eTryOn - Virtual try-ons of garments enabling novel human fashion interactions

Project Title:	**eTryOn - Virtual try-ons of garments enabling novel human fashion interactions**
Contract No:	951908 - eTryOn
Instrument:	Innovation Action
Thematic Priority:	H2020 ICT-55-2020
Start of project:	1 October 2020
Duration:	24 months

Deliverable No: 3.1

Pattern Recognition on Fashion Imagery

Due date of deliverable:	31 May 2021
Actual submission date:	4 June 2021
Version:	Final
Main Authors:	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ), Christos Koutlis (CERTH), Symeon Papadopoulos (CERTH)

Project funded by the European Community under the H2020 Programme for Research and Innovation.

Deliverable title	Pattern Recognition on Fashion Imagery
Deliverable number	3.1
Deliverable version	Final
Contractual date of delivery	31 May 2021
Actual date of delivery	4 June 2021
Deliverable filename	eTryOn_D3.1_final.docx
Type of deliverable	Demonstrator
Dissemination level	PU
Number of pages	64
Work package	WP3
Task(s)	T3.1
Partner responsible	MLZ, CERTH
Author(s)	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ), Christos Koutlis (CERTH), Symeon Papadopoulos (CERTH)
Editor	Elisavet Chatzilari (CERTH)
Reviewer(s)	Thomas De Wilde (QC)

Abstract	This deliverable prototypes a Machine Learning pipeline for the detection of categories and attributes within fashion imagery as well as the plans for the implementation backbone.
Keywords	machine learning, artificial intelligence, object detection, classification, fashion category, fashion attribute

Copyright

Deliverable history

Version	Date	Reason	Revised by
0.1	06/04/2021	Table of Contents	Martina Pugliese (MLZ)
0.2	23/04/2021	First draft	Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ)
0.3	26/04/2021	Comments and suggestions	Christos Koutlis (CERTH)
0.4	29/04/2021	Most content added by both teams	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ)
0.5	12/05/2021	Beta version	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ), Christos Koutlis (CERTH), Symeon Papadopoulos (CERTH)
0.6	26/05/2021	Comments by QC	Thomas De Wilde(QC)
0.7	28/05/2021	Revised version	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ), Christos Koutlis (CERTH), Symeon Papadopoulos (CERTH)
1.0	31/05/2021	Final version	Stefanos Papadopoulos (CERTH), Martina Pugliese (MLZ), Manjunath Sudheer (MLZ), Delphine Rabiller (MLZ), Christos Koutlis (CERTH), Symeon Papadopoulos (CERTH)
1.1	02/06/2021	Final edited version	Elisavet Chatzilari (CERTH)

List of abbreviations and Acronyms

Abbreviation	Meaning
B2C	Business-to-consumer
DF	DeepFashion (dataset)
MLZ	Mallzee
R-CNN	Region Based Convolutional Neural Networks
mAP	Mean average precision
OD	Object detection
CC	Category classification
AD	Attributes detection
VR	Virtual reality
API	Application Programming Interface
TF	TensorFlow
WP	Work package
AR	Average recall
RMSprop	Root Mean Square Propagation
COCO	The Common Objects in Context (dataset)
CLIP	Contrastive Language-Image Pre-Training
NLP	Natural language processing
DoA	Description of Action

● Table of Contents

1. Executive Summary

eTryOn's WP3 focuses on building systems to extract information from fashion imagery and then use these on the large datasets of product ratings MLZ owns to perform trend detections and give recommendations to users. The work package (WP) is machine learning-oriented and will build reusable software that can be utilised as SDK for a multiplicity of purposes. This report tackles the first part of the work, namely the creation of models that automatically extract detailed semantic information from fashion images. After this stage, further models will be developed that will use this information for the detection of trends in the fashion market and the matching of user preferences to products for recommendations.

This deliverable describes the research work carried out by the teams involved in WP3 (MLZ, CERTH), ranging from the generation and design of appropriate datasets to the machine learning methods developed based on them. It is also accompanied with a video showcasing the results of the presented work[1].

[11] https://www.youtube.com/watch?v=Pm_kqbb5jaY&ab_channel=eTryOnProject

2. Introduction

In the realm of e-commerce fashion, imagery is a key component: consumers judge products based on how they look, often taking snap decisions. Many retailers invest large resources in creating good, highly professional photography for their products in order to elicit the best possible sales response.

MLZ offers B2C apps (both mobile and web-based) for consumers to discover, judge and purchase fashion products (clothes, shoes and accessories). It is in possession of a large dataset of consumer opinions on fashion products, dating back several years and has done extensive exploratory work on how the content and the quality of fashion photography influences sales.

The MLZ ratings datasets consist of “positive” and “negative” opinions consumers have expressed on fashion products using various features of the app(s). Users can swipe on items, hence expressing a like or dislike, but they can also visualise them in a grid screen, explore their details, and buy them. All these behavioural hints, often composing funnels (e.g. user searching for products with the search feature, visualising results in a grid and exploring the details of the most promising ones) constitute the main data asset MLZ owns and uses to perform market research and predictions. The dataset is unique in its volume and historical aspect, as well as the fact that it has been gathered in a genuine way: users are not paid to express opinions and the apps are freely available to anyone, hence data comes from large and differentiated sectors of the population without incentive-based mechanisms.

Given the variety of product offers in fashion, many factors can impact consumers’ reactions and the eventual performance of items in the market. Items vary in category, colour, shape, patterns and general appearance, as well as style-independent factors such as price and brand. Furthermore, the way an item is presented may non-trivially affect consumer perception. For instance, the use of (human) models to display fit and general look can provide visual information that a flat-lay photography cannot convey, and this may matter more for certain categories than others (e.g., those which are more fit-critical, such as bottoms and dresses). The background against which the photo is taken can also have an impact. An important element to consider is whether the product is shown on its own or in conjunction with others (a model displaying it as part of a full outfit).

For all these reasons, it is critical to base prediction and recommendation engines on information extracted directly from the fashion imagery and not just the metadata and text accompanying items. As a matter of fact, the text retailers use to describe their items can often be incomplete or lack important details, if not outright uninformative and, as such, it cannot be relied upon for such a task.

This deliverable is aimed at building a system capable of extracting detailed information from retail images, outlining all the content in the picture to the best possible accuracy, and of automatically tagging content with clothing category and a detailed set of attributes. This system is structured in a pipeline of hierarchical information extraction (see section 4), where each step is run on the output of the previous one but is also usable independently.

Due to the complexity of the task, these systems normally use deep neural networks, an area of machine learning which in recent times has proved its effectiveness for many tasks (for example: object detection, clothing attribute predictions, fashion style predictions and recommendations) and is nowadays being increasingly adopted in many areas.

The development of this system has many potential applications, within the scope of the eTryOn project, but also more broadly outside of it. While software that performs information extraction from images is not a new field of work, the use of it within a

specialised domain, such as fashion, is at the forefront of innovation. Several fashion companies are researching this space for use to their own advantage [1, 2]. Challenges in tailoring the use of machine learning for a specific domain lie in both the creation of adequate datasets for the training of models and in the choice and fine-tuning of appropriate models. Both these components require research and development work due to the lack of commercially available data regarding the fashion domain. Our work aims at taking part in the innovation in these areas, contributing to the scientific community as well as creating software for the stakeholders in the project that has the potential to also be offered commercially to external clients.

For our research, we have utilised existing (public) datasets as well as datasets we have built ourselves starting from imagery, metadata and textual information owned by MLZ. These specialised training datasets, created for the purposes of this project, constitute per se new Intellectual Property the company can use for a variety of reasons and we will describe the methodologies we used to create them in section 3.

Deliverable D3.1 constitutes research and development work that will function as the technical basis and prerequisite for deliverables D3.2 (fashion trend detection) and D3.3 (recommendation engine to users). The models developed in D3.1 will be applied on the large dataset of images MLZ owns to extract information. The coupling of this information with the MLZ dataset of users' opinions on fashion products, will provide the basis for the other deliverables. On the MLZ apps, users interact with products primarily by evaluating their images, hence being able to know what exactly is present in an image the user has interacted with is essential to be able to interpret the ratings and use them to train prediction models.

The models developed in this deliverable could benefit from improvements and expansions we will describe in the text, but is considered quite satisfactory as a basis for the subsequent deliverables: subsection 4.4 summarizes the performance achieved.

2.1 Applications within the fashion domain

This subsection describes the many potential applications of our work, both within the scope of the eTryOn project and outside of it.

2.1.1 Applications within eTryOn

eTryOn aims at developing three applications focusing on virtual try-ons of clothing. All these applications will require input from WP3, namely:

- for the VR designer app, whose users will be fashion designers interested in developing new designs, WP3 will provide information about how popular a certain design is in the market,
- for the social (influencers) app and for the ecommerce app, WP3 will provide a score of how "recommended" a product is for the user, and possibly, also how popular it is in the market (as for the designers app)

This deliverable will provide the prerequisite step to train models to understand clothing details from a fashion image, which will be furnished at API endpoints as part of the implementation of said eTryOn applications

2.1.2 Applications for the fashion industry

The work developed in this deliverable has the potential to be applied outside of the framework of the eTryOn project.

The datasets created as part of this deliverable may be per se offered as a commercial component to train models upon. Additionally, a system that extracts information from fashion imagery can be packaged as its own offering for fashion companies willing to tag their photography with attributes.

Such a system could be used for a multiplicity of uses, such, for example:

- a catalogue search feature based on the imagery (visual search);
- a mechanism to perform product similarity based on the images, that can be used for retrieval of similar products to the one at hand;
- mechanisms to assess the quality and completeness of e-commerce photography (e.g., measuring how often t-shirts are shown in flat-lay vs. in full outfit - alongside which other items - and recommending the best option).

2.2 Structure of the report

The teams involved in this work (MLZ, CERTH) have devoted effort to building the large training datasets of machine learning models, and to the research and realisation of said models. Section 3 will outline the work performed in the creation and quality-check of novel datasets, while also noting the inherent difficulties that arose in the process.

Section 4 will summarise all results achieved in the machine learning modelling phase, highlighting strengths and weaknesses of the models developed and planned strategies for improvements.

Section 5 will discuss the implementation of this research, section 6 outlines the authors' conclusions and section 7 reports the scientific references used in this work.

3. Datasets gathering and evaluation

In order to train models, the problem of gathering appropriate datasets has been tackled. This section will describe the work performed in order to obtain quality datasets, which has occupied a fair amount of time and effort. The datasets created so far are not comprehensive of everything in fashion, but a structured procedure to extend them to missing categories and details is in place and can be utilised.

3.1 The MLZ taxonomy of clothes

The world of e-commerce fashion is very prone to ambiguity in the way clothes are described, and there are no standards for the categorisation of items into groups. Different retailers may not only give different names to the same item, but, more importantly, categorise the same kinds of items under different groups. An obvious example appears with women's footwear, where summer heeled shoes can be sometimes placed under “heels” and sometimes under “sandals”, but the issue remains for any kind of category and item.

MLZ worked around the problem by creating its own “fashion taxonomy”, where categories of clothing items are hierarchically structured and dependencies of attributes are clarified. Said taxonomy has been created by a mixture of domain knowledge in fashion and a statistical analysis of the market data (product information gathered from brands’ websites) the company owns.

In Figure 3-1, we furnish an example structure from the MLZ taxonomy, to illustrate the concept. The example is for upper-body (tops) clothing and shows the hierarchical structure in the taxonomy, with categories at the top of the graph, types (or sub-categories) at the following level and styles following**.**

Figure 3-1: Example of the MLZ taxonomy for upper-body clothes.

The MLZ taxonomy has been the standard used during the work carried out in this deliverable, in the sense that categories and attributes in the training sets have been derived using the hierarchical structures outlined in the taxonomy itself.

3.2 Datasets research and creation

The pipeline model has been designed to work as a succession of stages:

1. a first stage (*object detection*) will detect objects within the image: upper-body, lower-body, full-body, footwear
2. a second stage (*category classification*) will classify each object individually as belonging to a certain clothing category
3. a third stage (*attributes tagging*) will tag the object with finer-grained information from clothing attributes (e.g. the pattern and the style of the garment)

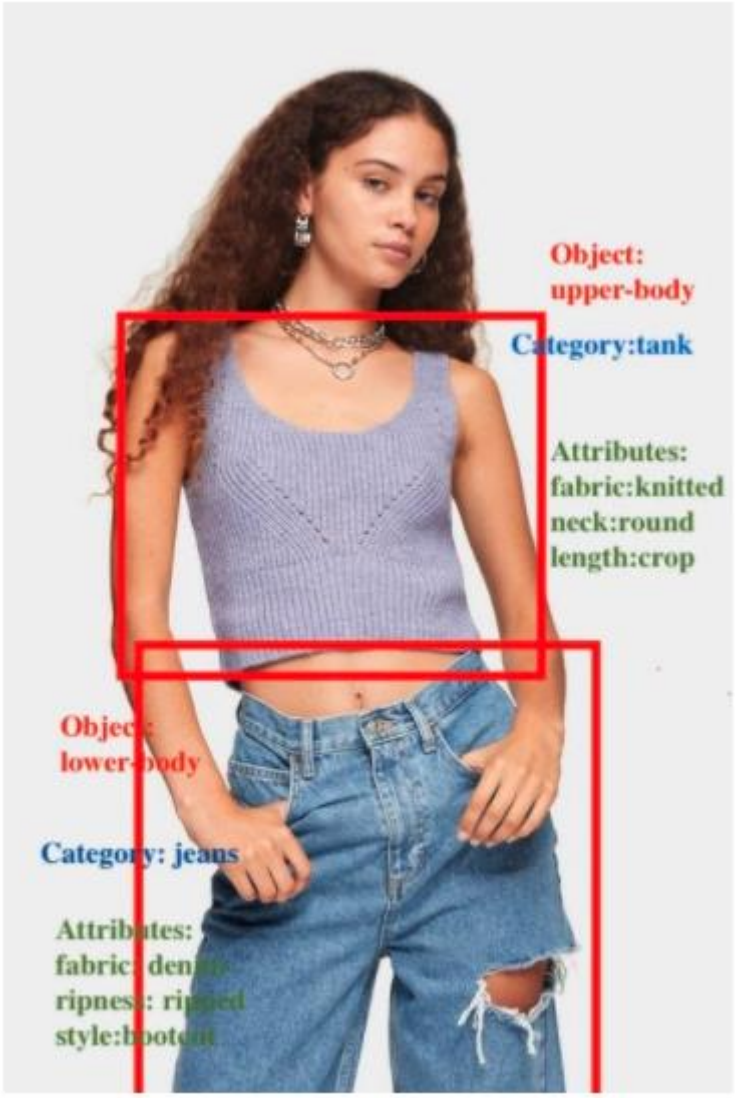

Figure 3-2: The various stages of the model pipeline.

Each of these stages, illustrated in Figure 3-2, has required a specific dataset.

We have conducted research with regards to what datasets are already available. There are a few that companies and research institutions have created while performing similar lines of research. They are normally free to use but under a licence that does not encompass commercial exploitation. One such dataset is the first and second version of DeepFashion [3, 4] (it is also the most well known comprehensive dataset built for fashion clothes classification and tagging as of now). We have used both at various stages of our work for research purposes. Other existing datasets have been deemed less interesting or too incomplete. Table 3-1 summarises the public datasets that we researched and explored for this work, the purpose of using these datasets has been purely exploratory and to draft comparisons with the results obtained with MLZ datasets. Note that in order to make use of datasets other than MLZ ones, we had to remap categories and attributes to the MLZ taxonomy, because of the non-universality and lack of standards.

Table 3-1: Public datasets used for exploratory and comparison work in our research.

Dataset	Description	Positives	Negatives	How we used it
DeepFashion, first version [3]	~800,000 images labelled for 50 categories, 1,000 attributes, bounding boxes	large set and large variety, presence of bounding boxes	lacks footwear, some attributes are actually categories (e.g., “shirt”, “cami”), bounding boxes are only for one of the objects at a time	Training the detection of objects and the classification of categories
DeepFashion, second version [4]	~490,000 images labelled for 13 categories, bounding boxes and more	large variety and detailed information, bounding boxes are annotated for all objects in a picture	lacks footwear, categories used are difficult to map to MLZ ones	Training the detection of objects
iMaterialist [5]	~1 million images tagged for fashion attributes	large set and large variety, does not have explicit licence constraints	large number of wrong labels	No easy mapping to MLZ categories and large error rates, so did not use
Zappos [6, 7]	~50,000 images, only footwear	large set of footwear images	only flat-lay images and in the same pose	Used as footwear dataset for object detection (using other sets for non-footwear), but because of the specific pose the results were not good enough

On top of these, we have considered some more datasets for potential exploratory use:

- FashionAI [8]: we could not get access to the dataset after requesting it, this would have potentially been a useful dataset due to the hierarchical categorization of garments;

- YFCC100 [9]: very general dataset not related to any specific domain, that would create noisy data when filtered;
- Fashionista [10]: rich dataset that contains very detailed information including polygons around clothes, bounding boxes could be reconstructed with the code provided but this was not deemed worth doing for exploration reasons;
- ModaNet [11]: dataset of polygonal annotations around clothes, bounding boxes could be reconstructed but this was not deemed worth doing for exploration reasons.

Because of the licencing limitations of existing public datasets as well as the fact that MLZ is interested in having its own training sets as intellectual property in order to be able to commercialize the results of this work, we have decided to also create training datasets based on MLZ products. Furthermore, the subsequent deliverables will use MLZ rating-image pairs, and the information extraction system from this deliverable will be run on MLZ imagery. Hence, for the consistency of datasets it is better to train models on MLZ datasets too. Also as described in the DoA, WP3 focuses on the use of datasets owned by MLZ, so we have worked specifically using those. Other datasets have been explored for comparison and investigation purposes.

Note that the main focus of our research has been on clothes, so we have left out the category of accessories. The latter, would have posed specific challenges to both the data gathering phase and the modelling one, due to the enormous difference in shapes.

3.2.1 Object detection phase

The object detection stage is rather time-consuming in terms of creating a reliable dataset from scratch. That is because the training images need to be not only tagged for their content but also annotated with the bounding boxes around said content. However, in our case it proved to be the most effective way to generate good datasets. In the following, we will explain our procedure and experiments.

3.2.1.1 Manually annotated images (by MLZ and CERTH)

MLZ had worked in the past on this topic and it suggested to participants in this deliverable to split the work of manual image annotation with bounding boxes. For the task, we chose the *labelImg* tool [35] (open-source), due to its ease of use and intuitive interface. The tool allows a user to quickly draw a bounding box around the desired object in an image and to save the data.

The images to annotate have been retrieved from the MLZ databases by querying for them under category constraints (e.g., avoiding accessories) and randomising sets to ensure that no parameter is over-represented (e.g. one brand, which might negatively impact model performance because they might privilege images/poses of a certain kind). Products in the MLZ database indeed contain, amongst other things, tags for their category and their brand. The category is extracted from the name using NLP modeling which however suffers from mistakes due to uninformative names and general model performance. We ensured that our data pulls excluded any unwanted categories (for example: swimwears, accessories) and mitigated the overwhelming presence of a few brands by introducing

Of course, in doing the annotations we had made sure to have a balance of the four types of objects. We gave ourselves simple but strict rules about how to perform the annotations:

- annotating objects only when fully present in the picture, and leaving those where the garment is not fully visible (e.g. when an image in upper-body also shows part of the trousers or when shoes are partly covered)
- annotate coats and jacket as upper-body objects

Figure 3-3 illustrates a couple of examples of images manually annotated by the teams.

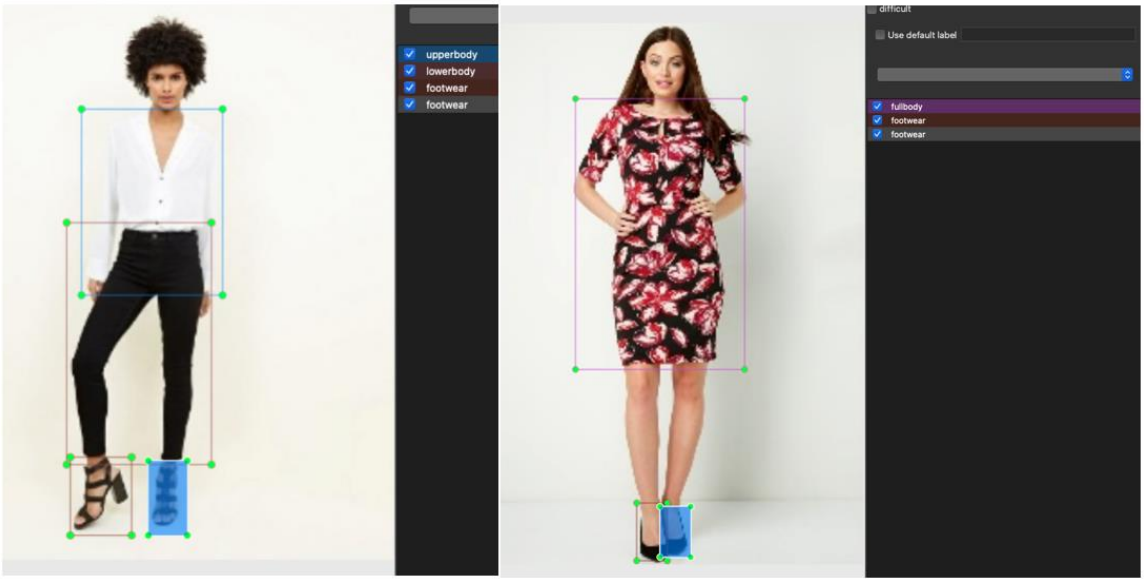

Figure 3-3: Examples of manually annotated images for object bounding boxes.

As specified, MLZ images like any collection of fashion images can present the products both in a flat-lay manner (only the product is present) or with a person modelling it (potentially other products are present too). For this reason, we decided to annotate a set which is balanced in the presence of flat-lay and non flat-lay ones.

The flat-lay images can be quickly annotated for the bounding box by using a simple computer vision mechanism, so they do not require human annotation. The mechanism we used is based on edge detection (using the Canny edge detector [36]) and the drawing of a box around the object by using the edges to recognise the pixels in the extremes. An example is shown below. Note that flat-lay images are particularly common for footwear. Figure 3-4 shows an example of a footwear image automatically annotated for the bounding box using this method.

Figure 3-4: Example of a (footwear) image annotated for the bounding box using the Canny edge detector.

In total, MLZ and CERTH have manually annotated a set of ~3,400 images, to which we added 906 more images for some specific types of garments which appeared to suffer the most from misdetections.

3.2.1.2 Images annotated via AWS Sagemaker Groundtruth

On top of the manually annotated images by MLZ and CERTH, to obtain more data we made use of the AWS Sagemaker Groundtruth [37] service. The latter allows crowdsourcing the annotation of image datasets by paying workers to perform tasks. It is a service AWS built on top of their Mechanical Turk offering and to the authors' knowledge, it is the best available in the market to quickly obtain annotated images for machine learning purposes.

Groundtruth offers an easy-to-use interface whereby one can upload a set of images and give written instructions to workers as to what to annotate them for and what rules to follow. We provided the same rules we gave ourselves as per the above. Workers will then annotate them, but there is no guarantee that they will respect the rules given. When multiple workers annotate the same picture, which means the bounding boxes may slightly differ in location, AWS provides the end annotation alongside a "confidence score" for the bounding box. The confidence score is calculated by consolidating the label results, if more than one worker annotates a single task. AWS Ground Truth calculates this score which ranges between 0 and 1 to indicate how confident the ground truth is in the label. It was advised to not to interpret the values of the confidence score as an absolute value and not to compare the confidence scores of human-labeled data objects and auto-labeled data objects. For example, if all of the confidence scores are between 0.98 and 0.998, we should only compare the data objects with each other and not rely on the high confidence scores.

The confidence scores for humans are calculated using the annotation consolidation function for the task, while the confidence scores for automated labeling are calculated using a model that incorporates object features. The two models generally have different scales and average confidence. It is worth remarking that the use of AWS Groundtruth

comes at a cost, so from the MLZ point of view it was important to assess the cost-quality balance in order to decide how to best use it for the task.

We asked workers on Groundtruth to annotate circa 2,000 images, and after retrieving and analysing the results, we have removed all the annotations which appeared to have a confidence score smaller than 60%. This choice was motivated by the fact that this value appeared to be the median of the distribution of scores, which meant that a fair amount of the scores were too low to be usable.

Unfortunately however, after experimenting with running the model with the images from AWS added into our set, we did not obtain improved results. This was also due to the fact that in the procedure of removing objects with a low confidence score there is a risk to remove just one of two footwear items (which are most often present in couples). Due to this and the fact that this image-gathering is expensive, and based on the quality of results we obtained with our annotated images only, we decided to not use the AWS Groundtruth service anymore but to rely on our own annotations only (which was feasible for four objects and a relatively small set). We leave the possibility to use the service in the future open, but our results suggest that in order to have quality data one needs to ask for many workers per image, hence lowering the overall confidence score, at a higher cost.

3.2.1.3 Data augmentation and final count of annotated objects

Data augmentation is a technique that significantly increases the diversity of the training image dataset without actually collecting new images. In our case, the augmentation for object detection training images were performed with the help of *imgaug* [38] python library, which supports a wide range of image augmentation techniques. This includes horizontal and vertical flips, rotation by a specified angle, shifting, padding and many other operations. The augmentations ensure that bounding boxes are transformed accordingly alongside the image.

Table 3-2 shows the count of annotated objects alongside the number of augmented images for each object and image type (flat-lay ones and non-flat-lay ones).

Table 3-2: Number of annotated and augmented objects based on object type and image type.

Type	Object	Count of annotated objects	Count of objects from augmented images	Total objects
Flat-lay images	upper-body	2,134	365	2,499
	lower-body	1,935	565	2,500
	full-body	1,008	1,510	2,518
	footwear	6,788	130	6,918
Non-flat-lay	upper-body	2,663	N/A	2,663

(presence of human models)	lower-body	1,248	489	1,737
	full-body	1,149	1,365	2,514
	footwear	2,565	339	2,904

In our case, several experiments were performed based on not only the object type (upper-body, lower-body, full-body footwear) but also on whether the given image had a human model or whether it was a flat-lay one. Based on these experiments, the augmentation operations we decided to use are:

- horizontal flip + vertical flip + rotation for manually annotated images
- vertical flip + rotation for flat-lay ones

The rotation uses an angle randomly chosen from the list of (45, 90, 180) degrees. Note that flat-lay images were not flipped horizontally because that would have not created different images (a t-shirt or trouser in flat-lay and flipped horizontally would only appear different if there are slight asymmetries in the two sides, something rare and not relevant) so it would have just added redundancy.

Augmentation was also used as a solution to better balance the number of training objects per object type. Due to reasons inherent to fashion datasets, certain objects are naturally more populated than others. The column of annotated objects in table 4-2, for non flat-lay images, shows that despite all the best effort in balancing the set upfront, upper-body objects are sensibly more frequent than others; this is because the vast majority of images with human models that display upper-body clothes show the model in the upper side only, so no other garments are present, and this is not the case with other objects.

We also chose these specific operations after investigating the recent literature about image augmentation in the case of object detection tasks [12].

3.2.1.4 Gold standard dataset of images

Apart from images we had to create for training purposes, it was essential for us to also create a set of novel images for the evaluation of the trained models, which is termed as a gold standard dataset. This is created, in order to ensure results are unbiased and correct, and also in order to best diagnose what the model outputs in terms of boxes for each object, we want to check the actual performance of the model on data that was not part of the data used to train the models. For the evaluation, new images that were not present as part of the training sets were retrieved from the MLZ database. This set was ensured to have the highest quality by thoroughly vetting for the presence of human models and uncommon postures. We also ensured a balance was kept between flat-lay and non-flat-lay images. A total of 635 images have been annotated for the gold standard set, as shown in table 3-3.

Table 3-3: Total number of objects and images annotated in the gold standard MLZ dataset.

Object	No of objects
upper-body	303

lower-body	205
full-body	175
footwear	508

3.2.1.5 Datasets naming convention used in the text for object detection

In the following, we will refer to the object detection datasets with the following naming convention ("OD" stands for "object detection"):

- *OD:MLZ dataset*: training dataset composed of manually annotated images, flat-lay images annotated via the Canny edge detection, augmented images, all balanced;
- *OD:DF1 dataset*: DeepFashion set (first version), selected for the objects;
- *OD:DF2 dataset:* DeepFashion set (second version), selected for the objects;
- *OD:AWS dataset*: set of images annotated for objects via AWS Sagemaker Ground Truth;
- *OD:MLZ-GS dataset*: the gold standard dataset, manually annotated for objects and built from MLZ images, used for evaluation of model performance.

3.2.2 Category classification phase

3.2.2.1 The MLZ dataset of category images

For the classification stage, we have also used the MLZ API to download randomised sets of images. This stage performs a set of supervised classifications (one per object type), thus a dataset of images labeled with garment category is needed.

We relied on the MLZ taxonomy and counted the number of images MLZ could retrieve for each category. Then, we chose the classes for each object type based on representativeness of the variety of fashion garments and class size (number of products).

The set of classes for each object type is specified in Table 3-4.

Table 3-4: List of category types based on object type.

Object	**Category classes**
upper-body	shirts, formal jackets, t-shirts, hoodies, tanks, camis, sweaters, cardigans, blouses, coats & jackets, shawls & capes & ponchos
lower-body	shorts, skirts, jeans, trousers
full-body	dresses, jumpsuits, playsuits
footwear	trainers, boots, flats, sandals, heels

To retrieve said images, MLZ queried its large database by class names, and used regular expressions to clean names in order to ensure minimal mismatch rate (e.g., for "shirts", garments like "shirt dress" must be excluded because they are dresses). In addition to this, synonyms of categories were taken into consideration while extracting the data from

the databases. For example, both “sweater” and “jumper” are used for the category of “sweaters”.

Each set of category images has also been manually checked by MLZ to make sure that spurious and wrong cases were removed. This procedure is time consuming but was feasible and deemed important to perform.

The number of images retrieved for each class is outlined in Table 3-5.

Table 3-5: Total number of images retrieved for each class under each object.

Object	Category	No. of images
upper-body	sweaters	14,380
	blouses	14,378
	coats and jackets	14,312
	shirts	14,247
	hoodies	14,100
	camis	6,655
	tshirts	12,953
	formal jackets	13,730
	cardigans	13,488
	tanks	3,575
	shawls, capes, ponchos	1,871
lower-body	jeans	13,224
	shorts	13,603
	skirts	14,021
	trousers	14,182
full-body	dresses	4,919

	jumpsuits	4,626
	playsuits	4,425
footwear	boots	14,125
	flats	13,355
	heels	13,559
	sandals	13,903
	trainers	14,303

Image augmentation is considered to upsample minority classes, prior to model training, in order to mitigate the effects of class imbalance, especially on the upper-body classes. New samples were generated proportionate to the imbalance ratio between a minority and the majority class. On this dataset, the image generator randomly performed horizontal flips, rotations with a factor of 0.3 and zooms within a range of [0.9, 1.2].

3.2.2.2 Datasets naming convention used in the text for category classification

In the following, we will refer to the datasets used for the category classification stage with this naming convention ("CC" stands for "category classification")

- CC:MLZ - dataset of images labelled for category, retrieved from the MLZ database and following the MLZ taxonomy;
- CC:DF - dataset of DeepFashion images, which are labelled for category, and remapped to the MLZ taxonomy.

The remapping of DeepFashion images to MLZ's taxonomy has been performed by a combination of quick matching of obvious cases (e.g., DeepFashion's category of "anorak" gets mapped to MLZ's category of "coats and jackets", DeepFashion's category of "culottes" gets remapped into MLZ's category of "trousers") and manually checking of the non-obvious ones.

3.2.3 Attributes detection phase

3.2.3.1 The MLZ dataset of images with attributes

The attributes of a garment can be many, and span many different features of a product. Using the MLZ taxonomy, the total number of main attribute keys is 9: pattern, style, leg style, sleeve style, hem style, fit, neckline style, rise, length. This count is excluding less important keys like sleeve length or back style. Each of the keys has a pool of possible values, which brings the total of attribute {key:value} pairs to 196.

Not every attribute key can apply to every object and category, e.g. leg style obviously applies solely to lower-body and to jumpsuits in the full-body group. Nevertheless, because the number of attribute keys a single product may have is large (e.g. a t-shirt can

have a pattern, a sleeve type, a length, a hem style and a neckline style), this renders the task rather complex to tackle. More precisely, producing all-between-all logically possible attribute combinations, from the MLZ taxonomy, results in approximately 107 million combinations.

We decided to then focus on the most important of the attribute keys, pattern and style (style expresses a detailed shape of the garment and functions as a subset of the category). Patterns can of course apply to any category, while styles may be attached to specific categories (e.g., a “skater” style applies only to shirts and dresses, a “gladiator” style applies to sandals, etc). The reasoning behind focusing on the most important one is of practical nature. It would be unfeasible to download and organise reliable datasets for all attributes, and there might also be problems in the training of neural networks with that much variety. In comparison with the ‘all-between-all’ combinations, when only using the attributes related to styles and patterns, the possible combinations are 1,840 which is more manageable. Taking a step-by-step approach and expanding the datasets for new attributes has been deemed the most reasonable approach to take for this step. In addition to this, synonyms of categories were taken into consideration while extracting the data from the databases. For example, “aztec”, “tribal”, “navajo” are synonyms for the “ethnic” attribute value of pattern/print. Hence as part of the pre-processing, the synonyms were replaced and maintained to the root word for easier identification and maintaining a unique attribute name.

For reference, we report in table 3-6 the list of values for each of these two chosen attribute keys, separated by the object type and category they apply to.

Table 3-6: Mallzee attribute values for the chosen keys, separated by object and category.

Attribute key	Attribute values list	Applies to (object: category)
pattern/print	ethnic, graphic, floral, tropical, striped, checked, animal print, polka dots, paisley, spots, tartan, geometric, colourblock, fair isle, camouflage, grid print, dip-dyed, tie-dyed, zigzag, washed	all objects, all categories
style	sweatpants, leggings, culottes, peg, harem, capri, formal, chino	lower-body: trousers
style	cargo	lower-body: trousers/shorts
style	jeggings, mom, boyfriend	lower-body: jeans
style	pencil, skater, a-line, frill, flowy,	lower-body: skirts; full-body:dresses
style	sweatshorts, cutoff, bermuda, skort, running, cycling	lower-body: shorts
style	wedding, bridesmaid, bodycon, tunic, jumper, shirt, shift, slip, tea, cocktail, pinafore, wrap,	full-body: dresses

	sundress, smock	
style	formal, collarless	upper-body: shirts
style	blazer	upper-body: formal jackets
style	boxy, varsity, baseball, boyfriend, fitted, sport, polo, muscle	upper-body: t-shirts
style	tie-front, bib, popover	upper-body: blouses
style	waterfall	upper-body: cardigans
style	duster, parka, trench, peacoat, coatigan, duffle, bomber, biker, puffer, anorak, windcheater, windbreaker, coach, borg, quilted, trucker	upper-body: coats and jackets
style	running, high-top, chunky, fashion	footwear: trainers
style	wellies, winter, chukka, chelsea, biker, cowboy, sock	footwear: boots
style	loafers, brogues, ballerinas, plimsolls, boat, moccasins	footwear: flats
style	mules, d'orsay, espadrilles	footwear: flats/heels
style	court	footwear: heels
style	gladiator, t-bar, toe-thong, flip-flops, sliders	footwear: sandals

Note that values like "shirt" which applies to dresses as a category, are also homonyms of categories themselves (see table 4-4) so care must be taken in retrieving the data for those in order to make sure that the category is right. Also, note that some values, like "boyfriend" or "biker" exist for multiple categories, hence again one needs to be careful when building such datasets. We have made sure that the retrieved datasets for these were as balanced as possible in the presence of multiple and/or single attributes, and manually checked for quality. In much the same way as in the case of categories, we have run specific queries against the MLZ database, taking care of all ambiguities as specified, and then also performed careful checks of the data for quality. This step was necessary, because there can be many situations where even if the product responds to a query for an attribute/keyword, the image retrieved may not (e.g., the attribute refers only to a specific detail, or to a side of the product which is not visible).

It is clear from table 4-6 that even with a few attribute keys the task of training classifiers is demanding in that labelled data is required for it, and values for each key may be many (for our choices, the pattern key has 20 values and the style key has 91 values). Furthermore, a standard classification approach would not bring the innovation we aim to achieve with this work, in that it would exploit the same logic used in the second stage, at a bigger scale. This work is aimed at creating novel intellectual property that is scientifically

sound, so we decided to also perform research using state-of-the-art approaches to this problem (see details in sub-section 4.3).

3.2.3.2 The DeepFashion dataset of images with attributes

In order to compare the results obtained from the dataset built with MLZ images, an attempt was made to evaluate the same models using the DeepFashion dataset. The DeepFashion dataset contained as many as 1,000 attributes with various levels and complexities. For example: the attribute “floral” appears in various forms and combinations, such as “floral”, “floral flutter”, “floral knit”, “floral lace”, “floral lace mini”, “floral lace sheath”, “floral lace skater”, “floral maxi”, “floral mesh”, “floral midi”, “floral mini” et cetera; this is due to the way the dataset has been created by the authors [3]. All the attributes in DeepFashion have been remapped to the attribute taxonomy that Mallzee created and only the ones matching the taxonomy have been considered. Given that we have selected only the “pattern” and “style” attribute keys from the taxonomy, for DeepFashion, we are left with 31 values in total after the pruning. The total number of images for these is 34,436. For reference, we report in table 3-7 the list of values for each of these two chosen attribute keys, separated by the object type and category they apply to.

Table 3-7: DeepFashion attribute values for the chosen keys, separated by object and category.

Attribute key	Attribute values list	Applies to (object: category)
pattern/print	floral, animal print, striped, tropical, checked, polka dots, paisley, tartan, fair isle, camouflage, grid print, zigzag	all objects, all categories
style	chino	lower-body: trousers
style	cargo	lower-body: trousers/shorts
style	boyfriend	lower-body: jeans
style	flare, flowy	lower-body: skirts; full-body: dresses
style	bodycon, smock	full-body: dresses
style	baseball, boxy, polo, sport,varsity	upper-body: t-shirts
style	biker, quilted, windbreaker	upper-body: coats and jackets

3.2.3.3 Datasets naming convention used in the text for attributes detection

In the following, we will refer to the datasets used for the attributes detection stage with this naming convention (“AD” stands for “attribute detection”)

- AD:MLZ - dataset of images labelled with attributes, retrieved from the MLZ database and following the MLZ taxonomy;
- AD:DF - dataset of DeepFashion images, which are labelled for attributes, and remapped to the MLZ taxonomy.

The remapping of DeepFashion's attributes to MLZ's ones has been performed with a combination of a quick matching of easy ones (e.g. DeepFashion's "abstract floral" gets mapped to MLZ's "floral" as a pattern) and manually checking of the non-obvious ones.

4. Illustration of the pipeline architecture

The employed architecture is a machine learning pipeline that hierarchically detects, classifies and annotates garments in image content with attribute values. Figure 3-2 in section 3 illustrates the idea:

1. The first stage is that of object detection: a model is trained to detect the presence, and location, of objects amongst the options of upper-body (clothes for the upper side of the body), lower-body (clothes for the lower side of the body), full-body (clothes that cover the whole body, like dresses and jumpsuits), or footwear. After the detection, each object gets cropped to its bounding box and passed to the second stage.
2. The second stage classifies each of these objects for its category. Categories are specific to each object (for instance the categories for lower-body objects are different from those of upper-body, thus different models are trained for each object type).
3. The third stage tags the objects with detailed attributes, which are structured into key:value pairs according to the taxonomy used (see section 4.1).

Using hierarchical pipeline architectures is common to solve complex tasks in computer vision [13]. In the following, we will summarise the results we obtained for each stage of the pipeline, the weaknesses we found and the improvements which could be made.

For each of the stages, we leveraged the power of Transfer Learning. We used pre-trained networks (open-access and freely available), that have been trained on large-scale datasets (of general type and category), tailoring their last layer to one's specific set of images. The networks then adapt their learning to the objects/categories desired. This is a common way to work with deep learning networks: training a deep network from scratch to make it learn shapes of interest in pictures would require enormous amounts of data and long computation times. Instead, transfer learning is commonly adopted as a working approach because the base network has already been learning general shapes in pictures, and can fine-tune its weights to the specific ones present in the specialised dataset.

Figure 4-1 clarifies the procedure and the chronological dependency of each stage to the subsequent one.

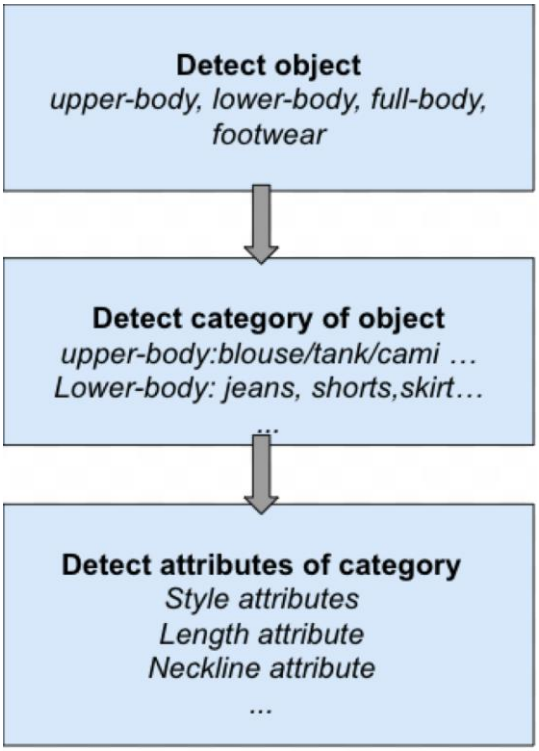

Figure 4-1: Outline of the hierarchical pipeline architecture.

4.1 Object detection phase

4.1.1 Model building and evaluation

This section describes the first stage of the machine learning pipeline which carries out the task of object detection. A pre-trained model on Microsoft's COCO dataset [14] as offered from TensorFlow's Object Detection API [39], is selected and fine-tuned to detect the existence and location of objects of interest in an image and then classify them among four high-level categories: "upper-body", "lower-body", "full-body" or "footwear". The experiments described in this section were performed on three different datasets, DeepFashion (OD:DF), DeepFashion2 (OD:DF2) and MLZ's object-level dataset (OD:MLZ).

For the evaluation of the Object Detection models, the metrics from COCO challenge [40] were utilized, which are integrated in TensorFlow's evaluation module. We will be focusing on the mean average precision (mAP) metric averaged over 10 intersections over union thresholds (IoU) (from 0.5 to 0.95 with steps of 0.05 size), which was the central metric of the competition, and on average Recall@K (AR@K) that signify the average recall given K detections per image. In the following sections, wherever "mAP" is referenced we refer to the "*mAP IoU = 0.50:0.95*" unless otherwise specified.

4.1.1.1 Using the DeepFashion dataset

We initially performed several experiments using the OD:DF dataset in a first comparative study to assess the best performing model for the task. After remapping the OD:DF dataset from the original attribute-level to the object-type-level it consisted of three classes, upper-body, lower-body and full-body items. Our initial comparative study consisted of 8 different object detection models, 4 provided by the TensorFlow1 (TF1) object detection API [41] and 4 provided by the TensorFlow2 (TF2) object detection API

[42]. In these experiments we down-sampled the OD:DF dataset to 10,000 instances for each class which were then split into 8,000 and 2,000 for the training and validation sets respectively. Additionally, the images were cropped around the object's bounding box, with a randomly selected factor of 5% to 20%, in order to force the model to focus on the desired part of the image and not on the background for instance. The models were fine-tuned for 300,000 training steps with a batch size of 1. The corresponding results are illustrated in Table 4-1, where "*Faster R-CNN InceptionV2"* [15] was able to outperform all other models. Although it is possible that the other models' predictive performances could be improved with further tuning of their hyper-parameters, this was not investigated further.

Table 4-1: Comparative study between 8 different object detection models, pre-trained on MS-COCO and fine-tuned on a down-sampled version of OD:DF. The evaluation is in terms of the mean average precision (mAP) and average recall at 1 (AR@1). (With bold we denote the best performance)

Model	TensorFlow version	mAP	AR@100
Faster R-CNN InceptionV2	1	**80**	**84.9**
SSD MobilenetV1 640x640 fpn	1	61.7	77.6
SSD InceptionV2	1	64.8	76.4
SSD MobilenetV2	1	57.6	72.4
SSD ResNet50V1 640x640 fpn	2	40.3	51.2
Faster R-CNN ResNet50V1 640x640	2	36.6	67.1
Faster R-CNN ResNet101V1 1024x1024	2	8.8	36.2
SSD ResNet152 v1 fpn 640x640	2	36.6	50

Afterwards, we empirically examined the idea of training three separate object detection models, one for each object type. The idea was spawned by one significant limitation of the OD:DF dataset. The dataset contains only a single annotated clothing item per image. This can be problematic when training object detection models because during the training phase, the model may detect garment items which are not annotated and are thus considered a wrong prediction by the loss function and by extension in the gradient updates. Similarly, while evaluating the model, correct predictions on non-annotated items will be considered as false positives by the evaluation metrics thus leading to partly uniformative results. We hypothesized that training three-separate models would ameliorate this issue. To this end, we randomly selected 10,000 images representing each class and trained three separate Faster R-CNN models. The results are illustrated in Table 4-2. The multiclass model had significantly higher predictive scores in both mAP@0.75 and AR@100. Additionally, the full-body-only model was found to recognise any combination of upper- and lower-body clothes as full-body, and was not able to discriminate between different classes.

Table 4-2: Faster R-CNN trained on a multi-class object detection task or three separate single class tasks (with bold we denote the best performance).

Dataset Version	mAP@0.75 IoU	AR@100
Multiclass OD:DF	**97.9**	**84.9**
Upper-Body only	79	74
Lower-Body only	92	84
Full-Body only	83	76

Based on the above outcome, we retraced to training one single multi-class model for the object detection task. A final experimentation was performed on a larger sample of OD:DF, where all classes were randomly down-sampled into 50,000 instances for each class. The larger down-sampled OD:DF was then split into 42,000 and 8,000 for the training and validation sets respectively. The model, after being trained for 1.5 million steps, was able to reach a mAP 83.7%. This amounts to a +3.7% improvement compared to the smaller dataset, indicating that larger datasets can result in better outcomes.

4.1.1.2 Using the DeepFashion 2 dataset

In order to mitigate the limitations and expand upon the OD:DF dataset, Yuying Ge et al collected and made publicly available the DeepFashion2 (OD:DF2) dataset. Therefore, we proceeded our experimentations with OD:DF2 since it contained multiple annotated garment items per image. OD:DF2 contains 13 classes (1 represents short sleeve top, 2 represents long sleeve top, 3 represents short sleeve outwear, 4 represents long sleeve outwear, 5 represents vest, 6 represents sling, 7 represents shorts, 8 represents trousers, 9 represents skirt, 10 represents short sleeve dress, 11 represents long sleeve dress, 12 represents vest dress and 13 represents sling dress), which were remapped for this task into upper-body (classes 1 to 6), lower-body (7 to 9) and full-body (10 to 13).

From the total 491,000 images consisting of 801,000 bounding boxes found in DeepFashion2, a smaller and balanced sample was selected, consisting of 42,000 objects per class in the training set and 8,000 objects per class in the validation set. The down-sampling limit was selected so as to create a dataset comparable to the OD:DF dataset. In order to balance the dataset, a simple undersampling heuristic was used, where all examples are kept until a class reaches our defined limit (42,000) and then ignores single instances of said class and its co-occurrence with other classes. The Imbalance Ratio (IR) for each class, the MeanIR and SCUMBLE metrics were calculated in order to examine the level of class co-occurrence and imbalance in multi-label data [16]. IR is calculated as the ratio between the majority class divided by all other classes individually while MeanIR simply reflects the mean value across all IRs. Similarly, SCUMBLE takes into account both the quotient and product among the IR of the various classes to "*evaluate the level of concurrence among minority and majority labels*" [30]. Lower values indicate lower levels of imbalance in the dataset and the lowest possible values are 1 for the meanIR and 0 for SCUMBLE. Our sample of OD:DF2 shows relatively low levels of imbalance in the initial dataset and since we are downsampling to a completely balanced set, MeanIR and SCUMBLE metrics are optimal as can be seen in table 4-3.

Table 4-3: The calculated meanIR and SCUMBLE metrics designate the level of imbalance present in the DeepFashion2 (OD:DF2) dataset (lower values indicate lower levels of imbalance).

Dataset State	Per-Class Distribution	meanIR	SCUMBLE
Original training set	upper-body: 139,789 lower-body: 122,838 full-body: 49,559	1.6528	0.1321
Balanced training set	upper-body: 42,000 lower-body: 42,000 full-body: 42,000	1	0

After balancing and transforming the dataset into “TensorFlow Records”, we selected to experiment with four object detention models, pre-trained on MS-COCO, that provide a balanced trade-off between accuracy and computational resources. To this end, we selected the models depicted in table 4-4 with the model’s mean average precision on the MS-COCO and its speed in milliseconds when run with 600x600 image on a GeForce GTX TITAN X, as offered from TensorFlow Object Detection model zoo (TF1 and TF2). It is important to note, that MS-COCO is a large scale dataset consisting of more than 330,000 images containing 1.5 million object instances of 90 object categories. Therefore the results of the aforementioned Object Detection models are not directly comparable - only relatively - to the selected fashion-related datasets which are of relatively smaller scale.

We kept the Faster R-CNN model from TF1 since it had the best performance on OD:DF but expanded the comparative study with three more models from TF2, two variants of EfficientNet and CenterNet with an HourGlass104 backbone.

Table 4-4: The selected object detection models from TensorFlow’s model zoo and their computational and predictive performance.

Model	Speed (ms)	COCO mAP
Faster R-CNN InceptionV2	58	28
CenterNet HourGlass104 512x512	70	41.9
EfficientDet D1 640x640	54	38.4
EfficientDet D2 768x768	67	41.8

The performance of each model can be seen in table 4-5, with CenterNet having the highest performance in 6 out of 10 metrics with a mean average precision (mAP) of 75.6% and an average recall at 1 (AR@1), of 83.5%. For all models the default hyper-parameters were used, as given by the TensorFlow object detection API, with the exception of the batch size that was significantly reduced to fit in the memory and the learning rate only in the case of EfficientDet-D1 which had a very high default learning rate that caused the model to quickly overfit while being fine-tuned on the OD:DF2 dataset.

Table 4-5: The performance of the selected Object Detection models, pre-trained on MS-COCO and fine-tuned on the DeepFashion2 (OD:DF2) dataset in terms of the mean average precision (mAP) and the average recall (AR) (with bold we denote the best performance).

Model	Faster R-CNN	EfficientDet-D1	EfficientDet-D2	CenterNet
Training Steps	1.5 million	360,000	400,000	300,000
Seconds per 100steps	~13.5	~40	~70	~75
Batch size	1	2	2	2
Parameters	Default	Learning Rate 1e-4	Default	Default
mAP	0.730	0.721	0.648	0.**756**
mAP (large)	0.738	0.723	0.650	0.**759**
mAP (medium)	0.484	0.**528**	0.420	0.437
mAP (small)	-	-	-	-
mAP@50IOU	0.919	0.**929**	0.898	0.905
mAP@75IOU	0.**852**	0.840	0.759	0.846
AR@1	0.806	0.791	0.728	0.**835**
AR@10	0.836	0.815	0.762	0.**868**
AR@100	0.837	0.819	0.767	0.**869**
AR@100 (large)	0.839	0.820	0.768	0.**871**
AR@100 (medium)	0.659	0.**667**	0.605	0.662

When the above models, fine-tuned on OD:DF2, were tested on OD:MLZ-GS excluding the footwear images they produced the results shown in the following table (table 4-6) where again the CenterNet with a HourGlass104 backbone showed the highest performance.

Table 4-6: Models trained on OD:DF2 and tested on the OD:MLZ-GS dataset (with bold we denote the best performance).

Model	**mAP**	**AR@1**	**AR@100**
Faster R-CNN	73.9	83.5	84.2
CenterNet	**76.9**	**85.1**	**86.8**
EfficientDet-D1	73.6	81.7	82.8

4.1.1.3 Using the MLZ dataset

Table 4-7 reports the results with a base model of Faster R-CNN, trained on the OD:MLZ dataset and considering all four objects and testing against the OD:MLZ-GS dataset. Refer to subsection 4.2.1.5 for the datasets.

Table 4-7: Faster R-CNN trained on variations of the OD:MLZ datasets, and evaluated on the OD:MLZ-GS dataset (with bold we denote the best performance).

Training dataset	mAP	AR@100
OD:MLZ	**80.6**	**85.8**
OD:MLZ + additional playsuits and tanks images	76	82.2
OD:MLZ + OD:AWS	58	72

The highest accuracy achieved was using the OD:MLZ which contains a balanced training set of manually annotated, flat-lay and augmented images. The number of training steps were increased to 500,000 due to performance of the model which is shown in the Precision-Recall graph (Figure 4-2 below). The mean average precision (mAP) for large, medium showed an improvement in the accuracy whereas the smaller objects plummeted significantly at about 200,000 epochs. The red line plotted on the graph shows the results of the OD:MLZ-GS dataset, which was used as part of the evaluation process with novel images.

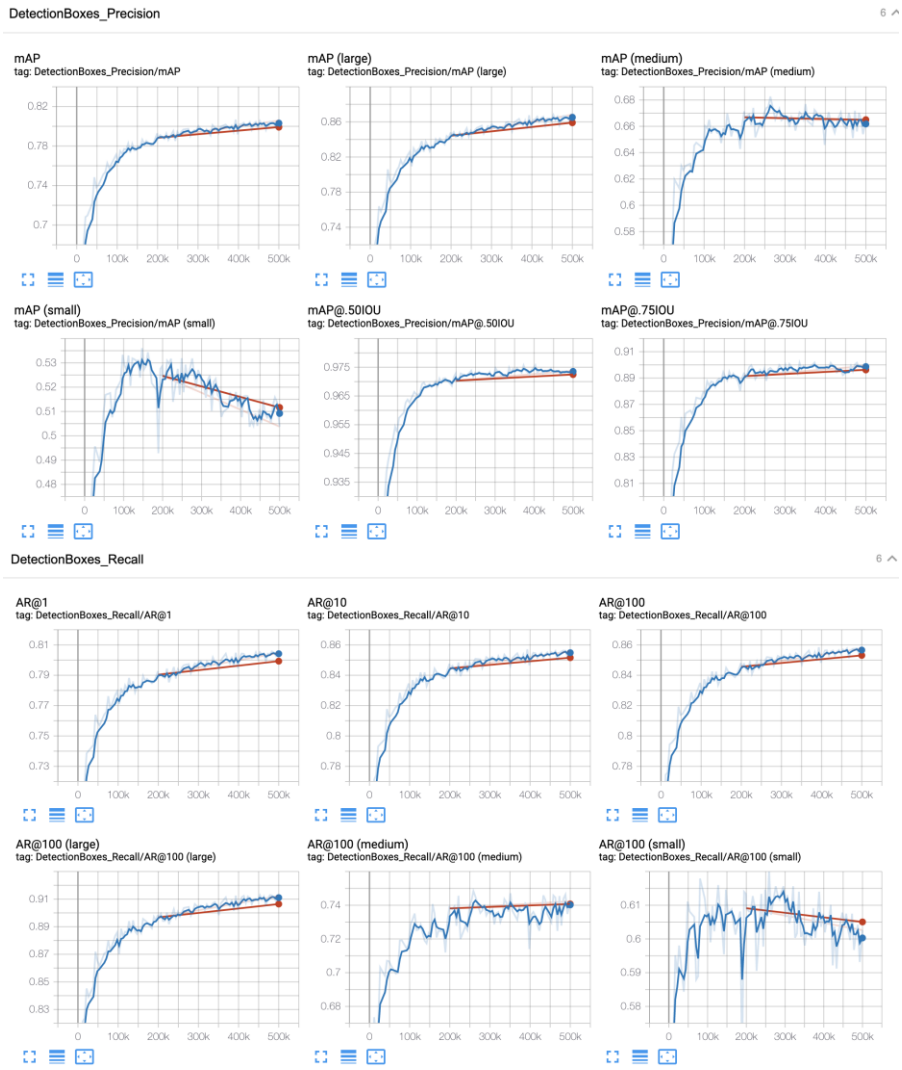

Figure 4-2: Precision and Recall graphs of the best Object Detection model (at 500k steps).

As part of the inference process, novel images from the OD:MLZ-GS were tested against the trained OD model, Figure 4-3 shows some of the results.

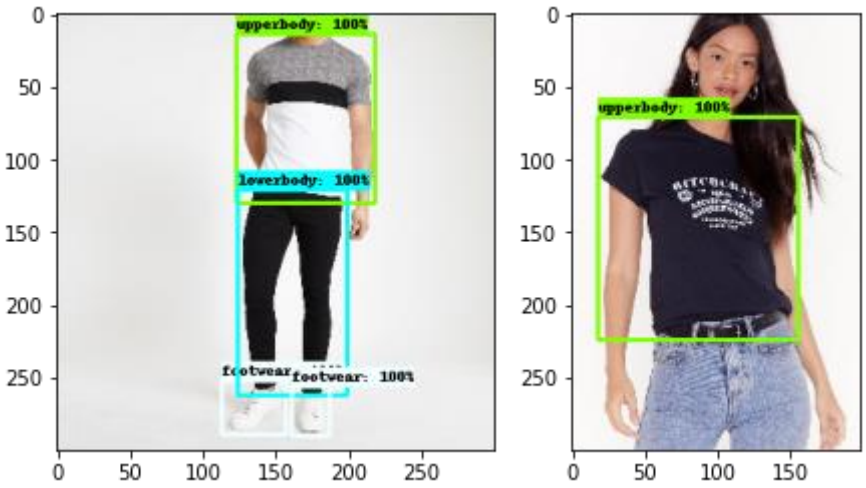

Figure 4-3: Inference results on some of the images in the OD:MLZ-GS dataset.

For completeness, we also report the main results obtained on the DeepFashion datasets, as per the description in sub-section 4.2.

As a point of comparison, the Match-RCNN, proposed in the original DeepFashion2 paper, was able to yield a mAP of 66.7% on the clothes detection task with 13 categories [4] More recently Alexey S. et. al., utilizing a modified CenterNet architecture and multiple post-processing techniques, were able to increase the mAP up to 73.7% on the same task [34]. While it can not be directly compared, we were able to reach 76.9% on the OD:DF2 dataset and 80.6% on the OD:MLZ dataset, with 3 and 4 object categories respectively. This indicates that our methodology is comparable with the State-of-the-Art especially when considering the results in the two following tasks, category classification and attribute detection.

4.1.2 Planned improvements

The model outputs show some recurring patterns of wrong detections which could be treated a-posteriori by virtue of simple heuristics.

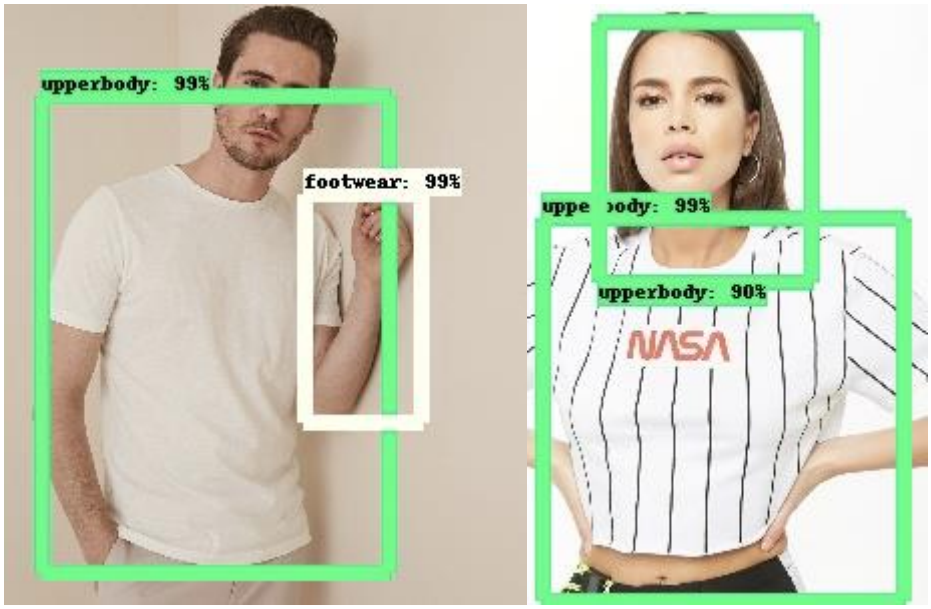

Figure 4-4: Examples of bad predictions from the object detection inference.

One such case is hands (of the human model) being detected as footwear, another is bare legs being detected as lower-body when the human model is wearing a short bottom garment (skirt or shorts), see Figure 4-4. There are some heuristics that can help, in at least the majority of cases, to clear out such situations:

- (for the hands): if a footwear is detected in the middle of the picture, or near upper-body garments, remove it as it is likely a hand, however note that this will not work in cases where hands are next to the feet, such as when the person appears in a seated pose, these are rare occasions but may be present;
- (for the bare legs): if two lower-body objects are detected in conjunction, one can think of considering only the smallest one, however this needs to be done with care because it may not work in all situations.

These are just suggested heuristics and likely non-comprehensive of all cases: a proper data analysis on a large set of outputs would have to be carried out to determine the best choice for robust post-processing rules. Furthermore, there might be other cases affected by similar problems which also need to be investigated.

Other improvements to the accuracy which could be implemented include further sophistication in terms of scene learning [31] and human parsing [32] for a better detection of the relations amongst objects.

4.2 Category classification phase

The second stage of the developed machine learning pipeline carries out the task of image classification into mid-level categories. This task was treated as a multi-class classification since there are multiple classes that are mutually exclusive. Additionally, since the images are first passed through an object detection model, apart from having the cropped images focused only on the object of interest, the high-level class (object) of the item will also be known (upper/lower/full-body or footwear). Thus, four different models could be trained, one for each object type category. This approach could improve the specialization of each model and thus improve their accuracy with the possible tradeoff that misclassified objects at the object detection phase would be passed in the wrong model. For example, if a playsuit (which belongs in the full-body object type) was wrongly predicted as a shirt

(which belongs in upper-body category), it would be passed in the 'upper-body' classification model which would not have previously seen playsuits. As a result, we experimented with both approaches, 1) having four separate models for each object type and 2) one model trained on all 22 classes in order to empirically examine the overall performance of each.

4.2.1 Model building and evaluation

Working with the annotated CC:MLZ dataset enabled the training of supervised multi-class classification models. Similarly to the object detection phase, the most central challenges of this stage were to:

1. Experiment with various deep learning models
2. Handle the issue of class imbalance
3. Tune the hyper-parameters of each model
4. Select the appropriate transfer learning approach

Due to constraints in computational resources and having a moderately sized dataset we decided to experiment with models that have a relatively limited number of layers and parameters but that also had shown adequate performance in benchmark datasets.
Given these criteria, we selected the following deep learning architectures: Xception, InceptionV3 and multiple variations of EfficientNet whose details as well as their performance on ImageNet can be seen in table 4-8.

Table 4-8: The selected pre-trained computer vision models for category classification.

Model	Total Parameters *	Layers *	Accuracy@1 on ImageNet **
EfficientNet-B1	6,575,239	339	78.8
EfficientNet-B2	7,768,569	339	79.8
EfficientNet-B3	10,783,535	384	81.1
EfficientNet-B4	17,673,823	474	82.6
Xception	20,861,480	132	79
InceptionV3	21,802,784	311	78.8

* As implementation in Keras API ** Source : Papers with Code

In order to train and evaluate the performance of these models, the image data and their labels had to be fed into the model with the use of a data generator that creates a batched TensorFlow dataset. The dataset was split into three separate sets for training, validation and testing based on a ratio of 0.8/0.1/0.1 respectively. The exact same sets were used for all experiments in order to ensure reproducibility and a fair comparison.

The workflow in all supervised multi-class experiments consisted of the following steps. Initially, one of the models shown in Table 4-8 was selected from Keras API with pre-trained weights on ImageNet without its previous final classification layer. Then, the input images from the training set are passed through an image augmentation preprocessing layer that performs subtle alterations to the images as a method of regularisation and by extension the mitigation of over-fitting. The images were horizontally flipped at random

and were randomly rotated and zoomed in by a low factor of 0.1 out of 1. The augmented images are then passed through the base model. The features that are extracted from the last convolutional layer of the base model are pooled with the use of global average pooling thus transforming the output into a 2D tensor. The original classification layer is replaced by a new dense layer, with uninitialized weights that is activated by the softmax function and has an output size equal to the number of classes in the dataset. Subsequently, the model is trained by an adaptive gradient descent optimizer (Adam or RMSProp) which performs gradient updates per individual parameter, with the use of sparse categorical cross-entropy as the loss function, since the labels are encoded as single integers. After each epoch, the model is evaluated on the validation set in terms of its predictive accuracy; if the validation accuracy is improved a callback is called to checkpoint the model's weights and if the validation accuracy is not improved for consecutive epochs, the training process is terminated and the weights of the best performing epoch are restored.

The issue of class imbalance was not a major challenge for this task due to the fact that the CC:MLZ dataset was collected with the explicit objective of creating a balanced dataset. However, in the cases that a slight imbalance was present, two approaches were examined: 1) using the class weight parameter or 2) upsampling the minority classes with the use of image augmentation. In the first approach, the imbalance found in the dataset is left unaltered but the class weight parameter that is available in Keras' *.fit()* function was set to be proportional to imbalance ratios between all classes. In the second approach the minority classes were upsampled with the use of data augmentation techniques. The generated samples were proportionate to the imbalance ratio between a minority and the majority class. Specifically, the image generator randomly performed horizontal flips, rotations with a factor of 0.3 and zooms within a range of [0.9, 1.2].

Both approaches were tested on the upper-body-only dataset, which had 11 different classes with a mean value of 11,244 images per class, a median of 13,730, a maximum of 14,380 in *"sweaters"* while *"camis"* had 6,655 samples and "*tanks"* 3,575. It was concluded that training an EfficientNet-B1 with the upsampled dataset yielded a better performance of +1.92% compared to the balanced class weight approach. Similarly, training the same model on the CC:DF upper-body-only dataset, there was a slight improvement of +0.52% with the upsampled dataset. We therefore proceeded using the "upsampling by image augmentation" technique for most experiments, with the exception of models that were trained on the whole CC:MLZ dataset since balancing the dataset would significantly increase its scale and as a result the required computational resources.

During the time that we were executing the initial experiments on the CC:MLZ dataset we were also implementing them on the CC:DF dataset. The central intention behind this idea was to later merge the two datasets, since a larger dataset would potentially lead to improving the model's performance. The CC:DF dataset was downsampled per class to match the CC:MLZ's dataset distribution. Additionally, the former's class taxonomy was remapped to match the latter's. However, the same network architecture would consistently underperform when applied on the CC:DF dataset. An EfficientNet-B1 model had +14.15%, +11.32% and 5.68% better performance, for upper-body-only, full-body-only and lower-body-only respectively, when trained on the CC:MLZ dataset compared with the CC:DF dataset. After randomly sampling images from the CC:DF dataset, it was assessed that its attribute-level annotations (the dataset's original state, annotated with fine-grained attributes, without applying any remapping of the categories) contained a certain amount of 'noise', wrongly labeled instances, that when mapped on the category-level resulted in lower performance. When the two full-body-only datasets were merged, from both CC:MLZ and CC:DF, the model's predictive accuracy was still 7.84% worse than when using only the CC:MLZ dataset. It was extrapolated that MLZ's category-level

dataset, despite being smaller in size, had lower rates of wrongly labeled instances and thus was chosen for the subsequent experiments.

In all aforementioned models, there were a few important hyper-parameters that required manual and careful selection in order to derive the best possible performance for each model. Those included the learning rate, the rate in the dropout layers, the batch size and the optimizer. The hyper-parameters and the selected values are shown in table 4-9. Due to computational limitations, we were unable to run a complete grid search between all models, hyper-parameters and datasets but rather enough comparative experiments were performed and combinatorial directions that did not perform optimally in consecutive experiments were discarded and consequently, we continued experimenting with those that had showed better performance.

Due to the fact that we mostly relied on fine-tuning certain portions of pre-trained networks, the learning rate had to be kept relatively low in order to avoid abruptly updating the gradients and by extension lead to either under- or over-fitting. Accordingly, we chose to experiment with low learning rate values of 1e-5, 5e-5 and 1e-4. In most cases, we found that all three rates resulted in approximately the same result but lower rates required more epochs to reach the global minimum of the problem and more computational resources and time. Thus, we maintained a learning rate of 1e-4 for the majority of the experiments. Secondly, regarding the dropout layer rate, we experimented with values ranging from 0.3 to 0.5. As can be seen in Figure 4-5, larger dropout rates, 0.4 and 0.5, led to a progressively smaller divergence between training accuracy and validation accuracy, which translates to a more stable training process and decreased rates of over-fitting. Additionally, in this particular experiment, the model with the higher dropout rate (0.5) was able to converge to the same result but two epochs faster than the models with 0.3 and 0.4 dropout probability. As a result, in most experiments, we maintained a dropout rate of 0.5.

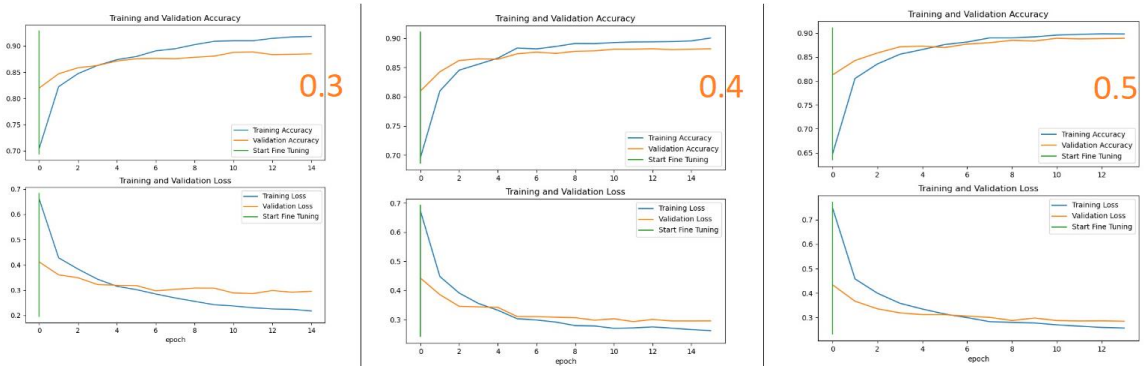

Figure 4-5: Training history of EfficientNet with the same hyper-parameters but with different dropout layer rates.

The defined batch size is naturally dependent on the available memory and consequently on the network's size. We selected batch sizes of 16, 32, 64 that were the maximum based on each individual network size and the available memory. Finally, following the original EfficientNet paper we utilized the RMSProp [17] optimizer with a decay rate of 0.9, a momentum rate of 0.9 and batch norm momentum 0.99 but when compared with the Adam optimizer, Adam had a very slight advantage of +0.67% over RMSprop but a significant 63% faster convergence time with EfficientNet-B1 trained on the full-body-only dataset. As a result, Adam was selected for the proceeding experiments.

Table 4-9: The hyper-parameters and their values in the parameter grid search.

Hyper-parameter	Values
Learning Rate	1e-5, 5e-5, 1e-4
Dropout Rate	0.3, 0.4, 0.5
Batch Size	16, 32, 64
Optimizer	RMSProp, ADAM

Transfer learning techniques were also utilized at this stage. Various deep learning models that were pre-trained on a large image dataset (e.g. ImageNet) for the task of image classification were selected as the 'base model' and a new untrained classification layer was added on top. This method allows for three main approaches:
1) **Feature Extraction**. The pre-trained parameters are 'frozen' and the pre-trained model is used to extract features from the images. In this approach only the weights and biases of the classification head are trained while the base model's parameters are left unaffected throughout the training process.
2) **Fine-Tuning**. All or a certain portion of the trainable parameters of a pre-trained deep learning architecture are 'unfreezed' and 'fine-tuned' for the target dataset.
3) **Hybrid**. A pre-trained model is initially used as a feature extractor and once the added classification layer is trained on the target task, a certain portion of the trainable parameters are 'unfrozen' and fine-tuned. This approach is recommended by TensorFlow's fine-tuning guideline [43] in order to avoid the base model forgetting what it had learned during pre-training by large gradient updates.

Considering that the dataset was specific to the domain of fashion, solely relying on 'feature extraction' would lead to suboptimal results since a model pre-trained on ImageNet, a general purpose dataset, would not have been exposed to enough fashion-related imagery. On the other hand, due to the relatively limited magnitude of the annotated CC:MLZ dataset, approximately 300,000 images compared to the 14,197,122 found in ImageNet [18], fine-tuning all trainable parameters of a pre-trained image classification model would most likely lead to overfitting and would also require significantly more computational resources. Thus, fine-tuning a certain portion of a pre-trained model was deemed as the most appropriate pathway.

In order to empirically compare the efficacy of each approach, Feature Extraction (FX) and Fine Tuning (FT), we utilized an Xception and an EfficientNet B1 model with exactly the same hyper-parameters where in the first case all the trainable parameters were 'frozen' while in the second, the 50 to last layers of the network were 'unfrozen' and fine-tuned during the training stage. As can be seen in the left side of both figures 4-6 and 4-7, when using the pre-trained model solely as a feature extractor the accuracy of the model was slowly reaching a plateau of approximately 80% in terms of validation accuracy while continuing the training process with unfreezing a few of the trainable parameters significantly increased the validation accuracy to approximately 90%.

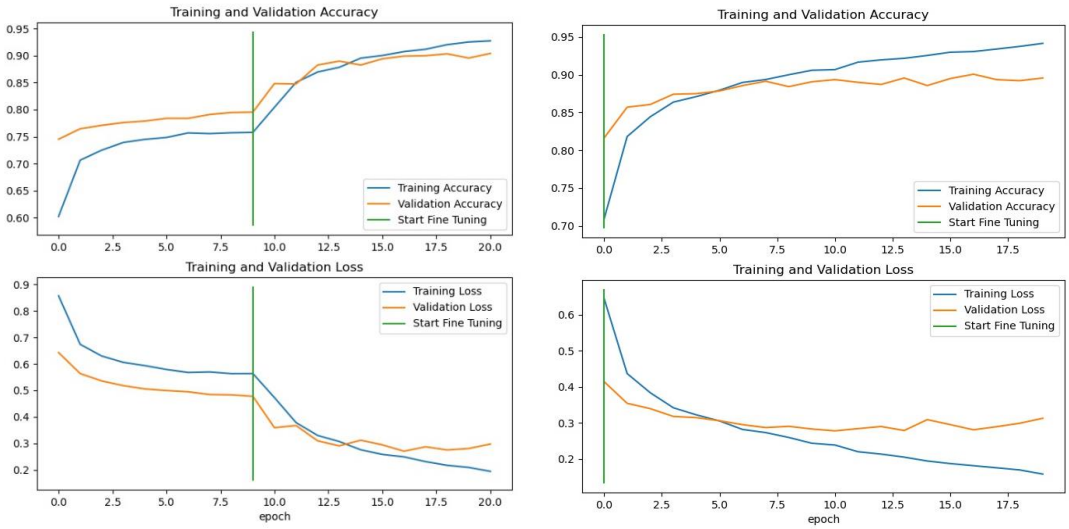

Figure 4-6: An Xception model trained on full-body categories used as feature extraction and then fine-tuned (left) and fine-tuned from the first epoch (right).

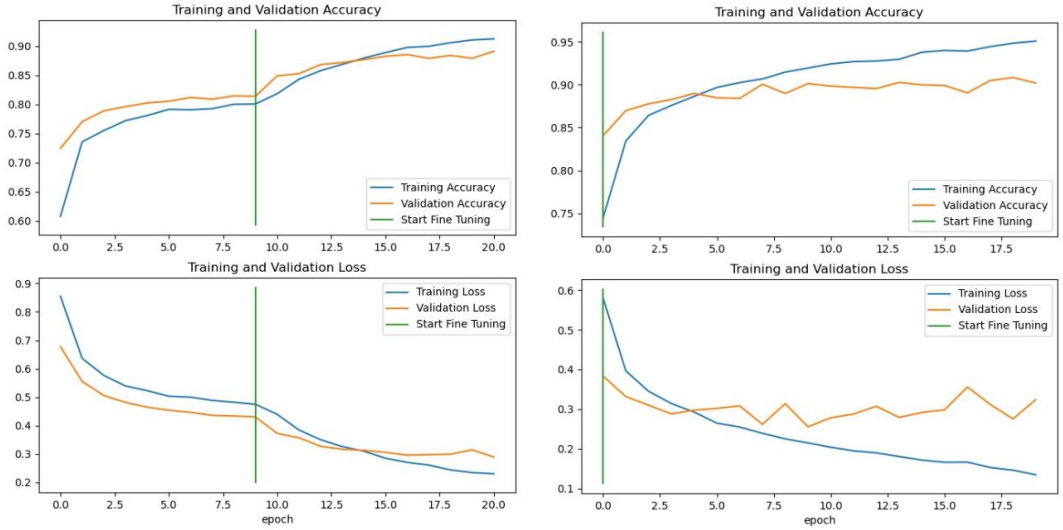

Figure 4-7: An EfficientNet-B1 model trained on full-body categories used as feature extraction and then fine-tuned (left) and fine-tuned from the first epoch (right).

Furthermore, comparing the results between direct Fine-Tuning and the Hybrid approach (table 4-10), the first approach led to a slightly better performance of +1.15% in Xception and +0.86% in EfficientNet B1 indicating that the concern of 'forgetting what the model had learned during pre-training' was not present in our experiments. The aforementioned experiments were performed on the full-body-only dataset but it was also reproduced in the upper-body-only dataset. Despite the seeming training instability of directly fine-tuning a pre-trained model and its visible over-fitting (on the right side of Figure 4-6 and Figure 4-7) we concluded that with stronger regularisation (higher dropout rate and image augmentation) the training process would become more stable and the divergence between training and validation accuracy would decrease thus mitigating the phenomenon of over-fitting on the training dataset. Additionally, utilizing an Early Stopping callback could end the process significantly earlier while conserving high levels of accuracy when compared with the Hybrid approach which required additional epochs for only training the classification layer. Therefore, we proceeded with selecting the second option, of direct fine-tuning, for experiments reported in table 4-10.

Table 4-10: Comparing a Hybrid Transfer Learning approach where a model is first utilized as a Feature Extractor (FX) and then Fine-Tuned (FT) with directly Fine-Tuning the model from the first epoch.

Model	Dataset	Fine-Tuning Method	Test Accuracy
Xception	Full-body-only	10 FX epoch + 10 FT epoch	90.7%
Xception	Full-body-only	20 FT epochs	91.85%
Efficient Net B1	Full-body-only	10 FX epoch + 10 FT epoch	91.13%
Efficient Net B1	Full-body-only	20 FT epochs	91.99%
Xception	Upper-body-only	10 FX epoch + 10 FT epoch	84.05%
Xception	Upper-body-only	20 FT epochs	83.91%
Efficient Net 1	Upper-body-only	10 FX epoch + 10 FT epoch	85.91%
Efficient Net 1	Upper-body-only	20 FT epochs	86.27%

Performing multiple experiments with direct fine-tuning of InceptionV3, Xception and EfficientNet variants we concluded that EfficientNet B3 and B4 were able to consistently outperform the other architectures in all datasets. A significant advantage of EfficientNet is that its larger variations are scaled efficiently in terms of all three neural network 'dimensions': input resolution, depth and width. We hypothesise that this fact enables for improvements in predictive accuracy while maintaining manageable demands in terms of computational resources. A selection from all performed experiments comparing the various network architectures when different amounts of the model's highest layers are fine-tuned, can be seen in table 4-11. Only the best performances are presented in the table by each network on different quantities of fine-tuned layers with a dropout rate of 0.5 and the batch size being constant for each dataset.

Table 4-11: The performance of various models pre-trained on ImageNet and partly fine-tuned for garment category classification. (With bold we denote the best performance)

Training Dataset	Model	Fine-tuned Layers	Learning Rate	Accuracy
CC:MLZ - full-body	EfficientNet-B1	30	1e-4	90.69
		50	1e-4	91.7
		100	1e-4	91.77
	EfficientNet-B2	80	1e-4	92.34
		100	1e-5	91.34

	EfficientNet-B3	80	1e-4	91.41
		100	1e-4	93.63
	EfficientNet-B4	80	1e-4	93.77
		100	1e-4	**94.2**
	Xception	30	1e-4	93.06
		50	1e-4	92.91
		100	1e-4	92.91
CC:MLZ - lower-body	InceptionV3	30	1e-4	89.79
		50	1e-4	89.72
		100	1e-4	92.7
	EfficientNet-B1	30	1e-4	90.37
		50	1e-4	90.19
		100	5e-5	91.79
	EfficientNet-B2	100	1e-4	91.46
	EfficientNet-B3	100	1e-4	93.19
	EfficientNet-B4	100	1e-4	**93.86**
	Xception	50	5e-5	93.75
		100	1e-4	93.3
CC:MLZ - footwear	InceptionV3	30	1e-4	90.75
		50	1e-4	91.87
	Xception	30	1e-4	92.36
		50	5e-5	93.17
		80	5e-5	93.59
		100	5e-5	92.86
	EfficientNet-B1	30	1e-4	92.52
		50	1e-4	92.61
		100	1e-4	93.39
	EfficientNet-B2	100	5e-5	93.89
	EfficientNet-B3	100	1e-4	93.94
	EfficientNet-B4	100	1e-4	**94.21**

CC:MLZ - upper-body	EfficientNet-B1	50	1e-4	80.14
		100	1e-5	83.16
		140	5e-5	85.31
	EfficientNet-B2	100	1e-4	83.91
		140	1e-4	86.04
	EfficientNet-B3	100	1e-4	86.33
		140	1e-4	85.75
	EfficientNet-B4	100	1e-4	**87.64**

All the above experiments were performed with the original MLZ dataset consisting of full scale images. Since the images were not cropped, separating different object types, some could entail items belonging in different garment categories. After training the Object Detection model, described in the previous section, we were able to crop the images around their predicted bounding boxes and keep only those matching the mid-level garment category. If for example, an image depicted a model wearing both a t-shirt (upper-body) and trousers (lower-body) but the image was retrieved from the CC:MLZ dataset labeled as a t-shirt (since that was the marketable product in that case), only the first cropped image would be kept for the category classification stage. This enabled the more 'focused' and attentive training of the four separate models, one for each object type. Comparing the models trained on the uncropped images with those that used the cropped ones, we can observe a modest improvement of +0.37% from a mean value of 92.47% to 92.84% respectively. The difference between the two was initially expected to be higher. We hypothesise that since each model only encountered items from a specific object type and would also encounter images of the same class photographed from various ranges (from close-ups to long-shots) they were able to extract and identify only the relevant features relating to each of the known classes. Additionally, using the cropped CC:MLZ enabled the training of one model for all 22 garment category classes. The best performing model for this task, a fine-tuned EfficientNet-B4, was able to yield a 92.24% accuracy score, a slight -0.6% compared to the mean accuracy of the four separate models. However, having a single model for all 22 garment categories has two significant advantages. First, it is easier to re-train and more efficient to employ compared to having four separate models. Secondly, having one model for all classes does not suffer from the misclassified items that may result from the object detection phase. In the case of having separate models for each body type, an object wrongly classified as an "upper-body" type while being "full body" will be passed in the wrong model that has not been trained to recognise full-body type items. Therefore, we consider that the very slight performance drop is outweighed by the two aforementioned advantages. The detailed results for each of the best performing models per dataset can be seen in table 4-12.

Table 4-12: Summary of best performing models for the task of category classification.

Training dataset	Cropped Images	Accuracy	Network model used	Hyper-parameters
CC:MLZ - **Full Dataset**	Yes	92.24%	EfficientNet-B4	**Fine-tuned layers**: 100 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** class weight
CC:MLZ - **Full-Body**	Yes	94.9%	EfficientNet-B4	**Fine-tuned layers**: 100 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** over-sampling
	No	94.2%		
CC:MLZ - **Lower-Body**	Yes	94.44%	EfficientNet-B3	**Fine-tuned layers**: 50 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** over-sampling
	No	93.86%	EfficientNet-B4	**Fine-tuned layers**: 100 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** over-sampling
CC:MLZ - **Footwear**	Yes	93.86%	EfficientNet-B4	**Fine-tuned layers**: 100 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** over-sampling
	No	94.21%		
CC:MLZ - **Upper-Body**	Yes	88.15%	EfficientNet-B4	**Fine-tuned layers**: 100 **Learning rate**: 1e-4 **Batch Size**: 32 **Dropout rate** : 0.5 **Balancing:** over-sampling
	No	87.64%		

As a point of comparison, the FashionNet model, proposed in the original DeepFashion paper, was able to yield 82.58% and 90.17% in terms of top-3 and top-5 accuracy respectively on the Category Classification task with 50 categories [3]. More recently Li. et. al., with the use of multi-task learning, were able to increase the top-3 and top-5 accuracy scores up to 93.01% and 97.01% on the same dataset [33]. Applying our proposed methodology on the DeepFashion dataset - first passing the images through our object detection model and then re-training an EfficientNet-B4 network on the cropped images - was able to slightly surpass the current state-of-the-art with 93.71% and 97.40% top-3 and top-5 accuracy. Additional advantages of our approach when compared with previous studies include the ability to work on full-scale real-world fashion imagery, that may depict multiple garment items per image, without utilizing overly complicated architectures, manual guidance from domain experts or requiring landmark and mask annotations; which are arguably two costly and time-consuming types of annotation.

4.2.2 Planned improvements

The models for this stage perform well, unsurprisingly given the current quality of state-of-the-art neural networks for basic classification tasks. However, there may still be room for improvement in terms of minimising misclassifications that may arise from certain ambiguous cases due to the general diversity of fashion imagery.

Some further data cleaning work could be performed to achieve a clearer separation of classes in the first place, so that ambiguity is improved, for instance in situations like the difference between jeans and trousers.

4.3 Attributes detection phase

The third stage of the developed machine learning pipeline carries out the task of detecting fine-grained attributes from garment imagery and classifies them into multiple low-level categories. The selected types of attributes that the model was trained to discern were 20 classes regarding the pattern or print of the garment and 92 types of low-level categories, a total of 112 classes. Due to the fact that styles and patterns are not mutually exclusive - meaning that a garment may possibly be characterised as having both a style and a pattern (e.g. a checked shirt or a floral sundress) - this task is treated as a multi-label classification problem. However, it was assumed that styles and patterns are internally mutually exclusive (an item can not have both “floral” and “camouflage” as its print, for instance), meaning that an item can not belong to more than one style or print/pattern at the same time. This assumption was also reflected in the collected MLZ dataset with only 1,175 instances exhibiting more than one style or pattern attributes. After manually examining it, it was deemed that most were either wrongly labeled or very rare outliers. The MLZ dataset utilized for attribute detection consists of a total of 310,358 image-text pairs, after dropping a few duplicates and filtering instances with more than one type pattern or style. The dataset was split with a ratio of 9 to 1, into a training set (279,322) and a validation set (31,035). The exact same sets are being used in all following experiments regarding attribute detection.

4.3.1 Model building and evaluation

For the task of Attribute Detection, two main approaches were examined:

1. Cross-modal Vector Alignment for Image-Text pairs
2. Multi-label Supervised Learning

The first approach was inspired by two recent papers by OpenAI [19] and Google Research [20] that utilized contrastive learning between Image-Text pairs in order to perform various classification and retrieval tasks. The objective of the first approach was to train a vision encoder for the task of detecting fine-grained fashion attributes without the need of manually annotating a considerable amount of images. The second approach was reserved as a back-up plan in case the first did not perform adequately but would require the collection of an annotated dataset.

4.3.1.1 Cross-modal Vector Alignment for Image-Text pairs

Both OpenAI’s CLIP and Google’s ALIGN follow a similar workflow that requires one Image Encoder, one Textual Encoder and for both to be trained simultaneously with the use of contrastive learning. During the training phase, the image and text of a pair are given to their respective encoder, the resulting embeddings of both encoders are ‘projected’ in the same embedding space. Thereafter, a dual encoder calculates the dot product between image and text embeddings and the loss is calculated as the mean cross

entropy between the predicted and the target image-text pairs, reflected in the main diagonal. After the training is completed, the dual encoder is discarded and only the two separate encoders are saved. Both models can be used for either zero-shot classification or retrieval tasks. In the first case, for performing the zero shot classification, all desired 'target sentences' are given in the text encoder and each image is being matched with the top-K most appropriate captions/sentences.

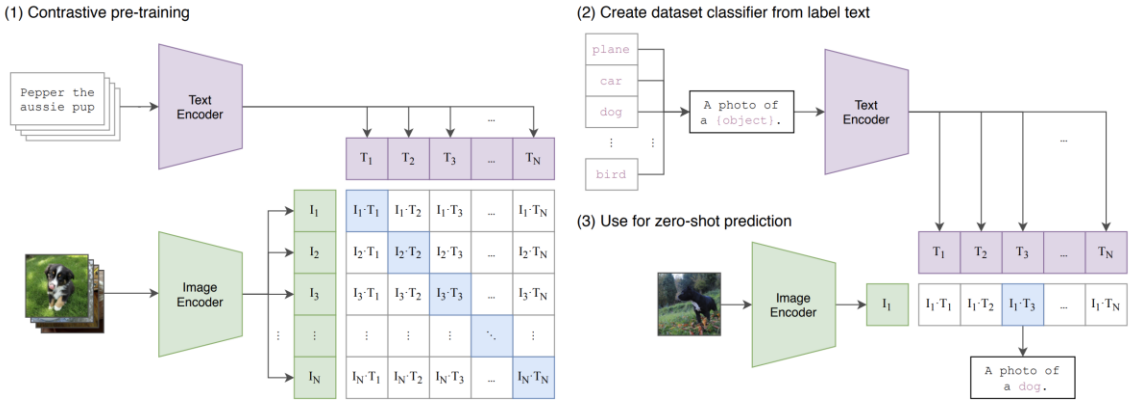

Figure 4-8: OpenAI's CLIP workflow for pre-training the model (left) and for performing zero-shot classification (right)

OpenAI's CLIP was made publicly available shortly after the paper's release, thus we were able to experiment with it as a zero-shot classifier or fine-tuning it on the fashion domain. On the other hand, Google's ALIGN has not yet been made available and therefore we considered recreating a similar architecture. Instead of using the Visual and Textual transformers used in OpenAI's CLIP, we used similar components to Google's ALIGN. Initially we used variations of the BERT (Bidirectional Encoder Representations from Transformers) model (bert-base-uncased) [21] for the text encoder and variants of EfficientNet for the Image Encoder, a family of models that has proven very effective in the Image Classification Task. The EfficientNet models pre-trained on ImageNet were taken from Keras API while the variants of BERT were taken from TensorFlow Hub. On top of both pre-trained encoders, one or more fully connected layers are added with a predefined output dimension so that the image and text embeddings can be matched in the same embedding space. These layers will be referred to as 'projection layers'. In the case where more than one projection layer is added on top of the encoders, the in-between layers are connected with an optional Dropout layer and are activated by a GeLU activation function.

During the training phase, the image-text pairs are read as a *tf.dataset* with the use of a data generator. The images are resized according to each EfficientNet variant expected input, passed through a data augmentation pre-processing layer that performs random horizontal flips, rotations and zooms by a factor of 0.1. The text is tokenized with BERT's tokenizer, also offered by TensorFlow Hub, and then trimmed by a user-defined sequence length, with the maximum possible value being 512. After preprocessing both the images and texts inside the batch, the data are passed through their respective encoder, their output embeddings are projected onto the same embeddings space and then are normalized with l2 normalization. Afterwards, the loss function is calculated, similarly to OpenAI's CLIP, as the mean cross entropy between the scaled pairwise cosine similarity of image and text embeddings and the target matrix, which is an identity matrix of size equal to the batch size. As with OpenAI's CLIP, the scaled pairwise cosine similarity is calculated as the dot product between Image Embeddings and the transpose of Text Embeddings multiplied by the exponential of a temperature parameter. Temperature is a

trainable variable which is initially set to 0.07 and its exponential is clipped to a maximum of 100 (so as to avoid training instability).

During the evaluation phase the text encoder receives all possible combinations between different attribute types in the form of sentences and matches the image embeddings with those sentences. All selected top-1 sentences for each image are transformed into multi-label binaries and evaluated against the ground truth of the validation set. In the case of using only styles and patterns/prints, there were 1840 possible combinations between the two. Because patterns/prints do not have a 'NONE' option (for the cases where no particular pattern is present) the 92 garment styles were also added on their own, resulting in 1,932 possible sentences for the multi-label evaluation phase. When using the category classification dataset, the 22 classes were given as the potential captions for the evaluation phase.

Similarly to the category classification stage, multiple models and hyper-parameters had to be carefully selected in order to access the best possible performance. In terms of model selection, we mostly experimented with EfficientNet B3 and B4 which had the best performance in category classification, and four small-BERT variants which can be seen in table 4-13. Indicatively, Bert-L-4_H-512_A-8 stands for: Small uncased BERT with L=4 transformer blocks, a hidden size of H=512, and A=8 attention heads.
The models were optimized with the use of AdamW, or Adam with weight decay, with a weight decay value of 0.01, 0.99 for the beta_1 and 0.999 for beta_2.

Table 4-13: Variations of small-Bert utilized as the text encoder.

Small BERT models	Transformer Blocks	Hidden Layer Size	Attention Heads	Total Parameters*
L-2_H-128_A-2	2	128	2	4,385,921
L-4_H-256_A-4	4	256	4	11,170,561
L-4_H-512_A-8	4	512	8	28,763,649
L-12_H-768_A-12	12	768	12	109,482,241

* As implemented in TensorFlow Hub

Regarding the selection of hyper-parameters, the Learning Rate, Batch Size, Dropout rate the number of projection layers and their dimensions had to be tuned. The detailed hyper-parameter space that we experimented with, can be seen in table 4-14.

Table 4-14: Hyper-parameter space for the conducted experiments

Hyper-parameter	Value
Learning Rate	1e-3, 1e-4, 5e-5
Batch Size	32, 64, 128, 256
Dropout rate	0.1, 0.3, 0.5
Projection Layers	1, 2, 4, 8
Projection Layer Dimensions	128, 256, 512

Transfer learning was also used for this task, where we experimented with the following approaches:

1. Using the pre-trained encoders as ‘frozen’ feature extractors
2. Only fine-tuning the Vision Encoder
3. Fine-tuning both the Vision and Textual Encoders.

During the time of retrieving the MLZ Attribute dataset, we performed numerous experiments with the Image-Text pairs from the category classification. For this task, the images and their “product names” - very short descriptions about the product - were used. A noteworthy, but expected, insight was that the contrastive model was very sensitive to changes in language. When the model was trained on the original, unprocessed, textual data (which is naturally diverse and contains synonyms) it would under-perform during the evaluation phase. When the original category classes were provided as sentences, for example “t-shirts / formal jackets / skirts” etc, the model would under-perform due to the fact that it had not encountered these words in this particular form, while changing the target sentences (after manual experimentation) to “tee / blazer / suit / skirt”, etc., there was an increase of +13%. As a result we chose to alter the training set by replacing all important keywords and their synonyms related to the garment categories with only one specific word. This way we could know that the target sentences were reflected in the text. The exact same process was also applied while retrieving the MLZ Attribute dataset.

Regarding the hyper-parameter tuning, comparably to the category classification, relatively higher learning rates (1e-3) would perform better for the feature extraction experiments, medium values (1e-4) when fine-tuning only the vision encoder and low values (5e-5) while fine-tuning both the vision and textual encoders. Furthermore, we concluded that, unlike the category classification task, the selected batch size was playing a significant role during the contrastive learning, effectively working as a means of regularisation for the model, with larger values (128 or 256) yielding improved performances ceteris paribus. Naturally, the selected batch size was constrained by the available memory and the size of the dual encoder network. However, even with larger models and while fine-tuning both encoders, we tried to define the highest possible value for the batch size (64 or 128). Concerning the number of projection layers and their dimension, an output embedding dimension of 256 was found optimal while a higher number of projection layers benefited only when both encoders were frozen or when only the vision encoder was fine-tuned. On the contrary, a single projection layer was sufficient when fine-tuning both encoders. Finally, a dropout rate of 0.3 was found as optimal, and the model did not benefit from further regularisation.

With regards to discovering the better performing transfer learning approach, the initial experimentation on the category classification dataset strongly indicated that fine-tuning both the vision and textual encoders could lead to significantly higher levels of accuracy. Larger BERT variants tended to over-fit and produce very low accuracy scores, while the smaller BERT (L2, H128, A2) yielded the best performance. The best accessed result by each transfer learning approach can be seen in table 4-15.

Table 4-15: Best performing model for each transfer-learning approach with the Category-Level Image-Text dataset. (With bold we denote the best performance)

Vision Encoder	Text Encoder	Accuracy	F1-Macro
EfficientNet B3 (frozen)	Bert L4, H512, A8 (frozen)	80%	77.5%
EfficientNet B3 (fine-tuned)	Bert L4, H512, A8 (frozen)	83.3%	78.8%
EfficientNet B4 (fine-tuned)	Bert L2, H128, A2 (fine-tuned)	**87.7%**	**85.8%**

Comparatively, our custom cross-modal vector alignment model was able to reach 87.7% in terms of accuracy, while the fully supervised multi-class classification on all 22 garment classes reached 92.24%. While the former is lower by 4.54%, we must take into consideration that this approach did not require nor utilized annotated data, and by leveraging image to text similarities it was able to reach close to the fully supervised experiments. This outcome validated the potential usefulness of the approach and therefore we proceeded with applying the same workflow to the attribute dataset. Unfortunately, similarly high accuracy scores could not be reached on the Attribute-level dataset. With the better performing model - fine-tuning both an EfficientNet-B4 and a small Bert with L2, H128, A2 - reaching 37.49% categorical accuracy, 40.47% F1 macro and 35.61% F1 micro.

Attempting to interpret our custom model's limited performance on the attribute dataset, we initially hypothesised that OpenAI's and Google's ALIGN's impressive performance on multiple benchmark datasets was only possible after being trained on approximately 400 millions and 1.8 billions image-text pairs respectively. These scales are far beyond our custom datasets that consisted of 300 thousands image-text pairs. In this respect, in order to test this hypothesis and additionally better assess the performance of our custom self-supervised contrastive model we performed additional experiments with OpenAI's CLIP which has been made publicly available with four different vision encoders, ResNet50, ResNet101, ResNet50x4 and ViT-B/32. We experimented with the three following approaches:

1. CLIP as a zero-shot classifier
2. Linear probing CLIP
3. Fine-tuning the whole CLIP model on the fashion domain

In the first approach, the CLIP model is called to classify fashion imagery in fine-grained attribute classes without being fine-tuned to the fashion domain. The second approach, similarly uses the CLIP model but as a feature extractor which features are passed to a multi-label linear classifier (a logistic regression model) in a supervised manner. In the third approach the training process from the original paper was re-created and the whole model was fine-tuned on MLZ's attribute-level dataset. The detailed results can be seen in table 4-16.

Table 4-16: Experimentation with OpenAI's CLIP on fine-grained fashion attribute detection in comparison with our custom contrastive image-text training model.

CLIP Task	Vision Encoder	F1 Micro	F1 Macro
Zero-shot	ResNet-50	17.19%	11.60%
Zero-shot	ResNet-101	18.36%	12.41%
Zero-shot	ViT-B/32	19.69%	13.14%
Linear Probing	ViT-B/32	54.9%	46.28%
Fine-tuning	ResNet-50	30.98%	23.61
Fine-tuning	ResNet-101	13.77%	9.07%
Fine-tuning	ViT-B/32	6.82%	4.3%

Custom model	Encoders	F1 Micro	F1 Macro
Fine-tuning	EfficientNet-B4 small-BERT L2, H128, A2	40.47%	35.61%

Fine-tuning OpenAI's CLIP expectedly was able to increase the model's predictive accuracy compared to zero-shot classification but it could not improve upon our custom contrastive model. Additionally, fine-tuning CLIP showed improvements only when employing the smaller computer vision network, a ResNet-50, while larger architectures, ResNet-101 and ViT-B/32, overfitted on the training data and their performance disintegrated. Our initial hypothesis that a model trained on a huge dataset would outperform our custom model was not confirmed. We believe that our collected Attribute-level dataset is relatively small containing only a few thousand samples for each attribute value, and at the same time very broad and diverse, having 1932 possible classification outcomes, and therefore it was not sufficient to train a self-supervised contrastive model.

4.3.1.2 Multi-label supervised classification

Due to the fact that the self-supervised image-text training resulted in limited performance on the attribute detection task, we proceeded by experimenting with a fully supervised framework.

4.3.1.2.1 Using the MLZ dataset

In this instance, the exact same dataset was utilized for this task, with identical training and validation sets (AD:MLZ). Since the target classes were not mutually exclusive, this was treated as a multi-label classification problem.

For the supervised framework, instead of using image-text pairs, we used the image-label pairs. The labels were transformed into multi-label binaries, and instead of CLIP's contrastive loss function, the sigmoid with the binary cross entropy (BCE) were used as the activation and loss functions respectively. In this way the multi-label task is treated as multiple binary classification tasks by the network. Additionally, we experimented with the soft-F1 loss function [44], in which the F1-score is made differentiable and as a result the network can be directly optimized on the F1 score, which is a more 'class-imbalance-aware' metric and mitigates the need for re-sampling the data and secondly it reduces the need for searching the optimal threshold during the inference phase [24, 25].

In terms of evaluating the model's performance, the Categorical and Binary Accuracy were used as well as the Macro F1 score with a threshold at 0.5. The categorical accuracy measures the exact match between the actual and predicted labels while the binary accuracy, which is equivalent to the *"1 - hamming loss"*, expresses the fraction of wrongly predicted labels to the total number of labels and is a less 'strict' metric.

On account of having limited time and computational resources we took some insights learned during the Category Classification stage as granted. We only experimented with EfficientNet B3 and B4 and did not experiment with smaller variants or with InceptionV3, Xception or ResNets. Low learning rates (1e-4 or 5e-5) were used during the fine-tuning phase and the dropout rate was always set at 0.5 and subtle image augmentations were also applied on the input images, with random horizontal flips, random rotations and zooms being performed with a factor of 0.1, the same augmentation strategy as in the contrastive experiments in order to ensure comparability. On the other hand, an addition during the attribute detection phase, was the experimentation with using pre-trained weights from self-trained EfficientNets with Noisy Student [26] instead of solely relying on weights pre-trained on ImageNet.

As the results shown in table 4-17 illustrate, utilizing the soft F1 loss function did not benefit the model's training process, resulting in significantly lower predictive accuracy when compared with models trained on the binary cross entropy. Furthermore, using the pre-trained weights from the *Noisy-Student* paper instead of ImageNet, yielded an impressive +3.86% improvement when fine-tuning an EfficientNet-B4 model with the same parameters. Attempting to fine-tune more of the EfficientNet's layers did not improve the performance any further thus leaving an EfficientNet-B4 model with *Noisy-Student* weights with 100 re-trained layers, as the best performing model.

Table 4-17: Supervised multi-label classification models for fine-grained attribute detection. (With bold we denote the best performance)

Model	Parameters	Exact Match	Binary Accuracy	F1 Macro
EfficientNet-B3	**Fine-tuned layers :** 140 **Learning rate :** 1e-4 **Batch Size :** 64 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : ImageNet **Loss function** : BCE	78.23	99.61	74.68
EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 5e-5 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : ImageNet **Loss function** : BCE	78.29	99.60	75.97
EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 1e-4 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : Noisy Student **Loss function** : BCE	**82.15**	**99.69**	**81.85**
EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 1e-4 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : ImageNet **Loss function** : Soft F1	68.27	98.86	52.39
EfficientNet-B4	**Fine-tuned layers :** 147 **Learning rate :** 5e-5 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** AdamW **Weights** : Noisy Student **Loss function** : BCE	80.63	99.69	80.56

A final attempt to further improve upon the task of Attribute Detection was the experimentation with supervised contrastive learning [27]. This approach takes advantage of contrastive training within a fully supervised framework. The model's training is divided

into two stages. First a visual encoder is pre-trained with the use of a supervised contrastive loss function with the objective of bringing image representations of the same class closer together in an embedding space and mapping them further apart from all other classes. After that, the pre-trained encoder is freezed, a classification head is added on top and is trained with the cross entropy loss. The authors showed that supervised contrastive learning was able to improve the model's predictive accuracy on ImageNet while being more 'robust' since it required less tuning of hyper-parameters. Our approach utilized the N-pair loss function modified for a multi-label problem, as offered by TensorFlow Addons [45]. However, the initial experiments with an EfficientNet-B3 model with 100 fine-tuned layers during contrastive training yielded a relatively low exact match score 53.66% and 38.32% F1 macro at best. It could be concluded that either the particular supervised contrastive framework is not optimal for a multi-label task or that wildly different hyper-parameters had to be selected. Nevertheless this question is left for future investigation.

4.3.1.2.2 Using the DeepFashion dataset

We have also performed a multi-label classification of attributes on the AD:DF dataset. The rationale behind this experiment was to explore whether the training dataset size and the relatively large number of classes were a contributing factor in determining the quality of the predictions in multi-label classification, hence we experimented with the larger dataset provided by DeepFashion. The Deepfashion dataset with the matched patterns and styles were split into training, validation and testing in 80:10:10 ratio. In much the same way as for the AD:MLZ dataset, a multi-label classification was planned for this experiment. The experiment was carried out with batch sizes and hyperparameter tuning from the experiments carried out with AD:MLZ dataset for more accurate comparison.

Due to computational limitations, the EfficientNet B4 model was considered for this experiment as it was the best performer with the multi-label classification on the AD:MLZ dataset. The input shape of (380,380,3) and a shuffle batch size of 1024 was used for shuffling the training data by chunks of 1024 observations. The data was autotuned to adapt preprocessing and prefetching dynamically to reduce the GPU and CPU idle times. Image augmentation with a factor of 0.1 was used in the training process to randomly rotate, flip and alter the contrast of the input images. The dropout rate was set at 0.5 throughout the training process. Both ImageNet and Noisy Student pre-trained weights were used in this experiment. The optimizer used was Adam with different learning rates (1e-4, 2e-5 and 5e-5), 'binary_crossentropy' was used as part of the loss function throughout this process.

Table 4-18 shows the results of multi-label classification performed with the AD:DF dataset. It is evident from the results that usage of pre-trained weights from Noisy Student has resulted in a good accuracy score overall, which was the same in case of AD:MLZ dataset. However the training process was unstable in this case and it clearly shows that the model is significantly overfitting. This can be resolved by editing the learning rate value. It can also be observed from the last model that increasing the fine-tuned layers results in affecting the performance of the model, which was the case with the experiments done on the AD:MLZ dataset. Hence, the EfficientNet-B4 model with *Noisy-Student* weights with 100 re-trained layers can be considered the best performing one on both the AD:DF and AD:MLZ dataset.

Table 4-18: Supervised multi-label classification models for fine-grained attribute detection - AD:DF dataset. (With bold we denote the best performance)

Model	Parameters	Training accuracy (in %)	Test dataset accuracy (in %)

EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 2e-5 **Batch Size :** 64 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : ImageNet **Loss function** : BCE	77.74	71.46
EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 5e-5 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : ImageNet **Loss function** : BCE	59.88	62.33
EfficientNet-B4	**Fine-tuned layers :** 100 **Learning rate :** 1e-4 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** Adam **Weights** : Noisy Student **Loss function** : BCE	**91.98**	**72.79**
EfficientNet-B4	**Fine-tuned layers :** 147 **Learning rate :** 5e-5 **Batch Size :** 32 **Dropout rate :** 0.5 **Optimizer :** AdamW **Weights** : Noisy Student **Loss function** : BCE	61.05	64.66

As a point of comparison, FashionNet, proposed in the original DeepFashion paper, was able to yield mean scores of 40.52% and 54.61% in terms of top-3 and top-5 accuracy respectively on the Attribute Detection task with 1000 attribute categories involving textures, fabrics, shapes, parts and styles [3]. More recently Li. et. al., with the use of multi-task learning, were able to increase the top-3 and top-5 accuracy scores up to 59.83% and 77.91% on the same dataset; but with cropped images around their annotated bounding boxes [33]. Re-training the best performing network from AD:MLZ on full DeepFashion for Attribute Detection showed a restricted performance; with 35.37% top-3 and 44.87% top-5 accuracy. We believe that further experiments with different hyper-parameter combinations are necessary to improve this outcome, due to the structural differences between DF and AD:MLZ; the former being more imbalanced and consisting of 1000 - relatively noisy [46] - attribute classes compared to the 109 of AD:MLZ.

4.3.2 Further work and planned improvements

After extensive experimentation with OpenAI's CLIP and our custom cross-modal image-text alignment we concluded that this direction, while being interesting from a research perspective, is not fruitful for further investigation on such a fine-grained task. On the contrary, after putting in the effort of collecting an annotated dataset, a fully supervised framework was able to reach relatively high rates of predictive accuracy. The presented experiments consisted of only two types of attributes, patterns and styles. If considered useful and necessary for the next stages, this work could be expanded to include more

types of attributes such as the garment's fit or the neckline or sleeve style for upper-body garments, etc.

For the supervised multi-label classification task, we may consider experimenting with specialised pairwise ranking techniques shown to improve a model's predictive accuracy on multi-label classification [28]. Secondly, hierarchical multi-label classification networks could be utilized, which are "capable of simultaneously optimizing local and global loss functions for discovering local hierarchical class relationships and global information from the entire class hierarchy while penalizing hierarchical violations" [29] in order to avoid logically impossible combinations on the predicted classes such as predicting more than one patterns or styles at the same time.

4.4 Merged pipeline and inference

After acquiring the best performing models for each task - object detection, category classification and the attribute detection, as shown in table 4-19, we merged them into an integrated pipeline. An image is first passed through the object detection models and the identified objects are cropped around their predicted bounding boxes. Thereafter, the cropped images depicting individual garment items were passed through the category and attribute detection models. The end result is a JSON file reporting the identified objects, their categories and attributes.

Table 4-19: The best-performing models for each task that constitute the merged pipeline

Task	**Model**	**Evaluation Metrics**	
Object Detection	Faster R-CNN	mAP : 80.6	AR@100 : 85.8
Category Classification	EfficientNet-B4	Accuracy : 92.24	F1 Macro : 92.28
Attribute Detection	EfficientNet-B4	Exact Match 82.15	F1 Macro : 81.85

Below we demonstrate a few visualised samples resulting from each step of the merged pipeline:

Step 1:

The objects are identified by the trained OD model, a sample is shown in Figure 4-9.

Figure 4-9: Prediction from Trained OD model

Step 2:

After the identification of objects, the cropping is performed on the desired objects from the predicted boundary box, as shown below. The predictions from category and attribute detection trained models are detected during this stage (Figure 4-10).

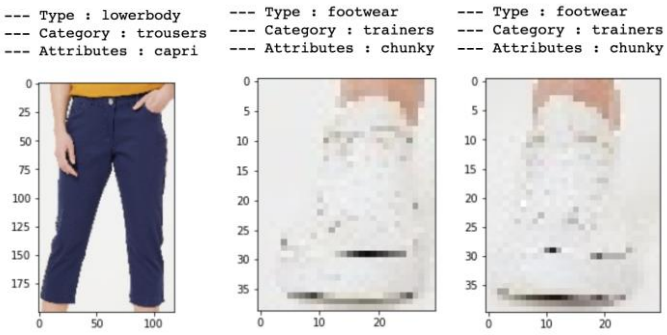

Figure 4-10: Prediction from Trained CC and AD model

Step 3:

Finally, all the predictions are integrated into a final image with a unique color code (upper body: red, lower body: blue, full body: green, footwear: orange) for identified objects with the predictions of categories and its corresponding attributes (Figure 4-11).

Figure 4-11: Final result from the merged pipeline

Below (Figure 4-12) are a few samples resulting from the merged pipeline:

Figure 4-12: Sample of pipelines results with predictions

From Figure 4-12, it can be seen that the categories and attributes were accurately detected. In the top left image the full body dress is detected with 'a-line' as attribute which is supported by the MLZ taxonomy. In the same image the attribute for footwear is projected as 'court', which comes under the 'heels' category. The majority of the pattern attributes apply to all objects, which can be seen in the bottom middle image, where floral is the attribute and dresses (full-body) as the category.

Though the majority of the results were accurate, there were few instances in which few attributes were mixed up and there were multiple predictions from the OD model. We performed heuristics on determining the threshold value of the object detection and in Figure 4-13 are few examples of before and after the threshold value was fixed to 0.99 for the OD model:

Before: **After**:

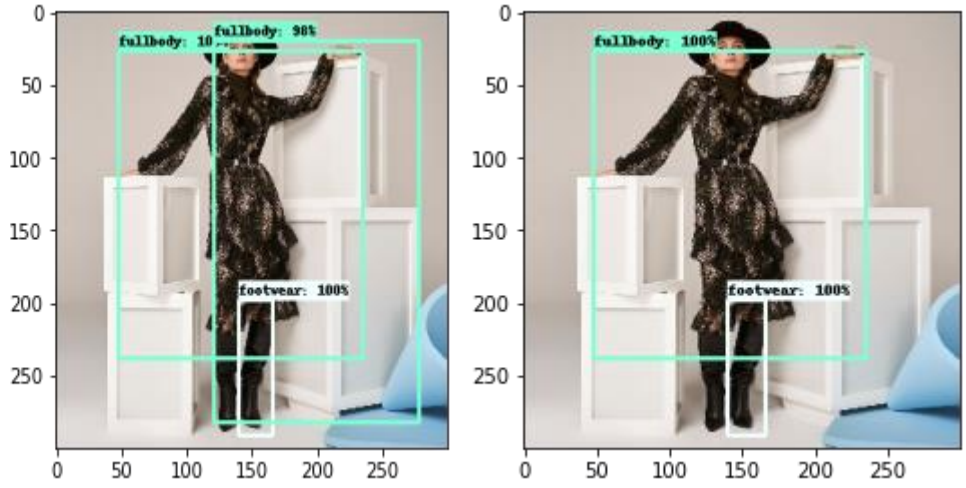

Figure 4-13: Before and After version of OD threshold value fix

4.5 Implementation

This deliverable is accompanied by a video demonstration of the pipeline architecture, that shows its performance on a novel set of images (which have been taken from the MLZ datasets).
The subsequent implementation of the D3.1 model, which is planned for use as a preliminary step in D3.2 and D3.3 will be carried out using the AWS Sagemaker service, which allows for easy deployment of an API endpoint.

The service allows for easy re-deploying of models at need, so we plan on carrying out multiple iterations of the implementation whenever updates are performed, namely when

the detection of new attributes will be possible and stronger heuristics for wrong classifications will be solidified.

5. Conclusions

The objective of the current deliverable was to extract detailed patterns from fashion imagery. Said patterns are to be used in subsequent work to detect the specific content of images, which will work as features for the forecasting of fashion trends and recommendation systems. To build systems capable of extracting patterns, we developed a hierarchical architecture that utilized deep machine learning models for object detection, category classification and fine-grained attribute detection. The employed pipeline receives fashion imagery and firstly detects the items present in the image, their location and their high-level class. Afterwards, the detected items are cropped around their predicted bounding boxes and the cropped images are passed through a category classifier and finally to a fine-grained attribute detection model. The end result is a JSON file that contains the classes of all the items present in the original image classified in mid-level categories and low-level attributes related to patterns and styles.

On the technical side, central inquiries for this work package involved the retrieval of correctly labeled datasets, handling the issue of class imbalance, identifying the best strategies for pre-processing and augmenting fashion imagery and the careful tuning of the deep learning model's hyper-parameters. For all three tasks on the hierarchical pipeline, transfer learning techniques were utilized, based on state-of-the-art deep learning models pre-trained on large-scale datasets and fine-tuned on custom datasets. We performed comparative studies between various models, while carefully tuning their hyper-parameters. Furthemore, apart from experimenting on MLZ's own datasets, we performed experiments on DeepFashion, a publicly available and widely used fashion-related dataset, for the sake of comparability and reproducibility. On the category classification and attribute detection tasks we experimented with cross-modal image to text alignment methodologies, which would alleviate the need for collecting manually labeled datasets. These experiments, while promising on the category classification task, had severe limitations on the fine-grained attribute detection task, and were outperformed by fully-supervised models.

After extensive experimentation the developed pipeline consists of the highest performing fine-tuned models for each task. More specifically, the Faster-RCNN model for the object detection with a mAP score of 80.6, the Efficient-B4 multi-class model for the category classification with 92.24 accuracy and the EfficientNet-B4 multi-label model for the attribute detection with 82.15 categorical accuracy score, were selected, all trained and evaluated on their respective MLZ dataset (see table 4-19). The extracted classes or the dense embeddings from the aforementioned models are going to be utilized, along with other fashion-related features, during the next phases of the program, namely the forecasting of fashion trends and the recommendation systems of garment items and we deem the aforementioned performances to be sufficient for these tasks. Each of the three tasks can benefit from further work in the preparation of larger and cleaner datasets for training, in the modelling and in the heuristics applied after the fact to eliminate wrong results, but due to the inherent ambiguity existing in fashion, we are satisfied with the results achieved so far and as specified in the implementation subsection, this pipeline will be implemented in an iterative process that allows for re-deployment whenever improvements are ready to be inserted.

6. References

1. B Quintino Ferreiro, J Faria, L Baía, R Gamelas Sousa, **A Unified Model with Structured Output for Fashion Images Classification,** *KDD workshop on AI for fashion*, 2018
2. A Cardoso, F Daolio, S Vargas, **Product Characterisation towards Personalisation,** *Proceedings of the 24th ACM SIGKDD International Conference on knowledge discovery & data mining*, 2018
3. Z Liu et al., **DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations,** *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2016
4. Y Ge et al., **DeepFashion2: a Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images,** *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2019
5. S Guo et al., **The iMaterialist Fashion Attribute Dataset,** *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019
6. A Yu, K Grauman, **Fine-Grained Visual Comparisons with Local Learning,** *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014
7. A Yu, K Grauman, **Semantic Jitter: Dense Supervision for Visual Comparisons via Synthetic Images**, I*EEE International Conference on Computer Vision (ICCV)*, 2017
8. X Zou et al., **FashionAI: A Hierarchical Dataset for Fashion Understanding**, *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019
9. B Thomee et al., YFCC100M: The New Data in Multimedia Research, *Communications of the ACM, 59(2),* 2016
10. K Yamaguchi at al., **Parsing Clothing in Fashion Photographs**, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2012
11. S Zheng et al., **ModaNet: A Large-Scale Street Fashion Dataset with Polygon Annotations**,*Proceedings of the 26th ACM international conference on Multimedia (MM '18),* 2018
12. B Zoph et al., **Learning data augmentation strategies for object detection**, *Lecture Notes in Computer Science, Springer,* 2020
13. N S Manikandan, K Ganesan, **Deep Learning based automatic video annotation tool for self-driving car**, preprint, 2019
14. T-Y Lin et al., **Microsoft: Common Objects in Context,** *ECCV 2014, Lecture Notes in Computer Science, 2014*
15. S Ren et al, **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,** *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017
16. F Charte et al., **Concurrence among Imbalanced Labels and Its Influence on Multilabel Resampling Algorithms**, *International Conference on Hybrid Artificial Intelligence Systems (HAIS) 2014*, 2014
17. G Hinton et. al., **Neural Networks for Machine Learning**, lecture notes
18. J Deng et al., **ImageNet: A large scale hierarchical image database**, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009
19. A Radford et al., **Learning Transferable Visual Models from Natural Language Supervision**, preprint, 2021
20. C Jia at al., Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision, preprint, 2021
21. J Devlin et al., **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HTL) 2019*, 2019

22. F Faghri et al., VSE++: Improving Visual-Semantic Embeddings with Hard Negatives, *Proceedings of the British Machine Vision Conference (BMVC)*, 2018
23. H Diao et al., **Similarity Reasoning and Filtration for Image-Text matching**, preprint, 2021
24. R Yacouby, D Axman, **Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models**, *EMNLP 2020 Workshop on Evaluation and Comparison of NLP Systems (Eval4NLP),* 2020
25. E Eban et al., **Scalable Learning of Non-Decomposable Objectives**, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017
26. Q ZIe et al., **Self-training with Noisy Student improves ImageNet classification**, *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020
27. P Koshla et al., **Supervised Contrastive Learning**, *Advances in Neural Information Processing Systems 22 (NeurIPS 2020),* 2020
28. Y Li et al., **Improving Pairwise Ranking for Multi-label Image Classification**, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2017
29. J Wehrmann et al., **Hierarchical Multi-label classification networks**, *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80:5075-5084, 2018
30. F Charte et al. **Tips, guidelines and tools for managing multi-label datasets: The mldr. datasets R package and the Cometa data repository.** Neurocomputing 289 (2018): 68-85.
31. S Sadegharmaki et al., FashionGraph: Understanding fashion data using scene graph generation, *International Conference on Pattern Recognition (ICPR),* 2020
32. K Gong, **Instance-level Human Parsing via Part Grouping Network**, *European Conference on Computer Vision (ECCV)*, 2018
33. Li, Peizhao, et al. **Two-stream multi-task network for fashion recognition**, 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.
34. Sidnev, Alexey, et al. **Deepmark++: Real-time clothing detection at the edge**, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021.
35. L TzuTa Lin. **LabelImg - graphical image annotation tool and label object bounding boxes in images**. 2018, https://github.com/tzutalin/labelImg
36. FC John. **The Canny edge detector**. 1986, https://en.wikipedia.org/wiki/Canny_edge_detector
37. **Amazon SageMaker Ground Truth**. 2021, https://aws.amazon.com/sagemaker/groundtruth/
38. J. Alexander. **Image augmentation in machine learning**. 2020, https://imgaug.readthedocs.io/en/latest/
39. R Vivek. **Training and Evaluation with TensorFlow 2**. 2020, https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_training_and_evaluation.md
40. Z Nick. **An Introduction to Evaluation Metrics for Object Detection**. 2018, https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/
41. S Yiming. **TensorFlow 1 Detection Model Zoo**. 2020, https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md
42. B Vighnesh. **TensorFlow 2 Detection Model Zoo**. 2021, https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
43. **Transfer learning and fine-tuning**. 2020, https://www.tensorflow.org/tutorials/images/transfer_learning#fine_tuning

44. M Ashref.**The Unknown Benefits of using a Soft-F1 Loss in Classification Systems**. 2019, https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d
45. **Tensorflow Addons.** 2021, https://www.tensorflow.org/addons/api_docs/python/tfa/losses/npairs_multilabel_loss
46. Shi, Mengyun, et al. "**The exploration of artificial intelligence application in fashion trend forecasting.**" Textile Research Journal (2021): 00405175211006212.