



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Transfer Learning Analysis of Fashion Image Captioning Systems

Master of Science in
Computer Science and Engineering - Ingegneria Informatica

Author: **Filippo Colombo**

Student ID: 940304

Advisor: Prof. Paolo Cremonesi

Co-advisors: Federico Sallemi, Umberto Pietroni

Academic Year: 2020-21

Abstract

Modern deep learning technologies generate text samples of outstanding quality, and when combined with a visual feature extractor, they accurately describe the subjects or the scenes depicted in images at the cost of time-consuming training procedures over a large number of data samples. However, the performance of these models and the quality of the generated texts drop when they process input samples that depart from the distribution of the training data. In this thesis work, we analyze the generalization capabilities of systems able to automatically generate captions of images, trying to overcome variations and perturbations in the input samples and still achieve high-quality descriptions. Specifically, we tackle this problem in the fashion domain, where clothing samples have several details, making the task of describing garments expensive and only feasible for experts. Besides, online catalogues continuously grow and change when new releases of fashion items enter the market, increasing the need for a robust model able to overcome the variations in new clothing samples, saving the time, energy, and resources required to train a new model to describe the last releases of said items.

We will design a pre-training procedure that, together with a noise generation strategy, improves the performance of fashion image captioners on unseen distributions of data. We will then observe that by performing a final adaptation stage of the pre-trained model using a very narrow set of target samples, the fashion image captioner achieves competitive performance and high-quality captions compared to the model extensively trained on the target source. Additionally, we will propose a novel Transformer-based approach that leverages the generative performance of the GPT-2 language model along with the Vision Transformer (ViT) and BERT encoders to generate text from an image of a garment and its metadata. To train this architecture, we will consider an additional contrastive objective to align the embeddings of the two input modalities; we will analyze how it reflects on the representation learned by the model and compare the results with baseline works. Finally, we will perform a user study to evaluate the quality of the description of clothing samples generated by image captioner systems pre-trained through our approach.

Keywords: Image Captioning, Transfer Learning, Fashion, NLP, Deep Learning

Sommario

Le moderne tecnologie di deep learning generano campioni di testo di qualità notevole e, se combinate con un componente che processa input visuali, descrivono accuratamente i soggetti o le scene rappresentate in immagini al costo di lunghe procedure di training su molti dati. Tuttavia, le prestazioni di questi modelli e la qualità dei testi generati diminuiscono quando si processano campioni che si discostano dalla distribuzione dei dati di training. In questo lavoro di tesi, analizziamo le capacità di generalizzazione di sistemi in grado di descrivere immagini automaticamente, cercando di limitare gli effetti provocati da variazioni nei dati in input e ottenere comunque descrizioni di alta qualità. Nello specifico, affrontiamo questo problema nel campo della moda, dove i prodotti sono caratterizzati da parecchi dettagli, rendendo il compito di descrivere i capi costoso e fattibile solo da esperti. Inoltre, i cataloghi online crescono e cambiano continuamente quando entrano nel mercato nuovi articoli, aumentando il bisogno di un modello robusto in grado di gestire le variazioni nei nuovi dati, risparmiando tempo, energia e risorse necessarie per formare un nuovo modello che descriva gli ultimi arrivi dei prodotti di cui sopra.

Progettiamo una procedura di pre-training che, insieme a una strategia di generazione del rumore, migliora le prestazioni dei sistemi che generano descrizioni di immagini di moda appartenenti a nuove distribuzioni di dati. Osserviamo che eseguendo una fase di adattamento finale del modello pre-addestrato utilizzando un insieme molto ristretto di dati appartenenti al dominio di destinazione, si ottengono prestazioni competitive e descrizioni di qualità comparabili al modello ampiamente addestrato sul dominio di destinazione. Inoltre, proponiamo un nuovo approccio basato su Transformer che sfrutta le prestazioni generative del modello GPT-2 insieme a Vision Transformer (ViT) e BERT per generare testo da un'immagine di un capo e dai suoi metadati. Per allenare questa architettura, consideriamo un ulteriore funzione obiettivo per allineare gli embedding delle due modalità di input; analizziamo come questo si riflette sulle rappresentazioni imparate dal modello e confrontiamo i risultati con altri algoritmi. Infine, eseguiamo un sondaggio per valutare l'opinione di utenti sulla qualità delle descrizioni dei capi di abbigliamento generati da sistemi sviluppati secondo il nostro approccio.

Parole chiave: Descrizione di immagini, Transfer Learning, Moda, NLP, Deep Learning

Contents

Abstract	i
Sommario	iii
Contents	v
1 Introduction	1
1.1 Context: Transfer Learning in Fashion Image Captioning	1
1.2 Scenario and Problem Statement	1
1.3 Contributions	2
1.4 Structure of Thesis	3
2 Background	5
2.1 Transfer Learning	5
2.1.1 The need for Transfer Learning	5
2.1.2 Definition	6
2.1.3 Approaches in Deep Learning	7
2.1.4 Applications in Computer Vision and NLP	8
2.1.5 Related areas: Multi-task and Contrastive Learning	9
2.2 Image Captioning	11
2.2.1 Convolutional Neural Networks	11
2.2.2 Natural Language Processing	13
2.2.3 LSTM	14
2.2.4 The Transformer	16
2.3 Summary	23
3 Related work	25
3.1 Transfer Learning	25
3.1.1 Target task adaptation	25

3.1.2	Contrastive Learning	27
3.2	Image Captioning	29
3.2.1	Show, Attend, and Tell	29
3.2.2	Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks	30
3.2.3	Image tagging and captioning for fashion catalogues enrichment . .	31
4	Datasets	33
4.1	Fashiongen	33
4.2	Private data sources	35
4.2.1	Industrial Dataset 1	35
4.2.2	Industrial Dataset 2	37
4.2.3	Industrial Dataset 3	39
4.3	Dataset Comparison	41
5	Pre-Training for Fashion Image Captioning	45
5.1	Goal and Requirements	45
5.2	Data preparation	46
5.3	Imbalance problem	47
5.4	Noise generation	49
5.4.1	Fashion taxonomies	50
5.4.2	Generation of forged samples	53
5.5	Fine-tuning	54
6	Contrastive Learning: approach and model architecture	57
6.1	Vision-Text Multi-Encoder Decoder	57
6.1.1	Extension of the Transformer decoder block	58
6.2	Contrastive objective	59
6.2.1	Multi-objective optimization	60
6.2.2	Pre-alignment	61
7	Experiments and Results	63
7.1	Evaluation metrics	63
7.2	Experimental setup	66
7.3	Model comparison	66
7.4	Transfer Learning analysis	70
7.5	Vision-Text Multi-Encoder Decoder	78
7.6	User study	89

8 Conclusion and Future Work	97
8.1 Outputs and contributions	97
8.2 Limitations	99
8.3 Future Works	99
Bibliography	101
A Taxonomy	105
B Additional Tables	107
C Examples of generated captions	109
List of Figures	113
List of Tables	117
Acknowledgements	119

1 | Introduction

1.1. Context: Transfer Learning in Fashion Image Captioning

This thesis work focuses on the research areas of Transfer Learning and Image Captioning applied to the fashion industry. The former consists of all the approaches whose goal is to leverage the knowledge acquired from the solution of past problems to solve new ones; the latter, instead, aims to generate syntactically and semantically correct descriptions of the subject or scene depicted in images. More precisely, Image Captioning systems exploit both Computer Vision and Natural Language Processing techniques to understand the context of visual sources and handle textual data to interpret and reproduce the human language.

We study and analyze the application of Transfer Learning techniques to Image Captioning systems to define a training procedure and a model architecture to implement fashion image captioners.

1.2. Scenario and Problem Statement

Fashion firms that want to make their products available online require detailed descriptions of each clothing item in their catalogues to provide the users with all the necessary information about the garment they could buy through an e-commerce platform.

The process of describing each clothing product is expansive and requires a professional stylist to look at the images and write a textual description taking into account all the details regarding the types of garments, colors, fabrics, and all the other features that distinguish a fashion item. In addition, online catalogues continuously increase and change when new releases of fashion items enter the market, causing a fashion expert to spend more time doing this mechanical task.

Modern deep learning technologies generate text samples of outstanding quality¹, and when combined with a visual feature extractor, they precisely describe the subjects or the scene depicted in images. These models can automatically provide such descriptions of clothing items at the cost of expensive implementations: they require powerful computational resources to address the time-consuming training procedures and a large amount of data. Moreover, the performance and the quality of the generated texts drop when the model needs to process input samples that depart from the distribution of the data used during training.

In the fashion industry, the process of describing clothing samples is expensive, and it is crucial to have high-quality descriptions of the products a fashion firm wants to sell online to attract more effectively the attention of customers. Moreover, online catalogues may change frequently: it would be beneficial to have a system able to overcome the variations in the distribution of the data samples and still generate high-quality descriptions of clothing products. It could save the time, energy, and resources required to train a new model from scratch.

1.3. Contributions

To train models able to overcome possible variations in the distribution of the data source in input, we propose a pre-training procedure that exploits Transfer Learning paradigms and the generation of noisy samples to improve the generalization capabilities of Image Captioning systems applied to the fashion domain. In this way, the fashion image captioners can still achieve high-level performance without explicit training on a target dataset or with a limited adaptation stage.

Our training procedure does not depend on the architecture of the image captioner, so we first compare the current state-of-the-art architecture in the “general” Image Captioning task with a recent promising model specifically designed for the fashion domain. We then perform a transfer learning analysis using both public and private datasets, showing how the performance of trained fashion image captioners varies according to the input distribution. Pre-training a model through our approach with a narrow conditioning stage on the target domain allows obtaining effective fashion image captioners even when constrained by resource requirements like limited computational time or memory, as well as few or costly training samples.

Additionally, we propose a novel Transformer-based architecture [Vas+17] that lever-

¹<https://openai.com/blog/better-language-models/>

ages the generative performance of the GPT-2 [Rad+19] language model, along with the recent Vision Transformer (ViT) [Wu+20] and BERT [Dev+19] encoders to process a multi-modal input, applied to the task of Image Captioning. In section 6, we present our architecture and the training options that arise when considering a self-supervised alignment between the embeddings of the two input modalities: we carry out a performance study showing how the alignment of the embeddings reflects on the representation learned by the model and how it improves the performance of fashion image captioners compared to baseline works.

Finally, we perform a user study to evaluate the quality of the description of clothing samples generated by image captioner systems pre-trained through our approach.

1.4. Structure of Thesis

In this section, we provide a brief description of the following chapters.

- Chapter 2 presents the background technologies and findings related to this thesis to provide the reader with the knowledge to understand our work.
- Chapter 3 outlines the relevant works of the research community both on Transfer Learning and Image Captioning.
- In chapter 4, we provide a description of the datasets used to train and evaluate the models involved in our analysis.
- Chapter 5 presents all the aspects and decisions that lead us to our pre-training approach.
- In chapter 6, we introduce a new architecture for Fashion Image Captioning and its training options.
- Chapter 7 presents the results of our pre-training procedure and architecture and compares them with baseline works.
- Chapter 8 summarizes our findings and the limitations of this thesis work and provides possible future research directions.

2 | Background

This chapter provides the reader with an overview of topics and challenges faced in this thesis work. Starting with Transfer Learning, we discuss the motivations worth exploring this field, how to approach it with practical applications, and the research areas related to it that are connected with our work. Later the focus moves to Image Captioning. We present the task and the technologies used in Computer Vision and Natural Language Processing to build image captioning systems.

2.1. Transfer Learning

Transfer learning, or *knowledge transfer*, is the idea of leveraging the knowledge acquired from the solution of past problems to solve new related ones.

Humans have this inherent ability that is used each time they need to learn a new task, by exploiting the skills acquired to solve similar problems. Machines instead are historically designed to solve problems in isolation, by tackling a task in a specific domain of interest. It is up to the designer of the algorithm to find a way to model or exploit previous knowledge concerning similar tasks or domains.

We start this section by briefly describing the reasons why transfer learning is important when developing machine learning solutions. Afterward, we provide a formal definition supported by practical applications in the *deep learning* field. Lastly, an overview of related research area is given.

2.1.1. The need for Transfer Learning

Machine learning and data mining methods are able to achieve good results in many supervised and unsupervised tasks. Thanks to the advent of deep learning technologies, we can also process effectively unstructured data (such as images, documents and sounds) at the cost of a long learning time and a large amount of data to extract relevant features from this kind of sources.

However, as discussed in [PY10], many of these methods rely on the assumption that training and test data are drawn from the same distribution. So, due to the statistical properties of the models under discussion, when there are perturbations or changes in the distribution of the test data, the performance significantly decreases, and a new model is built from scratch using fresh training data to be collected from the new distribution.

In this scenario, having a way to still exploit the knowledge acquired from a different distribution would be beneficial, because in real-world applications problems can arise from different perspectives.

- The collection of new data could be at a prohibitive cost or even impossible: the gathering of samples could require extensive manual-labeling efforts, or in case of data that become easily outdated, a sufficient amount of new training samples is collected over time and not immediately available, making the learning impossible.
- The computing resources could be limited, so the process of rebuilding the model from scratch may require an amount of training time that is not available, or it may even be impossible in case of memory or processing constraints.

Along the same lines, a way to transfer the learned representation of a model may come in handy when one wants to tackle a different task in the same domain of interest, saving computing time and resources. For example, two neural architectures that face different tasks in the same domain, such as regression and classification, can differ only at the last layer while reasonably assuming that all the remaining ones can have similar weights.

2.1.2. Definition

Here we define *transfer learning* as proposed in [PY10] and further analyzed in [Rud17]. First of all, the definition requires the concepts of “domain” and “task”, here explained.

A *domain* \mathcal{D} consists of a feature space \mathcal{X} and a marginal probability distribution $P(X)$ where $X = x_1, \dots, x_n, x_i \in \mathcal{X}$.

Given a domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a *task* \mathcal{T} is defined as the combination of a label space \mathcal{Y} and a predictive function which is the conditional probability distribution $P(Y|X)$, learned from the training data consisting of pairs $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

Now that the concepts of domain and task are provided, it is possible to formally define what is the goal of transfer learning as stated in [Rud17]:

Given a source domain \mathcal{D}_S , a corresponding source task \mathcal{T}_S , as well as a target domain \mathcal{D}_T and a target task \mathcal{T}_T , the objective of transfer learning is to enable

us to learn the target conditional probability distribution $P(\mathcal{Y}_T|\mathcal{X}_T)$ in \mathcal{D}_T with the information gained from \mathcal{D}_S and \mathcal{T}_S where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

Note that as both domain and task are tuples of two elements each, the definition allows four distinct transfer learning scenarios that are deeply discussed in [PY10] and [Rud17].

1. The feature spaces of the source and target domains are different: $\mathcal{X}_S \neq \mathcal{X}_T$.
2. The probability distributions over the feature space of the source and target domains are different: $P(X_S) \neq P(X_T)$.
3. Source and target tasks have different label spaces: $\mathcal{Y}_S \neq \mathcal{Y}_T$.
4. The conditional probability distributions over the label space of the source and target task are different: $P(Y_S|X_S) \neq P(Y_T|X_T)$.

Our work falls into the third scenario, generally known as *domain adaptation*: in our experiments, we consider the differences in the images and metadata in input to our models, that according to the classification above, translates into different marginal probability distributions over the feature space between the source and target domains.

2.1.3. Approaches in Deep Learning

As briefly mentioned in 2.1.1, deep learning makes it possible to face more challenging problems, but on the other hand, these techniques require longer training times and larger amounts of data with respect to classical data mining and machine learning techniques.

Deep learning models rely on *artificial neural networks* that are layered architectures whose goal is to learn a hierarchical representation of features. To take advantage of the representation learned by previous models, i.e. to *transfer* the *knowledge* of a pre-trained network about a domain or task, the idea is to avoid training a newly initialized deep network, and instead refine the pre-trained model to the new use case.

The way the pre-train model is used determines the type of transfer learning strategy. In [Sar18] those strategies are discussed, and we report them here:

- use of the pre-trained model as features extractor,
- fine-tuning of the pre-trained model.

The underlying idea of the former option is to exploit the pre-trained model’s weights of the first layers to extract features from the input data. Those layers are kept “frozen” while training either a set of newly initialized layers on top of the old ones or, more in general, a new shallow model that uses the extracted features as input. This option is

effective when little training data are provided and when the pre-trained model is expected to match the domain of the problem at hand.

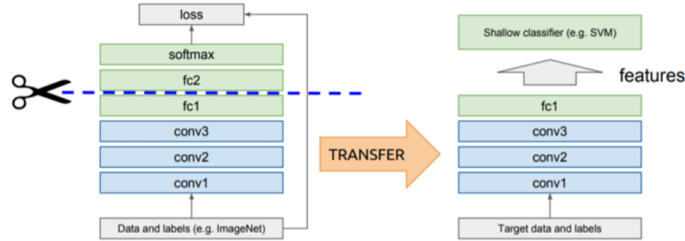


Figure 2.1: Transfer learning with a pre-trained model as features extractor [Sar18].

The latter strategy is more involved and does not just expect a substitution of the layer on top of the pre-trained network, but even the old layers are updated during the training phase. Since artificial neural networks are highly configurable through hyper-parameters, it is up to the designer to determine which subset of layers to keep frozen during training, if any.

Also known as *supervised-domain adaptation* through *fine-tuning*, this option is effective when enough training data are provided or when the pre-trained model is not expected to match the problem at hand, and an alignment of the source domain to the target one is desirable.

2.1.4. Applications in Computer Vision and NLP

In this section, we provide applications of transfer learning techniques in the fields of Computer Vision and Natural Language Processing.

Transfer Learning for Computer Vision

Deep learning is used extensively in computer vision tasks through several CNN architectures. When dealing with such tasks, a common procedure is to use the off-the-shelf features of a pre-trained CNN architecture that reaches state-of-the-art performance on a large, publicly available dataset such as *ImageNet*.

ImageNet[Den+09] is an image database containing data about a wide variety of objects. Researchers use this dataset to evaluate and compare their findings with related works of computer vision tasks such as object detection and image classification.

Architectures that are trained on this dataset capture, through their parameters, the useful, general knowledge related to the concepts depicted in the images that went through

the network during training. This very same knowledge can be exploited to excel in many other tasks.

So, transfer learning techniques allows reaching high-level performance in new tasks by either keeping the pre-trained parameters fixed or tuning them with a small learning rate to prevent the harmful behavior of knowledge forgetting. This last choice is affected by the amount of available data to face the new task. Popular architectures used as described include VGG[SZ15], Inception[Sze+14], ResNet[He+15].

Transfer Learning for NLP

Similar to Computer Vision, even Natural Language Processing models require very long training on huge amounts of data, thus transfer learning techniques became essential to build machine learning tools that handle words, sentences, or documents as input.

Historically the first applications of transfer learning applied to textual sources date back to word embeddings such as GloVe[PSM14] and Word2vec[Mik+13]. The embeddings were trained on multiple sources and used as a black box for tasks like word/sentence classification or information retrieval, sharing the knowledge acquired from the source domains to solve new tasks.

Later on, with the release of the Transformer architecture, new models have been developed and made available as already pre-trained with different objectives on massive datasets. Examples are BERT and GPT-2, which allow anyone to build machine learning models applied to textual data by fine-tuning the pre-trained model in a supervised manner, achieving high performance not only on word/sentence classification, but also on text generation tasks.

2.1.5. Related areas: Multi-task and Contrastive Learning

As discussed in the previous sections, transfer learning allows the implementation of high-performance models which generalize better when dealing with new domains, by leveraging a relatively small number of samples.

This is not the only research area that aims to pursue the goal above: several other directions related to transfer learning try to learn a domain-invariant representation of the input data, making related concepts that come from different domains more similar. Here we introduce two of them: *multi-task* and *contrastive learning*.

Multi-task Learning

While transfer learning focuses on a single target task or domain, *multi-task* learning takes advantage of the knowledge acquired by learning how to approach several different tasks simultaneously. This approach brings improvements in single tasks and reduces the need for labeled data [BS17].

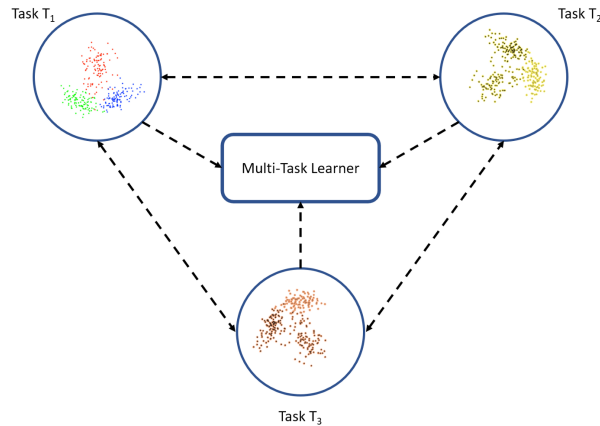


Figure 2.2: Multi-task approach: the learner receives information from all the tasks simultaneously [Sar18].

This approach is mainly studied when applied to NLP, where models process input texts that follows a structure designed to fulfill a particular task. In [Raf+19], they define a common format across tasks, making it possible to consider a unified challenge where a model receives text as input and produces new text as output. This approach allows machine learning developers to adopt the same architecture and the same training options, like loss function, hyperparameters, and others, to tackle a set of multiple tasks.

Contrastive Learning

This learning technique is centered on how to learn without labels using a *self-supervised* approach. We present its main aspects as described in [Tiu21].

The idea of *contrastive learning* is to teach the model which data points are similar, without knowing the target labels. This paradigm is *task-independent* and allows the model to learn general features by considering similarities and differences in the input samples.

In real-world scenarios collecting and preparing samples is an expensive task, and contrastive learning offers a viable option to have competitive models when only a fraction of the dataset is labeled.

A common approach is to use this technique to pre-train a model, which is then fine-tuned for a specific task with the labeled samples that are available. Having the initial self-supervised step can improve the performance of the learner, and potentially outdo even “full” supervised methods.

Contrastive learning most commonly deals with images through data augmentation, but recent works proposed this approach with textual or multi-modal sources.

2.2. Image Captioning

In this thesis work, we propose an analysis of captioning systems for fashion products from the perspective of their generative performance in different transfer learning scenarios.

The research community defines *image captioning* as the process of generating textual descriptions from images. Systems meant to tackle this assignment expect to handle visual and textual data, exploiting both Computer Vision and Natural Language Processing techniques. In this section, we review the technologies adopted in captioning systems, starting with how images are processed, and then moving towards the modern techniques to deal with textual data.

2.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are neural networks specifically designed to process data in a “grid-like” shape, such as images. The CNN typical structure is a hierarchy of blocks that transform the input in a sequence of *volumes*. Each block receives as input and returns a volume; as the depth increases, the spatial extent of the volume decreases. The main blocks of this class of networks include *convolutional* layers, *pooling* layers, and *fully-connected* layers with activation functions.

A convolutional layer performs a linear combination of all the values in a region of the input, whose parameters compose a so-called *filter* (or *kernel*). A filter moves through the whole spatial extent of the input volume producing a two-dimensional representation that is known as *activation map*. The size of the filter determines the *receptive field*, which is the region of the input involved in the convolution. In a convolutional layer usually, there are multiple filters, each of which contributes to a single slice of the output volume.

Pooling layers reduce the spatial extent of the volume. They operate on every slice of the volume by resizing its height and width. In this way, a smaller number of weights is required, which reflects in faster computations and smaller memory requirements.

Activation layers introduce non-linearities in the network, making the CNN able to learn non-linear separation boundaries in the feature space. As for multi-layer perceptron architectures (MLP), the typical activation function is the ReLU, which applies a threshold on the activation map through a *max* operator. ReLU is effective, but it suffers from the dying neuron problem: a behavior that makes neurons of the network insensitive to the input. To overcome this limitation, it was introduced the Leaky ReLU activation, which includes a slope for negative values.

CNNs achieve high performance when dealing with images because of the way they operate: while MLP networks use dense, fully connected layers that allow every output unit to be influenced by each input one, CNNs use filters whose dimension is smaller than the input, that slide over each region of the image, allowing *parameters sharing*. In this way, the hidden representations learned by the network are invariant of spatial translations.

ResNet

Residual Network [He+15] was one of the most relevant works in Computer Vision of the last decade. The ResNet architecture was presented in 2015, and from there on, researchers have deeply analyzed the network to find the reasons for its success, proposing refinements and even applications for other deep learning fields.

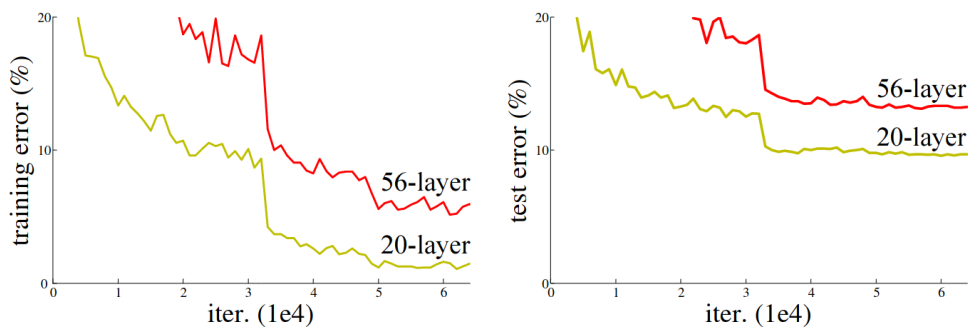


Figure 2.3: Training error (left) and test error (right) with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error [He+15].

In the deep learning research community, there is a common trend for which the network architecture needs to go deeper to avoid *overfitting* and achieve better results than shallower networks. This is not always the case as demonstrated in [He+15], where they analyze if it is possible to continuously improve the performance of a CNN by stacking more and more layers. As shown in figure 2.3, increasing the depth of a simple CNN is not always effective: the training and test errors follow the same behavior, highlighting that the decline of the performance of the deeper network is not due to overfitting. In that

case, the training error would have been low, while the test error would have diverged.

The reason why the deeper network fails in reaching the performance of the shallower one is the *vanishing gradient problem*: deeper networks are hard to train because the update of early weights requires repeated multiplications that can make the gradient very small, preventing the learning. To overcome this limitation, making possible the training of deeper networks, the ResNet architecture introduces an “identity shortcut connection” (figure 2.4).

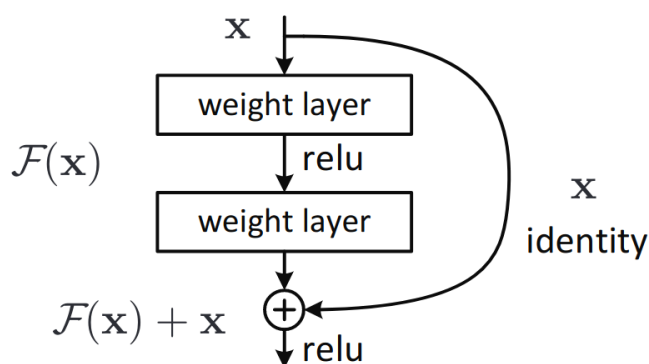


Figure 2.4: The ResNet block with the *identity shortcut connection* [He+15].

Skipping one or more layers through the new connection forces the network to learn a “residual” that is added on top of the identity, to improve the solution of the shallow network. In this way, it is ensured that higher layers will reach performance at least as high as the lower ones. Notice that the skip connection does not introduce any additional weight, and allows stacking more and more blocks in the overall architecture that can be trained through back-propagation also over path introduced by the skip connection itself, avoiding the vanishing gradient problem.

2.2.2. Natural Language Processing

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language¹. The goal of NLP techniques is to provide a machine able to understand natural language text and react accordingly. Computer systems that want to pursue this goal require models to face linguistics tasks such as word-level processing, syntactic processing, lexical and compositional semantics, dialog structure, and others. Historically these challenges have been faced through either finite-state and context-free methods, grammars, or

¹https://en.wikipedia.org/wiki/Natural_language_processing

first-order logic until the take-up of modern machine learning and deep learning methods.

This field of AI is in constant growth, driven by the enormous amount of knowledge available in machine-readable form and the fact that human-to-human communication is nowadays increasingly mediated by computers. Examples of applications of NLP are *question answering*, *conversational agents*, *summarization*, and *machine translation*; all of them require a representation of the distribution of sequences of words, achievable through a *language model*.

Language Modeling

A *language model* is a probabilistic model meant to predict the next word in a sequence, given the words that precede it. Also known as statistical language model, it learns the probability of words to occur based on text samples. These probabilistic models are fundamental for complex tasks that require language understanding.

A simple language model can be an n -gram model: the probability of observing the i^{th} word w_i given all the preceding ones is approximated by the probability of observing w_i in the context of the $n-1$ preceding words. The conditional probabilities are computed using the frequencies of the n -grams in the training corpus and a *smoothing* term. The latter is necessary to avoid inference problems when dealing with n -grams not seen before.

The main limitation of this method is that only the $n-1$ previous words influence the probability distribution of the next word. It is not easy to determine a "good" amount of preceding words to use as context, besides the higher memory and data requirements necessary to store and compute reliable n -gram probabilities, as the number of permutations of n words increases. Deep learning techniques based on *LSTMs* or the *Transformer* address this limitation.

2.2.3. LSTM

LSTM, short for Long-Short Term Memory, is a type of Recurrent Neural Network (RNN) proposed by Hochreiter and Schmidhuber in 1997. RNNs are a way to extend plain feed-forward neural networks via *recurrent connections*, making the network capable of memorizing a representation of past inputs. They are trained with *backpropagation through time* but they suffer of the *vanishing gradient* problem: several backpropagation using sigmoids and hyperbolic tangents make the gradients close to zero. LSTMs solve this problem using a memory cell composed of logistic and linear units: the information loops over the *cell state* having a fixed weight, so backpropagation is possible independently by

the length of the sequence. These architectures are designed to deal with sequences; thus, language modeling is a proper use of LSTMs since sentences are just sequences of words.

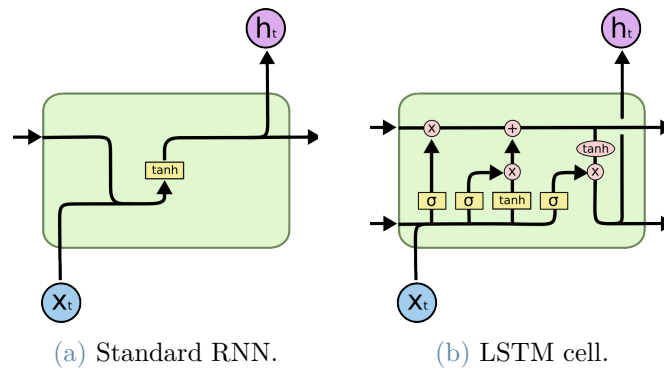
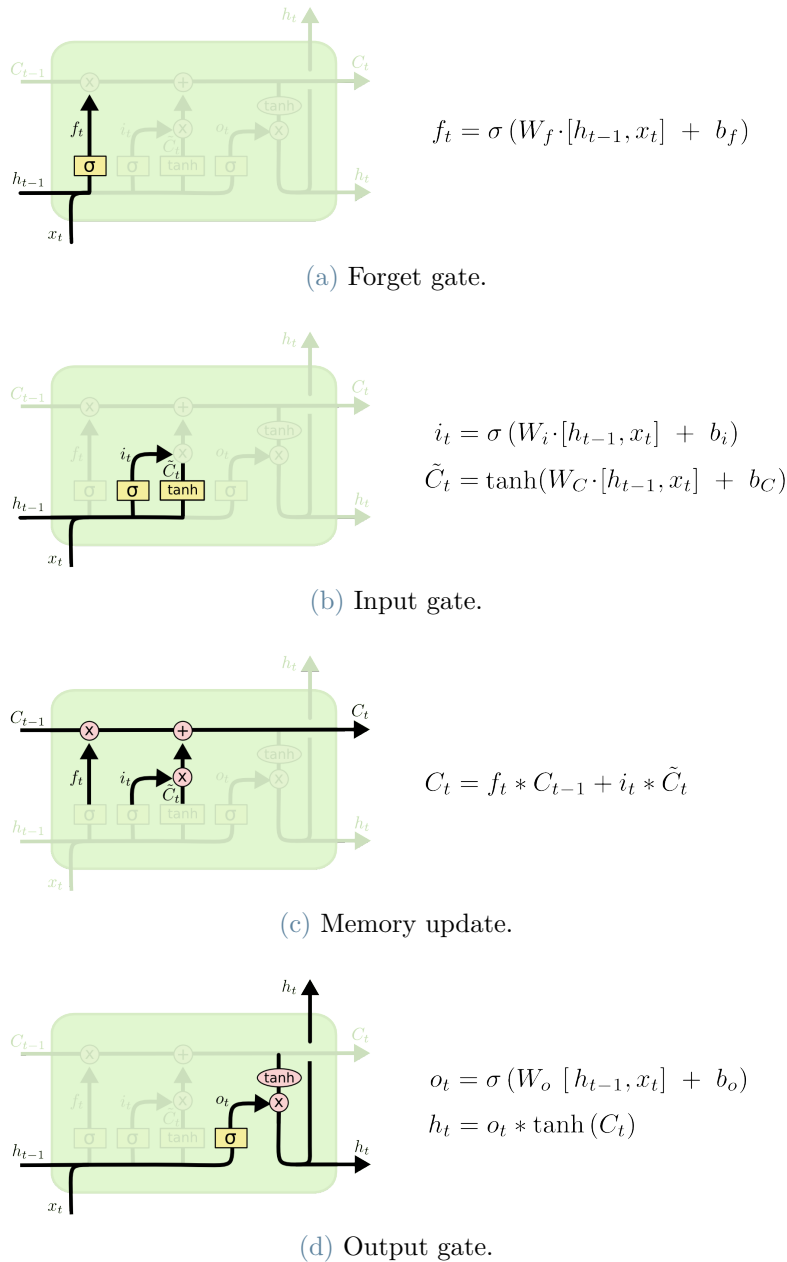


Figure 2.5: Comparison between the repeating module of RNNs and LSTMs.¹

Three distinct gates allow the update of the cell state by optionally letting information through. The “*forget*” gate determines the values of the cell state to keep. The information to store in the cell is computed by the “*input*” gate: first, it decides the values of the state to update, then it computes the new candidate values that could be added to the state. Now that both the values to forget and the new candidates are available, the new cell state is computed. Finally, the “*output*” gate filters the cell state to build the new hidden state. The new cell and hidden states are the input in the next timestep.

Figure 2.6: Gates of a LSTM cell.¹

2.2.4. The Transformer

RNN based architectures, like LSTMs, do not allow parallelization within training examples due to their intrinsic sequential nature, making the computation expansive when processing long sequences. The *Transformer* [Vas+17] model overcomes this limitation, besides outperforming LSTMs, and generally RNNs, in NLP tasks: without recurrence,

¹RNN and LSTM images from <https://colah.github.io/posts/2015-08-Understanding-LSTMs>

nor convolution, it deals with sequences through an attention mechanism and information about the position of tokens to preserve the concept of order in the utterance. The attention mechanism allows the model to learn to focus on the most important pieces of the sequence at each step.

The Transformer is an encoder-decoder model, where both the encoding and decoding components are a stack of repeating modules, respectively encoders and decoders blocks. Figure 2.7 shows the overall architecture of the model. All the details of the architecture are deeply analyzed in [Ala18].

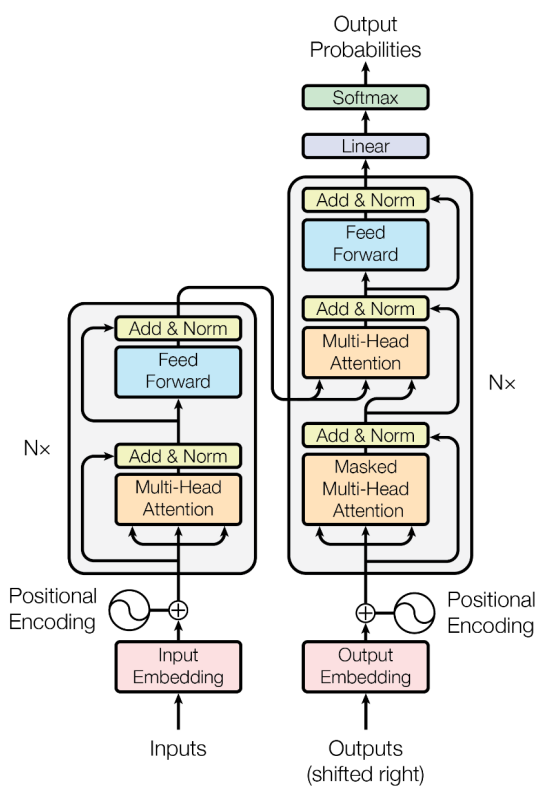


Figure 2.7: Architecture of the Transformer model [Vas+17].

The encoder block processes its input through an attention layer, then a feed-forward neural network. The decoder block interleaves those layers with an “*encoder-decoder attention*” which allows focusing on pieces of the input sequence.

At first, each input token is converted into the related embedding, making the input of the stack of encoders a list of vectors of the same size. As this model avoids recurrences, a *positional encoding* is added to each input embedding to account for the specific order of words in the sequence. The input of an encoder block can be either the initial tokens embeddings, with the addition of the positional encoding, or the output of the previous

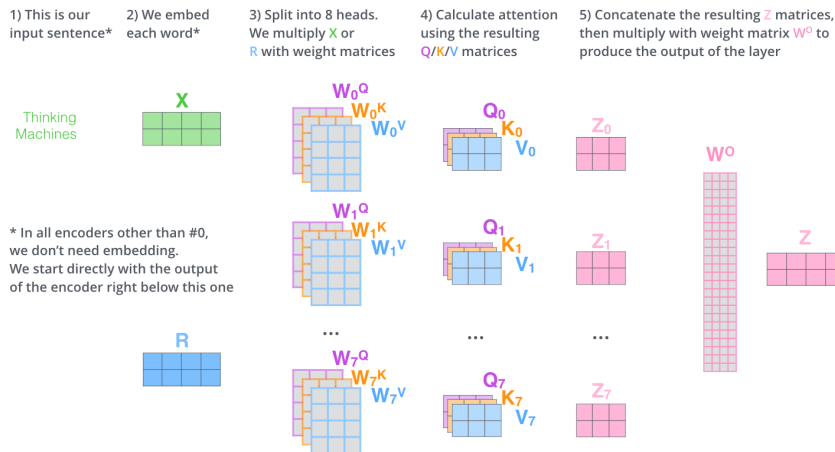


Figure 2.8: Summary of the steps of the *multi-head self-attention* layer [Ala18].

block of the stack.

The goal of the attention layer is to provide hints coming from other positions of the sequence, that help in encoding the current token. To do so, three vectors for each input embedding, namely *query*, *key*, and *value* vectors, are used to provide a score representing how much each other token is relevant for the current one. The three vectors are the result of the multiplication of the input with supplementary matrices learned during the training. The computation of the scores happens by taking the *dot* product of the query vector of the *current token* and the key vector of the *others*. The scores are rescaled and then normalized through a *softmax*, making them sum up to 1. The final softmax score determines how much each word is expressed at a position of a sequence; by multiplying these scores with the respective value vectors, the values of all the tokens are rescaled according to how much they are relevant to the current one. The final output is the sum of the rescaled value vectors.

Here we considered the computation of the attention for a single token; however, to allow faster processing, the Transformer process the entire sequence at a time in matrix form.

In each block of the stack, the self-attention described above is replicated, making a more complex mechanism called “*multi-head*” attention. The softmax scores of the attention heads are combined by multiplying the concatenation of the scores with a further weight matrix learned during training. In this way, the input of the feed-forward neural network is independent of the number of heads that are used. Figure 2.8 summarizes all the steps performed by the multi-head self-attention mechanism.

This mechanism improves the performance of the model mainly because it introduces distinct sets of *query/key/value* matrices for each head, that allow the projection of the

input embeddings into multiple representation subspaces.

The decoder block works similarly to the encoder one. It adds an “*encoder-decoder attention*” layer that operates as the base self-attention block, except it builds the query vectors from the output of the previous layer of the decoder stack, while the key and value vectors come from the encoder side output. Additionally, the self-attention layer can attend only to earlier tokens in the sequence. This is achieved by masking future positions before computing the softmax score.

The output of the decoder stack flows through a final linear layer and later through a softmax layer. The linear layer is a feed-forward neural network that projects the vector produced by the decoder stack into a higher-dimensions representation called the “*logits*” vector, whose size is equal to the vocabulary length, containing a score for each word. The softmax layer processes the scores by turning them into probabilities of words to occur. The most probable word is the final output of the current step.

BERT

With the release of the Transformer architecture, the research community started analyzing variations of the original model, and the understanding of how to represent sentences and words rapidly evolved.

BERT [Dev+19], or Bidirectional Encoder Representations from Transformers, is a method of pre-training language representations by learning a general-purpose Transformer-based model trained on a large corpus. The BERT architecture consists only of the encoding stack of the Transformer, using a higher number of attention heads and encoder blocks than the original implementation of the Transformer proposed in [Vas+17].

The pre-train phase is the first stage of the learning process of the model. In this phase, the model learns a language representation in which words are conditioned on both left and right contexts. BERT achieves this by training its stack of encoder blocks on the task of “masked language modeling”: a fraction of the input tokens of each sentence (15%) is replaced with the [MASK] token; moreover, the model tries to predict the original values given the context provided by all the non-masked tokens. Since the loss function is computed taking into account only the masked positions, the model converges slower than other models.

Additionally, in the BERT pre-training process, the model receives pairs of sentences and learns to predict whether the second one is the subsequent sentence in the original document. The input is formatted so that a [SEP] token is inserted at the end of each

sentence; also, the model uses an additional vector indicating which is the source sentence of the token in the corresponding position.

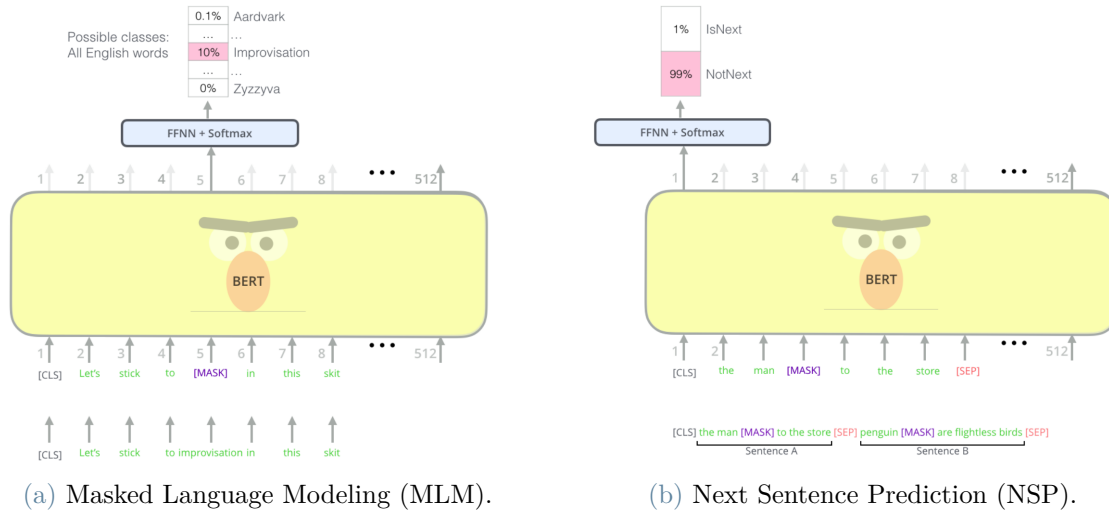


Figure 2.9: BERT architecture according to the pre-training task.²

By exploiting the pre-training procedure described above, one can download the pre-computed parameters; finetune the model; save time, knowledge, and resources; and achieve higher performance than a language-processing model built from scratch. BERT achieves state-of-the-art results on a wide range of NLP tasks.

GPT-2

GPT-2 [Rad+19] is a Transformer-based language model that outperforms previously developed architectures in text-generation capabilities. This model is trained on a very large corpus of ~ 40 GB of text data with the simple objective of predicting the next word, given all the previous words in a text.

Unlike BERT, the architecture of GPT-2 is made of Transformer decoder blocks only: it outputs one token at a time, adding the last predicted one to the sequence of inputs, making the model “auto-regressive” by nature.

The blocks composing the stack of the GPT-2 architecture include a masked self-attention layer and a feed-forward neural network. The self-attention layer achieves the “auto-regression” property by masking the tokens related to future words, blocking information from tokens right of the position being calculated.

²Images from <https://jalammr.github.io/illustrated-bert/>

During training, the input sentence comes entirely from the batch, whereas at inference time the last predicted token is appended to the current input to predict the next one; the inference process goes on until the prediction of the end-of-sequence token, or the maximum generation length is reached.

Decoder-only Transformer-based architectures are not used only for language modeling, but also in different NLP tasks: thanks to the simple but effective way it is pre-trained and the huge amount of training samples, GPT-2 allows reaching high performance through fine-tuning on text summarization, question answering, and others.

GPT-2 comes with variants according to the size of the architecture. Recently the same research team has developed GPT-3: a language model which is larger, more versatile, and qualitatively stronger than GPT-2. GPT-3 during the training procedure is exposed to 300 billion tokens of text, and it is estimated to cost 355 GPU years and cost \$4.6m. At the time of writing, it is accessible only through a cloud-based application programming interface with usage-based pricing.

Vision Transformer

CNNs have been used as standard technology in computer vision tasks since AlexNet (2012). When building a machine learning model that processes visual input, CNNs avoid using hand-designed features, while instead learning visual features directly from data. Architectures built upon them are effective for image processing, but are specifically designed for this type of input and can be computationally demanding when the goal is to build scalable vision models.

When dealing with textual sources, the Transformer model proposed in [Vas+17] (2017) is computationally efficient and scalable, making it possible to train large models on huge corpora. From 2020 researchers started wondering if it were possible to exploit a Transformer-like architecture to process visual inputs.

The Vision Transformer (ViT) [Wu+20] is a vision model based on the Transformer architecture. The goal of the model is to process an input image as if it were a textual input, making the fewest possible modification to the classical Transformer model. [Wu+20] demonstrates that ViT reaches excellent performance when trained on sufficient data, outperforming a comparable state-of-the-art CNN being at the same time more efficient (it requires four times fewer computational resources).

ViT exploits a stack of encoder blocks, each of which has the structure presented in the original model. The input image is split into a square grid of patches; each patch is

flattened into a single vector by concatenating the color channels and then projecting the vector to the desired dimension through a linear layer. A learnable positional embedding is added to each patch, making the model able to learn the structure of the images through information about the dispositions of the patches. The final sequence of vectors flows through the standard Transformer encoder. To perform the classification of the input images, a [class] learnable token is prepended to the sequence of embedded patches; the final class is the output of a prediction head on top of the last encoder block. Figure 2.10 summarizes all the steps done by the model.

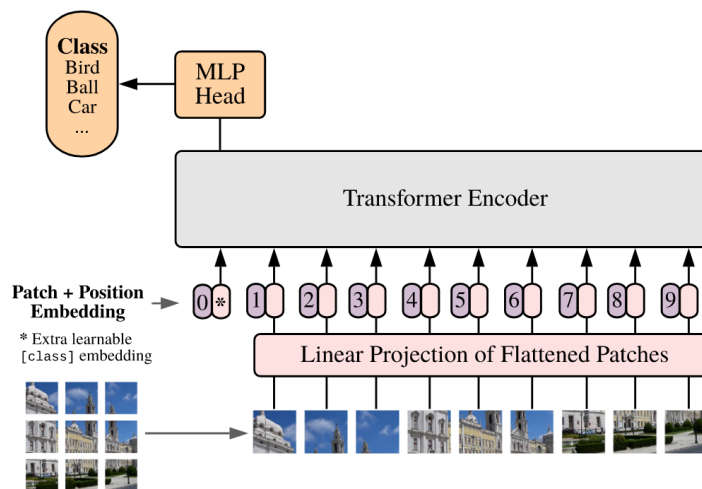


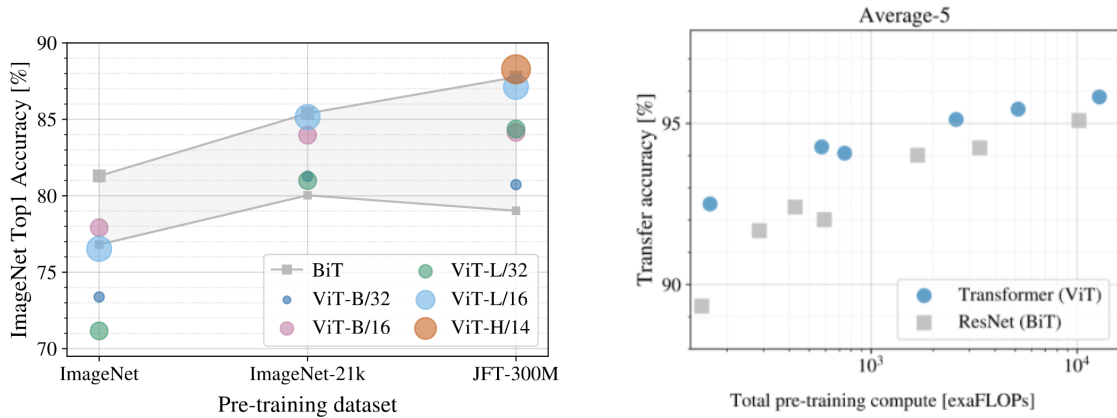
Figure 2.10: Model architecture and processing steps of ViT [Wu+20].

ViT is pre-trained on large datasets and finetuned to smaller downstream tasks; the finetuning step requires the removal of the pre-trained prediction head with a newly initialized feedforward layer.

The model comes with variants according to the number of parameters; the configurations of the variants are based on those used for BERT: ViT-Base (86M), ViT-Large (307M), and ViT-Huge (632M). The performance of the model depends on the size of the dataset used for pre-training: Figure 2.11a shows the results after finetuning to ImageNet when the model is pre-training on datasets of increasing size as ImageNet (1M images), ImageNet-21k (14M images), and JFT (300M images). Notice that the ImageNet pre-trained models are also finetuned on the same dataset because the resolution increases during finetuning, improving the final performance.

To compare the amount of computation performed when training the models, [Wu+20] provides an analysis of several different ViT models and CNNs trained on the same dataset (Figure 2.11b). For a given amount of compute, ViT yields better performance than the

equivalent CNNs.



(a) ViT models perform worse than BiT³ResNets (shaded area) when pre-trained on small datasets, while they excel when pre-trained on larger datasets.

(b) Performance versus pre-training compute for different architectures: Vision Transformers and ResNets. ViT generally outperform ResNets with the same computational budget.

Figure 2.11: ViT performance and training requirements analysis.

Recently a pre-training method inspired by BERT and its *masked language modeling* objective has been applied to ViT. BEiT [BDW21], which stands for Bidirectional Encoder representation from Image Transformers, proposes a *masked image modeling* task that uses two representations of the images through patches of pixels and visual tokens. The goal is to recover the original visual tokens related to patches of the input image that were randomly masked.

2.3. Summary

In this chapter, we present an overview of *transfer learning*: we define its goal and the practical needs that make it essential for the development of a competitive deep learning model. Among the research areas that aim to generalize better when dealing with new domains there is *contrastive learning*, which we adopt in the training procedure of our proposed fashion image captioner.

Regardless of the business application, an image captioning system has a standard structure that involves two fundamental modules: an image feature extractor and a caption generator. We present the CNN architecture and the modern Vision Transformer that we use as feature extractors; the caption generator module of the systems we analyzed

³<https://ai.googleblog.com/2020/05/open-sourcing-bit-exploring-large-scale.html>

is an LSTM architecture or a Transformer language models, implemented through either BERT or GPT-2.

3 | Related work

In this chapter we present the research works related to this thesis. The first section dives into works about transfer learning, initially by presenting different approaches and application scenarios, and further by providing an overview of recent achievements in applying a contrastive learning objective for the training of multi-modal (joint image and text) deep learning models. In the last section, we provide both common and modern choices of deep learning architectures meant to tackle the task of image captioning.

3.1. Transfer Learning

The transfer learning capabilities of deep learning models are widely studied with several distinct approaches. We focus on the techniques that allow the adaptation to a target task, providing a use-case of cross-domain adaptation. In the end, we discuss the role and the benefits of a contrastive pre-training objective and the model that exploit it.

3.1.1. Target task adaptation

The usual transfer learning procedure from a pre-trained deep learning model to a target task consists of two phases: the *pretraining*, where the model learns a representation of the input samples related to a source task, and the *adaptation*, where the representation learned by the model is transferred and applied to a new task. While the former stage involves the selection of a proper training objective and model architecture, the latter consists in the exploitation of practical methods to adapt the knowledge of the model, which is held in the weights of the model, to the final task.

As discussed in 2.1.3, the main paradigms of adaptation are *feature extraction* and *fine-tuning*. In [PRS19] the authors provide a set of adaptation guidelines that specify when the paradigms above are effective, therefore preferred, when applied in the NLP domain.

The feature extraction adaptation form provides for the use of the pre-trained representation as they are, keeping the weights of the model *frozen*; i.e., fixed during the adaptation process. This technique is suitable for “recycling” precomputed features on similar tasks,

so being cheaper as the features are computed once. Differently, fine-tuning adapts the pre-trained representation by allowing the eventual changing of the model weights that can be “unfrozen” while fine-tuning the model on the target task. This second approach is convenient when a general-purpose representation is used as starting point to face new tasks. Fine-tuning implies the selection of the parameter groups of the pre-trained model that can be trained during the adaptation process: this selection is a design choice that affects the final performance of the model on the target task. A possible approach is to gradually make the parameters learnable during fine-tuning (a.k.a. *gradual unfreezing*).

[PRS19] computes the relative performance of two popular pre-trained architectures (ELMo [Pet+18] and BERT [Dev+19]), which at the time of writing reach state-of-the-art performance across several NLP tasks, when employed with the two forms of adaptation described above. In our work, we use a similar approach to evaluate the effectiveness of both the pre-training and adaptation choices proposed in our transfer learning analysis, applied to different architectures for fashion image captioning.

Feature extraction and fine-tuning are two basic adaptation paradigms: it is possible to design adaptation processes that are more complex but more powerful, that are strictly related to the application domain and the pre-trained model architecture. [VN21] provides an adaptation procedure that aims to exploit the success of the GPT-2 English generative language model to other languages (Italian and Dutch). Their work is prompted by the fact that non-English language models are less powerful due to data limitations, so they describe a method to adapt existing pre-trained models to new languages. The adaptation of the language model requires four key steps. At first, the small version of the pre-trained English GPT-2 is trained by keeping fixed the transformer layers and re-initializing the lexical embeddings. In this way, the knowledge acquired in the transformer layers is preserved and the model is already capable to generate realistic text in the target language. The next step is adopted if one wants to increase the model size: a least-squares regression allows the project of the embeddings of the smaller model to a larger size; the newly projected embeddings can be optimized by additional training, always keeping the transformer layer weights fixed. Finally, the entire model is “unfrozen” so that all the weights are trainable and fine-tuned to the target language. Even though the retraining of the lexical embeddings could be an expensive procedure, the projection to a larger representation space keeps the overall cost limited.

3.1.2. Contrastive Learning

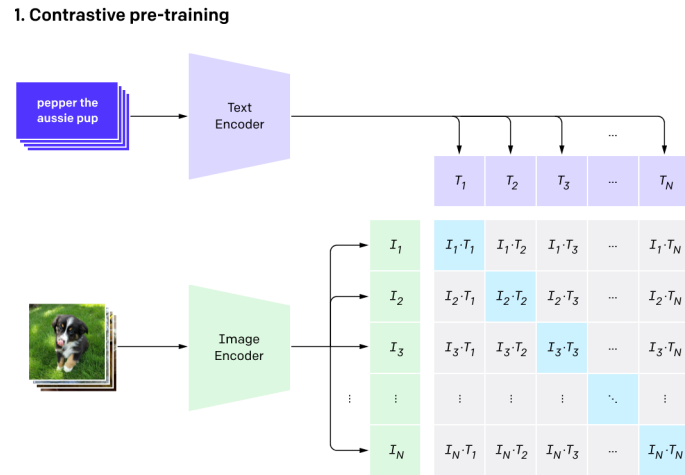
While the previous section focused on the adaptation of a pre-trained model, here we discuss the pre-training objectives that allow the learning of a general-purpose representation of the input data, more precisely to contrastive learning, whose key aspects are presented in 2.1.5.

Contrastive learning is extensively used in CLIP [Rad+21], short for *Contrastive Language-Image Pre-training*, which is a neural network model that uses natural language supervision to allow generalization and transfer of knowledge to correctly understand vision features of objects that were not seen during training. The main motivation that led to CLIP comes from the poor performance of current vision models when adopted with input data coming from different distributions from the one used for training, reflecting on a significant effort to adapt to new scenarios.

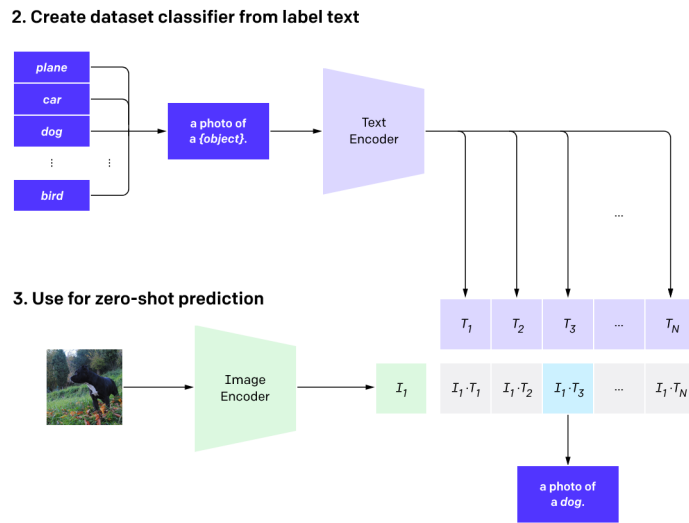
CLIP is an image classification model that exploits the meaning of class labels associated with images to make a powerful “*zero-shot*” learning model through a contrastive pre-training; differently, standard classification models discard the meanings related to class labels, by replacing the class names with numerical identifiers. To do so, CLIP uses an image encoder (ResNet/ViT-based) and a text encoder (Transformer-based) to predict the correct pairings of a batch of image and text pairs. The contrastive learning objective exploited by CLIP is the minimization of the cross-entropy loss over the cosine similarities scores of image and text embeddings in the batch: in this way, it is maximized the cosine similarity of the N correct pairs of image-text samples in the batch, while minimizing the cosine similarities of the $N^2 - N$ incorrect pairs. Figure 3.1a shows the contrastive pre-training steps. At test time, the dataset classes are converted into captions then CLIP predicts the class related to the caption with the highest similarity with a given image. Figure 3.1b shows the model behavior during inference.

We propose a deep learning model that exploits the innovative pre-training approach of CLIP while applied to the generative task of producing captions that describes input images.

The contrastive pre-training carried out by CLIP can be seen as *multi-objective* pre-training, as the model tries to learn two tasks at the same time: first, the learning of an image embedding space; second, the learning of a text embedding space used to perform the alignment with the image embeddings. As experimented in [Zha+21], contrastive pre-training on image-text solves both of these tasks simultaneously but it may be a sub-optimal approach. [Zha+21] proposes “*contrastive-tuning*”: a method that exploits contrastive learning to align image and text models while still taking advantage of their



(a) CLIP training steps.



(b) CLIP inference steps.

Figure 3.1: CLIP contrastive approach during training, and inference behavior.

pre-training and that allows you to select which modality to keep locked during the alignment. The use or not of a pre-trained model, together with the choice of keeping a modality locked, introduces several tuning options, which are explored in [Zha+21] and depicted in figure 3.2.

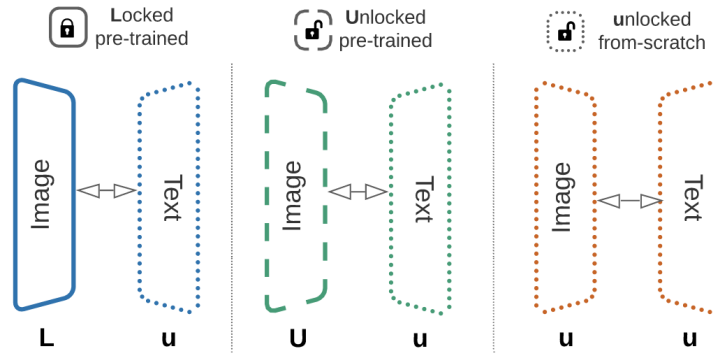


Figure 3.2: Contrastive-tuning: the design choices for the image-text pre-train. L stands for locked and initialized from a pre-trained model, U stands for unlocked and initialized from a pre-trained model, u stands for unlocked and initialized from scratch (random). Image from [Zha+21].

They empirically found out that keeping locked the pre-trained image model is the option that works best; anyhow, several options are available, and exploring them could improve the overall pre-training performance.

3.2. Image Captioning

The research community has developed several approaches to tackle the Image Captioning problem. While in section 1.5 we provide a description of the challenge and an overview of Computer Vision and NLP technologies, here we discuss classical and modern models that exploit those technologies to build a captioning system.

3.2.1. Show, Attend, and Tell

Show, Attend, and Tell [Xu+15] was the most common choice to build captioning systems until the advent of the Transformer architecture. The neural network model is based on an encoder-decoder architecture that uses a CNN encoder and an LSTM decoder. The two components are interconnected with an additional *attention mechanism* that makes the model focus on salient regions of the input image.

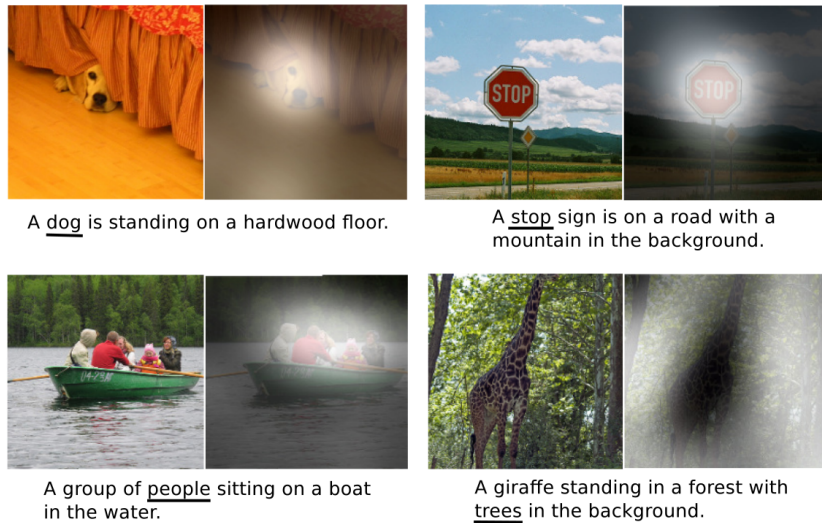


Figure 3.3: Examples of attended regions (in *white*) with the corresponding *word* generated as output [Xu+15].

The CNN extracts a set of feature vectors that are the output of the last convolutional block, without the use of fully connected layers on top: this allows to maintain a correspondence between the feature vectors and the 2D regions of the input images. The LSTM decoder composes the final caption by generating one word at a time. Each decoding step determines the word to output, conditioned on a context vector, the previous hidden state, and the last generated words. The context vector keeps the relevant features of the input image: each feature vector \mathbf{a}_i is rescaled according to a weight α_i , which represents the probability that \mathbf{a}_i contains the visual features of the region of the input image that is the most relevant to produce the next word. The weights are computed by the attention model as the output of a fully-connected layer, having as input the feature vectors and the previous hidden state of the LSTM. Figure 3.3 shows some examples of attended regions of the input image with the corresponding output word. The attention mechanism allows to improve the generative performance of the model and gives more interpretability into the caption generation process.

3.2.2. Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks

With the advent of the Transformer architecture, cross-modal pre-training methods applied to Transformer-based models have become the usual solution to approach vision-language tasks like Image Captioning. Standard pre-training approaches exploit the self-attention layer of the Transformer to align vision and text embeddings; differently, Oscar

(Object-Semantics Aligned Pre-training) [Li+20] uses object metadata detected in images as anchor points during the pre-training. This is achieved by extending the training samples to *triples*, each composed of a set of regions of the input image, the object tags, and a sequence of word embeddings (the textual caption in our scenario).

The object tags, along with the regions of the input images, can be computed by modern object detectors: [Li+20] uses a Faster R-CNN to retrieve that information. Figure 3.4 illustrates the Oscar pre-training procedure and the model architecture.

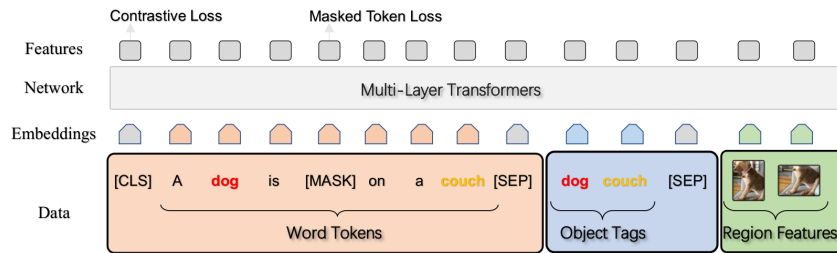


Figure 3.4: Oscar pre-training [Li+20].

The pre-training objective combines a *masked token loss* and a *contrastive loss*. The masked token loss, similarly to the masked language model adopted by BERT, randomly replaces 15% of the token with [MASK]. The goal is to predict the values of the masked tokens by minimizing the negative likelihood given the surrounding, which consists of the other unmasked tokens (caption and object tags) and the image features.

The contrastive pre-training objective randomly replaces with 50% probability the sequence of object tags, then performs a binary classification to predict whether the tags sequence is the original one or not.

Transformer-based models with Oscar pre-training define the state-of-the-art performance in several vision-language challenges, such as Image Captioning.

3.2.3. Image tagging and captioning for fashion catalogues enrichment

As opposed to the previous approaches discussed in this section, now we present a model that focuses on fashion captioning: the specific application of Image Captioning systems that we analyze in this thesis work.

[PSC20] tackles the broad challenge of fashion catalogs enrichment that aims to automatically generate metadata and captions of fashion image products to enrich the catalog

of e-commerce websites of fashion companies. They propose two distinct deep learning models used for the generation of clothing tags and garment descriptions, respectively; moreover, they combine the two in a unique model that performs both tasks simultaneously.

As our analysis investigates fashion captioning systems, we focus only on the captioner architecture they named *Multimodal GPT-2*, shown in figure 3.5.

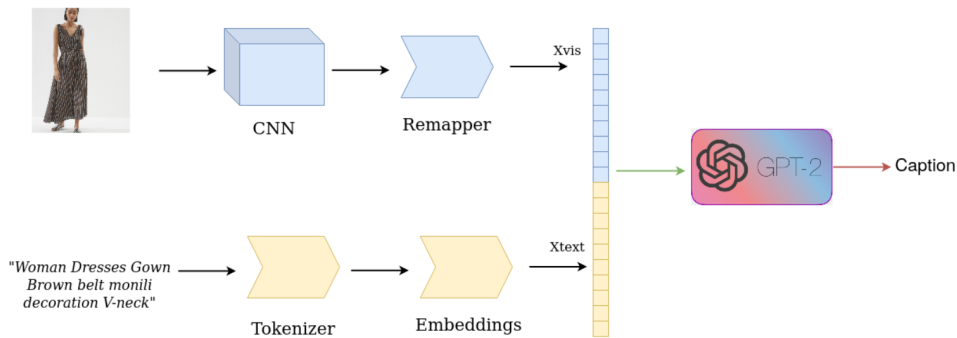


Figure 3.5: Multimodal GPT-2 architecture [PSC20].

The model exploits the successful generative capabilities of the GPT-2 architecture to produce captions of clothing images, taking advantage of the visual features extracted through a CNN encoder and the textual embeddings of the metadata related to the input product. In this way, the model can leverage additional textual information to ease the generation of the final caption and improve the quality.

The visual features extracted by the CNN encoder flow through a *remapper* block, which is a set of fully-connected layers each of which produces a visual token by projecting the input into the textual token embeddings size of GPT-2. The textual token embeddings associated with the product metadata are retrieved from the pre-trained GPT-2 tokenizer. Through the remapper block, the visual tokens share the same embedding space of the textual tokens, making GPT-2 able to process the multi-modal input composed of text and visual information.

4 | Datasets

In this chapter, we present the data sources used to train and evaluate the model architectures we investigate applied to the task of Fashion Image Captioning. The datasets considered in this thesis work are either publicly available or property of our industrial partners: we start presenting the public one, then the focus moves to three private datasets; finally, we compare the main properties of all the four datasets used in our research work.

4.1. Fashiongen

Fashiongen [Ros+18] is a large-scale dataset used in the Generative Fashion Challenge¹. It consists of fashion images (1360×1360 pixels²) annotated with descriptions and categories provided by professional stylists. The competition related to this data source is on the task of text to image generation, but, as for MS COCO [Lin+14], the research community exploits this dataset for other benchmarks that require image and text pairs, like image captioning. Each clothing product belongs to a main category and a more detailed subcategory. The dataset contains 48 main categories and 122 fine-grained ones. The plot in figure 4.1 shows the name and the distribution of the main categories of the dataset.

¹<https://fashion-gen.com/>

²Even though the authors present a dataset composed of high-resolution images of size 1360×1360 , the dataset available online consists of 256×256 fashion images.

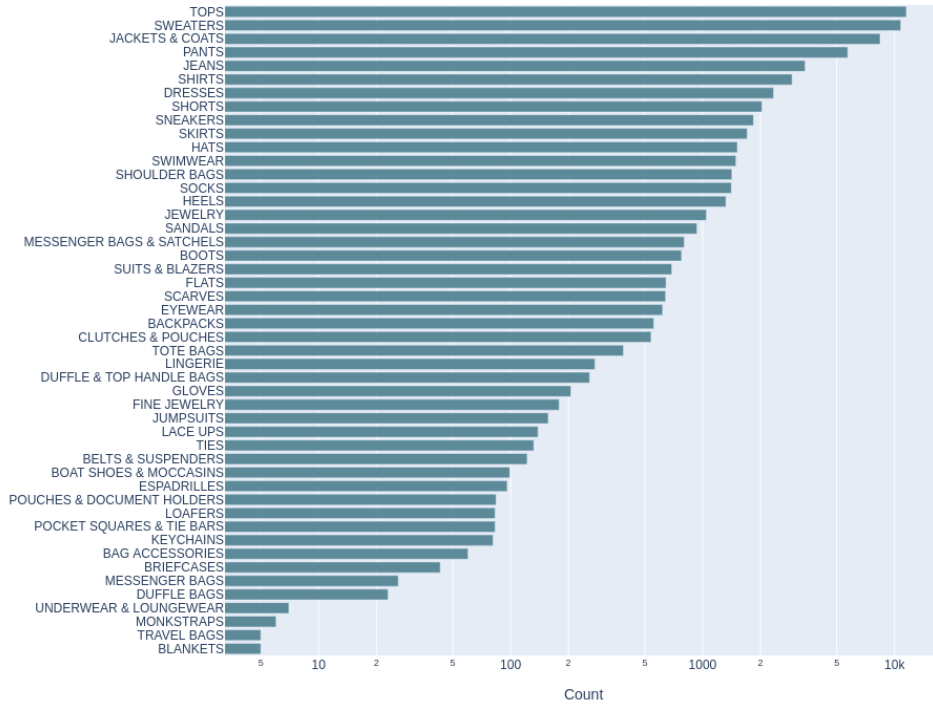


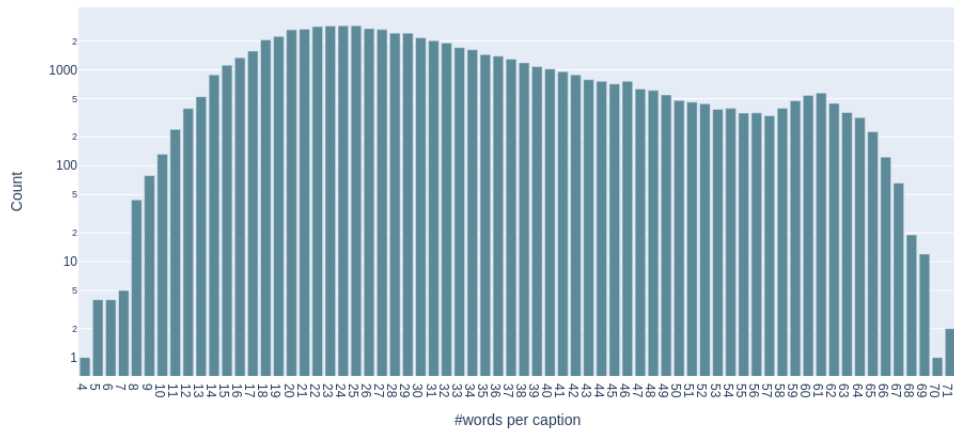
Figure 4.1: Bar plot representing the distributions of the categories of the fashion items belonging to Fashiongen.

The fashion items are paired with descriptive captions provided by professional stylists. Table 4.1 shows examples of descriptions with the main category of the related products.

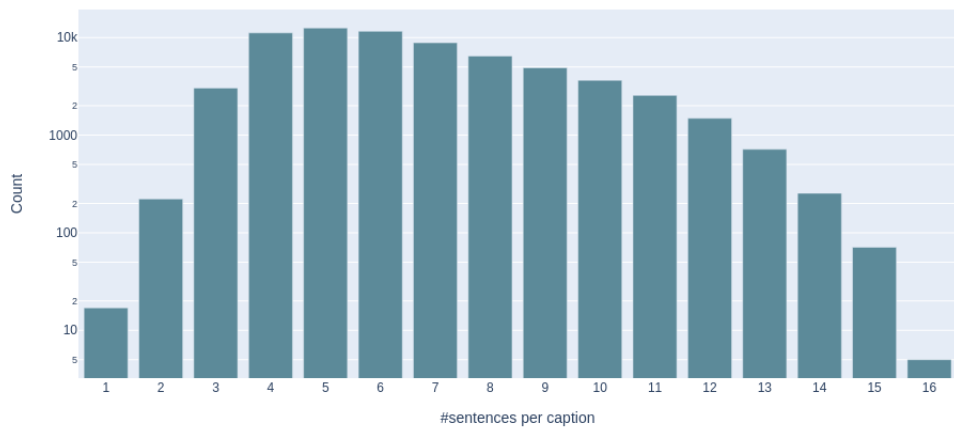
Category	Description
SWEATERS	Short sleeve rib knit jersey t-shirt in navy. Signature ‘slim’ fit. Mock neck collar. Stripes in pink and gold-tone at sleeves and bust. Tonal stitching.
TOPS	Short sleeve crewneck ribbon hemmed t-shirt in black. Tonal bar code stripe woven pattern throughout. Ribbed crewneck. Wide woven ribbon trim at hem. Tonal stitching.
PANTS	French terry slim-fit lounge pants in black. Elasticized woven cotton waistband with concealed drawstring. Button-fly. Three-pocket styling. Ribbed hems. Tonal stitching.

Table 4.1: Examples of descriptions of fashion items belonging to Fashiongen with the corresponding product categories.

We plot in figure 4.2 the number of words and the number of sentences per caption.



(a) Number of words per caption.



(b) Number of sentences per caption.

Figure 4.2: Statistics of the descriptions of clothing samples belonging to Fashiongen.

4.2. Private data sources

In the following sections, we describe three data sources that are property of our industrial partners. We cannot share these datasets, but we provide detailed descriptions of their features and statistics.

4.2.1. Industrial Dataset 1

The first private dataset we consider consists of high-resolution fashion images that are extensively annotated with descriptions and metadata. A clothing sample may have multiple photos that differ according to the particular pose of the model: each picture can illustrate the entire garment, the overall outfit of the model, or a detail of the fashion

item.

Metadata are either categorical data (classes) or attributes. The dataset contains 17 main categories, 12 colors, 102 more detailed subcategories, and 121 fine-grained colors; each fashion item may be annotated with multiple attributes among the 593 available. We provide the name and distribution of the main categories of the dataset in figure 4.3.

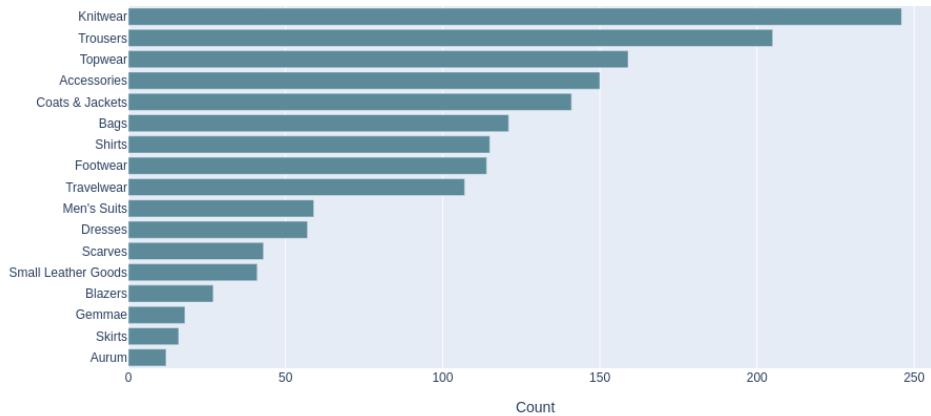


Figure 4.3: Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#1.

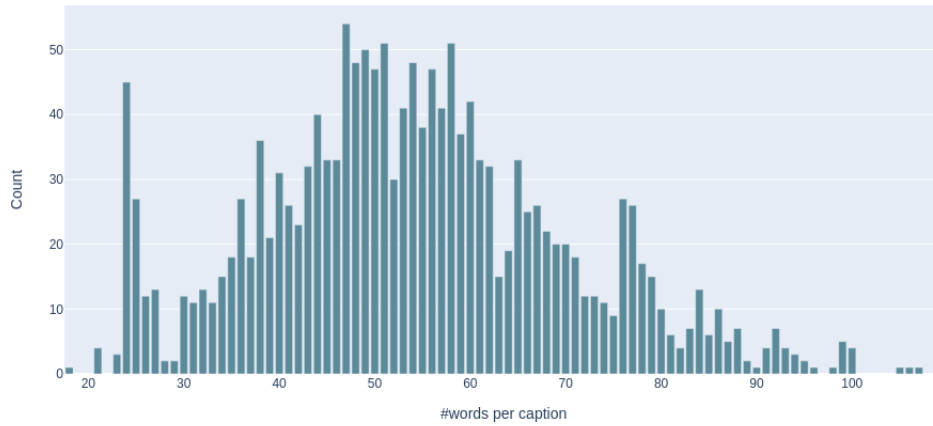
The captions of the fashion items follow a complex structure and refined words. Examples are available in table 4.2.

Category	Description
Shirts	Lightweight pure cotton striped Oxford enriches this spread collar shirt, a must-have menswear piece. Fitted lines that are close through the chest characterize the slim fit.
Topwear	Refined stretch silk satin enriches the essential design of the top with the excellence of materials. The slightly rounded silhouette pairs with the fabric's fluid effect to enrich summer looks with a shiny touch.
Trousers	The excellence of materials defines the style of these new Bermuda shorts, a must-have of the summer wardrobe. Slight color shading along the edges and seams characterizes the workmanship with the garment-dyed cotton gabardine, adding a sporty note to the style of this piece.

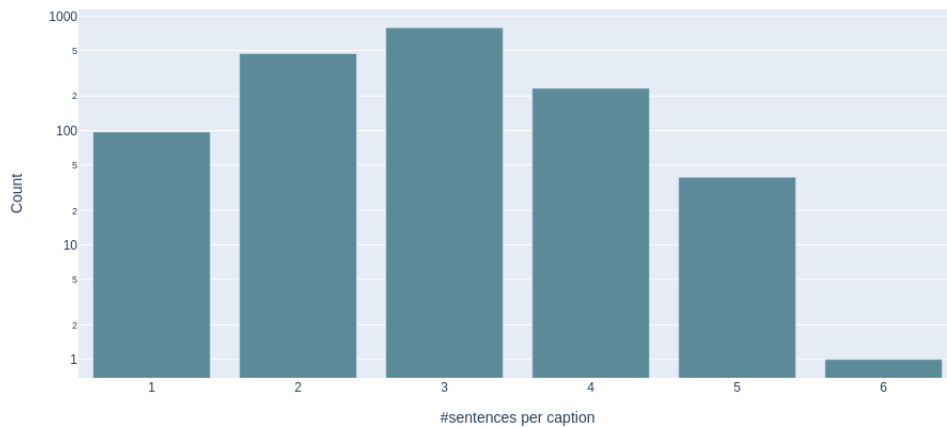
Table 4.2: Examples of descriptions of fashion items belonging to the industrial dataset#1 with the corresponding product categories.

We plot in figure 4.4 the distributions of the number of words and the number of sentences

per caption.



(a) Number of words per caption.



(b) Number of sentences per caption.

Figure 4.4: Statistics of the descriptions of clothing samples belonging to the industrial dataset#1.

4.2.2. Industrial Dataset 2

This second dataset consists of high-resolution images that, differently from Fashiongen and the industrial dataset#1, represent clothing items without fashion models wearing them. Besides the corresponding image and the textual caption, each clothing sample has a category representing its main class and a set of attributes describing details like the color, a possible subcategory, and wearability. The dataset contains 25 categories and 255 attributes; the plot in figure 4.5 shows the distribution of the categories of the clothing samples belonging to the dataset.

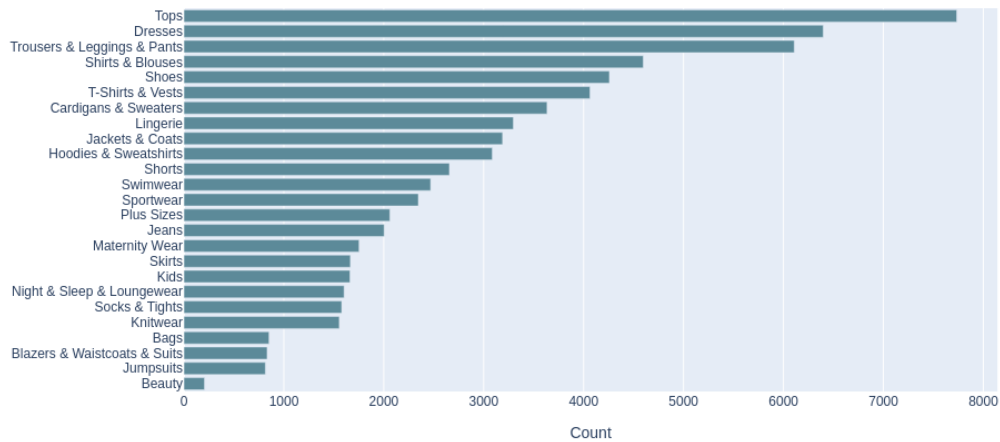


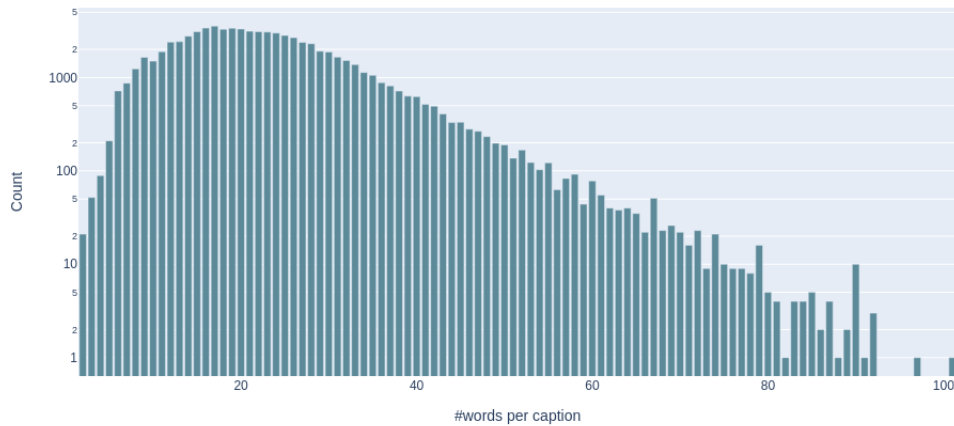
Figure 4.5: Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#2.

The captions of the clothing samples consist, on average, of a few short sentences; table 4.3 shows some examples.

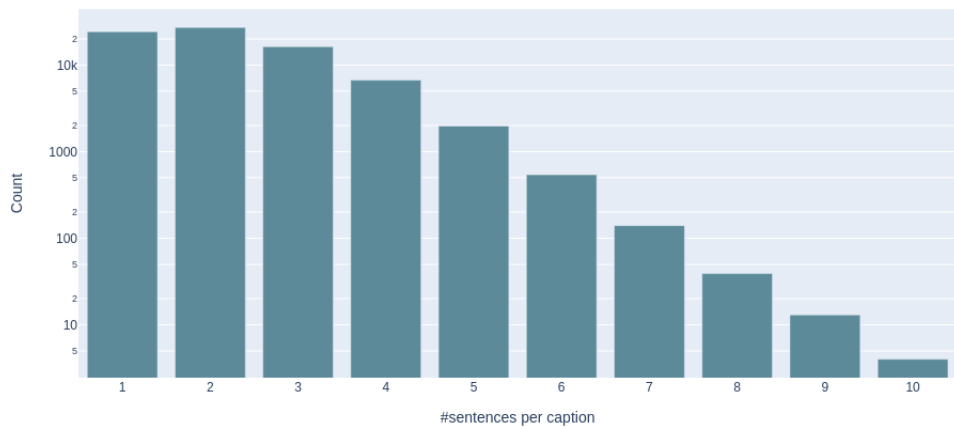
Category	Description
Knitwear	Cardigan in a soft knit with dropped shoulders long sleeves and front pockets. No buttons.
T-Shirts & Vests	T-shirt in soft cotton jersey. Regular Fit.
Shirts & Blouses	V-neck blouse in woven fabric with long sleeves with wide flared cuffs with a slit and a rounded hem with slits in the sides.

Table 4.3: Examples of descriptions of fashion items belonging to the industrial dataset#2 with the corresponding product categories.

Figure 4.6 provides the statistics about the number of words and sentences per caption.



(a) Number of words per caption.



(b) Number of sentences per caption.

Figure 4.6: Statistics of the descriptions of clothing samples belonging to the industrial dataset#2.

4.2.3. Industrial Dataset 3

The last dataset we use in our experiments is designed to tackle the challenge of *fashion outfit completion*. As for the industrial dataset#1, a clothing sample may have multiple pictures, each of them representing either the garment worn by a fashion model through different perspectives or the clothing product alone.

In addition to the description, each sample has a category and may have attributes regarding its composition. Besides, it is possible to infer the color of the garment: a clothing sample is identified by a unique string structured as `COLOR_CODE` (e.g., `NAVY BLUE_2253-401` and `BEIGE_3834-520`). Overall, the dataset contains 16 categories and 134 colors; the plot in figure 4.7 shows the distribution of the categories.

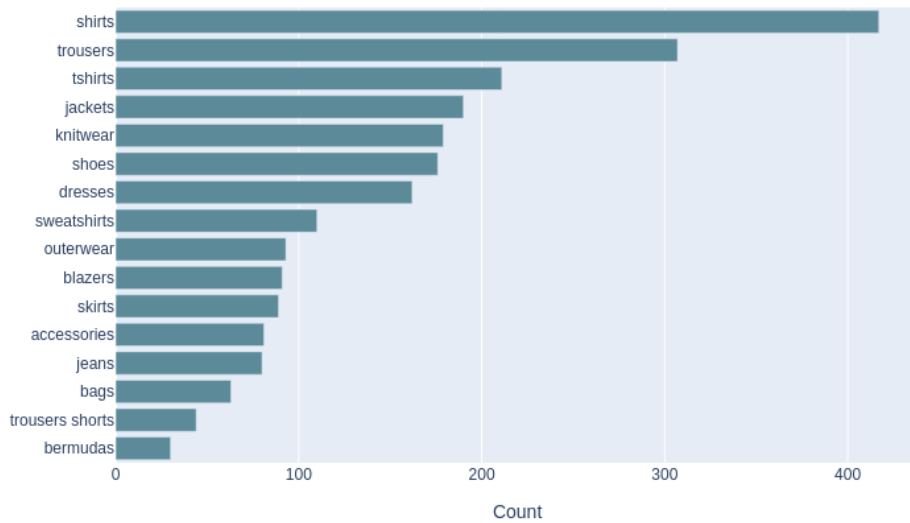


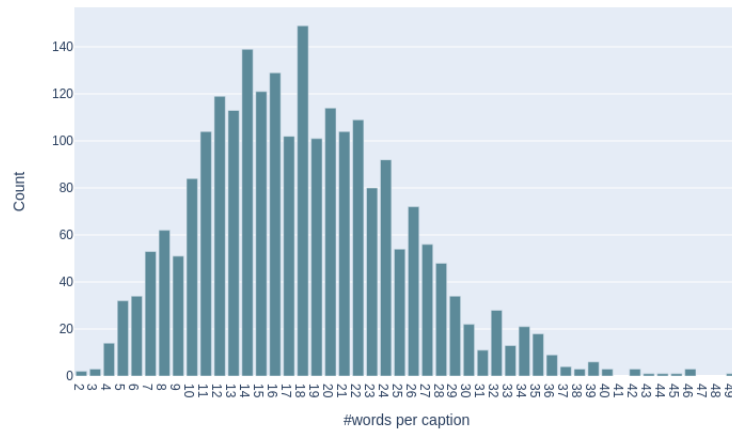
Figure 4.7: Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#3.

The captions of the clothing samples are short and usually single-sentence; table 4.4 shows some examples.

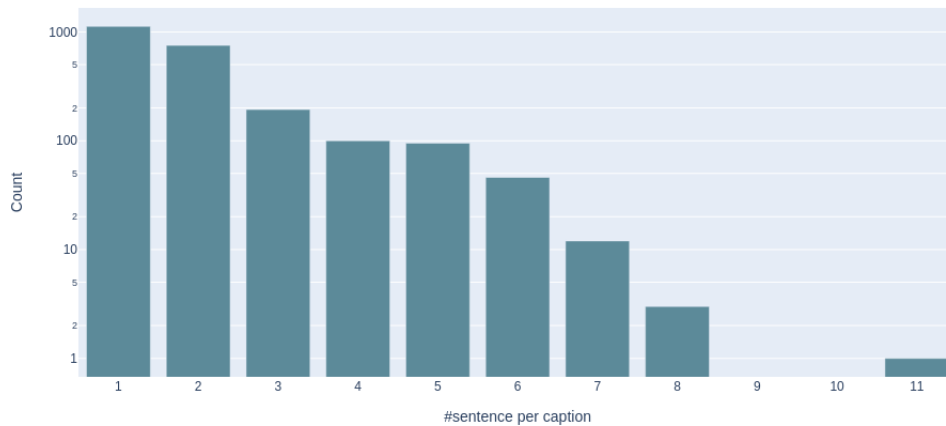
Category	Description
tshirts	High neck T-shirt with short sleeves.
knitwear	Loose-fitting short sleeve sweater made of cotton.
dresses	Dress with a straight neckline, thin straps with adjustable tie fastenings, and ruffle trims.

Table 4.4: Examples of descriptions of fashion items belonging to the industrial dataset#3 with the corresponding product categories.

Figure 4.8 provides the statistics about the number of words and sentences per caption.



(a) Number of words per caption.



(b) Number of sentences per caption.

Figure 4.8: Statistics of the descriptions of clothing samples belonging to the industrial dataset#3.

4.3. Dataset Comparison

The datasets described in the previous sections differ in their sizes and the structures of the captions of fashion items. Table 4.5 provides how we split the samples of each dataset along with the average number of words per caption \bar{w} , and the average number of sentences per caption \bar{s} .

Dataset	Train	Valid	Test	\bar{w}	\bar{s}
Fashiongen	54132	6015	7519	30.9	6.5
Industrial dataset#1	1235	153	138	54.1	2.9
Industrial dataset#2	39282	4365	5921	23.0	2.2
Industrial dataset#3	1881	233	209	18.0	1.9

Table 4.5: Number of items in train, validation, and test splits along with the average number of words (\bar{w}) and sentences (\bar{s}) per caption.

Fashiongen and the industrial dataset#2 are the datasets with the highest number of samples among the four available. Considering the average number of words and sentences per caption, the industrial dataset#2 and the industrial dataset#3 are similar, while Fashiongen has captions that contain several short sentences. On the contrary, the industrial dataset#1 contains clothing samples with very long captions and few sentences.

Figure 4.9 shows examples of images belonging to each of the four datasets.



(a) Fashiongen.



(b) Industrial dataset#1.



(c) Industrial dataset#2.



(d) Industrial dataset#3.

Figure 4.9: Example of fashion images of the dataset used in this thesis work.

5 | Pre-Training for Fashion Image Captioning

In this chapter we present our pre-training method for Fashion Image Captioning. We start by providing the goal of our analysis; afterwards, we dive into the challenges behind each aspect of our pre-training procedure, including processing data from multiple sources and a technique to improve the generalization capabilities of the model. Finally, we discuss why fine-tuning to a target domain is necessary for this application scenario.

5.1. Goal and Requirements

Our analysis aims to find a way to design image captioning systems applied to the fashion domain that achieve high-level performance without explicit training on the target dataset or with little adaptation through fine-tuning. In this way, it would be possible to get effective deep learning models even when constrained by resource requirements (limited computational time or memory, few or costly training samples, as well as environmental considerations).

To benefit from the information of other domains, in which the clothing products can differ in the way they are depicted in the images, the kind of metadata, and the style of the description, we follow an approach similar to the *multi-task* paradigm presented in section 2.1.5.

Multi-task learning usually refers to a pre-training procedure that solves multiple challenges simultaneously (here intended as “tasks”). Examples in NLP are models that tackle at the same time sentence classification, question answering, machine translation, and possibly others. We instead use this idea applied to the challenge of Fashion Image Captioning in a broader sense: captioning models learn simultaneously from different sources that highly differs in several aspects, such as the metadata labels associated with clothes, the poses and the surrounding in which garments are pictured, and the description to generate.

5.2. Data preparation

Our pre-training approach leverage the information of various sources, which are presented and compared in chapter 4. Textual data coming from multiple sources have differences that make it difficult for the model to learn general patterns without considering non-relevant information specific to a single dataset. Therefore, we have designed a set of textual transformations used to manipulate clothing metadata and captions to obtain a common format across datasets.

Table 5.1 lists these transformations, highlighting whether they are applied to metadata and/or captions.

	Description	M	C
<code>ToLowerCase</code>	It returns a string where all characters are in lower case. The models we train use subword-based tokenizers that differentiate between upper and lower case texts. Since clothing metadata are upper or lower case depending on the dataset, this transformation provides a format independent of the source dataset.	✓	
<code>Debranding</code>	It removes brand names from the input string.	✓	✓
<code>NumberReplacement</code>	It returns a string where percentages and numbers are replaced with <code><PERC></code> and <code><NUM></code> tokens, respectively.	✓	
<code>FilterDuplicates</code>	It returns the metadata string filtered by possible duplicates. This transformation avoids duplicate clothing tags in the metadata string, which can arise once the <code>NumberReplacement</code> and <code>Debranding</code> transformation are applied.	✓	

Table 5.1: Description of the textual transformations; the last two columns show whether a transformation is applied to metadata (**M**) and/or captions (**C**).

Table 5.2 shows samples before and after applying the transformation described above.

Scarves, Solid Scarf, Beige, Beige, Woman, Scarf measures approx. 70 cm x 220 cm, Short side with fringe	woman, solid scarf, scarf measures approx. <NUM> cm x <NUM> cm, beige, scarves, short side with fringe
SHIRTS, CLOTHING, Men, SHIRTS	clothing, men, shirts
Clothing, Trousers & Leggings & Pants, trousers, chinos, slim, beige, adult, <i>BRAND</i> man, man	clothing, trousers & leggings & pants, trousers, chinos, slim beige, adult, man
woman, sweatshirts, PETROL BLUE, 100% polyester, CHENILLE SWEATSHIRT	woman, sweatshirts, <PERC> polyester, petrol blue, chenille sweatshirt

Table 5.2: Metadata samples before (*left*) and after (*right*) applying the textual transformations.

5.3. Imbalance problem

The data sources employed during the pre-training stage can be very different in size, making the overall training set imbalanced. In this case, it gets difficult for a deep learning model to generalize over all the available sources without a bias towards the dominant one in the overall training set.

Machine learning solutions usually tackle this issue through over/under-sampling techniques. *Oversampling* of minor sources means the replication of random samples to get larger sources whose length is comparable to the length of the dominant one. *Undersampling* of the dominant source, instead, means the selection of a random subset of samples to get a smaller source whose length is comparable to the length of minor ones.

Both the techniques rebalance the distribution of samples of the training set but have drawbacks: oversampling introduces redundant data that can cause the model to overfit, furthermore the training time increases; undersampling can discard plenty of useful information. To avoid too long training procedures with redundant samples, but still leverage all the available information, we train the models using a stratified batch sampler algorithm together with a weighted loss function.

Batch sampler

During the pre-train, the model weights should be updated using batches that contain at least one sample per data source to ease the learning process and avoid single dataset specialization.

As explained before, standard oversampling would be too expansive in terms of training

time, so we implement a batch sampler that minimizes the total number of training batches without losing samples. The number of samples per dataset in the batch depends on all the sizes of the data sources used for pre-training the model and can be optimized by solving the minimax problem in equation 5.1.

$$\begin{aligned} & \min \max_i \frac{L_i}{x_i} \\ & \text{subject to } \sum_{i=1}^n x_i = BS, \\ & \quad x_i \in \mathbb{N}^+, \quad i = 1, \dots, n \end{aligned} \tag{5.1}$$

The batch size BS and the length of the n data sources L_i are the input of the ILP optimization. The total number of batches is determined by $\max_i \frac{L_i}{x_i}$. The goal of the optimization is to determine the number of samples per batch for each data source that minimizes the total number of batches, thus the optimization needs a *minimax* objective function. All the training batches are complete, so a limited number of data points are replicated to compose the final batches of the epochs and avoid truncation.

Weighted train loss

While the batch sampler allows controlling the number of batches, thus the training time, the weights in the loss function give more importance to samples coming from sources less prevalent in the training batch.

As our goal is to learn a language model, which translates into the estimation of the conditional probability distribution over labels of the model vocabulary, the optimization criterion we use is the *cross-entropy* loss. Equation 5.2 shows this error function for a given step t , where V is the vocabulary, s^t is the t^{th} label of the input sequence, and \hat{y}^t is the probability distribution over the vocabulary at the t^{th} step, i.e., how probable is each label of the vocabulary given the input up to $t - 1$.

$$l^{(t)}(s) = - \sum_{v \in V} s_v^t \log(\hat{y}_v^t) \tag{5.2}$$

When dealing with a sequence of length T , the *cross-entropy* error over the entire input is:

$$l(s) = \frac{1}{T} \sum_{t=1}^T l^{(t)}(s) = -\frac{1}{T} \sum_{t=1}^T \sum_{v \in V} s_v^t \log(\hat{y}_v^t) \tag{5.3}$$

At each training step, to make each data source contribute the same as the other, independently by its frequency in the batch, we aggregate the loss of the batch samples as in equation 5.4:

$$L = \sum_{i=1}^{BS} \frac{w(s_i)l(s_i)}{\sum_{i=1}^{BS} w(s_i)} \quad (5.4)$$

Weights are computed as the reciprocals of the frequencies of the data source in the batch:

$$w(s) = \frac{BS}{\#\text{samples coming from the same source of } s} \quad (5.5)$$

Figure 5.1 shows an example when the training batches contain 8 samples.

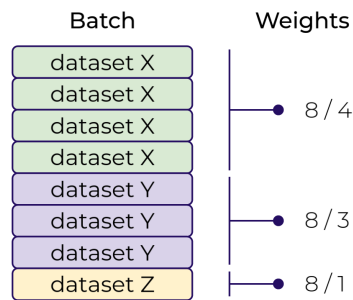


Figure 5.1: Example of weights given a batch assignment.

5.4. Noise generation

In this section, we present how it is possible to combine existing clothing samples to improve the generalization capabilities of a fashion captioner.

Given that the data sources have different probability distributions over the feature space, learning a common representation across datasets allows the model to catch and process correctly more details in input samples; nevertheless, the model is prone to recognize which is the data source that most probably includes the input sample. Due to this behavior, the model generates the final caption using a generative style and information related only to the dataset identified as the most suitable for the current input.

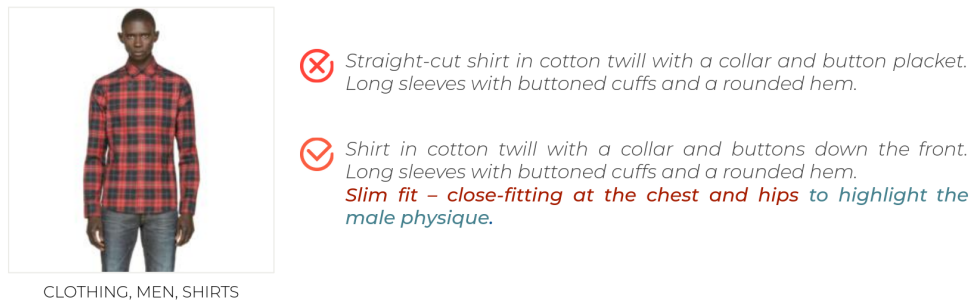


Figure 5.2: The input image and the metadata are from *Fashiongen*, which is not used in the pre-training stage. The upper generated caption is less desirable as it relies only on captions seen in the *industrial dataset #3*; contrary, the lower one mixes captions seen in *industrial dataset #3* and *#1*.

We present a method that introduces *noise* in the training samples as forged products that mix data coming from different sources: in this way, the model is more robust and able to generate more general caption of clothes without focusing only on a single source of information, as the unintended discrimination among the training data sources is now prevented.

5.4.1. Fashion taxonomies

We introduce noise in the training samples by combining products of different data sources: a forged product has the image and the metadata coming from a dataset X while the target caption from another dataset Y . The goal is to teach the model that both the semantic and the syntax of the generated captions are independent by not relevant features like the background of the image, having or not a human wearing the garment, or the metadata format (e.g., the order of words).

The way the samples are mixed affects the performance of the captioner: combining products at *random*, despite the ease of development, confuses the model as there is no semantic relationship between the two samples merged; contrary a *one-to-one* mapping between products of different sources would be very effective but too complex to achieve. Our approach uses a reference *taxonomy* that guides the matching between products of various sources, starting from the metadata associated with each product.

Dataset taxonomy

The first step is to build a *taxonomy* for each source involved in the pre-training.

Each dataset organizes the metadata associated with products in a tabular structure

specific to the particular dataset. This structure consists of a set of fields and a list of admissible values for each of them. In principle, there is no hierarchical relationship between the fields; however, there are fields more general than others that instead provide low-level details about the products, making it possible to infer a hierarchy like “*gender*” → “*product department*” → “*category*”. Values admissible for “*product department*” could be “*footwear*” and “*clothing*”; while possible “*category*” values for items whose “*product department*” is “*footwear*” could be “*sneakers*” and “*boots*”.

We follow this insight to build a taxonomy of a dataset: first, we infer a hierarchy among the metadata fields; then, we scan the products of the dataset to build a tree according to the previously defined hierarchy. Algorithm 5.1 shows the pseudo-code of the procedure.

Algorithm 5.1 Build taxonomy from dataset.

Require: D : dataset, H : hierarchy of fields of D

```

1:  $T \leftarrow Tree()$ 
2:  $T.root \leftarrow "root"$  //Insert a helper root in the tree
3: for  $d$  in  $D$  do
4:    $node \leftarrow T.root$ 
5:   for  $h$  in  $H$  do
6:     if  $d.h$  is not child of  $node$  then
7:        $insert(node, d.h)$  //Insert the value of  $h$  as child of  $node$ 
8:     end if
9:      $node \leftarrow d.h$ 
10:  end for
11: end for
12: return  $T$ 

```

By running algorithm 5.1 over all the data sources involved in the pre-training, we obtain a taxonomy for each of them. Results are available in appendix A.

Reference taxonomy

Now that there is a way to get a tree-shaped taxonomy for each dataset, the challenge of matching similar products across sources translates into finding pairings between leaves of the trees. Assignments between taxonomies are manually designed by inspecting the trees, and due to different levels of granularity in describing the garments through metadata, pairings between leaves are *not* always symmetric. Figure 5.3 shows an example.

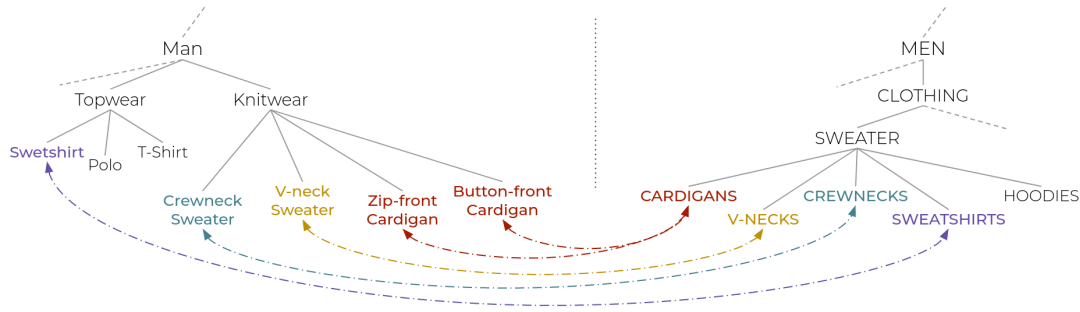


Figure 5.3: Pairings between two taxonomies when dealing with *sweaters*. The connections are usually symmetric, except for *cardigans*. The taxonomy on the left (*industrial dataset #1*) is more granular than the one on the right (*fashiongen*).

Having N taxonomies to match together with connections that are not always symmetric implies at least $\frac{N \times (N-1)}{2}$ set of rules to be coded. This process is expensive and time-consuming, so we define an additional reference taxonomy \mathcal{T}^* that acts as a proxy: each dataset taxonomy \mathcal{T}_n is mapped to the reference \mathcal{T}^* , reducing the total number of mappings to be coded to N . If clothing products of various sources share a leaf of the tree of \mathcal{T}^* , then they are *similar*, thus suitable to forge hybrid samples.

We choose to define a new taxonomy \mathcal{T}^* instead of using already available ontologies like *Fashionpedia*: “an ontology built by fashion experts containing 27 main apparel categories, 19 apparel parts, 294 fine-grained attributes, and their relationships”¹. This complex ontology mainly provides semantic relationships between apparel parts of garments, which is not the kind of relationship we require: we find out that only a tight portion of the overall ontology might help in our use case, so we opt to design a new taxonomy. It is entirely available in Appendix A, while figure 5.4 shows only a subset.

¹<https://fashionpedia.github.io/home/>

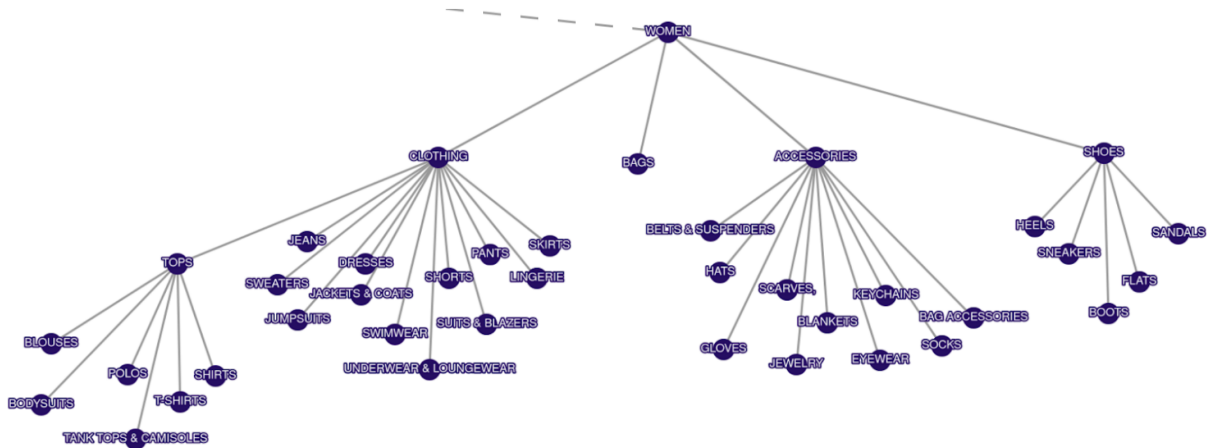


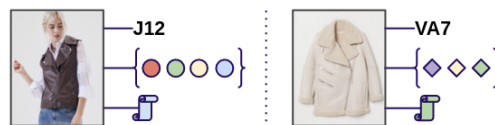
Figure 5.4: Subset of the reference taxonomy.

5.4.2. Generation of forged samples

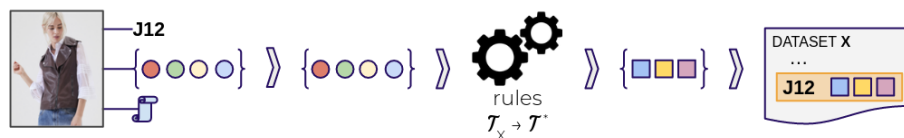
The generation of *forged* (or *hybrid*) samples leverages the mappings to the reference taxonomy \mathcal{T}^* to introduce noise in the training samples by mixing similar clothing products. As specified in the previous section, two products of various sources are similar, therefore joinable, if they share the same leaf of the tree of \mathcal{T}^* .

Hybrid samples do not replace the original ones; they are new clothing products added to the overall training set. Here we describe the entire generation process.

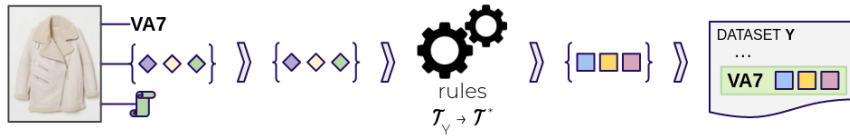
Each training sample consists of the triple $\langle \text{image}, \text{metadata}, \text{caption} \rangle$, identified through an *id*. The example below shows two product samples of two distinct datasets.



Before starting the training procedure, iterating over a source dataset X allows the extraction of the metadata of each clothing sample s in X . The coded rules $\mathcal{T}_X \rightarrow \mathcal{T}^*$ convert the metadata from the structure defined by the dataset taxonomy \mathcal{T}_X to the structure of the reference \mathcal{T}^* . The *id* of s and its converted metadata are stored to be used later during the pre-training.



The same procedure is repeated for all the datasets.



At training time, with probability p , a product a is merged into another b to build a new hybrid sample h . The mapping computed before allows retrieving the metadata of a , represented according to the structure defined by \mathcal{T}^* : in this way, a similar product b is selected by scanning the previously computed mapping of another dataset (chosen at random). The final hybrid sample h contains the image and the metadata of a and the caption of b .

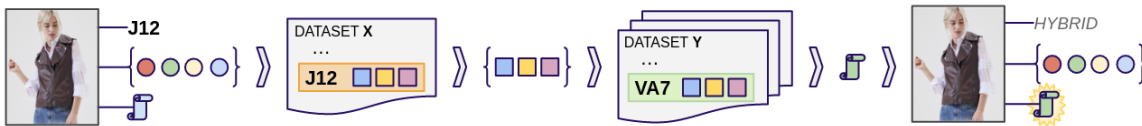


Figure 5.5 shows a sample forged using the procedure described above.



Figure 5.5: Hybrid sample: it introduces noise in the training set by mixing similar clothing products.

5.5. Fine-tuning

As discussed in 2.1.3, fine-tuning is an adaptation paradigm that aims to transfer and apply the knowledge learned by a model into a new scenario.

In chapter 4, we show the main difference among the data sources we consider by comparing the various styles of the captions. When dealing with fashion captioning systems,

besides producing a semantically correct description, the model should generate captions according to a particular structure: fine-tuning allows matching the stylistic decisions of a fashion firm in describing garments through captions.

In our analysis, the pre-trained model is fine-tuned to a target dataset that was not used during the previous learning stage. We only use a restricted fraction of the clothing samples available (5%) and a fixed limited number of epochs (5) to simulate a scenario in which few training samples are available, and the learning procedure has a time constraint.

6 | Contrastive Learning: approach and model architecture

We present a new architecture for Fashion Image Captioning that takes inspiration from recent works that exploit *contrastive learning* objectives to tackle joint Computer Vision and NLP tasks. In this chapter, first, we describe our Transformer-based model and how it processes multi-modal input samples; afterward, we describe the pre-training options that arise when considering a contrastive objective.

6.1. Vision-Text Multi-Encoder Decoder

The Transformer architecture [Vas+17] processes textual input samples through two “towers” of blocks: the encoding and the decoding stacks. Over the years, the research community has revised this milestone architecture proposing several variations that exploit one of the two stacks of blocks: BERT [Dev+19], an encoder-only Transformer, and GPT-2 [Rad+19], which instead use only decoding blocks.

Transformer-based models achieve state-of-the-art performance in all the NLP challenges, and recently its application with images and joint image-text samples is being studied. As analyzed in section 3.2, joint CV and NLP challenges like Image Captioning deal with visual and textual sources, and models that pursue these challenges usually leverage a projection or an adaptation layer to map the visual features into the very same stack of blocks of the transformer-based architecture.

In the Fashion Image Captioning scenario, an input sample consists of a triple `<image, metadata, caption>`, where `image` and `metadata` are two distinct but complementary modalities in describing a garment, so their hidden representations should be quite alike as they refer to the same clothing product.

Driven by this insight, we propose a Transformer-based architecture named *Vision-Text*

Multi-Encoder Decoder that learns the visual and the metadata hidden representations using two different encoding stacks that keep the two modalities apart. It generates captions of clothing products focusing on images and metadata by exploiting two cross-attention layers on the decoding side. Figure 6.1 shows an overview of the model architecture.

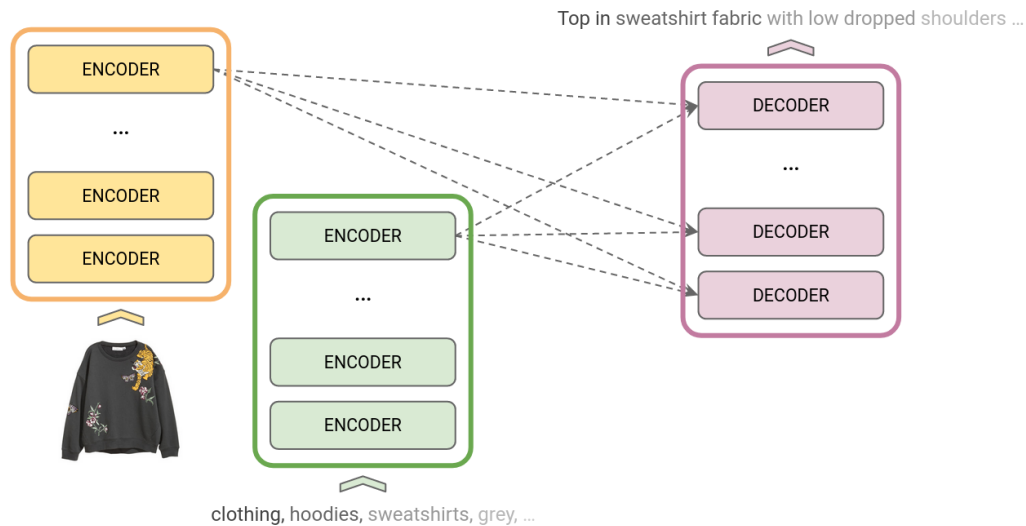


Figure 6.1: Vision-Text Multi-Encoder Decoder architecture.

The encoders learn their hidden representations according to the overall language modeling objective but mutually conditioned by a *contrastive* restraint that keeps their representations close. This additional contrastive loss is described later on in the chapter.

6.1.1. Extension of the Transformer decoder block

The decoder block proposed in [Vas+17], here presented in figure 6.2a, has a *cross-attention* layer that uses *key* and *values* vectors obtained by the output of the top encoder block, and *query* vectors from the previous block of the decoding stack. The Transformer leverages the cross-attention mechanism to focus on relevant parts of the input during the generation process.

Our model requires that the decoder attends not only the output of a single encoding stack but two of them, as the visual and textual modalities are treated separately. To do so, we extend the standard decoder block by adding a *multi-head attention layer* and an additional *normalization layer* as reported in figure 6.2b.

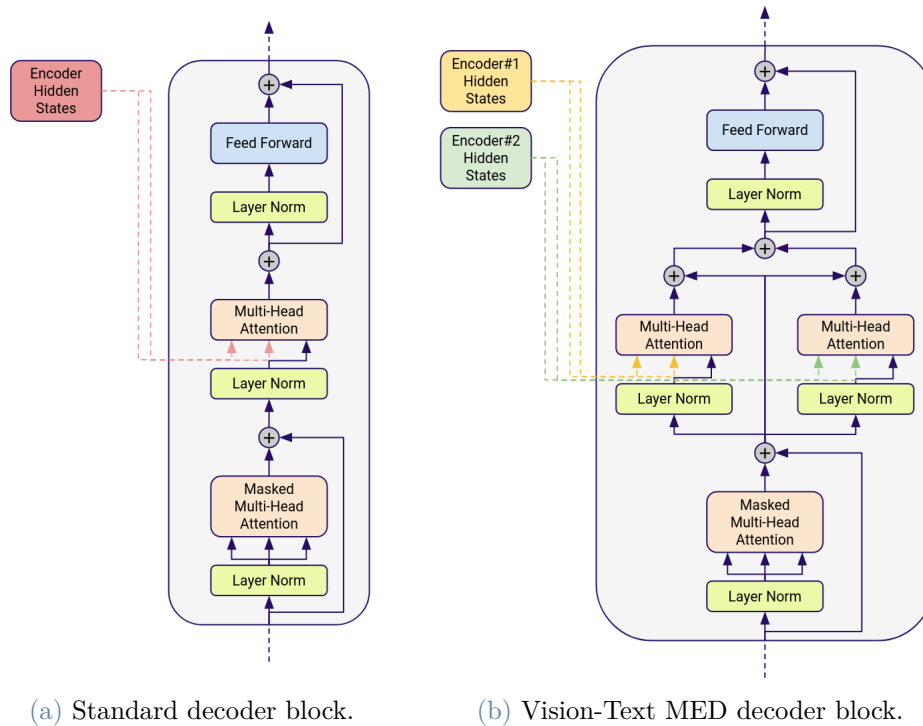


Figure 6.2: Comparison between the decoder blocks of the standard Transformer and Vision-Text Multi-Encoder Decoder.

This extension of the decoder block is independent of the modality of the encoders (visual/textual). This structure is also proposed in [Zho+20], where used in a transformer architecture that faces the challenge of *code-switching* for *speech recognition*.

6.2. Contrastive objective

The image and metadata are complementary ways to describe a clothing product, and ideally, regardless of the modality, their embeddings are nearby in shared representation spaces. To promote this behavior, our *Vision-Text Multi-Encoder Decoder* leverages the contrastive pre-training objective proposed in CLIP [Rad+21], applied to images and textual metadata of the fashion domain. Figure 6.3 provides an illustration of the pre-training steps.

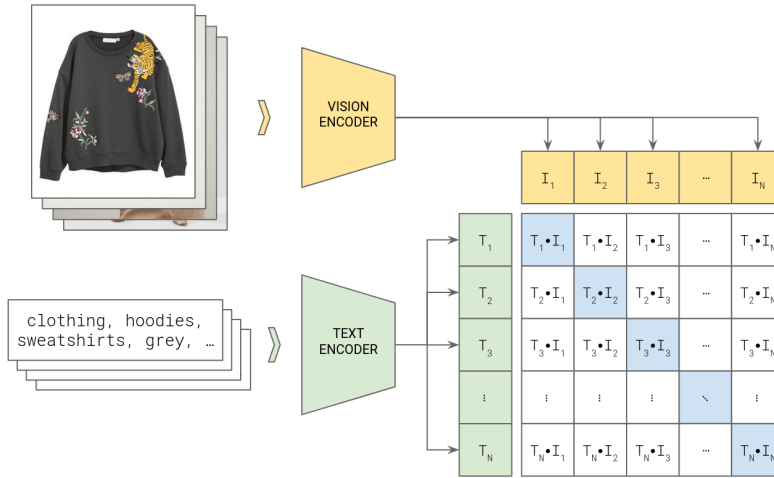


Figure 6.3: The vision and the text encoders are jointly trained to predict the correct pairings between images and metadata of the current training batch. Image adapted from [Rad+21].

The goal is the prediction of the correct pairings between images and metadata of the current training batch. Given a batch size of N , there are $N \times N$ pairings, so $N^2 - N$ incorrect matchings. In practice, the embeddings of the image encoder and the text encoder are projected to a *multi-modal* representation space, and a similarity measure (*cosine*) is maximized for the N correct pairs of image and metadata embeddings while minimized for the other $N^2 - N$. To perform this optimization is used a symmetric cross-entropy computed over the similarities scores. Algorithm 6.1 shows the pseudo-code to compute the final loss.

Unlike CLIP, which is used for Image Classification, our goal is the generation of captions of clothing products having as input their images and metadata. The model generates the captions through a decoding stack trained using a language modeling objective, while the overall learning procedure can leverage the contrastive pre-training in two different ways: using a multi-objective loss function or through a pre-alignment stage.

6.2.1. Multi-objective optimization

To take advantage of the contrastive alignment of vision and text embeddings, one option is to train *simultaneously* the model with the language modeling objective (*cross entropy*) and the loss computed as in Algorithm 6.1 in a multi-objective manner:

$$L = \lambda L_{contrastive} + (1 - \lambda) L_{c.e.} \quad (6.1)$$

Algorithm 6.1 Contrastive loss computation.

Require: I : set of N images; T : set of N textual metadata;

```

//extraction of image and metadata features
1:  $I_f \leftarrow \text{vision\_encoder}(I)$  //shape:  $N \times d_{\text{image}}$ 
2:  $T_f \leftarrow \text{text\_encoder}(T)$  //shape:  $N \times d_{\text{text}}$ 
//projection to a shared embedding space
3:  $I_e \leftarrow \text{vision\_projection}(I_f)$  //shape:  $N \times d_e$ 
4:  $T_e \leftarrow \text{text\_projection}(T_f)$  //shape:  $N \times d_e$ 
//normalization and computation of the scaled cosine similarity matrix
5:  $I_e \leftarrow \text{normalize}(I_e)$ 
6:  $T_e \leftarrow \text{normalize}(T_e)$ 
7:  $S = I_e \cdot T_e \times \text{scale\_param}$  //shape  $N \times N$ 
//loss computation
8:  $\text{labels} = \text{range}(0, N - 1)$ 
9:  $\text{loss}_{\text{image}} = \text{cross\_entropy}(S, \text{labels}, \text{axis} = 0)$ 
10:  $\text{loss}_{\text{text}} = \text{cross\_entropy}(S, \text{labels}, \text{axis} = 1)$ 
11:  $\text{loss} = \frac{\text{loss}_{\text{image}} + \text{loss}_{\text{text}}}{2}$ 
12: return  $\text{loss}$ 

```

where the hyperparameter $\lambda \in [0, 1]$.

In this way, the pre-training procedure has a single stage in which the model is trained according to a joint generative and contrastive objective function.

6.2.2. Pre-alignment

Another option that arises when considering the contrastive objective is to split the pre-training procedure into two separate stages: a former that performs the self-supervised alignment of the encoders embeddings and a second one in which the overall model is updated based on the generative criteria. The model input in the first stage consists of `<image, metadata>` pairs, while in the second one the `caption` of clothing products is also included.

As we describe in chapter 4, two out of four data sources have multiple images associated with the same product sample, and when training an image captioner to learn a language model, we use the most relevant image among the available ones. When performing the initial contrastive alignment, we add the possibility to include also less informative images through a *weighted random selection*: if available, the data source organizes the images of a product according to their importance (for example, first the frontal picture, then the complete outfit, and lastly the cropped details of the garment); we associate a probability to each image proportional to its importance.

Since the number of images associated with each product is variable, we compute the probabilities of each image using a cumulative distribution function defined as:

$$F_X(k) = \begin{cases} \tanh(\alpha \cdot k), & \text{if } k < n_{IMAGES} \\ 1, & \text{if } k = n_{IMAGES} \end{cases} \quad (6.2)$$

where α is a hyperparameter that we set equal to $\frac{2}{n_{IMAGES}}$. Figure 6.4 shows an example.

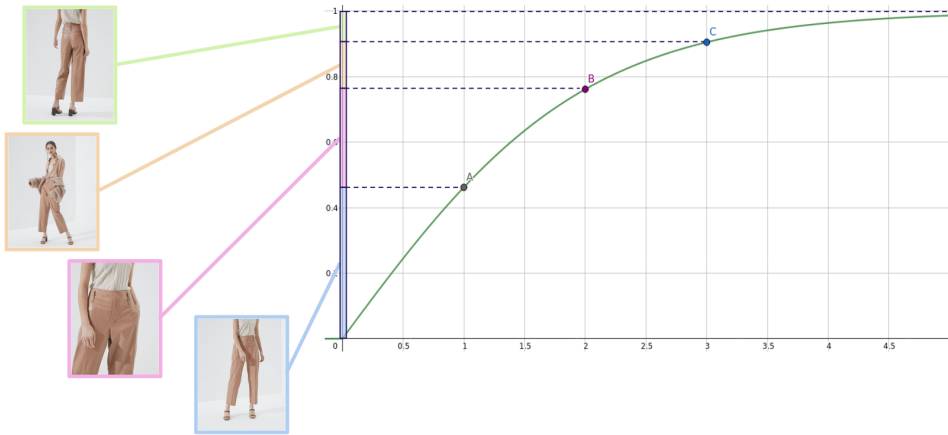


Figure 6.4: Weighted random selection of the images of a garment when the total number of images is four.

7 | Experiments and Results

7.1. Evaluation metrics

To compare the results of several experiments involving image captioning systems, we use a set of evaluation metrics that use different criteria to measure the correspondence between the machine computed sentences and the ground truth captions. The scores we consider in the performance evaluations are BLEU, GLEU, METEOR, and ROUGE: each generated caption of the model under analysis is compared with its reference, computing a single score; then they are averaged over the whole test partition to provide an estimate of the overall quality of the image captioner.

BLEU

The underlying idea of BLEU [Pap+02] to measure the performance of a machine translation is that “the closer a machine translation is to a professional human translation, the better it is”. BLEU- n measures the similarity between a candidate text with a reference by using a modified n -gram precision.

The classic n -gram precision computes the number of n -grams in the candidate text which occur in the reference (ground truth) divided by the total number of n -grams in the candidate text. Reaching a high precision does not always imply a high-quality generated text: generating a large number of times an n -gram which is in the reference text reflects in a high precision score even though the resulting candidate is highly improbable. Therefore, BLEU- n computes a modified n -gram precision by first counting the maximum number of times an n -gram occurs in the reference text, then the total count of each candidate n -gram is clipped by its reference count. Finally, the sum of the *clipped* count of each candidate n -gram is divided by the *unclipped* count of the n -grams in the candidate text. Table 7.1 includes the example proposed in [Pap+02].

Candidate	<u>the</u> <u>the</u> the the the the the.
Reference #1	<u>The</u> cat is on <u>the</u> mat.
Reference #2	There is a cat on the mat.

Table 7.1: The *unigram precision* is $\frac{7}{7}$, as the word “*the*” is a matching unigram that appears 7 times in the candidate text, whose total number of unigrams is 7. The *modified unigram precision* is instead $\frac{2}{7}$: the clipped count of the word “*the*” is 2, which is the maximum count of that unigram in any reference text. The words used to compute the modified precision are underlined.

A slightly different score known as GLEU (Google BLEU) [Wu+16] was designed to be a single sentence metric instead of a corpus measure. It computes the 1, 2, 3, and 4-grams of the candidate and target sentences, then provides as final score the *minimum* between the *recall* and *precision*, where the former is the ratio between the number of matching n -grams and the total number of n -grams in the ground truth sentence, and the latter is the ratio between the number of matching n -grams and the total number of n -grams in the candidate sequence. This score is still in $[0, 1]$ and is symmetrical when switching the candidate and target sentences.

METEOR

METEOR [BL05] is a machine translation metric that relies on *unigram matching*, but differently from other scores, it leverages linguistic properties to make matchings that take into account the root and the meaning of words by using stemming and synonyms matching techniques.

Given a candidate and a reference sentence, METEOR computes an F-score weighted through a penalty term that measures the level of *fragmentation* of the matched unigrams in the reference sentence. Unigram precision P and recall R are computed as:

$$P = \frac{m}{u_C}, \quad R = \frac{m}{u_R}$$

where m is the number of unigrams of the candidate sentence that are also in the reference, u_C is the total number of unigrams in the candidate sentence, and u_R is the total number of unigrams in the reference text. The F-score weights the recall nine times more than the precision:

$$F = \frac{10PR}{P + 9R}$$

To take into account how the matching of unigrams reflects over larger portions of texts, METEOR computes an alignment between the two sentences. An alignment is a mapping between unigrams of the two sequences, such that every unigram can be associated with at most one unigram of the other. Matched unigrams that are adjacent both in the candidate and in the reference compose a *chunk*. METEOR computes the alignment that has the fewest possible number of chunks. The penalty term P is computed as follows:

$$penalty = 0.5 \times \left(\frac{\#chunks}{\#unigrams_matched} \right)^3$$

It decreases as the number of chunks decreases while it is at most 0.5 if there are no matched *bigrams*. The final meteor score is:

$$METEOR = F \times (1 - penalty)$$

ROUGE

ROUGE [Lin04] is a set of metrics devised to evaluate automatic summarization but generally adopted when dealing with machine translations. Among all the available ROUGE measures, we consider ROUGE-N and ROUGE-L.

Given two sentences, a candidate and a reference, ROUGE-N computes the n -gram recall between the candidate and the reference as

$$\text{ROUGE-N} = \frac{\text{number of matching } n\text{-grams}}{\text{total number of } n\text{-grams in the reference}}$$

ROUGE-N is formally defined as a recall measure, but it is possible to extend this metric considering either the precision or the F-score.

ROUGE-L computes the *longest common subsequence* between the generated text and the reference sentence. The idea proposed in [Lin04] is that “the longer the LCS is, the more similar the two texts are.” Given a candidate sequence X and a reference text Y , whose lengths are m and n respectively, the final score F_{lcs} is:

$$R_{lcs} = \frac{LCS(X,Y)}{n}, \quad P_{lcs} = \frac{LCS(X,Y)}{m}$$

$$F_{lcs} = \frac{(1 + \beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2P_{lcs}}$$

The advantage of ROUGE-L is that it considers in-sequence matches instead of exact

consecutive matchings; moreover, it does not require a predefined n -gram length.

7.2. Experimental setup

We performed all the experiments using Amazon Web Services¹ instances (p2.xlarge) equipped with one NVIDIA Tesla K80 GPU.

The models along with the training and evaluation procedures are developed in Python using PyTorch² and PyTorch Lightning³ frameworks. The core implementation of the Transformer-based architectures used in our experiments is available in the Hugging Face Transformers⁴ library. This library provides thousands of pre-trained models that deal with different modalities such as text, vision, and audio.

The visual processor component of each image captioner model we analyze receives as input a normalized tensor of size 224x224, which is randomly horizontally flipped with 50% probability during training. All the experiments have been conducted using an *early stopping* callback and the *Adam* optimization algorithm with a *plateau* learning rate scheduler. The scheduler allows the learning rate to decrease by a 0.1 factor when the validation loss stops improving; we set the *patience* parameter of the early stopping callback as $\sim 2\times$ the one of the scheduler.

7.3. Model comparison

In this section, we analyze the performance of different approaches of Image Captioning systems that deal with fashion products. The architectures under observation are based on the ones presented in chapter 3.

Show, Attend, and Tell vs Image GPT-2

The first architecture we analyze is the one proposed in *Show, Attend, and Tell* [Xu+15]. It consists of a ResNet encoder that process images, an LSTM decoder to generate captions of products, and an attention layer that allows the decoder to focus on relevant parts of the input image at each decoding step. More details of this captioning architecture are available in section 3.2.1.

We compare this model with *Multimodal GPT-2* presented in [PSC20]. *Multimodal GPT-*

¹<https://aws.amazon.com/>

²<https://pytorch.org/>

³<https://www.pytorchlightning.ai/>

⁴<https://github.com/huggingface/transformers>

2, described in sec 3.2.3, is an encoder-decoder architecture that exploits a ResNet encoder and the GPT-2 Transformer as decoding module. To perform a fair comparison, as the *Multimodal GPT-2* generates captions receiving as input images and metadata while the *Show, Attend, and Tell* model process only images, we do not use the additional textual metadata as input of *Multimodal GPT-2*, here referenced simply as *Image GPT-2*. Additionally, since GPT-2 leverages an extensive pre-training over almost 40 GB of textual sources as initialization of the token embeddings, we initialize the embeddings of the *Show, Attend, and Tell* decoder using the one of GPT-2. The goal is to leverage the expensive pre-training on both two architectures: to do so, besides the embeddings initialization, the *Show, Attend, and Tell* decoder uses the vocabulary and the tokenizer of the pre-trained GPT-2 model. This is necessary to enforce that the *Show, Attend, and Tell* decoder module maps word pieces of captions in the correct token embeddings, as learned by the pre-trained GPT-2 model.

Results Table 7.2 summarizes the results obtained by the two models on the *Fashiongen* dataset and the *industrial dataset#1*.

Algorithm	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Results on Fashiongen					
Show, Attend, and Tell	37.39	22.89	41.49	53.12	50.33
Image GPT-2	48.19	29.56	51.70	61.56	57.83
Results on the industrial dataset#1					
Show, Attend, and Tell	28.65	12.95	23.32	35.57	26.09
Image GPT-2	31.85	16.03	27.33	38.25	29.55

Table 7.2: Results of *Show, Attend, and Tell* and *Image GPT-2* on the Fashiongen dataset and the industrial dataset#1.⁵

According to the results, *Image GPT-2* achieves a higher score than the *Show, Attend, and Tell* model in all the metrics. Both architectures use a ResNet-150 visual feature extractor but differ from their decoding module and the way the attention is computed. *Image GPT-2* uses a GPT-2 decoding module that leverages the self-attention layer of the Transformer to focus on the input and all the previously generated textual tokens. Using this layer in each block of the decoding stack, *Image GPT-2* captures fine-grained details of the input image and achieves better performance than the *Show, Attend, and Tell* model, which instead uses an LSTM decoding module.

⁵The table reports the model results according to a subset of metrics: the ‘BLEU’ and ‘ROUGE’ scores refer to BLEU-1 and ROUGE-1, respectively. Scores related to other choices of n -grams are consistent with the ones provided, thus omitted. More detailed results are available in Appendix B.

OSCAR vs Multimodal GPT-2

OSCAR [Li+20] is one of the most common architectures for Image Captioning, so we analyze its performance when applied to caption fashion products. As described in 3.2.2, OSCAR defines a pre-training method that processes as input the regions of the input image along with the object tags provided by an object detection module. Its performance relies on the ability of the object detector to correctly provide object tags and relevant regions of the input images: OSCAR uses a Faster R-CNN to extract a set of visual features along with their semantics in the form of textual labels, as for the example shown in figure 7.1.

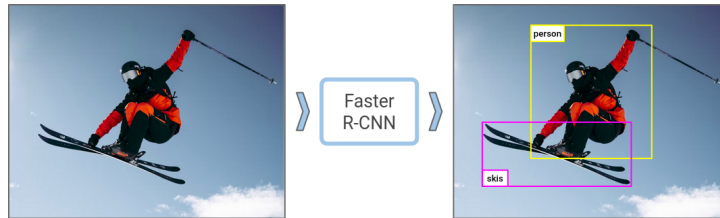


Figure 7.1: Example of the visual semantics extracted by the object detector module used by OSCAR.

Since in our scenario the metadata associated with clothing products are already available, there is no need to rely on modules that provide such information: we build training samples as the triple $\langle \text{image}, \text{metadata}, \text{caption} \rangle$ and analyze the OSCAR pre-training method applied to its BERT-based architecture without relying on an object detection module. In our analysis, we refer to this architecture as *OSCAR**. Figure 7.2 shows how we adapt the architecture presented in [Li+20] to our use case.

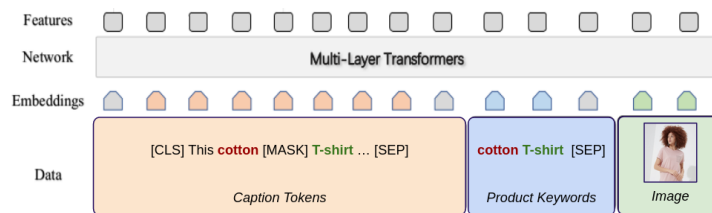


Figure 7.2: Our adaptation of OSCAR for Fashion Image Captioning.

We choose to compare this model with *Multimodal GPT-2* because the former reaches state-of-the-art performance on several datasets available for Image Captioning by using a masked language model objective, and the latter is specifically designed to deal with fashion clothing products and reaches high-level performance by solving a language modeling task using a generative learning objective.

Results Table 7.3 summarizes the results obtained by the two models on the *Fashiongen* dataset and the *industrial dataset#1*.

Algorithm	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Results on Fashiongen					
OSCAR*	42.16	24.01	46.01	56.00	52.03
Multimodal GPT-2	48.87	30.33	52.44	62.41	58.72
Results on the industrial dataset#1					
OSCAR*	39.26	22.71	37.84	49.60	38.59
Multimodal GPT-2	41.14	24.35	39.97	50.66	39.39

Table 7.3: Results of *OSCAR** and *Multimodal GPT-2* on the Fashiongen dataset and the industrial dataset#1.⁵

The two architectures achieve a good level of performance in all the metrics, with *Multimodal GPT-2* that overcome *OSCAR** in both datasets. According to the results, training a Transformer-based fashion image captioner using a generative objective allows reaching higher performance in describing clothing products: such captions, as we described in chapter 4, are long descriptions of the input images and are usually more complex than the ones available in the general Image Captioning task, making the use of the GPT-2 decoder and its generative learning objective more effective than using a pre-trained BERT encoder.

Comparing the results achieved by *Multimodal GPT-2* in the two scenarios, where in the first one it generates captions using only images (therefore named *Image GPT-2*) while in the second one it also uses the metadata associated with clothing products, having a multi-modal input leads to better performance. This achievement is deeply analyzed in [PSC20]: leveraging the metadata of products along with their images improves the generation capability of the image captioner; in our experiments, this improvement is more significant for the *industrial dataset#1* than the *Fashiongen dataset*, as the latter contains less detailed and shorter metadata than the former.

Notice that in the experiments described above, we do not perform an extensive hyperparameter search: it is out of the scope of our research work, and the results we collect through these experiments are used to determine the most promising architecture for Fashion Image Captioning, which according to our work turns out to be *Multimodal GPT-2*. We use this model to perform a deep transfer learning analysis following the approach described in chapter 5.

7.4. Transfer Learning analysis

The image captioning architectures we have compared in the previous section are made up of deep learning technologies that, as discussed in chapter 2.1.1, can effectively process visual and textual inputs, achieving high-level performance at the cost of a long learning time and a large amount of data. Those models rely on statistical properties that cause a significant drop in the model performance when the distribution of the input data changes. Moved by this behavior, our work focuses on exploiting *transfer learning* to build fashion image captioning models that still achieve competitive performance when employed with clothing products collected from a different distribution from the one used for training.

In the following sections, first, we define a *source* and a *target* dataset. The models trained on only one of these two datasets determine the upper and lower bounds of the performance of the model under analysis; then, we follow the pre-training approach presented in chapter 5 to bridge the gap to “target-only” models, which are the models that are extensively trained and then tested on clothing samples that come from the same data distribution.

Reference models

The first step of our analysis is the selection of a *source* and a *target* dataset. In our transfer learning scenario, we define as *target-only* model the model with the best achievable performance on the target dataset; therefore, it is obtained when train and test datasets lean on the same data distribution, i.e., belong to the same dataset. On the other hand, the *source-only* model represents the simple condition in which an already available captioning system, trained on the *source* dataset, is used with clothing samples of a different dataset without adopting any pre-training or adaptation technique that eases the transfer of learned representations.

Data In the following experiments, among the data sources presented in chapter 4, we use *Fashiongen*, the *industrial dataset#1*, and *industrial dataset#2*. Depending on whether we consider the *source-only* or the *target-only* model, we compute the final performance on *Fashiongen* and the *industrial dataset#2* when trained using the train partition of the very same source or the one of the *industrial dataset#1*.

Target-only experiments We train *Multimodal GPT-2* using the train partition of *Fashiongen* and the *industrial dataset#2* and assess its performance on their test one, respectively. Table 7.4 shows the results.

Algorithm	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Results on Fashiongen					
Multimodal GPT-2	48.87	30.33	52.44	62.41	58.72
Results on the industrial dataset#2					
Multimodal GPT-2	49.41	31.72	53.50	61.56	56.28

Table 7.4: Results of *Multimodal GPT-2* on the Fashiongen dataset and the industrial dataset#2. These models are trained and tested on the respective *target* source.

The time and data requirements to achieve these results are provided in table 7.5.

Target Dataset	#Train Samples	GPU Time
Fashiongen	54132	12169 sec (1.55 days)
Industrial dataset#2	39282	8456 sec (1.08 days)

Table 7.5: Time and data requirements to train the *target-only* models.

Source-only experiments Here we train *Multimodal GPT-2* using the train partition of the *industrial dataset#1* and then assess its performance using the test partitions of *Fashiongen* and the *industrial dataset#2*. The results are available in table 7.6. No adaptation or pre-training techniques are applied in this experiment: we use the model as it is, after training on the *industrial dataset#1*.

Model	BLEU		GLEU		METEOR		ROUGE		ROUGE-L	
Results on Fashiongen										
<i>source-only</i>	3.18	-93.50%	0.83	-97.26%	3.15	-93.99%	5.46	-91.26%	4.64	-92.10%
<i>target-only</i>	48.87		30.33		52.44		62.41		58.72	
Results on the industrial dataset#2										
<i>source-only</i>	8.62	-82.55%	2.40	-92.43%	10.74	-79.92%	12.91	-79.02%	9.64	-82.87%
<i>target-only</i>	49.41		31.72		53.50		61.56		56.28	

Table 7.6: Results of *Multimodal GPT-2* on the Fashiongen dataset and the industrial dataset#2 when trained on the industrial dataset#1. These models are the *source-only* references when considering as *target* source Fashiongen or the industrial dataset#2. Together with the scores, we report also the drop in the performance.

Table 7.6 reports the drop in the performance between the *source-only* and *target-only* models. The gap between the two defines the region in which we operate through our pre-training and adaptation techniques to reduce the drop.

The *target-only* model clearly determines the correct product category and high-level details of the clothing sample in input, like the color of the garment and the target gender of the clothes; sometimes, it misses fine-grained features and generates a final product description that contains similar but wrong details compared to corresponding the ground truth. For example, it might mix up the type of fastening of a garment without distinguishing press-studs, zips, or buttons.

As expected, the *source-only* model struggles even in determining details of clothing products that seem easy to catch: changing the distribution of the input samples makes the model unable to distinguish the main features of the garment like its category or the target gender. This wrong behavior is due to the less relevant details in the input sample that the model learns as critical for the final description of the garment. Examples could be the background, having or not having a person wearing the clothing product, and even the particular type of shot of the input image (like the pose of the fashion model).

Figure 7.3 provides examples of captions generated by the *source-only* and the *target-only* models, compared with the corresponding ground truth.



Metadata: clothing, women, pants, trousers.

Ground truth: Pillar-leg trousers in black. Accordion pleats throughout. Six-pocket styling with removable patch pocket at legs with press-stud fixtures. Tonal stitching. Button fly.

Source-only: The excellence of materials enriches the essential and versatile style of the new Crispy silk **shirt**. Manufacturing of the refined transparent and slightly shimmering fabric characterizes the garment with a sporty touch, while the edges of the sleeves showcase a precious artisanal embellishment. The fit is comfortable with slightly dropped shoulders.

Target-only: Wide-leg pleated culottes in black. **Four-pocket** styling. **Zip-fly**. Tonal stitching.



Metadata: clothing, t-shirts & vests, polo, men, tshirts, blue.

Ground truth: Short-sleeved polo shirt in cotton piqué with a printed pattern. Collar button placket and short slits at sides.

Source-only: The excellence of materials enriches the casual style of these new **shorts** from the Travelwear line, dedicated to moments of relaxation and free time. Lightweight techno cotton French terry is combined with contrast color elasticated edges on the sleeves and bottom. The fit is comfortable both in the seat and along the leg.

Target-only: Short-sleeved polo shirt in cotton jersey with a collar button placket and short slits at sides.

Figure 7.3: Comparison between the captions generated by the *target-only* and the *source-only* models with the corresponding ground truth: the captions generated by the *target-only* models match the clothing categories of the related ground truths; the *source-only* models, instead, predict a **wrong** clothing category. Finer details that the target models **mispredict** are highlighted. The upper clothing product belongs to *Fashiongen* while the lower one to the *industrial dataset#2*.

Pre-Training

The first way to tackle the gap in performance between the *source-only* and *target-only* models is by considering a pre-training procedure that learns a more general representation of the input samples. We showed that using an already available fashion image captioner on clothing samples that belongs to a new input distribution is not feasible because the model learns non-relevant patterns in the input samples of the training set that prevent the model generalization capabilities. We use the pre-training procedure defined in chapter 5 to make the model learn a more general representation of the input, improving its performance on unseen data distributions. The main aspects of this procedure are the simultaneous learning from multiple sources through a stratified batch sampler along with a weighted loss function and the generation of noisy samples. Chapter 5 provides all the implementation details.

Data We assess the performance of our pre-training method on the test partition of the *Fashiongen* dataset and the *industrial dataset#2*. The model learns from multiple sources but is tested on a new distribution, so, according to the final *target* dataset, we leverage all the datasets described in chapter 4 as stated in table 7.7.

Pre-Train	Target
Industrial dataset#1	Fashiongen
Industrial dataset#2	
Industrial dataset#3	
Industrial dataset#1	Industrial dataset#2
Fashiongen	
Industrial dataset#3	

Table 7.7: Assignment of datasets among pre-training and test sources: the train partition of the dataset used to assess the model performance is not available during the pre-training stage.

We add 5% of noise according to the procedure described in chapter 5: before training, we use Algorithm 5.1 to generate a taxonomy for each train source; they are used to define the rules that allow the generation of noisy samples following the steps discussed in section 5.4.2.

Results Table 7.8 compares the results achieved by the pre-trained model with the *source-only* and the *target-only* models, using the available data sources as described above. A model trained according to our pre-training approach achieves higher scores

than the *source-only* model and reduces the gap in performance between the two reference models. Models of both datasets benefit from the pre-training procedure, even though the model tested on *Fashiongen* registered the highest increment.

Model	BLEU		GLEU		METEOR		ROUGE		ROUGE-L						
Results on Fashiongen															
<i>source-only</i>	3.18	-93.50%	0.83	-97.26%	3.15	-93.99%	5.46	-91.26%	4.64	-92.10%					
<i>pre-training</i>	10.59	-78.33%	↑15.17	3.89	-87.18%	↑10.08	13.59	-74.09%	↑19.89	23.52	-62.32%	↑28.94	17.95	-69.44%	↑22.66
<i>target-only</i>	48.87		30.33		52.44		62.41		58.72						
Results on the industrial dataset#2															
<i>source-only</i>	8.62	-82.55%	2.40	-92.43%	10.74	-79.92%	12.91	-79.02%	9.64	-82.87%					
<i>pre-training</i>	10.85	-78.03%	↑4.52	3.68	-88.40%	↑4.04	12.94	-75.81%	↑4.10	18.77	-69.51%	↑9.51	14.48	-74.27%	↑8.60
<i>target-only</i>	49.41		31.72		53.50		61.56		56.28						

Table 7.8: Performance of our pre-training methodology applied to *Multimodal GPT-2* when tested on the Fashiongen dataset and the industrial dataset#2. Together with the scores and the drop in the performance, the table shows the improvement compared to the *source-only* model.

As reported in figure 7.4, the generalization capability of the fashion captioner increases: the model is now able to recognize the correct clothing category of the input product (*source-only* models identify “trousers” for “jackets” and a “polo shirt” for “shorts”); though, it can struggle with close colors or finer details like the type of collar (a “polo” collar is described as a “crewneck”).

Besides catching the correct details in the input, an important aspect is the structure of the generated caption: each data source adopt a singular style in describing products, which is the result of brand-related design choices such as the type of narrative, or which kind of information should or should not be in a textual description. For example, the *Fashiongen* dataset requires that the final captions include information about the color of the garment and are characterized by multiple short sentences; otherwise, the ground truth captions of the *industrial dataset#2* are single-sentence and do not include information about the color. A fashion image captioner does not learn these stylistic choices when trained only with sources that adopt a structure of captions different from the target one, and metrics that rely only on n -gram matching can get low-level scores even with semantically correct output captions.

A captioning model can learn source-dependent properties, like the ones identified above, through an adaptation paradigm like *fine-tuning*, which is the approach we chase in the following experiment.

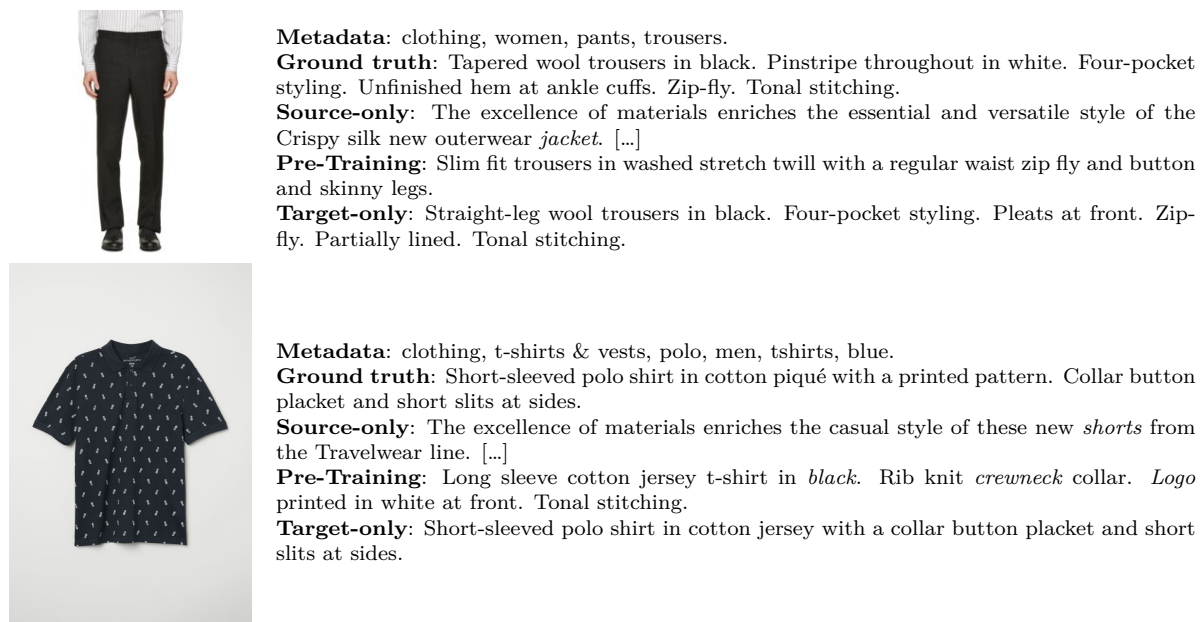


Figure 7.4: Comparison between the captions generated by the pre-trained model with the ones provided by the model references and the corresponding ground truth: the quality improves since the main features of the clothes are predicted correctly, but still there could be *mispredicted* words in each caption.

Fine-tuning

Leveraging previously learned representations of deep learning models without adapting them to the new scenario does not allow the model to learn dataset-dependent properties. As discussed in the previous experiment, the structure of captions to predict is a determining factor for the quality of a fashion image captioner, and models that tackle this challenge can provide such structures of output sentences only by performing additional learning steps with clothing samples belonging to the target distribution.

In this experiment, we fine-tune the pre-trained model on the *target* dataset by using a limited amount of training samples for a fixed number of epochs. More in detail, the architecture is initialized using the weights of the pre-trained model, and the training procedure does not end according to an early stopping callback but it lasts five epochs only. In this way, besides producing a semantically correct description, the output sentences of the fashion image captioner follow the stylistic decisions of the target data source.

Data We use the two pre-trained models of the previous section and fine-tune them on the *Fashiongen* dataset and the *industrial dataset#2*, respectively. The number of training samples used in this experiment is limited and fixed to 5% of the train partition.

Table 7.9 provides a detailed recap of the training datasets used in each stage of our analysis according to the *target* dataset.

		Target dataset		
		<i>Fashiongen</i>	<i>Industrial dataset#2</i>	
Train datasets	<i>target-only</i>	Fashiongen	Industrial dataset#2	
	<i>source-only</i>	Industrial dataset#1	Industrial dataset#1	
	<i>pre-training</i>	Industrial dataset#1	Fashiongen	
		Industrial dataset#2	Industrial dataset#1	
		Industrial dataset#3	Industrial dataset#3	
<i>fine-tuning</i>	Fashiongen (5%)	Industrial dataset#2 (5%)		

Table 7.9: Recap of the training datasets used in each stage of our analysis according to the *target* dataset.

Results Table 7.10 provides the scores achieved by the fine-tuned models along with the performance of the pre-trained ones and the two references. As expected, fine-tuning allows covering a large portion of the gap in the performance between the *source-only* and *target-only* models.

Model	BLEU		GLEU		METEOR		ROUGE		ROUGE-L						
Results on Fashiongen															
<i>source-only</i>	3.18	-93.50%	0.83	-97.26%	3.15	-93.99%	5.46	-91.26%	4.64	-92.10%					
<i>pre-training</i>	10.59	-78.33%	↑15.17	3.89	-87.18%	↑10.08	13.59	-74.09%	↑19.89	23.52	-62.32%	↑28.94	17.95	-69.44%	↑22.66
<i>fine-tuning</i>	31.28	-35.99%	↑57.50	19.60	-35.37%	↑61.89	36.62	-30.16%	↑63.83	50.06	-19.80%	↑71.46	47.80	-18.60%	↑73.50
<i>target-only</i>	48.87		30.33		52.44		62.41		58.72						
Results on the industrial dataset#2															
<i>source-only</i>	8.62	-82.55%	2.40	-92.43%	10.74	-79.92%	12.91	-79.02%	9.64	-82.87%					
<i>pre-training</i>	10.85	-78.03%	↑4.52	3.68	-88.40%	↑4.04	12.94	-75.81%	↑4.10	18.77	-69.51%	↑9.51	14.48	-74.27%	↑8.60
<i>fine-tuning</i>	28.17	-42.97%	↑39.58	14.67	-53.76%	↑38.67	30.14	-43.66%	↑36.25	41.36	-32.82%	↑46.21	36.81	-34.60%	↑48.26
<i>target-only</i>	49.41		31.72		53.50		61.56		56.28						

Table 7.10: Comparison of the performance achieved in each step of our transfer learning methodology applied to *Multimodal GPT-2* when tested on Fashiongen and the industrial dataset#2. The last fine-tuning step allows covering a large portion of the gap in performance between *source-only* and *target-only* models.

Notice that, as stated in table 7.11, these levels of performance are achieved through a highly constrained training procedure characterized by a small number of training samples (5%) and training epochs (5), which reflects on a drastic reduction of the time requirements with respect to the one of the *target-only* model provided in table 7.5 (3.5% of the total time to train the target is now required). These results underline the benefits of leveraging a pre-trained architecture when dealing with constraints such as costly or limited training samples and computing resources with bounded training time or memory.

Target Dataset	#Train Samples	GPU Time
Fashiongen	2706	4275 sec (1.18 h)
Industrial dataset#2	1964	3556 sec (0.98 h)

Table 7.11: Time and data requirements to fine-tune the *pre-trained* models.

Through this adaptation step, the model learns the information to include in the final caption and the particular type of narrative to use in describing the garment in input. Figures 7.5 provide complete examples taken from *Fashiongen* and the *industrial dataset#2*.

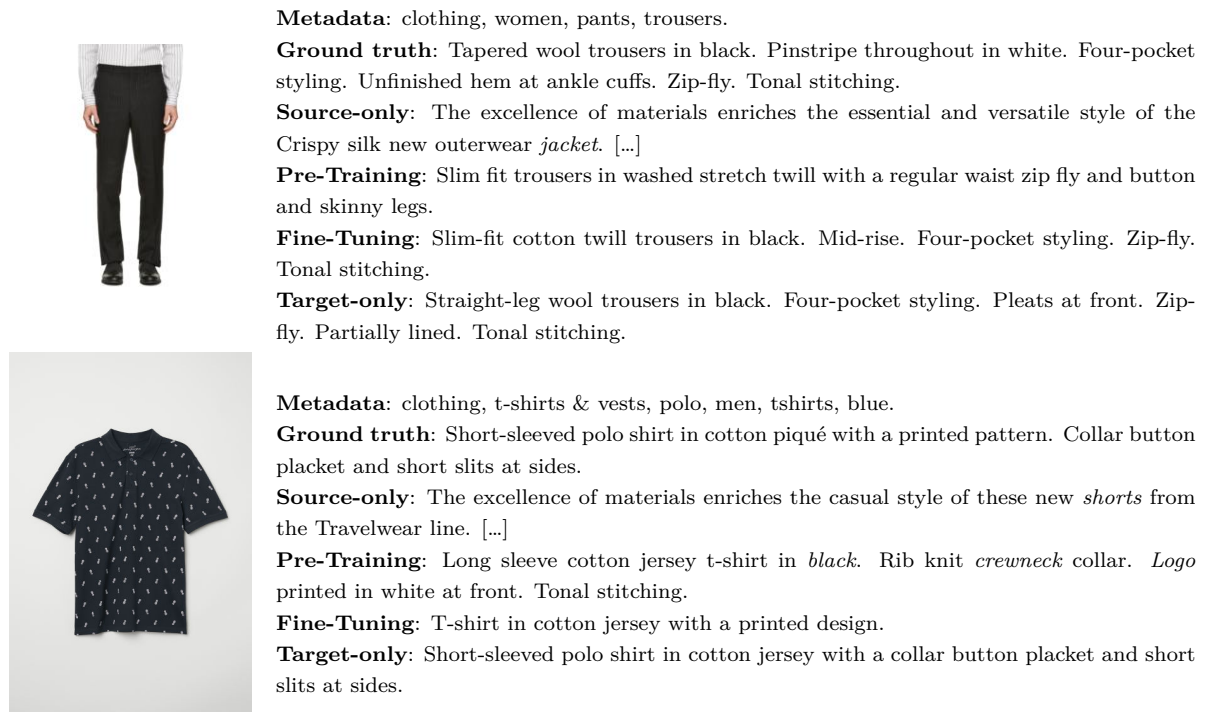


Figure 7.5: Comparison between the captions generated by the models involved in each step of our transfer learning analysis: the adaptation through fine-tuning improves the quality of the generated captions by learning source-dependent properties that were missed by leveraging only the pre-training stage.

The captions generated by the fine-tuned models follow the structure of the ground truth: the model fine-tuned on *Fashiongen* predicts an output caption that, differently from the pre-trained model, includes correct information about the color and consists of multiple short sentences as more appropriate to describe clothing products belonging to this data source.

7.5. Vision-Text Multi-Encoder Decoder

In this section, we analyze the *Vision-Text Multi-Encoder Decoder* architecture presented in chapter 6 applied to the task of Fashion Image Captioning. We discuss the training options that arise when considering a *contrastive* objective applied to our transfer learning analysis, comparing the performance of our architecture with *Multimodal GPT-2*.

In the following experiments, each encoder/decoder stack of the *Vision-Text Multi-Encoder Decoder* architecture is initialized using the weights of a pre-trained model available in the Hugging Face Transformers⁶ library. The vision encoder is a ViT model pre-trained using the BEiT masked image modeling task [BDW21], the text encoder is a pre-trained uncased BERT, and the decoder uses the weights of the pre-trained GPT-2 model.

Transfer Learning analysis

The training methodology presented in chapter 5 does not depend on the particular choice of deep learning technologies that build the overall image captioning model. So, moved by the same goal of the analysis carried out in section 5, we investigate the transfer learning capabilities of the *Vision-Text Multi-Encoder Decoder* architecture through the same learning procedure but adding a contrastive learning objective.

As done for the experiments in section 7.4, the first step of our analysis is the definition of the *source-only* and the *target-only* models that define a range of performance that the pre-training method aims to thin. Then, we leverage our pre-training procedure exploiting several data sources to learn a general representation according to multiple data distributions; finally, the fine-tuning stage allows the adaptation of the pre-trained architecture to the *target* dataset, making the model able to match the stylistic properties defined by the target source.

Data The setup of the experiments and the use of data sources are the same as the ones in section 7.4. We assess the performance of *Vision-Text Multi-Encoder Decoder* using the test partitions of the *Fashiongen* dataset and the *industrial dataset#2*; the train data sources vary according to the learning stage or if the model is one of the two references. The *target-only* model is trained using the samples of the train partition belonging to the same dataset used for testing, while the train dataset of the *source-only* reference is the *industrial dataset#1*. The pre-training procedure leverages all the datasets presented in chapter 4 but the one used to test the final performance. We add 5% of noise to

⁶<https://github.com/huggingface/transformers>

the training samples using the noise-generation procedure in chapter 5.4.2. In the next section, we discuss all the details of the pre-training procedure applied to the *Vision-Text Multi-Encoder Decoder* architecture. In the final fine-tuning stage, the number of training samples is limited to 5% of the train partition of the target dataset.

Results Table 7.12 provides the results of the pre-training and the fine-tuning stages compared with the scores achieved by the two references.

Model	BLEU		GLEU		METEOR		ROUGE		ROUGE-L						
Results of <i>Vision-Text Multi-Encoder Decoder</i> on Fashiongen															
<i>source-only</i>	2.00	-96.08%	0.56	-98.24%	1.84	-96.65%	3.00	-95.37%	2.62	-95.69%					
<i>pre-training</i>	12.59	-74.24%	↑19.25	4.08	-86.54%	↑10.71	13.97	-73.35%	↑20.63	23.31	-62.66%	↑28.60	17.31	-70.53%	↑21.57
<i>fine-tuning</i>	39.76	-18.64%	↑74.86	22.74	-25.03%	↑72.23	43.03	-17.95%	↑76.03	54.88	-12.08%	↑79.18	51.13	-12.94%	↑79.16
<i>target-only</i>	51.08		32.14		54.90		64.78		60.71						
Results of <i>Vision-Text Multi-Encoder Decoder</i> on the industrial dataset#2															
<i>source-only</i>	6.54	-87.39%	1.95	-94.23%	6.19	-89.17%	9.32	-85.53%	7.45	-87.39%					
Pre-Training	14.19	-71.28%	↑11.27	4.95	-84.40%	↑8.03	14.60	-72.71%	↑7.21	24.86	-59.62%	↑19.40	18.82	-66.57%	↑16.30
Fine tuning	34.91	-29.34%	↑53.22	18.50	-41.69%	↑50.74	38.40	-28.23%	↑51.69	47.70	-22.51%	↑56.51	42.72	-24.11%	↑58.76
<i>target</i>	51.85		33.74		57.18		64.42		59.12						

Table 7.12: Comparison of the performance achieved in each step of our transfer learning methodology applied to *Vision-Text Multi-Encoder Decoder* when tested on Fashiongen and the industrial dataset#2.

Even for *Vision-Text Multi-Encoder Decoder*, the *source-only* models struggles in determining all the most relevant features of the clothing product in input; the *target-only* models instead generate high-quality descriptions of products and identify all the details of the input garment. Figure 7.6 shows examples of captions generated by all the models and the references involved in our training approach.

The encoding side of *Vision-Text Multi-Encoder Decoder* consists of two separate transformer stacks, whose outputs are handled at the decoding side through a double cross-attention layer. Even when exploiting the decoder weights of the GPT-2 architecture, those cross attention layers are newly initialized because they were not required when pre-training GPT-2 over the 40GB of text available as done in [Rad+19]. Newly initialized cross-attentions layers affect the performance of the *source-only* models: in the experiments involving the *source-only* models, those layers are trained only over the input samples available in the *industrial dataset#1*, making the models sensitive to changes in the input samples, causing the generation of low-quality output captions. By learning over multiple distributions of clothing samples, the pre-training stage overcomes this limitation and allows the model to generate fluent descriptions that predict the relevant information

of the input garment but may miss details or generate misleading information. The caption of the *coat* provided in figure 7.6 mentions a “concealed zip” that is not the correct type of closure of the garment depicted in the corresponding input image; the lower clothing sample concerns a *t-shirt* which the pre-trained model describes as *black* while instead has a *printed pattern*, and “rib knit” details or “button fastening” should not be present in the output description. The pre-training stage of the *Vision-Text Multi-Encoder Decoder* includes a contrastive objective and allows multiple learning options that we analyze in the next section.

Fine-tuning the pre-trained *Vision-Text Multi-Encoder Decoder* architectures allows reaching high-level performance: even though this final stage considers a small number of samples and a fixed number of epochs, the final captions are high-level quality descriptions of the clothing samples in input.



Metadata: clothing, women, jackets & coats, coats.

Ground truth: Long sleeve wool coat in 'dark night' navy. Notched lapel collar. Double-breasted woven faux-leather button closure at front. Welt pockets at waist. Padded shoulders. Central vent at back hem. Fully lined. Tonal stitching.

Source-only: The natural, natural and versatile inspiration of the elegant world define the style of this new Pino *scarf*. Cable yarn elegant, [...]

Pre-Training: Long coat with a collar and *concealed zip* at front. Side pockets in matching fabric-seam side seams. Lined.

Fine-Tuning: Long sleeve wool coat in black. Notched lapel collar. Button closure at front. Flap pockets at waist. Single-button barrel cuffs. Tonal stitching.

Target-only: Long sleeve wool-blend coat in black. Notched lapel collar. Button closure at front. Flap pockets at waist. Central vent at back hem. Fully lined. Tonal stitching.



Metadata: clothing, tops, short sleeves, ladies, black, adult, basics.

Ground truth: T-shirt in lightweight cotton jersey with a rounded hem. Slightly longer at the back.

Source-only: Refined and shiny effect of the new, the new Feather Silver [...]

Pre-Training: Short sleeve cotton jersey t-shirt in *black*. *Rib knit* crewneck collar and cuffs striped at front. Tonal stitching featuring signature trims throughout with *button fastening*, note zippered pocket detail on the back of neck.

Fine-Tuning: T-shirt in soft viscose jersey with a slightly wider neckline and short sleeves. Rounded hem for best fit over the body.

Target-only: T-shirt in lightweight cotton jersey with a rounded hem. Slightly longer at back.

Figure 7.6: Examples of captions generated by the models involved in each learning stage, compared with the ground truth and the *source-only* and *target-only* references: the *target* datasets are Fashiongen (up) and the industrial dataset#2 (down).

Pre-training options

The pre-training procedure defined in chapter 5 allows the model to learn a more general representation of the input data by leveraging the simultaneous learning from multiple sources through a stratified batch sampler along with a weighted loss function, and the generation of noisy samples. This pre-training procedure aims to improve the performance of a fashion image captioner on unseen data distributions.

As described in chapter 6, The *Vision-Text Multi-Encoder Decoder* architecture introduces the possibility to consider an additional contrastive learning objective in the pre-training procedure: the underlying idea is that images and metadata are complementary ways to describe a garment, and ideally, their embeddings should be close in a shared representation space, regardless of the input modality. We compute a *contrastive loss* between image and metadata embeddings using algorithm 6.1, as it is done in [Rad+21].

In the pre-training stage applied to *Vision-Text Multi-Encoder Decoder*, we analyze three possible ways of leveraging the contrastive alignment:

- a *multi-objective optimization*, in which the model updates its weights simultaneously according to the language modeling objective and the contrastive loss;
- a *pre-alignment* step that performs the contrastive alignment between metadata and image embeddings to condition the embeddings before the language modeling task;
- a combination of the two options above.

The multi-objective optimization and pre-alignment techniques are described in section 6.2.1 and 6.2.2, respectively. Figure 7.7 summarizes the two approaches. In our experiments, we set the hyperparameter $\lambda = 0.3$.

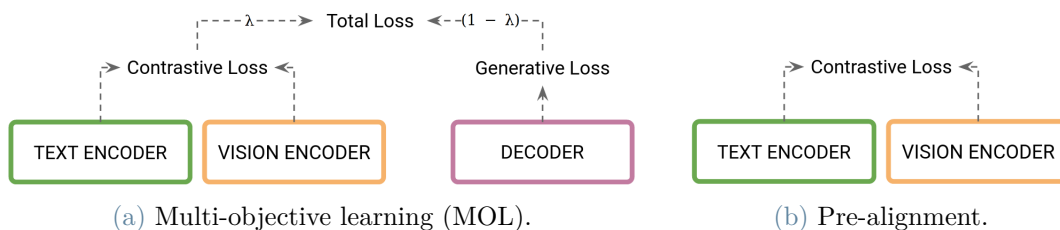


Figure 7.7: Visual representation of the two techniques we use to perform the contrastive alignment between the image and metadata embeddings.

Data The three scenarios defined above are tested on *Fashiongen* and the *industrial dataset#2*. We use the data sources in the same way as in the pre-training stage of *Multimodal GPT-2* in section 7.4: *Vision-Text Multi-Encoder Decoder* is pre-trained using

all the data sources but the one for testing. So, the model tested on *Fashiongen* uses the *industrial datasets#1*, *#2*, and *#3* as train sources, while the model tested on the *industrial datasets#2* uses *Fashiongen* together with the *industrial datasets#1* and *#3* as train sources. We add 5% of noise to the train samples using the procedure in section 5.4.2.

Notice that adding the *pre-alignment* as an additional step of the pre-training stage requires splitting the train partitions to avoid overfitting (figure 7.8).

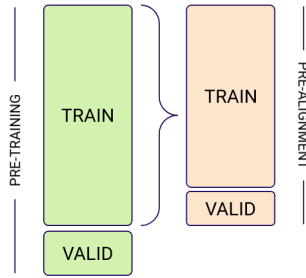


Figure 7.8: Train and validation partitions according to the pre-training step.

Results Table 7.13 provides the results of the three approaches identified above tested on *Fashiongen* and the *industrial dataset#2*.

Approach	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Pre-training stage on Fashiongen					
MOL	12.59	4.08	13.97	23.31	17.31
Pre-alignment	10.21	3.40	13.11	21.31	15.75
MOL + Pre-alignment	10.95	3.71	14.27	21.79	16.08
Pre-training stage on the industrial dataset#2					
MOL	14.06	4.53	16.54	23.06	16.53
Pre-alignment	13.94	4.66	15.84	23.15	17.15
MOL + Pre-alignment	14.19	4.95	14.60	24.86	18.82

Table 7.13: Comparison of the results of the three pre-training approaches when tested on *Fashiongen* and the *industrial dataset#2*.

On both datasets, the pre-alignment of image and metadata embeddings alone is less effective than the single pre-training stage that leverages the multi-objective learning (MOL) task. Combining the two approaches does not guarantee an improvement of the performance: having an additional pre-alignment step before the MOL one is beneficial on the *industrial dataset#2* while it worsens the scores of four metrics out of five on the *Fashiongen* dataset. These opposite behaviors may be due to the differences in the metadata of

the target datasets: metadata of clothing products belonging to the *Fashiongen* datasets consist of a few general pieces of information regarding the garment in input; *industrial dataset#2*, instead, uses a higher number of tags and classes, catching more details of the clothing samples. The conditioning of the embeddings through the pre-alignment step before the MOL pre-training improves the performance on datasets that make extensive use of additional metadata related to clothing samples. This consideration is further confirmed by the comparison carried out in the next section: the increment in performance provided by the *Vision-Text Multi-Encoder Decoder* pre-training stage with respect to *Multimodal GPT-2* is higher for *industrial dataset#2* compared to *Fashiongen*. The more a dataset leverages the metadata related to garments, the more the contrastive alignment of the embeddings is effective.

Notice that we analyze three potential settings, but in principle, there could be others according to the order of the pre-training steps. Besides, a deep optimization of the hyperparameter λ could provide additional insights.

Comparison with *Multimodal GPT-2*

In this section, we analyze the performance of *Vision-Text Multi-Encoder Decoder* compared to *Multimodal GPT-2*. The comparison is performed for each stage of our transfer learning analysis, including the *source-only* and *target-only* models.

Table 7.14 provides the results on *Fashiongen* and *the industrial dataset#2*. For each dataset and learning stage of the analysis conducted in sections 7.4 and 7.5, we consider the performance achieved by the *Vision-Text Multi-Encoder Decoder* (*Vision-Text MED* in table) and *Multimodal GPT-2*.

Model	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Results on Fashiongen					
<i>source-only</i>					
Multimodal GPT-2	3.18	0.83	3.15	5.46	4.64
Vision-Text MED	2.00	0.56	1.84	3.00	2.62
<i>pre-training</i>					
Multimodal GPT-2	10.59	3.89	13.59	23.52	17.95
Vision-Text MED	12.59	4.08	13.97	23.31	17.31
<i>fine-tuning</i>					
Multimodal GPT-2	31.28	19.60	36.62	50.06	47.80
Vision-Text MED	39.76	22.74	43.03	54.88	51.13
<i>target-only</i>					
Multimodal GPT-2	48.87	30.33	52.44	62.41	58.72
Vision-Text MED	51.08	32.14	54.90	64.78	60.71
Results on the industrial dataset#2					
<i>source-only</i>					
Multimodal GPT-2	8.62	2.40	10.74	12.91	9.64
Vision-Text MED	6.54	1.95	6.19	9.32	7.45
<i>pre-training</i>					
Multimodal GPT-2	10.85	3.68	12.94	18.77	14.48
Vision-Text MED	14.19	4.95	14.60	24.86	18.82
<i>fine-tuning</i>					
Multimodal GPT-2	28.17	14.67	30.14	41.36	36.81
Vision-Text MED	34.91	18.50	38.40	47.70	42.72
<i>target-only</i>					
Multimodal GPT-2	49.41	31.72	53.50	61.56	56.28
Vision-Text MED	51.85	33.74	57.18	64.42	59.12

Table 7.14: Comparison of the performance achieved in each step of our transfer learning methodology by *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* when tested on Fashiongen and the industrial dataset#2.

Our architecture improves the performance of *Multimodal GPT-2* in all the stages, except the *source-only*: using an additional encoder stack that processes the metadata related to the clothing samples becomes effective when the overall architecture is pre-trained over multiple sources. In that case, the fine-tuned *Vision-Text Multi-Encoder Decoder* outperforms *Multimodal GPT-2* by a significant margin. As discussed in the previous section, the relative improvement in the pre-training stage is different according to the target data source: the improvement is more significant for the *industrial dataset#2* than the *Fashiongen* dataset, as the latter contains less detailed and shorter metadata than the former. Figure 7.9 shows examples of captions generated by the two fine-tuned

architecture.



Metadata: clothing, women, jackets & coats, coats.

Ground truth: Long sleeve wool coat in 'dark night' navy. Notched lapel collar. Double-breasted woven faux-leather button closure at front. Welt pockets at waist. Padded shoulders. Central vent at back hem. Fully lined. Tonal stitching.

Multimodal GPT-2: Long sleeve French terry coat in black. *Spread* collar. Button closure at front. Welt pockets at waist. *Two-way zip closure at back*. Fully lined. Tonal stitching.

Vision-Text Multi-Encoder Decoder: Long sleeve wool coat in black. Notched lapel collar. Button closure at front. *Flap* pockets at waist. *Single-button barrel cuffs*. Tonal stitching.



Metadata: clothing, tops, short sleeves, ladies, black, adult, basics.

Ground truth: T-shirt in lightweight cotton jersey with a rounded hem. Slightly longer at the back.

Multimodal GPT-2: Short-sleeved top in soft jersey with a *V-neck*.

Vision-Text Multi-Encoder Decoder: T-shirt in soft viscose jersey with a slightly wider neckline and short sleeves. Rounded hem for best fit over the body.

Figure 7.9: Comparison between the captions generated by the fine-tuned *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* when the *target* dataset are Fashiongen (up) and the industrial dataset#2 (down).

The captions are similar since both recognize the main properties of the clothing sample in input; the two descriptions differ in small details of the garment.

Visualization of the contrastive alignment between embeddings

In this section, we analyze the behavior of image and metadata embeddings when leveraging the contrastive alignment of *Vision-Text Multi-Encoder Decoder*.

We train two models using the data of the *industrial dataset#2*: for the first one, we use only the language modeling objective without forcing the alignment between the two modalities; differently, we train the second one using the multi-objective approach defined in section 6.2.1 with the hyperparameter $\lambda = 0.3$. For both the models, we extract the image and metadata embeddings of the clothing samples of the test partition of the *industrial dataset#2*.

Regardless of the training objective, the image and metadata embeddings related to a clothing sample should be close in a shared representation space: we use the *t-distributed stochastic neighbor embedding* (t-SNE) dimensionality reduction technique to project the embeddings in a 2-dimensional space and visualize them. To highlight the clothing cat-

category of the embeddings, we leverage the reference taxonomy that we also exploit to generate noisy samples during the pre-training procedure. In this way, independently of the particular taxonomy of a data source, we are able to visualize how similar clothing products are projected in the representation space. Figure 7.10 shows the projection of the embeddings when the model either does or does not use the contrastive alignment between the two modalities. We use \cdot and $+$ to represent the embeddings of images and metadata, respectively.

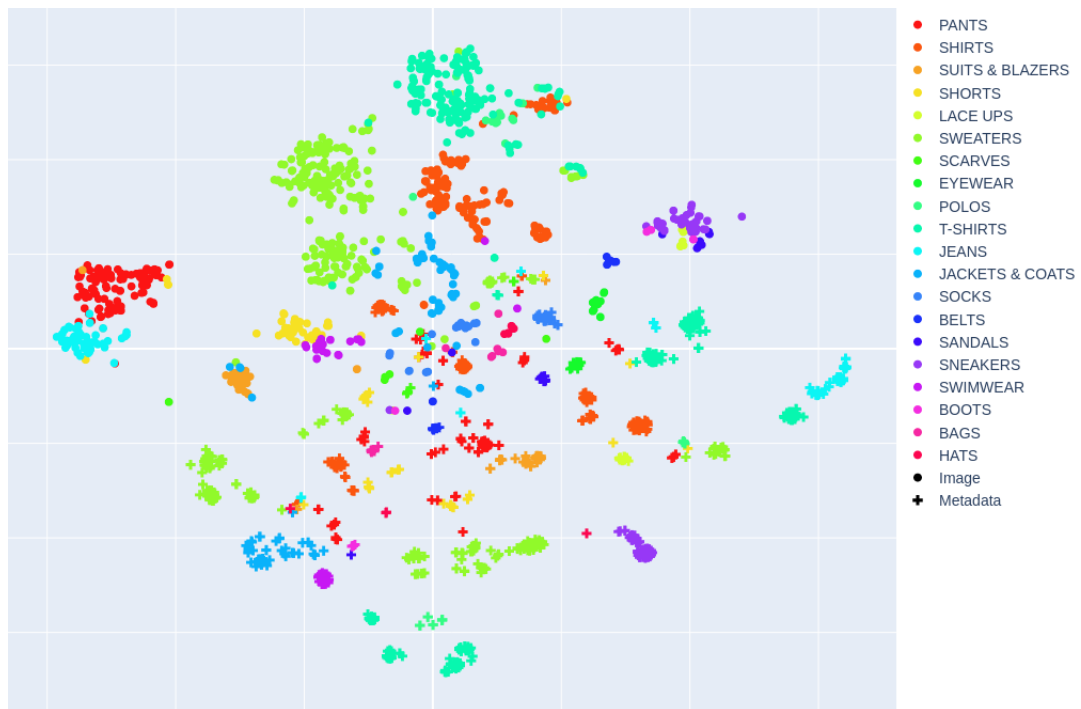
Using the contrastive objective, the embeddings of the two modalities are close; otherwise, the model differentiates between image and metadata embeddings even though the clothing category is the same. Besides, leveraging the contrastive objective helps the model in understanding the clothing category of embeddings of the same modality more clearly: the clusters of the image embeddings in figure 7.10a are less cohesive and separated than the ones in figure 7.10b, showing that the alignment between image and textual embeddings improves the ability of the model to differentiate among clothing categories.

Following this insight, we test if using a contrastive objective improves the performance not only of pre-trained models but also models trained and tested on the same data source (the *target-only* models). Table 7.15 provides the results of *Vision-Text Multi-Encoder Decoder* on *Fashiongen* and the *industrial dataset#2* compared with *Multimodal GPT-2*, in both the scenarios in which it is either used or not used the contrastive alignment through the multi-objective learning approach.

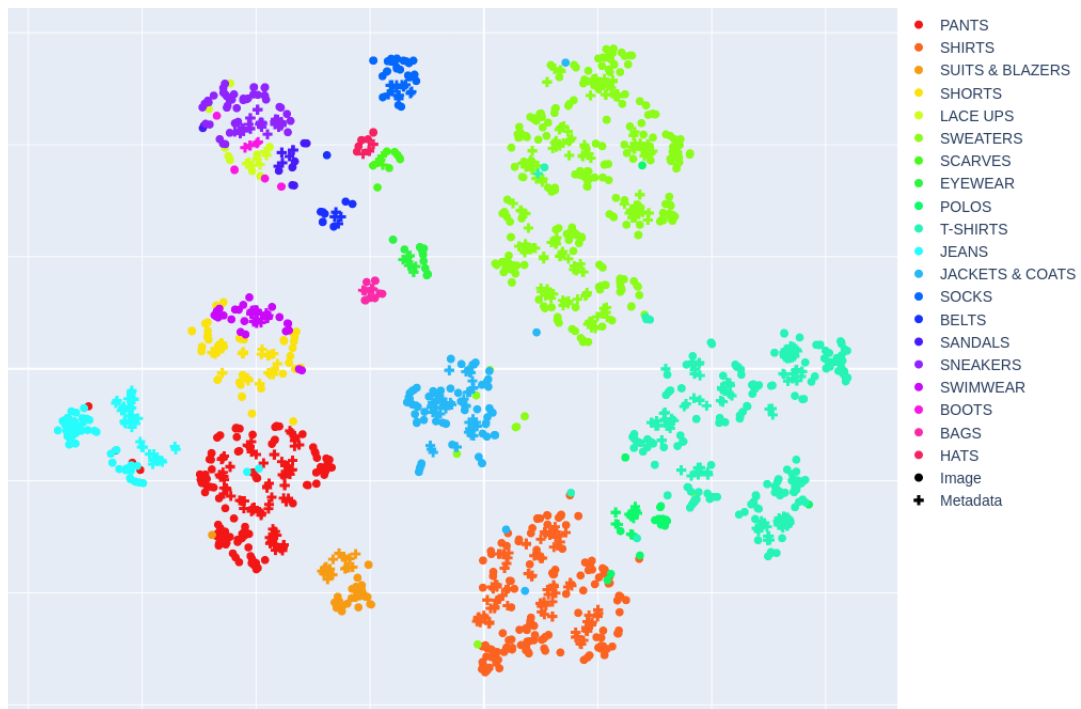
Model	BLEU	GLEU	METEOR	ROUGE	ROUGE-L
Results on Fashiongen					
Multimodal GPT-2	48.87	30.33	52.44	62.41	58.72
Vision-Text MED	51.08	32.14	54.90	64.78	60.71
Vision-Text MED + MOL	51.13	32.63	55.06	65.42	61.47
Results on the industrial dataset#2					
Multimodal GPT-2	49.41	31.72	53.50	61.56	56.28
Vision-Text MED	51.85	33.74	57.18	64.42	59.12
Vision-Text MED + MOL	53.38	35.52	58.65	66.00	60.73

Table 7.15: Comparison of the results of *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* when trained either using used or not using the contrastive alignment through the multi-objective learning approach.

The data source that benefits the most of the contrastive alignment is the *industrial dataset#2*, but this additional objective in the overall loss function determines an improvement of the final performance on both datasets. To summarize the comparison



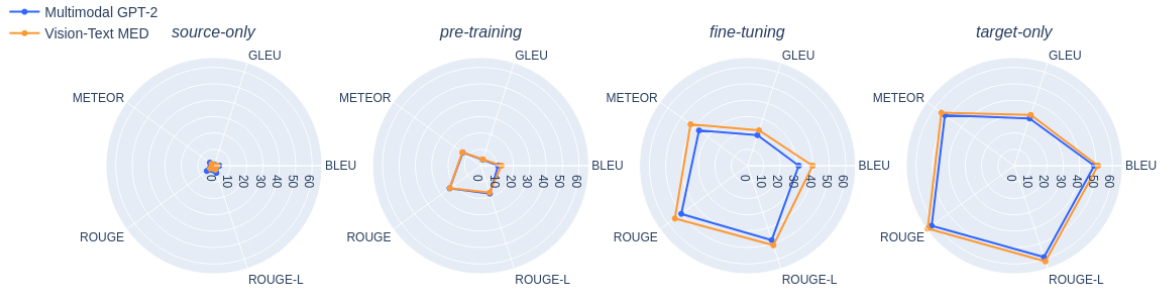
(a) Training without the contrastive alignment.



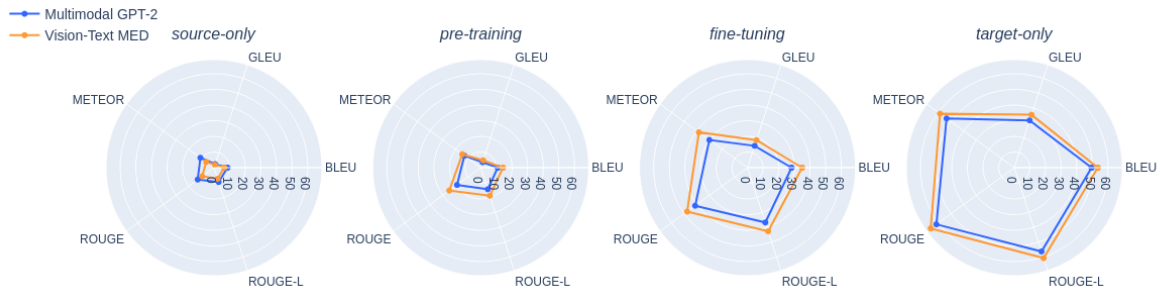
(b) Training with the contrastive alignment.

Figure 7.10: Visualization of image and metadata embeddings of clothing samples through t-SNE according to the training procedure of *Vision-Text Multi-Encoder Decoder*. It consists of the language modeling objective either alone (a) or with the additional contrastive alignment (b).

between *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder*, figure 7.11 shows the results achieved on *Fashiongen* and the *industrial dataset#2* in each stage of our transfer learning analysis.



(a) Results on Fashiongen.



(b) Results on the industrial dataset#2.

Figure 7.11: Results of *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* in each stage of our transfer learning analysis using as test data source Fashiongen (a) and the industrial dataset#2 (b).

7.6. User study

In the previous sections, we evaluate the performance of the fashion image captioners through automatic metrics designed for Natural Language Generation tasks. They are inexpensive, deterministic, thus repeatable, and quick to compute ways to approximate the quality of automatically generated text samples. As described in section 7.1, all the metrics rely on the computation of a mapping between the generated candidate and the ground truth reference: the most common way to achieve it is by considering n -gram overlaps, but there are also different approaches like the alignment computed by METEOR or the longest common subsequence of ROUGE-L. Their main drawback is that the words in both candidate and reference sentences are equally weighted, so missing out on content-bearing pieces of sentences instead of less significant ones is valued the same rather than being penalized more.

We design a survey to elicit judgments from users about the quality of descriptions of clothing products, following some of the guidelines and best practices discussed in [van+21]. The goal is to compare the captions generated by models trained through the methodology described in chapter 5 with the corresponding ground truths. The models we consider in the survey are *Multimodal GPT-2* and our *Vision-Text Multi-Encoder Decoder*, which are trained as analyzed in sections 7.4 and 7.5, respectively.

Evaluation Evaluating the quality of a text sample is not easy to assess, especially when dealing with the description of fashion items characterized by several details; thus, we identified three criteria that jointly determine the overall quality of the caption of a clothing product. First, a *precision* measure to evaluate the correctness of the information provided in the description; second, a *recall* oriented score stating the number of relevant details mentioned in the description concerning the garment; finally, we ask to give feedback on the *syntactic correctness* of the text sample under evaluation.

We provide the image and metadata related to a garment, and the users express their level of agreement or answer to the following sentences using 5-point Likert scales:

1. *The description contains only correct information about the garment in question.*
2. *The description contains all relevant information to describe the garment.*
3. *How do you evaluate the syntactic correctness of the sentence?*


Eventually, the user can provide textual hints to help us analyze the results.

The survey contains each clothing sample three times, differentiating the caption of the

fashion item among the ground truth and the texts generated by *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder*. Figure 7.12 shows an example of the user interface.

CONTENTWISE FILIPPO@MAIL.COM LOG OUT INFO

Question 7



Keywords: tops, basics, ladies, adult, clothing, black, short sleeves.

Description:
T-shirt in soft viscose jersey with a slightly wider neckline and short sleeves. Rounded hem for best fit over the body.

1. The description contains only correct information about the garment in question.

Strongly disagree Disagree Neutral Agree Strongly agree

2. The description contains all relevant information to describe the garment.

Strongly disagree Disagree Neutral Agree Strongly agree

3. How do you evaluate the syntactic correctness of the sentence?

Very poor Poor Fair Good Excellent

NEXT

Figure 7.12: UI of the user study.

Data The clothing items evaluated in the survey belong to *Fashiongen* and the *industrial dataset#2*. We extract a random subset of 15 samples of the test partitions of each dataset and generate their descriptions using the fine-tuned *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder*, making the overall number of captions of the survey equal to 90.

Overall, we collect 786 ratings of descriptions of items by 33 users. Notice that users answering our survey usually do not rate all the 90 captions: on average, a user provides 24.1 ratings of descriptions of clothing items. Additionally, we check the distribution of answers per user and item and filter out the ratings of users that answered to a number of descriptions below a threshold (< 5 items rated) and the descriptions of clothing items that received few ratings (< 5 ratings of users) in at least one of the three scenarios (ground truth, *Multimodal GPT-2*, *Vision-Text Multi-Encoder Decoder*). Finally, we standardize

the remaining rates by removing the user biases: in this way, each rate does not depend on the user preference.

Results Figure 7.13 shows the distributions of the ratings the users give in the three questions to the ground truths and the captions generated by the two models. By visual inspection of the plots, the distributions of ratings related to the syntactical quality of the captions are almost completely overlapped (7.13c), highlighting that both *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* generate captions of clothing items whose syntactical quality is very close to captions written by professional stylists. Considering the first two questions, the distributions of the ratings of the generated captions are similar to the one of the ground truths but not as close as in the third question: the measure of correctness of the information provided in the description generated by *Vision-Text Multi-Encoder Decoder* is closer to the ground truth reference than *Multimodal GPT-2* (7.13a); instead, for both models, the distribution of ratings related to the number of relevant details mentioned in the descriptions is more distant to the reference with respect to the other two questions (7.13b).

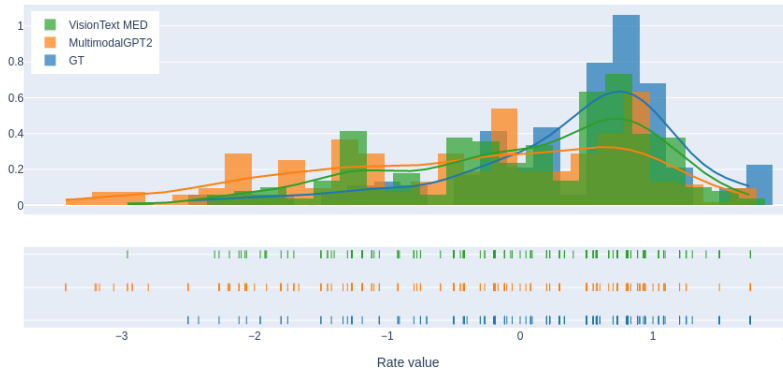
Given the observed data, we perform equivalence tests to draw statistical inferences concerning the average ratings of the description either generated by *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* or provided by experts. More precisely, given independent samples of the average rating of items in the three scenarios (*ground truth*, *Multimodal GPT-2*, and *Vision-Text Multi-Encoder Decoder*), we perform an equivalence test for each question of the survey to compare whether the mean ratings related to descriptions generated by our models differ by a small amount to the average ground truth rating. The statistical test we use is the *two one-sided t-test* (TOSTs): given a pre-defined margin of equivalence, it determines whether the means of two populations are equivalent, i.e., “statistically reject the presence of effects large enough to be considered worthwhile” [Lak17]. Given a margin of equivalence θ on a 5-point Likert scale, TOST considers the null and alternative hypotheses defined as:

$$\begin{aligned} H_0 & : \mu_2 - \mu_1 \leq -\theta \text{ or } \mu_2 - \mu_1 \geq \theta \\ H_A & : -\theta < \mu_2 - \mu_1 < \theta \end{aligned} \tag{7.1}$$

Practically, this test performs two one-sided t-tests considering one part of the null hypothesis at a time. The p -value of the TOST is the p -value with the higher value among the two tests.

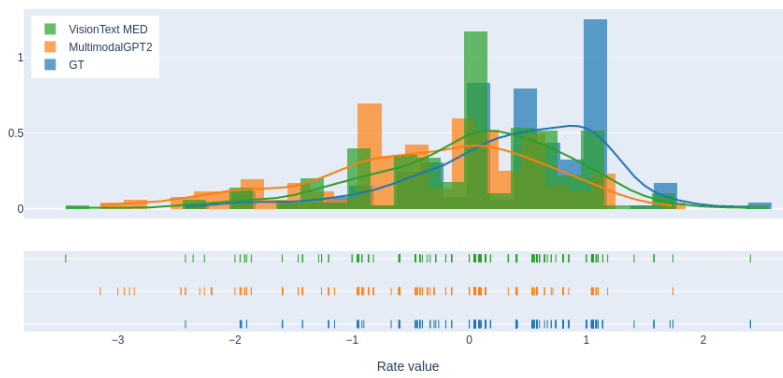
By considering a statistically significant threshold of **0.05** and an equivalence region with ± 1.0 rating margin, for both models and all the three questions, we can reject the null

*The description contains **only correct** information about the garment in question.*



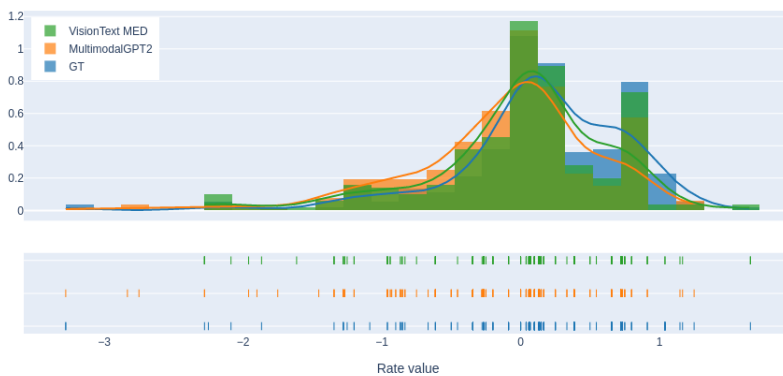
(a) First question.

*The description contains **all** relevant information to describe the garment.*



(b) Second question.

How do you evaluate the syntactic correctness of the sentence?



(c) Third question.

Figure 7.13: Distributions of the ratings provided by the users: the plots highlight the scenario and the question under observation.

hypothesis in favor of the alternative one and claim that the means of the ratings of the generated captions and the reference descriptions are equivalent, implying “either an effect that falls within the bounds or the absence of an effect that is worthwhile to examine” [Lak17]. Table 7.16 provides the results of the tests.

$\theta=1.0$	Q1	Q2	Q3
Multimodal GPT-2	0.042114	0.000546	1.265574e-13
Vision-Text MED	9.508813e-08	0.000005	4.157115e-09

Table 7.16: The resulting p -values of the equivalence tests (TOST) when considering an equivalence bound $\theta = 1.0$.

Reducing to ± 0.5 rating margin, thus considering a smaller equivalence region, we can draw different conclusions concerning the means of ratings related to the captions generated by *Multimodal GPT-2* and *Vision-Text Multi-Encoder Decoder* compared to the mean of the ratings of the ground truth captions. The results of the tests are provided in table 7.17.

- The syntactical quality (Q3) of both models can be assumed equivalent to the one of the ground truth, as we can reject H_0 and accept H_A at the given significance level.
- Considering the first two questions, we can’t reject effect sizes larger than the equivalence bound for the ratings of the descriptions generated by *Multimodal GPT-2*, while we can accept H_A for the ratings related to the captions generated by *Vision-Text Multi-Encoder Decoder*.

$\theta=0.5$	Q1	Q2	Q3
Multimodal GPT-2	0.881406	0.877911	0.000014
Vision-Text MED	0.008875	0.035432	0.000165

Table 7.17: The resulting p -values of the equivalence tests (TOST) when considering an equivalence bound $\theta = 0.5$.

In this analysis, we compare using three different criteria the quality of descriptions written by professional stylists with captions related to the same fashion items generated by architecture pre-trained according to our approach and fine-tuned using a very narrow set of target samples. Besides plotting the distribution of the survey data, we leverage statistical equivalence tests to highlight the performance of our training methodology and

novel architecture, showing whether the mean values of the collected ratings are equivalent according to different values of tolerance (equivalence margins). We find that the mean syntactical quality of generated captions is indistinguishable from the one related to the ground truths independently by the automated model. Differently, the question 1 and 2 measure the correctness of the descriptions in relation to the precision of the information provided and the relevance of the details mentioned, and according to the samples collected and the fixed significance level $\alpha = 0.05$, their mean values of ratings related to the captions generated by *Multimodal GPT-2* are statistically equivalent to the mean values related to the ground truths within a 1.0 rating margin. For *Vision-Text Multi-Encoder Decoder*, the same equivalence properties hold even within a 0.5 rating margin.

Using TOST, we can draw statistical conclusions about the means of the distributions of the ratings given by the users. To perform a more detailed comparison, we can consider other properties of the distributions of ratings, such as the dispersion of the observed data from their average value. We use the *two-sample Kolmogorov-Smirnov* test (KS test) to determine if there are significant differences between the distributions of the collected ratings. The KS statistic computes the distance between the empirical cumulative distribution functions of the two samples. Figure 7.14 shows the empirical CDFs of the observed data divided by questions.

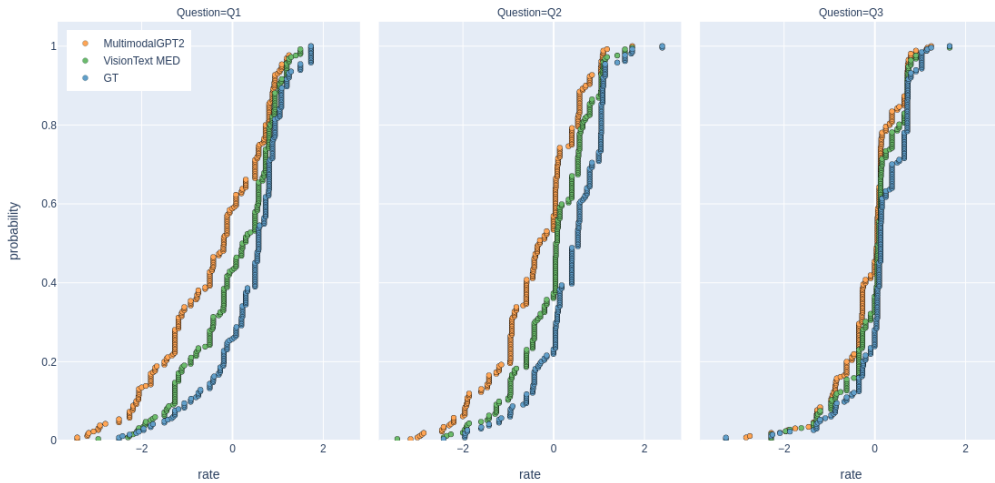


Figure 7.14: Empirical CDFs of the observed data divided by questions. The KS statistic is the distance between the empirical CDFs.

Given the empirical cumulative distribution function of ratings related to descriptions generated by an automated model $M(x)$ and the empirical cumulative distribution function

of ratings related to ground truth descriptions $G(x)$, the KS statistic is defined as:

$$D = \max_x |M(x) - G(x)|$$

The null hypothesis is H_0 : the two samples come from the same distribution; the alternative is H_A : the two samples do not come from the same distribution.

Table 7.18 provides the results of the KS tests. At the significance level alpha $\alpha = 0.05$, we can reject the null hypothesis in favor of the alternative in all the three tests related to *Multimodal GPT-2*, claiming that there is a statistical difference between the distribution of ratings related to the ground truth and the captions generated by the model. Differently, considering the results of the KS tests to compare the distributions of ratings related to the ground truths and the captions generated by *Vision-Text Multi-Encoder Decoder*, we can reject the null hypothesis in favor of the alternative only for the ratings related to the second question of our survey.

KS test, $\alpha=0.05$	Q1	Q2	Q3
Multimodal GPT-2	0.00333382	2.4583e-05	0.00889019
Vision-Text MED	0.18855666	0.02166983	0.09956245

Table 7.18: The resulting p -values of the KS tests to compare the distributions of observed data relative to generated captions and the ground truths.

While the TOST tests claim that, at a predefined equivalence margin θ , the means of the distributions of the ratings are statistically equivalent, the KS tests highlight a statistical difference between the distributions of ratings related to captions generated by automated models and the description written by professional stylists. This distance may be due to the structure of captions generated by the models. Through the final adaptation stage, the image captioners implicitly learn a suitable “average number” of details that should be present in the texts describing the clothing items; differently, the descriptions provided by fashion experts have higher variability in the number of details provided in the descriptions. This difference between the ground truths and the generated captions does not necessarily imply a “bad” behavior of the fashion captioner, as we provide statistical guarantees that the mean of the distributions of ratings related to ground truths and generated captions are similar.

To better describe the dispersion of the ratings, further analysis should be performed, even considering variations in the structure of the survey and maybe the consciousness level of the users regarding the fashion domain: details that fashion stylists deem obvious

could be relevant for usual e-commerce consumers and vice versa.

8 | Conclusion and Future Work

In this chapter, we discuss the main outputs and the contribution of our research work. Modern deep learning technologies generate text samples of outstanding quality, and when combined with a visual feature extractor, they precisely describe the subjects or the scene depicted in images at the cost of time-consuming training procedures over a large number of data samples. Moreover, the performance and the quality of the generated text samples drop when the model needs to process input samples that depart from the distribution of the data used during training.

In this thesis work, we analyze the generalization capabilities of such models, trying to overcome variations and perturbations in the input samples and still achieve high-quality descriptions of images. Specifically, we tackle this problem in the fashion domain, where clothing samples have a large number of details, and it is crucial to have high-quality descriptions of the products a fashion firm wants to sell online to attract more effectively the attention of customers. Besides, online catalogues continuously increase and change when new releases of fashion items enter the market: it would be beneficial to have a robust model able to overcome the variations in new clothing samples, saving the time, energy, and resources required to train a new model from scratch that describes the last releases of fashion items.

8.1. Outputs and contributions

We study and analyze the performance of fashion image captioning systems when the distribution of the data in input to the model changes. We design a pre-training procedure and a model architecture for the task of Fashion Image Captioning that we test using both the Fashiongen public dataset and other private ones. This task has recently gained more attention from the research community; we hope to provide further insights to help the research in this direction.

We propose a pre-training procedure that allows learning simultaneously from different sources and leverages a noise generation strategy to improve the generalization capabilities

of fashion image captioners. This pre-training approach does not depend on architectural choices, so we first compare the current state-of-the-art architecture in the “general” Image Captioning task with a recent promising model specifically designed for the fashion domain. We then perform a transfer learning analysis, showing how the performance of trained fashion image captioners varies according to the input distribution: in section 7.4, we show that the use of a model which is simply trained over a different source reflects a drastic reduction in the performance and the quality of the generated captions, not being able to recognize the clothing samples in input anymore, and by leveraging our pre-training method, we can improve the performance over unseen distributions of data, making the model generalize better. Still, the pre-trained model may struggle in identifying fine-grained details of clothing samples belonging to the target data source, but most importantly, a little conditioning stage on the target source allows to adapt the structure of generated description to the style of the target domain. We show that by performing a final adaptation stage of the pre-trained model using a very narrow set of target samples, the fashion image captioner achieves competitive performance and high-quality captions compared to the model extensively trained on the target source. This adaptation stage is far less expensive than the complete training procedure of a fashion image captioner from scratch: we show in 7.4 that only 5% of training samples and almost 3.5% of training time is required, underlying the saving in time, data samples, and resources.

Additionally, we design a novel Transformer-based [Vas+17] approach for Fashion Image Captioning that leverages the generative performance of the GPT-2 [Rad+19] language model along with the recent Vision Transformer (ViT) [Wu+20] and BERT [Dev+19] encoders to process a multi-modal input. Having two encoder stacks allows considering an additional self-supervised contrastive objective that aligns the embeddings of the two input modalities: we carry out a performance study among the different pre-training options that arise when using this additional learning objective. Moreover, we show in section 7.5 how the contrastive alignment between embeddings reflects on the representation learned by the model and how it improves the performance of fashion image captioners compared to baseline works.

Finally, we perform a user study to evaluate the quality of the description of clothing samples generated by image captioner systems pre-trained through our approach. We perform equivalence tests on the observed data to draw statistical inferences concerning the mean of the distributions of the ratings provided by the users. We demonstrate that the mean values of the distributions of the ratings given to the ground truths and to the captions generated by fashion captioners are statistically equivalent, assuming a

pre-defined tolerance value. Additionally, we find out that the tolerance value required to ensure the equivalence property between the mean values of the ratings related to the ground truths and the captions generated by our model is lower than the tolerance required by other fashion captioners.

8.2. Limitations

Our pre-training procedure presented in chapter 5 aims to improve the generalization capabilities of fashion image captioners by leveraging a noise generation strategy that combines clothing samples of different sources. We manually designed a reference taxonomy and the rules that guide the mapping between data sources: the granularity of the rules that map the clothing samples of different sources reflects the likelihood of noisy samples, thus the performance of our pre-training approach. Moreover, this pre-processing task is time-consuming and requires the design of a set of rules for each data source involved in the pre-training procedure.

Another limitation regards the hyperparameters optimization of our pre-training method and architecture: our analysis involves extensive training procedures, and our novel architecture introduces further contrastive learning options. Additionally, [Zha+21] empirically finds out that the simultaneous contrastive alignment of image-text embeddings using pre-trained architectures may be suboptimal while keeping a modality locked could improve the performance. In this work, we analyze some of the learning settings among the ones that, in principle, could arise when considering our pre-training method and a contrastive alignment.

Finally, the sample size of the user study we perform to evaluate the quality of the description of clothing samples generated by our models is limited, and a higher user involvement would have been better to back our findings.

8.3. Future Works

This research work introduces a pre-training method that exploits the taxonomies of datasets and hand-crafted rules to map clothing samples. A possible research direction is the study of automated techniques that generate a taxonomy given a data source or classify a fashion item to a taxonomy different from the one the clothing sample belongs to. These techniques would provide a “universal” mapping across datasets that, besides improving the performance of our pre-training method, could be used as a structured input to the encoder of our Transformer-based architecture. Having a standardized input

structure across sources could ease the learning procedure of the model and improve its generalization capabilities.

In section 2.1.2, we provide a definition of transfer learning which involves the notions of domain and task. Given source and target domains \mathcal{D}_S and \mathcal{D}_T , and source and target tasks \mathcal{T}_S and \mathcal{T}_T , there are four possible transfer learning scenarios, and the one we analyze in this thesis work is known as *domain adaptation*. Our experiments focus on the generalization capabilities of models when employed with data sources whose marginal probability distributions over the feature space are different from the training source, i.e., $P(X_S) \neq P(X_T)$. In particular, we consider differences in the images and the metadata in input to our models. Another difference worth considering between source and target task concerns the language of the descriptions of the clothing products: according to the classification provided in section 2.1.2, a *cross-lingual* analysis is characterized by $\mathcal{Y}_S \neq \mathcal{Y}_T$ and $P(Y_S|X_S) \neq P(Y_T|X_T)$. The label spaces and the conditional probability distributions between source and target tasks are different. This research direction has the practical implication that fashion firms often sell their products worldwide and, therefore, require descriptions in several languages.

As mentioned in the section concerning the limitation of our work, the research community is deeply investigating contrastive and self-supervised learning objectives to align embeddings of different modalities, and our architecture could benefit from these techniques.

In section 7, we draw statistical inferences regarding the mean values of the distributions of the survey data, and then we extend our analysis considering more generally the distance between the distributions of the ratings. The distributions of the ratings have different levels of dispersion: we impute this difference to the variability in the number of details provided in the descriptions written by fashion experts. To better describe this behavior, further analysis should be performed, even considering variations in the structure of the survey and maybe the consciousness level of the users regarding the fashion domain: details that fashion stylists deem obvious could be relevant for usual e-commerce consumers and vice versa.

Another research step related to our work is the exploration of our approach along with Fashion Tagging solutions: fashion firms that sell their products online leverage systems that automatize the labeling of the clothing products, and they need to work together with the fashion image captioner. The integration of the two modules could not be trivial and can affect their performance.

Bibliography

- [Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].
- [Rad+19] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.
- [Wu+20] Bichen Wu et al. *Visual Transformers: Token-based Image Representation and Processing for Computer Vision*. 2020. arXiv: 2006.03677 [cs.CV].
- [Dev+19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].
- [PY10] Sinno Jialin Pan and Qiang Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- [Rud17] Sebastian Ruder. *Transfer Learning - Machine Learning’s Next Frontier*. <http://ruder.io/transfer-learning/>. 2017.
- [Sar18] Dipanjan Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>. 2018.
- [Den+09] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [SZ15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].
- [Sze+14] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].
- [He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].
- [PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Empirical Methods in Natural*

- Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: <http://www.aclweb.org/anthology/D14-1162>.
- [Mik+13] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].
- [BS17] Joachim Bingel and Anders Søgaard. *Identifying beneficial task relations for multi-task learning in deep neural networks*. 2017. arXiv: 1702.08303 [cs.CL].
- [Raf+19] Colin Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *arXiv preprint arXiv:1910.10683* (2019).
- [Tiu21] Ekin Tiu. *Understanding Contrastive Learning*. <https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607>. 2021.
- [Ala18] Jay Alammar. *The Illustrated Transformer*. <https://jalammar.github.io/illustrated-transformer/>. 2018.
- [BDW21] Hangbo Bao, Li Dong, and Furu Wei. *BEiT: BERT Pre-Training of Image Transformers*. 2021. arXiv: 2106.08254. URL: <https://arxiv.org/abs/2106.08254>.
- [PRS19] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. “To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks”. In: *ArXiv abs/1903.05987* (2019).
- [Pet+18] Matthew E. Peters et al. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].
- [VN21] Wietse de Vries and Malvina Nissim. “As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages”. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021* (2021). DOI: 10.18653/v1/2021.findings-acl.74. URL: <http://dx.doi.org/10.18653/v1/2021.findings-acl.74>.
- [Rad+21] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV].
- [Zha+21] Xiaohua Zhai et al. *LiT: Zero-Shot Transfer with Locked-image Text Tuning*. 2021. arXiv: 2111.07991 [cs.CV].
- [Xu+15] Kelvin Xu et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2048–2057. URL: <https://proceedings.mlr.press/v37/xuc15.html>.
- [Li+20] Xiujun Li et al. *Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks*. 2020. arXiv: 2004.06165 [cs.CV].

- [PSC20] Umberto Pietroni, Federico Sallemi, and Paolo Cremonesi. “Image tagging and captioning for fashion catalogues enrichment”. In: (2020). URL: <https://www.politesi.polimi.it/handle/10589/169410?mode=complete>.
- [Ros+18] Negar Rostamzadeh et al. “Fashion-gen: The generative fashion dataset and challenge”. In: *arXiv preprint arXiv:1806.08317* (2018).
- [Lin+14] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [Zho+20] Xinyuan Zhou et al. *Multi-Encoder-Decoder Transformer for Code-Switching Speech Recognition*. 2020. arXiv: 2006.10414 [eess.AS].
- [Pap+02] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: <https://aclanthology.org/P02-1040>.
- [Wu+16] Yonghui Wu et al. *Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].
- [BL05] Satanjeev Banerjee and Alon Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”. In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 65–72. URL: <https://aclanthology.org/W05-0909>.
- [Lin04] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: <https://aclanthology.org/W04-1013>.
- [van+21] Chris van der Lee et al. “Human evaluation of automatically generated text: Current trends and best practice guidelines”. In: *Computer Speech & Language* 67 (2021), p. 101151. ISSN: 0885-2308. DOI: <https://doi.org/10.1016/j.csl.2020.101151>. URL: <https://www.sciencedirect.com/science/article/pii/S088523082030084X>.
- [Lak17] Daniël Lakens. “Equivalence Tests: A Practical Primer for t Tests, Correlations, and Meta-Analyses”. In: *Social Psychological and Personality Science* 8.4 (2017). PMID: 28736600, pp. 355–362. DOI: 10.1177/1948550617697177. eprint: <https://doi.org/10.1177/1948550617697177>. URL: <https://doi.org/10.1177/1948550617697177>.

A | Taxonomy

Here we report the entire reference taxonomy we leverage in our noise generation approach.

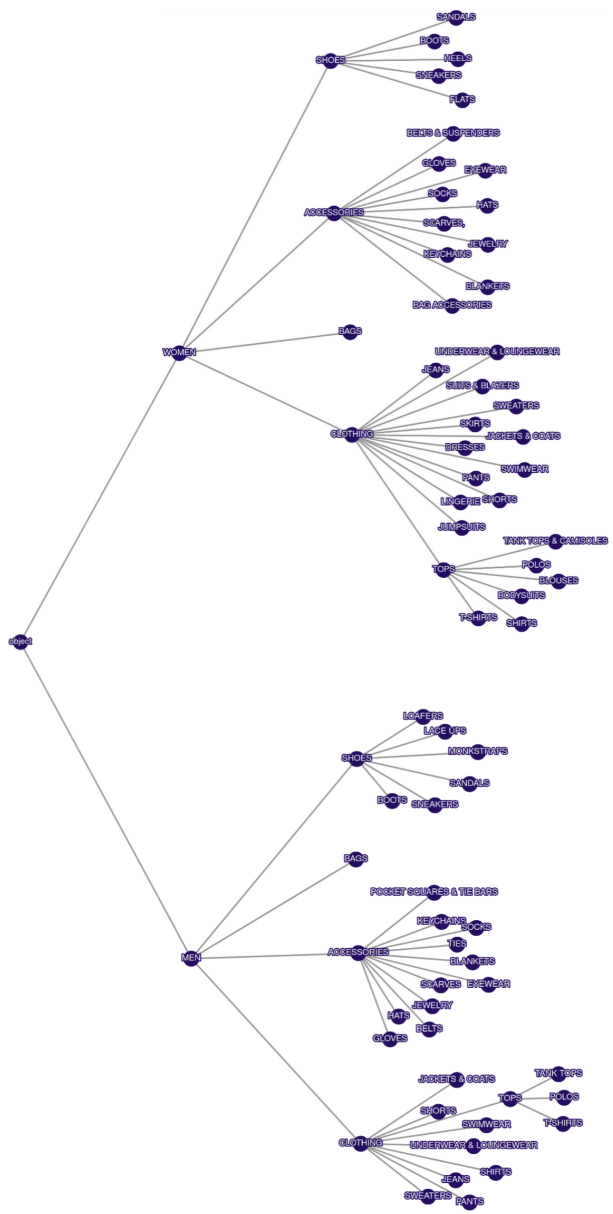


Figure A.1: Reference taxonomy.

B | Additional Tables

This chapter provides more detailed results about the architectures we present in 7: we consider multiple choices of n -grams when computing the BLEU and ROUGE scores. As mentioned in chapter 7, the scores related to choices of n -grams with $n > 1$ are consistent with the ones related to *unigram*.

Algorithm	BLEU-1	BLEU-2	BLEU-3	BLEU-4	GLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Results on Fashiongen									
Show, Attend, and Tell	37.39	28.87	19.80	14.28	22.89	41.49	53.12	32.81	50.33
Image GPT-2	48.19	38.30	28.68	21.58	29.56	51.70	61.56	40.14	57.83
Results on the industrial dataset#1									
Show, Attend, and Tell	28.65	15.79	9.06	6.48	12.95	23.32	35.57	12.05	26.09
Image GPT-2	31.85	18.81	11.50	9.37	16.03	27.33	38.25	15.00	29.55

Table B.1: Results of *Show, Attend, and Tell* and *Image GPT-2* on the Fashiongen dataset and the industrial dataset#1 considering multiple choices of n -grams.

Algorithm	BLEU-1	BLEU-2	BLEU-3	BLEU-4	GLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Results on Fashiongen									
OSCAR*	42.16	32.27	22.09	15.38	24.01	46.01	56.00	34.47	52.03
Multimodal GPT-2	48.87	39.10	29.60	22.24	30.33	52.44	62.41	41.15	58.72
Results on the industrial dataset#1									
OSCAR*	39.26	27.36	20.23	16.87	22.71	37.84	49.60	25.81	38.59
Multimodal GPT-2	41.14	28.63	21.73	18.58	24.35	39.97	50.66	27.00	39.39

Table B.2: Results of *OSCAR** and *Multimodal GPT-2* on the Fashiongen dataset and the industrial dataset#1 considering multiple choices of n -grams. The architecture of *OSCAR** is described in 7.3.

Algorithm	BLEU-1	BLEU-2	BLEU-3	BLEU-4	GLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Results on Fashiongen									
Multimodal GPT-2	48.87	39.10	29.60	22.24	30.33	52.44	62.41	41.15	58.72
Vision-Text MED	51.13	41.56	32.31	24.90	32.63	55.06	65.42	44.22	61.47
Results on the industrial dataset#2									
Multimodal GPT-2	49.41	38.41	30.19	24.18	31.72	53.50	61.56	39.23	56.28
Vision-Text MED	53.38	42.82	34.74	28.45	35.52	58.65	66.00	44.24	60.73

Table B.3: Results of *Multimodal GPT-2* and our *Vision-Text Multi-Encoder Decoder* on the Fashiongen dataset and the industrial dataset#2 considering multiple choices of n -grams. The training objective of *Vision-Text Multi-Encoder Decoder* is the combination of the language modeling objective loss and the contrastive loss as discussed in section 6.2.1; The hyperparameter λ is set to 0.3.

C | Examples of generated captions

In this section we provide examples of captions generated by our approach, here identified as *Vision-Text Multi-Encoder Decoder*, and the *Multimodal GPT-2* architecture proposed in [PSC20]. Both models are fine tuned as discussed in chapter 7. The clothing samples are random examples taken from the test partitions of the industrial dataset#2 and the Fashiongen dataset.

Fashiongen



Metadata: clothing, men, tops, t-shirts.

Ground truth: Short sleeve t-shirt in black. Ribbed crewneck collar. Monster eyes leather appliqué in yellow with silver-tone studded detail. Tonal stitching.

Multimodal GPT-2: Short sleeve cotton jersey t-shirt in black. Rib knit crewneck collar. Tonal stitching.

Vision-Text Multi-Encoder Decoder: Short sleeve cotton jersey t-shirt in black. Rib knit crewneck collar. Logo printed at front. Tonal stitching.



Metadata: clothing, men, jeans.

Ground truth: Skinny-fit jeans in 'mariner' blue. Distressing throughout. Five-pocket styling with logo embroidered at back pocket in off-white. Button-fly. Contrast stitching in tan.

Multimodal GPT-2: Skinny-fit stretch denim jeans in black. Four-pocket styling. Zip-fly. Tonal stitching. Approx. 6.5" leg opening.

Vision-Text Multi-Encoder Decoder: Slim-fit jeans in blue. Mid-rise. Fading, whiskering, and honeycombing throughout. Five-pocket styling. Logo patch at back waistband. Zip-fly. Silver-tone hardware. Contrast stitching in tan. Approx. 6" leg opening.



Metadata: clothing, men, shorts.

Ground truth: Relaxed-fit French terry lounge shorts in black. Drawstring at elasticized poplin waistband. Three-pocket styling. Raw edge at hem. Tonal stitching.

Multimodal GPT-2: Relaxed-fit denim shorts in black. Four-pocket styling. Button-fly. Tonal stitching. Zip-fly.

Vision-Text Multi-Encoder Decoder: Relaxed-fit cotton jersey sarouel shorts in black. Drawstring at elasticized waistband. Four pocket styling. Zip-fly. Tonal stitching.



Metadata: clothing, women, jackets & coats, jackets.

Ground truth: Long sleeve ribbed silk jacket in faded black. Self-tie straps at spread collar, sleeves, and waist. Patch pockets at chest. Press-stud closure at front. Tonal stitching.

Multimodal GPT-2: Long sleeve French terry jacket in black. Spread collar. Two-way zip closure at front. Welt pockets at waist. Fully lined. Tonal stitching.

Vision-Text Multi-Encoder Decoder: Long sleeve cotton jacket in navy. Spread collar. Button closure at front. Flap pockets at body. Single-button barrel cuffs. Tonal stitching.



Metadata: clothing, men, sweaters, sweatshirts.

Ground truth: Long sleeve sweatshirt striped in black and white. Ribbed crewneck collar. Logo print at front center. Overlong zip-away sleeves with two functional zippers. Tonal stitching.

Multimodal GPT-2: Long sleeve French terry sweatshirt in black. Rib knit crewneck collar, cuffs, and hem. Tonal stitching.

Vision-Text Multi-Encoder Decoder: Long sleeve French terry sweatshirt in white. Rib knit crewneck collar, cuffs, and hem. Logo printed at front. Tonal stitching.

Industrial dataset#2



Metadata: clothing, shirts & blouses, casual, men, shirts, blue.

Ground truth: Shirt in an airy patterned weave made of cotton blend with a turn-down collar French front and yoke at the back. Long sleeves with adjustable buttoning at the cuffs and a rounded hem.

Multimodal GPT-2: Short-sleeved shirt in woven fabric with a stand-up collar button down the front and long sleeves with buttoned cuffs.

Vision-Text Multi-Encoder Decoder: Shirt in woven cotton fabric with a collar buttons down the front and long sleeves with adjustable buttoning at cuffs. Rounded hem slightly longer at back. Regular fit – designed to create a comfortable tailored silhouette.



Metadata: clothing, hoodies & sweatshirts, sweatshirts, hoodies & sweatshirts, men, pink.

Ground truth: Long-sleeved top in printed sweatshirt fabric with ribbing around the neckline cuffs and hem. Soft brushed inside.

Multimodal GPT-2: Long-sleeved sweatshirt in soft cotton jersey with a printed pattern. Ribbing at neckline cuffs and hem.

Vision-Text Multi-Encoder Decoder: Sweatshirt in soft cotton jersey with a printed design. Ribbing at neckline cuffs and hem. Soft brushed inside.



Metadata: clothing, vests, tops, ladies, pink, divided.

Ground truth: Pleated top in woven fabric containing glittery threads in a narrow cut at the top with short narrow shoulder straps and a faceted button at the back. Lined.

Multimodal GPT-2: V-neck top in a viscose blend with short sleeves and a rounded hem.

Vision-Text Multi-Encoder Decoder: Sleeveless top in woven fabric with a narrow shoulder straps and smocking at the back. Lined.



Metadata: clothing, t-shirts & Vests polo, men, tshirts tanks, blue.

Ground truth: Short-sleeved polo shirt in cotton piqué with a printed pattern. Collar button placket and short slits at sides.

Multimodal GPT-2: T-shirt in cotton jersey with a printed design.

Vision-Text Multi-Encoder Decoder: T-shirt in soft cotton jersey with a printed design. Regular fit.



Metadata: clothing, skirts, shorts, ladies, red, adult, everyday fashion.

Ground truth: Short skirt in viscose-blend twill with a high waist and removable tie belt. Pleats concealed side-seam zip and gathered seam at hem with flounce.

Multimodal GPT-2: Knee-length skirt in woven fabric with an elasticated drawstring waist and a concealed zip at the back.

Vision-Text Multi-Encoder Decoder: Short skirt in woven crêpe with a high waist and concealed zip. Unlined.



Metadata: clothing, t-shirt & vests, short sleeves, men, tshirts tanks, white.

Ground truth: T-shirt in cotton jersey with a printed pattern.

Multimodal GPT-2: Short-sleeved T-shirt in cotton jersey.

Vision-Text Multi-Encoder Decoder: T-shirt in cotton jersey with a printed motif. Regular fit.

List of Figures

2.1	Transfer learning with a pre-trained model as features extractor [Sar18].	8
2.2	Multi-task approach: the learner receives information from all the tasks simultaneously [Sar18].	10
2.3	Training error (left) and test error (right) with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error [He+15].	12
2.4	The ResNet block with the <i>identity shortcut connection</i> [He+15].	13
2.5	RNN vs LSTM.	15
2.6	Gates of a LSTM cell.	16
2.7	Architecture of the Transformer model [Vas+17].	17
2.8	Summary of the steps of the <i>multi-head self-attention</i> layer [Ala18].	18
2.9	BERT pre-training.	20
2.10	Model architecture and processing steps of ViT [Wu+20].	22
2.11	ViT performance.	23
3.1	CLIP approach.	28
3.2	The design choices of “contrastive-tuning”.	29
3.3	Examples of attended regions (in <i>white</i>) with the corresponding <i>word</i> generated as output [Xu+15].	30
3.4	Oscar pre-training [Li+20].	31
3.5	Multimodal GPT-2 architecture [PSC20].	32
4.1	Bar plot representing the distributions of the categories of the fashion items belonging to Fashiongen.	34
4.2	Statistics of the descriptions of clothing samples belonging to Fashiongen.	35
4.3	Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#1.	36
4.4	Statistics of the descriptions of clothing samples belonging to the industrial dataset#1.	37

4.5	Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#2.	38
4.6	Statistics of the descriptions of clothing samples belonging to the industrial dataset#2.	39
4.7	Bar plot representing the distributions of the categories of the fashion items belonging to the industrial dataset#3.	40
4.8	Statistics of the descriptions of clothing samples belonging to the industrial dataset#3.	41
4.9	Example of fashion images of the dataset used in this thesis work.	43
5.1	Example of weights given a batch assignment.	49
5.2	Comparison of generated captions.	50
5.3	Pairings between two taxonomies	52
5.4	Subset of the reference taxonomy.	53
5.5	Hybrid sample	54
6.1	Vision-Text Multi-Encoder Decoder architecture.	58
6.2	Comparison between decoder blocks.	59
6.3	The vision and the text encoders are jointly trained to predict the correct pairings between images and metadata of the current training batch. Image adapted from [Rad+21].	60
6.4	Weighted random selection of the images of a garment when the total number of images is four.	62
7.1	Example of the visual semantics extracted by the object detector module used by OSCAR.	68
7.2	Our adaptation of OSCAR for Fashion Image Captioning.	68
7.3	Example of captions generated by the <i>target-only</i> and the <i>source-only</i> models.	72
7.4	Example of captions generated by the pre-trained model and the ones provided by the model references and the corresponding ground truth.	75
7.5	Comparison between the captions generated by the models involved in each step of our transfer learning analysis.	77
7.6	Examples of captions generated by the models involved in each learning stage, compared with the ground truth and the <i>source-only</i> and <i>target-only</i> references.	80
7.7	Visual representation of the two techniques we use to perform the contrastive alignment between the image and metadata embeddings.	81
7.8	Train and validation partitions according to the pre-training step.	82

7.9	Comparison between the captions generated by the fine-tuned <i>Multimodal GPT-2</i> and <i>Vision-Text Multi-Encoder Decoder</i>	85
7.10	t-SNE visualization of image and metadata embeddings.	87
7.11	Results of <i>Multimodal GPT-2</i> and <i>Vision-Text Multi-Encoder Decoder</i> in each stage of our transfer learning analysis.	88
7.12	UI of the user study.	90
7.13	Distributions of the ratings provided by the users	92
7.14	Empirical CDFs of the observed data divided by questions. The KS statistic is the distance between the empirical CDFs.	94
A.1	Reference taxonomy.	105

List of Tables

- 4.1 Examples of descriptions of fashion items belonging to Fashiongen with the corresponding product categories. 34
- 4.2 Examples of descriptions of fashion items belonging to the industrial dataset#1 with the corresponding product categories. 36
- 4.3 Examples of descriptions of fashion items belonging to the industrial dataset#2 with the corresponding product categories. 38
- 4.4 Examples of descriptions of fashion items belonging to the industrial dataset#3 with the corresponding product categories. 40
- 4.5 Number of items in train, validation, and test splits along with the average number of words (\bar{w}) and sentences (\bar{s}) per caption. 42

- 5.1 Textual transformations applied to clothing metadata and captions. 46
- 5.2 Metadata samples before (*left*) and after (*right*) applying the textual transformations. 47

- 7.1 BLEU-*n* example 64
- 7.2 Results on Fashiongen and the industrial dataset#1 of *Show, Attend, and Tell* and *Image GPT-2* 67
- 7.3 Results on Fashiongen and the industrial dataset#1 of *OSCAR** and *Multimodal GPT-2*. 69
- 7.4 Results of the *target-only* model on Fashiongen and the industrial dataset#2 71
- 7.5 Time and data requirements to train the *target-only* models. 71
- 7.6 *Source-only* scores on Fashiongen and the industrial dataset#2 71
- 7.7 Assignment of datasets among pre-training and test sources. 73
- 7.8 Performance of our pre-training methodology applied to *Multimodal GPT-2* when tested on the Fashiongen dataset and the industrial dataset#2. . . 74
- 7.9 Recap of the training datasets used in each stage of our analysis according to the *target* dataset. 76

7.10	Comparison of the performance achieved in each step of our transfer learning methodology applied to <i>Multimodal GPT-2</i> when tested on Fashiongen and the industrial dataset#2.	76
7.11	Time and data requirements to fine-tune the <i>pre-trained</i> models.	77
7.12	Comparison of the performance achieved in each step of our transfer learning methodology applied to <i>Vision-Text Multi-Encoder Decoder</i> when tested on Fashiongen and the industrial dataset#2.	79
7.13	Comparison of the results of the three pre-training approaches when tested on Fashiongen and the industrial dataset#2.	82
7.14	Comparison of the performance achieved in each step of our transfer learning methodology by <i>Multimodal GPT-2</i> and <i>Vision-Text Multi-Encoder Decoder</i> when tested on Fashiongen and the industrial dataset#2.	84
7.15	Comparison of the results of <i>Multimodal GPT-2</i> and <i>Vision-Text Multi-Encoder Decoder</i> when trained either using used or not using the contrastive alignment through the multi-objective learning approach.	86
7.16	The resulting p -values of the equivalence tests (TOST) when considering an equivalence bound $\theta = 1.0$	93
7.17	The resulting p -values of the equivalence tests (TOST) when considering an equivalence bound $\theta = 0.5$	93
7.18	The resulting p -values of the KS tests to compare the distributions of observed data relative to generated captions and the ground truths.	95
B.1	Complete results on Fashiongen and the industrial dataset#1 of <i>Show, Attend, and Tell</i> and <i>Image GPT-2</i>	107
B.2	Complete results on Fashiongen and the industrial dataset#1 of <i>OSCAR*</i> and <i>Multimodal GPT-2</i>	107
B.3	Complete results on Fashiongen and the industrial dataset#2 of <i>Multimodal GPT-2</i> and <i>Vision-Text Multi-Encoder Decoder</i>	108

Acknowledgements

First, I would like to thank my supervisor, Prof. Paolo Cremonesi, for the support and the opportunity to join the high-level working environment of ContentWise.

This work would not have been possible without Federico Sallemi and Umberto Pietroni: thank you for your constant help and guidance during these months.

I would like to thank my parents, who have always supported and trusted me along this journey, and my brothers, Riccardo and Tommaso, who have been a precious presence during these years.

I also wish to thank all my friends and coursemates for all the time and fun we have shared during these university years.

Finally, my deepest thanks to Arianna: thank you for being with me since the beginning of this journey and for your fundamental support, especially in the most tiring moments.

