Master Data Sciences

Internship report

Leveraging Weakly Annotated Data for Fashion Image Retrieval and Label Prediction

Author:
Charles Corbière

Supervisors:
Charles Ollion
Hedi Ben-Younes
Erwan Le Pennec

December 9, 2024

Declaration of Authorship

I, Charles Corbière, declare that this internship report titled, 'Leveraging Weakly Annotated Data for Fashion Image Retrieval and Label Prediction' and the work presented in it are my own. I confirm that:

- This work was done wholly during my internship at Heuritech.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Acknowledgements

Foremost, I would like to thank my internship supervisors, Charles Ollion and Hedi Ben-Younes, for their support during my internship. Their knowledge, their advises, their patience and our discussions about this research subject not only helped me in all the time of research and writing of this report but also improved my understanding in deep learning and in conducting applied research.

My sincere thanks also goes to all the Research team at Heuritech for the discussions we have. It was really enjoyable to evolve in a such motivating and challenging environment. More precisely, I'd specially like to thank Alexandre Ramé and Pierre Dubreuil for their help on transfer learning evaluation tasks, respectively clothing attributes tagging and image retrieval. Still at Heuritech, I'd like to thank all the Heuritech team for providing an efficient network infrastructure and on a more general matter, for their welcoming and the great atmosphere in the company.

I'd finally like to thank my advisor at Polytechnique, Prof. Erwan Le Pennec and the rest of my internship report committee: Prof. Eric Moulines, Prof. Zoltan Szabo and Ms. Michèle Gesbert.

Contents

Introduction

While online shopping has been an exponentially growing market for the last two decades, finding exactly what you want from online shops is still not a solved problem. For instance, on Zalando, Europe's leading online fashion platform, one can denotes more than 150K articles [1], which makes efficient search a crucial matter. Traditional fashion search engines allow consumers to look for products based on well chosen keywords. Such engines match those textual queries with meta-data of products, such as a title, a description or a set of tags. In online luxury fashion for instance, they still play an important role to address this customer pain point: 46% of customers use a search engine to find a specific product; 31% use it to find the brand they're looking for [2]. However, those meta-data information may be incomplete, or use a biased vocabulary. For instance, a description may denote as "marinière" a long sleeves shirt with blue/white stripes. It then appears crucial for online retailers to have a rich labeled catalog to ensure good search. Moreover, these search engines don't incorporate the visual information of the image associated to the product.

Computer vision for fashion e-commerce images has drawn an increasing interest in the last decade. It has been used for similarity search [1, 2, 3, 4], automatic image tagging [5, 6], fine-grained classification [7, 8] or N-shot learning [9]. In all of these tasks, a model's performance is highly dependent on a visual feature extractor. Using a Convolutional Neural Network (CNN) trained on ImageNet [10] provides a good baseline. However, there are two main problems with this representation. First, it has been trained on an image distribution that is very far from e-commerce, as it has never (or rarely) seen such pictures. Second, the set of classes it has been trained on is different from a set of classes that could be meaningful in e-commerce. A useful representation should separate different types of clothing (*e.g.* a skirt and a dress), but it should also discriminate between different lengths of sleeves for shirts, trouser cuts, types of handbags, textures, colors, shapes,...

My goal is to learn a visual feature extractor designed for e-commerce images. This representation should:

[1] https://www.zalando.fr/presse-informations-et-chiffres-cles/

[2] http://www.mckinsey.com/business-functions/marketing-and-sales/my-insights/the-opportunity-in-online-luxury-fashion

- encode multiple levels of visual semantics: from low level signals (color, shapes, textures, fabric,...) to high level information (style, brand),
- be separable over visual concepts, so I can train very simple classifiers over clothing types, colors, attributes, textures, *etc.*,
- provide a meaningful similarity between images, so I can use it in the context of image retrieval.

As a matter of fact, my architecture must learn useful representations with (very) noisy labels, scale to a vocabulary up to 100K words, present good properties for transfer learning.

To these ends, I train a visual feature extractor on a large set of weakly annotated images crawled from the Internet. These annotations correspond to the textual description associated to the image. The model is learned on a dataset at *zero* labeling cost, and is exclusively constituted of data points extracted from e-commerce websites. My main contribution is an in-depth analysis of the model presented in [11], through applications to fashion image recognition tasks such as image retrieval and attribute tagging. I also improved the method by upgrading the CNN architecture and dealt with multiple languages, mainly English and French. In Chapter 1, I first contextualize my work in Heuritech by describing deep learning and computer vision concepts and by giving examples of applications to the fashion industry. Then in Chapter 2, I explain the model, how I handled noise in the dataset, as well as some implementation details. In Chapter 3, I provide results given by my representation on image retrieval and classification, over multiple datasets. Finally in Section 4, I go over possible improvement tracks we tried afterwards.

Chapter 1

Computer vision for fashion

Tony Pinville, Charles Ollion, Didier Martin and Paul Tonelli met during their thesis in artificial intelligence at LIP6, a computer science research laboratory, part of Pierre et Marie Curie University and French National Center for Scientific Research (CNRS). Together, they founded Heuritech in 2013 to build on their knowledge in machine learning and deep learning. Now, the company is composed of more than twenty employees, mostly technical experts with half of them holding a PhD in Artificial Intelligence. Heuritech focus on fashion, luxury and e-commerce applications. Their key technology is deep learning for visual recognition. After introducing the basic notions of neural networks and deep learning, this chapter will detail why they are used heavily in computer vision tasks.

1.1 Neural network and deep learning

Deep learning is a class of machine learning algorithms based on a deep multilayered structure of non linear processing units called neurons. As show on Figure 1.1, a single neuron consists of inputs, an activation function and an output. Let the inputs be a n-dimensional vector $x \in \mathbb{R}^n$. The output is computed by the following function:

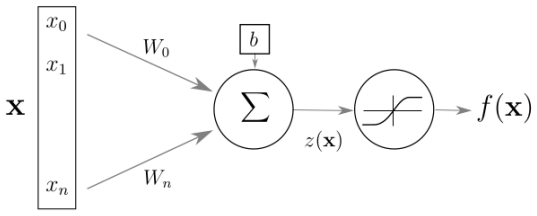


Figure 1.1: Schema of an artificial neuron[1]

$$f(x) = g(w^T x + b) \tag{1.1}$$

[1] https://github.com/m2dsupsdlclass/lectures-labs

where g defines the activation function. This function is also called a nonlinearity and commonly used examples are the sigmoid function:

$$g(x) = sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

or the hyperbolic tangent function :

$$g(x) = tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1.3}$$

or, more recently, the Rectified Linear Unit (ReLU) :

$$g(x) = max(0, x) \tag{1.4}$$

The sigmoid activation function maps any real number to the $[0, 1]$ interval. With this unit, the activation can be interpreted as the probability for the "neural unit" parameterized by w and the bias b. Despite the loss of a probabilistic interpretation, the ReLU function [12] gained a lot of popularity these last years, due to its empirical performance to address the vanishing gradient issue. A multilayer feedforward network stacks such single neurons in multiple layers and add a final output layer depending on the tasks.

A feed-forward network benefits of the Universal Approximation Theorem. Namely, with only a single hidden layer containing a finite number of neurons and under certain assumptions on the activation function, the model can approximate any continuous functions on compact subsets of $\mathbb{R}^n$. More precisely, let σ be a nonconstant, bounded, and monotonically-increasing continuous function. For any $f \in C([0, 1]^d)$ and $\epsilon \geq 0$, Kurt Hornik showed in 1991 [13] that there exists $h \in \mathbb{N}$, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^d$ such that:

$$| \sum_{i}^{h} v_i \sigma(w_i^T x + b_i) - f(x) | \leq \epsilon \tag{1.5}$$

This still holds for any compact subset of $\mathbb{R}^d$ and, more importantly, if σ is the ReLU function [14]. Although neural networks can approximate any continuous function, they can suffer from the vanishing gradient problem and encounter some difficulties converging to a minimum. Vanishing gradients affect any gradient-based learning methods and back-propagation. In such methods, each of the neural network's weights receives an update proportional to the gradient of the error function. Sigmoid or hyperbolic tangent activation have gradients respectively is in the range (0,1) and (-1, 1), and backpropagation computes gradients by the chain rule. The error signal will then decreases exponentially when getting to the front layers and not be able to update correctly their weights.

1.2 Visual feature extraction

In order to recognize objects in complex scenes, one needs a flexible representation of the world. One main aspect in computer vision research is to create a model able to extract good visual features for classifier or regressors. During the 2000's, the best accuracy was obtained using a hand-crafted model called the Bag of visual Words (BoW). First, robust features extractors, such as SIFT [15], were applied to the dataset for extracting local descriptors from the images. Then, a clustering algorithm was used to obtain a visual descriptor codebook. Finally, each image were assigned to their own representation in a lower space thanks to a pooling step aggregating all the descriptors.

But in 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton released a paper where they introduced a "large, deep convolutional neural network" [16] crafted to address the well-known ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). Not only they win the 2012 ILSVRC, but they outperformed the previous year top-5 classification error by nearly 10 points. Since, convolutional neural networks (CNNs) have trusted every year best places. Figure 1.2 plots the evolution of top-5 error from 2010 to 2015. From 25,8% before the release of Alexnet to 3,57% in 2015 with the ResNet50 [17] of Microsoft, CNN's have even outperformed human results [18]. Another interesting results is that the deeper the network is, the better it performs: AlexNet was made of 8 layers at the time while the latest ResNet model is now 152 layers.

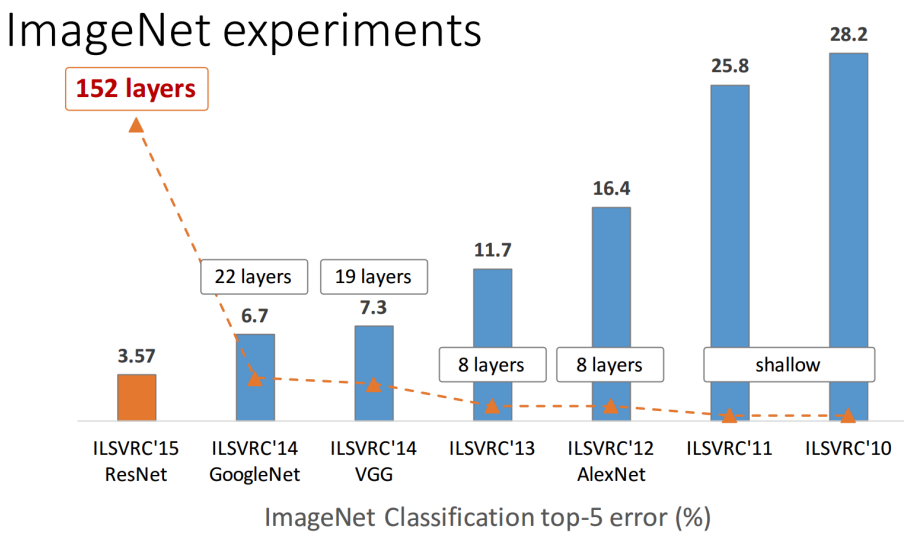


Figure 1.2: Evolution of classification top-5 error on ILSVRC challenge from 2010 to 2015.

Although there is now a lot of different architectures of convolutional neural networks such as LeNet on Figure 1.3, they usually have in common to be a sequence of the following layers:

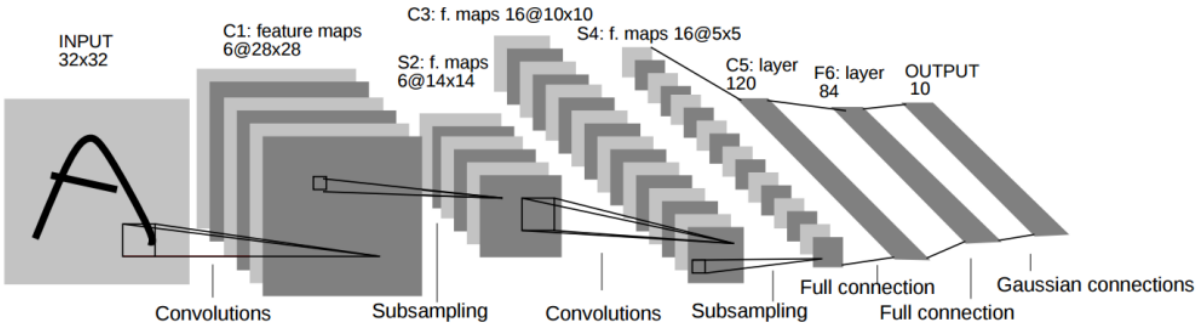


Figure 1.3: Yann Lecun's LeNet architecture [19]

Convolution layers Followed by an activation function such as ReLU, a convolution layer operates a convolution operation on a image. For instance, considering Figure 1.4, the operation writtes as:

$$(k \circledast im)(x, y) = \sum_{n=0}^{2} \sum_{m=0}^{2} k(n, m) \cdot im(x - n - 1, y - m - 1) \tag{1.6}$$

Here, in a convolution layer, each output neuron is parametrized with the kernel weights w.

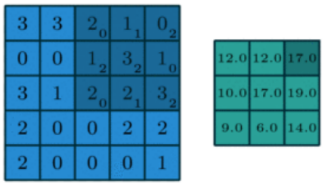


Figure 1.4: Example of a convolution of a 5x5 image by a 3x3 kernel

Spatial pooling layers Placed after a convolution, a pooling layer provides local invariance and a spatial dimension reduction. The aggregation functions mainly used are a maximum or an average pooling, defined with the high and width dimensions of the area where it is applied. Note that there is no parameters to learn here.

Batch normalization As the cost function of a neural network is non convex, there is no theoretical guarantee to converge to a global minimum. However, when 'wide enough', training error can often reach zero. Moreover, Stochastic Gradient Descent (SGD) with momentum seem to be able to evade saddle structures quite easily [20]. With convolutional networks becoming deeper and deeper, batch normalization quickly became very popular mostly because it helps to converge faster [21]. It relies on adding a a normalization step (shifting inputs to zero-mean and unit variance) to make the inputs of each trainable layers comparable across features. By doing this it ensures a high learning rate while keeping the

network learning.

Coupled with batch normalization layer, ResNet [17] use special skip connections, called residual connections, for training deeper architecture (for the full architecture detail of Resnet50, please refers to Appendix A). Those two methods help addressing exploding or vanishing gradient issues.

1.3 Applications in fashion

In the fashion industry, many tasks require human-level cognitive skills, such as detecting similar products or identifying facets in products (e.g. stripes, textures, or how does a customer wear his bag). As stated in the introduction, traditional fashion search engines allow consumers to look for products based on keywords present in a title, a description or a set of tags. Here, with using CNN, one can empower its product knowledge by extracting visual information from product's associated images. Visual applications in fashion are numerous:

Attribute tagging Descriptive visual attributes are naturally amenable to fashion tasks, since garments are described by their materials, fit, and patterns (denim, polka-dotted, tight). They rely on deep neural network trained on fashion images coupled with meta-data. Attributes are used to recognize articles of clothing [5, 6] or describe clothing [22]. Here in Heuritech, we split those attributes into 4 types : clothing category (bag, shirt, pants,...), textures (striped, dots, printed,...), colors (black, blue, rosé,...), and styles (classic, cowboy, vintage,...). However, one weakness is that an attribute vocabulary is defined a priori and manually. Even if it works well with clothing types which are not infinite, this is not really suited to discover new styles.

Similarity search Image retrieval brings an interesting value to fashion as firstly an attempt to cross the bridge between street photos and e-commerce images. In [2], Kiapour et al. learn a neural network similarity function between query images and candidates. Based on convolutional networks, they use a triplet approach with a ranking loss to learn embeddings. They output a matching score which can be used later to find street photos associated to catalogues ones. Last year, Liu et al. released a 800K dataset fashion images for academic research purposes called Deep Fashion [8]. One of those datast is a shop-to-shop dataset. The associated use case would consist of a recommendation system based on similarity between images on a website catalogue. On this paper, they also introduce the neural network model "FashionNet" which learns clothing features by jointly predicting attributes and landmarks.

Trends prediction Another major application of computer vision in fashion is understanding temporal and spatial trends, and try to predict them. This year, with StreetStyle [23], the authors introduce a 27K-dataset of photos of people annotated with clothing attributes and use this dataset to train attribute classifiers. Then they apply clusterization methods based on spatial and temporal data to output some insight. However, those insights are quite basics, such as finding that hijabs are largely presents in Indonesia, a Muslim country, or that people wears more white clothings in summer than in winter. [24] also tackles trends analysis topic with learning attributes classifiers and try to clusterize to find patterns. Their method is slightly different as after learning attributes, they apply a non-negative matrix factorization (NMF) framework to discover style. In a way, this is kind of a topic modelling method to discover topics where words are attributes.

In all those tasks, learning good visual representations fitted for fashion domain is not only essential but crucial. The method proposed in the next chapter uses a weakly supervised model in order to learned it.

Chapter 2

Learning image and text embeddings with weak supervision

One major issue in applied machine learning for fashion is the lack of large clothing e-commerce datasets with a rich, unique and clean labeling. Some very interesting work has been done on collecting datasets for fashion [2, 8]. However, we believe it is very hard to be exhaustive in describing every visual attribute (pieces of clothing, texture, color, shape, *etc.*) in an image. Moreover, even if this labeling work could be perfectly carried, it would come at very high cost, and should be manually done each time we wanted to add a new attribute. A possible source of annotated data is the e-commerce website catalogs. They provide a great amount of product images associated with descriptions, such as the one in Figure 2.1. While this description contains information about the visual concepts in the image, it also comes with a lot of noise that could harm the learning.

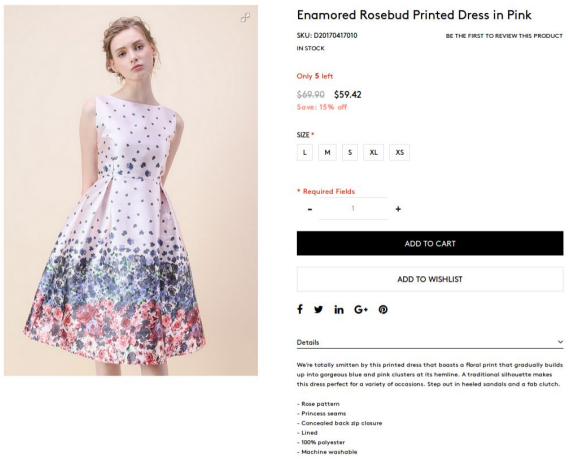


Figure 2.1: My dataset is composed of images and a few associated text descriptors such as their title and their description

We explain now the approach we used to train a visual feature extractor on noisy weakly annotated data.

2.1 Weakly supervised approach

We define weakly supervised learning as a machine learning framework where the model is trained using examples that are only partially annotated, labeled and/or contain a lot of non relevant information.. Learning with noisy labeled training data is not new to the machine learning and computer vision community [25, 26, 27]. Label noise in image classification [28] usually refers to errors in labeling, or to cases where the image does not belong to any of the classes, but mistakenly has one of the corresponding labels. In our setting, in addition to these types of noise, there are some labels in the classes vocabulary that are not relevant to any input. Text descriptions are noisy as they contain common words (*e.g.* 'we', 'present'), subjective words (*e.g.* 'wonderful') or non visual words (*e.g.* 'xl', 'cm'), which are not related to the input image. As we don't have any prior information on which labels are relevant and which are not, we keep the preprocessing of textual data as light as possible.

2.2 Model description

Our work builds upon the one presented in [11], which we explain in this section. The model's training scheme is exposed in Figure 2.2

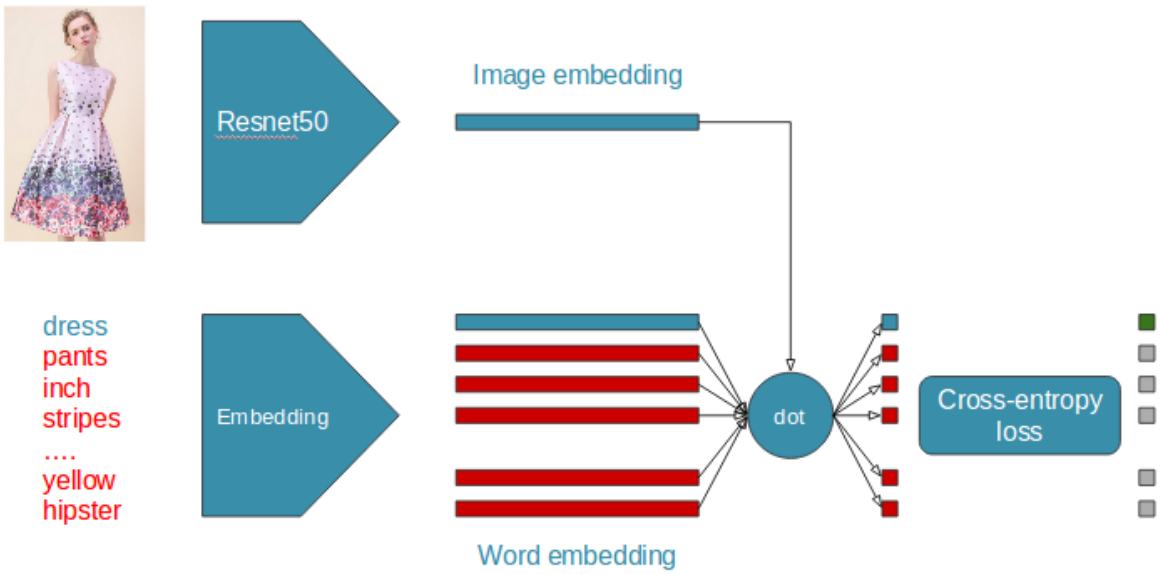


Figure 2.2: Training of our model: predict one label, picked from the bag-of-words description, from an image. Both image and words are embedded before being coupled in a dot product. We output a probability for each word in the vocabulary.

Let $x \in \mathcal{I}$ be an image, and $y \in \{0,1\}^K$ the associated multi-label vector, such that $\forall k \in [1,K], y^k = 1$ if the k-th label of the vocabulary is true for the image x. We use a CNN to compute a visual feature $z = f(x,\theta) \in \mathbb{R}^I$, where θ are the weights of our convolutional neural network. This image embedding is given to a classification layer:

$$\hat{y} = softmax\left(W^T z\right) \tag{2.1}$$

where $W \in \mathbb{R}^{I \times K}$. Note that for all $k \in [1,K]$, the column vector in $w_k = W[:,k]$ corresponds to the embedding of the k-th word in the vocabulary.

2.3 Label imbalance management

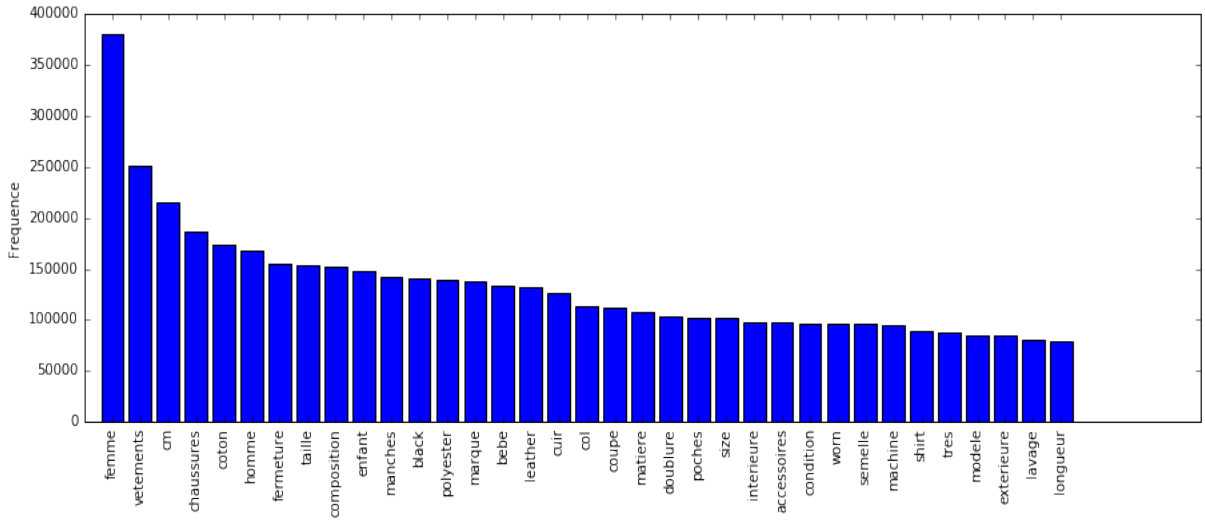


Figure 2.3: Most frequent labels in training dataset.

As seen on Figure 2.3, the distribution of words in our dataset is highly unbalanced. Due to our minimal preprocessing, we observe a high frequency for some non-visual words such as "xl", "cm" or "size" as they appear very frequently in descriptions. Many examples contain those non-visual words that our model would be asked to predict, which is likely to harm the training.

To overcome this issue, and as it was done in [11], we perform *uniform* sampling. Specifically, during training, we sample uniformly a word w from the vocabulary. We then randomly choose an image x whose bag-of-words contains w and we try to predict w given x.

2.4 Loss

As we want to predict one label among a vocabulary K for each image, we use the cross-entropy loss. It minimizes the negative sum of the log-probabilities over all positive labels

$$L(\theta, W, \mathcal{D}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_n^k \log \frac{\exp\left(w_k^T f(x_n, \theta)\right)}{\sum_{i=1}^{K}\exp\left(w_i^T f(x_n, \theta)\right)} \tag{2.2}$$

2.5 Implementation details

Negative sampling

We operate in a context where the vocabulary can be of arbitrary size. Computing probabilities for all those classes for each sample can be very slow. Negative sampling [29] is one way of addressing this problem. After selecting a positive label for an image sample, we randomly draw N_{neg} negative words within the vocabulary. We compute the scores and the softmax only over those chosen words.

Learning

We trained our model with stochastic gradient descent (SGD) on batch of size 20. We consider that an epoch is achieved when the model saw a number of images equivalent to 1/10 of the dataset size, which is approximately 1.3M images in total. After each epoch, we compute a validation error based on a held-out validation set. The initial rate was set to 0.1 and divided by 10 after 10 epochs without improvement. We stop the training after 20 epochs without improvements on our validation dataset. We use the ResNet50 architecture [17] for the visual feature extractor $f(x, \theta)$, with pre-trained weights on ImageNet. Because the last layer has been initialized randomly, we start by learning only the last layer W for 20 epochs, and then we fine-tune the parameters θ in the CNN.

2.6 Training dataset

We built a dataset of about 1,3M images and their associated labels from multiple e-commerce website sources, mostly French, English and Italian. We crawled most of the time one image per product, except when multiple images where available. In that case, we consider them as four different samples with same associated bag of labels.

For each source, we select the relevant fields to keep (title, category name, description,...) and concatenate them. After lower-casing and removing punctuation, we use the RegexpTokenizer provided by NLTK [30] to get a list of words. We remove stopwords, frequent non-relevant words (name of the website, 'collection', 'buy',...) and non alphabetic words. Our final dataset is a list of product images, associated to their respective bag-of-words obtained by the previous preprocessing.

We have deliberately kept preprocessing as minimal as possible, so it is easy to scale to many sources. Thus, we need our model to adapt to this noise in the data. After preprocessing and aggregating the multiple sources, our final vocabulary is composed of 218,536 words. We chose to restrain the vocabulary to the 30,000 most frequent words. The average number of labels per sample is 26,88. We split our dataset into a training and a validation dataset. The validation set is made of the same labels as the training dataset and represent 0.5% of the total size.

Chapter 3

Experiments and transfer learning

After learning the representation on our large weakly annotated dataset, we want to evaluate its quality. To what extent is this representation useful for tasks such as garment classification, attribute tagging or image retrieval ?

3.1 Evaluation datasets

We evaluate our representation on 5 datasets: two public datasets (DeepFashion) used for tagging and image retrieval; three in-house datasets used respectively for category classification, fine-grained classification and image retrieval.

DeepFashion Categories and Attributes Prediction evaluates the performance on clothing category classification, and on attribute prediction (multi-labelling). It contains 289,222 images for 50 clothing categories and 1,000 clothing attributes. While an image can only be affected to one class, it can be associated to multiple labels. The average number of labels for an image is 3.38. For each image in train and test sets, we select a crop available from a ground truth bounding box.

DeepFashion In-Shop Retrieval contains 7,982 clothing items with 52,712 images. 3,997 classes are for training (25,882 images) and 3,985 items are for testing (28,760 images). The test set is composed of a query set and a gallery set, where query set contains 14,218 images of 3,985 items and database set contains 12,612 images of 3,985 items. As in the Categories and Attributes Prediction benchmark, we cropped each image using a ground truth bounding box.

ClothingType We have labeled a dataset with 18 classes, each one corresponding to a garment type (*e.g.* bag, dress, pants, shoes, ...). This in-house dataset contains approximately 736,000 images.

HandBag In addition to the previous dataset, this in-house dataset focuses on bags for fine-grained recognition. Here, the differences between classes are more subtles: bucket bag, doctor bag, duffel bag, etc... It contains 3,060 samples within 13 classes, each one corresponding to a specific type of handbag.

Dress Retrieval This in-house similarity dataset was gathered by crawling an e-commerce website. We collected a list of sets of images, each corresponding to the same item. We used an image classifier to filter out all non-dress items. The final dataset contains 9,009 items for training (20,200 images) and 1,001 items for testing. On this dataset, we keep only images where clothing are worn on humans.

3.2 Label prediction

During training, we evaluate at each epoch precision and recall @1, @5, @10 and @15 on our validation set. In other words, we look at the @k words with the highest probability for an image and check whether those k words are in the true labelling. Those metrics assess the quality of our model in predicting tags for an image on our weakly labeled dataset. In our experiment, precision ranges from 1,84% for precision @20 to 2,98% for precision @1. While these scores may seem low, they are mainly due to the bad quality of our dataset. Remembering our weakly settings, our true labeling from the crawled dataset is sometimes incomplete, redundant and multi-lingual. We show in Figure 3.1 some examples for which our model scores the label "woolen" within its top 15 highest scoring labels. It appears visually that those clothing are indeed made of wool. However, the label was not present in their associated bag-of-words.

As a matter of fact, measuring precision and recall in labeling for our representation isn't suited to quantify whether our learned representation captures precisely both visual and semantic information. Nevertheless, looking manually at predictions for some words can help us get a more precise understanding of the representation.

Manually investigating predictions given by our tagger, we observed two phenomena, related to the learned word representations. The first one is the prediction of non-visual words. In the validation set, our model did not predict words such as 'xl' or 'cm', even though they are very frequents in the training set. It has learned that some words don't correspond to visual concepts, since they appear within a too large visual context. The second phenomenon we observe is what we call *concept cannibalization*. Due to very mini-

Figure 3.1: Most confident predictions on validation dataset for woolen: although the label 'woolen' wasn't present in their associated bag-of-words, we clearly guess on images that those clothing contain wool.

mal preprocessing, our vocabulary is full of synonyms, and of words in different languages that refer to the same visual concept. We first observed that our model never predicts the color tag 'black'. After investigating, it seems that synonyms in other languages, such as 'nere' in italian or 'noir' in french, tend to be predicted and to "cannibalize" the concept. It is still unclear what leads the model to use one tag or the other when they refer to the same visual concept.

3.3 Image retrieval

In this task, given a query image containing an item, we aim at retrieving images that contain the same item. To do so, we compute the score between two images using the cosine similarity between their representation. For a given query image, we sort all gallery images in decreasing order of similarity, and evaluate our retrieval performance using top-k retrieval accuracy, as in [8, 31]. For a given test query image, we give the model a score of 1 if an image of the same item is within the k highest scoring gallery images, 0 else. We adopt this metric for both our image retrieval datasets (DeepFashion In-Shop Retrieval and Dress Retrieval).

In Figure 3.2, we show the results on top-k retrieval accuracy on DeepFashion In-shop Retrieval dataset, for multiple values of k. FashionNet corresponds to the model presented in [8], and HDC+Contrastive is the model in [31]. We denote by [F] (resp. [C]) models that use the full image (resp. an image cropped on the product) to compute retrieval scores. We provide the ImageNet baseline both [F] and [C] models, where we use as feature extractor the penultimate layer of a CNN trained on ImageNet. We would like to emphasize on the fact that our Weakly model, as well as the ImageNet baseline, *do not use the training set* of DeepFashion In-shop Retrieval, unlike HDC+Contrastive and FashionNet.

First, we note that not using bounding boxes for our ImageNet baseline or our Weakly model considerably increase accuracy. Our intuition is that human models in the Deep-

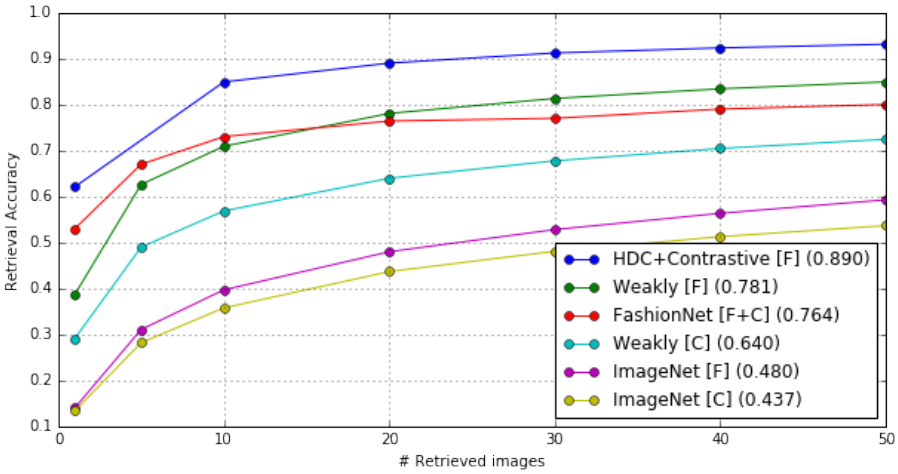


Figure 3.2: Retrieval accuracy for top-k (k=1,5,10,20,30,40,50). We give the top-20 retrieval accuracy between brackets for each model in the caption.

Fashion In-shop dataset often wear the same ensemble of items together, meaning for one shirt item considered for instance, the human model would be wearing the same pants and shoes on all item's image. As a consequence of this bias, it seems easier to evaluate similarity on a ensemble of clothings than on a single clothing on this dataset.

Our Weakly model without crop performs as well as FashionNet, and even outperforms it when $k \geq 20$: considering top-20 retrieval accuracy, it predicts the correct item 78,1% of the time, against 76,4% for FashionNet. Besides, in both the [F] and [C] setups, our Weakly model improves over the ImageNet baseline (from 48% to 78.1% for [F], and from 43.7% to 64.0% for [C]). This validates our hypothesis that our model has learned a specific e-commerce representation. In Figure 3.3, we show an example of a query image, its top-5 similar images according to our weakly learned visual features, and its top-5 similar images according to ImageNet based visual features. As we can see, the similarity encoded by network trained on ImageNet brings together products that are on a same coarse semantic concept, while our representation encodes a more precise and rich closeness, which is based not only the image type, but also on their shape, texture, and fabric. Plus, our representation seems less dependent to human model's pose.

On our in-house dress retrieval dataset, we also observed that the Weakly model improved over ImageNet Model on retrieval accuracy. The Weakly model obtained a top-20 retrieval accuracy of 83,71%, against 65,65% for the ImageNet model. Once again, we point out that we do not perform any training on the retrieval task of this dataset.

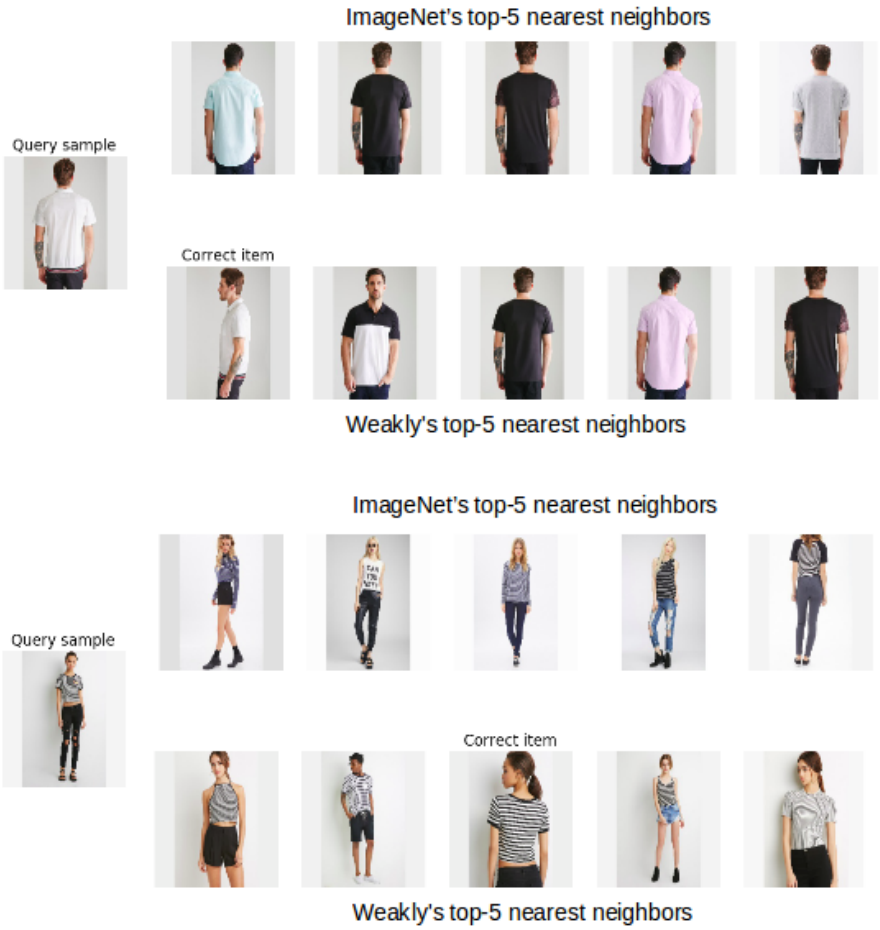


Figure 3.3: Comparison between visual similar images from DeepFashion In-Shop dataset according to our weakly learned visual features and ImageNet based visual features. Our representation seems more robust to human pose (on the top) and successfully captured fine-grained concepts such as stripes (on the bottom).

3.4 Tagging

We conducted multi-class classification and multi-labelling experiments to assess the quality of our visual representation on transfer learning.

On the public DeepFashion Categories dataset, we pre-computed images representation using our Weakly image feature extractor on image crops. Then, we train a simple classifier using a fully-connected layer followed by a softmax activation function. The results are shown on the Table 3.1, at the column Category. With this simple classifier, our results are on par with the state-of-art model by Lu et al. [32].

On DeepFashion Attributes, we train a fully-connected layer with a sigmoid output and

	Category		Texture		Fabric		Shape		Part		Style		All	
	top-3	top-5	top-3	top-5	top-3	top-5	top-3	top-5	top-3	top-5	top-3	top-5	top-3	top-5
WTBI [22]	43.73	66.26	24.21	32.65	25.38	36.06	23.39	31.26	26.31	33.24	49.85	58.68	27.46	35.37
DARN [33]	59.48	79.58	36.15	48.15	36.64	48.52	35.89	46.93	39.17	50.14	66.11	71.36	42.35	51.95
FashionNet [8]	82.58	90.17	37.46	49.52	**39.30**	**49.84**	39.47	48.59	**44.13**	**54.02**	**66.43**	**73.16**	**45.52**	**54.61**
Lu et al.* [32]	**86.72**	92.51	-	-	-	-	-	-	-	-	-	-	-	-
Weakly	86.30	**92.80**	**53.60**	**63.20**	39.10	48.80	**50.10**	**59.50**	38.80	48.90	30.50	38.30	23.10	30.40

*Attribute scores not tested.

Table 3.1: Performance of category classification and attribute prediction on DeepFashion dataset

a binary cross-entropy loss. As we can see in Table 3.1, our model significantly improves over previous state-of-the-art on textures and shape labels top-k recall. However, part and style attributes seem more difficult to separate for our Weakly representation. This might be due to the fact that texture-like and shape-like labels are more represented than part and style words in the large weak dataset. This would require further investigation.

We carried out experiments on our in-house ClothingType dataset where images are annotated according to their clothing category (such as bags, shirt, dress, shoes, *etc.*). Table 3.2 shows the improvement on AUC scores over the ImageNet model for each of the clothing categories using our new representation. This indicates that our training scheme was able to learn discriminative features for garment classification.

	bag	belt	body	bra	coat	combi	dress	eyewear	gloves	hat	neckwear	pants	shoes	shorts	skirt	socks	top	underpants
ImageNet	99.63	98.02	98.20	99.27	99.00	96.01	98.68	99.93	**99.99**	99.45	99.18	99.46	99.87	99.23	98.67	99.35	98.67	99.5
Weakly	**99.86**	**99.46**	**99.08**	**99.67**	**99.31**	**98.11**	**99.13**	**99.99**	99.97	**99.73**	**99.33**	**99.56**	**99.93**	**99.32**	**99.21**	**99.83**	**99.26**	**99.67**

Table 3.2: AUC classification score for clothing categories

	backpack	baguette	bowling bag	bucket bag	doctor bag	duffel bag	hobo bag	luggage	clutch	saddle bag	satchel	tote	trapeze
ImageNet	95.15	87.63	90.42	94.35	90.99	87.97	92.73	**87.65**	96.52	91.77	88.58	96.77	92.11
Weakly	**95.94**	**91.85**	**92.13**	**94.87**	**91.59**	**90.12**	**95.19**	86.96	**97.24**	**93.12**	**91.45**	**97.64**	**93.61**

Table 3.3: AUC classification score for fine-grained type of bags

Finally, we now focus on a fine-grained recognition task. The HandBag dataset contains images annotated with their specific type of bag. In this dataset, the differences between classes are more subtles than in the ClothingType dataset. The training and evaluation are the same as for the previous experiment. As in the previous experiment, we improved AUC scores for nearly each type of bags (see Table 3.3).

3.5 Exploratory visualization using t-SNE

To obtain some insight about our Weakly representation, we applied t-SNE [34] on features extracted using our Weakly feature extractor. We did this for 1,000 randomly chosen images

Figure 3.4: t-SNE map of 1,000 image samples from DeepFashion Categories dataset based on our Weakly image features extractor. We can identify some local subcategories, such as colorful dresses (a), black pants (b), stripes (c), checked (d) or printed shirt (e).

from DeepFashion Categories test set. Figure 3.4 shows full map and some interesting close-ups. On top left (a), we can see a cluster of dresses sub-divised into multiple sub-clusters corresponding to different colors. The cluster (b) shows a focus on black pants. In the zone (c), we can easily see that the model gathered images containing stripes, and it seems like it has separated tops from dresses inside this cluster (with large striped sweaters on top). Checked clothings are grouped in cluster (d), while printed t-shirts are represented in cluster (e). This plot shows that our representation is able to group together concepts that are close in terms of clothing type, texture, color and style.

3.6 Word Semantic features

As said earlier, the matrix $W \in \mathbb{R}^{I \times K}$ can be used as a word embedding table. The characterization of the structure of this word representation space can bring valuable information,

such as detection of related concepts or synonyms. Thus, this section focuses on evaluating the word embeddings resulting from training.

Table 3.4: Results on selected sample from triplet dataset

Word	Positive	Negative	Word2Vec score*	Weakly score*
pants	jodhpur	hat	-0.21	0.14
chino	jodhpur	chemisette	-0.06	0.08
tartan	scozzese	basic	-0.09	0.19

*Score corresponds to $sim(word, positive) - sim(word, negative)$. A good representation should results a positive difference.

To evaluate word similarities we use triplets of words such as ('yellow', 'mustard', 'sneakers'). For each triplet, the first word is expected to be closer to the second that to the third. We use an in-house thesaurus, presented in the form of a hierarchical tree, that contains fashion concepts (clothing categories, colors, textures, styles, ...) to create the dataset, resulting in a 24,428 triplets dataset.

Using our word features extractor, we measure for each triplet whether the word and its positive match are more similar than the word and its negative match. By measuring the error percentage for all samples in the triplet dataset, we can qualify the fitness of our representation for e-commerce. We compare our weakly learned word embedding to a Word2Vec embedding trained on our list of bag-of-words from the preprocessed dataset.

Our text embedding scored 68,98% accuracy against 71,01% for Word2Vec and 50,44% using a random embedding. Note that the label space of our weakly supervised model is only learned using the visual modality. In Word2Vec, word vectors are positioned in the vector space such that words that appear in the same context are close together [29]. As a matter of fact, our visually-learned embedding tends to catch semantic similarites nearly as good as Word2vec. Moreover,as we can see in Table 3.4, when Word2Vec lacks to understand the difference between 'pants' and 'hat', our model is able to separate the two concepts since it has built its representation using visual concepts.

Chapter 4

Further attempts to improve the model

Following our first model, here's below a few attempts made to improve it and try to address the issues raised in the previous chapter.

4.1 Improving pre-processing and pre-trained word embeddings

As stated in the last chapter, our model slightly suffered from concept cannibalization, meaning some words such as 'nere' (black in Italian) wasn't predicted while 'black' itself was predicted too much. When investigating languages of our website sources, we found that, French and English put aside, all the others language were in minority. As a consequence, in order to avoid adding more complexity in our model, we chose to keep only French and English sources.

The vocabulary is also a key parameter in our model. For the next experiments, we increase it to 100,000 words.

On the weakly model, the image features extractor uses both the architecture and the weights of a ResNet50 neural network trained on ImageNet dataset. Meanwhile, the rest of the model has been randomly initialized, including the word embedding matrix. Considering image and word embedding on the same level, there is an unfair quality of initialization. That's why we attempt to pre-trained our word embeddings with a Word2Vec model [29]. Bag-of-words of each image are considered as sentences and we run our word2vec model on it. Finally, only the words presents in the vocabulary are loaded and set in the initial word embbeding. Here, our words are embedded in a 400-vector. We chose that number because this is usually the one chosen in well-known pre-trained word2vec models.

Finally, we don't forget to adapt our image embedding vector, which are 2048-vector from the Resnet50 pool5 output. A fully connected layer of size 400 is then added just

before the dot product. Its weights are initialized using Glorot Uniform [35], such as in the other layers and which ensures a better convergence afterwards. Basically, our text embedding $W \in \mathbb{R}^{I \times K}$ learned in our model is split into a fully dense network $W_1 \in \mathbb{R}^{I \times S}$ and a smaller embedding $W_2 \in \mathbb{R}^{S \times K}$ where S is 400. Our intuition is to catch synonymous by forcing our embedding matrix to have a low rank.

4.2 Intelligent negative sampling

Due to computation time, we do not train on the whole vocabulary at each image but process a negative sampling which have an impact on our training. On our batch, we select N_{neg} negative words and the gradient is back-propagated only on the word embeddings related to the positive and those negatives words. To learn a fine-grained representation, semantic closed words to the positive should be part of those negatives, which is not necessarily the case.

To force it, we improved our negative sampling to include those closed negatives. Each time we chose uniformly a positive word, we look for its top-10 closed neighbors according to the initial word2vec model, where a word is embedded in a 400-vector. From that list, we then sample three out of ten, in order not to brute force our sampling too much. We then complete the batch with randomly selected words as usual.

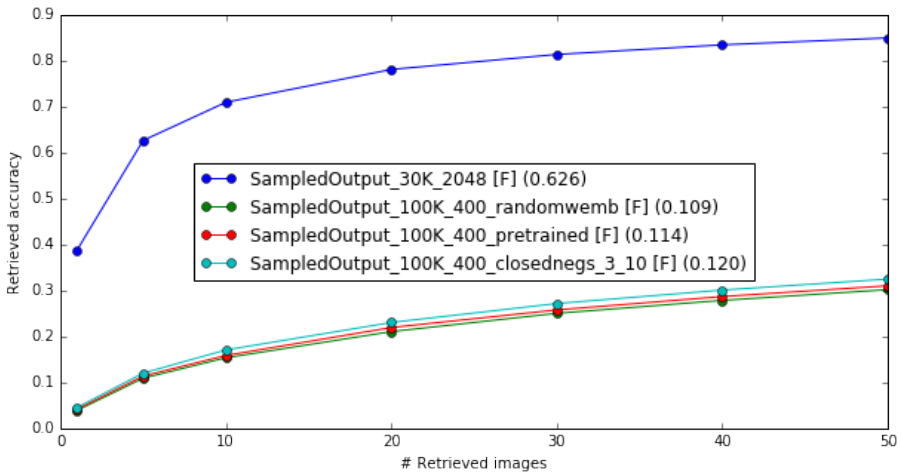


Figure 4.1: Retrieval accuracy on DeepFashion InShop dataset for top-k (k=1,5,10,20,30,40,50). We give the top-20 retrieval accuracy between brackets for each model in the caption.

On Figure 4.1, we plot the results on the image retrieval task for Deep Fashion InShop dataset. Even if we can observe slightly better results with adding closed negatives, those scores are uncommonly low compared to the results for the original model. Putting aside

we had a problem of data lost at some point, one explanation would be the constraining size of the word embedding which may be too small to capture the richness of the vocabulary. Since the first model, we also added more data and filtered out every other languages than English and French. To investigate this issue, we have currently a model training with a 1,000-dimension vectors for word embedding. Another idea would be the increased vocabulary size. As we sample uniformly each word, by augmenting the vocabulary size we decrease the probability for the most common words to be picked up.

4.3 Pseudo-attention model

We believe that working on incorporating structure in the space of labels is a promising path. Not only could it improve our vision system by reducing the synonyms problem, but it could also help us in automate tasks such as the creation and update of a concept ontology or the thesaurus matching.

An image of a clothing contains multiple information such as its color, its shape, its textures... As stated in Chapter 1, although fashion styles may differ depending on time or culture, attribute types are more constant. When using the public dataset Deepfashion [8], authors [24] usually follow the attributes split in categories: colors, fabric, shape, texture and parts. We'd like to capture this structure in attributes and incorporate it in our model.

When applying the dot product between a word embedding and an image embedding, we group all the image information in its vector, without any human-understandable structure in it. Let's take the word 'red' for instance. We'd like to focus only on the color property of the image and not its shape or its clothing type. In a way, we would need to have the part of the vector associated to the color type. In Heuritech, we want to focus on textures, colors, clothing types and style. Our idea is then to force the vector to be structured in four sub-vectors and to apply the dot product on the appropriated part.

From our base model, we add a fully connected layer on top of the output of our Resnet50 feature extractor in order to get a 1,600-vector that will be split in a (4,400)-vector after reshaping. On top of our word embedding, we also add a fully connected layer applied only on the positive word to output a 4-vector that will represent its fashion semantic types. For 'red' for instance, we'll expect this vector to be a close to an one-hot vector (0,0,1,0) where the third position corresponds to colors type. We then applied the dot product between this 4-vector and our reshaped image embedding vector in order to obtain a single 400-vector that will represent the filter properties of the image. This is can be understood as a way to focus on a specific part of a vector, as in an attention model. The weights of this word-fashiontypes layer will also be learned also during training.

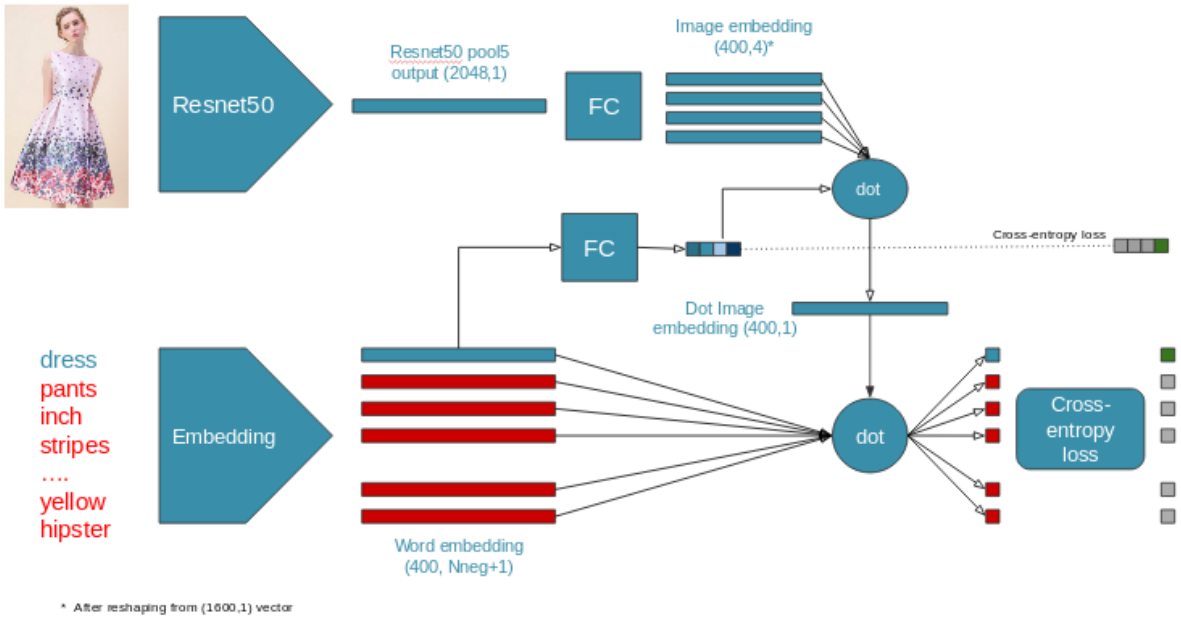


Figure 4.2: Attention weakly model: predict one label picked from the bag-of-words description, from an image. Predict positive word fashion type and applied a dot product to focus only on that part of the image embedding. Finally, couple word and image embedding in a dot product and output a probability for each word in the vocabulary.

Creating ground truth types Our ground truth dataset for fashion types has mainly been constructed on our ontology. This ontology has been manually constructed with words frequently encountered in fashion articles. We chose to enrich it with list of textures [1] and clothings [2] to obtain a bigger ground truth dataset. In total, we have a list of 1,582 words and their associated one-hot vectors, the distribution is available on Table 4.1

	clothings	colors	textures	styles
Words count	618	89	673	202

Table 4.1: Number of words identified per fashion types

On our vocabulary, we still have a majority of words which have not been assigned true fashion types. Most of them are common words which have nothing to do with fashion (e.g. 'wonderful', 'buy') or cannot be assigned to one particular category (e.g. 'size', 'xl'). We consider three options to deal with them :

- either a zero vector,

[1] https://letterpile.com/writing/Describing-Texture-400-words-to-describe-texture
[2] http://www.words-to-use.com/words/clothing/

- either a uniform vector with its components summing to one (meaning here 0.25 for each),
- either adding a entropy term in the total loss for those missing words.

More precisely on the last option, let's called $p = (p_i)_{i=1}^{l}$ the fashion-type probability vector output after the fully connected layer, with l the fashion-type dimension (here four). The entropy term would be written as:

$$H(p) = \sum_{i}^{l} p_i \log p_i \tag{4.1}$$

reaching a minimum for $p_i = 0$ for every i except one where $p_I = 1$. This will force our model to produce a peaky distribution.

Learning We trained our model the same way that the original way. For the fashion type prediction part, we chose to implement a cross-entropy loss in order to reward/penalize probabilities of correct classes only.

When computing scores at each epoch end for validation, we do not have access to the positive words and as a consequence not to the fashion type vector used in the first dot product. Computing every possible words would not been computing efficient. We instead chose to compute five possible cases which are the four one-hot vector and the uniform vector. We then select the top-k among those five vectors of prediction scores.

At the time of writing this report, the attention weakly model is still running. More than obtaining a better image representation for fashion, we also expect to use the fashion type layer to discover new color, texture or style and enrich our thesaurus.

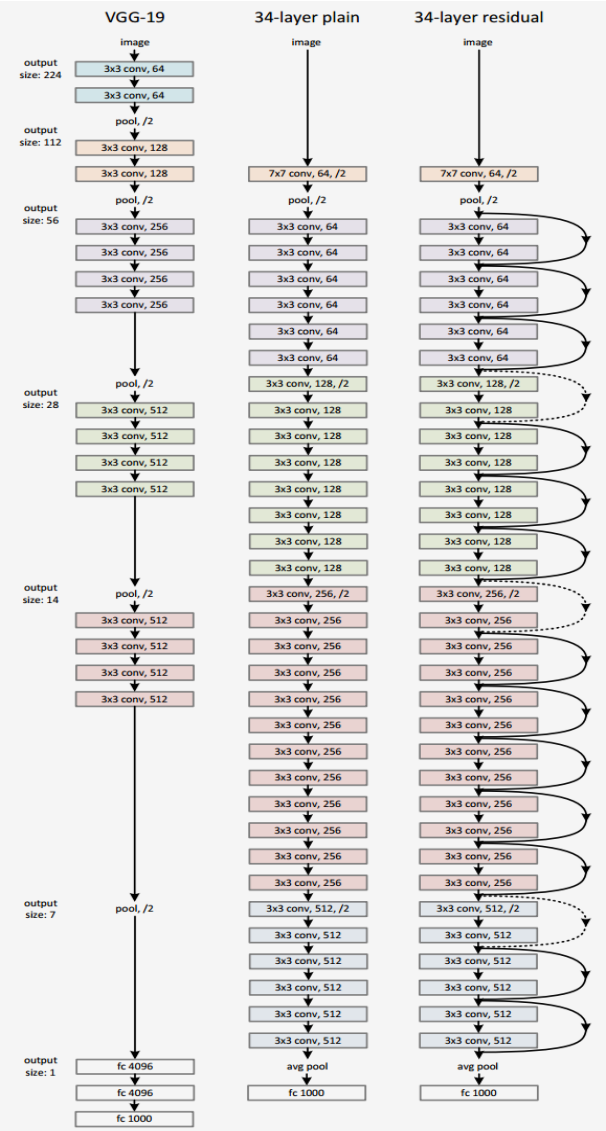
Conclusion

In this internship report, I studied the contribution of deep learning to computer vision for fashion through the prism of learning a good visual representation with weakly supervised learning. In a first chapter, I reviewed why Convolutional Neural Networks can achieve such good accuracy and explained how they are used in fashion applications such as image retrieval or image tagging. In a second chapter, I introduced a method to learn a visual representation adapted to fashion. Based on a weakly supervised framework, this method has the major advantage to overcome the issue of finding a large and clean e-commerce dataset. The results, presented in the third chapter, shows clear improvements compared to a visual representation trained on ImageNet, improving performance on multiple tasks such as image retrieval, classification and fine-grained recognition. I detailed in the last chapter further attempts to improve it.

My closed negatives and pre-trained word embedding intuitions didn't give any promising results yet. At the time I'm writing this conclusion, I am investigating if decreasing too much the size of the embedding could be the reason or if it is due to the increased vocabulary size. The attention model seems promising on the paper for its purpose to constrained our representation into a structure that will reflect the fashion domain. This will need to be confirmed in the experiments still ongoing. I'm hoping to also be able to discover new words for fashion that could enrich our ontology at Heuritech.

Appendix A

Architecture of a ResNet



Bibliography

[1] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking.," in *CVPR*, pp. 1386–1393, IEEE Computer Society, 2014.

[2] M. H. Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg, "Where to buy it: Matching street clothing photos in online shops.," in *ICCV*, pp. 3343–3351, IEEE Computer Society, 2015.

[3] X. Wang, Z. Sun, W. Zhang, Y. Zhou, and Y.-G. Jiang, "Matching user photos to online products with robust deep features.," in *ICMR* (J. R. Kender, J. R. Smith, J. Luo, S. Boll, and W. H. Hsu, eds.), pp. 7–14, ACM, 2016.

[4] D. Shankar, S. Narumanchi, H. A. Ananya, P. Kompalli, and K. Chaudhury, "Deep learning based large scale visual recommendation and search for e-commerce," *CoRR*, vol. abs/1703.02344, 2017.

[5] Y. Kalantidis, L. Kennedy, and L.-J. Li, "Getting the look: Clothing recognition and segmentation for automatic product suggestions in everyday photos," in *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, (New York, NY, USA), pp. 105–112, ACM, 2013.

[6] L. Bossard, M. Dantone, C. Leistner, C. Wengert, T. Quack, and L. Van Gool, "Apparel classification with style," in *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part IV*, ACCV'12, (Berlin, Heidelberg), pp. 321–335, Springer-Verlag, 2013.

[7] W. Di, C. Wah, A. Bhardwaj, R. Piramuthu, and N. Sundaresan, "Style finder: Fine-grained clothing style recognition and retrieval," in *IEEE International Workshop on Mobile Vision*, (Portland, OR), 2013.

[8] Z. Liu, P. Luo, S. Qiu, X. Wang, X. Tang, undefined, undefined, undefined, and undefined, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, pp. 1096–1104, 2016.

[9] A. Bergamo and L. Torresani, "Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach.," in *NIPS* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 181–189, Curran Associates, Inc., 2010.

[10] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *In CVPR*, 2009.

[11] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache, "Learning visual features from large weakly supervised data," *CoRR*, vol. abs/1511.02251, 2015.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (J. Fürnkranz and T. Joachims, eds.), pp. 807–814, Omnipress, 2010.

[13] K. Hornik, "Hornik, k.: Approximation capabilities of multilayer feedforward network. neural networks, 251-257," vol. 4, 01 1991.

[14] S. Sonoda and N. Murata, "Neural network with unbounded activations is universal approximator," *CoRR*, vol. abs/1505.03654, 2015.

[15] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, (Washington, DC, USA), pp. 1150–, IEEE Computer Society, 1999.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, p. 2012.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[20] Y. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *CoRR*, vol. abs/1406.2572, 2014.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[22] H. Chen, A. C. Gallagher, and B. Girod, "Describing clothing by semantic attributes.," in *ECCV (3)* (A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7574 of *Lecture Notes in Computer Science*, pp. 609–623, Springer, 2012.

[23] K. Matzen, K. Bala, and N. Snavely, "Streetstyle: Exploring world-wide clothing styles from millions of photos," *CoRR*, vol. abs/1706.01869, 2017.

[24] Z. Al-Halah, R. Stiefelhagen, and K. Grauman, "Fashion forward: Forecasting visual style in fashion," *CoRR*, vol. abs/1705.06394, 2017.

[25] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 845–869, 2014.

[26] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 1196–1204, Curran Associates, Inc., 2013.

[27] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *CoRR*, vol. abs/1406.2080, 2014.

[28] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification.," in *CVPR*, pp. 2691–2699, IEEE Computer Society, 2015.

[29] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, Curran Associates, Inc., 2013.

[30] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[31] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," *CoRR*, vol. abs/1611.05720, 2016.

[32] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. S. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," *CoRR*, vol. abs/1611.05377, 2016.

[33] J. Huang, R. S. Feris, Q. Chen, and S. Yan, "Cross-domain image retrieval with a dual attribute-aware ranking network," *CoRR*, vol. abs/1505.07922, 2015.

[34] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[35] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

Abstract — In this paper, I present a method to learn a visual representation adapted for e-commerce products. Based on weakly supervised learning, my model learns from noisy datasets crawled on e-commerce website catalogs and does not require any manual labeling. I show that my representation can be used for downward classification tasks over clothing categories with different levels of granularity. I also demonstrate that the learned representation is suitable for image retrieval. I achieve nearly state-of-art results on the DeepFashion In-Shop Clothes Retrieval and Categories Attributes Prediction tasks, *without* using the provided training set.

Key words computer vision, deep learning, fashion, neural networks, visual representation, weakly supervised learning, word embedding.

Ecole polytechnique
Route de Saclay
91128 Palaiseau CEDEX

Université Paris-Saclay
Espace Technologique,
Bat. Discovery - RD 128
91190 Saint-Aubin