Rapport de Stage:

Recommandation de produits vestimentaires à partir de visuels utilisateurs

---

Antoine Habis, Télécom-Paris, M2 Data Science
Début Mai 2020 - Fin Octobre 2020
Supervisé par: Alexandre Duhamel

Remerciements:

Avant de commencer ce rapport, j'aimerais remercier tout particulièrement *Alexandre Duhamel* pour sa patience, ses précieux conseils, sa confiance et sa gentillesse. J'aimerais aussi remercier toute l'équipe Data pour leur accueil et leur bienveillance car ce fut un réel plaisir de travailler avec eux. Un grand merci également à *Clément Olivier* et *Naïma Touloum*, mes collègues dans l'équipe Data Science avec qui j'ai beaucoup aimé travailler et partager des moments. Enfin merci à *Damien Garzili* qui a toujours fait en sorte de garder une bonne cohésion d'équipe et une bonne ambiance de travail.

Sommaire

Tables des figures:

1 Introduction: Présentation de Showroomprive et de l'équipe data:

1.1 Présentation de Showroomprive:

Showroomprive est un site de vente événementiel qui a été fondé en 2006 par Thierry Petit et David Dayans.

Le site internet propose chaque jour entre 15 et 17 ventes événementielles limitées dans le temps sur des articles de grandes marques mais également de marques en devenir.

L'entreprise propose un large choix de catégories de produits mais surtout des vêtements. On peut y trouver de la vente de voyages tout compris, de maquillages, de places de concert, de mobiliers, etc...
Elle permet ainsi à de grandes marques de pouvoir écouler leurs stocks grâce à une plate-forme très visitée et des prix avantageux.

Historiquement, l'entreprise a connu une croissance relativement rapide avec une capitalisation boursière de 655 millions d'euros à la fin de l'année 2015.
Aujourd'hui, l'entreprise est internationale et a ainsi décliné son modèle avec des versions locales de sa plateforme dans 5 autres pays (Espagne, Italie, Belgique, Portugal et Maroc).

Elle comptabilise aujourd'hui près de 950 salariés travaillant à:

- La Plaine Saint-Denis : siège historique (mon lieu de travail)
- Saint-Witz : centre de logistique
- Roubaix : atelier de production
- Vendée : direction des systèmes d'information
- Madrid et Barcelone : filiale espagnole
- Milan : filiale italienne
- Rabat : filiale Marocaine

1.2 Mes débuts en tant que stagiaire en Data Science:

J'ai commencé mon stage chez Showroomprive le 5 mai 2020 suite à un entretien avec mon actuel tuteur *Alexandre Duhamel*, chef de l'équipe Data Science.

Durant cette période, la situation sanitaire m'interdisait de me rendre sur les lieux. J'ai été contraint de commencer mon stage en télétravail.
Ainsi, ma rencontre avec l'ensemble de l'équipe Data s'est faite par l'intermédiaire

de Microsoft Teams. Certes ce n'était pas l'idéal mais cela m'a au moins permis de connaître le nom de l'ensemble de mes collègues dés ma première journée.

Alexandre a très vite fait en sorte que je puisse rencontrer les chefs de chaque pôles de l'équipe data. Que ce soit la partie Business Intelligence, Web Analytics ou la Data Engineer, j'ai pu avoir un bref exposé de ce que chacun des pôles faisait et quels étaient les membres qui le composait.

J'ai également fait la connaissance de *Damien Garzili*, chef de l'ensemble de l'équipe Data. Il m'a très bien accueilli et je me suis rapidement senti à l'aise avec tout le monde même si je ne pouvais pas les rencontrer physiquement.

Pour ce qui est de mon équipe, j'ai pu rencontrer chacune des personnes qui la composait. ils m'ont présenté des projets sur lesquels ils travaillaient.

1.3 Présentation des sous-équipes de l'équipe data:

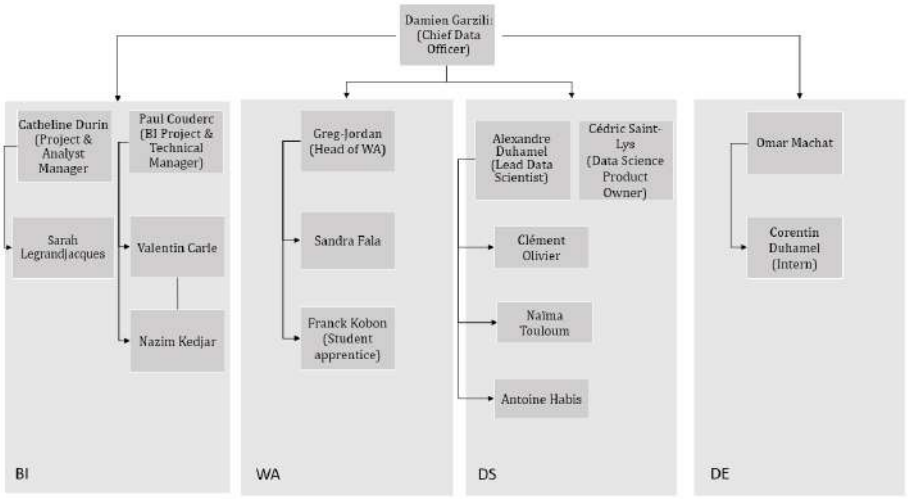

Figure 1: Organigramme de l'équipe Data

1.3.1 La Business Intelligence (BI):

L'équipe BI est composée de 5 personnes:

- *Catheline Durin (Project Analytics Manager)*
- *Sarah Legrandjacques*
- *Valentin Carle*

- *Nazim Kedjar* (Intern)
- *Paul Couderc* (BI Project Technical Manager)

La Business Intelligence s'occupe de l'analyse des données et de la présentation d'informations pour aider les dirigeants ou managers de Showroomprive à prendre des décisions commerciales intelligentes.

Nazim par exemple travaillait sur "l'atterrissage":

Il s'agit d'un outil qui permet de réaliser un pilotage du chiffre d'affaire.
Avec cela on peut permettre aux différents niveaux de décision du service achats de faire une analyse du chiffre d'affaire prévisionnel sous différents angles et d'agir par la suite en dégageant les bonnes décisions en vue de l'atteinte des objectifs fixés.

1.3.2 La Data Engineer (DE):

L'équipe DA est composée de 2 personnes:

- *Corentin Duhamel* (Intern)
- *Omar Machat*

La Data Engineer se charge de développer, de mettre en place et de maintenir les outils et infrastructures adéquats à l'analyse de la donnée. Elle veille à créer une solution permettant le traitement de volumes importants de données tout en garantissant la sécurité de celles-ci.

Les outils principalement utilisés par la DE sont: AWS S3, Redshift ou encore Airflow.

Durant mon stage, j'ai eu à régler à plusieurs reprises des problèmes d'accès à des données ou des répertoires sécurisés. C'est Omar qui s'occupait de m'accorder les droits ou de résoudre le souci auquel j'étais confronté.

1.3.3 La WEB Analytics (WA):

L'équipe WA est composée de 3 personnes:

- *Franck Kobon* (Student apprentice)
- *Sandra Fala*
- *Greg-Jordan Metoui* (Head of Web analytics)

La Web analytics dispose de plusieurs outils permettant entre autre:
d'observer si le site internet de Showroomprivee fonctionne correctement en recensant les clics, d'étudier le comportement des utilisateurs sur le site ou encore de faire des A/B tests pour évaluer l'efficacité de certains ajouts.

1.3.4 La Data Science (DS):

L'équipe DS est composée de 5 personnes:

- *Alexandre Duhamel* (Lead Data Scientist)
- *Naïma Touloum*
- *Clément Olivier*
- *Cédric Saint-Lys* (Data Science Product Owner)
- *Antoine Habis* (Intern)

L'équipe Data-Science travaille sur de nombreux projets différents et variés.

Par exemple, l'un des très gros projets est le Targetor qui consiste à cibler les bons utilisateurs lors de l'envoi de mails promotionnels par la compagnie. Ce projet est ancien et a donc beaucoup évolué au cours du temps, ainsi, presque tous les membres de l'équipe ont pu travailler dessus.

Cependant, certains projets sont aussi réalisés exclusivement par une personne de l'équipe.
Par exemple, *Alexandre* a travaillé sur un projet qui consiste à anonymiser les photos de mannequins dont le contrat d'utilisation de leur image a expiré.
Clément travaille notamment sur un projet de détection de fraude sur le site de Showroomprive avec, entre autre, de la mise en évidence de multi-compte.

2 Prise en main des outils de travail et réalisation d'un premier projet

2.1 Prise en main des outils de travail:

2.1.1 Dataiku Data Science Studio(DSS)

Ayant commencé mon stage en télétravail, j'ai du apprendre à utiliser les différents outils à distance mais ça n'était pas très handicapant.
Alexandre a commencé par m'apprendre à me servir de Dataiku DSS.

DSS de la société Dataiku est un service permettant entre autre :

- D'ordonnancer et de scénariser des flux de codes
- De traiter les données facilement et rapidement grâce à une interface graphique intuitive
- De centraliser le traitement des données sur une unique plate-forme.

Sur la plate-forme, il est possible de créer différents projets.
Chaque membre de l'équipe travaille sur des projets différents parfois seul, parfois en binôme .

Un projet se caractérise par:

- Un flux de données, le "Flow":
  Il permet de retracer l'historique du traiterment de la donnée: de sa phase brute à sa phase finale, lorsqu'elle est prête à être utilisée par la Data-Scientist afin de faire du machine learning.

- Des datasets:
  Ce sont les maillons du flow, ils peuvent être stockés sur différentes plateformes (S3, Redshift, etc..) et sont reliés entre eux par des recettes.

- Des recettes:
  Ce sont des fonctions, elles permettent souvent de passer d'un dataset à un autre. Ces fonctions peuvent être écrites en Python, R, SQL, Julia, Shell, Hadoop et Spark.

- Un scenario:
  Il permet de planifier l'exécution des différentes recettes

Durant mon stage j'ai eu l'occasion de faire 3 projets sur DSS:
le projet A29, le projet A30 et le projet E20 que je détaillerai par la suite.

2.1.2 Amazon S3

Amazon S3 est un service de stockage d'objets non structurés. Il permet notamment de stocker des datasets ou encore des notebooks ou même des images. S3 est organisé selon des "buckets".

Ce sont de gros volumes de stockage qui peuvent accueillir autant de fichiers que l'on veut. A l'intérieur d'un bucket il est possible de faire des dossiers et de les organiser comme bon nous semble selon une arborescence.

2.1.3 Amazon EC2

Amazon EC2 est un service qui permet d'allouer des machines à Amazon.
Il est possible d'allouer des machines différentes qui ont chacunes des propriétés qui leur sont propres (certaines ont plus ou moins de RAM, certaines sont des GPU, etc..).Il est également possible d'attacher plus de mémoire aux machines si l'on le souhaite.

Durant mon stage j'ai beaucoup utilisé des p3.2xlarge. Ce sont de gros GPU qui permettent de faire du deep learning sur des datasets importants.

Pour se qui est de l'allocation, il est possible de demander des machines non utilisées de manière temporaire. Le prix au comptant de chaque type d'instance dans chaque zone de disponibilité est fixé par Amazon EC2, et est ajusté progressivement en fonction de l'offre et de la demande à long terme des instances selon le principe d'enchère. Ce genre de demande se nomment les "Spots Requests".

2.1.4 Putty

Putty est un outil permettant simplement de se connecter en ssh aux machines à l'aide de l'adresse Public IPv4 DNS générée par Amazon .
Putty permet donc d'ouvrir une console sur laquelle l'utilisateur peut entre autre: exécuter des commandes pour importer des données de S3, faires des commandes linux usuelles ou encore lancer un notebook jupyter par exemple.

2.2 Projet initiatique: Détection de doublons

Lors de mon arrivée, après m'être entraîne à utiliser DSS, *Alexandre* m'a confié un projet initiatique utilisant Tensorflow[6] et Keras[7] avant de commencer le projet autour duquel mon stage avait été conçu.

2.2.1 Sujet et contexte:

Showroomprive possède une base de donnée de produits et chaque produit possède un identifiant produit qui lui est associé ainsi que 5 aperçus.

Cependant deux produits identiques peuvent avoir des identifiants différents, c'est ce que l'on nomme des doublons.

Figure 2: Exemple de doublon

Ce genre de doublons peut être généré lorsqu'un produit apparaît dans 2 ventes

différentes qui ont lieu à deux périodes distinctes. Il sera alors référencé de deux manières différentes dans les datasets et donc apparaîtra comme un doublon.

2.2.2 Objectif:

Le but de ce projet est de trouver un moyen rapide de détecter les doublons par l'image.

Toutefois, il est possible que les images des doublons ne soient pas exactement les mêmes, parfois à cause d'un décalage de pixel ou à cause de photos qui ne sont pas exactement les mêmes, d'angles différents, de retouches différentes, etc..

Ce projet me sera aussi utile par la suite car il me permet d'obtenir un dataset avec beaucoup d'images de produits sans doublons. Il y a donc aucune chance que des mêmes images se retrouvent à la fois dans le *training set* et le *validation set* du modèle pour le second projet.

2.2.3 Solution envisagée:

Bien sûr, la solution envisagée n'est pas de comparer les images pixel par pixel car beaucoup trop coûteux. J'ai donc commencé par proposer une solution afin de résoudre le problème en utilisant des réseaux siamois. Cependant la solution était assez longue à implémenter pour un simple projet initiatique.

La solution qui fut vite envisagée était d'utiliser un encodeur pré-entraîné afin de comparer les distances entre les images. L'encodeur pré-entraîné permet de récupérer l'aspect sémantique de l'image en détail et a le mérite d'être prêt à l'emploi.

Les prédictions de doublons étant les produits dont la distance des deux encodages était inférieure à un certain seuil.

2.2.4 Récupération des données:

Pour commencer il a fallu récupérer un dataset de l'ensemble des produits de 'Prêt à Porter' grâce à DSS. (Voir Flux: Figure 3)

Figure 3: Flux de récupération et traitement du dataset

Le dataset comporte:

- l'identifiant des produits
- l'identifiant des ventes
- une liste de 5 booléens pour indiquer la présence ou l'absence des 5 aperçus des produits
- la catégorie des produits

2.2.5 Évolution des approches de traitement du sujet:

Première approche:

La première approche était la plus intuitive mais aussi la moins optimisée. J'ai utilisé l'encodeur VGG16[10] pré-entraîné sur ImageNet[14] afin d'encoder les images dans un vecteur de taille 4096.

La seconde étape était de comparer toutes les images entres elles.

Cependant avec un nombre d'images $n_{tot} = 120000$, la complexité *comp* vaut:

$$\begin{aligned} comp &= \frac{n_{tot}(n_{tot}-1)}{2} \\ &= 7.2 \times 10^9 \end{aligned} \tag{1}$$

Le nombre d'opérations est démesuré ce qui engendre un temps d'exécution très long. Il a donc fallu changer de méthode.

Deuxième approche:

La deuxième approche cherche donc à limiter le nombre de comparaisons et consiste ainsi à ne comparer que les images dont les types sont identiques. En effet rien ne sert de comparer un manteau avec un t-shirt, on sait d'avance que ce ne sont pas des doublons.

Il s'agit donc de parcourir tous les produits un à un puis calculer les distances du produit à ceux qui ont un label identique puis d'enlever les doublons peu à peu.
Pour calculer la complexité prenons l'exemple d'une catégorie de vêtement: les t-shirts verts

Soit n_0 le nombre initial de t-shirts verts.
Pour le premier t-shirt vert il y a $c_0 = n_0 - 1$ comparaisons à faire.
On introduit maintenant p_k la probabilité que le $k^{ème}$ t-shirt vert choisi ait un doublon dans le dataset restant.

- Avec une probabilité p_0 un doublon sera éliminé et il y aura donc en moyenne plus que $n_0 - 3$ t-shirts verts (car les produits ont 3 visuels en moyenne sur les 5) et donc $c_1 = c_0 - 3 - 1$ comparaisons à faire pour le prochain t-shirt.
- Avec une probabilité $1 - p_0$ il restera à l'inverse $c_1 = c_0 - 1$ comparaisons à faire.

On trouve donc une formule de récurrence pour la suite $(c)_{k \in N}$ de variables aléatoires:

$$c_0 = n_0 - 1$$

$$c_{k+1} = \begin{cases} c_k - 1 - 3 & \text{avec probabilité } p_k \\ c_k - 1 & \text{avec probabilité } 1 - p_k \end{cases} \tag{2}$$

Il en découle que:

$$\forall k \in \mathrm{N}^*,\ E[c_k] = -3p_k + E[c_{k-1}]$$
$$\forall k \in \mathrm{N},\ E[c_n] = n_0 - 1 - 3\sum_{i=1}^{n} p_i$$

Ainsi on en déduit la complexité moyenne:

$$comp = E[\sum_k c_k 1_{c_k > 0}] \tag{3}$$

Cette méthode est plus rapide certes mais encore beaucoup trop lente pour une implémentation. Avec un temps estimé de 25h pour 120.000 images.

Troisième approche:

La troisième approche est de former des petits groupes d'images (de taille 200 approximativement) de manière aléatoire puis d'éliminer les doublons dans

chacun des petits groupes puis de fusionner les petits groupes sans doublon 2 à 2 et ré-appliquer la déboulonnage ainsi de suite.
Cependant cette approche est toujours très lente avec un temps d'exécution proche de 20H pour 120.000 images.

La complexité moyenne est difficile à calculer théoriquement mais comme le nombre de groupes est divisé par 2 à chaque itération on s'attend à ce qu'elle s'exprime comme une somme de termes au carré avec un nombre de terme en $n_{iter} = \left\lfloor log(\frac{n_{tot}}{200log(2)}) \right\rfloor$.

En prenant $n_{tot} = 120000$ images, la complexité maximale (dans le pire des cas) s'exprime comme suit:

$$\begin{aligned} comp &= \left\lfloor \frac{n_{tot}}{200} \right\rfloor \frac{200(200-1)}{2} + \sum_{k=1}^{n_{iter}} \left\lfloor \frac{n_{tot}}{200 \times 2^k} \right\rfloor \frac{200 \times 2^{k-1}(200 \times 2^{k-1} - 1)}{2} \\ &= 3.7 \times 10^8 \end{aligned} \tag{4}$$

Quatrième approche:

La quatrième et dernière approche est de loin la plus optimisée. Cette approche consiste à encoder l'ensemble des images (en vecteur de taille 4096) puis à regrouper les encodages intelligemment (et non plus aléatoirement comme dans l'approche 3) dans des clusters de petites tailles.

Tout d'abord, il faut extraire des features intéressantes de l'encodage. J'ai choisi d'utiliser:

- le max
- le min
- la somme
- les 6 derniers déciles

Grâce à ces 9 nouvelles features, on peut appliquer un algorithme de k-means afin de regrouper intelligemment les images entres elles avant de les comparer.

En faisant ceci, on réduit considérablement le temps de calcul. Cette dernière approche permet d'arriver à seulement environ 30 mins d'execution pour 120.000 images.

Si l'on se concentre uniquement sur le nombre de comparaisons effectué, avec des clusters de taille 200 environ on obtient une complexité *comp* qui s'exprime comme suit:

$$comp = \left\lfloor \frac{n_{tot}}{200} \right\rfloor \frac{200(200-1)}{2} \\ = 1.2 \times 10^7 \tag{5}$$

Détermination du seuil idéal:

Pour comparer les images, j'ai utilisé une simple distance MSE (*mean square error*) sur les encodages. Ensuite, il faut déterminer un hyper-paramètre crucial pour la détection de doublon: la valeur du seuil s

Comme le but du projet était de faire un un algorithme non supervisé pour la détection de doublon j'ai du déterminer le seuil idéal empiriquement.

Figure 4: Distance entre deux vrais doublons

Figure 5: Distance entre deux faux doublons

Seulement, la détection de doublons se fait entre deux images de produits différents et qu'un produit comporte souvent 5 visuels, j'ai choisi un seuil s assez bas car si un doublon n'était pas détécté sur un visuel d'un produit, il le

serait sûrement sur les 4 restants.

Vers une mise en production de la détection de doublons?

Lorsque le projet s'est terminé, *Damien Garzili* a pu m'organiser un échange avec *Stéphane Bonnard* (Coordinateurs Projets et Projets Stratégiques) afin d'expliquer mon projet.
Stéphane était déjà en charge d'un projet d'uniformisation de références produits (Projet PIM) et a donc proposé de faire un test sur un dataset de chaussures avec environ 450 produits.

Ce dataset avait déjà été testé au préalable par sa méthode et était donc censé ne plus contenir de doublons.
Une fois mon algorithme de détection de doublons appliqué au dataset j'avais détecté 2.2% doublons qui ont été validés par *Stéphane Bonnard* après vérification.(voir Figure 6)

Figure 6: Mail récapitulatif de la liste de doublons détectés dans le dataset de chaussures

Aujourd'hui, le projet reste fini et est en cours de mise en production.

3 Projet Street 2 Shop (A30)

3.1 Sujet et Contexte:

Le projet Street 2 Shop est le projet principal de mon stage.
Ce projet consiste à construire un algorithme de recommandation de produits Showroomprive à partir d'une image envoyée par un utilisateur d'un vêtement porté ou non.

La recommandation doit être basée sur une ressemblance entre le produit envoyé par l'utilisateur et ceux présents sur le site de Showroomprive.

L'utilisation du projet sur le site web doit se présenter comme une barre de recherche sur le site dans lequel l'utilisateur peut déposer une image ou se prendre en photo directement depuis sa pellicule et le modèle propose une recommandation.

Ce projet a débuté environ un mois après mon arrivée à Showroomprive avec une partie recherche. Le but était d'étudier les différentes façons de faire de la recommandation par l'image de vêtement.

3.2 État de l'art

Cette partie a duré environ 2 semaines durant lesquelles j'ai cherché tout ce qui avait déjà été fait sur le sujet. Après avoir fait un tri sur les méthodes trouvées, 3 papiers ont particulièrement retenu mon attention.

La particularité commune à ces 3 articles est d'aller plus loin que le simple traitement objectif et très descriptif du vêtement : la notion de style est introduite, elle me semble être primordiale pour de la recommandation visuelle.
Les paragraphes qui suivent décrivent brièvement ces 3 manières d'aborder le problème.

3.2.1 Towards better understading the clothing fashion Styles: A Multimodal Deep Learning Approach [1]

Cet article est très intéressant de part son approche assez innovante et originale. Il propose une nouvelle manière de prédire le style d'un vêtement, ce que l'on nommera par la suite son "esthétique".

Dans tout le reste du rapport, nous distinguerons deux notions importantes:

Définition 3.1. (Les catégories auxquelles un produit appartient):
Ce sont des descriptifs visuels objectifs du vêtement comme par exemple le col, la couleur, le type de vêtement, la matière, etc..

Définition 3.2. (L'esthétique du produit):
C'est une description plutôt subjective du produit lié à son style, comme par exemple: moderne, élégant, chic, dynamique,etc..

L'idée du papier part du postulat suivant:

Postulat: Lorsqu'un mannequin est pris en photo pour un shooting, les produits qu'il porte (le haut comme le bas) ont de styles semblables.

Une fois ce postulat admis, le modèle proposé est le suivant:

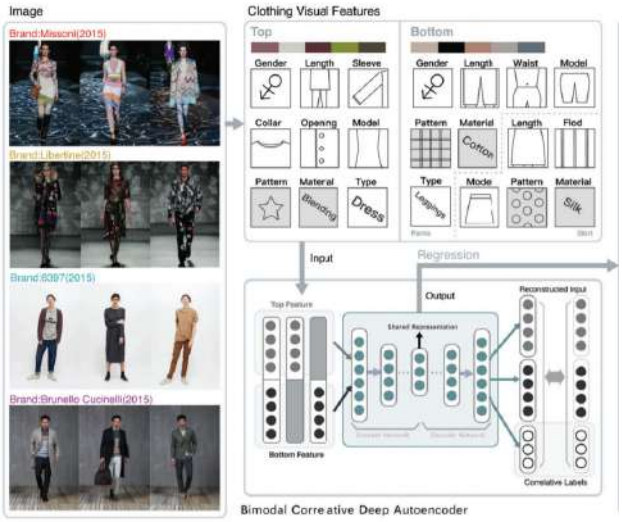

Figure 7: Représentation du modèle

Description du modèle pour l'entraînement:

1. Un modèle de détection permettant d'identifier le haut et le bas d'une tenue
2. Un modèle multilabel cherchant à prédire certaines catégories comme:
   - le type de col
   - la couleur
   - la taille des manches
   - le genre
   - etc..

Ce modèle est entraîné à part et sert à renvoyer uniquement l'encodage des images du haut et du bas.

3. Un modèle Bimodal Correlative Deep Autoencoder:

   Ce modèle prend en input l'encodage en sortie du second modèle du haut et du bas concaténé et, à la manière d'un auto encodeur, il cherche à réduire la dimension en mêlant l'information du haut et du bas en un vecteur que les auteurs appellent "Shared Representation" .
   Puis il reconstruit les encodages et prédit les type de vêtements par ailleurs (les correlatives labels).

Description du modèle pour la prédiction:

Ce modèle prend en input le haut et le bas d'un mannequin. Comment faire alors pour évaluer le style d'un unique produit?
Le papier propose simplement de mettre en entrée la représentation encodée du vêtement et de compléter le reste par un vecteur nul.

Ce modèle permet ainsi d'obtenir l'encodage esthétique d'un vêtement.
En l'associant à son encodage catégoriel obtenu en sortie du second modèle, il est possible de faire une recommandation de produit en concaténant les deux représentations et en introduisant une distance dans l'espace de représentation du produit.

Évaluation du modèle

Pour ce qui est de l'évaluation du modèle, les auteurs ont eu une idée assez astucieuse:

Ils ont listé 527 mots décrivant des esthétiques de vêtements et ont récupéré leurs représentations vectorielles faites par WordNet.
Ils ont ensuite fait une régression linéaire depuis l'espace des "Shared représentation" vers celui des représentations vectorielles sur Wordnet, qu'ils ont ensuite pu afficher sur un plan. (voir Figure 8).

De cette manière, ils ont pu vérifier que leur modèle fonctionnait correctement en observant des clusters de marques sur le plan.

3.2.2 Image-based Recommendations on Styles and Substitutes [2]

Encore une fois, ici l'approche est très originale car elle mêle deux méthodes différentes pour la recommandation:

1. La ressemblance des produits basée sur l'image

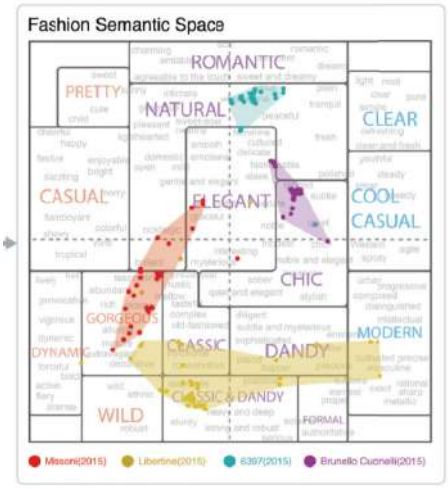

Figure 8: Plan de représentation des styles

2. L'appétence des clients pour les produits sur le site du vendeur

Postulat:
Le papier part du postulat suivant: Il existe 4 relations entre les produits lors de la visite d'un utilisateur sur le site:

1. Un utlisateur regarde le produit X puis regarde le produit Y
2. Un utilisateur regarde le produit X puis achète le produit Y
3. Un utilisateur qui a acheté dans le passé le produit X achète le produit Y
4. Un utilisateur achète simultanément X et Y

Les relations 1 et 2 indiquent que X et Y sont des produits de substitutions.
Les relations 3 et 4 indiquent que X et Y sont des produits complémentaires et donc dans le cas des vêtements impliquent qu'ils sont esthétiquement proches.

Présentation du raisonnement:
Pour la suite, nous dirons donc que X et Y sont reliés si ils satisfont la relation 3 ou 4. On définit alors la relation r par:

$$r_{ij} = \begin{cases} 1 & \text{si } X \textit{ et } Y \textit{ sont reliés} \\ 0 & \text{sinon.} \end{cases}$$

Le papier propose d'encoder les images des produits grâce à un modèle catégoriel (qui prédit les catégories auxquelles le produit appartient).

L'encodage du i^{eme} produit est alors noté $x_i \in R^F$\{ F: le "Feature Space"

L'idée importante est de construire une distance intelligente entre les encodages des produits grâce à la relation r.

L'auteur détaille dans le papier le fil de sa réflexion et finit par aboutir à une solution pour cette distance. Il introduit alors Y un matrice de projection $Y \in \mathcal{M}_{KF}$ *avec* $K < F$ de telle sorte que:

$$d_Y(x_i, x_j) = ||(x_i - x_j)Y||_2^2 \tag{6}$$

R^K désigne l'espace esthétique, et $s_i = x_i Y$, la projection de l'encodage du produit x_i dans l'espace esthétique R^K.
Ensuite, il faut définir une probabilité P qui découle de la distance d qui traduit la probabilité que deux produits i et j soient en relation:

$\forall i, j \in N^{n_{product}} \times N^{n_{product}}$,

$$\begin{aligned} P(r_{i,j} \in R) &= \sigma_c(-d_Y(x_i, x_j)) \\ &= \frac{1}{1 + e^{d(x_i, x_j) - c}} \end{aligned} \tag{7}$$

$$\tag{8}$$

avec R, l'espace des relations entre produits et $Q = \{r_{ij} | r_{ij} \notin R\}$

Il s'agit d'une sigmoid shiftée d'un paramètre c de telle sorte que lorsque la distance entre deux produits est inférieure à c, la probabilité qu'ils soient reliés est supérieure à 0.5.

Nous avons donc 2 paramètres:

- le seuillage c
- la matrice de projection Y

Il suffit maintenant d'optimiser ces paramètres. Avec l'introduction de la loi de probabilité dans l'équation (7) pour r, il suffit de minimiser la "log likelihood" associée.

$$l(Y, c | R, Q) = \sum_{r_{i,j} \in R} \log(\sigma_c(-d_Y(x_i, x_j))) + \sum_{r_{i,j} \in Q} \log(1 - \sigma_c(-d_Y(x_i, x_j))) \tag{9}$$

Une fois la minimisation terminée, cette méthode nous permet d'obtenir la matrice Y de projection dans l'espace esthétique.
Nous pouvons donc pour n'importe quel produit obtenir son vecteur esthétique et catégoriel.

La recommandation découle ainsi par l'introduction d'un distance appropriée entre les vecteurs de chaque produit.

3.2.3 Aesthetic-based Clothing recommendation [3]

Ce dernier papier est celui que nous avons choisi communément avec mon tuteur *Alexandre Duhamel*.

L'auteur propose de traiter les vêtements afin de leur extraire deux grands types d'information:

- Ésthétique
- Catégorielle

Les termes Esthétique et Catégoriel font références à ceux définis plus tôt dans (3.1) et (3.2).

Présentation du modèle

Le modèle proposé par l'auteur est découpé en deux taches: un modèle catégoriel et un modèle esthétique.

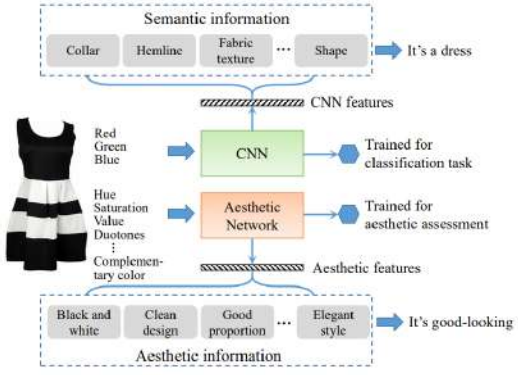

Figure 9: Présentation de la pipeline

Le premier modèle catégoriel est prévu pour faire de la classification multi-classe, multi-label tandis que le deuxième est un modèle de classification couplé

avec une régression évaluant avec une note entre 1 et 5 l'esthétique du vêtement.

Pour le modèle catégoriel, aucune information n'est donnée sur l'architecture mais il s'agit de la même tâche que celle effectuée dans les papiers précédents, à savoir de la classification sur le vêtement.

Pour ce qui est du modèle esthétique, les auteurs ont choisi d'implémenter un réseau de neurones avec une architecture plus spécifique (voir Figure 10):

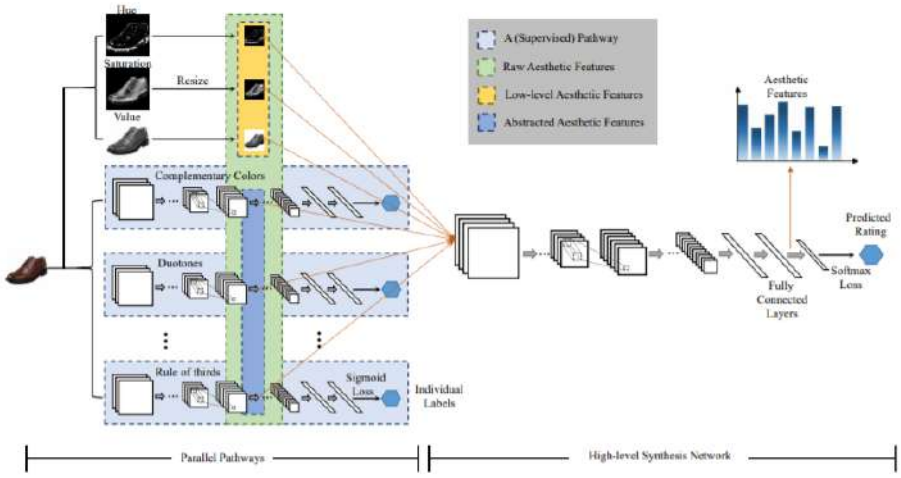

Figure 10: Architecture du modèle esthétique

Les données (le dataset AVA):

Le modèle est entraîné sur un dataset de photographies de tout type et non sur des vêtements.

Chacune des photographies est annotée avec 14 labels censés décrire l'esthétique de la photo comme par exemple:

- La présence de couleurs complémentaires
- La présence d'un point de fuite
- Si la photo est bichromatique
- Si la photo est très saturée
- etc..

En plus des annotations, chaque photo dispose d'une répartition de votes sur des notes de 1 à 5 données pour évaluer l'esthétique.

Les auteurs justifient l'utilisation d'un dataset sur des photographies avec des labels qualifiant l'esthétique visuelle de ces-dernières de la manière suivante:

"Il n'y a pas de doute quant aux similarités entre l'esthétique photo et vestimentaire, les deux émanent de facteurs communs comme la combinaisons de couleurs, saturation, luminosité, etc.." (Voir Figure 11)

Description du modèle esthétique:

La première partie du modèle, le *"Parallel pathways"* (voir Figure 10) consiste à extraire la teinte, la saturation et la valeur. Ensuite, chaque label est traité indépendemment des autres dans une pipeline et aboutit à une prédiction sur la présence ou non du label.

Une fois cette partie entraînée, la deuxième partie, le *"High level Synthesis Network"* (Voir Figure 10) consiste à assembler les encodages de chacun des labels afin de faire une prédiction sur la note esthétique.

Encore une fois, ce modèle permet de récupérer l'encodage catégoriel d'un produit et l'encodage esthétique grâce à la première partie du modèle (Voir Figure 10). Avec ceci on peut construire un système de recommandation vestimentaire.

5.4 Rationality of using the *AVA* dataset (RQ3)

The BDN is trained on the *AVA* dataset, which contains photographic works labeled with aesthetic ratings, textual tags, and photographic styles. We utilize aesthetic ratings and photographic styles to train the aesthetic network. In this subsection, we simply discuss if it is reasonable to estimate clothing by the features trained for photographic assessment.

With no doubt that there are many similarities between esthetical photographs and well-designed clothing, like delightful color combinations, saturation, brightness, structures, proportion, etc. Of course, there are also many differences. To address this gap, we modify the BDN. In [43], there are 14 pathways to captures all photographic styles. In this paper, we remove several pathways for the photographic styles which contribute little in clothing estimation, like high dynamic range, long exposure, macro, motion blur, shallow DOF, and soft focus. These features mainly describe the camera parameters setting or photography skills but not the image, so they help little in our clothing aesthetic assessment task. Experiments show that our proposed model can uncover consumers' aesthetic preference and recommend the clothing that are in line with their aesthetics, and the performance is obviously promoted.

Figure 11: Justification de l'utilisation du Dataset AVA

3.3 Les données Showroomprivee

3.3.1 Description des données:

Les données de Showroomprive sur lesquelles j'ai travaillé étaient toutes stockées dans des buckets S3.

Pour ce qui est des images, celles-ci étaient rangées dans des dossiers à l'intérieur d'un grand bucket: "srp-media-archive".

Pour retrouver l'image dans le bucket, il faut connaître 3 caractéristiques:

1. Le numéro de la vente
2. l'identifiant du produit
3. le numéro de l'aperçu (chaque produit a au maximum 5 aperçus)

Très vite, j'ai réalisé que seul les deux premiers aperçus de chaque produit seraient utiles pour construire les modèles. En effet, les deux premiers aperçus sont souvent une vision globale du produit tandis que le reste des images peuvent être des photos de dos ou des zooms sur le produit .

Pour ce qui est de la labelisation, il existait déjà avant mon arrivée un dataset qui répertoriait pour chaque identifiant produit:

- L'identifiant de la coupe du produit

- L'identifiant de la matière
- L'identifiant des couleurs
- Le sexe

coloris	coupes	matière	sexe	id_prod
string	string	string	string	bigint
Decimal (com...	Integer	Decimal (com...	Integer	Integer
3 4		4	5	4220201
4		14 4	1	5802301
9 17			2	5749228
163		14 4	5	6037110
18		4 8	1	4551014
4		20	2	5417274
3 4		4	5	6559017
3		4	3	6328917
4		20	3	6381251
4		4	2	4718733

Figure 12: Dataset déjà existant sur les produits Showroomprive

Il existe aussi un autre dataset avec l'ensemble des produits sur le site de Showroomprive avec beaucoup de features dont le type des produits (manteaux, t-shirts, chemises, etc..)

3.3.2 Traitement des données:

Toute la partie traitement des données en pre-processing s'est faite sur Dataiku DSS. J'ai réalisé un projet sur la plateforme afin d'extraire les données des datasets S3 et de les réorganiser comme je le voulais.

Figure 13: Flux du projet Street 2 Shop

Tout d'abord, Il faut savoir que Showroomprive ne travaille pas que sur des vêtements. IL existe aussi pleins d'autres types de produits comme des portefeuilles, des montres, des objets pour la maison,etc... J'ai donc commencé par récupérer uniquement les produits de prêt à porter.

Ensuite il a fallu faire un traitement sur les types de produits car certains étaient trop précis et donc sous représentés. J'ai donc regroupé beaucoup de produits ensemble sous des labels plus généraux.
Par exemple, les blousons, trenchs, perfectos vont être regroupés derrière un unique label: vestes/manteaux.

Enfin après quelques traitements additionnels comme la jointure des 2 datasets, on obtient le dataset final qui contient toutes les informations dont j'ai besoin pour commencer.

- L'identifiant produit
- Les aperçus
- L'identifiant matière
- L'identifiant couleur
- Le type de vêtement
- L'identifiant du sexe

3.3.3 Analyse des données:

Showroomprive possède une très grande base de donnée d'images.
En sélectionnant seulement une partie des produits de prêt à porter j'avais déjà plus de 200 000 images à disposition.

Sur les images, la grande majorité des produits sont portés par des mannequins mais certains produits sont aussi non portés.
Les images des produits se présentent souvent de la même manière avec le mannequin au centre devant un fond uni.

Cette uniformité des images est un problème pour entraîner notre modèle car le but est de faire de la recommandation à partir d'une image prise en photo par un utilisateur quelconque. Il faut donc pouvoir entraîner notre modèle sur des données type Showroomprive mais aussi des données utilisateurs que Showroomprive ne possède pas.
J'ai réussi à pallier à ce problème par la suite vers la fin de la période de stage en faisant du *scraping* sur un site de vente de vêtements de particulier à particulier mais j'évoquerai cela un peu plus tard.

Répartition des données dans les classes

Que ce soit pour la couleur, la matière, le type de produit ou le sexe, les données sont assez mal réparties dans les classes.
Pour ce qui est du type de produits, Il y a beaucoup plus de robes et de t-shirts dans la base de donnée que de doudounes par exemple. De la même manière il y a beaucoup plus de produits pour femme que de produits pour bébé et plus de produits en coton que de produits en polyesther.

Pour palier à ce problème, j'ai effectué un *upsampling* des classes minoritaires et un *downsampling* des classes majoritaires afin d'équilibrer un peu le dataset. L'*upsampling* d'une classe consiste à tirer des représentants de cette même classe selon une loi de décision fixée afin de les rajouter au dataset pour enrichir la classe.
Mes critères pour re-sampler mes données étaient seulement sur l'*upsampling*. Je me suis fixé comme limite de ne pas multiplier par plus 2.5 les classes sous représentées et cela en utilisant une loi de décision aléatoire.La Figure 14 représente l'évolution de la répartition dans les classes suite au *sampling*.

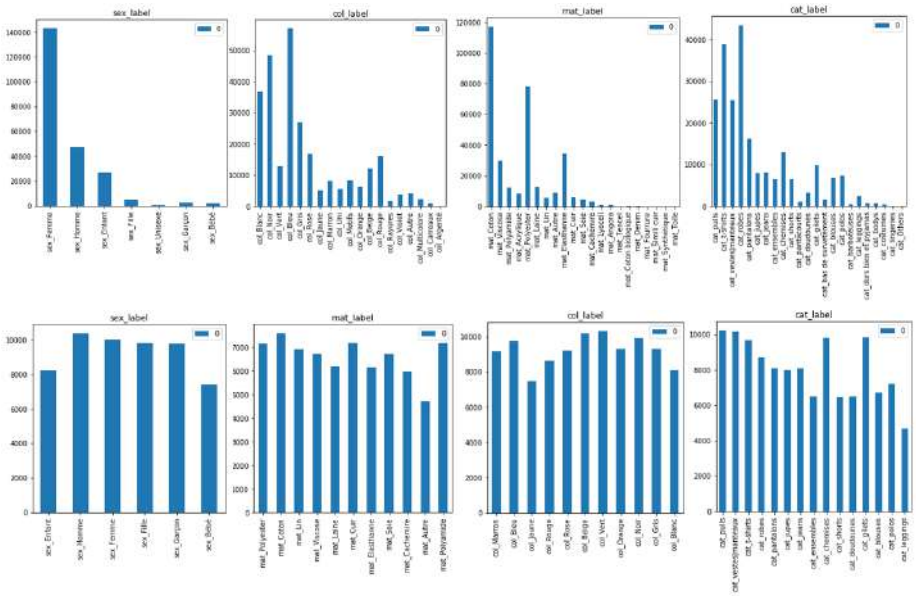

Figure 14: Comparaison des répartitions avant et après *sampling*

3.4 La pipeline générale de mon modèle:

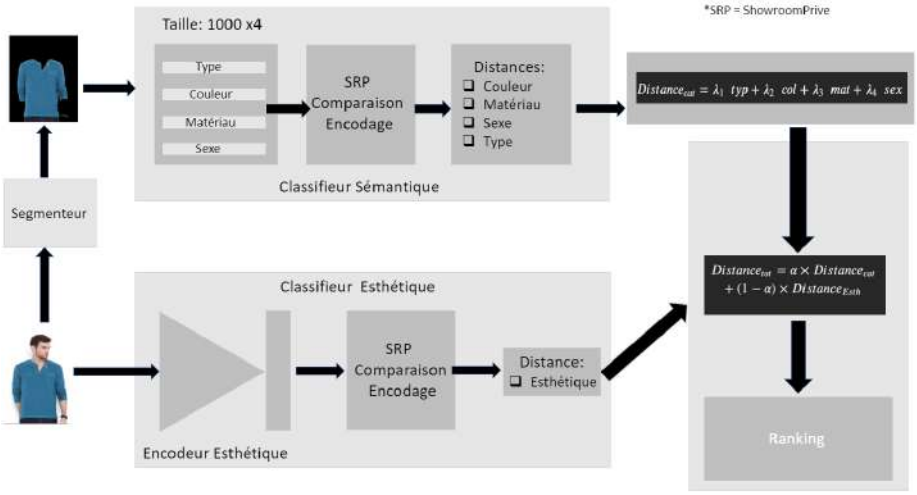

Figure 15: Pipeline générale du modèle Street 2 Shop

La pipeline générale du modèle (Voir Figure 15) a plusieurs fois changé tout au long de mon stage. Les pages qui suivent décrivent chacun des éléments de

la pipeline un à un en commençant par le segmenteur:

3.5 Partie Segmentation:

Nous avons choisi de faire une première partie permettant de segmenter les vêtements sur les photos afin d'améliorer les performances des modèles mais comme Showroomprive ne possède pas d'images de vêtements avec des masks associés, j'ai opté pour un modèle pré-entraîne trouvé sur un repos github[9]. En adaptant le code j'ai pu segmenter l'ensemble des images sur le dataset.

Toutefois je me suis assez rapidement rendu compte que beaucoup d'images présentaient des erreurs de segmentation qui parfois effaçaient presque la moitié du produit (Voir Figure 16).
Il fallait donc trouver une solution fiable pour segmenter les vêtements.

Figure 16: Exemple de produit mal segmenté par le modèle

J'ai donc décidé d'utiliser le modèle de segmentation afin d'en extraire la box de détection, c'est à dire les deux pixels non noirs situés les plus en bas à gauche et en haut à droite de l'image.
Cette box est particulièrement intéressante pour un vêtement puisqu'il a la particularité d'être situé dans une unique région de l'image. Autrement dit, les pixels qui le constituent sont tous adjacents entre eux.

Grâce à la box de détection j'ai utilisé un algorithme non supervisé pour segmenter le vêtement à l'intérieur de la box.
Cet algorithme est le grabcut[5] crée par OpenCV[8].

Il s'agit d'un algorithme permettant d'extraire et segmenter un objet du premier plan à partir d'une box de détection.

L'algorithme fonctionne de la manière suivante:

L'utilisateur renseigne la box de détection. Tout ce qui est en dehors de cette box est comptabilisé comme le fond et sera donc masqué.
Pour ce qui est de l'intérieur de la box, il s'agit d'un Gaussian mixture model

qui détermine ce qui est au premier plan et ce qui est au second plan.

Pour éviter de couper le produit, il faut donc augmenter légèrement la box de détection avec une faible marge de sûreté afin de s'assurer que l'ensemble du produit est bien dans le rectangle.

la segmentation utilisant le modèle et le grabcut est beaucoup plus efficace que le modèle à lui seul. La Figure 17 montre un exemple de comparaison sur un produit donné entre les deux methodes.

Figure 17: Comparaison de la segmentation entre le modèle (à gauche) et le modèle + grabcut (à droite)

3.6 Encodeur Catégoriel

Une fois l'ensemble des images segmentées, il faut s'attaquer à l'encodeur catégoriel. Le but de l'encodeur catégoriel est de prédire 4 labels sur la photo:

- Le sexe de l'acheteur
- La matière du vêtement
- La couleur du vêtement
- Le type de vêtement

Ce qu'il est important de constater c'est qu'un vêtement peut avoir plusieurs matières, couleurs et peut aussi être unisexe.
En revanche, un vêtement n'a qu'un seul type: c'est un pantalon, un gilet ou une doudoune etc..

Pour les 3 premiers labels, il s'agit donc d'un problème multi-class et multi-label tandis-que pour le type il s'agit d'un problème multi-class uni-label.

Ainsi, nous avons rapidement choisi,*Alexandre* et moi et même d'entraîner ces 4 parties de manière indépendante selon quatre modèles différents que nous regrouperons ensuite en un grand modèle.

Pour la répartition des données, nous avons déjà fait en sorte de rendre les classes plus homogènes dans (3.3.3). Il s'agit maintenant de construire le générateur.

3.6.1 Générateur et fonction de Pré-processing

Après avoir utilisé les générateurs déjà implémentés par TensorFlow[6] avec ImageGenerator, j'ai opté pour un générateur créé avec une classe à la main afin d'avoir d'avantage la main mise sur l'ensemble des processus s'appliquant lors de la génération d'une image.

Pour ce qui est de la taille de l'image en entrée, après avoir beaucoup changé j'ai fini par choisir une taille souvent utilisée: 224 par 224.

J'ai ensuite utilisé plusieurs types de fonctions de pré-processing:

Tout d'abord une "feature normalization" mais j'ai très vite changé d'avis puisque les images étaient segmentées donc lorsque l'on se trouvait dans une zone segmentée pour une image et non segmentée pour une autre, la normalisation pénalisait d'avantage que le contraire. Ensuite j'ai utilisé une normalisation classique sur les images puis un simple rescaling de $\frac{1}{255}$ et enfin la fonction de pré-processing du Resnet50[11] lorsque j'ai commencé à faire du transfert learning.

3.6.2 Augmenteur

Lorsque j'ai commencé à vouloir utiliser un augmenteur afin de simuler artificiellement de la diversité dans les données, je me suis très vite orienté vers la librairie Imgaug. Imgaug est très facile d'utilisation et permet de faire beaucoup de transformations différentes sur les images.

J'ai finalement choisi d'implémenter un augmenteur comportant:

- Une rotation avec un angle relativement faible $(-20°, 20°)$
- Une faible translation horizontale
- Un flip horizontal
- Une faible compression Jpeg

et tout cela avec une probabilité d'application relativement faible, tournant autour de 0.3, 0.4 et 0.5 pour le flip horizontal.

La compression Jpeg était intéressante pour le projet car le but final est qu'un utilisateur quelconque puisse envoyer une photo sur le site et avoir une recommandation associée. Il est donc probable que la qualité de la photo soit mauvaise, il faut donc que le modèle puisse fonctionner correctement si tel est le cas.

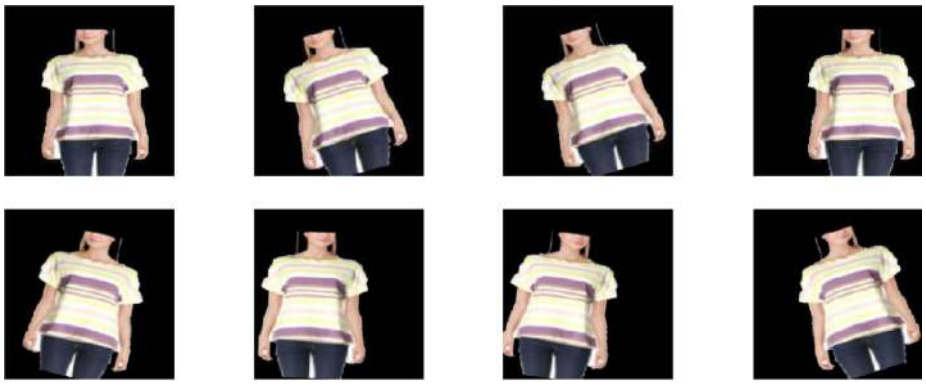

Figure 18: Exemples d'images générées

3.7 Modèle et entraînement

3.7.1 Architecture:

Le modèle a beaucoup évolué au cours du temps mais j'ai fini par opter pour du transfert learning pour l'entraînement.

Le transfert learning consiste à utiliser une partie d'architecture déjà implémentée avec des poids pré-entraînés sur un dataset afin de l'inclure dans le modèle et de l'utiliser comme initialisation pour l'entraînement.

Au début , j'ai choisi le VGG16[10] comme modèle pré-entraîné pour l'initialisation.

Malheureusement, après beaucoup d'entraînements, de changements dans les hyper-paramètres et les optimiseurs, le modèle n'apprenait pas et rendait en sortie un vecteur contenant uniquement des probabilités autour de 0.5, comme si un minimum local était atteint et que le modèle ne parvenait pas à en sortir.

Finalement j'ai choisi de changer de modèle pour le transfert learning et de prendre l'architecture et les poids du Resnet50[11] comme initialisation et le modèle a finalement commencé à apprendre et renvoyer des résultats plus convainquant.

L'architecture que j'ai finalement choisi d'utiliser est la suivante:

- Input: image suite au pré-processing — taille en sortie (3, 224, 224)
- Modèle Resnet 50 sans le haut — taille en sortie: (7, 7, 2048)
- Global Average Pooling 2D — taille en sortie: (2048)
- Dense (activation = 'relu') — taille en sortie: (1000)
- Dropout (0.2) — taille en sortie: (1000)

- Dense (activation = 'softmax ou sigmoid') taille en sortie: $(n_{classes})$

Avec une activation = *softmax* pour le modèle multi-class et *sigmoid* pour les modèles multi-labels.

Pour ce qui est de la valeur de $n_{classes}$, il y a :

11 classes pour la couleur: Orange, Jaune, Blanc, Vert, Gris, Rouge, Marron, Beige, Bleu, Noir et Rose.

11 classes pour la matière: Coton, Viscose, Polyamide, Elasthane , Polyesther, Laine, Cuir, Soie, Cachemire, Lin et Autre.

6 classes pour le sexe: Homme, Enfant, Femme, Fille, Garçon, Bébé.

15 classes pour le type: Pulls, T-shirts, Vestes/Manteaux, Pantalons, Jupes, Jeans, Ensembles, Chemises, Shorts, Doudounes, Glets, Robes, Blouses, Polos, Leggings.

3.7.2 La fonction de coût:

Pour les modèles multi-label, il s'agit de prédire la présence ou l'absence de chacun des labels.
Si la répartition des données dans les classes a pu être traité auparavant, il reste un souci majeur:
Il y a bien plus de produits qui ne sont pas orange que de produits qui le sont ou de vêtements qui ne sont pas en cuir que de vêtements qui le sont par exemple.

Ce problème est majeur car il se pose pour chacune des classes et si rien n'est fait pour le traiter alors le modèle prédira constamment l'absence de chacun des labels peu importe le produit et il aura raison 80% du temps.

Pour palier à ce problème nous introduisons une nouvelle loss: la *weighted binary cros-entropy*. Pour chacune des classes on définit alors le poid p de présence et d'absence comme suit:

$$p^i_{pres} = \frac{n_{sample}}{2n_i} \qquad p^i_{abs} = \frac{n_{sample}}{2(n_{sample} - n_i)} \tag{10}$$

n_i le nombre d'images qui présentent le label i
Dans notre cas $\forall i, p^i_{pres} > p^i_{abs}$:

La fonction de coût s'exprime alors comme suit:

$$loss(y_{true}, y_{pred}) = -p_{pres} y_{true} \log(y_{pred}) - p_{abs}(1 - y_{true}) \log(1 - y_{pred}) \tag{11}$$

Ainsi, la descente de gradient sera beaucoup plus importante si le modèle prédit qu'un label est absent alors qu'il est présent plutôt que si il prédit qu'un label est présent alors qu'il est absent.

3.7.3 Optimiseur et learning rate:

Pour ce qui est de l'optimiseur, J'ai toujours utilisé Adam[12] durant l'ensemble de la période.
L'hyper-paramètre principal que j'ai du changer à plusieurs reprises était le *learning rate*. Comme il s'agissait de transfert learning, le *learning rate* pouvait être plutôt bas pour commencer. Pour ma part, j'ai choisi un *learning rate* initial de $lr = 10^{-4}$.

Cependant même avec un *learning rate* plutôt bas, lors de l'entraînement il arrivait que la descente de gradient amène la loss à tendre vers $+\infty$.
Pour régler ce problème il a donc fallu prévoir un *learning rate* décroissant selon les itérations allant jusqu'à 10^{-7}, 10^{-8} pour les dernières itérations

3.8 Model esthétique

La pipeline du modèle esthétique est visible Figure 12, cependant nous n'avons pas exactement suivi le même procédé.
Nous avons un peu adapté la pipeline en retirant toute la partie *rating* ainsi que l'extraction de la HSV ("hue, saturation & value").
Le but était donc de faire autant de modèles qu'il y avait de label à prédire: à savoir 14.

Chacun des modèles avait donc la même architecture, celle présentée dans [4] visible sur la Figure 19:

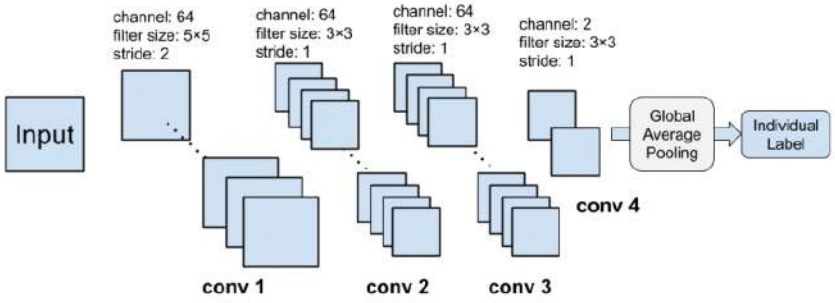

Figure 19: Architecture du modèle esthétique

Comme indiqué Figure 11, il fallait tout de même supprimer certains labels qui décrivaient les paramètres techniques de la prise de photo et étaient donc propre à l'appareil capturant la photo et non l'esthétique de l'objet photographié.

Après un *remapping* des labels , j'ai pu entraîner le modèle prédisant les différents labels avec la même fonction de coût et le même optimiseur que ceux utilisés précédemment (en 3.7.2 et 3.7.3).

3.9 Nouvelles données: Projet *Scraping*

Comme expliqué dans la partie description des données, nous nous sommes très vite rendu compte d'un souci majeur. Le but du projet consiste à faire de la recommandation de vêtement pour un utilisateur quelconque photographiant un produit depuis un magasin ou même chez lui.

Cependant notre modèle n'a été entraîné que sur des visuels Showroomprive qui ont tous la même forme. Il s'agit souvent d'un mannequin posant devant un fond uni ou d'une photo d'un produit non porté sur un fond uni.
Ces données ne sont pas représentatives de ce que l'utilisateur peut envoyer mais Showroomprive ne possède pas de données d'utilisateurs annotées avec:

- La couleur
- La matière
- Le sexe de l'acheteur
- Le type de vêtement

Pour avoir accès à ces données nous avons décidé de les extraire d'un site de vente de vêtements en ligne de particulier à particulier.

Après quelques jours d'attente d'un retour du service juridique afin de commencer le projet, nous avons eu leur feu vert.

J'ai commencé le projet en visitant la plateforme de vente en question afin de comprendre l'agencement des données sur les différentes pages et de réfléchir à un moyen de les extraire.
Puis, j'ai commencé à paralléliser les taches en *scrapant* en même temps toutes les pages des différents types de vêtements du site.

Par exemple: pour les t-shirts, j'ai suivi le processus suivant:

1. Commencer sur la première page t-shirt du site.
2. Récupérer l'ensemble des urls produits sur la page.
3. Ouvrir toutes les pages des produits pas encore visités.

4. Récupérer l'image, le sexe, le type de produit et la couleur sur les pages et les transférer dans un dataset S3
5. Enregistrer les identifiants des produits visités dans un second dataset S3.
6. Passer à la page suivante sur l'accueil t-shirt.

Ce projet a été réalisé sur Dataiku DSS en utilisant la librairie BeautifulSoup[13] en Python. Le flux correspondant est représenté ci-dessus Figure 20:

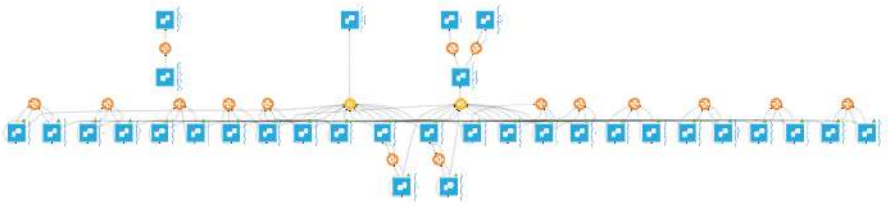

Figure 20: Flux du projet *scraping*

Chacune des recettes python en orange exécute les étapes décrites précédemment pour un type de produit donné. La recette met ainsi à jour deux datasets (Figure 21, 22):

- Les données colectées (Figure 21)
- Les identifiants visités (Figure 22)

brand	categorie	couleur	prix	sexe
string	string	string	string	string
Text	Text	Array	Money amount	Text
Roberto bassi	costumes-and-blazers	[u'kaki', u' moutarde']	50,00 €	Hommes
Emidio Tucci	costumes-and-blazers	[u'bleu']	90,00 €	Hommes
Mango	costumes-and-blazers	[u'gris']	34,99 €	Hommes
Mango	costumes-and-blazers	[u'noir']	49,00 €	Hommes
nouveaux ateliers	costumes-and-blazers	[u'gris']	170,00 €	Hommes
The Kooples	costumes-and-blazers	[u'bleu', u' marine']	68,00 €	Hommes
Caramelo	costumes-and-blazers	[u'noir']	85,00 €	Hommes
EL GANSO	costumes-and-blazers	[u'gris']	40,00 €	Hommes
Burton	costumes-and-blazers	[u'gris']	210,00 €	Hommes
Devred	costumes-and-blazers	[u'marine']	55,00 €	Hommes
Lotrec	costumes-and-blazers	[u'noir', u' blanc']	15,00 €	Hommes
Yves Saint Laurent	costumes-and-blazers	[u'marron']	285,00 €	Hommes
Easy Wear	costumes-and-blazers	[u'marine']	27,00 €	Hommes
The Kooples	costumes-and-blazers	[u'beige ', u' cr\xe8me']	250,00 €	Hommes

Figure 21: Dataset de données collectées pour les costumes

Figure 22: Datasets des identifiants produits visités pour les costumes

Comme le *scraping* se fait en parallèle sur tous les types de produits,j'ai utilisé beaucoup de machines différentes travaillant parallèlement.

Pour se faire, j'ai utilisé Kubernetes.

Il s'agit d'une plate-forme créée pour la gestion de charges de travail et de services conteneurisés. Grâce à Kubernetes, nous avons pu définir "un container" sur DSS.

Un container est un regroupement de plusieurs "nodes" (qui sont eux mêmes des regroupements de machines) sur lequel on définit un environnement d'exécution.

Lorsque l'on exécute un travail sur un container, celui-ci répartit l'exécution dans plusieurs nodes différents en fonction des ressources disponibles et allouées.

Il a donc fallu créer un environnement d'exécution pour faire du *sampling* et définir les machines mises à disposition dans les nodes.

Le *scraping* du site s'est fait plutôt lentemenetrop de requêtes.

Après une semaine de *scraping* nous avons pu récupérer environ 100000 images de produits et les labels correspondant sur le site.

Avec ces nouvelles données de particuliers, j'ai pu ré-entraîner tout le modèle pour avoir de meilleurs résultats.

3.10 Résultats des modèles

Dans cette section je ne parlerai que des résultats obtenus en ayant ajouté les nouvelles données à celles de Showroomprive. La comparaison du modèle entraîné seulement grâce aux données Showroomprive avec celui entraîné avec les nouvelles données sera faite dans le partie Évaluation (3.11).

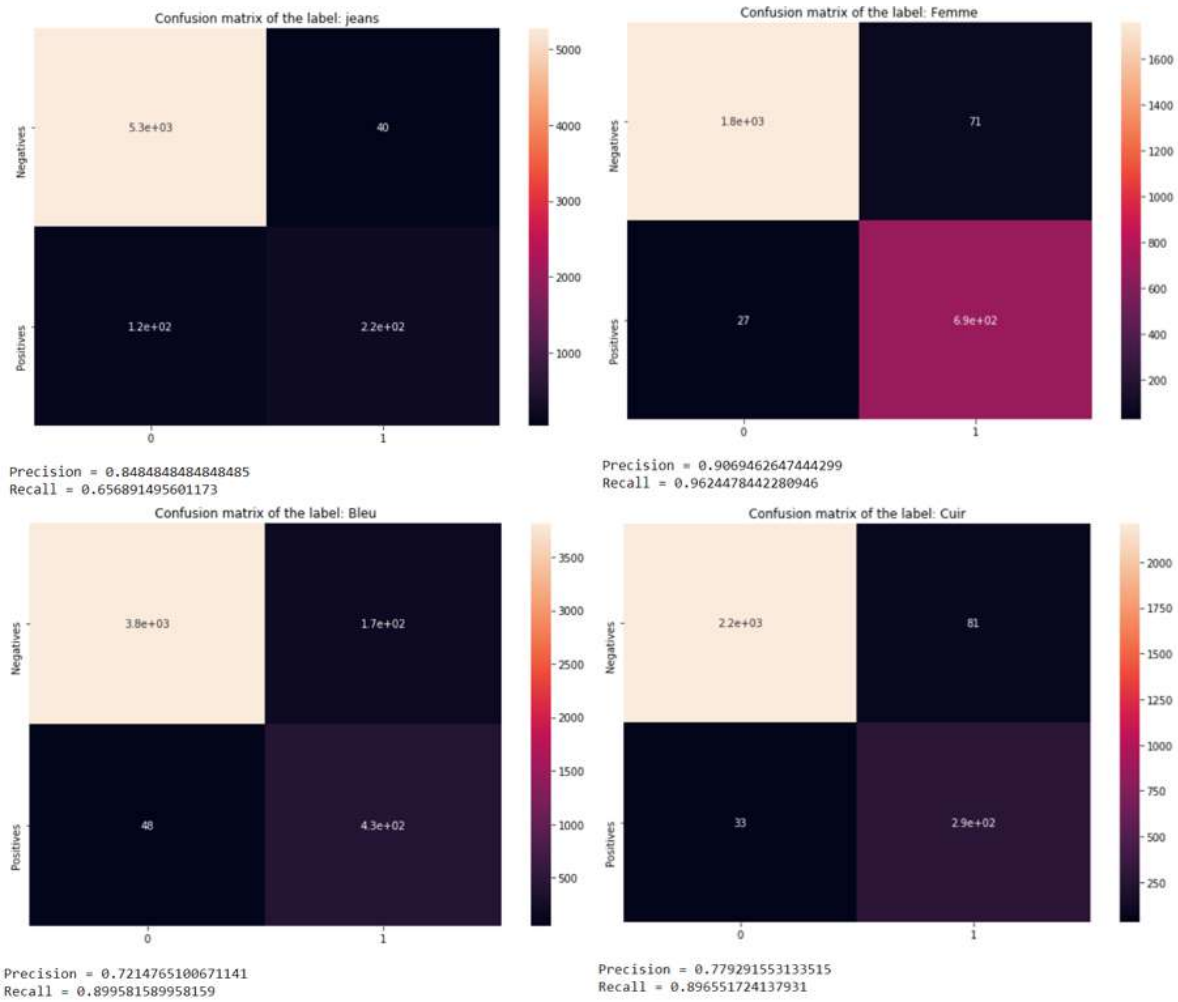

Figure 23: Matrices de confusion de certaines classes

Pour 3 des 4 modèles, les résultats ont été très satisfaisants avec plus de 90% d'exactitude (*accuracy*) pour chacun d'entre eux.
La figure 23 représente 4 matrices de confusions de classes suite à l'entraînement du modèle.

Cependant, pour ce qui est du modèle prédisant la matière, comme on pouvait sûrement s'en douter les résultats sont moins bons mais restent satisfaisants puisque de toute façon je n'utiliserai pas les prédictions en elles-même mais plutôt les encodages.

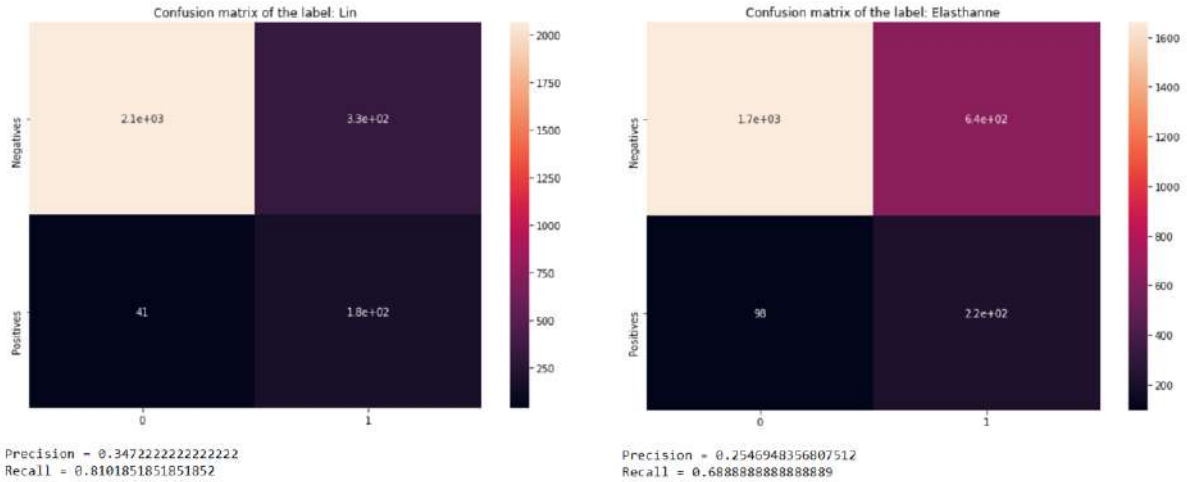

Figure 24: Exemple de Matrices de confusion du modèle prédisant la matière

Voici maintenant quelques exemples de prédictions des différents modèles (Figure 25).

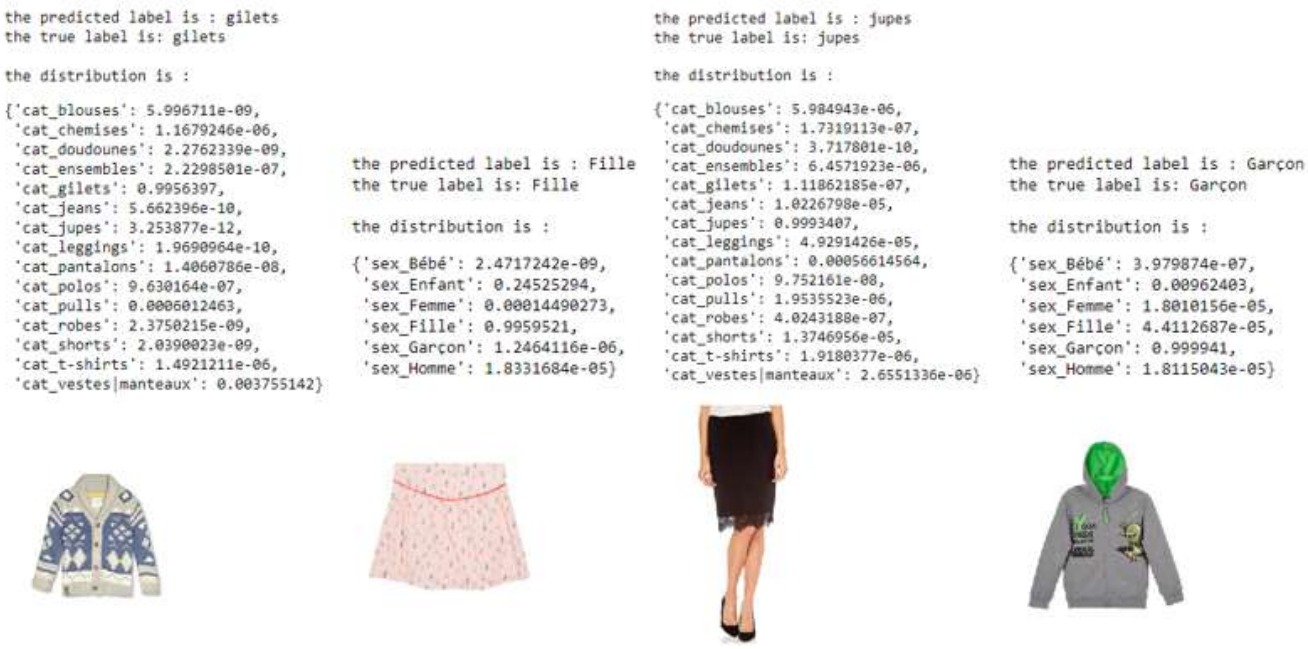

Figure 25: Quelques distributions de prédictions du modèle

La figure 26 représente un plot T-SNE de l'encodage couleur des produits showroomprivre:

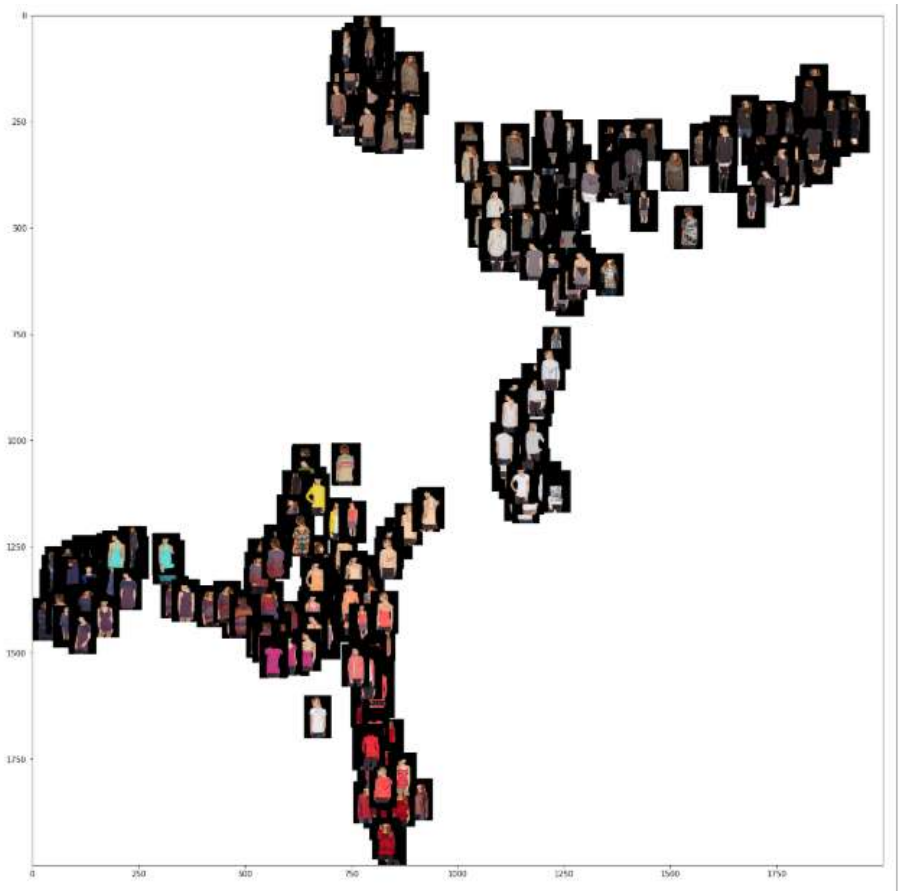

Figure 26: Plot T-SNE des encodages couleurs de produits showroomprive

Enfin, les résultats les plus importants sont présentés figure 29, 30, 31 à la fin du rapport. Il s'agit de simulations de recommandations de produits Showroomprivee à partir de photos de vêtements prises par des particuliers.

3.11 Évaluation du modèle

Pour évaluer le modèle, j'ai longuement réflechi et cherché avant de trouver une solution.
Le Dataset Deep Fashion2 contient une catégorie nommée *Clothes Retrieval*.
On trouve dans cette catégorie beaucoup d'images de produits portés par des utilisateurs ainsi que des visuels de ces mêmes produits issus d'un site de vente.

Chaque image est légendée par un $pair_{id}$ qui permet d'identifier le produit sur la photo ainsi qu'un attribut *source* qui permet de savoir si la photo est issue d'un site de vente ou si elle a été prise par un utilisateur. (Voir Exemple Figure 27)

Figure 27: Exemples de visuels avec le même $pair_{id}$ et une 2 *sources* différentes: 'user' à gauche et 'shop' à droite

Pour évaluer le modèle il suffit de prendre une image utilisateur avec un $pair_{id}$ quelconque puis d'utiliser cette image comme *input* de la recommandation de mon modèle sur les données issues du site.
Si le modèle fonctionne parfaitement le modèle devrait pouvoir recommander toutes les images issues du site avec les même $pair_{id}$ que l'image en *input* en premier. (Voir Figure 28)

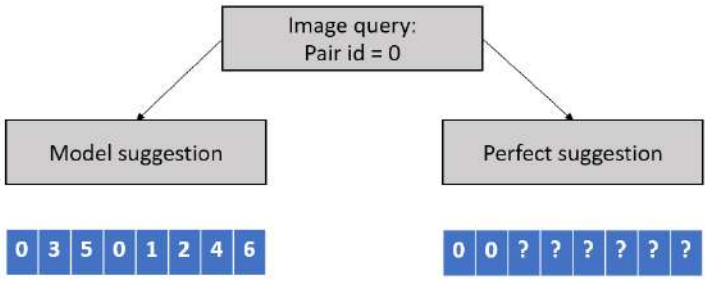

Figure 28: Schema du processus d'évaluation du modèle

Pour évaluer le modèle, j'ai calculé le pourcentage de simulations de recommandation pour lesquelles l'une des images avec le même $pair_{id}$ que l'image *query* apparaît dans les 10 premières recommandations.

Avec le modèle entraîné avec les données Showroomprive uniquement, le score obtenu est de 22% contre 31% pour celui entraîné avec les nouvelles données issues du site de vente de particulier à particulier.

4 Conclusion

En somme, j'ai beaucoup appris durant ce stage et tout le long de ma mission.
Au niveau technique, j'ai pu m'améliorer dans l'utilisation de Keras[7] et TensorFlow[6].
J'ai aussi appris à faire du scraping ou encore à utiliser de nouveaux outils comme AWS ou DSS.
Pour ce qui est de l'aspect recherche, j'ai découvert beaucoup d'articles très intéressants utilisant des méthodes novatrices et très originales que ce soit pour la recommandation ou simplement le traitement d'images.
J'ai eu aussi la chance d'être bien encadré et de pouvoir travailler dans un équipe soudée et très accueillante.
Aujourd'hui, le projet Street 2 Shop est terminé et prêt à être mis en production.

References

[1] Yihui Ma, Jia Jia, Suping Zhou, Jingtian Fu, Yejun Liu, Zijian Tong *Towards Better Understanding the Clothing Fashion Styles: A Multimodal Deep Learning Approach* https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/viewFile/14561/13740

[2] Julian McAuley, Christophe Targett, Qinfeng('Javen') Shi, Anton van den Hengel *Image-based Recommendations on Styles and Substitutes* https://arxiv.org/pdf/1506.04757.pdf

[3] Wenhui Yu, Huidi Zhang Xiangnam He, Xu Chen, Li Xiong Zheng Qin *Aesthetic-based Clothing Recommendation.* https://arxiv.org/pdf/1809.05822.pdf

[4] Zhangyang Wang, Shiyu Chang, Florin Dolcos, Diane Beck, Ding Liu, and Thomas Huang *Brain-Inspired Deep Networks for Image Aesthetics Assessment* https://arxiv.org/pdf/1601.04155.pdf

[5] OpenCV *Interactive Foreground Extraction using GrabCut Algorithm* https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html

[6] *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* https://arxiv.org/abs/1603.04467

[7] *Keras* https://keras.io/api/

[8] *OpenCV* https://opencv-python-tutroals.readthedocs.io/en/latest/

[9] anish9 *Fashion-AI-segmentation* https://github.com/anish9/Fashion-AI-segmentation

[10] Karen Simonyan, Andrew Zisserman *Very Deep Convolutional Networks for Large-Scale Image Recognition* https://arxiv.org/abs/1409.1556

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition* https://arxiv.org/abs/1512.03385

[12] Diederik P. Kingma, Jimmy Ba *Adam: A Method for Stochastic Optimization* https://arxiv.org/abs/1412.6980

[13] *BeautifulSoup* https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[14] *ImageNet* http://www.image-net.org/

5 Annexe:

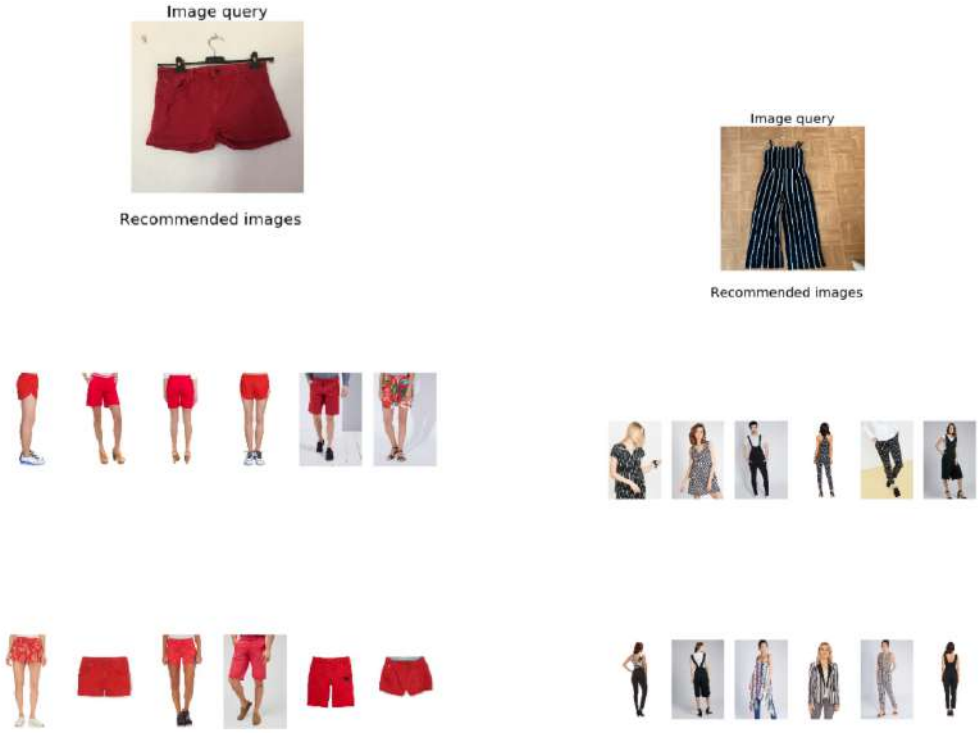

Figure 29: Simulation 1 & 2

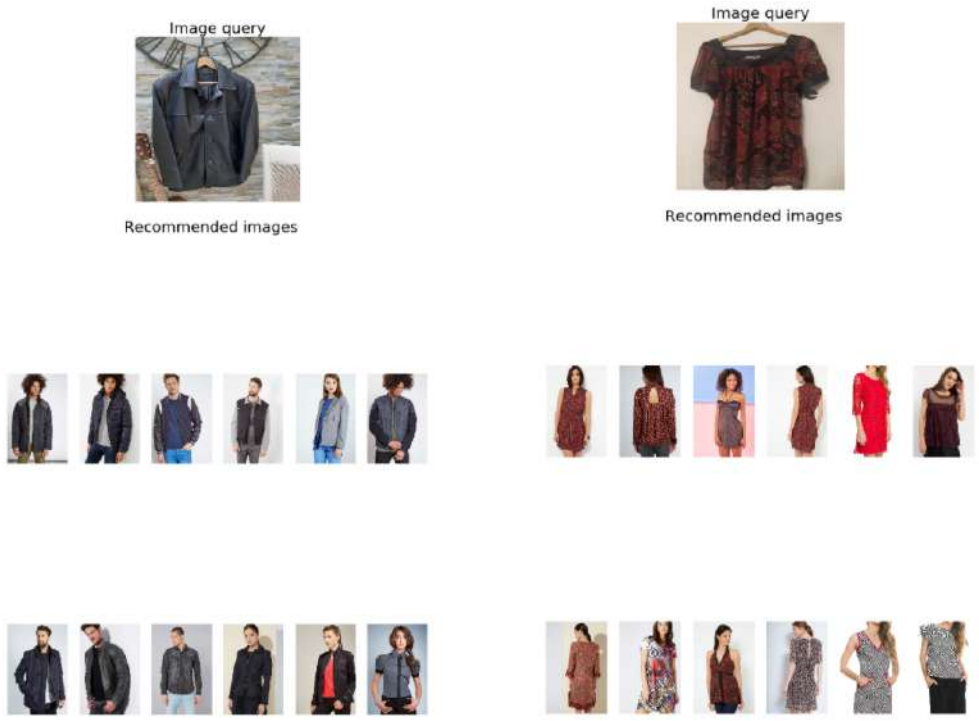

Figure 30: Simulation 3 & 4

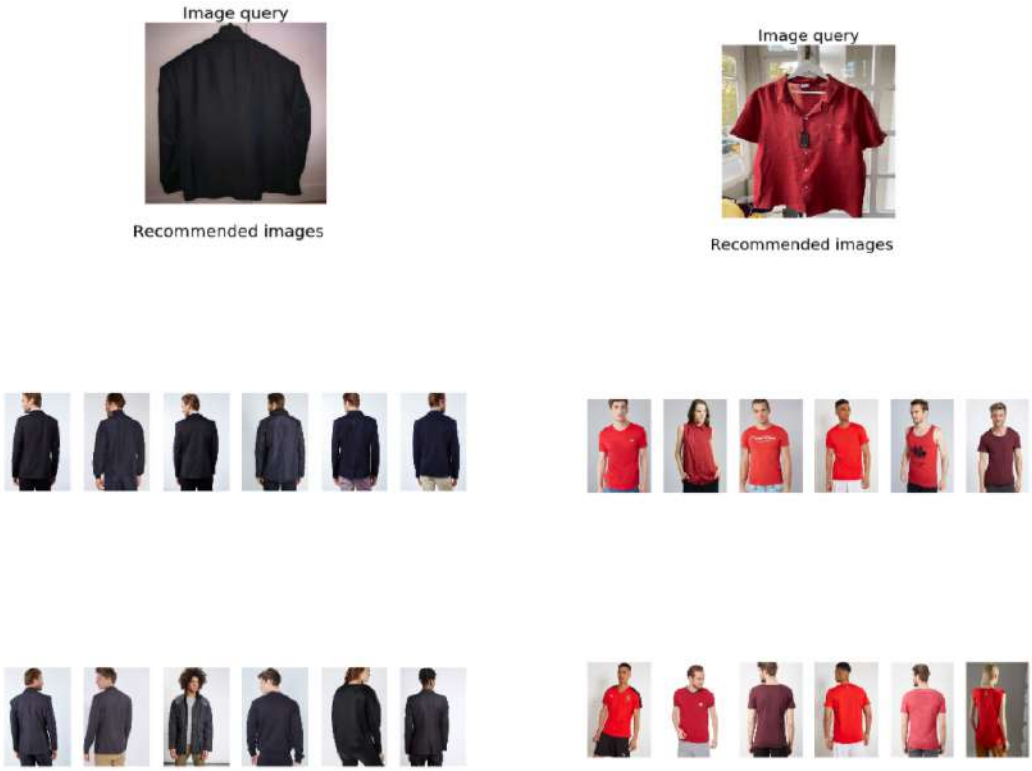

Figure 31: Simulation 4 & 5