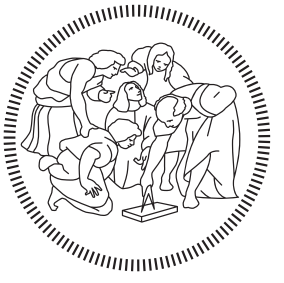
POLITECNICO DI MILANO
Master of Science in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria



# Image Tagging and Captioning for Fashion Catalogues Enrichment

Supervisor: Paolo Cremonesi
Co-supervisor: Federico Sallemi

M.Sc. Thesis by:
Umberto Pietroni, 912872

Academic Year 2019-2020

# Abstract

With the growth of e-commerce, the need to enrich online catalogues data has become an important factor for companies in all sectors. In particular, this thesis focuses on the Fashion domain, where clothing companies offer online catalogues overflowing with images which require tags and captions so that the client is able to find the wanted item. However, assigning tags and descriptions to this incredible amount of images is not an easy task, as it requires not only a lot of time but also the knowledge of a fashion expert to provide detailed and correct information. Therefore, the goal of this thesis work is to design and implement systems able to automatically generate tags and captions for clothing images leveraging Deep Learning techniques applied to the Computer Vision and Natural Language Processing fields.

We develop an approach to generate tags from the image based on Convolutional Neural Networks and we compare the obtained results with those of others works related to ours, evaluated on a public dataset. We observe that our approach reaches similar performance without leveraging landmarks, additional annotations which identify a set of points of the garment. Furthermore, to generate image captions we propose a novel approach based on GPT-2, a language model which is able to generate complex sentences from an initial text. The novelty of our approach is exploiting GPT-2 to generate text from an image and, if it is available, also textual information such as tags or metadata related to the item itself. We perform several experiments to examine how the two input modalities, visual and textual, influence our model and we compare its performance with other algorithms on industrial datasets, to verify the quality of our work. Finally, we propose a system in which we combine our approaches for tag and caption generation into a unique model capable of performing both tasks simultaneously.

# Sommario

Con la crescita dell'e-commerce, la necessità di arrichire i dati dei cataloghi online è diventato un fattore sempre più importante per le aziende di ogni settore. In particolare, questo lavoro di tesi affronta questa sfida nel dominio Fashion, in cui le aziende di abbigliamento mettono a disposizione i propri cataloghi online con numeri elevati di immagini a cui devono essere associati tag e descrizioni per fare si che il cliente sia in grado di trovare efficacemente il capo che sta cercando. Tuttavia, assegnare tag e descrizioni per questa mole incredibile di immagini non è un compito facile, siccome richiede non solo molto tempo ma anche una esperta conoscenza della moda per poter fornire informazioni dettagliate. Il nostro lavoro di tesi ha, quindi, come obiettivo quello di studiare e implementare sistemi in grado di generare tag e descrizioni per immagini di capi d'abbigliamento sfruttando tecniche di Deep Learning applicate nei campi di Computer Vision e Natural Language Processing.

Sviluppiamo un approccio per generare tag dall'immagine basato su Convolutional Neural Network e ne confrontiamo i risultati ottenuti con quelli di altri lavori relativi al nostro, valutati su un dataset pubblico. Osserviamo che il nostro approccio raggiunge simili risultati senza sfruttare i landmarks, annotazioni aggiuntive che identificano alcuni punti del capo d'abbigliamento. Inoltre, per generare descrizioni per le immagini proponiamo un nuovo approccio che sfrutta GPT-2, un modello in grado di generare frasi complesse dato un testo di partenza. L'aspetto innovativo del nostro approccio è sfruttare GPT-2 per generare frasi partendo da una immagine e, se sono disponibili, anche informazioni testuali come tag o metadata legati allo stesso oggetto. Svolgiamo diversi esperimenti per studiare come le due modalità di input, visuale e testuale, influenzano il nostro modello e ne confrontiamo le performance con altri algoritmi su dataset industriali, per verificare la qualità del nostro lavoro. Infine, presentiamo un sistema in cui combiniamo i nostri approcci per la generazione di tag e descrizioni in unico modello in grado di svolgere entrambi i task.

# Acknowledgements

First of all, I would like to thank my supervisors, Paolo Cremonesi and Federico Sallemi, for all the supports they have given me during this work. With their help and guidance I was able to improve my critical thinking and learn how to work in the research field. A special thanks goes also to ContentWise which gave me the possibility to complete a research internship despite all the challenges brought by the Covid-19 pandemic.

I would like to thank my family that always supported me during these years of university and allowed me to always do what I thought would be best for me. I would like to thank Francesca, for all the support she gave me during this time, making me smile even in the most tiring days. Finally, I want to send a big thank you to all my friends, including those in my hometown Parma, the ones I met at Politecnico in Milan and also all the good friends I made during my exchange in Brisbane.

# Contents

# Chapter 1

# Introduction

We are currently going through a digital transformation, where our actions and our way of living are heavily influenced by the technology around us, with its advantages and disadvantages. This can be noticed even more in the industry where companies are always investing more resources in new technologies, starting from e-commerce, i.e. the trade of products and services online. With the growth of e-commerce, new challenges have come up to deal with the incredible amount of data that is available online. Many of these problems are related to managing and enriching data in online catalogues, where users go in order to buy and sell products and services over the Internet. Online catalogues store millions, even billions, of items with the related images, tags and descriptions and, therefore, it's almost impossible to manually manage all of this data. Hence, the goal of this thesis is to study the current state-of-the-art methods in the research community and propose a novel approach that helps in managing web catalogues by automatically labelling and providing descriptions to items online.

In this work we are specifically focusing on the fashion domain, since it is one of the e-commerce domains where our approach is needed the most. As a matter of fact, almost every fashion company nowadays is also selling its products online, relying on catalogues filled with products images that have to be correctly tagged and captioned so that the potential client is able to find the garment which is looking for.

## 1.1 Context: Fashion Image Tagging and Captioning

The research area of this thesis work is an intersection between two vast area of research: Computer Vision and Natural Language Processing. The

former consists of all the techniques whose goal is to give machines the ability to understand, at different levels, the content of an image, a video or their surroundings just by "looking" at it. The latter, instead, focus on creating systems able to comprehend and reproduce human language for different tasks.

More precisely, we focus on Image Tagging and Captioning applied to the fashion domain in order to enrich fashion online catalogues. Even though Image Tagging and Captioning are two tasks that have been vastly explored over the last decades, the research community is still very active on these topics and novel approaches are continuously proposed. Image Tagging is a classification task where the model must be able to identify all the smallest details and patterns in an image in order to correctly discriminate between all the possible classes and generate tags. Image Captioning goal is to analyse an image and produce a syntactically correct sentence that describes the content of the image. We study and analyse the most important techniques in these vast areas of research and from these we build our own approach to tackle the inherent challenges of the fashion domain, which we present in the next chapter.

## 1.2 Scenario and Problem Statement

Every company that wants to sell its products online needs to have a complete and detailed catalogue with all the available products and their related information. This list can contain thousands of different products and, moreover, it is always changing, with new items added continuously. In the fashion domain, this happens very frequently as each season or trend correspond to new products to be inserted in the catalogue. One problem is that there has to be someone responsible for manually checking each item, looking at the image and assigning all the labels related to that item and, possibly, a description. This process can take a great amount of time depending on the number of products and how frequently this work has to be done, while, instead, the people responsible for this could spend more time on more creative and less mechanical tasks.

Furthermore, in the fashion domain, discriminating between different garments types, styles, materials and fine-grained details is not always trivial. As an example, there are more than 50 types of dresses[1], that differ depending on the silhouette, sleeves, length, fabric, hem shape and so on. Therefore it is necessary to have an expert with this knowledge capable of recognising

[1] `https://sewguide.com/types-of-dresses/`

even the smallest details which characterise a garment. Another example of the different details which characterise a fashion item is shown in figure 1.1. The amount of clothing parts, attributes and details make clothing recognition a difficult task for humans and machines alike.

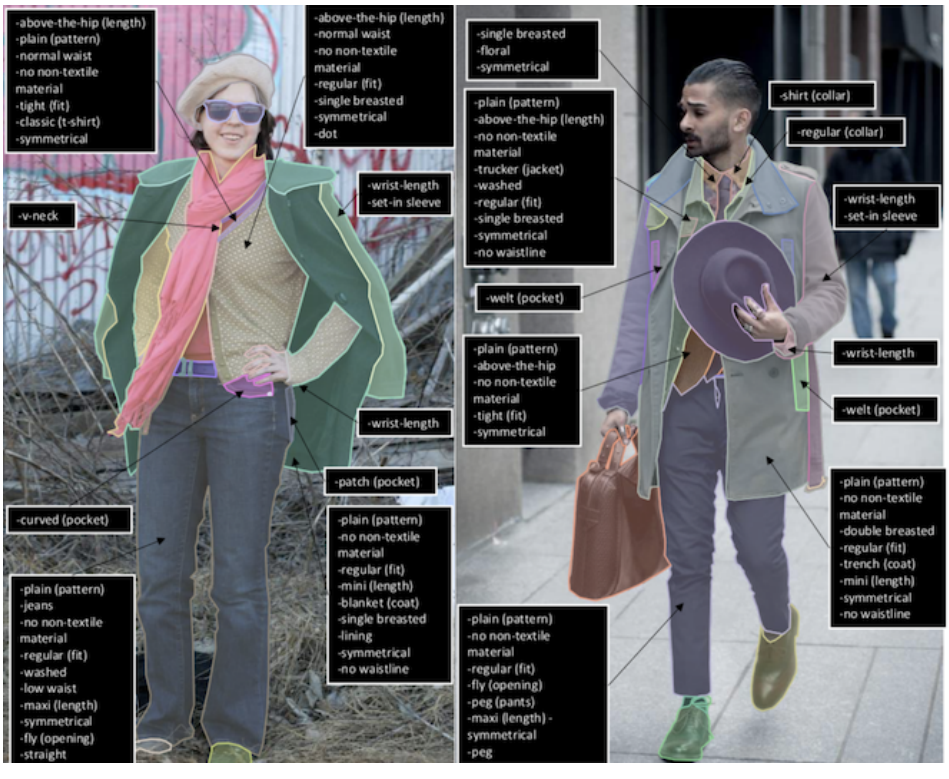



*Figure 1.1: Example of fine-grained details for fashion items. Image taken from iMaterialist (Fashion) 2020 Kaggle Competition*[2]

In summary, this problem is an important challenge that has to be faced by fashion companies for both time and expert knowledge needs and having a system able to assist manual work or even replace it would certainly be an asset.

## 1.3 Contributions

With the knowledge acquired from the analysis of the state of the art related to Image Tagging and Image Captioning, we propose novel approaches for these two tasks applied in the fashion domain and we test and compare them with other baselines on several datasets, both public and private. The

[2] `https://www.kaggle.com/c/imaterialist-fashion-2020-fgvc7`

task of Fashion Image Captioning, in particular, has not received a lot of attention so far from the research community and, therefore, with our work we also hope to help the research in this direction.

First, regarding the Fashion Image Tagging task, we compare our approach with state-of-the-art algorithms evaluated on a public large-scale dataset for fashion called DeepFashion [LLQ+16]. We observe that the current state-of-the-art approach [LL18] relies on a landmark attention branch to improve the performances in the Category and Attribute prediction task, where two different type of tags have to generated. Given the fact that landmarks, which correspond to a set of key-points on the clothes structure, are a type of annotation that is rarely available in real-word datasets, we prefer to build a model that doesn't exploit this extra information. We then compare the results of our model with the current state-of-the-art and, as explained in chapter 6.1.5, we show that our approach slightly outperforms the baseline with landmark attention, suggesting that using this type of mechanism based on the additional landmarks annotations doesn't seem to be necessary to improve the performance for the generation of fashion tags.

For the Fashion Captioning task, we propose a novel approach based on GPT-2 language model [RWC+19] to generate captions for an image. GPT-2 architecture has surprised many for its ability to generate coherent and complex text but, to the best of our knowledge, is yet to be used in the context of generating text given features extracted from an image and, therefore, with our work we want to provide a contribution to this research question. Furthermore, we don't limit our model to leverage only visual features but we consider the possibility of improving the quality of the generated caption by exploiting also additional textual information associated to the image, if available. This is particularly useful when generating captions for catalogues which already have some information available as tags or metadata that can serve as additional features to help the model understanding the image content. We present a performance study comparing our approach with other baselines and we also perform a multimodal analysis by examining whether our model relies on one input modality more than the other in different settings, with both quantitative analysis on evaluation metrics and qualitative ones by observing the generated captions and visualizing the attention layers inside our GPT-2 model.

Finally, we present a model capable of generating simultaneously tags and caption by combining our approaches for Fashion Tagging and Captioning into a unique system. We carry out a performance study also for this approach by comparing its performance with those of our models which perform, separately, Tagging and Captioning.

## 1.4 Thesis Structure

The rest of the thesis is structured as follows:

- Chapter 2 introduces the state of the art techniques on which this thesis work is based and present them in depth to provide the reader with the essential knowledge.
- Chapter 3 consists of the description and analysis of the datasets we use to train and evaluate our algorithms, highlighting similarities and differences between them.
- Chapter 4 outlines works by other authors which tackle similar problems, both in Image Tagging and Image Captioning tasks.
- in Chapter 5 we introduce our models architecture and design decisions that lead us to implement our solutions.
- Chapter 6 presents the results obtained in the different tasks and compares them with baseline works.
- Chapter 7 summarizes the contributions, limitations of our work and proposes future research directions

# Chapter 2

# Background

In this chapter we present the background related to the two challenges we face in this thesis work: Image Tagging and Image Captioning. For the former we present the most important techniques and algorithms which involve the generation of tags for an image, at different levels of granularity from the entire image categorization to object detection. For the latter we list the most relevant architectures in the Natural Language Processing field and their application to Image Captioning.

## 2.1 Image Tagging

Image Tagging is a big part of Computer Vision and it is one of the core research tasks where the community is always proposing new and improved solutions. This field includes all the techniques that focus on generating labels for an input image.

Depending on the goal and on the type of labels, we can identify different tasks that relate to this broad area of research. If the focus is on generating a single label for the entire image, choosing from a finite set of labels, then this task is generally referred to as Multi-Class classification. If, instead, multiple labels can be used to describe a single image then we are usually talking about a Multi-Label classification problem. In this thesis we are mainly focusing on this two tasks of Image Tagging since we want to generate multiple tags for a fashion image in order to enrich a catalog with data.

There are other problems that can be considered as subsets of Image Tagging. Object Detection consists of identifying different object in an image and also marking their position with a bounding box. Another example is Image Segmentation where the goal is to assign a label to each pixel in an image, one of the most fine-grained level of classification.

We also provide a brief description of Convolutional Network Networks for their contribution in moving forward the research in Computer Vision as effective visual feature extractors and since they are a fundamental component in all our approaches.

### 2.1.1 Convolutional Neural Networks

A Convolutional Neural Network is a Deep Learning algorithm made specifically for learning autonomously how to extract useful features from an image while reducing the image into a form that is easier to process, making it scalable for big datasets. It is a trainable end-to-end architecture through back-propagation, as normal feed forward neural network, with the addition of convolutional and pooling layers.

Inside a convolutional layer, a convolution is performed between the input image and a filter (or kernel), typically a square matrix with values around 3 or 5 pixels for width and height and same depth as the input. The filter slides across the input moving from left to right and from top to bottom with a certain stride value, and, at each step, a dot product is computed between the filter weights and the values in the portion of the image over which the filter is hovering.

The reason why these layers are so effective when it comes to image processing is that they can capture the spatial dependencies between the pixels in an image and the filters are trained to search for relevant features. Typically, the first convolutionals layer in a CNN architecture are responsible for capturing low-level features such as edges, color, gradient orientation. By adding more layers, the architecture learns also features that are more high-level and obtain a general understanding of the image.

Another type of layer that is commonly used in a ConvNet is the pooling layer. The goal of a pooling layer is to reduce the spatial dimensionality of the feature to decrease the computational effort required to process them.

After convolutional and pooling layers, a convolutional block is completed by a non-linear activation function such as ReLU that is responsible for the neurons activation as in the normal feed forward neural networks. Thanks to these non-linear operations the neural network is able to model complex non-linear functions that otherwise, as a linear regression model, would not be capable of.

In a image classification setting, for example, the final output of CNN is flattened into 1-dimensional array that serve as input to a fully connected layer. The last layer gives as output a vector with size equal to the number of classes which, after passing through a softmax or sigmoid activation

function, returns the probabilities of all the classes.

### ResNet

ResNet, short for Residual Networks [HZRS16], is one of the most used backbone architectures in Computer Vision tasks and the winner architecture of the ImageNet[1] challenge in 2015 with a 3.57% error rate.

A common thought in the Deep Learning field is that deeper architectures always achieve better results than their shallow counterpart. However, this is not always the case, as He et al. [HZRS16] prove by plotting the training and test error of a 20-layer CNN versus a 56-layer CNN (figure 2.1 ) where the deeper network has both higher training and test errors, therefore the worse results don't seem to be due to an overfitting problem which is evident when the training error is low but the test error is high.

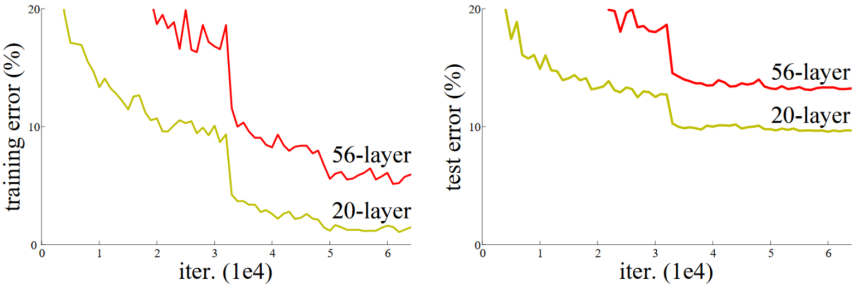



*Figure 2.1: Training error (left) and test error (right) with 20-layer and 56-layer networks. The deeper network has higher training error, and thus test error. Image taken from [HZRS16]*

There could be different reasons for the failure of 56-layer CNN: optimization function, initialization of the network, vanishing/exploding gradient problem.

He et al. [HZRS16] propose a solution that makes it possible to train deeper networks that obtain better results: the Residual Block. As shown in figure 2.2, the residual block is composed by a normal layer and a identity mapping that does not have any parameters and is just there to add the output from the previous layer to the layer ahead. ResNet is a CNN with the addition of residual blocks that help in the training of deeper models. According to [Dwi19], the Residual Blocks work because they mitigate the problem of vanishing gradient by allowing this alternate shortcut path for

[1] `http://www.image-net.org/challenges/LSVRC/`

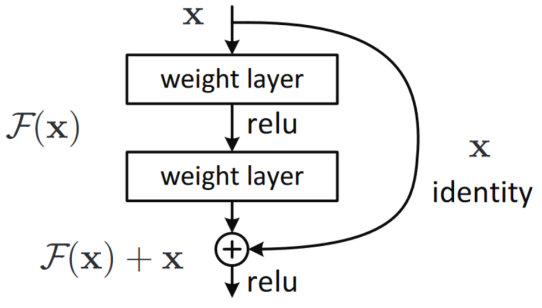



*Figure 2.2: Residual Learning: a building block. Image taken from [HZRS16]*

gradient to flow through and also because they allow the model to learn an identity function which ensures that the higher layer will perform at least as good as the lower layer, and not worse. In this work we use ResNet as the backbone of our architectures.

### 2.1.2 Object Detection

Object Detection represents a group of problems in which the task is to identify the type of objects present in an image and also localize their position, typically with bounding boxes (e.g. defined by a point, width, and height). Therefore, it includes both image classification and object localization tasks.

One of the most successful application of Deep Learning to the problem of Object Detection is Faster R-CNN [RHGS15]. This model is composed by three different components: a Feature Extractor,a Region Proposal Network and a Classifier. The Region Proposal Network (RPN) analyse the output feature map generated by the feature extractor and it decides whether an object is present in the input image and return its location and its estimate size as Region of Interest. This is done by using anchors, i.e. rectangular boxes of different dimensions, placed on the output features and the RPN check whether the anchors contain objects while refining the anchors coordinates to return bounding boxes. The regions proposed by RPN are then resized to a fixed size with a RoI pooling layer. Finally, all the feature maps are flattened and fed as input to fully connected layers that have two outputs. The first is the softmax classification layer that finds the probabilities for the object in the region. The second is the Bounding Box regressor, which outputs the bounding box coordinates for each object class. Figure 2.3 illustrate the architecture of Faster R-CNN for object detection and recognition applications.

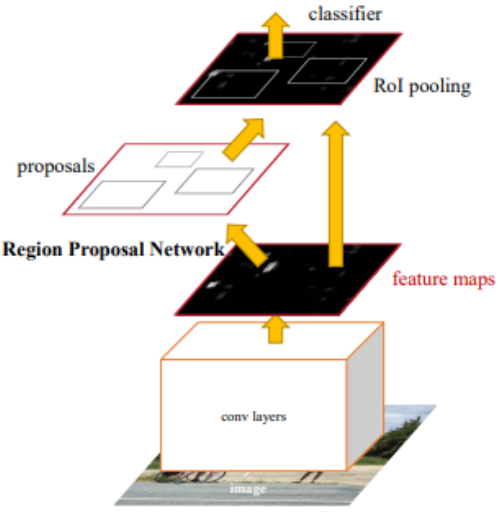



*Figure 2.3: Faster R-CNN Architecture. [RHGS15]*

### 2.1.3 Image Segmentation

Among all the tasks than can be considered as Image Tagging, Image Segmentation is the one that operates at the most fine-grained level. As a matter of fact, the goal of Image Segmentation is to assign a label to each pixel in an image such that pixels with same labels share certain characteristics. Pixels clustered in the same regions can identify different surfaces, objects or part of objects. Semantic Segmentation corresponds to the task of finding objects of different classes in the picture, without discriminating different entities of the same class. This last part is done in the Instance Segmentation task which, in fact, is typically more challenging than semantic. Semantic Segmentation is also used in the Fashion Industry to extract clothing items from an image to provide similar suggestions from retail shops [Raj19].

One of the state-of-the-art image segmentation techniques is Mask R-CNN [HGDG17]. Mask R-CNN is built on top of a Faster R-CNN and, therefore, in addition to the class label and the bounding boxes of each object in the image, it will also return the object segmentation masks. It extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition and it outputs a binary mask for each RoI.

## 2.2 Image Captioning

In this thesis work we are exploring different ways of enriching fashion online catalogues, not only by generating tags from an image but also a correct caption, capable of describing an item in a syntactically and semantically correct way. The task of making machines able to describe the content of a picture is often referred to as Image Captioning and it is also one of the most studied problems in the Computer Vision research community. In this section we are going to provide an overview of the current background of Image Captioning, starting from Natural Language Processing techniques which are at the core of many solutions.

### 2.2.1 Language Modeling

Natural Language Processing (NLP) is defined as the automatic manipulation of natural language, like speech and text, by software. It's a vast area of research, with different application such as speech recognition, language translation and sentiment analysis, with the goal of making machines capable of understanding language and also generating it. A machine, however, operates on the basis of mathematical rules and therefore it needs to rely on a mathematical model to perform a kind of reasoning that a human would do with ease. The creation of this mathematical model is called Language Modeling.

Language Modeling (LM) is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence[2]. It includes different techniques to translate a text into a machine-understandable format and analyse it. Language is usually very complex to understand even for a human and all languages have their unique words and rules that makes them different from one another. However, each word in a sentence is somehow related to the other words in the sentence that are frequently used with it. As an example, a simple statistical model could analyse how each word is frequently used after another word and, with this, compute the probability of each word and build a language model. However, this model is not capable of learning all the intrinsic nature of a language so neural models based on deep learning are used to learn the rules of a grammar, all without explicit teaching.

[2] `https://searchenterpriseai.techtarget.com/definition/language-modeling`

### 2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a particular type of Neural Network that are made specifically for modelling sequence data and, since sentences are just sequences of words where the order matters, they are used also for language modeling. The main difference with the simple statistical model described in the previous section, is that RNNs don't simply compute the probability $P(X|Y)$ of a word X given the past word Y, but they also have a memory that stores information about previous words in a hidden state vector. RNN work sequentially as we feed it one word as input at each step, it returns the output and the modified hidden state which will be used at the next step. One problem of traditional Recurrent Neural Networks is that they suffer from short-term memory meaning that, if a sentence becomes quite long, the network will tend to forget the first words in the sequence and it may leave out important information. There are two specifics types of Recurrent Neural Networks that are meant to mitigate this problem: GRU and LSTM.

LSTM and GRU works almost identically as traditional RNNs, but they have additional layers called Gate layers that are capable of learning long-term dependencies. These gates can learn what information to add or remove to the hidden state so that it can pass the relevant information down the long chain of sequences.

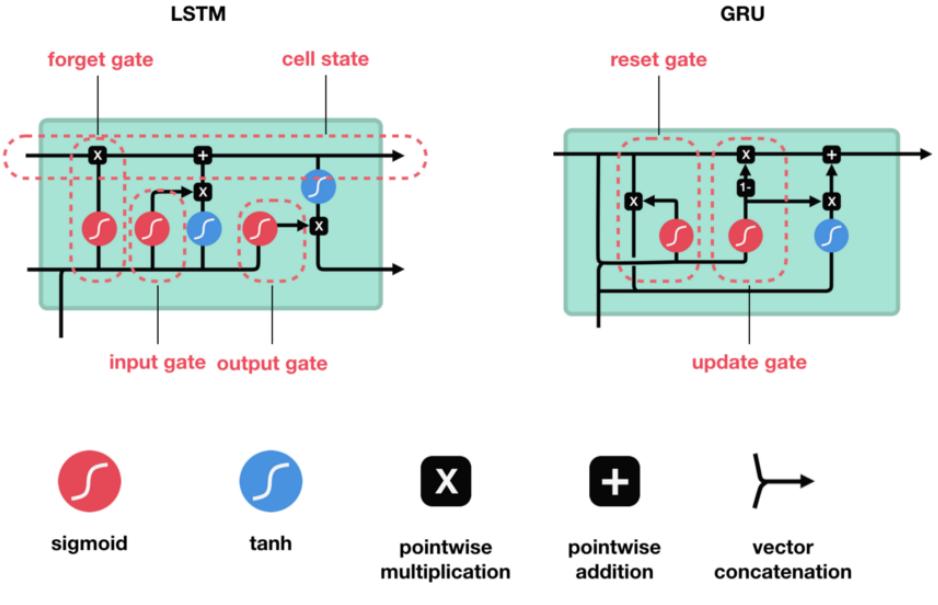



*Figure 2.4: LSTM and GRU cells internal architecture. Image taken from [Phi20]*

In figure 2.4, the internal structures of LSTM's and GRU's cells are showed. LSTM cell is characterised by three gates: forget gate, input gate and output gate. The first one decides what information should be thrown away or kept, the input gate is responsible for updating the cell state and the last one is used to compute how the next hidden state should be. The cell state and the hidden state are the final output which is fed as input in the next time step. GRU's structure is similar to LSTM's but it removes the cell state and use the hidden state to transfer information. It also only has two gates, a reset gate and update gate. GRUs are faster to train than LSTMs but typically they are less used than their counterparts. These networks, however, don't completely solve the long-term dependencies problem and they are quite long to train for their recurrent structure that prevents parallel computation.

### 2.2.3 Transformers

The Transformer [VSP+17] architecture represents one major breakthrough in language modeling for its ability to overcome the computational limitations of its predecessors and also to achieve better performances. Transformer relies on self-attention blocks, without any recurrent unit, making it possible to analyse an input sentence and generate a new one in a parallel way and not sequentially as for RNN. This accelerates greatly the computation and, thanks to a self-attention mechanism, the model is able to attend to all the input words at any time and learn which are the most important to rely on at each step.

As an example, in figure 2.5 it is possible to visualize at an high level what the attention mechanism is really about. Here the focus is on the attention related to the word "it" in the sentence "The animal didn't cross the street because it was too tired". The attention highlights more the part of the sentence close to the word "animal", to which "it" refers to, while others are less important and, therefore, their attention score is lower.

The model is composed by two components: a stack of encoders and one of decoders having the same number of layers. The encoders have all the same structure composed of two blocks: a Self-Attention block and a Feed-Forward Neural Network. Decoders are composed by the same blocks, interleaved by a block of Encoder-Decoder Attention that helps focusing on relevant parts of the encoder outputs. The model architecture is shown in figure 2.6 where the module of the left is the Encoder stack and the one on the right is the stack of decoders.

Firstly, before being fed to the encoder, each word in the input sentence

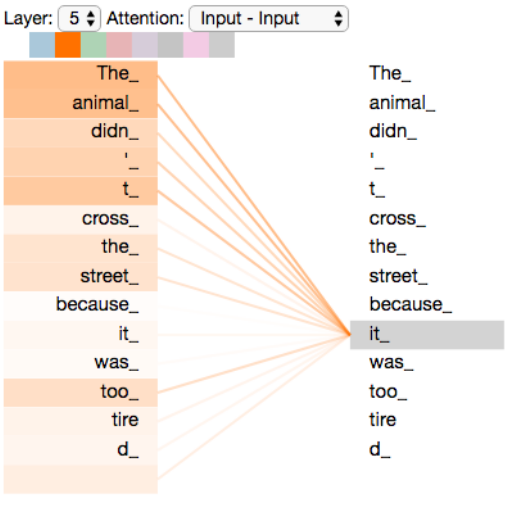



*Figure 2.5: Visualization of the Attention Mechanism. Image taken from [Ala18]*

is converted into an embedding vector of a fixed size. This is done thanks to an embedding matrix, learned during training, with rows equals to the vocabulary size and columns number as the embedding size. This new input vector flows through the bottom encoder and its self-attention layer. Here, for each input word, query, key and value vectors are obtained from learnable matrices and they are used to compute the attention scores. In fact, for each word, its corresponding query vector is multiplied with the key vectors of the other words to obtain attention scores. These scores encode how much each word is related to the other words in the sentence and are used to obtain the best encoding for each word given the context. The attention scores are passed through a softmax layer which squeezes them to values between 0 and 1 and that all sum to 1. Each of these scores is multiplied with its corresponding value vector which are then summed together to obtain the final embedding. The particularity of the Transformer is that this process happens concurrently for each word in input, so each word flows through its path but, at the same time, the computation takes into consideration all the other words thanks to the self-attention.

The self-attention mechanism just explained describes what happens in a single attention head but, actually, the complete process is called Multi-Head Attention where there are different query, key and value representation for each attention head. This extension expand the model ability to focus on different positions while, with only one head, it could happen that the attention focuses only on the same word. Another reason is that by learning

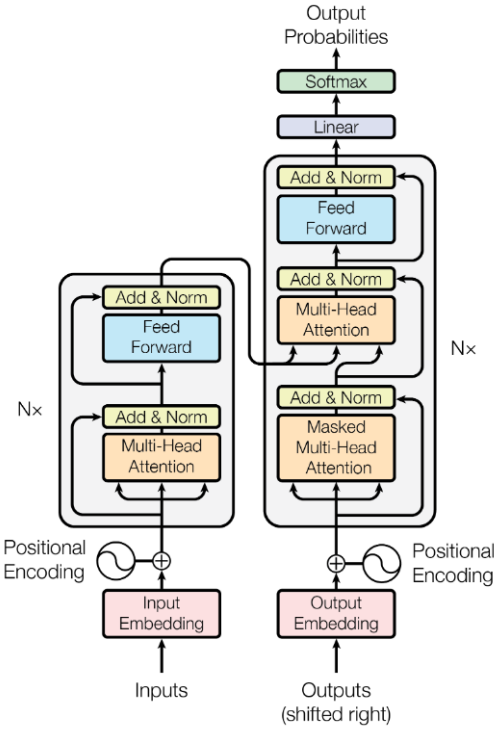



*Figure 2.6: Transformer Model Architecture. Image taken from [VSP+17]*

different query, key and value matrices the model also projects the input embedding into different representation subspaces.

Another important feature of Transformer is the fact that they address the problem of considering the word order in the sentence, a relevant information in a sentence translation, by using positional encodings, vectors which follow a particular pattern that the model learn and are added to the input embedding.

The decoding phase consist in generating each output word step by step. After the encoder processed the input sentence, its output is converted into keys and value matrices which are fed to the decoder in the Encoder-Decoder attention layer that helps the decoder focus in the more relevant part of the input sequence. The bottom of the decoder stack is fed with the target sequence which is encoded in the same way as for the encoders input sequence with the embedding matrix and positional encodings. The attention layers in the decoder works in a slightly different way than the encoder since in the decoder self-attention, each token can attend only those preceding it and not those that follows it in the sentence. This is done by masking future positions so that a current position can only be influenced by past positions

and not future ones.

The final output embedding of the decoder is passed through a linear layer that maps it into a vector of size equal to the vocabulary length. Here each score, after a softmax layer, represents the probability for each word in the vocabulary of being decoded at the current step.

### 2.2.4 GPT-2

GPT-2 [RWC+19] is a large-scale transformer-based language model created by OpenAI[3] and trained on a dataset composed by 40GB of Internet text. Even if it is trained to simply predict the next word given all the previous words within some text, the diversity of the training dataset gives this model the capability to perform well also in other datasets and other tasks. As a matter of fact, GPT-2 outperforms on language tasks like question answering, reading comprehension, summarization, and translation other language models trained on these specific domains. Therefore GPT-2 implicitly learns these tasks from the raw text, using no task-specific training data.

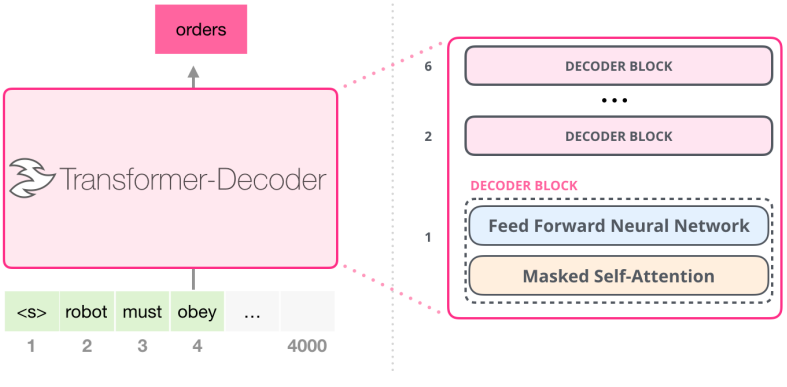



*Figure 2.7: Decoder-Only block in GPT-2 stack of decoders. [Ala19]*

GPT-2 architecture is transformer-like but it's different from the traditional model as it exploits only a stack of decoders, without the stack of encoders that completes the Transformer. The Decoder-Only block of GPT-2 are composed by a layer of Masked Self-Attention and a Feed Forward Neural Network. The Masked Self-Attention is necessary for the next token computation as it allows the model to attend to only to previous words in the sentence, without peaking at tokens to its right. This is what makes GPT-2 an auto-regressive model: predicting each new token on the basis of previous tokens, which will then be added to the sequence of inputs for the

[3] `https://openai.com/`

model in its next step.

The authors have released several variant of GPT-2 with different sizes going from 117 million parameters to the biggest one of 1.5 billions. More recently, OpenAI created GPT-3 [BMR+20], an even bigger language model with 175 billion parameters. The model algorithm and weights, however, are not disclosed yet at the time of writing.

### 2.2.5 NLP and Image Captioning

After the undeniable success of techniques such as Recurrent Neural Networks and, more recently, Transformers in the Natural Language Processing tasks, these models are also starting to be at the core of many solutions for Image Captioning where the goal is to generate a description for an image. However, the main difference is that, in this case, the task is not sequence to sequence, i.e. generating a sentence from another one, but image to sequence and, therefore, more challenging as they belong to different representation spaces. As a matter of fact, sentences correspond to a sequence of words while images are collections of pixels with a spatial distribution. Furthermore, an image descriptor needs to be able to recognise what kind of object or details are present in an image and describe them in a natural and cohesive way.

Before examining a bit more in depth how Deep Learning NLP models are applied to Image Captioning, it is worth saying that Image Captioning models without Deep Learning are mainly based on template-based methods and search-based approaches. The former firstly align each sentence fragments (e.g., subject, verb, object) with detected words from image content and then generate the sentence with pre-defined language templates. The generated sentences have a correct syntactical structure but this approach is not very flexible as it depends on the templates of sentences and their goodness. Search-based approaches find a sentence for an image by selecting the most semantically similar sentences from a sentence pool or directly copying sentences from other visually similar images. Hence, the generated captions will be most likely syntactically correct and human-like, but these models lack the ability to explore all the possible ways to describe an image and to learn objects representations and their relations.

Language modeling architectures combined with the feature extraction ability of Convolutional Neural Networks are, instead, able to learn the probability distribution in the common space of visual and textual content and exploit it to generate new sentences with a flexible syntactical structure. Typically, these architectures are composed by an encoder which is a CNN

that extract features from the image, and a decoder represented by a RNN that takes the image embedding as input and, starting from it, generate sequentially the sentence tokens. An example of this model is shown in figure 2.8. One problem with this approach is that the model fully attends to the image embedding only at the beginning when it is fed as input and then it starts to lose track of that information as it decodes the next tokens, same as in the long-term dependencies problem which affects RNNs. A solution to this problem is presented in [XBK+15], where, thanks to a visual attention mechanism, the model is able to attend to the image embedding at each decoding step and, therefore, it learns to focus on the most relevant part of the image related to the current token.

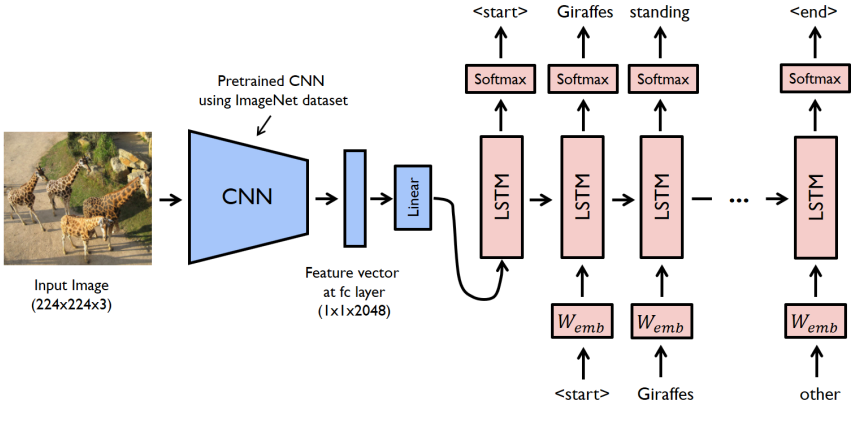



*Figure 2.8: CNN+RNN Image Captioning Architecture. Image taken from [Yun]*

Since Transformers have demonstrated their effectiveness in many NLP tasks, the research community is also proposing new applications of this architecture in other fields, including Computer Vision. In fact, some approaches [SZC+19] [LHL+19] [KBFT19] use Transformer model as the backbone, and extends it to take both visual and linguistic embedded features as input. In [SZC+19], each element of the input is either of a word from the input sentence, or a region-of-interest (RoI) extracted using a Faster R-CNN on an input image and the goal is to solve Visual Question Answering task. In [LHL+19] a similar approach is used in a unified model whose goal is to solve not only VQA task but also Image Captioning with a single model. [KBFT19] it's an interesting approach as it combines both text embedding and image embedding and feed them as input to a BERT [DCLT19] architecture, based on Transformers, that has to classify both images and text.

## 2.3 Summary

In this chapter we present the background related to our work, with the most relevant techniques for Image Tagging and Image Captioning tasks. The most important references to better understand our approach are:

- Convolutional Neural Network that we exploit to extract visual features from fashion images.
- LSTM architectures which are employed as caption generator in Image Captioning baselines that we compare to.
- GPT-2 language model which is a fundamental component in our novel approach for Fashion Image Captioning.

# Chapter 3

# Datasets

In this chapter we present the datasets we use to train and evaluate our models on the Fashion Tagging and Captioning tasks. In particular, we use one public dataset and two private datasets which we obtain thanks to a collaboration with our industrial partners. We first describe a large-scale publicly available fashion dataset called DeepFashion which is one of the few large datasets on fashion rich of annotation such as classes, attributes, bounding boxes and landmarks and it allows us to compare our models with related work evaluated on this dataset. Our other datasets have less items but have also caption annotation, making them very useful for the Fashion Captioning task. Our datasets and the related tasks for which we use them are summarised in table 3.1.

| **Datasets and Tasks** | | |
|---|---|---|
| Dataset | Tagging | Captioning |
| DeepFashion | X | |
| Industrial Dataset 1 | X | X |
| Industrial Dataset 2 | | X |

*Table 3.1: Tasks and related datasets*

## 3.1 DeepFashion

DeepFashion [LLQ+16] is a large-scale fashion dataset with comprehensive annotations which is used for several task related to fashion as Category and Attribute Prediction, In-Shop Retrieval, Consumer-To-Shop Retrieval, Fashion Landmark Detection and Fashion Synthesis. We focus on the Category and Attribute Prediction as our goal is to generate tags for a clothing

item, i.e. generate a list of labels which characterise the garment class and more fine-grained details. This dataset is available online[1] and it contains 289,222 fashion images ranging from high quality shop images to unconstrained customer photos collected from fashion websites and image search engines. We follow the same setting of [LLQ+16] which splits the dataset into 209,222 for training, 40,000 for validation and the remaining 40,000 for the evaluation.

This dataset is annotated with:

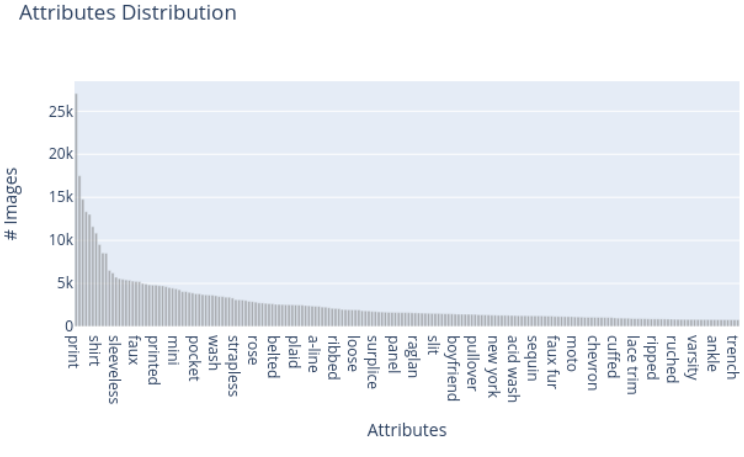
- 50 different classes which are a set of mutually exclusive labels where each image can be associated with only one class. Classes represent the high-level type of garment which can be, for example, "Sweater", "Blouse", "Shorts" or "Jeans".
- 1000 attributes which corresponds to different characteristic and details of clothing items and are, therefore, more fine-grained than classes and are not mutually exclusive, i.e. different attributes can be assigned to a single item. There is an average of 3.32 attributes per item in the dataset. Attributes are further distinguished into five groups: Texture, Fabric, Shape, Part and Style.
- clothing landmarks which correspond to a set of key-points on the clothes structure as left/collar end, left/right sleeve end, left/right waistline and left/right hem.
- one bounding box per image which identifies the clothing item in the picture

Figure 3.1a shows that the class distribution has a long-tailed shape with thousand of items in the top classes and few hundreds in the last. The top-10 represented classes in the dataset are, in order: Dresses, Tee, Blouse, Shorts, Tank, Skirt, Cardigan, Sweater, Jacket and Top.

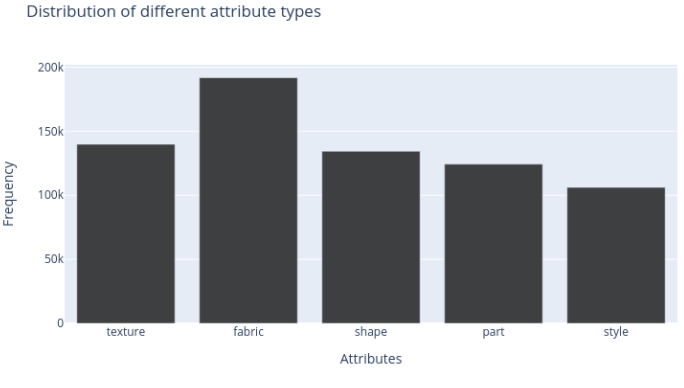
Attributes have also a highly unbalanced distribution as it can be noticed in figure 3.1b, meaning that there are attributes which appear a lot more frequently than others. As a matter of fact, the most frequent attributes in the training set are assigned to thousands of images while the least frequents are associated with less than fifty items. Another significant statistic is that more than half of the total number of attributes annotated in the training set belong to only 63 attributes types of the 1000. The top 10 most frequent attributes are print, floral, lace, knit, sleeve, maxi, shirt, denim, striped and chiffon.

[1] `http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion/AttributePrediction.html`


*(a)*




*(b)*




*(c)*

*Figure 3.1: DeepFashion Class and Attributes Distributions*

Since attributes can be distinguished into five types (Texture, Fabric, Shape, Part and Style), we also analyse how the attributes are distributed between the different types. Texture attributes consist of labels related to visual pattern and design in the clothing item such a floral design, stripes, dots or other prints. Fabric describes the material of the garment which can be for example lace or leather. Shape refer to the fit and shape while Part identify specific details in the item such as a pocket, sleeves or V-neck. Finally, Style attributes are different from the other types as they don't refer to a particular visual feature but rather to a concept related to the garment such as elegant, summer or shopping. Figure 3.1c shows that attributes type are quite balanced, apart from the fabric type which is more represented than others and style attributes are less frequent.

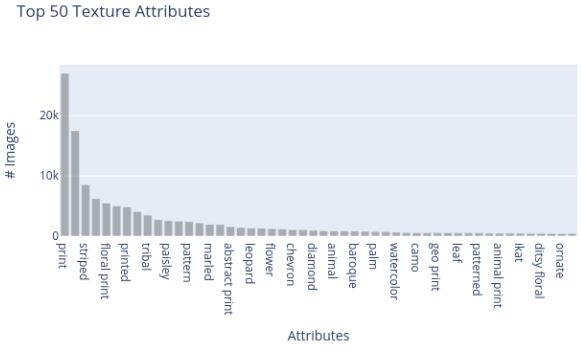
Furthermore, we observe also the attributes frequency inside each attributes group. A common pattern in Texture (figure 3.2a), Shape (figure 3.2c) and Part (figure 3.2d) groups is that one or two attributes stand out in terms of appearances in each group, with a number of images that is more than double the others. Fabric (figure 3.2b) and Style (figure 3.2e ) attributes have a more gradual descent in the form of a long-tail distribution. The Style attributes, in particular, don't have any attribute which exceed 10000 appearances as other types but, on the other hand, the appearances seem to be more evenly distributed between the attributes.

## 3.2 Industrial Dataset 1

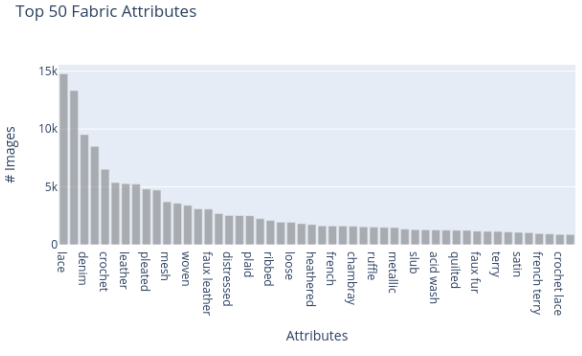
Thanks to the collaboration with our industrial partner H&M, we are also able to work with a new industrial dataset that we cannot share, but we can provide a description of its statistics and its main features.

We start creating the correct dataset for our needs from a complete catalogue of items in different languages and types also unrelated to fashion such as homeware and personal care. From this we select only those with English labels and from Clothing, Footwear and Accessories super groups and we remove duplicates. We obtain a dataset of clothing items with different features such as the main category, supercategories, colors, age groups, concepts, sizes and a description. We restrict the main categories to 27 classes and the remaining features are all grouped into a Attributes set composed by 255 labels. We keep the description as a separate feature and we don't use it to enrich the labels set.

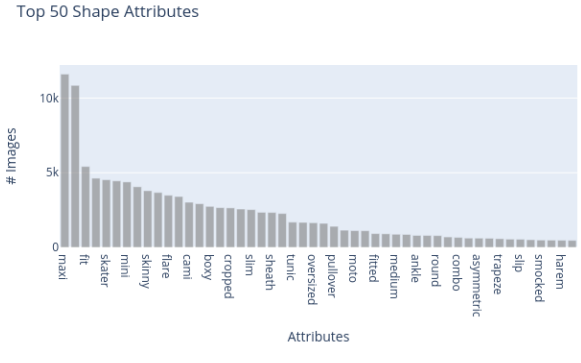
After all these steps, the final datasets is composed by 77254 images which we split into 54387 for training, 13597 for validation and 9270 as test set. The dataset is annotated with:



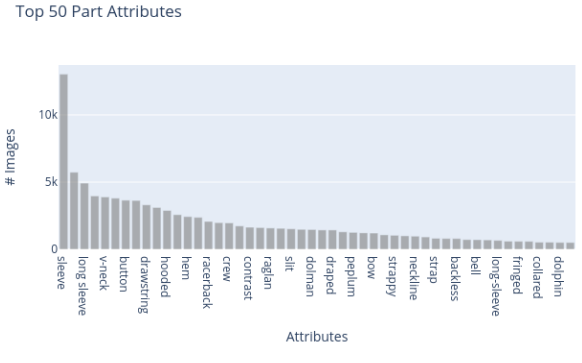

*(a) Texture*



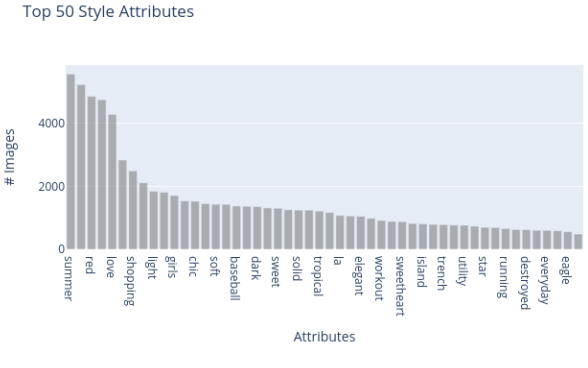

*(b) Fabric*




*(c) Shape*




*(d) Part*




*(e) Style*

*Figure 3.2: DeepFashion Attribute Types Distributions*

- 27 classes which are mutually exclusive.

- 255 attributes composed by more specific clothing categories, colors, age groups and concepts. Several attributes may be assigned to an item and there are on average 5.41 attributes per item.

- A description in English for each item with an average of 23.36 words per description. Caption examples can be seen in table 3.2.

Given the fact that this dataset is annotated with tags and descriptions, we perform experiment on it in both the Fashion Tagging and Fashion Captioning tasks.

As it is shown in figure 3.3a, classes are more evenly distributed that the DeepFashion dataset classes, however, there are classes which are much more frequent than others such as Tops, Accessories, Dresses, Trousers with more than 5000 counts. Attributes have a long-tail distribution, as it can be seen in figure 3.3b, with ladies, kids, adult, black and blue as top-5 attributes.

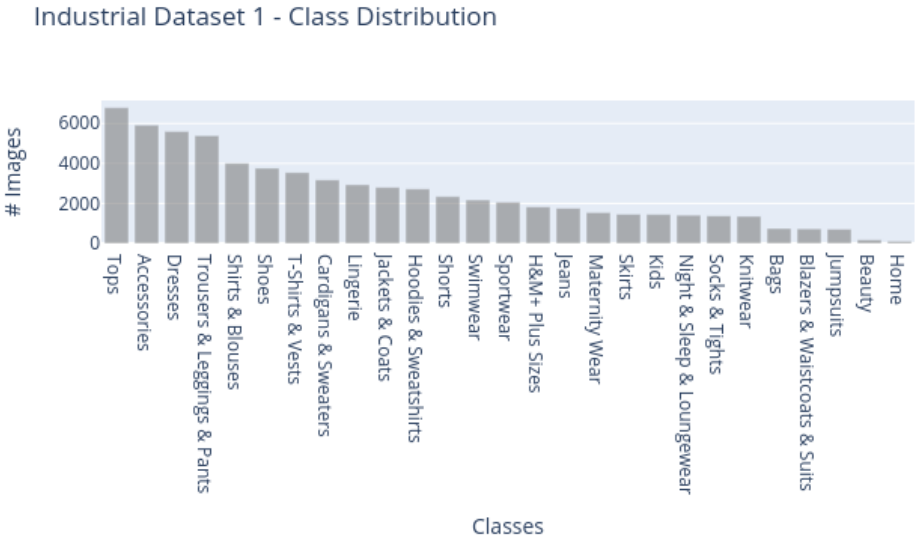
| **Class** | **Caption** |
|---|---|
| Dresses | Short dress in washed woven viscose fabric with a round neck short sleeves with sewn cuffs and a rounded hem. Slightly longer at back. |
| Tops | T-shirt in soft cotton jersey with embroidered text at the top |
| Trousers | Trousers in stretch twill with a high elasticated waist and a concealed zip in one side. Fake welt pockets at the back and slim legs with creases and short slits at the hems. |

*Table 3.2: Industrial Dataset 1 - Caption Examples*

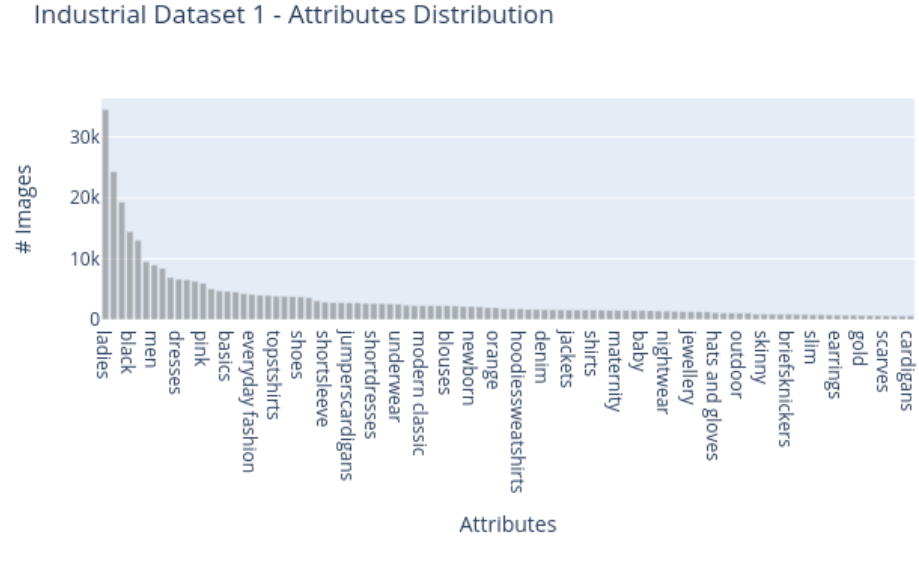
## 3.3 Industrial Dataset 2

We can leverage a second industrial dataset thanks to the collaboration with another fashion industrial partner. This dataset is different from the others both in terms of dimension and type of annotations. First of all, it consist of 1526 images, which is a small number compared to the other datasets, which we split into 1225 for training, 137 for validation and 164 for the evaluation. Despite its size, it is a relevant dataset in our experiments since it has fine-grained annotations and rich and complex descriptions which we exploit for the Fashion Captioning task.

The dataset is annotated with:




*(a)*




*(b)*

*Figure 3.3: Industrial Dataset 1 - Classes and Attributes Distributions*

- 1087 attributes which are not mutually exclusive as they contain both clothing categories, colors and more specific detail as fabric and clothing parts. Therefore, there can be more than one attribute for each item.
- 593 details which are short sentences (the average length is 5.64 words) which describe fine-grained details as pockets, zip closures, neckline shape and so on.
- one long description for each item which is characterised by a complex structure and refined words which are meant to arouse the customer interest in the garment. The average length of these captions is 52.77 which is more than double the length of our Industrial Dataset 1 descriptions (23.66). Caption examples can be seen in table 3.3.

| **Class** | **Caption** |
|---|---|
| Trousers | The slightly shaded color of the delave linen chevron enhances the style of the trousers with a light touch. The drawstring and pleating details are paired with the modern leisure fit, which offers soft proportions in the seat and rise, while the cut of the leg remains close to the body. |
| Sneaker | New materials inspired by the Active world update the style of these sneakers. The upper combines a sophisticated fabric which mimics the look of mesh, with washed suede details on the toe and heel. Fine grooves enrich the back of the two-tone EVA outsole and the TPU insert at the base of the heel counter. A lightweight rubber tread with custom design provides the fit with added comfort. |
| Pullover | Precious cashmere fiber enhances the long sleeve pullover, four-season piece for a man's wardrobe. Narrow, contrasting color trim on cuff and collar edges adds refined detail to the garment which offers a regular fit with a close-fitting shape. |

*Table 3.3: Industrial Dataset 2 - Caption Examples*

## 3.4 Datasets Comparison

As we summarise in table 3.1, we use these datasets for specific tasks according to the type of annotations that they have available.

For the Image Tagging task we train and evaluate our models with DeepFashion and Industrial Dataset 1 since they both have tags annotations in

the form of Category, coarse-grained and mutually exclusive labels, and Attributes, more fine-grained and independent tags. In table 3.4, we compare the two datasets to highlights the differences in term of size of the dataset splits and number of classes and attributes types.

| **Tagging Datasets Comparison** | | | | | |
|---|---|---|---|---|---|
| Dataset | Train | Val | Test | Classes | Attributes |
| Deep Fashion | 209222 | 40000 | 40000 | 50 | 1000 |
| Industrial Dataset 1 | 54387 | 13597 | 9270 | 27 | 255 |

*Table 3.4: Comparison between datasets for Fashion Tagging*

Regarding Fashion Captioning, we present a comparison between our datasets provided with clothing item descriptions. As it is possible to notice from table 3.5, the dimensions of the two dataset differ greatly as Industrial Dataset 2 has very few items with respect to Industrial Dataset 1. On the other hand, the former dataset has an average caption length of 52.77 which is more than double than the length of the latter.

| **Captioning Datasets Comparison** | | | | |
|---|---|---|---|---|
| Dataset | Train | Val | Test | Avg. Caption Length |
| Industrial Dataset 1 | 54387 | 13597 | 9270 | 23.36 |
| Industrial Dataset 2 | 1225 | 137 | 164 | 52.77 |

*Table 3.5: Comparison between datasets for Fashion Captioning*

# Chapter 4

# Related Work

In this section we present the works in the literature which are mostly related to our approach in the performed tasks and the algorithm structure. We present separately the approaches related to the Image Tagging task from those that are more relevant for Image Captioning.

## 4.1 Fashion Tagging

Several interesting Fashion Image Tagging approaches related to our work are trained and tested on the large-scale fashion dataset DeepFashion [LLQ+16]. They propose Deep Neural Networks architectures which differ in the model structure and in the tasks that they achieve, where some focus also in Landmark Localization, i.e. identifying a set of points related to clothing part, in addition to Category and Attributes Prediction, which consist in generating two type of tags, categories and attributes, as explained in chapter 3.1.

### 4.1.1 Deep Fashion Analysis with Feature Map Upsampling and Landmark-driven Attention

This work [LL18] is the current state-of-the-art in terms of evaluation metrics on landmark localization, category classification and attribute prediction for DeepFashion public dataset. The model is based on the VGG-16 CNN and the authors add a landmark localization branch and a landmark-driven attention branch. In particular, they combine the predicted landmark information with the convolutional features to form an attention map which helps the network in focusing on the most functional parts of the clothes for category and attribute prediction with the reference to both local landmark positions and global features. The goal of this attention mechanism is to enhance the visual features which are more related to fashion analysis

while filtering out unrelated features. The attention branch and the network architecture are shown in figure 4.1.

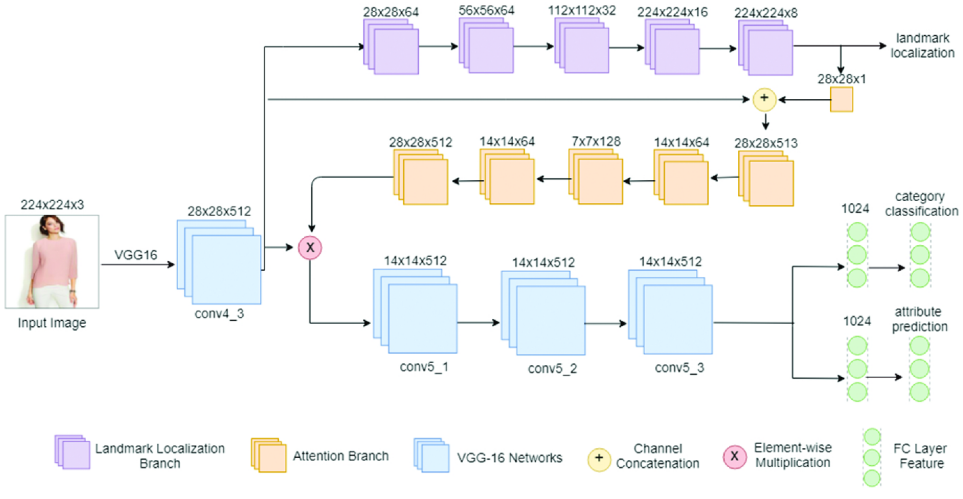



*Figure 4.1: Architecture with Feature Map Upsampling and Landmark-driven Attention. Image taken from [LL18]*

In order to obtain heatmaps with high resolution, the landmark localization branch is a series of transposed convolution used to upsample the feature map. In this way the predicted landmark heatmaps have the same size as the input image, which improves the accuracy of landmark localization. The features obtained for landmark localization are combined with the visual features to obtain the landmark-attention mask, after passing through a series of downsampling and upsampling operations in the attention branch. The masked visual features are fed as input to the last layer of VGG16 and then used to predict scores for categories and attributes.

### 4.1.2 Leveraging Weakly Annotated Data

The model presented by Corbiere et al. [CBYRO17], differs from other approaches evaluated on the Deep Fashion dataset as it doesn't exploit landmark information to achieve good results for both category and attributes prediction. This model is trained in a weakly supervised way, learning from a dataset crawled on e-commerce catalogues and without any explicit labeling. The labels associated to each item are extracted from the textual description associated to the image, keeping preprocessing as light as possible.

The model is based on a ResNet-50 as the image feature extractor and the image features are coupled with the word embedding in a dot product to compute the compatibility scores of the image with respect to all the

labels. In order to manage the highly unbalanced word distribution in the noisy dataset, they perform uniform sampling where, during training, they sample uniformly a word w from the vocabulary and then randomly choose an image whose bag-of-words contains w and they try to predict w given the image. Since they want to predict a single label w, the compatibility scores obtained with the dot product are passed through a softmax layer and the loss is computed with a categorical cross-entropy loss as in a multi-class classification scenario.

With this model, they reach performances comparable to the state-of-the-art on the DeepFashion dataset on both Image Retrieval and Attribute Predictions tasks, without using the training dataset to refine the image features, thus providing a way to overcome the issue of finding a large and clean e-commerce dataset.

## 4.2 Image Captioning

Image Captioning is a widely studied problem with several interesting approaches presented in recent years. This section is devoted to analysing different models in Image Captioning, from more traditional approaches with a encoder-decoder paradigm where a CNN encodes the input image and a RNN decoder generate the description, to recent architecture that exploit the capabilities of Transformers architecture. We only refer to one work of Image Captioning related to fashion, recently published, since this specific domain has not received a lot of attention from the research community yet. Therefore, with our work we also hope to help the research in this direction.

### 4.2.1 Show, Attend and Tell

Show, Attend and Tell [XBK+15] is one of the most popular approaches related to Image Captioning in recent years. In this work, Xu et al. present a model based on a encoder-decoder architecture with a CNN as encoder and a LSTM as decoder, incorporating also an attention mechanism that allows to focus on different parts of the image each time a new word is decoded.

The CNN encoder extracts the features from the image and, instead of using fully connected layers, the authors keep the output features of the lower convolutional layers maintaining a correspondence between the feature vectors and portions of the 2-D image and this allows the decoder to selectively focus on certain parts of an image by selecting a subset of all the feature vectors. These features are combined together with the hidden states of the LSTM to compute a set of weights, one for each image feature

vector. These weights indicates how much each image feature is relevant in the current decoding step and are used with an attention mechanism to obtain the final image features that become the context vector. The model architecture is shown in figure 4.2.

The authors present also two different variant of the attention mechanism: a "soft" deterministic attention mechanism trainable by standard back-propagation methods and a "hard" stochastic attention mechanism trainable by maximizing an approximate variational lower bound.

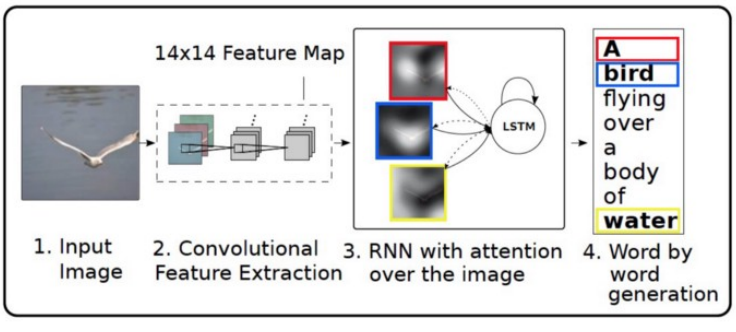



*Figure 4.2: Show, Attend and Tell architecture. Image taken from [XBK+15]*

### 4.2.2 Unified Vision-Language Pre-Training

With the rise of Transformer as state-of-the-art for NLP, different Computer Vision tasks, including Image Captioning, are also being tackled by approaches based on these architectures. In the work "Unified Vision-Language Pre-Training for Image Captioning and VQA" [LHL+19], Zhou et al. present a unified approach for Image Captioning and Visual Question Answering based on a shared multi-layer Transformer network for encoding and decoding. Firstly, they extract a fixed number of object regions from the image using an off-the-shelf object detector and combine region features, object labels and coordinate of the bounding boxed into a unique region embedding. These visual embeddings and the sentence are fed as input to the Transformer layers with three special tokens [CLS], [SEP], [STOP], where [CLS] indicates the start of the visual input, [SEP] marks the boundary between the visual input and the sentence input, and [STOP] determines the end of the sentence. The [MASK] tokens indicate the masked words, where some random tokens are masked and the task is to predict them in the form of a classification problem, thus forming a language model. The main difference between the Image Captioning and VQA tasks lies in the self-attention mask. The mask used for the VQA bidirectional objective

allows unrestricted message passing between the visual modality and the language modality while in text generation, the next word to be predicted cannot attend to the words in the future, therefore the tokens on the right of the next word in the caption are masked. The model structure and the self-attention mask are illustrated in figure 4.3.

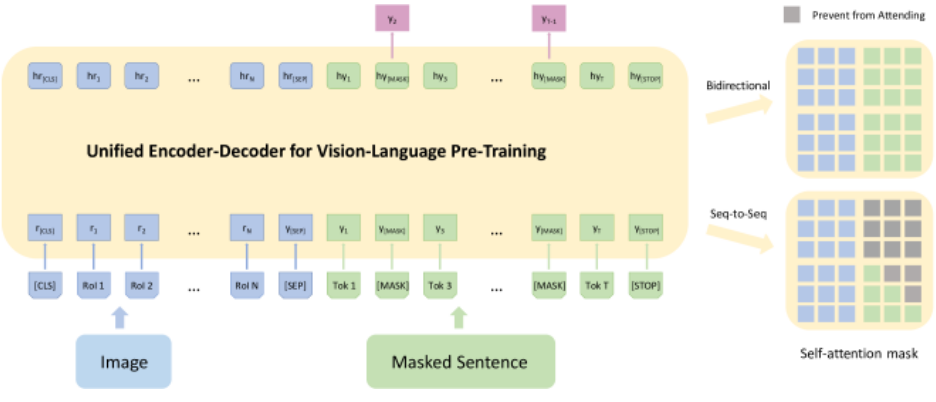



*Figure 4.3: Unified Vision-Language Pre-Training. Image taken from [LHL+19]*

VLP model achieves state-of-the-art results on both image captioning and visual question answering tasks, across three benchmark datasets: COCO Captions, Flickr30k Captions, and VQA 2.0, relying on a unified pre-training without needing multiple trained models for distinct tasks.

### 4.2.3 Fashion Captioning: Towards Generating Accurate Descriptions with Semantic Rewards

Yang et al. [YZJ+20] present an approach which focus on our same task of Fashion Captioning. Their algorithm is based on an architecture similar to the one of Show, Attend and Tell [XBK+15], i.e. a encoder-decoder model with a CNN as encoder and LSTM decoder equipped with a a visual attention mechanism to focus on specific regions of the input image at each decoding step. The novelty of their approach is that they extend this model with additional mechanisms meant to increase the accuracy and quality of the generate caption for a fashion setting. The idea which guides their algorithm design is that each clothing item has several attributes which define it and the model should learn to give more relevance to these details when generating a caption.

The first innovation they introduce is the use of a visual attribute predictor, a feed forward layer, which extract from the image features a attribute embedding vector used to condition the LSTM caption generator as

shown in figure 4.4 where $z$ is the attribute embedding vector. The attribute predictor is trained as in a multi-label classification setting with attributes as label and binary cross entropy loss function.

They also present another way of increasing the accuracy of generated captions by exploiting two semantic rewards as the objective to optimize the model using Reinforcement Learning. Specifically, they propose an attribute-level semantic (ALS) reward with an attribute-matching algorithm to measure the consistency level of attributes between the generated sentences and ground-truth. The goal of this reward is to encourage the model to generate as many correct attributes as possible in a caption. As a second reward, they propose a sentence-level semantic (SLS) reward to capture the semantic meaning of the whole sentence. They train a text classifier to predict the correct item category given the input caption and then use the output of this classifier during training to enforce the generated sentence to describe an item with the correct category. Since both ALS reward and SLS reward are non-differentiable, they use Reinforcement Learning to optimize them.

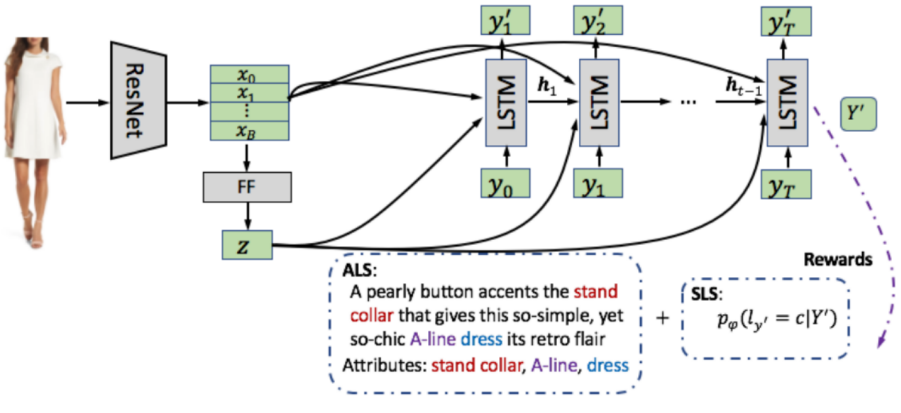



*Figure 4.4: Semantic Rewards guided Fashion Captioning (SRFC) architecture. [YZJ+20]*

# Chapter 5

# Approach

In this chapter we present our approaches for Fashion Tagging and Captioning and we discuss the design decision behind each component. First, we describe our model for tag generation, an architecture composed by a CNN and a Double-Head classification layer to predict class and attributes tags. We then propose a novel approach based on GPT-2 language model for Image Captioning. Finally, we combine our approaches for Tagging and Captioning into a unique model capable of generating both tags and captions.

## 5.1 Double Head TagNet for Fashion Tagging

Tags generation algorithms for fashion items have to deal with various challenges given by the nature of clothing items and real-world applications. First, difficulties arise from the fine-grained details that an algorithm has to find in order to identify correctly a garment. Additionally, clothing items are sometimes deformed or occluded and images often show serious variations depending on the setting in which they are taken, such as in-shop catalogue images or social network images.

In our datasets we have two different types of tags: categories and attributes. The former are mutually exclusive, meaning that each item can belong to at most one category. The latter refer to labels that don't necessarily exclude one another and can be assigned together to an image, in a multi-label setting. Therefore, our solution has to generate both type of labels in order to perform a complete tagging for a fashion image.

Another design decision is to have an architecture which doesn't rely on any other information apart from tags during training. While several state-of-the-art models on DeepFashion dataset leverage landmark information to

enhance the visual features extracted from the image in order to improve category and attribute prediction, we want to keep our approach as flexible as possible, since landmark data has a very high labeling cost and rarely, both in research and industry, we have this information available.

In order to satisfy these requirements, we design an architecture based on Convolutional Neural Network with two heads, one for classifying the category and the other for classifying attributes as two different tasks, as represented in figure 5.1.

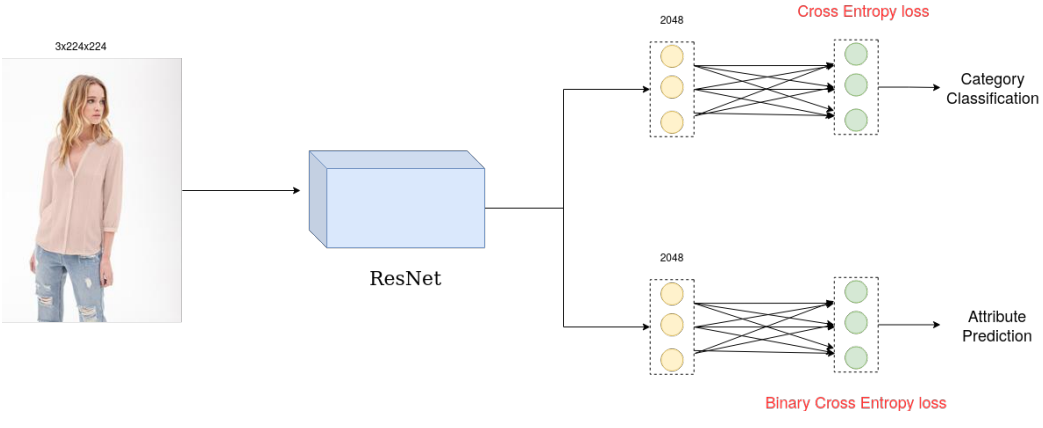



*Figure 5.1: Double Head TagNet Architecture for Categories and Attributes Prediction*

### Feature Extractor

To encode images in a meaningful representation, we choose ResNet architecture presented in [HZRS16] as feature extractor. This architecture has proven its efficiency in different challenges and it is used as backbone in many works across several Computer Vision tasks. Among all the ResNet variant which differs in the number of parameters, we mostly employ ResNet50, as it is used in related works such as [CBYRO17] and it performs well in our experiments.

The input to the feature extractor is an image $\mathbf{I} \in R^{c_i \times w_i \times h_i}$ with width $w_i$, height $h_i$ and 3 channels $c_i$ for the RGB format. The image pass trough the convolutional blocks of the feature extractor which outputs a convoluted feature maps $\mathbf{M} \in R^{c_o \times w_o \times h_o}$ , with more channels and smaller size.

### Double-Head Classifier

The output maps of the feature extractor are flattened into a single vector with a 2D adaptive average pooling with output size=(1,1) which pools each

channel into a single value, transforming $\mathbf{M}$ into a single vector $\mathbf{v} \in R^{c_o}$. $\mathbf{V}$ is fed to the classification heads which output the scores for categories and attributes. These heads apply a linear transformation to the incoming data $y = xA + b$ where A is a weight matrix of size $R^{c_o \times n}$ and $b \in R^n$ is the bias vector with $n$ equal to the number of categories for the category head and number of attributes for the attribute head.

### Loss Functions

Given the different nature of category and attribute tags, they also correspond to two different classification tasks: multi-class and multi-label classification. The main difference between these tasks is that in the former only one specific class can be assigned to the item while in the latter multiple labels may coexist together. To each tasks correspond a loss function that is responsible of expressing the "distance" between the model results and the target so that the model parameters can be optimized to make this value as small as possible. In a classification setting, Cross-Entropy is the default loss function and it is defined as follows:

$$CE = -\sum_{i}^{C} t_i \, log(s_i) \tag{5.1}$$

Here $t_i$ and $s_i$ are, respectively, the target and score for each class $i$ in $C$. There exist different variants of Cross-Entropy depending on the task. In our case for the Multi-Class classification task of predicting categories we use the so-called Categorical Cross-Entropy which is a Softmax activation plus a Cross-Entropy and the target vector is one-hot encoded and only one element in the target vector has non-zero value. The loss function is different in the Multi-Label attribute prediction tasks where, instead of a Softmax layer, there is a Sigmoid activation function, which squeezes the input values between 0 and 1 independently of one another, while the Softmax forces also the values sum to be 1. Furthermore, this loss sets up a binary classification problem for each label, with negative and positive class to indicate whether the label "activates" or not. For this reason this loss is called Binary Cross-Entropy and it is defined as:

$$BCE = -\sum_{i}^{C} (t_i \, log(\sigma(s_i)) + (1 - t_i) \, log(1 - \sigma(s_i))) \tag{5.2}$$

Where $t_i$ and $s_i$ are, respectively, the target and score for each class $i$ in $C$ and $\sigma$ corresponds to the sigmoid function.

## 5.2 Multimodal GPT-2 for Fashion Captioning

Image Captioning is a widely studied problem with interesting and successful solutions as we show in chapter 4. In this thesis work, our focus is on the specific Fashion domain that has few relevant differences with respect to the general Image Captioning setting.

Firstly, as in Fashion Tagging, Fashion Captioning has to recognise the fine-grained attributes of a single item while in a general captioning setting (e.g MS COCO [CFL+15]) generally the caption just need to describe the objects and their relations in the image. Furthermore, since fashion captions need to include different details, they also tend to be longer with a more complex and engaging expression style meant to arouse the customer interest. In figure 5.2 it is possible to notice the differences between an image taken from the MS COCO dataset and a fashion image from one of our industrial datasets.

Another important difference is that current state-of-the-art approaches as [LHL+19] rely on object detectors to extract regions of interests related to specific objects in the image. In our case, object detectors cannot be exploited as the fashion captioner doesn't describe objects and their relations but fine-grained details as patterns, shapes or clothing parts which cannot be easily identified by bounding boxes to train a object detector.

With these challenges guiding our design decisions, we propose a novel architecture based on GPT-2 [RWC+19] that is able to generate a caption leveraging the features extracted from a fashion image. So far, GPT-2 has surprised many because of its ability to generate long and complex sentences starting from a textual context as input. However, the research community has still not answered to the question if these capabilities can serve to generate text from an image and therefore, with our work, we want to provide an answer, or at least give our contribution, to this question.

In addition, given the fact that fashion catalogues are rich not only of images, but also metadata and tags, we consider also the possibility of exploiting both visual and textual features together in a multi-modal approach. This gives our model the flexibility to rely on image features and, if additional information is available, it can leverage it to improve the quality of the generated caption. The architecture of Multimodal GPT-2 is shown in figure 5.3.

*(a) "The man at bat readies to swing at the pitch while the umpire looks on." Image taken from [CFL+15].*

*(b) "Materials with light patterns reinterpret the retro silhouette of this dress with a lively, fresh touch. The pure cotton fabric reworks a classic striped pattern with the season's colors and enhances the garment's fluid lines and feminine proportions, which are gathered at the waist with a removable belt. Rows of shiny monili embroidery on the shoulder straps are paired with the fabric's delicate pleated effect, creating an elegant note." Image taken from our industrial dataset.*

*Figure 5.2: Comparison between General Captioning and Fashion Captioning*

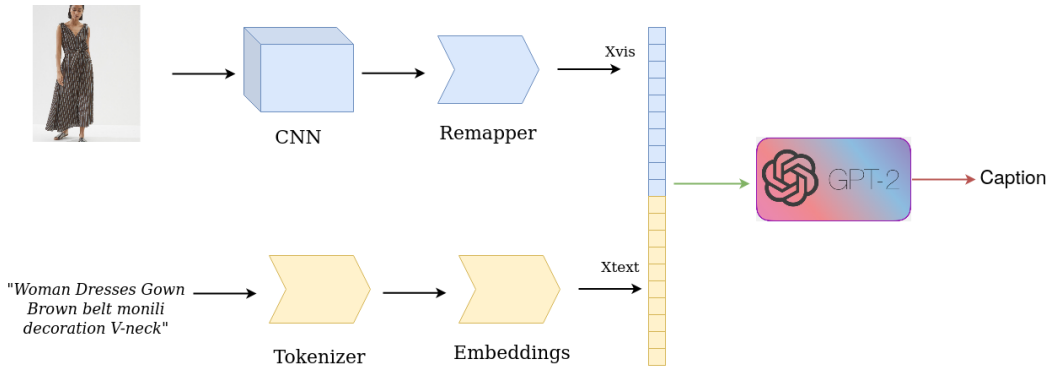



*Figure 5.3: Multimodal GPT-2 Architecture for Fashion Captioning*

### GPT-2

As presented in 2.2.4, GPT-2 is a large-scale transformer-based language model, trained on a vast dataset composed of raw text taken by the web, with the sole purpose of predicting the next word given the past context. Thanks to this simple task, it is able to learn a language model, meaning it is able to learn all the correlations and rules between words that form a language and this knowledge can be leveraged for other downstream tasks. GPT-2 expects as input a sequence of words, with the proper preprocessing and formatting, which goes through its two main blocks: the transformers block and the language modeling head. The former is a stack of Decoder Transformer blocks which exploit multi-head attention to compute an embedding of the next word. The latter is a linear fully-connected layer that takes this embedding as input and return a vector of length equal to the vocabulary length (50257) whose values represent the probabilities of assigning each word to that embedding. There are different variants of GPT-2 that vary in the size of the network, from the smallest version with 117M parameters to the biggest with 1.5 billions.

### Multimodal Embedding

Normally, the input of GPT-2 consists only of a sequence of words that must be tokenized, i.e. converted from human readable words into tokens, enumerated from 0 to the model vocabulary size. We use GPT-2 pre-trained **Tokenizer** which is based on byte-level Byte-Pair-Encoding, a tokenization algorithm which consider text as a sequence of bytes and is language agnostic. It keeps the frequent words as they are while representing rare words with sub-word units, which solves the out-of-vocabulary problem. In addition to the vocabulary tokens, GPT-2 uses another special token to indicate padding and sequence end: $< |endoftext| >$. Internally, GTP-2 converts the tokens to their corresponding embedding $\mathbf{X_{text}} \in R^{n \times hidden}$ where $n$ is the number of tokens and $hidden$ is the embedding size (768 for gpt2-small, 1024 for gpt-2 medium), before giving them as input to the transformer layers.

In our multi-modal setting, we want GPT-2 to accept as input also visual features and not exclusively word tokens. Visual features are extracted from the image using a Convolutional Neural Network, a ResNet architecture as in 5.1, and flattened into a vector $\mathbf{v} \in R^{c_o}$ where $c_o$ is the number of output channels of the CNN. We then convert $\mathbf{v}$ into visual token embeddings $\mathbf{X_{vis}} \in R^{m \times hidden}$ of the same $hidden$ size as the token embeddings using a **Remapper** block. The Remapper is composed by $m$ linear fully-connected

layers which map the visual embedding into the token embedding size, with $m$ being an hyper parameter to select the number of visual token embeddings to produce. The reasoning behind this block is that we want to map visual features into different subspaces in the form of visual tokens which possibly represents attribute types of the clothing item. While in general captioning the state-of-the-art approaches exploit object detectors to get features related to entities in the image, we can't use such method as we don't have specific objects to identify, and, therefore, the role of Remapper block is to learn subspaces to help the model in focusing and discriminating different clothing attributes.

Since visual token embeddings cannot be fed as input to GPT-2 as word tokens with the vocabulary index, we also convert word tokens to their embeddings using GPT-2 Embedding matrix, performing the same process that GPT-2 does internally, obtaining $\mathbf{X_{text}}$. After this, we concatenate the visual token embeddings $\mathbf{X_{vis}}$ with word token embeddings $\mathbf{X_{text}}$. During training, all the input vectors inside a batch must have the same length, however items can have different number of tags and, therefore, the word token embeddings will have various lengths. For this reason we pad input vectors with GPT-2 padding token up to the length of vector with maximum length plus one, leaving always at least one pad token to separate input tokens and the ground truth caption. In fact, during training, the ground truth captions is fed as input, after being tokenized and turned into embeddings as the tag tokens, so that GPT-2 can learn the language model with the teacher forcing method using the ground truth from a prior time step as input. The complete input embedding is illustrated in figure 5.4, where VT correspond to visual tokens $\mathbf{X_{vis}}$, WT to $\mathbf{X_{text}}$ and CT are tokens of ground truth caption. We mask GPT-2 attention for pad tokens between the input and the ground truth caption tokens so that GPT-2 doesn't attend to the pad tokens when computing the next token embedding.

### Loss function

The model output is $\mathbf{O} \in R^{b \times o \times vocabsize}$ where $b$ is the batch size, $o = m + n + p_1 + c$ is the total number of tokens in the multi-modal input embedding given by the sum of visual tokens $m$, tag tokens $n$, extra padding $p_1$ and caption tokens $c$, $vocabsize$ is the number of words in the vocabulary. As the output represents the scores of all the vocabulary words for each output token, the loss is computed by comparing only the last $c$ output tokens which corresponds to the generated caption with the ground truth caption tokens. As in a multi-class classification task the loss we use is a

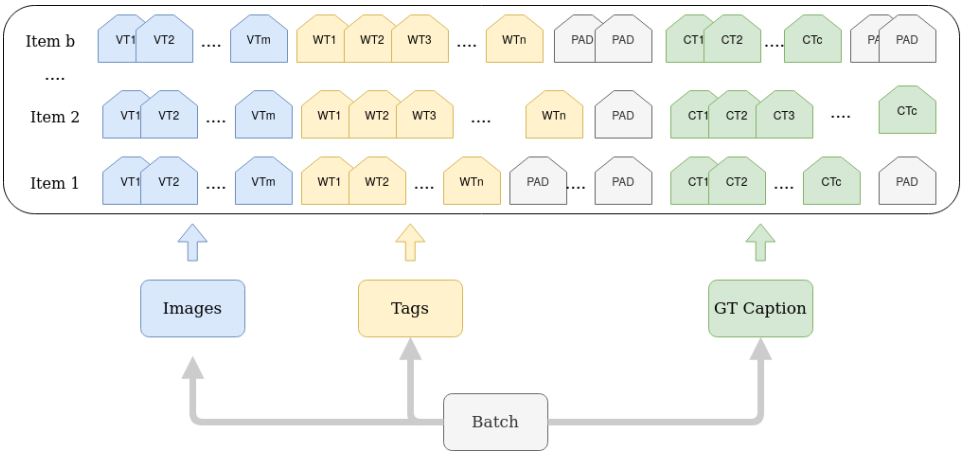



*Figure 5.4: Multimodal Embedding at Training Time*

Categorical Cross Entropy loss function, where the target classes are the ground truth caption tokens.

### Caption Generation

During inference, instead of feeding the ground truth caption as input as at training time, the model relies solely on the visual and tag tokens combined in the multi-modal embedding. In order to generate a caption we use Beam Search, a decoding method that doesn't simply selects the word with the highest probability as its next word as Greedy Search, it keeps a number of most likely hypothesis sentences and eventually it chooses the one that has the overall highest probability. The problem with Greedy Search is, in fact, that if first words chosen are not the best, then everything that follows is sub-optimal. Beam Search gives the possibility to consider, at each step, the top k words candidates, consider the combinations with the best overall score and discard the rest. This reduces the risk of missing hidden high probability word sequence and guarantees that Beam Search will find an output sequence with higher probability than Greedy Search. Figure 5.5 provides an example of Beam Search decoding method.

## 5.3 TagCaptioner GPT-2 for Fashion Tagging and Captioning

We also present a model which is able to perform both the Image Tagging and Image Captioning tasks simultaneously. As a matter of fact, this model, which we refer to as TagCaptioner GPT-2, is a combination of our

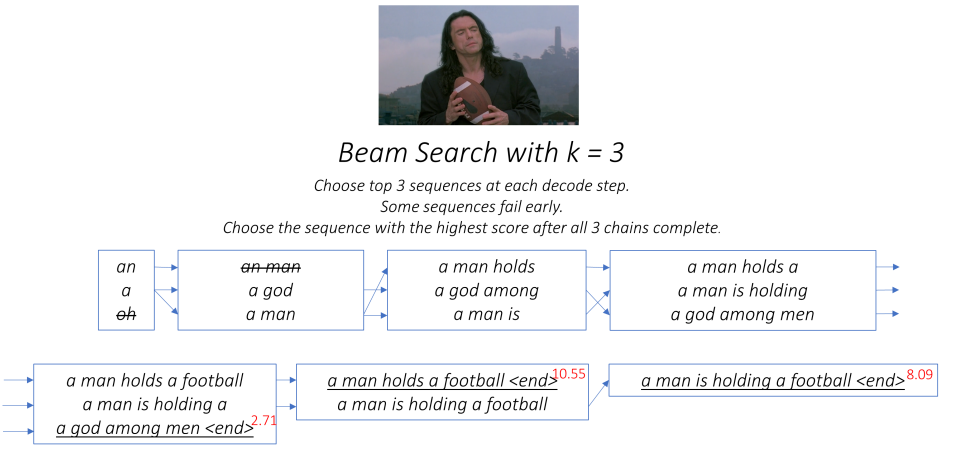



*Figure 5.5: Beam Search Example. Image taken from [Vin]*

approaches explained in the previous sections where, instead of using ground truth tags as additional information in the multimodal embedding to generate the caption, it generates itself the tags from visual features. As it possible to see from figure 5.6, the architecture is very similar to Multimodal GPT-2, however here the CNN output is also fed to a double-head classification layer, same as the one explain in 5.1, which return the prediction scores for categories and attributes. This model is flexible and is not strictly related to the type and number of tags where, for example, we could add more heads to take care of other type of tags or split the attributes head into one head for each attributes type for a more fine-grained classification.

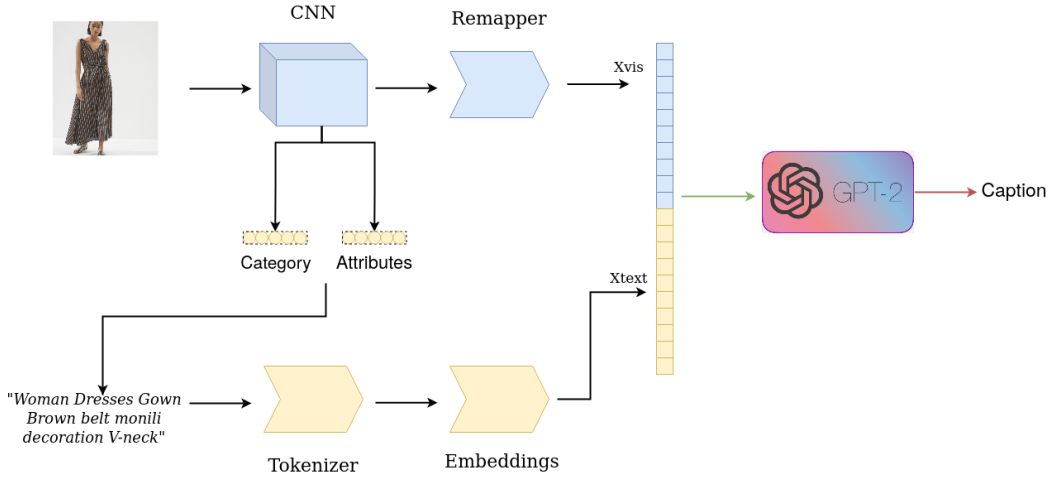



*Figure 5.6: TagCaptioner GPT-2 Architecture*

### Loss Function

In our experiments we keep this setting with two classification heads, therefore the model has 3 different loss functions: a Categorical Cross-Entropy loss for category prediction and a Binary Cross-Entropy for attributes prediction, as explained in 5.1, and the Categorical Cross Entropy for captions as explained in 5.2. For the experiments on Industrial Dataset 2, however, which has different types of tags than categories and attributes, we use a different approach with two Binary Cross Entropy losses for tag generation, which we explain more in depth in chapter 6.2.6.

### Tag Generation

Before being converted to human language text and pre-processed, the tags are just vectors of prediction scores returned by the double-head classification head. From these scores, we take the top-k scores for both categories and attributes, top-1 for the former since there should be only one category for each item and top-5 for the attributes which are, instead, not mutually exclusive. After this, we take the corresponding labels in natural language from the category and attributes dictionaries and concatenate them in unique sentence which is processed to become a part of the multimodal embedding as we explain in 5.2.

# Chapter 6

# Experiments and Results

In this chapter we present an analysis of the performances of our approaches for Fashion Tagging and Captioning on the respective datasets. We are going to present separately the experiments and results on the two tasks and analyse them quantitatively using evaluation metrics and qualitatively by providing examples. Table 6.1 summarizes which datasets we use to evaluate our approaches. DeepFashion and Industrial Dataset 1 allow us to test our model on the tag generation, thanks to their class and attributes annotations and the latter is also used for the generation of captions with Industrial Dataset 2, since they both have caption annotations.

| **Datasets and Tasks** | | |
|---|---|---|
| Dataset | Tagging | Captioning |
| DeepFashion | X | |
| Industrial Dataset 1 | X | X |
| Industrial Dataset 2 | | X |

*Table 6.1: Tasks and related datasets*

## 6.1 Fashion Tagging

In this section we present different experiments and results that we perform on the Fashion Tagging task, i.e. label an item with the corresponding garment class and more fine-grained attributes. We evaluate the performance of the Double Head TagNet architecture presented in chapter 5.1 on two datasets: the public fashion dataset DeepFashion and a private dataset which we refer to as Industrial Dataset 1. One of our goals is to compare the performance of our approach, capable of recognising classes and attributes

of a fashion garments without relying on the additional landmark data, with other works on DeepFashion which instead exploit it. As a matter of fact, this type of extra annotation is rarely available in an industrial setting and, therefore, we want to find out whether we can obtain similar performance without using it.

### 6.1.1 Evaluation Metrics

In order to evaluate and compare the experiments we compute different evaluation metrics to obtain reliable results. The Fashion Tagging task is a type of classification and, therefore, we use traditional classification metrics as accuracy, precision and recall and also others more related to the information retrieval field, as precision@k, recall@k and mean average precision (mAP).

First, we must distinguish between the two type of tags that our models generates for each item: class and attributes. The class tag refers to the unique type of garments to which an item belong, e.g Dress, and the generation of this tag corresponds to a multi-class classification, where only one label among multiple ones is assigned to an item. The attributes refer, instead, to more fine-grained details that can characterise an item such as the fabric, shape or style and, therefore, multiple labels may be associated to an item. This correspond to a multi-label classification task and it's more challenging than the multi-class as the algorithm needs to discriminate between more labels with an increased attention to particular details. Since these two type of tag generation correspond to different classification task, we also use different evaluation metric to evaluate them.

For the class prediction task, we compute the top-k accuracy metric, i.e. for each prediction we assign 1 if one of the top-k scores correspond to the correct class, and 0 otherwise and the final metric is the mean of all the item scores in the evaluation set. A top-1 accuracy corresponds to the traditional accuracy that verifies if the class with highest prediction score is the correct one.

To evaluate attribute prediction, we use other metrics related to precision and recall for information retrieval where, given a query, they count the number of relevant item among the retrieved ones. Precision is actually the fraction of true positives (retrieved and relevant items) and true positive plus false positives (the retrieved items). Recall, on the other hand, is the percentage of true positives out of the sum of true positives and false negatives which correspond to all the relevant items. In our setting the retrieved items correspond to the top k attributes with the highest score

and relevant items are the attributes in the ground truth for each item. This is how precision@k and recall@k metrics are computed.

Recall@k metric is in a way limited by the choice of $k$, where it may happen that the number $g$ of attributes in the ground truth is higher than $k$ meaning that, even if the model puts all the ground truth attributes in the top scores the maximum recall achieved is $\frac{k}{g}$. On the contrary, if $k$ is much higher than $g$ then it becomes more easy to obtain a high recall. Therefore, we also employ another metric, which we refer to as recall@sample_k, that is based on recall@k with the difference that $k$ is not fixed, but it correspond to the number $g$ of ground truth items for each samples. In this way, the number of retrieved attributes is always equal to the number of relevant items, which helps in mitigating the limitations just mentioned.

Another metric we use to obtain more robust and reliable evaluations, is the Mean Average Precision (mAP). Average Precision metric is obtained by computing, for each $k$ position of the ordered retrieved items $n$, the precision@k score only if the item at position $k$ is relevant, then sum all the scores and divide by the number of relevant items $g$. The mAP score is the mean of AP scores of all queries:

$$\mathrm{AP} = \frac{1}{g} \sum_{k}^{n} (precision@k \times rel@k) \tag{6.1}$$

$$\mathrm{mAP} = \frac{1}{N} \sum_{i}^{N} \mathrm{AP}_i \tag{6.2}$$

### 6.1.2 Experimental Setup

All of our models, training and evaluation blocks are implemented in Python with PyTorch[1] deep learning framework. Our architecture for tag generation, the Double-Head TagNet described in 5.1, is composed by a Convolutional Neural Network, as image features extractor, and two linear layers, responsible for computing the scores respectively for class and attributes, given the image features. As feature extractor, we select a ResNet50 pre-trained on ImageNet for our experiments, since related works use this architecture or other comparable ones, such as VGG-16. The dimension of the output vector of the ResNet is 2048 which is fed directly to the linear layers for the downstream classification tasks. Each input image is resized to a 224 square, normalized with the same mean and standard deviation

[1] `https://pytorch.org/`

of the models trained on ImageNet and, during training, is randomly horizontally flipped with a 50% probability. Furthermore, images from DeepFashion dataset, which is also annotated with one bounding box per image, are cropped to the bounding box size, since all other related works trained on DeepFashion perform this transformation. We use a stochastic gradient descent optimizing algorithm with momentum set to 0.9. A Plateau learning rate scheduler is responsible of decreasing the learning rate with a factor of 0.1 after observing that the validation loss stops decreasing. After performing hyper-parameter tuning with a grid search on the metrics evaluated on the validation set, results reported at appendix A.1, we select as initial learning rate $1e-3$ with batch size 8. We perform early stopping on our training when the validation loss stops decreasing for 3 epochs. All our experiments have been carried out on a p2.xlarge instance available on Amazon Web Services[2] with one NVIDIA K80 GPU.

### 6.1.3 Data

As we explain in chapter 3.4, we evaluate our approach for Fashion Tagging on the datasets DeepFashion and Industrial Dataset 1. Table 3.4 in the same chapters highlights the most relevant differences between them which are the number of items (289222 vs 77254 total images respectively), number of classes (50 vs 27) and number of attributes (1000 vs 255).

### 6.1.4 Experiment 1: Attributes Loss Function

Our model performs two different tasks simultaneously: Class Prediction and Attributes Prediction. In the case of DeepFashion dataset, the latter task is quite challenging as there are 1000 attributes and they represent fine-grained details of the garment which are difficult to identify and discriminate. Therefore, our goal is to improve the Attribute Prediction performance without affecting the Class prediction task performance. In this section we present the results on different experiment we perform on the loss function for attributes.

#### Attributes Loss Scaling

First, we try a different reduction method than the default method of the Binary Cross Entropy (BCE), the loss function we employ for the Attributes Prediction task, as we explain in 5.1. In the default implementation the final loss value for a batch is computed by reducing with the mean over the

[2] `https://aws.amazon.com/`

attributes loss values and then averaging over all the items in the batch. Instead of averaging over the attributes, which shrinks the loss function to low values, we sum their values and then compute the mean over the batch. The main difference is that we don't divide the loss by the total number of attributes, which is 1000 in DeepFashion, and therefore, the attributes loss function has an higher value thanks to this scaling.

| **Loss Function Experiments with scaling** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attribute Loss | p@k | | | r@k | | | r@sample_k | map |
| | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | | |
| BCE (default) | 40.95 | 28.11 | 21.92 | 13.51 | 26.26 | 33.51 | 26.87 | 31.28 |
| BCE with scaling | **47.96** | **33.88** | **26.22** | **16.14** | **31.91** | **40.16** | **32.68** | **37.9** |

*Table 6.2: DeepFashion Attribute Prediction - Improvement with Scaling*

As it is possible to notice from table 6.2, the model trained with the attribute loss function with scaling outperforms the one with the default reduction method. This is probably due to the fact that, without the scaling the attribute loss function has a low value and, therefore, the optimization of our model parameters is mainly guided by the loss function of the Class Prediction task which has an higher value. With the attribute loss scaling, our model and, in particular, the feature extractor is more optimized to encode image features that contain fine-grained attribute details than the more class-specific characteristic to discriminate high-level classes.

| **Loss Function Experiments** | | | |
|---|---|---|---|
| Attribute Loss | class accuracy@k | | |
| | top-1 | top-3 | top-5 |
| BCE (default) | 75.01 | 91.98 | 96.26 |
| BCE with scaling | 75.44 | 92.04 | 96.33 |

*Table 6.3: DeepFashion Class Prediction is not affected by Attributes Loss Scaling*

On the other hand, this procedure might have a drawback on the Class Prediction task. The results on table 6.3 seem to indicate that this is not the case given the fact that performance on Class Prediction are very similar between the experiment with the default attribute BCE loss and the one with scaling. Therefore, the features encoded by the CNN optimized by the scaled attribute loss, are also useful to discriminate between classes.

### Weighted Loss for Unbalanced Datasets

One of the main challenges that can be encountered in a classification task is dealing with an unbalanced dataset. Unbalance means that there is a

disparity in the class distribution where some classes have much more instances than other and, therefore, the algorithm tends to assign items more to labels that belong in the majority rather than the less frequent ones. We notice a similar characteristic also in DeepFashion attributes distribution as we explain in chapter 3.1.

There are sampling techniques to mitigate this problem such as undersampling and oversampling, where the former consists in randomly deleting samples from the classes with an higher frequency while the latter increase the number of samples from the minority classes by just presenting to the algorithm multiple copies of the same data point. Another way of handling this disparity is to use class-wise weights in the training loss with the goal of re-weighting the loss function to give more relevance to the less frequent labels, with the advantage that it doesn't involve manually changing the data distribution. In our setting, in order to deal with the unbalance of the attributes distribution in the DeepFashion dataset, we choose to perform experiments with different weighting schema for the attribute loss function.

As we explain in chapter 5.1, the loss function that we use to optimize our models parameters for the attribute prediction task is the Binary Cross Entropy loss function which compares the attributes prediction scores with the ground truth binary vector. This loss functions computes a value for each label depending on the prediction score and if it is in the ground truth. In the PyTorch implementation of this function[3], there is the possibility to pass as input to the loss function also a vector of positive weights which are multiplied independently with the positive term of the loss function:

$$\text{PW} = -\sum_{i}^{C}( \, p_i \, t_i \, log(\sigma(s_i)) + (1 - t_i) \, log(1 - \sigma(s_i))) \tag{6.3}$$

$$p_i = \log\left(\frac{negative\_count_i}{positive\_count_i}\right) \tag{6.4}$$

here $t_i$ and $s_i$ are, respectively, the target and score for each class $i$ in $C$ and $\sigma$ corresponds to the sigmoid function, $p_i$ is the positive weight of label $i$ and it is computed as the fraction between the number of negative example and positive examples of label $i$ in the dataset. Therefore, the loss acts as if the number of positive and negative samples in the dataset were balanced. When $p_i$ is a vector of ones, then this correspond to the normal BCE loss.

A different type of weighting is used in Focal Loss [LGG+17], which is a loss function, based on the BCE loss, used to address the problem of

[3] `https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`

class imbalance. In the original paper, Focal Loss is adopted in the contest of Object Detection where there can be imbalance between the foreground (the objects that the algorithm wants to detect) and the background where the latter usually represent the vast majority. The goal of Focal Loss is to control the weighting of the BCE loss to give more importance to "hard" misclassified foreground samples than the "easy" correctly classified background. Focal Loss is formulated as:

$$\text{FL} = -\sum_{i}^{C}( \alpha \, (1-\sigma(s_i))^{\gamma} t_i \, log(\sigma(s_i)) + \, (1-\alpha) \, \sigma(s_i)^{\gamma} \, (1-t_i) \, log(1-\sigma(s_i))) \tag{6.5}$$

where $\alpha$ and $\gamma$ are hyper parameters set respectively to 0.25 and 2 which, according to the study mentioned in [LGG+17] give the best results.

We also experiment with another type of loss function in which we implement our own weighting schema based on the ranking of the scores, i.e. the label ordering according to their prediction scores. The idea is to compute the loss not only with the score of each label independently from one another, by penalizing low score for correct labels and high scores for wrong ones, but also with a knowledge about the ordering of the labels where correct classes should be ranked higher than wrong ones. The loss formulation is very similar to PW weighting schema with the difference that here the positive weights are not computed a priori from the training dataset and fixed but change for each sample:

$$\text{RL} = -\sum_{i}^{C}( \, r_i \, t_i \log(\sigma(s_i)) + (1-t_i) \log(1-\sigma(s_i))) \tag{6.6}$$

$$r_i = \alpha \log(rank_i + 1) + 1 \tag{6.7}$$

here, $r_i$ are weights built from the rank of each label in the scores order where $rank_i$ corresponds to the number of false labels which have a score higher than label $i$, $\alpha$ is an hyper parameter set to 0.25. Since we use $r_i$ as a positive weight, i.e. it multiplies the positive term of BCE when $t_i = 1$, this weight penalizes when a correct label is ranked lower than false labels which happens often with less frequent classes. We refer to this loss function as Rank Loss.

In table 6.4 we compare the results of the model with different weighted loss function with the performance of the standard BCE loss with scaling. For a fair comparison, we apply scaling also on the weighted losses to make sure they all have similar values at training time. These models are trained until early stopping. As it can be noticed from table 6.4, the experiments we

| Loss Function Experiments | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attribute Loss | p@k | | | r@k | | | r@sample_k | map |
| | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | | |
| BCE | **47.96** | **33.88** | **26.22** | **16.14** | **31.91** | **40.16** | **32.68** | **37.9** |
| PW | 44.49 | 31.76 | 24.97 | 15.02 | 30.12 | 38.56 | 30.72 | 36.06 |
| FL | 42.08 | 28.98 | 22.51 | 13.79 | 26.92 | 34.21 | 27.49 | 31.91 |
| RL | 47.06 | 33.49 | 26.08 | 15.83 | 31.58 | 39.99 | 32.31 | 37.7 |

*Table 6.4: Deep Fashion Attribute Prediction - Weighted Loss Function Experiments*

perform with different weighting schema don't achieve better results than the standard BCE loss with attributes scaling. Therefore, trying to balance the long-tail attributes distribution using weighting schema for the loss function doesn't seem to improve the performance of our model.

### 6.1.5 Experiment 2: Comparison with Baselines

#### Baselines

We compare our approach with related works on the Deep Fashion dataset and, in particular, we focus on two baselines, presented also in chapter 4, which are among the reported state-of-the-art approaches in terms of metrics for this dataset.

The first baseline corresponds to the work of Liu et al. in the paper *Deep Fashion Analysis with Feature Map Upsampling and Landmark-driven Attention* [LL18] in which the authors present a model based on a Convolutional Neural Network that leverages the extra landmark annotations in the dataset with a landmark localization branch and a landmark-driven attention branch. In particular, they combine the predicted landmark information with the convolutional features to form an attention map which helps the network in focusing on the most functional parts of the clothes for category and attribute prediction with the reference to both local landmark positions and global features. The results reported in the paper are the current state-of-the-art for DeepFashion dataset and our main reason for comparing our results with this approach is to check whether the landmark attention mechanism is needed to obtain good results or it is possible to have good performance without the additional landmarks annotation. The official implementation of this paper is available online[4], without pre-trained models.

The second baseline we refer to is from the paper *Leveraging Weakly Annotated Data for Fashion Image Retrieval and Label Prediction* [CBYRO17]

[4] `https://github.com/fdjingyuan/Deep-Fashion-Analysis-ECCV2018`

written by Corbiere et al. This model differs from other approaches evaluated on the Deep Fashion dataset as it doesn't exploit landmark information to achieve good results for both category and attributes prediction and it is trained in a weakly supervised way, learning from a dataset crawled on e-commerce catalogues and without any explicit labeling. After this pretraining, the feature extractor is used to encode image features for DeepFashion Class and Attribute prediction, without further training.

We also compare to a naive baseline, that we refer to as Top Popular which simply ranks class and attributes according to their frequency in the training set, returning a fixed score ordering for each item.

**Deep Fashion Dataset**

In this section we present the results of our approach on the Deep Fashion dataset in comparison with the mentioned baselines using the evaluation metrics reported in the literature for this dataset: accuracy for Class Prediction and top-3, top-5 recall for the Attributes Prediction, considering also individually each one of the attributes type: Texture, Fabric, Shape, Part and Style.

First of all, since the code of Liu et al. [LL18] baseline is available online, we try to reproduce ourselves the results reported in the paper by retraining the model and using also their evaluation code. What can be noticed from table 6.5, is that Liu et al. trained by us achieve similar results to the reported one apart from the top-3 and top-5 recall for all the attributes where the reported results are 54.69 and 63.74 while the reproduced are 25.60 and 33.40 and also for only-Style attributes type where the reported Style top-3 and top-5 recall are, respectively, 68.82 and 74.13 while the ones reproduced reach 30.29 and 38.80. Style attributes are quite different from the other types as they don't specifically refer to a visual feature, but rather to a particular meaning related to the garment, as an example attributes related to style are "summer", "elegant", "baseball", "shopping" and so on. Therefore, for the fact that they are less strictly related to a single visual feature, they are also less easy to discriminate and this could suggest why the algorithm struggles on that part. However, the results they report on the paper go against this reasoning as the Style recall metrics are higher than all the other attributes types metrics. Above all, they are also very different from the results that we have reproduced. We notice in the 'Issues' page[5] of the online repository that other people as well don't succeed in reproducing the results of the Style attributes and, since the authors don't

[5] `https://github.com/fdjingyuan/Deep-Fashion-Analysis-ECCV2018/issues`

share their pre-trained models online and we have been trying to contact them without success, we decide to give more relevance to the reproduced results rather than the reported ones.

| **Deep Fashion Results** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Class - Accuracy | | | Attributes - Recall@k | | | | | | | | | | | |
| | | | | Texture | | Fabric | | Shape | | Part | | Style | | All | |
| | top-1 | top-3 | top-5 | top-3 | top-5 | top-3 | top-5 | top-3 | top-5 | top-3 | top-5 | top-3 | top-5 | top-3 | top-5 |
| Top-Popular | 24.91 | 46.18 | 58.43 | 37.84 | 45.96 | 19.61 | 27.27 | 21.09 | 27.70 | 18.51 | 25.06 | 14.94 | 23.69 | 08.52 | 12.27 |
| Corbiere et al. | | 86.30 | 92.8 | 53.60 | 63.20 | 39.10 | 48.80 | 50.10 | 59.50 | 38.80 | 48.90 | 30.50 | 38.30 | 23.10 | 30.40 |
| Liu et al. (reported) | | 91.16 | 96.12 | 56.17 | 65.83 | 43.20 | 53.52 | 58.28 | 67.80 | 46.97 | 57.42 | 68.82 | 74.13 | 54.69 | 63.74 |
| Liu et al. (trained) | 72.30 | 90.2 | 95.1 | 55.00 | 64.76 | 42.00 | 52.20 | 55.90 | 65.36 | 44.10 | 54.36 | 30.29 | 38.80 | 25.60 | 33.40 |
| Ours (VGG16) | 73.01 | 90.86 | 95.74 | 57.05 | 66.77 | 44.57 | 54.72 | 58.89 | 68.09 | 47.33 | 57.75 | 32.84 | 41.46 | 27.69 | 35.88 |
| Ours (ResNet50) | **75.44** | **92.04** | **96.33** | **60.00** | **69.70** | **49.40** | **59.70** | **62.06** | **71.00** | **52.3** | **62.6** | **37.8** | **46.04** | **30.71** | **39.63** |

*Table 6.5: Deep Fashion - Comparison with Baselines*

The results reported on table 6.5, shows that our approach, based on a ResNet50 as feature extractor, a double head classification layers for the Class and Attributes prediction and with attribute loss scaling achieve results which are better than the other baselines. Corbiere et al. [CBYRO17] is the one more similar to our approach since it doesn't exploit the additional landmark information and it's also different from us as it doesn't fine-tune its feature extractor on DeepFashion and this is probably why our metrics are higher, since we train the CNN to extract features more specific to this dataset. Liu et al., instead, exploit a landmark localization and landmark attention branch to enhance the features extracted by a VGG16 and use them to improve category and attribute prediction. However, what can be noticed from the results is that our model without landmark attention achieve higher evaluation metric values, therefore this suggests that using a landmark attention branch doesn't seem to be really effective in the classification of classes and fine-grained attributes of clothing items. To make the comparison more fair, we also test our approach with a VGG-16, the same one used by Liu et al. and, while obtaining results lower than our approach with ResNet50, we still have similar values, even slightly better, to the reproduced results of Liu et al.

### Industrial Dataset 1

We perform experiments with our model also on a private dataset that we refer to as Industrial Dataset 1. This dataset differs from the large-scale public dataset as it has a lower number of classes and attributes, respectively 27 and 255, and the latter don't contain fine-grained details of clothing item but more categorical information as particular clothing sub-categories, gender, age or color, making it less challenging than DeepFashion.We can't

compare our results with the same baselines of DeepFashion as we can't reproduce the Corbiere et al. [CBYRO17] approach since we don't have their pretrained model and we can't train the Liu et al. baseline since for this dataset we don't have the additional landmark information. Nevertheless, we present the results of our best model in comparison with the TopPopular naive baseline in table 6.6, using the evaluation metrics presented in 6.1.1. It is possible to state that our model is able to recognise more than 87% of the item classes in the test set, by placing the correct label in the first position. As we explain in 3.2, there are on average 5.41 attributes per item in this dataset and this explain why the r@sample_k (74.34) is higher than the r@5 metric (70.67), since r@sample_k computes the recall after retrieving the top attributes in equal number to the ground truth attributes while r@5 always retrieve the first top 5 attributes.

| **Industrial Dataset 1 Results** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Class | | | Attributes | | | | | | | |
| | accuracy | | | p@k | | | r@k | | | r@sample_k | map |
| | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | | |
| Top-Popular | 10.1 | 27.1 | 41.2 | 50.1 | 38.2 | 31.0 | 9.20 | 21.0 | 28.8 | 29.58 | 30.2 |
| Ours (ResNet50) | **87.61** | **98.30** | **99.36** | **97.09** | **87.78** | **75.56** | **18.46** | **49.78** | **70.67** | **74.34** | **83.74** |

*Table 6.6: Industrial Dataset 1 - Comparison with Baselines*

## 6.2 Fashion Captioning

In this section we present the results of Multimodal GPT-2 for Fashion Captioning, explained in chapter 5.2, evaluated in our datasets annotated with captions. Given the fact that the model relies on two different types of input (visual and textual) we also carry out a multimodal analysis, comparing the model performance in difference settings to understand which of these components affect the performance the most. In addition, we also compare our model with other approaches based on the CNN-LSTM encoder-decoder paradigm, with and without visual attention. The comparison between different models consists of a quantitative and qualitative evaluation, where the former is a comparison between evaluation metrics and the latter, instead, corresponds to comparing the generated captions, thus it is a human evaluation to verify the text quality, correctness and coherence.

### 6.2.1 Evaluation Metrics

In order to compare different experiments and models, we compute several evaluation metrics that are also used to evaluate caption generation on the

COCO-Caption dataset [CFL+15]. The evaluation code is available online[6] and we adapt it to support our datasets format. In particular, we refer to the BLEU-n, ROUGE-L, METEOR, and CIDEr metrics in our experiments.

BLEU-n [PRWZ02] is a modified form of precision that compares n-grams (groups of n adjacent words in a sentence) of the candidate sentence with the reference sentence (the ground truth caption). This metric is created to overcome a problem with the standard unigram precision which simply counts up the number of candidate translation words (unigrams) which occur in any reference translation and then divides by the total number of words in the candidate translation. In this case, however, candidate captions which repeat words with high probability, such as "the", can have a high-precision score just because they overgenerate words that are present also in the reference caption. For example if the candidate caption is "The the the the the the the" and the reference caption is "The cat is on the mat" then the standard unigram precision score will be 7/7. BLEU-1 is a modified unigram precision as it first counts the maximum number of times a word occurs in the reference caption, then it compute the "count clip" for each word, meaning it clips the total count of each word by its maximum reference count, and it sums all count clips in the candidate sentences. Finally, it divides this sum by the total number of distinct unigram (unclipped) in the candidate. The BLEU-1 score for the above example will be 2/7 since 2 is the maximum number of time that the word "the" appears in the reference sentence. The modified n-gram precision is computed similarly by counting all candidates n-gram and their corresponding maximum reference counts. Generally, BLEU scores are based on an average of unigram, bigram, trigram and 4-gram precision. The higher the n is, the more the metric is able to capture word order,but it also becomes more restrictive and constrained to the exact form of the reference. The strength of BLEU is that it is an intuitive and easy to compute metrics, however there are some drawbacks as the fact that all n-gram are treated equally, therefore minor errors as prepositions difference (such as "in" or "on") are penalized as heavily as more important words related to the content.

While BLEU focuses on computing the precision, i.e the number of candidate words that are present in the reference caption, ROUGE [Lin04] is a recall-oriented evaluation measure which, instead, focuses on how many of the reference captions n-gram are captured in the generate sentence. In particular, COCO-Caption evaluation framework uses a variant of ROUGE called ROUGE-L which is a F-measure based on the computation of the

[6] `https://github.com/ruotianluo/coco-caption`

Longest Common Subsequence between candidate and reference sentence. ROUGE-L estimates the similarity between the reference caption $r$ of length $m$ and the candidate caption $c$ of length $n$, as follows:

$$\mathrm{R} = \frac{LCS(c,r)}{m} \tag{6.8}$$

$$\mathrm{P} = \frac{LCS(c,r)}{n} \tag{6.9}$$

$$\mathrm{ROUGE\_L} = \frac{\left(1+\beta^2\right)\mathrm{RP}}{\mathrm{R}+\beta^2\mathrm{P}} \tag{6.10}$$

where $LCS(c,r)$ is the length of a longest common subsequence of $c$ and $r$ and $\beta$ is an hyper parameter usually set to favor recall($\beta = 1.2$). ROUGE has similar strengths and weaknesses as BLEU, however using LCS is an advantage as it automatically includes longest common n-grams, therefore it is not required to specify a predefined n-gram length.

The METEOR [DL14] metric is computed by aligning the candidate sentence and the reference sentence which corresponds to constructing a set of alignments by identifying all possible matches between the two sentences according to exact word matches, matches between word stems, synonyms or paraphrases. The final assignment is found as the largest subset of matches which maximize the number of covered words across both sentences and minimizes the number of chunks, contiguous and identically ordered tokens in the sentence pair. After this, the METEOR score for the obtained set of matches $m$ is computed as the harmonic mean of precision $P$ and recall $R$ :

$$\mathrm{R} = \frac{|m|}{\sum_k h_k\left(r\right)} \tag{6.11}$$

$$\mathrm{P} = \frac{|m|}{\sum_k h_k\left(c\right)} \tag{6.12}$$

$$\mathrm{F} = \frac{\mathrm{PR}}{\alpha\mathrm{P}+\left(1-\alpha\right)\mathrm{R}} \tag{6.13}$$

$$\mathrm{Pen} = \gamma\ \left(\frac{ch}{m}\right)^{\theta} \tag{6.14}$$

$$\mathrm{METEOR} = (1-\mathrm{Pen})\mathrm{F} \tag{6.15}$$

where $h_k(s)$ is the number of times an n-gram $w_k$ occurs in a sentence $s$, $m$ is the total number of matched words, Pen is penalty to account for gaps and differences in word order and $ch$ is the number of chunks. All parameters $\alpha, \gamma, \delta$ are tuned to maximize correlation with human judgments.

CIDEr [VLZP15] is a metric specifically made for Image Captioning and it measures consensus in image captions by performing a Term Frequency

Inverse Document Frequency (TF-IDF) weighting for each n-gram. As explained in the work of Vedantam et al. [VLZP15], CIDEr computes the TF-IDF weighting for each n-gram $w_k$ of a sentence $s$:

$$g_k(s) = \frac{h_k(s)}{\sum_{w_l \in \Omega} h_l(s)} \log \left( \frac{|I|}{\sum_{I_p \in I} \min(1, \sum_q h_k(r_{pq}))} \right) \tag{6.16}$$

where $\Omega$ is the vocabulary of all n-grams and $I$ is the set of all images in the dataset, $r_{pq}$ is a reference caption for image $p$ and reference number $q$ for multiple references. The first term computes the TF of each n-gram and gives more weights to n-grams which appear frequently in the sentence describing an image. The second term, instead, computes the IDF reducing the weight of n-grams which appear frequently in the dataset descriptions. The CIDEr score for n-gram of length n is computed using the average cosine similarity between the $g^n(s)$ vector formed by $g_k(s)$ corresponding to all n-grams of length $n$ and $||g_n(s)||$ is the magnitude of the vector $g_n(s)$, of both candidate and reference sentences ($S_i$ is the set of reference captions for image $i$):

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) \cdot g^n(s_{ij})}{||g^n(c_i)||\ ||g^n(s_{ij})||} \tag{6.17}$$

The final CIDEr score is computed as a weighted sum of the $\text{CIDEr}_n$ scores of n-gram of different lengths. The advantages of CIDEr are that it gives more weight to important n-grams and it has a higher correlation with human consensus scores compared to other metrics.

### 6.2.2 Experimental Setup

All of our models, training and evaluation blocks are developed in Python with PyTorch deep learning framework and PyTorch Lightning[7], a PyTorch wrapper which is useful to structure the code into functional modules and it handles the training procedure, for a faster and more scalable project deployment. As presented in 5.2, Multimodal GPT-2 is composed by a Convolutional Neural Network as feature extractor and a GPT-2 network which generates the caption from the multimodal embeddings created by the intermediate layers. For memory reasons, we use the smallest variant of GPT-2 pre-trained models, available in the Hugging Face transformers library[8]. This version has a hidden size equal to 768, 12 transformer decoder

[7] `https://pytorchlightning.ai/`

[8] `https://huggingface.co/transformers/`

layers, a 12 heads attention and 117 millions parameters. As feature extractor we use a ResNet50 as we do in our experiments on Fashion Tagging. The dimension of the output vector of the ResNet is 2048 which is mapped to 768 for each visual token that is fed as input to GPT-2.

Each input image is resized to a 224 square, normalized with the same mean and standard deviation of models trained on ImageNet and, during training, is randomly horizontally flipped with a 50% probability. Text tokens go only through stop words removal, i.e. removing the most common words, before being tokenized, as we want to keep preprocessing as light as possible to make our model more flexible and robust. The tokenizer we use is the GPT2Tokenizer, available in the Hugging Face transformer library, and its role is to convert the input text into a vector of integers where each integer correspond to a token in the dictionary, composed by 50257 words. During training, we unfreeze gradually the layers of the feature extractor at fixed milestones until they are all unfrozen. We use ADAM optimizer to update the parameters with the loss function presented in 5.2. A Plateau learning rate scheduler is responsible of decreasing the learning rate with a factor of 0.1 after observing that the validation loss stop decreasing for a number of epochs equal to the *patience* parameter, which we set to 3. The initial learning rate is automatically set using the learning rate finder offered by PyTorch Lightning. We perform early stopping on our training when the validation loss stops decreasing for 3 epochs. All our experiments have been carried out on a p2.xlarge instance available on Amazon Web Services with one NVIDIA K80 GPU.

### 6.2.3 Data

As we explain in chapter 3.4, we evaluate our approach for Fashion Captioning on the datasets Industrial Dataset 1 and Industrial Dataset 2. Table 3.5 in the same chapter highlights the most relevant differences between them, which are the number of items (77254 vs 1526 total images respectively) and average captions length (23.36 vs 52.77).

The datasets differ also on the types of tag annotations for each item. Industrial Dataset 1 items are annotated with one Class label, which represent the clothing item type, and several Attributes label which include other information such as clothing subcategories, gender, age group, color and concepts. Class and Attributes of each item are concatenated together to form the tag tokens inside the multimodal embedding. The tags for Industrial Dataset 2 are, instead, obtained by the concatenation of Attributes, which are labels including clothing categories, colors and more specific detail

as fabric and clothing parts, with Details annotations, which are short sentences that describe more fine-grained details such as pockets, zip closures, neckline shape and so on. Therefore, tags in Industrial Dataset 2 include more detailed information and are more numerous for each item, as the corresponding average of tag tokens per item is 22.30 while in Industrial Dataset 1 there are on average 9.06 tags per item.

### 6.2.4 Experiment 1: Multimodal Analysis

In this section we present a multimodal analysis of our model where we study its performance in different settings according to the type of input modality: only image (visual tokens), only tags (text tokens) or both (multimodal). The text tokens are additional information related to the item, in our experiments they correspond to tags such as class and attributes. With this study we want to show whether our model exploits one modality more than the other and how it performs with a single input type. We evaluate the performance on our two fashion industrial dataset which contain captions for all the images.

#### Industrial Dataset 1

In this subsection, we present our results on the multimodal analysis performed on Industrial Dataset 1. The models share the same architecture as Multimodal GPT-2 described in 5.2 and they differ only in the input type during both training and inference, where Image-Only GPT-2 corresponds to the one which uses only visual tokens from image features to generate the caption, Tags-Only leverages only text tokens (class, attributes or other textual information) associated to the item and Multimodal exploits both modalities.

| **Ind. Dataset 1 - Multimodal Analysis** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Image-Only GPT-2 | 54.27 | 41.9 | 33.59 | 27.76 | 27.31 | 53.71 | 204.50 |
| Tags-Only GPT-2 | 47.28 | 35.45 | 27.76 | 22.61 | 22.78 | 46.46 | 148.61 |
| Multimodal GPT-2 | **58.63** | **46.98** | **38.84** | **32.96** | **30.10** | **57.89** | **242.96** |

*Table 6.7: Industrial Dataset 1 - Multimodal Analysis*

As it can be seen from table 6.7, Multimodal GPT-2 outperforms the other models in all the metrics, indicating that using both modalities clearly seems to improve the quality of the generated captions. This could be due to the fact that leveraging both image and tags helps the model in understanding the fashion item better thanks to the features extracted directly

from the image and the additional knowledge coming from tags associated to the item. The results also suggest that the most important contribution to the caption inference seems to be coming from the visual features rather than tag; in fact, the metrics of Image-Only GPT-2 are all better that those of Tags-Only GPT-2. A possible explanation of this performance is that tags in Industrial Dataset 1 are coarse-grained, i.e. they generally refer to the class of the item and other high-level attributes such as colors, gender or age group, without more fine-grained details. Therefore, the caption generated using only this information is not able to correctly describe particular attributes of the garment or, when it does, it usually refers to characteristic that are frequent in a certain item category.

*(a)* **Tags**: *Swimwear swimwear tops ladies bikini sets yellow adult*
**Ground Truth**: *Fully lined, strapless bikini top with a frill trim at the top, removable inserts and no fasteners. The polyester content of the bikini top is recycled.*
**Multimodal**: *Fully lined off-the-shoulder bikini top with wide elastication at the top. The polyester content of the top is recycled.*
**Image-Only**: *Fully lined textured off-the-shoulder blouse in an airy organic cotton weave with elastication and a wide flounce at the top.*

*Figure 6.1: Industrial Dataset 1 - Multimodal vs Image-Only Captions*

Similar trends can be noticed also when analysing the captions generated in these different settings. In figure 6.1 are reported the captions generated by Multimodal GPT-2 and Image-Only GPT-2 for the bikini top shown in the image. The model that generates the captions only from visual features makes a mistake since the caption is describing a blouse, a type of upper garment which has similar features of the bikini in the image. On the contrary, the caption generated by the model which exploits both images and tags is correct as it describes a bikini top as reported in the ground truth caption. The difference is given by the fact that the tags tokens contain "swimwear" and "bikini" which helps the model in overcoming the ambiguity of the image.

In figure 6.2, the focus is on the different performance of the Multimodal and Tags-Only models. In this case, the target item is a turtleneck sweater but the associated tags are noisy since they contain wrong tokens as "t shirt" and "shirts". Misled by these noisy tags, the Tags-Only caption describes the item erroneously as a "Short-sleeved top". On the other hand, the Multimodal model describes the garment with correct visual attributes as

*(a)* **Tags**: *Tops t shirts kids girl 14 y shirts white*
**Ground Truth**: *Turtleneck sweater in a soft cotton-blend rib knit.*
**Multimodal**: *Long-sleeved turtleneck top in soft cotton jersey.*
**Tags-Only**: *Short-sleeved top in printed cotton jersey.*

*Figure 6.2: Industrial Dataset 1 - Multimodal vs Tags-Only Captions*

"long-sleeved" and "turtleneck" without, however, specifying the sweater and its fabric. This is an example of how having also visual features can improve the quality of the generated caption and that they are also necessary when dealing with noisy labels.

### Industrial Dataset 2

In this subsection we present the results of the multimodal analysis performed on our second Industrial Datasets, with the same settings as the previous section. What can be noticed from the results of table 6.8 is that Multimodal GPT-2 outperforms the other algorithms by a large margin, similar to what happens with Industrial Dataset 1. This is an additional proof of the fact that the model benefits from both input modalities as it can leverage visual features and additional information to construct a quality caption.

| **Ind. Dataset 2 - Multimodal Analysis** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Image-Only GPT-2 | 43.27 | 27.43 | 20.09 | 16.29 | 17.7 | 32.48 | 88.42 |
| Tags-Only GPT-2 | 48.82 | 35.35 | 28.51 | 24.43 | 21.91 | 40.44 | 143.67 |
| Multimodal GPT-2 | **54.68** | **42.66** | **36.46** | **32.83** | **25.89** | **47.08** | **236.44** |

*Table 6.8: Industrial Dataset 2 - Multimodal Analysis*

However, there is a significant difference with respect to the other dataset as, in this case, the model that exploits only tags information performs better in all the metrics than the Image-Only approach, while in the other dataset it was the opposite. There could be various explanation of this behaviour. First, the dataset size plays an important role as Industrial Dataset 1 has a training split composed by more than 50 thousands elements, while Industrial Dataset 2 has just above 1 thousands element in the training

set. Therefore, the feature extractor for Industrial Dataset 2 might not have enough data points to learn effectively what are the most important features needed to generate a caption and this might explain the worse performance of Image-Only GPT-2 in this situation. Furthermore, the difference in the number and type of tags tokens between the two datasets is also an important factor. In fact, the average number of tags associated to an item in Industrial Dataset 2 is 22.30 while in Industrial Dataset 1 is 9.06, as we specify in section 6.2.3, therefore in the former one, tags include more information that can be leveraged by the model.

*(a)* **Tags**: *Man Trousers Bermuda Shorts Beige Steppe Back welt pockets Double pleat Front pockets Zipper closure metal hooks drawstring*
**Ground Truth**: *The light summery touch of linen and cotton cover, pairs perfectly with the sporty style of these new Bermuda shorts, complete with a drawstring elastic waistband. The garment features a regular fit.*
**Multimodal**: *The crisp, light and compact texture of the twisted linen fabric combines perfectly with the sporty style of these Bermuda shorts, a summer wardrobe must-have. An elasticated drawstring waistband, pleat details and regular fit complete the garment.*
**Image-Only**: *The classic character of the Italian fit trousers is enriched with the excellence of American Pima cotton gabardine, a fabric with superior characteristics for the length, sheen and durability of its fibers. The new Italian fit remains close to the body, but at the same time, offers slightly softer lines than the more traditional fit.*
**Tags-Only**: *The refined qualities of American Pima cotton gabardine enrich the sporty character of these Bermuda shorts, a must-have piece for the summer. The drawstring waistband and double pleating details are paired with a modern leisure fit, offering soft proportions in the seat and a slightly low rise.*

*Figure 6.3: Industrial Dataset 2 - Multimodal Captions Comparison*

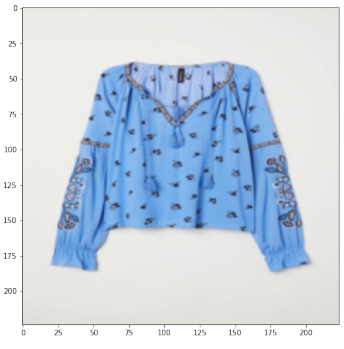
In figure 6.3 is shown a comparison between the captions generated by the models for one image of Bermuda shorts. The Multimodal and Tags-Only captions are very similar to the original one since they both recognise

the Bermuda shorts and also more fine-grained attributes as the summery and sporty style and the drawstring waistband, thanks to tags tokens which contain "Bermuda shorts" and "drawstring". On the other hand, Image-Only which exploits only visual features is not able to discriminate the Bermuda shorts characteristic and it describes the item as a pair of trousers. This shows the role of rich and fine-grained tags and how they can improve the quality of the generated captions, especially, when the dataset size is small and it is not easy to discriminate between image features.

#### Attention Visualisation

We perform an additional analysis of our multi modal approach by visualizing the attention layers inside GPT-2 architecture. As we explain in chapter 2.2.4, GPT-2 is composed by a stack of Transformer decoder blocks where each of them has a layer of Masked Self-Attention and a Feed Forward layer. The role of the attention layers is to compute how much each input token is related to the others and use this information to produce the output embedding by focusing on the input tokens which are more relevant to it. We analyse this layer in our model to better understand from which type of input modality (visual or tag tokens) our model is exploiting more information each time it is decoding a new word. One way of performing this analysis is by visualizing how the attention shift each time a new token is decoded. In figure 6.5, we show this type of visualisation for our Multimodal GPT-2 model caption generation for the sample on figure 6.4, focusing on the attention for three decoded tokens at the last Transformer block. For each figure, the words on the left column are from the generated caption, split in different tokens by the GPT-2 tokenizer using Byte-Pair Encoding, while the right column contains the multimodal input, with visual tokens and tag tokens. Visual tokens are the output of the Remapper block which convert the visual features into a fixed number, in this case 5, of visual embedding tokens of the same dimension as word tokens, as we explain in chapter 5.2. Tag tokens are the actual tag annotations, processed by GPT-2 tokenizer. The lines connecting tokens represent the level of attention between each output-input token pair and the colored rectangles correspond to the attention for each attention head, which are 12 for gpt2-small.

It is interesting to notice from figure 6.5 that the model seem to benefit from both types of input modalities, varying its focus according to the current decoded token. For example in figure 6.5a, the model attention for token "Long" of the n-gram "Long-sleeved blouse" seem to attend almost equally to all input tokens, which probably means that its exploiting visual



*(a)*

**Tags**: *Shirts Blouses blouses ladies divided shirts blouses blue adult*

**Ground Truth**: *Wide V-neck blouse in a crinkled weave with an embroidered pattern and decorated with embroidered ribbon. Double ties with tassels at the neck and long sleeves with narrow elastication and a flounce at the cuffs.*

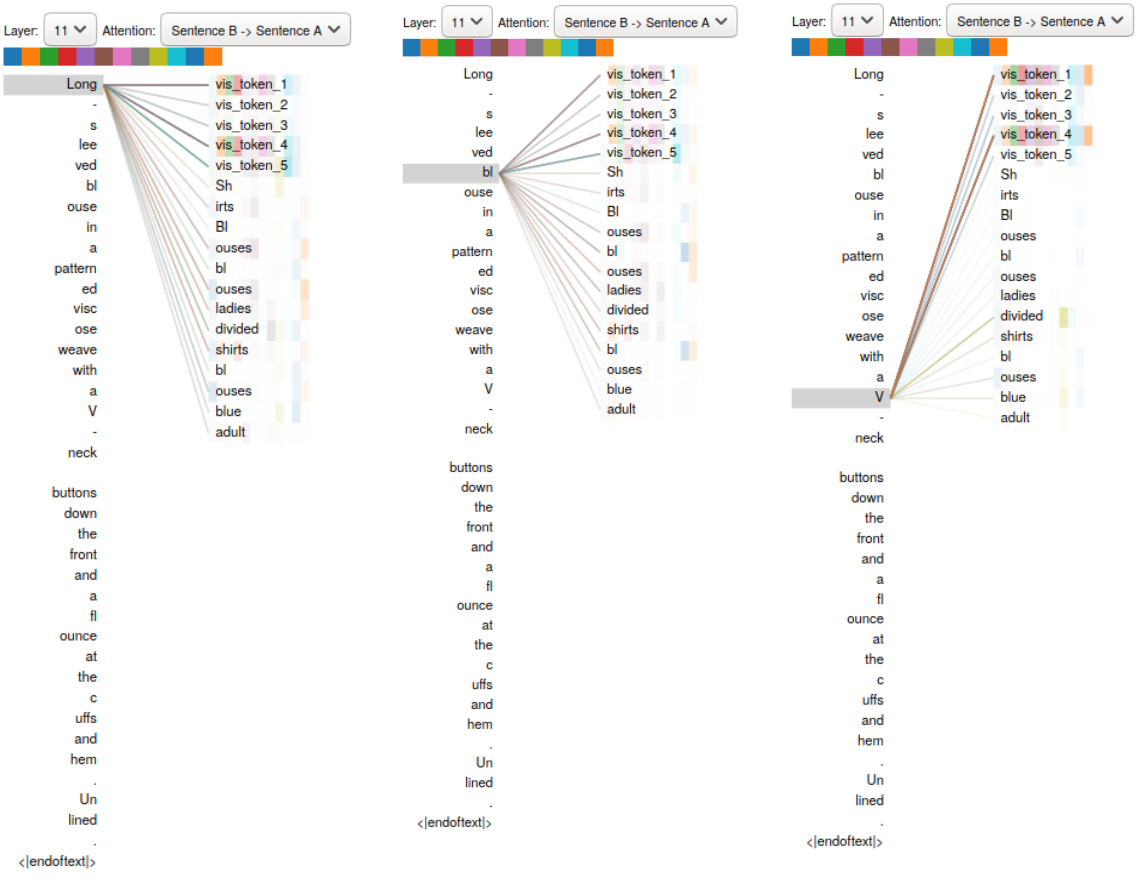
**Multimodal GPT-2**: *Long-sleeved blouse in a patterned viscose weave with a V-neck buttons down the front and a flounce at the cuffs and hem. Unlined.*

*Figure 6.4: Sample from Industrial Dataset 1*

features to infer the long sleeves attribute and it's also benefiting from tags such as "blouses" and "shirts" to better understand the type of garment it's going to describe. Same reasoning can be applied to the attention visualization for token "bl" of word "blouse" in figure 6.5b. In figure 6.5c it is possible to observe a different behaviour for token "V" of "V-neck" as, in this case, the attention focuses mostly on the visual tokens rather than the tag tokens. One reason for this can be that at this point the model has already returned the tokens which describe the garment type and it's now shifting its attention to fine-grained attributes by "looking" at the visual tokens.

Furthermore, it is also worth noticing that the model seem to attend to all visual tokens and this suggests that mapping the visual features into different subspaces is helping the model to distinguish different type of attributes. There seems to be also a form of redundancy as there are two visual tokens, the first and the fourth, with similar attention head values and this could mean that less visual tokens are actually needed. We present additional experiments with different visual tokens numbers in appendix. A.2.

To sum up, the attention visualisation in the GPT-2 layers confirms the same trend that we observe by analysing the evaluation metrics and caption generated by the different model, where the model which rely on the multimodal input benefits from both type of inputs and this leads it to outperform the models with a single input modality.




*(a) Attention for token "Long"* *(b) Attention for token "bl"* *(c) Attention for token "V"*

*Figure 6.5: GPT-2 Self Attention Visualisation*

### 6.2.5 Experiment 2: Comparison with Baselines

#### Baselines

In order to evaluate the performance of our model with respect to other approaches we use two different baseline algorithms: one that uses both image and text tokens as input and another one, based on *Show, Attend and Tell* paper, presented in Related Word (chapter 4), which relies only on image features.

The first baseline, that we refer to as Multimodal LSTM, is a model based on the encoder-decoder paradigm with a Convolutional Neural Network (a ResNet as in our model) and a LSTM as decoder. The LSTM is a Recurrent Neural Network, as explained in 2.2.2, made specifically for modelling sequence data and used frequently in many text generation ap-

proaches. The input size and hidden size of the LSTM are both set to 768, as the hidden size of our GPT-2. The multimodal embedding composed by visual and tag tokens is built in the same way that we do for our model, as we describe in 5.2, and is fed as input to the LSTM, using the same vocabulary and tokenizer. Therefore, this baseline only differs from our approach in the decoder, where we use GPT-2 instead of a LSTM. Since GPT-2 can leverage a pre-training on 40GB dataset of text while LSTM is not pre-trained on a large corpora, we make the comparison more fair by giving the token embeddings and language model linear head weights of pre-trained GPT-2 to the LSTM, so that it can better discriminate tokens among all the 50257 words in the vocabulary.

The second baseline, based on the *Show, Attend and Tell* [XBK+15] image captioning paper, is taken from a image captioning repository available online[9]. It is a model based on a encoder-decoder architecture with a CNN as encoder and a LSTM as decoder, incorporating also an attention mechanism that allows to focus on different parts of the image each time a new word is decoded, as described in 4.2.1. Since this approach support only image as input, we compare this baseline with our model trained only with image visual tokens, without relying on additional textual tags. As in the first baseline we "upload" the token embeddings and language model linear head weights of pre-trained GPT-2 to the LSTM to make the comparison more fair.

We choose to compare our results with these baseline because other state-of-the-art approaches for Image Captioning, as [LHL+19], train also object detectors using bounding boxes to extract features related to the entities in the image and, in our datasets, we don't have bounding boxes that identify clothing items or parts in an image. Furthermore, the task of our models is to generate a caption which describe a single garment with its attributes and details and not different objects in an image and their relations as in a general captioning scenario.

### Industrial Dataset 1

In this subsection we report the results of our approach compared to the baselines on our Industrial Dataset 1.

*Multimodal LSTM Baseline*

In table 6.9, we compare Multimodal GPT-2, our model explained in 5.2, and the baseline Multimodal LSTM which has the same structure with

[9] `https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning`

a LSTM as decoder instead of GPT-2. Observing the results, it is possible to state that having GPT-2 as decoder improves the performance by a good margin in all the metrics than having a LSTM as decoder.

| **Ind. Dataset 1 - Multimodal** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Multimodal LSTM | 50.29 | 37.79 | 29.29 | 23.44 | 24.22 | 48.76 | 150.83 |
| Multimodal GPT-2 | **58.63** | **46.98** | **38.84** | **32.96** | **30.10** | **57.89** | **242.96** |

*Table 6.9: Industrial Dataset 1 - Comparison with Multimodal Baseline*

One explanation of this improvement is to be found in the self-attention block of the Transformer layers inside GPT-2 which allows the model to attend to all the input words and focus on the most important ones, partially solving the long-term dependency problems that affect Recurrent Neural Networks. Thanks to these layers, GPT-2 learns the correlations between the visual tokens and tag tokens inside the multimodal input and the output tokens of the generated caption and this lead to the generation of caption with better quality and more attention to fine-grained details. We also report an example of the caption generated by both models in figure 6.6. In the first image both captions are overall correct, apart from some errors in details as "rounded hem" for Multimodal GPT-2 and "with a motif" in Multimodal LSTM, however the model with GPT-2 recognises also the "viscose crepe" fabric and "flounced sleeves" which is similar to "frilled"[10] of the ground truth. Similarly, in the second image, GPT-2 identifies correctly the item as "Joggers" and other details as "side pockets" and "ribbed hems" while LSTM confuses it for "Leggings" and provides less details. This example also shows that even if tags tokens contain the word "leggings", even repeated, our model is still able to infer correctly that the item is a pair of joggers from the visual feature without being negatively affected by tag tokens. Therefore, it seems that Multimodal GPT-2 is more able to infer correctly the item type and build a more correct and precise caption than its counterpart with LSTM.

*Show, Attend and Tell Baseline*

The other baseline that we want to compare with our approach is one based on the work presented by Xu et al.[XBK+15] in which they propose a Image Captioning framework based on a CNN encoder and a LSTM decoder. The novelty of this approach is that they apply a visual attention mechanism on the image features which, at each decoding step, attends to the most relevant features to produce the next word in the caption. Our model doesn't

[10] `https://papertheorypatterns.com/pages/frill-and-flounce`

*(a)* **Tags**: *Tops tops short sleeves ladies short sleeve green adult everyday fashion*
**Ground Truth**: *Top in an airy viscose crepe weave with a small frill around the neckline short frilled sleeves that slope downwards at the sides and a small opening and button at the back of the neck.*
**Multimodal GPT-2**: *Top in viscose crepe jersey with short flounced sleeves and a rounded hem.*
**Multimodal LSTM**: *Short-sleeved top in jersey with a motif.*

*(b)* **Tags**: *Trousers Leggings Pants trousers kids girl 14 y trousers leggings grey*
**Ground Truth**: *Joggers in a soft fine-knit viscose blend with elasticated ribbing and a drawstring at the waist side pockets and tapered legs with ribbed hems.*
**Multimodal GPT-2**: *Joggers in soft sweatshirt fabric with an elasticated drawstring waist side pockets and tapered legs with ribbed hems. Soft brushed inside.*
**Multimodal LSTM**: *Leggings in soft patterned cotton jersey with an elasticated waist.*

*Figure 6.6: Industrial Dataset 1 - Captions generated by Multimodal GPT-2 and Multimodal LSTM baseline*

have this kind of attention mechanism and we perform this experiment to understand if our approach reaches similar perfomances to one that exploits it. Since this baseline works with only image as input, for a fair comparison we confront its results with those of our model trained and tested with only images as input and not with the multimodal embedding.

| **Ind. Dataset 1 - Only Images** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Show, Attend and Tell | **56.09** | **44.65** | **36.65** | **30.85** | **28.84** | **56.27** | **224.13** |
| Image-Only GPT-2 | 54.27 | 41.9 | 33.59 | 27.76 | 27.31 | 53.71 | 204.50 |

*Table 6.10: Industrial Dataset 1 - Comparison with Show, Attend and Tell baseline*

What can be noticed from table 6.10 is that the results from the baseline are all higher than our approach, meaning that having a visual attention mechanism as the one described in [XBK+15], seems to improve the perfomances of caption generation. Therefore, this can be seen as a future improvement of our current work and it could require more investigations and experiments to include this kind of mechanism in our approach. One possible limitation is that a network such as GPT-2, based on Transformers, performs the operations on the input tokens in parallel (which is actually

one of the most important advantages since it reduces training time), differently from RNN which performs operation sequentially and the attention mechanism operates between the recurrent steps. Therefore, working on introducing such mechanism on our approach is not so trivial and would require further research.

After comparing also the captions produced by both models, we notice that the quality of the generated captions is very similar, with sometimes some differences in more fine-grained details where the Show, Attend and Tell baseline is more precise, probably thanks to its attention mechanism.

*(a)* **Ground Truth**: *Dress in a soft fine-knit viscose blend a round neckline 3/4-length sleeves and a gently flared skirt. Unlined.*
**Image-Only GPT-2**: *Short dress in ribbed jersey with a turtle neck and long sleeves.*
**Show, Attend and Tell**: *Short dress in ribbed jersey with a round neckline and 3/4-length sleeves Unlined.*

*(b)* **Ground Truth**: *Set with a top and dungarees in soft cotton. Top in patterned jersey with a press-stud on one shoulder. Dungarees in a crinkled weave with flounces at the front elastication at the back of the shoulder straps press-studs at the crotch and down the legs (not in sizes 1-3Y) and elasticated hems.*
**Image-Only GPT-2**: *Set with a long-sleeved bodysuit pants and accessory in soft organic cotton jersey. Bodysuit with snap fasteners at gusset. Pants with wide foldover ribbing at waist and ribbed hems.*
**Show, Attend and Tell**: *Set with a long-sleeved top and pair of dungarees in soft organic cotton. Bodysuit in a patterned weave with elastication and a frill at the top concealed press-studs at the crotch and down the legs and elasticated hems.*

*Figure 6.7: Industrial Dataset 1 - Captions generated by Image-Only GPT-2 and Show, Attend and Tell baseline*

In the first image in figure 6.7, Show, Attend and Tell baseline describes also the details "round neckline" and "3/4-length sleeves" while our model misses them and mistakes the neckline for a "turtleneck". In the other example, the baseline recognises correctly the set made of a top and dungarees with specific details while our model describes a set with a bodysuit and pants. Therefore, while, in general, the captions of the two models are very similar, these two example suggests that the reason why the baseline

performs better than our approach is because of it higher attention to certain details.

### Industrial Dataset 2

In this subsection we report the results of our approach compared with the baselines on Industrial Dataset 2.

*Multimodal LSTM Baseline*

The results reported in figure 6.11 suggests that, as in Industrial Dataset 1, our Multimodal GPT-2 performs better than its counterpart with LSTM, according to all the metrics. Furthermore, in this case the gap between the two algorithms is even greater than that of the previous dataset, reported in figure 6.9. There are several differences between the two dataset and the model architectures that could explain this trend. First, this dataset has an average number of tags of 22.30, i.e. each item is labelled, on average, with 22 tag tokens, while Industrial Dataset 1 average is much lower with 9.06. With the higher number of input tokens in the multimodal embedding, the LSTM model might not be able to embed all the relevant information into a unique state and use it as a context to generate the next work, similar to the long-term dependencies problem that notoriously affects RNNs. GPT-2, on the other hand, has several multi-head self-attention layers which allows it to attend at all the tokens simultaneously with the key, query and value vectors. This difference might be crucial when dealing, as in this case, with a multimodal input of this length. Similarly, Industrial Dataset 2 captions are longer (52.77 average length) than Industrial Dataset 1 (23.36) and, for the same problems related to learning long-term dependencies, the quality of the generated captions might suffer from this.

| **Ind. Dataset 2 - Multimodal** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Multimodal LSTM | 39.07 | 25.16 | 18.80 | 15.55 | 15.77 | 30.09 | 64.26 |
| Multimodal GPT-2 | **54.68** | **42.66** | **36.46** | **32.83** | **25.89** | **47.08** | **236.44** |

*Table 6.11: Industrial Dataset 2 - Comparison with Multimodal Baseline*

In figure 6.8, we report an example of the captions generated by the two algorithms. In this case Multimodal LSTM is not able to correctly classify the garment as it recognises a sweater instead of a shirt, which, instead, Multimodal GPT-2 does correctly with other details as the "button-down collar" and the "basic fit" with "straight lines". The LSTM model seems to have problem at correlating the high number of tags and the visual features

while our approach with GPT-2 is more capable of discriminating the most important visual and text token to produce the correct caption.

*(a)* **Tags**: *Man Shirts Button Down Shirt Blue Azure Basic-fit Button-down collar Cuffs mother pearl buttons Patch chest pocket Placket closure mother pearl buttons*
**Ground Truth**: *The season's colors enrich the pure cotton striped Oxford, a classic menswear fabric. This button-down collar shirt is paired with a basic fit, which offers regular lines that follow the body through the shoulders and along the torso.*
**Multimodal GPT-2**: *The colors of the season enrich the classic two-tone check pattern of this button-down collar shirt, a menswear must-have. Pure cotton poplin shirt features a basic fit, with comfortable, straight lines.*
**Multimodal LSTM**: *The excellence of [BRAND] materials enriches the classic style of the lightweight and cotton sweater. The classic and versatile design of the top is enriched with the small striped cotton ribbon embroidered with a stripe of shiny monili. The fit is comfortable and relaxed.*

*Figure 6.8: Industrial Dataset 2 - Captions generated by Multimodal GPT-2 and Multimodal LSTM baseline*

*Show, Attend and Tell Baseline*

| **Ind. Dataset 2 - Only Images** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Show, Attend and Tell | 38.62 | 23.94 | 17.40 | 14.19 | 15.39 | 30.04 | 86.081 |
| Image-Only GPT-2 | **43.27** | **27.43** | **20.09** | **16.29** | **17.7** | **32.48** | **88.42** |

*Table 6.12: Industrial Dataset 2 - Comparison with Show, Attend and Tell baseline*

In this dataset, there is an important inversion of trends with respect to the comparison between the Show, Attend and Tell baseline and our Image-Only model on Industrial Dataset 1 (table 6.10) as the baseline performs worse than our model in all the metrics. This suggest that the visual attention mechanism of the baseline is not enough to overcome the inherent challenges of this dataset. Despite the fact that in this experiment we are

not using the tags token as input, since both models only exploits visual features, the problem related to the caption length is still relevant, where LSTM might not have enough complexity to model captions of this length and rich of details. In addition, the size of this dataset (1225 items in the training set) is much smaller than the other one (54387) and LSTM with the attention mechanism might not have enough data points to correctly infer which are the most relevant features in the image. In figure 6.9, for example, our model produces a caption which describes correctly the outerwear garment of the image, with both correct details as the "ideal to wear in warm weather" and wrong ones as "shiny monili embroidery on the side". On the other hand, the LSTM with attention mechanism is not able to recognise the correct type of garment since it refers to it as a sweatshirt. These results could suggests that our model can be the best choice when it comes to fine-tuning a small dataset, since it leverages GPT-2 pre-training and its Transformers layers with self-attention blocks.

*(a)* **Ground Truth**: *Versatile materials and comfortable proportions reinterpret the style of this outerwear, inspired by classic raincoats. The lightness and fluidity of the gabardine exterior is paired with the technical and casual look of the water-resistant microfiber, a lightweight fabric ideal for the summer season. Comfortable and straight lines are offered by the fit.*
**Image-Only GPT-2**: *The excellence of [BRAND] materials enriches this new outerwear jacket inspired by the classic flavor of men's shirts. Shiny monili embroidery on the side adds a precious touch and introduces the iconic element of the [BRAND] collections. Ideal to wear in warm weather in the city or on a trip, this outerwear jacket features a practical fabric pouch to be stored in the smallest of spaces.*
**Show, Attend and Tell**: *The quality of [BRAND] materials enhances the Travelwear line, dedicated to moments of relaxation and free time. A sporty flavor of Activewear is interpreted by the lightweight cotton French terry sweatshirt with a drawstring hood and personalized two-way zipper closure with the Solomeo logo.*

*Figure 6.9: Industrial Dataset 2 - Captions generated by Image-Only GPT-2 and Show, Attend and Tell baseline*

### 6.2.6 Experiment 3: Caption and Tag Generation

In this section we present the results of the experiments we perform with our TagCaptioner GPT-2 model, presented in chapter 5.3, which combines our approaches for Tagging and Captioning into a unique solution which is able to generate both tags and captions.

#### Industrial Dataset 1

Regarding our first industrial dataset, we present the results on both the Captioning and Tagging tasks. For the former, we confront the performance of TagCaptioner with the results obtained by our Multimodal GPT-2 approach, explained in chapter 5.2, which generate the caption using both visual features and the associated tags and the baseline with LSTM decoder. As it is possible to notice from table 6.13, TagCaptioner performance doesn't reach the same level of Multimodal GPT-2, which is a result that we expect given the fact that Multimodal leverages the ground truth tags while TagCaptioner is using the tags that it generates by visual features, therefore there is a performance gap between these two approaches. Nevertheless, we observe that the quality of the captions generated by TagCaptioner is good as it is better than the baseline with LSTM which leverages the ground truth tags. We notice few errors in the caption generation that can be traced back to incorrect or incomplete generated tags. For example, in figure 6.10, the clothing item is a sports top and Multimodal GPT-2 is able to provide the correct caption thanks also to the help the ground-truth tags which contain the tag "Sport". On the other hand TagCaptioner doesn't return any tags which is related to the sport details and, therefore, also the generated description is missing this particular detail which, apparently, is not able to discriminate correctly from the image features. This represents also a limitation of this model, since it depends on the quality of the generated tags to achieve good performances also in the caption generation.

| **Ind. Dataset 1 - TagCaptioner for Captioning** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Multimodal LSTM | 50.29 | 37.79 | 29.29 | 23.44 | 24.22 | 48.76 | 150.83 |
| Multimodal GPT-2 | **58.63** | **46.98** | **38.84** | **32.96** | **30.10** | **57.89** | **242.96** |
| TagCaptioner GPT-2 | 53.08 | 40.71 | 32.40 | 26.65 | 26.54 | 52.67 | 196.90 |

*Table 6.13: Industrial Dataset 1 - TagCaptioner for Fashion Captioning*

We also evaluate the performance of TagCaptioner on the Fashion Tagging task and we compare it with the results that we obtain with our Double-Head TagNet model, presented in chapter 5.1, on Industrial Dataset 1. As

*(a)* **Tags**: *Sport wear tops short sleeves sport ladies short sleeve black adult*
**Ground Truth**: *Sports top in fast-drying sweatshirt fabric with half-length raglan sleeves and a raw-edge hem. Slightly longer and rounded at the back.*
**Multimodal GPT-2**: *Wide sports top in fast-drying functional fabric with short cap sleeves.*
**TagCaptioner GPT-2**: *Top in organic cotton jersey with a ribbed neckline in a contrasting colour and a gently rounded hem.*
**Generated Tags**: *Tops kids tops t shirts short sleeves black tops*

*Figure 6.10: Industrial Dataset 1 - TagCaptioner Example*

| **Industrial Dataset 1 - TagCaptioner for Tagging** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Class | | | Attributes | | | | | | | |
| | accuracy | | | precision@k | | | recall@k | | | recall@sample_k | map |
| | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | | |
| Top-Popular | 10.1 | 27.1 | 41.2 | 50.1 | 38.2 | 31.0 | 9.20 | 21.0 | 28.8 | 29.58 | 30.2 |
| Double-Head TagNet | **87.61** | **98.30** | **99.36** | **97.09** | **87.78** | 75.56 | **18.46** | **49.78** | 70.67 | 74.34 | **83.74** |
| TagCaptioner | 85.19 | 97.57 | 99.04 | 96.01 | 86.73 | **75.57** | 18.24 | 49.17 | **70.68** | **74.67** | 83.69 |

*Table 6.14: Industrial Dataset 1 - TagCaptioner for Fashion Tagging*

it can be noticed from table 6.14, TagCaptioner reaches similar results of those obtained by the TagNet, which its only task is generating tags, also slightly outperforming it in few metrics.

### Industrial Dataset 2

We perform the same experiment, for the Fashion Captioning task, also for Industrial Dataset 2. This dataset tags annotations, as we explain in section 3.3, are different from those of DeepFashion and Industrial Dataset 1 which are divided into two types: mutually-exclusive classes and multi-label attributes. In this dataset, tags are divided into a type which is similar to attributes as it contains labels related to categories, colors and more specific detail as fabric and clothing parts, and another type which is a set of short description of fine-grained details as pockets, zip closures, neckline shape and so on. In order to generate these type of tags we perform two tasks, each of them with the same approach of attribute prediction in the Image Tagging task, computing the loss function as a Binary Cross Entropy after passing the scores through a Sigmoid activation function (Tag Loss Functions chapter 5.1), one for predicting the attributes and the other for the detail sentences.

As expected, the performances of TagCaptioner are worse than Multimodal GPT-2, however, in this case the gap between them is more wide

than in the case of Industrial Dataset 1. This is probably related to the fact that, as we explain in 6.2.4, tags in this dataset plays a much important role in the generation of captions, for the small dataset size and the fact that they contain also fine-grained detail that the model can leverage. Therefore TagCaptioner generated tags quality is still lower than that of ground-truth tags and this affects the model performances on caption generation. As an example, in figure 6.11, we notice that the generated tags contain the token Sweater while the item is crew-neck T-shirt. In this case Multimodal GPT-2 is able to identify correctly the garment as a crew-neck T-shirt while TagCaptioner is misled by the wrong token "Sweater" in the generated tags and, therefore, it generates a caption for a sweater. This is another example of how the performance of TagCaptioner can be affected by the generated tags quality. Even so, TagCaptioner performs better than the baseline with LSTM which leverages ground-truth tags, proving again the qualities of our approach based on GPT-2.

| **Ind. Dataset 2 - TagCaptioner for Captioning** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| Multimodal LSTM | 39.07 | 25.16 | 18.80 | 15.55 | 15.77 | 30.09 | 64.26 |
| Multimodal GPT-2 | **54.68** | **42.66** | **36.46** | **32.83** | **25.89** | **47.08** | **236.44** |
| TagCaptioner GPT-2 | 43.79 | 28.37 | 21.2 | 17.43 | 18.36 | 33.58 | 107.04 |

*Table 6.15: Industrial Dataset 2 - TagCaptioner for Fashion Captioning*

| **Industrial Dataset 2 - TagCaptioner for Tagging** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Task | precision@k | | | recall@k | | | recall@sample_k | map |
| | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 | | |
| Attributes Prediction | 93.29 | 84.95 | 65.97 | 15.54 | 42.47 | 54.97 | 59.34 | 64.55 |
| Details Prediction | 76.21 | 53.25 | 41.46 | 21.07 | 41.36 | 52.19 | 48.61 | 56.26 |

*Table 6.16: Industrial Dataset 2 - TagCaptioner for Fashion Tagging*

For a complete evaluation, we also report the performance of the model in the tag generation of attributes (1087 labels) and details (593 short descriptions treated as single labels). In table 6.16, we show the results of TagCaptioner on the two tasks that the model performs simultaneously with the Caption Generation. What it can be noticed from the results is that the model seems to perform better on the Attributes Prediction task than the generation of Details (the short description labels), despite the larger number of attributes than the one of details. The reason for this is that Details are less easier to discriminate given the fact that they represent very fine-grained details and, therefore, the model might not have enough data points in the dataset to better learn these features.

*(a)* **Tags**: *Man Topwear Long Sleeve T-Shirt Blue Navy Blue Crew-neck Flat pressed hem cuffs Slim fit*

**Ground Truth**: *The refined qualities of [BRAND] materials enrich the long sleeve crew-neck T-shirt, a menswear essential. Lightweight cotton jersey offers a comfortable, subtle texture that pairs perfectly with the slim fit.*

**Multimodal GPT-2**: *The crew-neck T-shirt, a year-round component of the male wardrobe, combines the qualities of lightweight cotton jersey with the form-fitting lines of a slim-fit, which remains close to the body both through the chest and shoulders.*

**TagCaptioner GPT-2**: *The virgin wool, cashmere and silk new sweater combines the excellence of [BRAND] materials with details inspired by the Active world. Raglan-style sleeves and the rib knit neckband, triangle insert, cuffs and bottom band characterize the garment's sporty flavor, while the contrast color edge of the cuffs adds a sophisticated finishing touch. The proportions are regular both through the chest and shoulders.*

**Generated Tags**: *Man Blue Knitwear Crewneck Sweater Black Navy Blue Black Rib knit cuffs and bottom band Plain stitch Crew-neck with triangle Raglan sleeve Crew-neck in rib knit*

*Figure 6.11: Industrial Dataset 2 - TagCaptioner Example*

# Chapter 7

# Conclusion and Future Work

In this chapter, the key outputs and contribution of our research work are going to be presented and discussed. This work stems from the current challenges related to the maintenance and enrichment of online catalogues where millions of items and the corresponding images needs to be correctly tagged with labels and descriptions in order to make them more appealing and easy to find for the users. Specifically, the purpose of this work is to tackle these problems in the fashion domain, where almost every fashion company is now exploiting e-commerce websites to sell its products online. Given the fact that each one of these online catalogue can contain thousands of images and new items are continuously added for new seasons, collections or trends it is crucial that all this vast amount of data is correctly annotated but, at the same moment, it becomes extremely difficult and time-consuming for human annotators to do it. Therefore, with our work we explore different solutions in the state-of-the-art and propose our own novel approaches to generate tags and descriptions for a fashion item by exploiting visual features extracted directly from its image.

## 7.1 Outputs and Contributions

We study and propose approaches for two tasks in particular: Fashion Tagging and Fashion Captioning. We test the performances of our models with respect to different datasets, both public and private, and we compare them with baselines from state-of-the-art approaches in order to have a complete understanding of their capabilities and limitations. The task of Fashion Image Captioning, in particular, has not received a lot of attention so far from the research community and, therefore, with our work we also hope to help the research in this direction.

For the Fashion Tagging task, our main research contribution is the insight we gain from the analysis and comparison with other approaches. We observe that the current state-of-the-art approach [LL18] in the public large-scale dataset DeepFashion [LLQ+16] relies on a landmark attention branch to improve the performances in the Category and Attribute prediction task. Given the fact that landmarks, which correspond to a set of key-points on the clothes structure, are a type of annotation that is rarely available in real-word datasets, we prefer to build a model that doesn't exploit this extra information. We then compare the results of our model with the current state-of-the-art and, as explained in chapter 6.1.5, we show that our approach performs better than the baseline with landmark attention, suggesting that using this type of mechanism based on the additional landmarks annotations doesn't seem to be necessary to improve the Category and Attributes prediction tasks.

Regarding Fashion Captioning, we propose a novel approach based on GPT-2 language model to generate captions for an image. GPT-2 architecture has surprised many for its ability to generate coherent and complex text but, to the best of our knowledge, is yet to be used in the context of generating text given features extracted from an image and, therefore, with our work we want to provide a contribution to this research question. We carry out a performance study and baselines comparison on two industrial datasets to test our model in a real-world industrial setting (chapter 6.2.5). Another important feature that distinguishes our approach from the current state-of-the-art models is that it is not based on object detectors which recognise entities as in the general Image Captioning but it focuses on the entire image features to extract details and attributes.

Furthermore, we consider the possibility of improving the quality of the generated caption by leveraging not only visual feature but also additional textual information, if available. This information can be metadata or tags that are associated to the item and we combine them with the visual features in a multimodal embedding which is fed as input to our model. We perform a multimodal analysis to investigate whether our model relies on one input modality more than the other in different settings, with both quantitative analysis on evaluation metrics and qualitative by observing the generated captions and visualizing the attention layers inside our GPT-2 model.

Finally, we present also a model capable of generating simultaneously tags and caption by combining our approaches for Fashion Tagging and Captioning into a unique system. We carry out a performance study also for this approach on the industrial datasets comparing its results with those of our previous models which perform, separately, Tagging and Captioning.

## 7.2 Limitations

Even though several contributions are brought to the research field by this work, limitations are still present and need to be addressed, as it is going to be discussed in this section. First of all, in our experiments on Fashion Tagging and the comparison with baselines, we can't compare with the actual performance of one of the state-of-the-art model since no pre-trained model is available and we observe a discrepancy between the reported results and the one we reproduce. Therefore, even though our reproduced results should be reliable as we use the code and setting released by the author, the original model provided by the authors would clear any doubts.

Another limitation is brought by the fact that we can't compare our results with other works on Fashion Captioning as this particular task has yet to be thoroughly studied by the research community. We find a work by Yang et al. [YZJ+20], recently released in August 2020, which focus on Fashion Captioning but we are not able to compare our approach with this work as the authors say, in reply to our questions, that they can't release the code yet.

## 7.3 Future Work

A lot remains to be studied and experimented following this research work, in particular in the field of Fashion Captioning which is yet to be explored thoroughly and the applications of GPT-2 language model and Transformer-based models for Image Captioning.

Restricting the focus in particular on this research work, a possible next step is to include a layer of visual attention in our Multimodal GPT-2 approach for captioning giving the model the capability to not only to give more relevance to certain visual or textual tokens, but also to learn which are the most important visual features to select each time a word is decoded. We observe in our experiment with the Show, Attend and Tell baseline (chapter 6.2.5) that this type on visual attention, applied to an LSTM, seems to improve the quality of the caption and the attention to more fine-grained details, therefore, it looks like a promising approach to try. Furthermore, we could also explore new ways of mapping visual features into visual tokens, using more complex architectures than linear fully-connected layers to learn more precise sub-spaces mappings.

We identify another possible direction in the research on how to improve the quality of the generated caption by increasing their structure complexity and the use of polished words. For example, after observing that the

captions of Industrial Dataset 1 have a simpler structure and less refined words than Industrial Dataset 2 we try to perform a Caption Style Transfer Learning by controlling the text generation, as it is done in [DML+20], of a model trained on mixed dataset with captions coming by the two different datasets. The goal is to "transfer" the complex structure and refined words of Industrial Dataset 2 to generated captions for the other dataset, characterised by captions with a less complex structure. We are currently working on this approach.

Another important future work is to add the possibility of translating the generated caption in different language as online catalogues often need to offer the same product in different countries with the description in the proper language.

# Bibliography


[Ala18] Jay Alammar. The illustrated transformer. http://jalammar.github.io/illustrated-transformer/, Jun 2018.

[Ala19] Jay Alammar. The illustrated gpt-2 (visualizing transformer language models). http://jalammar.github.io/illustrated-gpt2/, Aug 2019.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[CBYRO17] Charles Corbiere, Hedi Ben-Younes, Alexandre Rame, and Charles Ollion. Leveraging weakly annotated data for fashion image retrieval and label prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[CFL+15] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[DL14] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[DML+20] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2020.

[Dwi19] Priya Dwivedi. Understanding and coding a resnet in keras. https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33, Mar 2019.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[KBFT19] Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. Supervised multimodal bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950*, 2019.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[LHL+19] Zhou Luowei, Palang Hamid, Zhang Lei, Hu Houdong, Corso Jason, and Gao Jianfeng. Unified vision-language pre-training for image captioning and vqa. *arXiv preprint arXiv:1909.11059*, 2019.

[Lin04] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[LL18] Jingyuan Liu and Hong Lu. Deep fashion analysis with feature map upsampling and landmark-driven attention. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[LLQ+16] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Phi20] Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, Jun 2020.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[Raj19] Bharath Raj. A simple guide to semantic segmentation. https://www.topbots.com/semantic-segmentation-guide/, May 2019.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[RWC+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[SZC+19] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[Vin] Sagar Vinodababu. a-pytorch-tutorial-to-image-captioning. https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning.

[VLZP15] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Pro-*

*ceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[XBK+15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[Yun] Choi Yunjey. Pytorch image captioning tutorial. https://github.com/yunjey/pytorch-tutorial.

[YZJ+20] Xuewen Yang, Heming Zhang, Di Jin, Yingru Liu, Chi-Hao Wu, Jianchao Tan, Dongliang Xie, Jue Wang, and Xin Wang. Fashion captioning: Towards generating accurate descriptions with semantic rewards, 2020.

# Appendix A

# Additional Tables

## A.1 Tagging

### Hyperparameter Tuning

We report the result of the hyperparameter tuning we mention in chapter 6.1.2. The model that perform better on the validation set has learning rate equal to 1e-3 and batch size 8.

| Parameters | | Class | Attributes | | | | | |
|---|---|---|---|---|---|---|---|---|
| lr | batchsize | accuracy | p@k | | | r@k | | |
| | | | top-1 | top-3 | top-5 | top-1 | top-3 | top-5 |
| 0.01 | 4 | 74.02 | 89.85 | 73.95 | 61.28 | 17.06 | 41.71 | 57.11 |
| 0.01 | 8 | 75.62 | 92.06 | 77.82 | 64.46 | 17.46 | 43.59 | 60.06 |
| 0.001 | 4 | 76.35 | 91.75 | 78.19 | 65.15 | 17.44 | 44.14 | 60.74 |
| 0.001 | 8 | 79.93 | 93.59 | 81.08 | 67.40 | 17.79 | 45.86 | 62.94 |
| 0.0001 | 4 | 79.22 | 89.59 | 71.54 | 57.99 | 16.99 | 40.26 | 53.96 |
| 0.0001 | 8 | 78.19 | 86.84 | 6861 | 54.85 | 16.44 | 38.54 | 51.02 |
| 0.00001 | 4 | 60.03 | 64.90 | 45.77 | 38.06 | 12.20 | 25.57 | 35.40 |
| 0.00001 | 8 | 56.95 | 59.76 | 41.08 | 34.12 | 11.19 | 22.84 | 31.73 |

*Table A.1: Industrial Dataset 1 - Hyperparameter tuning*

## A.2 Captioning

### Visual Tokens Experiments

Below we present the results of experiment with different visual tokens number, i.e. the number of embedding in which the visual features are mapped. From the results on Industrial Dataset 1 it seem that the number of visual tokens doesn't affect the model performances while for Industrial Dataset

2 performance improves with an higher number of visual tokens. The reasons for this behavior are probably to be found on the differences in term of dimension and annotations between the dataset but further experiments would be required to obtain more reliable results.

| **Ind. Dataset 1 - Visual Tokens Experiments** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Visual Tokens | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| 1 | 57.69 | 46.23 | 38.12 | 32.39 | 29.52 | 58.29 | 266.85 |
| 5 | 57.61 | 46.22 | 38.17 | 32.38 | 29.63 | 57.32 | 240.77 |
| 10 | 57.61 | 46.29 | 38.47 | 32.95 | 29.56 | 57.40 | 260.84 |

*Table A.2: Industrial Dataset 1 - Experiments with different number of visual tokens*

| **Ind. Dataset 2 - Visual Tokens Experiments** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Visual Tokens | BLEU-1 | BLEU-2 | BLEU-3 | BLEU-4 | METEOR | ROUGE-L | CIDEr |
| 1 | 39.97 | 24.03 | 16.92 | 13.34 | 16.22 | 30.41 | 80.94 |
| 5 | 43.27 | 27.43 | 20.09 | 16.29 | 17.7 | 32.48 | 88.42 |
| 10 | 42.15 | 27.25 | 20.26 | 16.56 | 17.53 | 32.82 | 97.49 |

*Table A.3: Industrial Dataset 2 - Experiments with different number of visual tokens*

# Appendix B

# Generated Tags and Caption Examples

## B.1 Generated Tags

In this section we show the class and attributes tags generated by our model from DeepFashion test set.

*(a)* **GT Class**: *Blouse*
**GT Attributes**: *chiffon, floral, floral print, print*
**Predicted Class**: *Blouse*
**Top-5 Attributes**: *print, floral, shirt, floral print, ruffle*


*(b)* **GT Class**: *Blouse*
**GT Attributes**: *lace, long sleeve, shopping, sleeve, woven*
**Predicted Class**: *Blouse*
**Top-5 Attributes**: *sleeve, chiffon, button, shirt, long sleeve*

*Figure B.1: Blouses*

*(a)* **GT Class**: *Dress*
**GT Attributes**: *bejeweled, leopard, leopard print , maxi , print*
**Predicted Class**: *Dress*
**Top-5 Attributes**: *leopard , leopard print , print , maxi , animal*

*(b)* **GT Class**: *Dress*
**GT Attributes**: *heathered , trapeze*
**Predicted Class**: *Dress*
**Top-5 Attributes**: *trapeze , heathered , high-slit , knit , knit trapeze*

*Figure B.2: Dresses*

*(a)* **GT Class**: *Jeans*
**GT Attributes**: *distressed , skinny*
**Predicted Class**: *Jeans*
**Top-5 Attributes**: *skinny , distressed , ripped , wash , acid*

*(b)* **GT Class**: *Jeans*
**GT Attributes**: *dark , fit , slim , wash*
**Predicted Class**: *Jeans*
**Top-5 Attributes**: *slim , wash , fit , dark , classic*

*Figure B.3: Jeans*

*(a)* **GT Class**: *Cardigan*
**GT Attributes**: *hooded*
**Predicted Class**: *Cardigan*
**Top-5 Attributes**: *print , tribal , knit , dolman , hooded*

*(b)* **GT Class**: *Cardigan*
**GT Attributes**: *knit , striped*
**Predicted Class**: *Jacket*
**Top-5 Attributes**: *cropped , crop , striped , faux , knit*

*Figure B.4: Cardigans*

*(a)* **GT Class**: *Jacket*
**GT Attributes**: *fringed , leather , suede*
**Predicted Class**: *Jacket*
**Top-5 Attributes**: *suede , fringe , faux suede , fringed , faux*

*(b)* **GT Class**: *Jacket*
**GT Attributes**: *floral , floral print , print*
**Predicted Class**: *Blazer*
**Top-5 Attributes**: *floral , floral print , print , daisy , daisy print*

*Figure B.5: Jackets*

*(a)* **GT Class**: *Leggings*
**GT Attributes**: *capri , heathered , performance , pocket*
**Predicted Class**: *Leggings*
**Top-5 Attributes**: *suede , fringe , faux suede , fringed , faux*

*(b)* **GT Class**: *Leggings*
**GT Attributes**: *floral , floral print , print*
**Predicted Class**: *Leggings*
**Top-5 Attributes**: *floral , print , floral print , pink , rose*

*Figure B.6: Leggings*

*(a)* **GT Class**: *Poncho*
**GT Attributes**: *buttoned , refined*
**Predicted Class**: *Cardigan*
**Top-5 Attributes**: *dolman , hooded , knit , batwing , dolman sleeve*

*(b)* **GT Class**: *Poncho*
**GT Attributes**: *fringe*
**Predicted Class**: *Poncho*
**Top-5 Attributes**: *fringe , print , knit , crochet , fringed*

*Figure B.7: Poncho*

*(a)* **GT Class**: *Shorts*
**GT Attributes**: *belted , classic , flat-front*
**Predicted Class**: *Shorts*
**Top-5 Attributes**: *belted , polo , chino , cotton , classic sleeve*

*(b)* **GT Class**: *Shorts*
**GT Attributes**: *belted , chino*
**Predicted Class**: *Shorts*
**Top-5 Attributes**: *flat front , chino , flat , cargo , twill*

*Figure B.8: Shorts*

*(a)* **GT Class**: *Tee*
**GT Attributes**: *crew , crew neck , pocket*
**Predicted Class**: *Tee*
**Top-5 Attributes**: *pocket , tribal , print , pattern , neon sleeve*

*(b)* **GT Class**: *Tee*
**GT Attributes**: *crew , crew neck , linen*
**Predicted Class**: *Tee*
**Top-5 Attributes**: *heathered , knit , crew , sleeve , crew neck*

*Figure B.9: Tee*

*(a)* **GT Class**: *Kimono*
**GT Attributes**: *embellished , island , print , tropical*
**Predicted Class**: *Kimono*
**Top-5 Attributes**: *floral , print , floral print , chiffon , lace*

*(b)* **GT Class**: *Kimono*
**GT Attributes**: *chiffon , embroidered*
**Predicted Class**: *Kimono*
**Top-5 Attributes**: *embroidered , fringe , print , tassel , fringed*

*Figure B.10: Kimono*

## B.2 Generated Captions

### Industrial Dataset 1

In this section we show the captions generated by our model. We provide random examples for different categories of clothing from the test set of Industrial Dataset 1.

*(a)* **Ground Truth**: *Rain jacket in windproof water-repellent functional fabric with closed seams at critical points. Detachable hood with elasticated sides a zip down the front and elastication at the cuffs and hem. Reflective details. Unlined. The jacket has a water-repellent coating without fluorocarbons.*
**Multimodal GPT-2**: *Lightly padded jacket in water-repellent functional fabric with a detachable lined hood stand-up collar and zip down the front. Front pockets elasticated cuffs and folded seams and reflective details at the back and on the hood. Fleece lining. The jacket has a water-repellent coating without fluorocarbons.*

*(b)* **Ground Truth**: *Lightly padded parka in thick woven cotton fabric with appliqués. Stand-up collar and detachable hood. Zip and wind flap at front with snap fasteners. Front pockets with flap and snap fastener vent at back and quilted lining.*
**Multimodal GPT-2**: *Padded parka in woven fabric with a soft brushed finish. Detachable pile-lined hood with faux fur trim zip and wind flap at front with snap fasteners and front pockets with flap and snap fastener. Inner ribbing at cuffs. Quilted lining.*

*Figure B.11: Jackets & Parka*

*(a)* **Ground Truth**: *Treggings in stretch denim-look jersey. Elasticized waistband mock fly mock front pockets and regular back pockets.*
**Multimodal GPT-2**: *Treggings in stretch denim-look jersey. Elasticized waistband mock fly mock front pockets and regular back pockets.*

*(b)* **Ground Truth**: *Trousers in stretch twill with a high elasticated waist and a concealed zip in one side. Fake welt pockets at the back and slim legs with creases and short slits at the hems.*
**Multimodal GPT-2**: *Ankle-length suit trousers in woven fabric with a regular waist concealed hook-and-eye fastener and zip fly. Side pockets fake back pockets and tapered legs with creases.*

*Figure B.12: Trousers*

*(a)* **Ground Truth**: *Fitted dress in ribbed jersey made from a cotton blend with a V-neck decorative buttons down the front and a flare to the hem.*
**Multimodal GPT-2**: *Short fitted dress in sturdy jersey with a V-neck buttons down the front and long sleeves. Unlined.*

*(b)* **Ground Truth**: *Short dress woven in a patterned linen blend with a round neckline opening with a button at the back of the neck and short sleeves. Seam and detachable tie belt at the waist and a gently flared skirt with side pockets. Flounced trim around the sleeves and hem. Lined.*
**Multimodal GPT-2**: *Short dress in a patterned crêpe weave with a round neckline and an opening with a concealed button at the back of the neck. Short flounced sleeves a seam at the waist and a flared skirt. Unlined.*

*Figure B.13: Dresses*

*(a)* **Ground Truth**: *Ankle boots in suede with pointed toes and block heels. Leather insoles and rubber soles. Heel height 2 1/4 in.*
**Multimodal GPT-2**: *Ankle boots with pointed toes elastic gores in the sides and a loop at the back. Fabric linings and insoles and rubber soles. Covered heels 6 cm.*

*(b)* **Ground Truth**: *Trainers in nylon and imitation suede with a lightly padded edge and tongue lacing at the front and patterned soles.*
**Multimodal GPT-2**: *Trainers with lacing at the front mesh linings and insoles and rubber soles.*

*Figure B.14: Shoes*

*(a)* **Ground Truth**: *Metal hoop earrings with a pendant. Diameter approx. 5 cm.*
**Multimodal GPT-2**: *Thin metal hoop earrings. Diameter 2 cm.*

*(b)* **Ground Truth**: *Sunglasses with plastic and metal frames and UV-protective tinted mirrored lenses.*
**Multimodal GPT-2**: *Sunglasses with plastic frames and UV-protective tinted lenses.*

*Figure B.15: Accessories*

*(a)* **Ground Truth**: *Triangle bikini top with removable inserts and narrow adjustable shoulder straps that cross at the back. Wide elasticated hem and no fasteners.*
**Multimodal GPT-2**: *Triangle bikini top with removable inserts and narrow adjustable shoulder straps that cross at the back. Adjustable fastening and an elastic trim at the hem.*

*(b)* **Ground Truth**: *Print-patterned swimming trunks with an elasticated drawstring waist and lined front.*
**Multimodal GPT-2**: *Swim trunks with a printed pattern. Elasticized drawstring waistband and lined front..*

*Figure B.16: Swimwear*

*(a)* **Ground Truth**: *Suede shoulder bag with a detachable shoulder strap zip at the top and zipped inner compartment. Lined. Size 4x20x30 cm.*
**Multimodal GPT-2**: *Shoulder bag in grained imitation leather with a narrow detachable shoulder strap and flap with a magnetic fastener. Two inner compartments one with a zip. Lined. Size 6x15x20 cm.*

*(b)* **Ground Truth**: *Shoulder bag in glittery imitation leather with a flap with a concealed hook and loop fastening. Narrow shoulder strap with a press-stud at one end. Lined. Size approx. 4x9.5x13 cm.*
**Multimodal GPT-2**: *Small shoulder bag in glittery imitation leather with a zip at the top and narrow shoulder strap with a press-stud at one end. Lined. Size 5x12x20 cm.*

*Figure B.17: Bags*

*(a)* **Ground Truth**: *Sports tights in fast-drying functional fabric with ribbing at the waist and a concealed key pocket in the waistband.*
**Multimodal GPT-2**: *Sports tights in fast-drying functional fabric with wide ribbing to hold in and shape the waist. Concealed mesh key pocket in the waistband.*

*(b)* **Ground Truth**: *Short-sleeved sports shirt in fast-drying breathable functional fabric made partly from recycled polyester.*
**Multimodal GPT-2**: *Short-sleeved sports top in fast-drying breathable functional fabric. The polyester content of the top is recycled.*

*Figure B.18: Sports*

*(a)* **Ground Truth**: *5-pocket skirt in washed stretch cotton denim with worn details an adjustable elasticated waist (in sizes 8-12Y) zip fly and button and a raw-edge hem.*
**Multimodal GPT-2**: *Skirt in washed stretch denim with an adjustable elasticated waist (in sizes 8-12Y) zip fly and button fake front pockets and real back pockets.*

*(b)* **Ground Truth**: *Pleated skirt in woven fabric with a high waist concealed side zip and asymmetric hem. Lined.*
**Multimodal GPT-2**: *Calf-length skirt in a pleated weave with an elasticated waist. Lined.*

*Figure B.19: Skirts*

*(a)* **Ground Truth**: *Cardigan in soft fine-knit cotton with a V-neck buttons at front and contrasting elbow patches.*
**Multimodal GPT-2**: *Cardigan in soft fine-knit cotton with a V-neck buttons down the front long sleeves and ribbing around the neckline cuffs and hem.*

*(b)* **Ground Truth**: *Sweater in a soft fluffy knit with long sleeves motif at front and slits at sides. Slightly longer at back.*
**Multimodal GPT-2**: *Fine-knit sweater in a soft viscose blend with a motif at front. Dropped shoulders and long sleeves.*

*Figure B.20: Cardigans & Sweaters*

*(a)* **Ground Truth**: *Shirt in an organic cotton weave with a collar buttons down the front a yoke with a pleat at the back and long sleeves with buttoned cuffs.*
**Multimodal GPT-2**: *Straight-cut shirt in a soft viscose weave with a collar buttons down the front long sleeves with narrow buttoned cuffs and a rounded hem.*

*(b)* **Ground Truth**: *Shirt in washed superstretch denim with embroidered details on one sleeve and at hem. Collar snap fasteners at front and long sleeves with snap fasteners at cuffs. Chest pockets with snap fastener. Yoke at back with pleat.*
**Multimodal GPT-2**: *Shirt in washed stretch denim with a collar buttons down the front flap chest pockets with a button and long sleeves with buttoned cuffs.*

*Figure B.21: Shirts*

### Industrial Dataset 2

In this section we show the captions generated by our model. We provide random examples from the test set of Industrial Dataset 2.

*(a)* **Ground Truth**: *The excellence of BRAND materials elevates the style of these timeless cigarette trousers, an emblem of elegant, feminine design. The trousers offer proportions, which pair perfectly with the lightweight stretch cotton cover fabric: regular in the seat and slim down the legs. A thin, rounded leather belt completes the garment with an elegant contrast detail.*
**Multimodal GPT-2**: *The cotton twill's lightweight and dry texture pairs perfectly with the proportions of the new boy fit trousers, reworking the classic cigarettes silhouette with a modern twist. The fit is comfortable both in the seat and along the leg.*

*Figure B.22: Trousers*

*(a)* **Ground Truth**: *The deconstructed blazer's style is defined by the combination of the elegant linen, wool and silk hopsack with the casual patch pockets detail. The proportions that distinguish BRAND style are offered by the fit: tapered through the chest to highlight the male physique with slightly pronounced shoulders.*
**Multimodal GPT-2**: *New natural fabrics enrich the casual style of blazer with patch pockets. Lightweight linen fabric updates the classic flavor of the chalk stripe pattern with a subtle slubbed note. The proportions that distinguish BRAND style are offered by the fit: tapered through the chest to highlight the male physique with slightly pronounced shoulders.*

*Figure B.23: Blazer*

*(a)* **Ground Truth**: *Precious craftsmanship and refined BRAND materials elevate the casual style of the Travelwear line, dedicated to moments of relaxation and free time. Soft and lightweight, this cotton French terry sweatshirt showcases two different colors to create a striped pattern in the season's colors. On the neckline, a V-insert embroidered with fine rows of monili updates a classic Sportswear detail with a shiny touch.*
**Multimodal GPT-2**: *Refined BRAND materials marry with the casual style of the Travelwear line, dedicated to moments of relaxation and free time. A sporty flavor of Activewear is interpreted by the lightweight cotton French terry sweatshirt with a precious touch: the V-insert embroidered with fine rows of shiny monili adds a sparkling BRAND touch. The fit is comfortable and relaxed.*

*Figure B.24: Sweatshirt*

*(a)* **Ground Truth**: *A combination between the casual flavor of the materials and the feminine lines defines the style of this dress with an elegant touch. The striped cotton poplin's lightweight texture pairs perfectly with the fluid, unstructured silhouette, characterized by soft proportions in the bust and the handkerchief hem skirt that is slightly elongated on the side. A precious detail completes the garment: an acetate and silk fabric insert on the sleeves is embroidered with rows of shiny monili to recreate a classic "roll tab" with the iconic embellishment of the BRAND collections.*
**Multimodal GPT-2**: *A combination between casual materials and feminine lines defines the style of this dress. The cotton poplin's lightweight texture enhances this garment's fluid volume, gathered at the waist with a removable fabric belt. Behind the neck, shiny monili embroidery completes this dress with the iconic embellishment of the BRAND collections.*

*Figure B.25: Dress*

*(a)* **Ground Truth**: *Refined materials enrich traditional knitwear patterns with delicate shiny details. The lightweight cotton, linen and silk sweater alternates classic cable knitting with ribbing obtained with a special technique, creating an elegant raised texture effect. In addition, the small paillettes incorporated into the yarn complete the garment with a precious feminine touch. Regular proportions with slightly dropped shoulders are offered by the fit.*
**Multimodal GPT-2**: *Inspired by artisanal weaves and traditional net stitches, knitting of this new cotton Feather yarn Cable Net sweater creates an elegant effect of texture and proportions. A netting technique forms a three-dimensional geometric pattern with a texture that enhances the yarn's full, lightweight and comfortable traits. The fit is comfortable with slightly dropped shoulders.*

*Figure B.26: Sweater*

*(a)* **Ground Truth**: *Feminine inspiration and refined materials define the style of this new pure silk Pongee shirt. The ultra lightweight, fluid and semi-glossy fabric is printed with a floral-inspired design that combines the season's colors to create an elegant three-dimensional effect. Relaxed, straight proportions with dropped shoulders and slightly elongated lines in the back are offered by the silhouette.*
**Multimodal GPT-2**: *Precious feminine materials define the style of this pure silk pongee shirt. The ultra-lightweight, fluid and semi-glossy fabric is printed with a floral design that combines the season's colors to create an elegant three-dimensional effect. The garment features a regular fit and slightly dropped shoulders.*

*Figure B.27: Shirt*

*(a)* **Ground Truth**: *For travel or everyday moments of leisure, these swimming trunks are a must-have summer accessory. The lightweight nylon fabric, with its smooth and semi-matte look, is enriched by the sporty striped pattern in the season's colors. A drawstring waist, practical pockets and stylish button-flap front pocket detail complete the garment.*
**Multimodal GPT-2**: *For travel or everyday moments of leisure, this swimsuit is a must-have accessory for the summer. Dyed in the season's colors, the lightweight nylon fabric stands out against the contrasting side bands. A sporty touch of the cargo pockets and the drawstring waist completes this accessory.*

*Figure B.28: Swimwear*

*(a)* **Ground Truth**: *An elegant and refined accessory, the new pocket square completes the style of the jackets and suits in the collection with a modern touch.*
**Multimodal GPT-2**: *An elegant and refined accessory, the new pocket square completes the style of the jackets and suits in the collection with a modern touch.*

*Figure B.29: Accessory*

*(a)* **Ground Truth**: *The classic two-tone sporty striped pattern enriches the cotton jersey T-shirt, a must-have menswear piece. A cotton insert visible along the edges of the neckband completes the garment with a contrast detail. The slim fit offers tapered lines that are close through the chest.*
**Multimodal GPT-2**: *The crew-neck T-shirt, a year-round component of the male wardrobe, combines the qualities of lightweight cotton jersey with the form-fitting lines of a slim-fit, which remains close to the body both through the chest and shoulders.*

*Figure B.30: T-shirt*