50s

Figure 36: larger overview of training loss

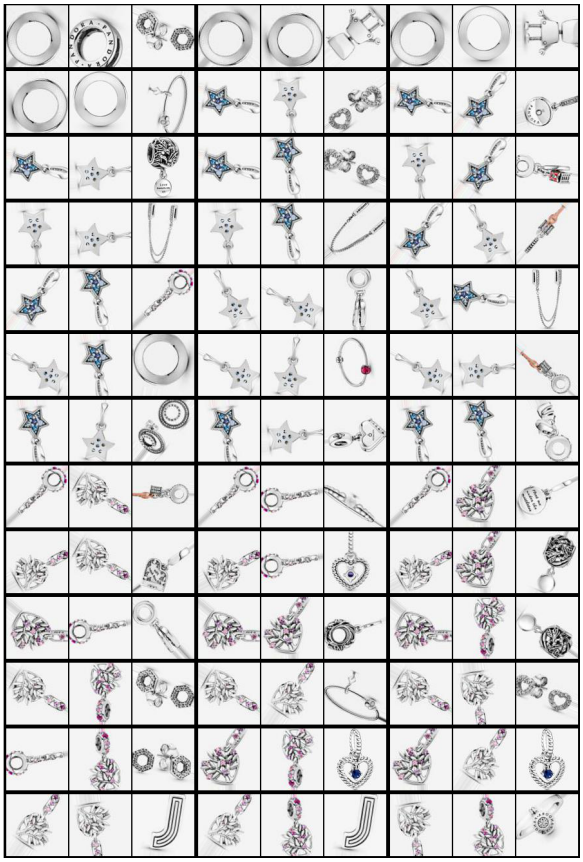


Name	Smoothed	Value	Step	Time	Relative
hard_10ep_1m_0.0001lr_20210617-140419	1.02	1.01	10	Thu Jun 17, 14:17:49	12m 16s
hard_15ep_1m_0.0001lr_20210617-150051	1.006	1.005	15	Thu Jun 17, 15:21:09	19m 4s
hard_15ep_1m_0.005lr_20210617-183420	1.382	1.074	15	Thu Jun 17, 18:54:42	19m 8s
hard_25ep_1m_0.0001lr_20210617-163244	1.008	1.006	25	Thu Jun 17, 17:06:43	32m 46s
hard_25ep_1m_0.005lr_20210617-200440	1.016	1.016	25	Thu Jun 17, 20:38:42	32m 48s
random_10ep_1m_0.0001lr_20210617-133537	0.9539	0.9546	10	Thu Jun 17, 13:49:55	13m 0s
random_10ep_1m_0.005lr_20210617-171101	1.007	0.9937	10	Thu Jun 17, 17:25:22	13m 2s
random_15ep_1m_0.0001lr_20210617-141757	0.9316	0.9208	15	Thu Jun 17, 14:39:22	20m 7s
random_15ep_1m_0.005lr_20210617-175234	0.9887	0.9898	15	Thu Jun 17, 18:14:07	20m 15s
random_1ep_1m_0.0001lr_20210617-133109	0.9731	0.9731	1	Thu Jun 17, 13:32:36	0s
random_25ep_1m_0.0001lr_20210617-152118	0.9298	0.9302	25	Thu Jun 17, 15:57:03	34m 27s
random_25ep_1m_0.005lr_20210617-185451	0.9623	0.9674	25	Thu Jun 17, 19:30:42	34m 32s
semihard_10ep_1m_0.0001lr_20210617-135004	0.481	0.4818	10	Thu Jun 17, 14:04:10	12m 48s
semihard_15ep_1m_0.0001lr_20210617-143931	0.4799	0.4804	15	Thu Jun 17, 15:00:43	19m 53s
semihard_1ep_1m_0.0001lr_20210617-133246	0.5132	0.5132	1	Thu Jun 17, 13:34:06	0s
semihard_25ep_1m_0.0001lr_20210617-155712	0.4763	0.4768	25	Thu Jun 17, 16:32:35	34m 5s

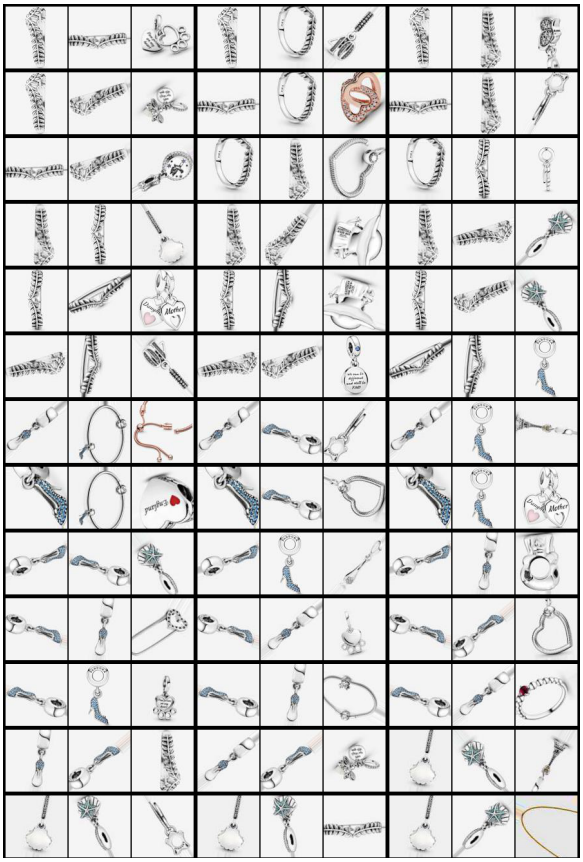
Figure 37: zoomed in overview of training loss

C Triplet pairings for the various sampling methods

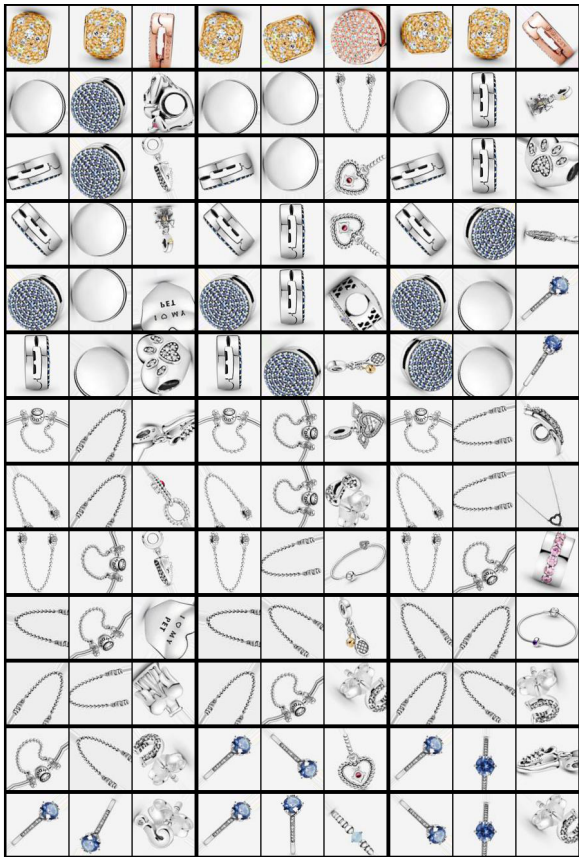
These figures show the pairings such that the each input is a triplet, the images are grouped such that the left image is the anchor image, the middle is the positive and the right is the negative. Here is just shown a subset of the input images.



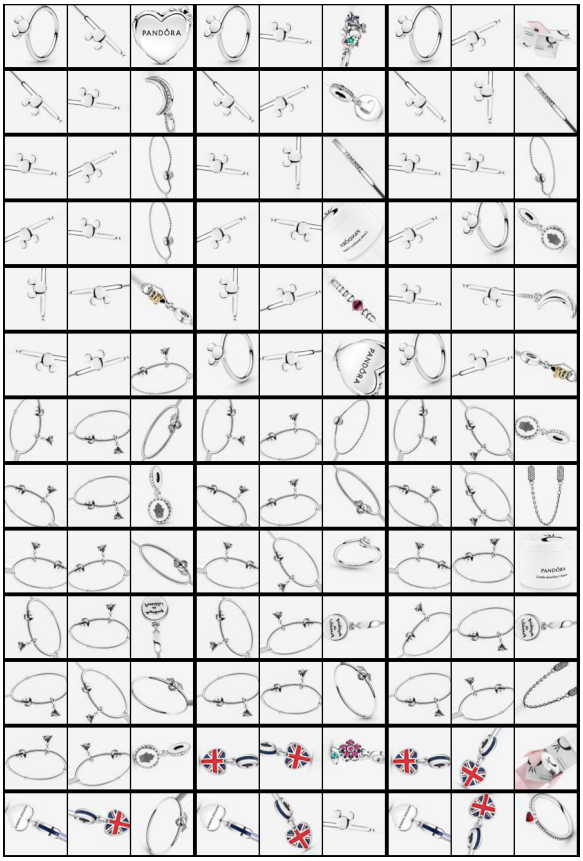
(a) Random 0.1 margin



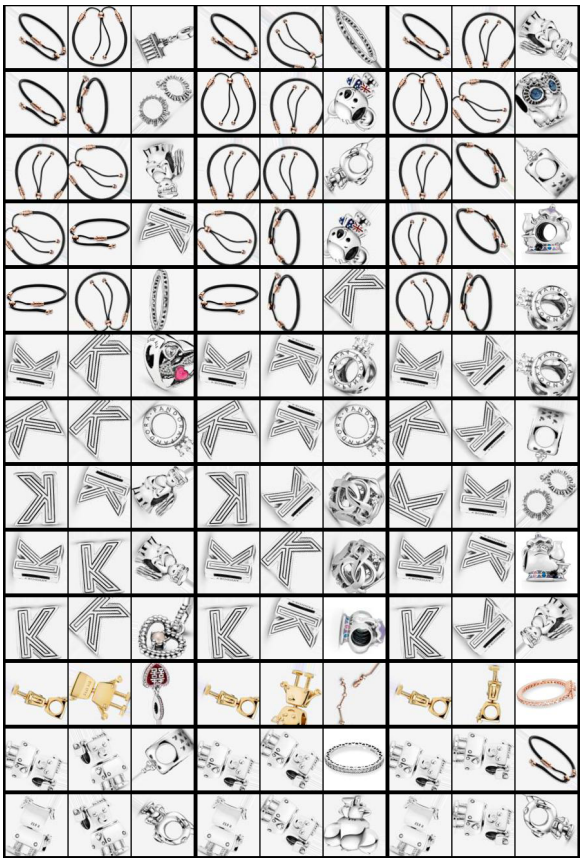
(b) Random 0.5 margin



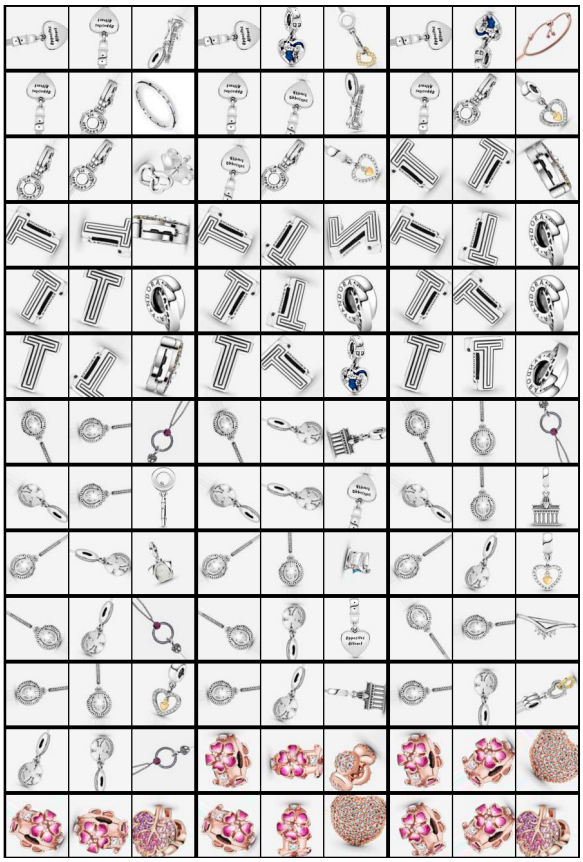
(a) Random 1 margin



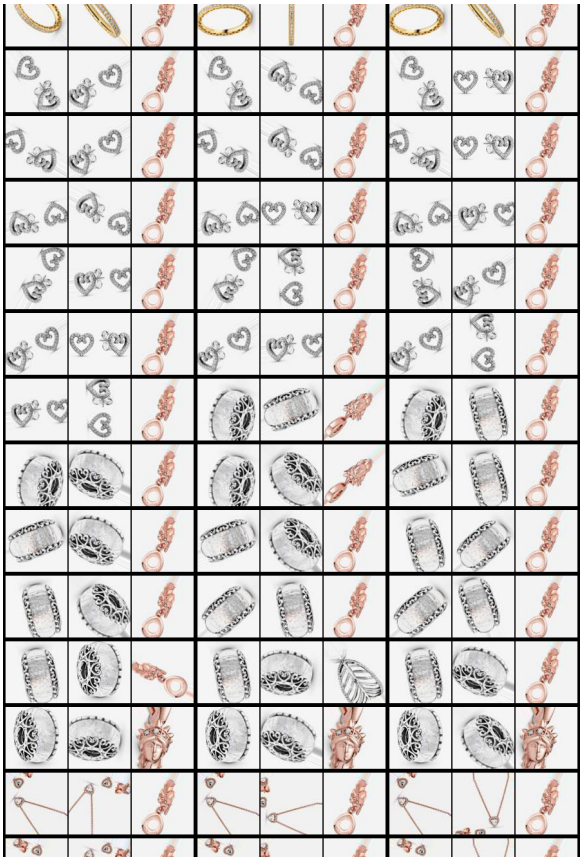
(b) Semihard 0.1 margin



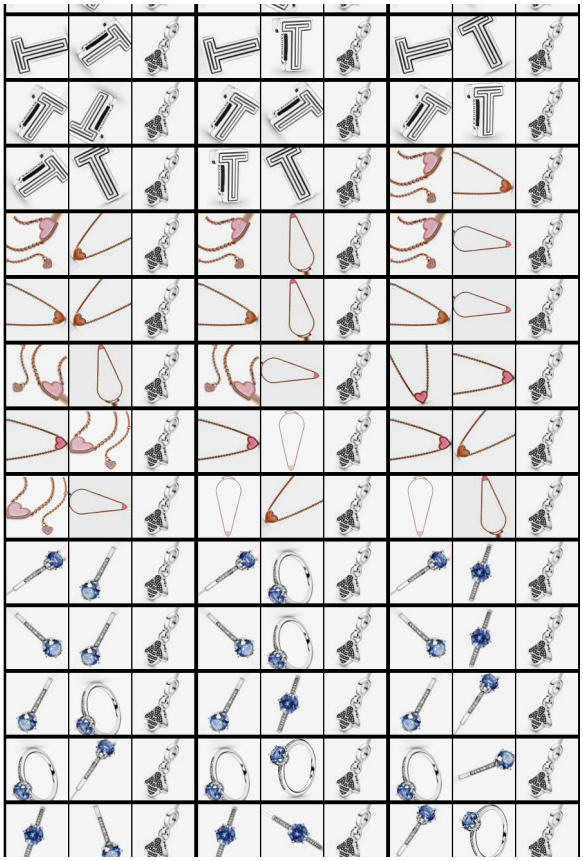
(a) Semihard 0.5 margin



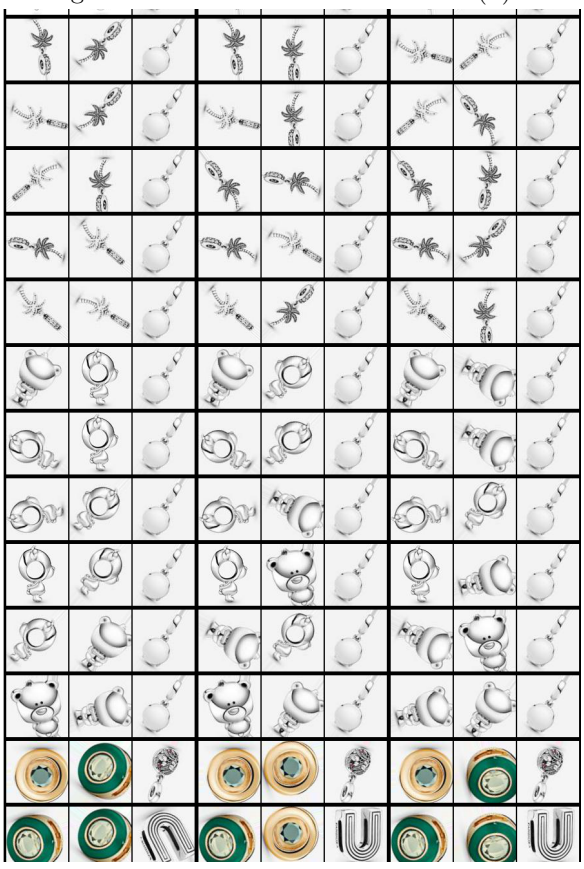
(b) Semihard 1 margin



(a) Hard 0.1 margin



(b) Hard 0.5 margin



(c) Hard 1 margin

D Model Architectures

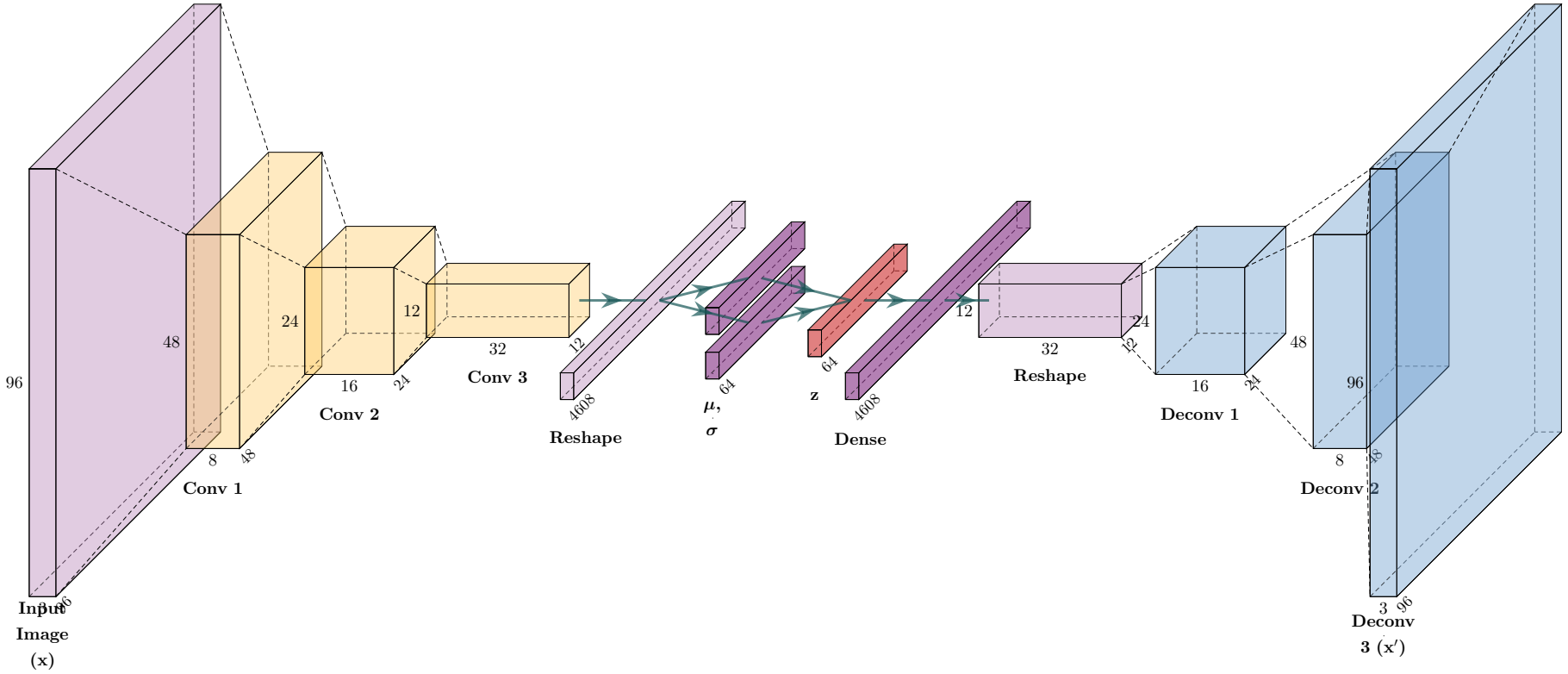


Figure 42: Architecture of the autoencoder.

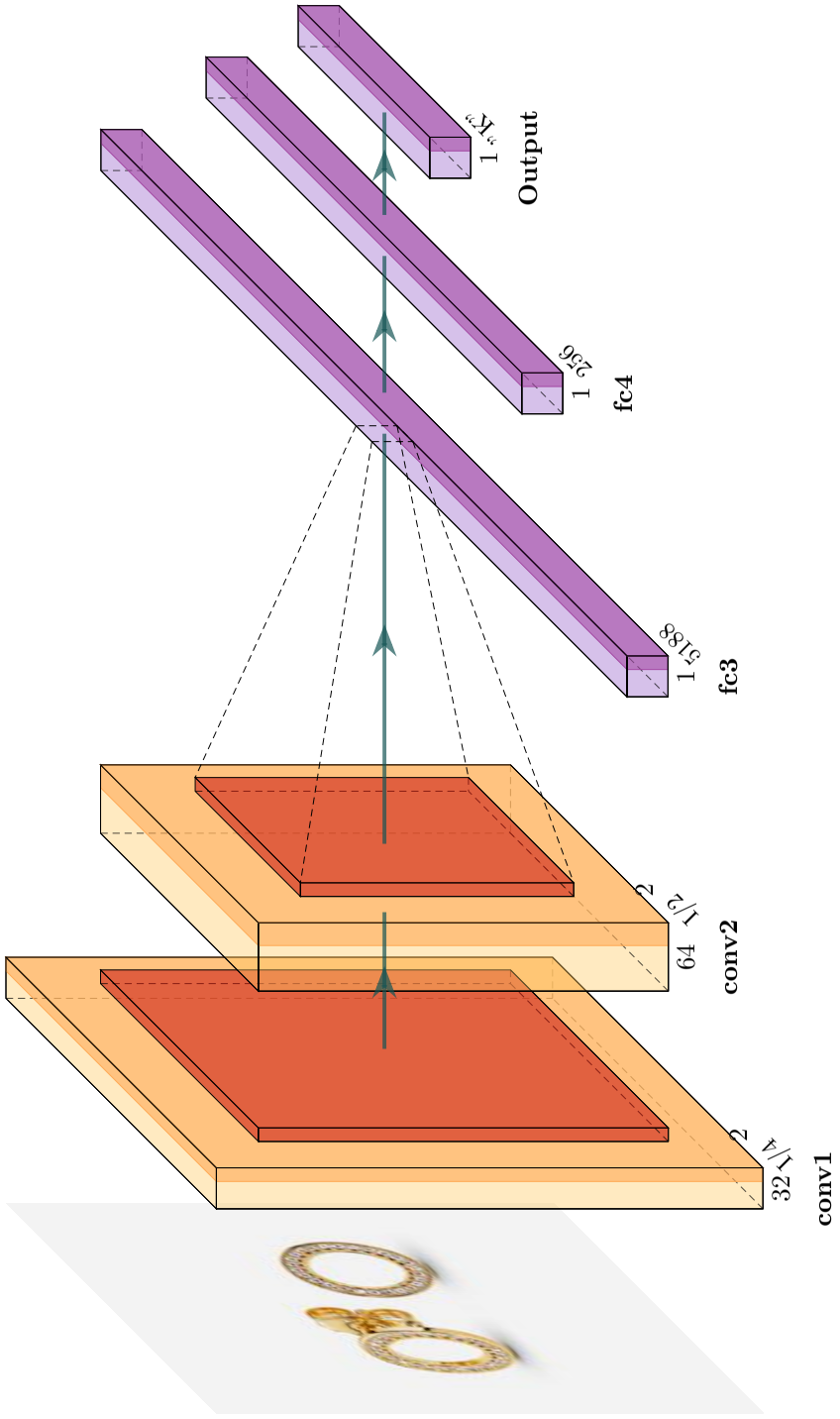


Figure 45: Architecture of the triplet network.

E Survey questions

The images that respondents to our survey were presented with. If the input image was segmented, the result of the segmentation will be placed to the right of the questionnaire image. The first three letters in the string in the upper right denote the order of the models, where a is the VAE, d is the TRN, and r is the random recommendations. For example 'raddradar' means A is the random recommendations, B is the VAE, and C is the TRN.

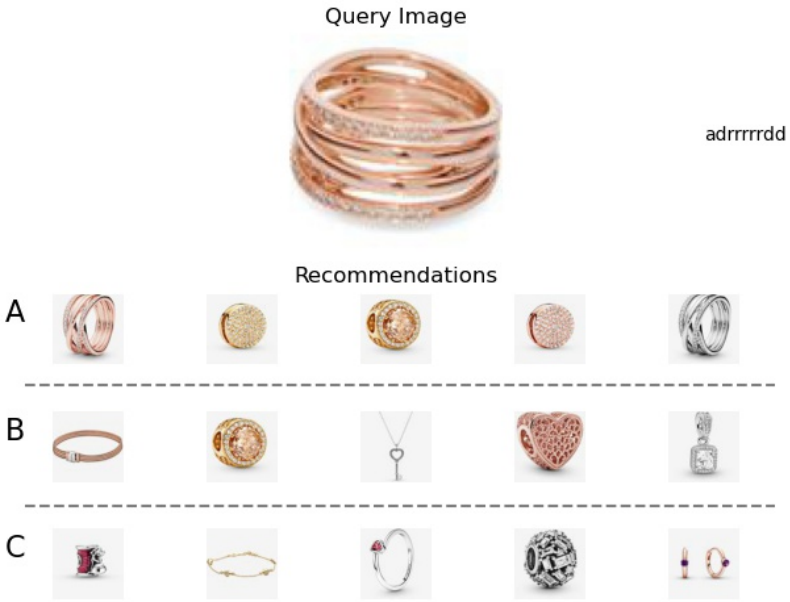


Figure 46: Question 1

Figure 47: Question 2

Figure 48: Question 3

Figure 49: Question 4

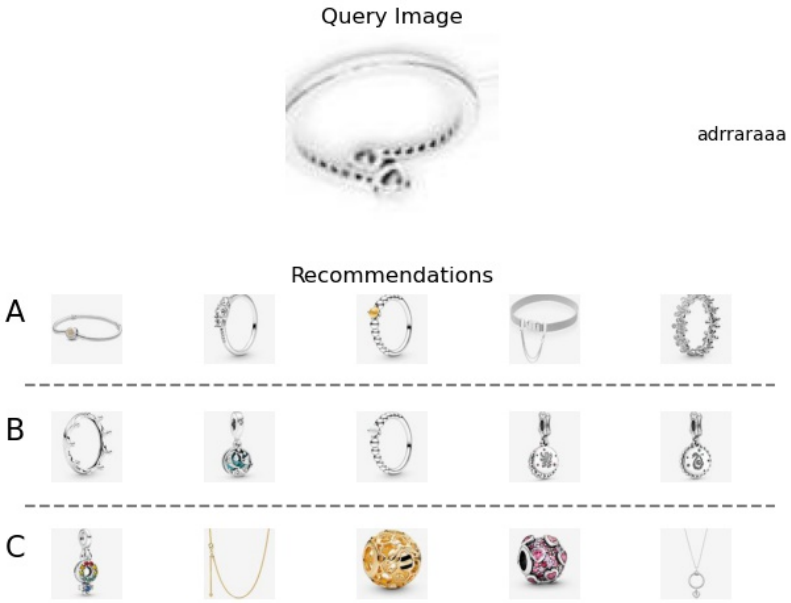


Figure 50: Question 5

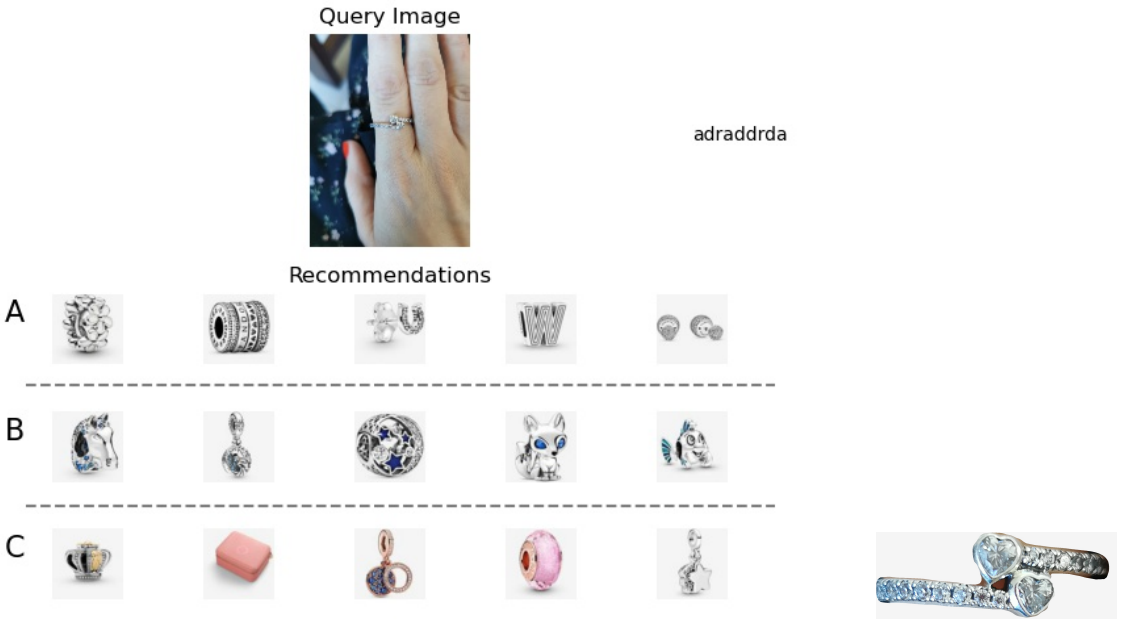


Figure 51: Question 6

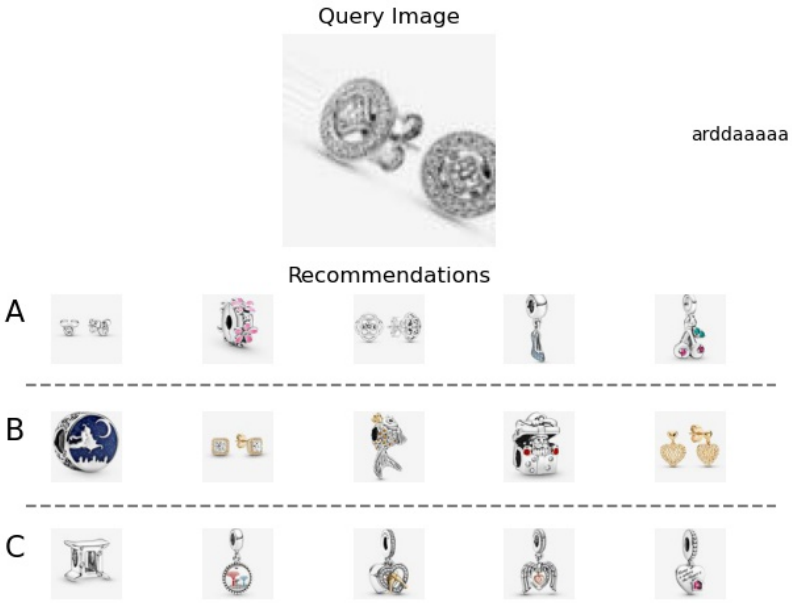


Figure 52: Question 7

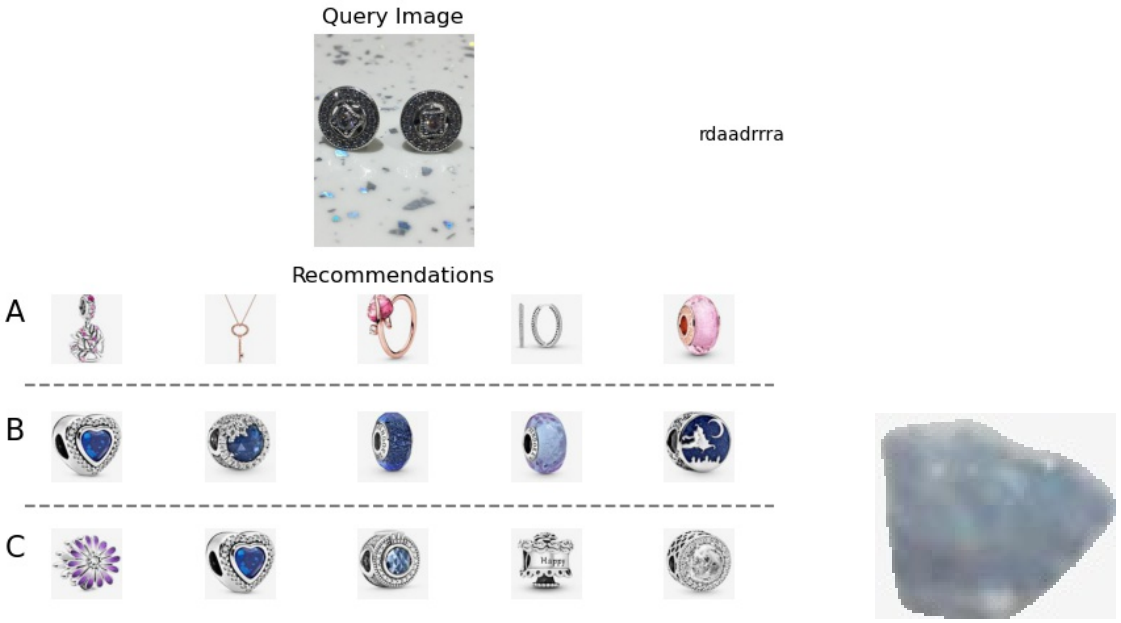


Figure 53: Question 8

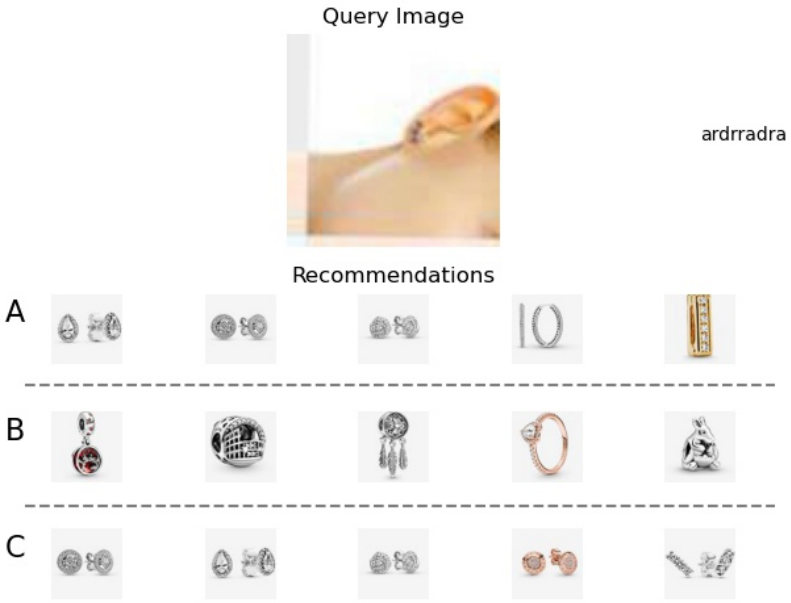


Figure 54: Question 9

Figure 55: Question 10

Figure 56: Question 11

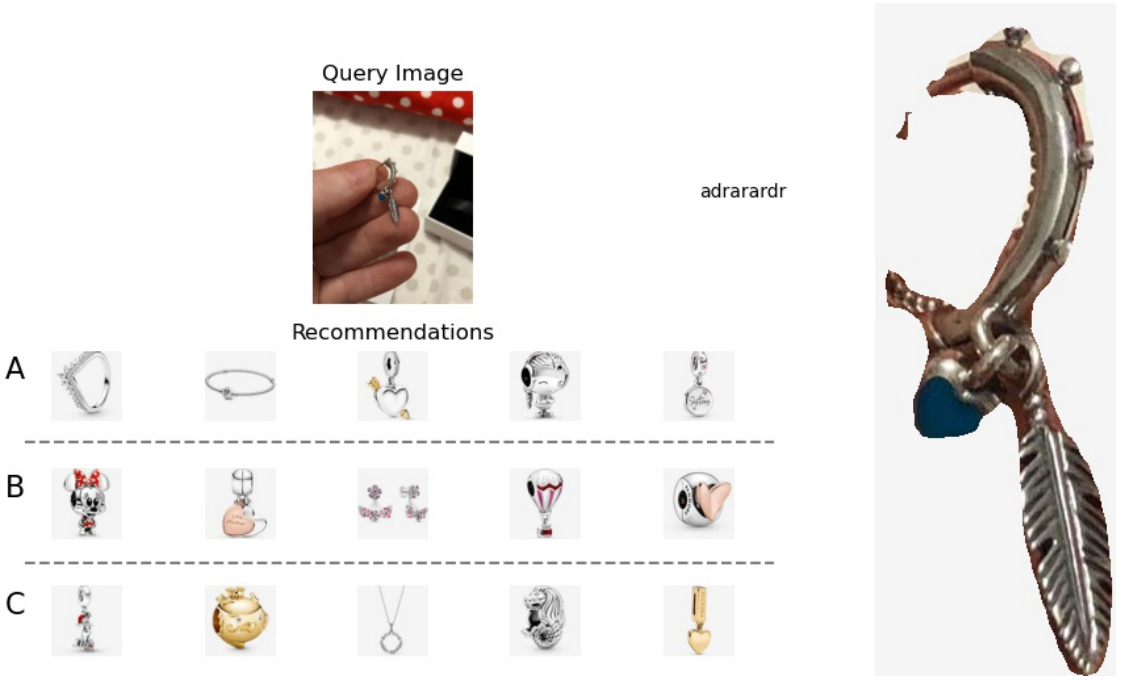


Figure 57: Question 12

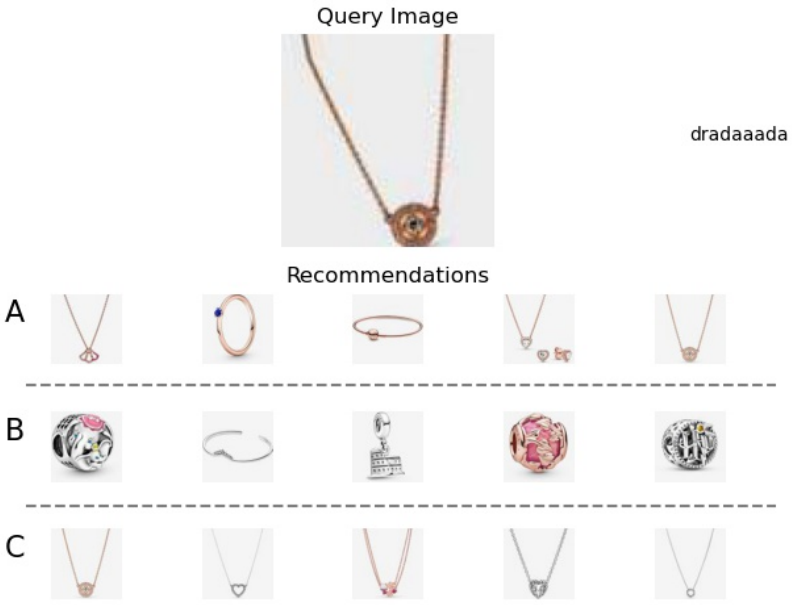


Figure 58: Question 13

Figure 59: Question 14

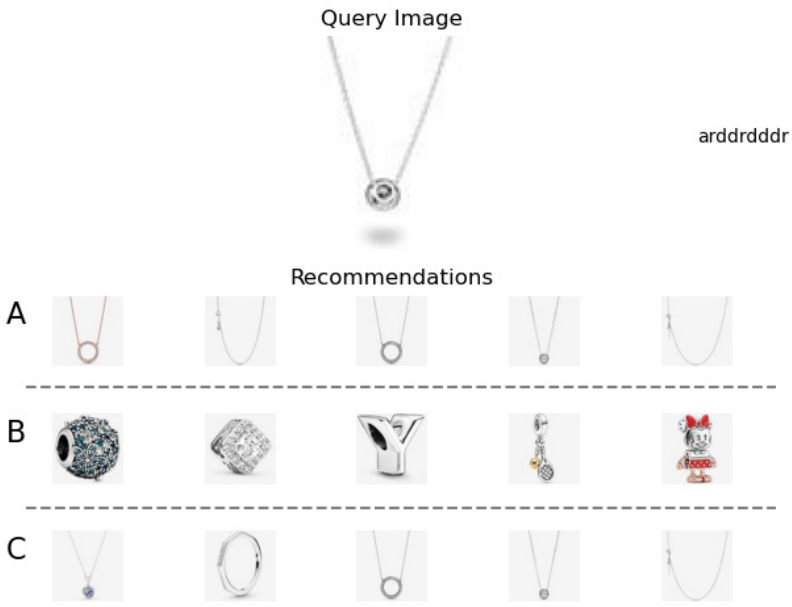


Figure 60: Question 15

Figure 61: Question 16

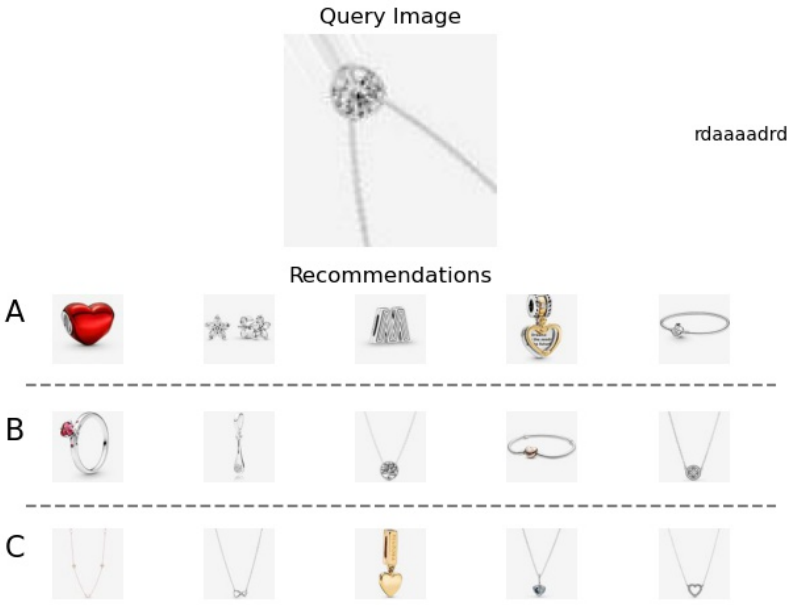


Figure 62: Question 17

Figure 63: Question 18

Figure 64: Question 19

Figure 65: Question 20

Figure 66: Question 21

Figure 67: Question 22

Figure 68: Question 23

Figure 69: Question 24

Figure 70: Question 25

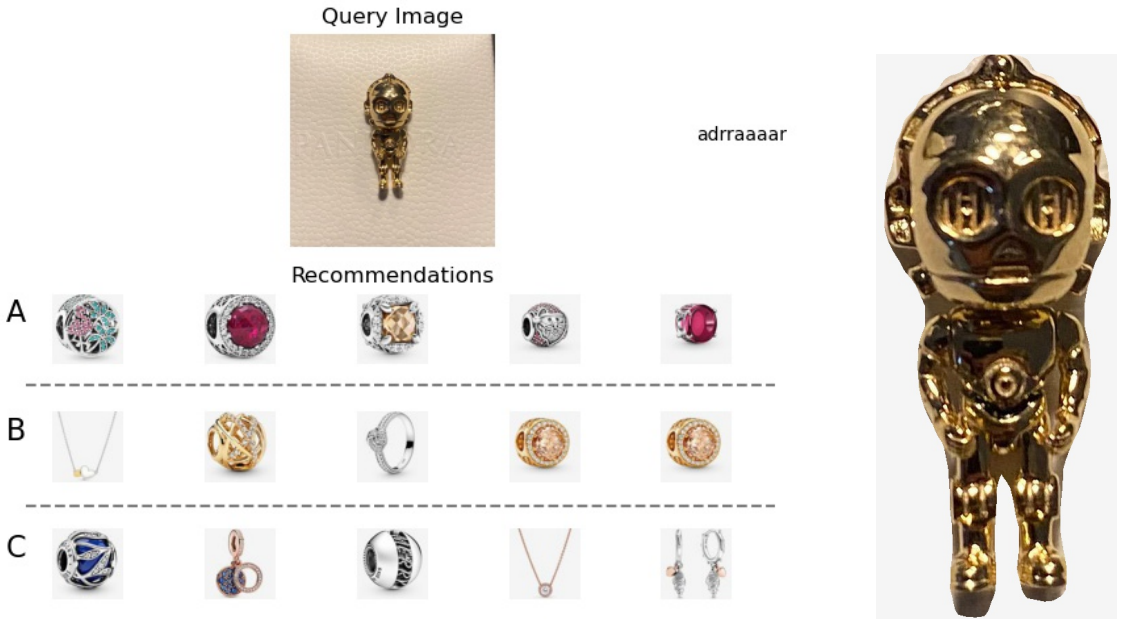


Figure 71: Question 26

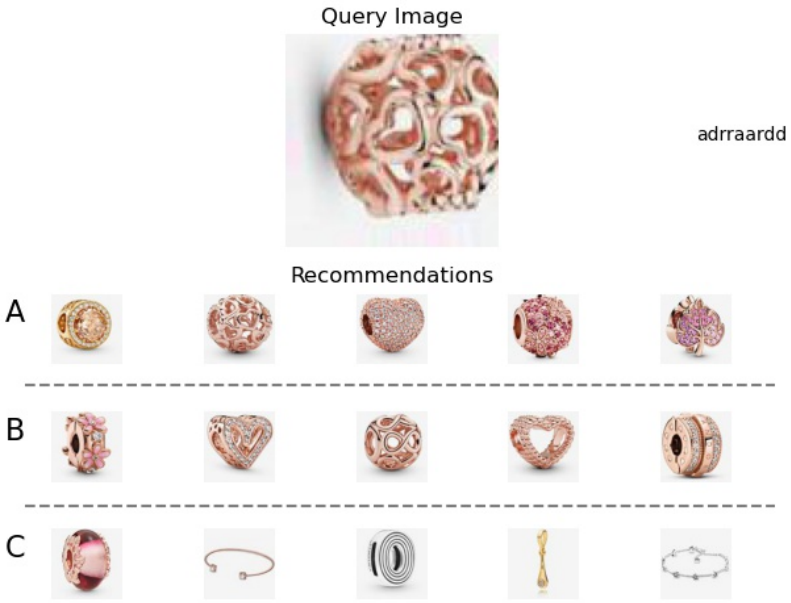


Figure 72: Question 27

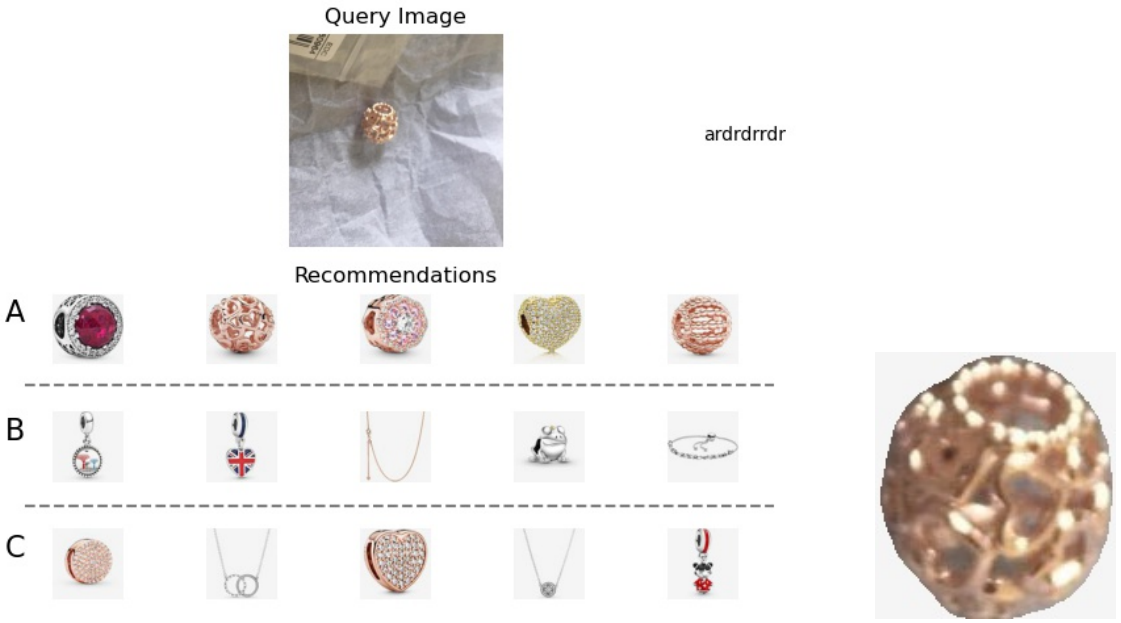


Figure 73: Question 28

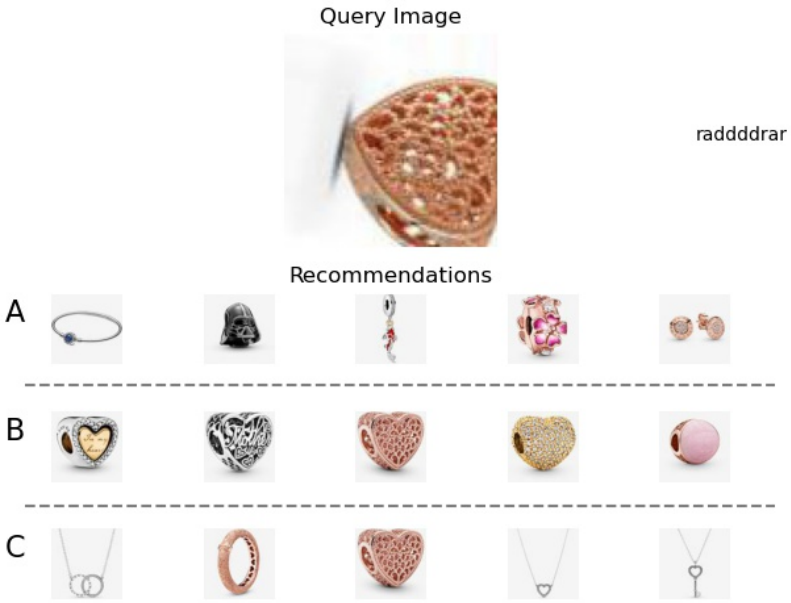


Figure 74: Question 29

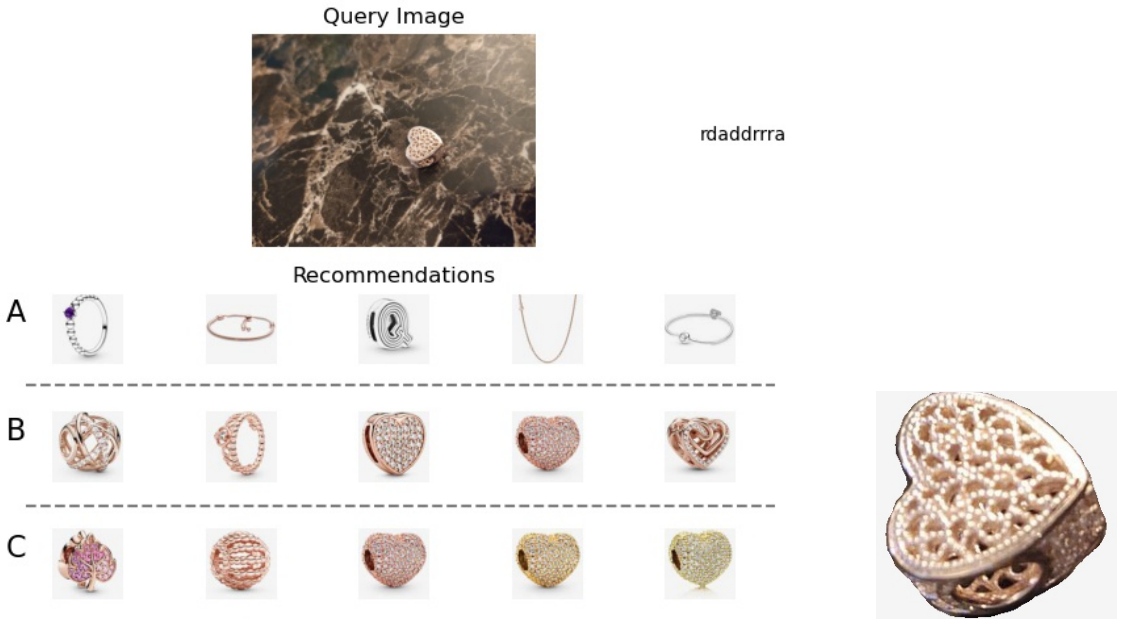


Figure 75: Question 30