Université Abdelmalek Essaadi

Faculté des sciences et techniques Tanger

Logiciels & Systèmes Intelligents

Mini-Projet

MACHINE LEARNING

REALISER PAR: Youssefi Nora & Mellouk Ahmed| 22-05-2022

ENCADRÉ PAR : PR. LOTFI EL AACHAK

Remerciement

L'année universitaire touchant à sa fin, nous tenions à vous dire que vous avez été un excellent professeur ! Nous vous remercions d'avoir partagé vos connaissances avec nous, d'avoir toujours été juste dans votre éducation monsieur Lotfi el aachak et de nous avoir toujours soutenus et aidés. Merci encore de la part de tous vos étudiants.

SOMMAIRE

Introduction

H&M Group est une famille de marques et d'entreprises avec 53 marchés en ligne et environ 4 850 magasins. Cette boutique en ligne offre aux acheteurs une vaste sélection de produits à parcourir. Mais avec trop de choix, les clients pourraient ne pas trouver rapidement ce qui les intéresse ou ce qu'ils recherchent, et finalement, ils pourraient ne pas faire d'achat.

Pour améliorer l'expérience d'achat, les recommandations de produits sont essentielles. Plus important encore, aider les clients à faire les bons choix a également des implications positives pour la durabilité, car cela réduit les retours et minimise ainsi les émissions du transport. Dans ce concours, H&M Group nous invite à développer des recommandations de produits basées sur les données des transactions précédentes, ainsi que sur les métadonnées des clients et des produits. Les métadonnées disponibles vont des données simples, telles que le type de vêtement et l'âge du client, aux données textuelles des descriptions de produits, aux données d'image des images de vêtements.

Présentation du Projet

Le monde d'achat en ligne est en évolution constante et la quantité des transactions de commerce en ligne qui se fait à un moment donné sont immenses. Toutes ces données peuvent être analysées et traitées afin de créer la meilleure expérience possible pour un client.

Cela explique l'importance des systèmes de recommandation, ce qui nous introduit à notre projet.

Notre Projet consiste à réaliser un Système de Recommandation basée sur classification d'images pour la société H&M.

Ce système sera capable de classifier plus que 100000 produits de la société H&M, selon des catégories, ce système sera créé avec un réseau neuronal convolutif via les librairies Tensor Flow et Keras.

Outils utilisés

Kaggle :

Kaggle, une filiale de Google LLC, est une communauté en ligne de scientifiques des données et de praticiens de l'apprentissage automatique. Kaggle permet aux utilisateurs de trouver et de publier des ensembles de données, d'explorer et de créer des modèles dans un environnement de science des données basé sur le Web, de travailler avec d'autres scientifiques des données et des ingénieurs en apprentissage automatique, et de participer à des concours pour résoudre les défis de la science des données.

TensorFlow

TensorFlow est une bibliothèque de logiciels gratuite et open source pour l'apprentissage automatique et l'intelligence artificielle. Il peut être utilisé dans une gamme de tâches, mais se concentre particulièrement sur la formation et l'inférence des réseaux de neurones profonds.

Google Colab

Colaboratory, ou « Colab » en abrégé, est un produit de Google Research. Colab permet à quiconque d'écrire et d'exécuter du code Python arbitraire via le navigateur, et est particulièrement bien adapté à l'apprentissage automatique, à l'analyse de données et à l'éducation.

I. Explication du dataset de la competition:

Pour cette competition,on dispose de l'historique des achats des clients au fil du temps, ainsi que des métadonnées à l'appui. Notre défi consiste à prédire quels articles chaque client achètera après le choix d'un autre article.

Les dossiers existant :

images/ - un dossier d'images correspondant à chaque article_id ; les images sont placées dans des sous-dossiers commençant par les trois premiers chiffres de l'article_id ; notez que toutes les valeurs article_id n'ont pas d'image correspondante.

articles.csv - métadonnées détaillées pour chaque article_id disponible à l'achat

customers.csv - métadonnées pour chaque customer_id dans l'ensemble de données

sample_submission.csv - un exemple de fichier de soumission au format correct

transactions_train.csv - les données de formation, composées des achats de chaque client pour chaque date, ainsi que des informations supplémentaires. Les lignes en double correspondent à plusieurs achats du même article. Votre tâche consiste à prédire les article_ids que chaque client achètera au cours de la période de 7 jours immédiatement après la période de données d'entraînement.

REMARQUE : On va utiliser pour notre model juste les images.

TELECHARGEMENT DU DATASET DE KAGGLE :

On doit suivre les étapes ci-dessous pour télécharger et utiliser les données kaggle dans Google Colab :

1. Accédez à votre compte, faites défiler jusqu'à la section API et cliquez sur Expire API Token pour supprimer les jetons précédents.

2. Cliquez sur Create New API Token- Il téléchargera le fichier kaggle.json sur votre machine.

3. Accédez à votre fichier de projet Google Colab et exécutez les commandes suivantes :

```
! pip installer -q kaggle
```

```
! pip install kaggle

Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (6.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.10.8)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.64.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.10)
```

from google.colab import files
files.upload()

```
from google.colab import files
files.upload()

Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving kaggle.json to kaggle (2).json
{'kaggle.json': b'{"username":"norayoussefi2000","key":"7e74c4fa79094fa2ee2da38e246a18ea"}'}
```

Choisissez le fichier kaggle.json que vous avez téléchargé

! mkdir ~/.kaggle

```
[ ]  !mkdir ~/.kaggle
```

! cp kaggle.json ~/.kaggle/

```
!cp kaggle.json ~/.kaggle/
```

Créez un répertoire nommé kaggle et copiez-y le fichier kaggle.json.

! chmod 600 ~/.kaggle/kaggle.json

```
[ ]  ! chmod 600 ~/.kaggle/kaggle.json
```

Modifiez les autorisations du fichier.

! kaggle datasets list

```
! kaggle datasets list
```

ref	title	size	lastUpdated	downloadCount	vot
muratkokludataset/date-fruit-datasets	Date Fruit Datasets	408KB	2022-04-03 09:25:39	8272	
victorsoeiro/netflix-tv-shows-and-movies	Netflix TV Shows and Movies	2MB	2022-05-15 00:01:23	742	
mdmahmudulhasansuzan/students-adaptability-level-in-online-education	Students Adaptability Level in Online Education	6KB	2022-04-16 04:46:28	5549	
muratkokludataset/rice-image-dataset	Rice Image Dataset	219MB	2022-04-03 02:12:00	1605	
paradisejoy/top-hits-spotify-from-20002019	Top Hits Spotify from 2000-2019	94KB	2022-04-26 17:30:03	1615	
victorsoeiro/disney-tv-shows-and-movies	Disney+ TV Shows and Movies	718KB	2022-05-13 23:24:57	481	
muratkokludataset/raisin-dataset	Raisin Dataset	112KB	2022-04-03 00:23:16	664	
muratkokludataset/pistachio-dataset	Pistachio Dataset	2MB	2022-04-03 08:38:21	620	
muratkokludataset/rice-msc-dataset	Rice MSC Dataset	102MB	2022-04-03 01:33:52	260	
muratkokludataset/pistachio-image-dataset	Pistachio Image Dataset	27MB	2022-03-28 18:01:27	562	
muratkokludataset/grapevine-leaves-image-dataset	Grapevine Leaves Image Dataset	109MB	2022-04-03 09:00:54	184	
muratkokludataset/durum-wheat-dataset	Durum Wheat Dataset	983MB	2022-04-03 00:02:29	110	
muratkokludataset/pumpkin-seeds-dataset	Pumpkin Seeds Dataset	393KB	2022-03-28 18:28:16	645	
muratkokludataset/dry-bean-dataset	Dry Bean Dataset	5MB	2022-04-02 23:19:30	556	
rinichristy/covid19-coronavirus-pandemic	COVID-19 Coronavirus Pandemic	9KB	2022-04-05 08:43:16	4359	
muhmores/spotify-top-100-songs-of-20152019	Spotify Top 100 Songs of 2010-2019	139KB	2022-04-09 06:35:36	5652	
surajjha101/stores-area-and-sales-data	Supermarket store branches sales analysis	10KB	2022-04-29 11:10:16	1488	
aslanahmedov/walmart-sales-forecast	Walmart Sales Forecast	3MB	2022-04-21 05:28:20	2709	
digvijaysinhgohil/covid19-data-deaths-and-vaccinations	Covid-19 Data Deaths and Vaccinations	2MB	2022-05-04 19:06:58	985	
xhlulu/cpc-codes	Cooperative Patent Classification Codes Meaning	5MB	2022-03-22 03:04:36	2451	

- C'est tout ! Vous pouvez vérifier si tout va bien en exécutant cette commande.

Télécharger les données

! kaggle competitions download -c 'name-of-competition'

```
[ ] !kaggle competitions download -c h-and-m-personalized-fashion-recommendations

Downloading h-and-m-personalized-fashion-recommendations.zip to /content
100% 28.7G/28.7G [02:38<00:00, 173MB/s]
100% 28.7G/28.7G [02:38<00:00, 194MB/s]
```

Utilisez la commande unzip pour décompresser les données :

```
!unzip h-and-m-personalized-fashion-recommendations.zip

Streaming output truncated to the last 5000 lines.
  inflating: images/089/0890639002.jpg
  inflating: images/089/0890677001.jpg
  inflating: images/089/0890677002.jpg
  inflating: images/089/0890677004.jpg
  inflating: images/089/0890683001.jpg
  inflating: images/089/0890683002.jpg
  inflating: images/089/0890684001.jpg
  inflating: images/089/0890684002.jpg
  inflating: images/089/0890686002.jpg
  inflating: images/089/0890697001.jpg
  inflating: images/089/0890697002.jpg
  inflating: images/089/0890700001.jpg
  inflating: images/089/0890717001.jpg
  inflating: images/089/0890717002.jpg
  inflating: images/089/0890717003.jpg
  inflating: images/089/0890722001.jpg
  inflating: images/089/0890722002.jpg
```

II. Exploration et Analyse des données:

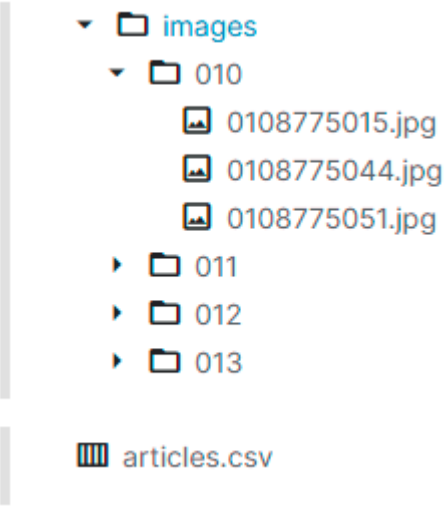
Dans cette partie, on vise à explorer nos données brutes initiales et les mieux comprendre et clarifier.

Et en plus on va essayer d'analyser et détecter des problèmes et des anomalies dans les données qui peuvent influencer notre modèle négativement, avec l'objectif de les traiter après dans la partie du traitement des données.

A. EXPLORATION DU DATASET:

Notre dataset initial consiste de deux documents principales:

- ***"Image"*** : Un dossier contenant toutes les images des produits dans notre dataset.
- "**article.csv**" : un fichier excel contenant des métadonnées sur les images, principalement leur catégories et types.



- ***Images :***

A propos des images qui construisent notre dataset, il en existe plus de **100 000** images de différentes tailles et dimensions, ce qui va nécessiter un **redimensionnement** dans la partie traitement.

```
import PIL
import matplotlib.pyplot as plt
import pathlib

data_dir = pathlib.Path("images")
image_count = len(list(data_dir.glob('*/*.jpg')))
print("Nombre total d'image : ",image_count)
```

```
Nombre total d'image :  105100
```

Toutes ces images sont des images professionnelles, qui suivent la même structure, un produit au milieu avec une arrière-plan blanche, ce point qui sera bénéficiaire pour notre modèle puisque toutes les images sont de qualité similaire.

- ***Catégories***:

On trouve qu'il existe 19 catégories d'images dans notre dataset, donc 19 classes.

```
#nombre de classe - 19
print("Nombre des categories (classes) initial est : ",len(set(labelled_images["category"])))
```

```
Nombre des categories (classes) initial est :  19
```

```
Nombre de produit par categorie :
Garment Upper body       42741
Garment Lower body       19812
Garment Full body        13292
Accessories              11158
Underwear                 5490
Shoes                     5283
Swimwear                  3127
Socks & Tights            2442
Nightwear                 1899
Unknown                    121
Underwear/nightwear         54
Cosmetic                    49
Bags                        25
Items                       17
Furniture                   13
Garment and Shoe care        9
Stationery                   5
Interior textile             3
Fun                          2
Name: category, dtype: int64
```

Nombre d'images par classe

- ***article.csv :*** ce fichier qui contient des métadonnées à propos des images, parmi lesquelles les catégories des images, qui vont nous permettre de les classifier.

```
Data columns (total 25 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   article_id                    105542 non-null  int64
 1   product_code                  105542 non-null  int64
 2   prod_name                     105542 non-null  object
 3   product_type_no               105542 non-null  int64
 4   product_type_name             105542 non-null  object
 5   product_group_name            105542 non-null  object
 6   graphical_appearance_no       105542 non-null  int64
 7   graphical_appearance_name     105542 non-null  object
 8   colour_group_code             105542 non-null  int64
 9   colour_group_name             105542 non-null  object
 10  perceived_colour_value_id     105542 non-null  int64
 11  perceived_colour_value_name   105542 non-null  object
 12  perceived_colour_master_id    105542 non-null  int64
 13  perceived_colour_master_name  105542 non-null  object
 14  department_no                 105542 non-null  int64
 15  department_name               105542 non-null  object
 16  index_code                    105542 non-null  object
 17  index_name                    105542 non-null  object
 18  index_group_no                105542 non-null  int64
 19  index_group_name              105542 non-null  object
 20  section_no                    105542 non-null  int64
 21  section_name                  105542 non-null  object
 22  garment_group_no              105542 non-null  int64
 23  garment_group_name            105542 non-null  object
 24  detail_desc                   105126 non-null  object
dtypes: int64(11), object(14)
```

Information sur article.csv

B. ANALYSE DES DONNEES:

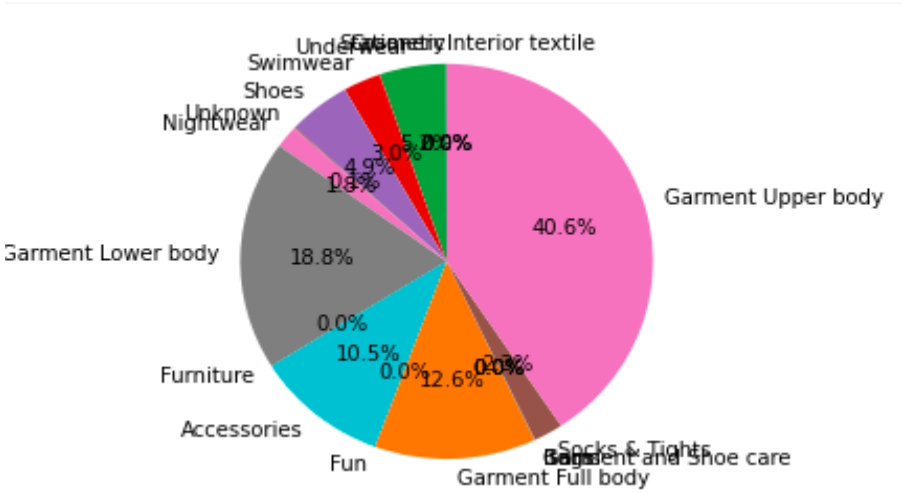
Dans cette section, on va s'intéresser par analyser nos données afin de détecter des problèmes potentiels pour notre modèle, et essayer de détecter toutes les anomalies afin de les traiter dans la partie traitement des données.

Parmis les plusieurs problèmes et anomalies qui existe dans notre dataset brute, on trouve:

- ***<u>Déséquilibrement des données</u>***:

On voit que le nombre d'images par catégorie est mal distribué entre les classes, avec des classes qui contiennent plus de 40% des images par contre d'autres qui contiennent ~ 0% même si ces derniers contiennent des images.

Ce déséquilibre qui va influencer notre modèle négativement, et va créer un précision fausse et non crédible, et en plus il augmente la chance d'un overfitting.



On peut solutionner ce problème de déséquilibre par le sous-échantillonnage (Undersampling).

- ***<u>Dimensions différentes des images</u>***:

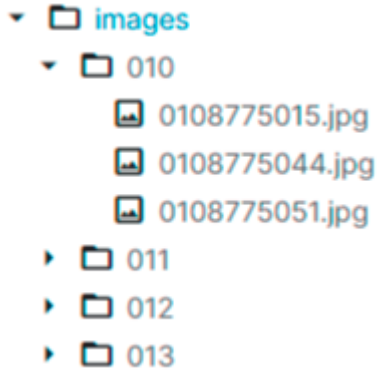
On voit qu'il y en des images avec des dimensions différentes, ces différentes dimensions qui vont affecter notre modèle, donc on doit penser à un redimensionnement.

```
print("Dimension d'image :")
print(PIL.Image.open(str(product[0])).size)
print(PIL.Image.open(str(product[1])).size)
print(PIL.Image.open(str(product[2])).size)
```

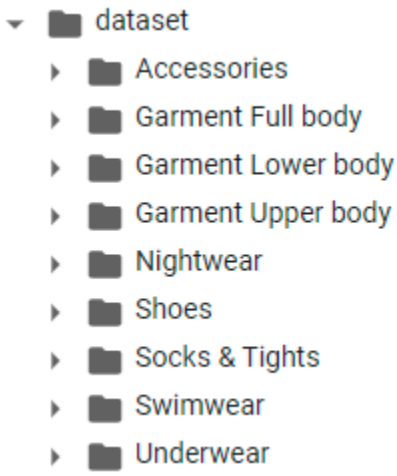
```
Dimension d'image :
(1166, 1750)
(1166, 1750)
(1166, 1750)
```

- ***<u>Dés-organisation des images dans la dataset</u>***:

On voit que la structure des images dans notre dataset est non organisée.



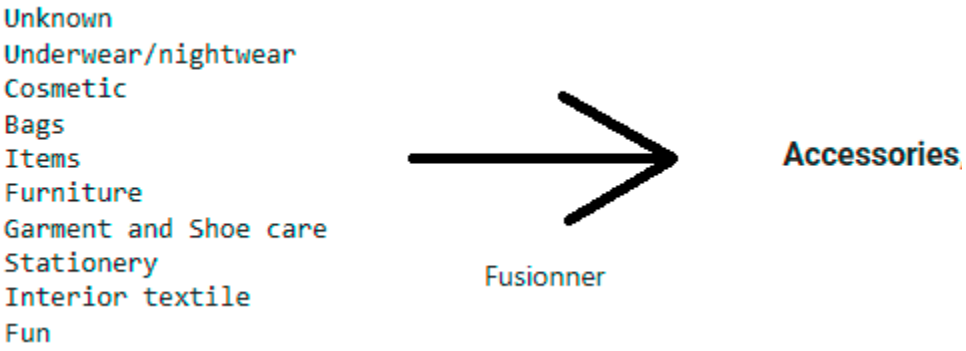
Notre objectif sera de regrouper les images qui appartiennent à une classe dans un dossier avec le nom de la classe, tel que:



- ***Classes similaires***:

On observe qu'il y en des classes qui sont très similaires qui peuvent être groupés en une seule classe.

Tel que:

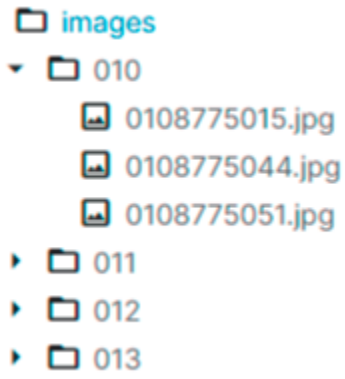


III. Traitement des données:

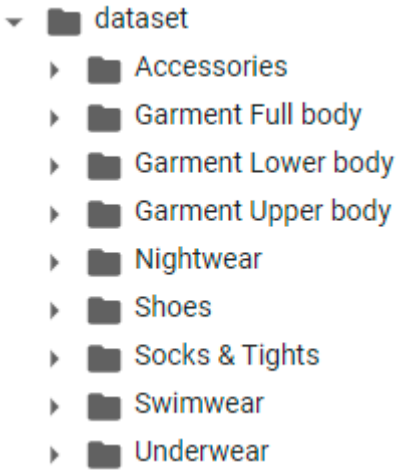
Dans cette partie, on va adresser tous les problèmes détectés au niveau de la partie d'analyse des données.

- ***Grouper les images de même classe dans un seul dossier:***

On veut grouper les images qui appartiennent à la même classe dans le même dossier , nommé selon cette classe, cette manipulation va simplifier la séparation des données après.



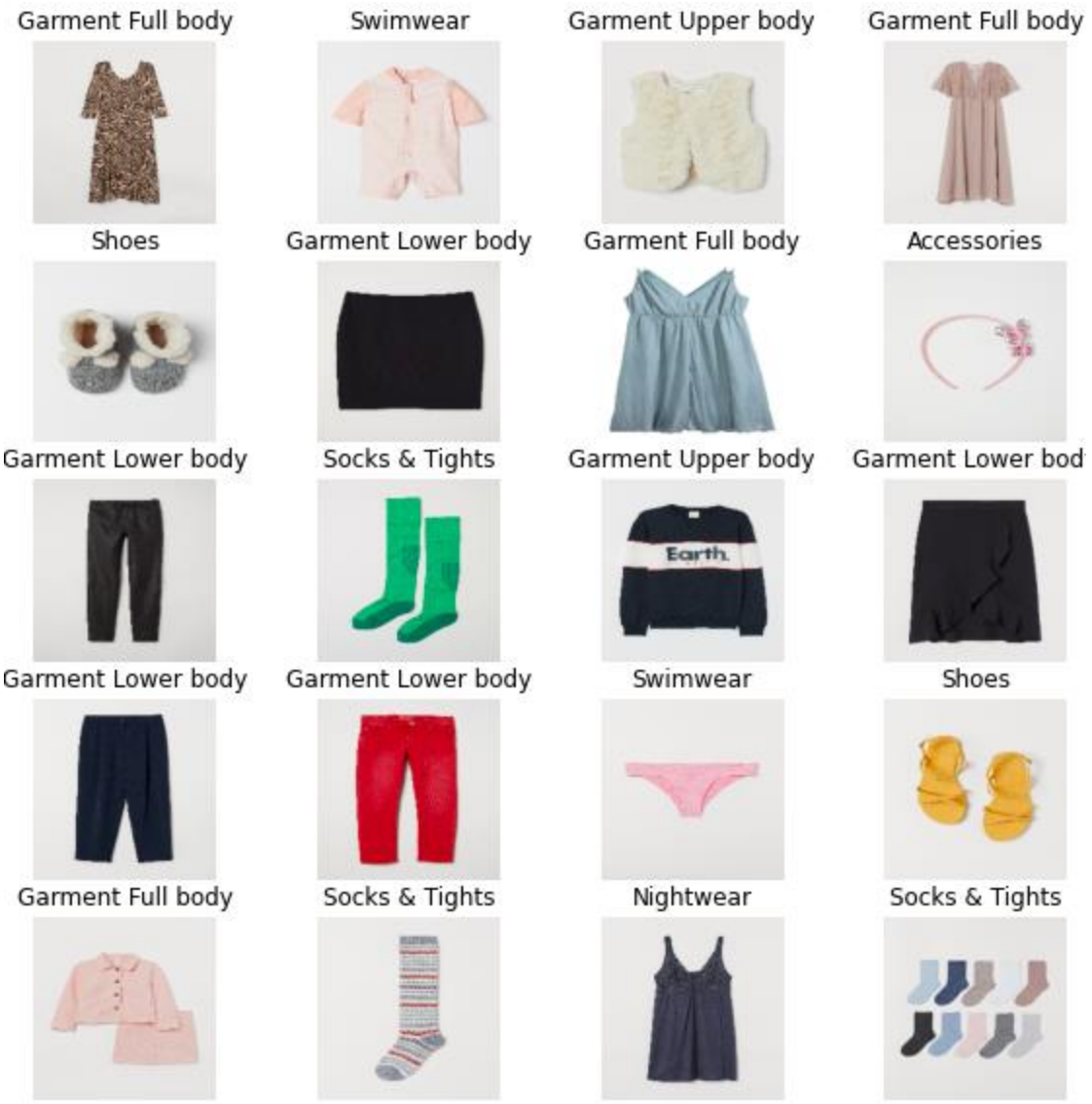
Avant groupement



Apres groupement

```
import shutil,os

#Construction du nouvelle dataset, dont chaque dossier va representer une classe.
for index, row in labelled_images.iterrows():
  if not os.path.exists("dataset/"+row['category']):
    os.makedirs("dataset/"+row['category']) #Creation du repertoire

  if os.path.exists(row['image_path']):
    shutil.copy2(row['image_path'],"dataset/"+row['category']) #Copier les images dans le dossier
```

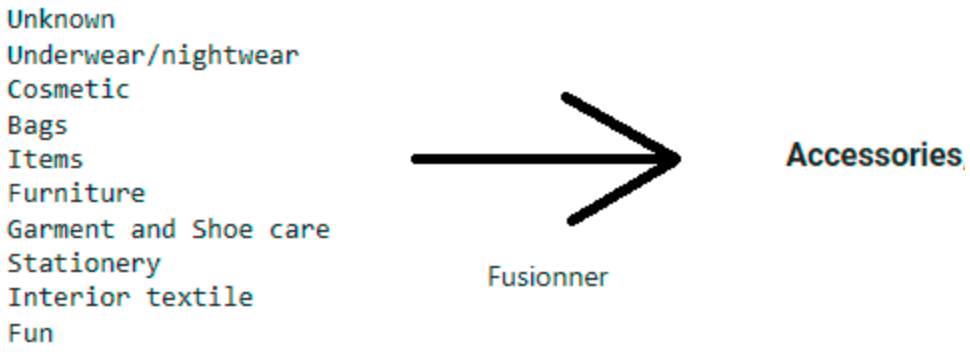
Code de groupement



Produit apres groupement

- ***Fusionner des classes*** :

On vise à fusionner les classes qui sont de thématiques similaires, pour réduire le nombre de classes à prédire, ces classes qu'on veut fusionner sont:

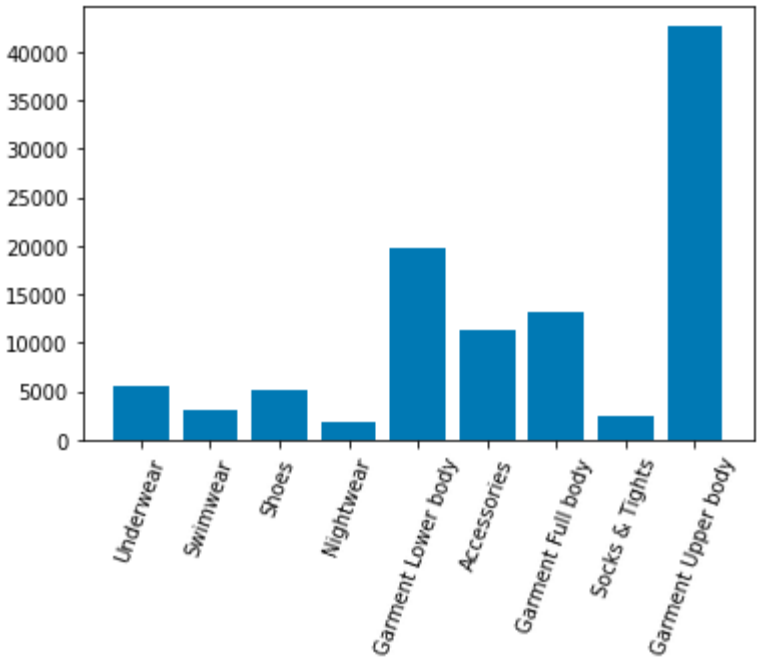


La classe cible est *"**Accessories**"* puisque toutes les classes précèdent peuvent être classées dans la classe ***Accessories***.

```
labels_to_combine=["Unknown","Underwear/nightwear","Cosmetic","Bags","Items",

for labels in labels_to_combine:
  for image in os.listdir("dataset/"+labels):

    #On deplace les fichiers vers la classe Accessoirs
    os.replace("dataset/"+labels+"/"+image,"dataset/Accessories/"+image)

#On supprime les dossiers des classes fusionne
for labels in labels_to_combine:
  if os.path.exists("dataset/"+labels):
    shutil.rmtree("dataset/"+labels)
```

Code du fusionnement

On voit que la distribution des classes (nombre d'image par classe) est encore en déséquilibre. donc il faut un sous-échantillonnage.



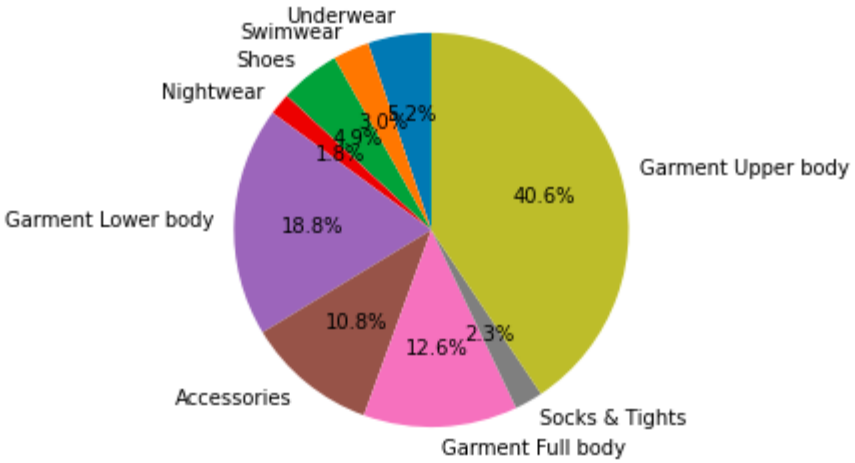
Distribution après fusionnement

- ***Sous-Échantillonnage (Undersampling)***:

On observe que encore le nombre des images par classes est très déséquilibrée:

```
Nombre d'image par classe apres fusionnement est :
Underwear  :  5462
Swimwear  :  3125
Shoes  :  5156
Nightwear  :  1898
Garment Lower body  :  19770
Accessories  :  11302
Garment Full body  :  13276
Socks & Tights  :  2431
Garment Upper body  :  42680
```

Nombre d'image par classe



% d'image par classe

Ce déséquilibre qui va influencer la précision du modèle et augmenter les chances d'avoir un sur apprentissage (Overfitting) .

Pour éviter ce problème, on va faire un ***sous-échantillonnage (Undersampling)***, ce dernier fonctionne par réduire le nombre d'images dans les classes dominantes jusqu'à ce que toutes les classes soient plus ou moins bien équilibrées.

Les classes qu'on va sous-échantillonner sont:

```
Garment Lower body  :  19770
Accessories  :  11302
Garment Full body  :  13276
Socks & Tights  :  2431
Garment Upper body  :  42680
```

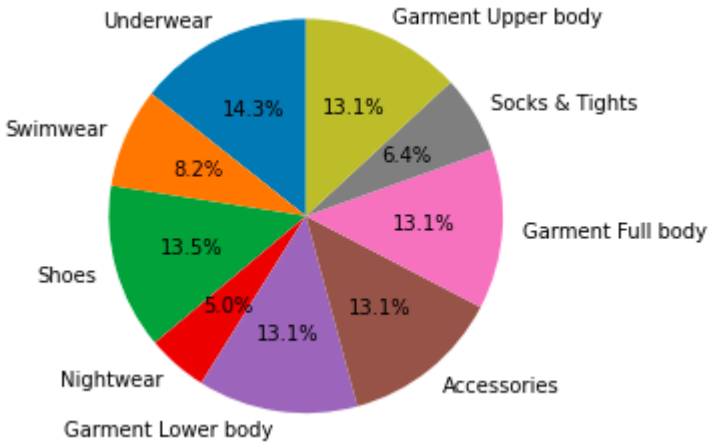
classes dominantes

la taille cible est 5000 images par classe comme un maximum

➔ *Code du sous-échantillonnage:*

```
import os

#Classes majorantes
large_datasets=["Garment Upper body","Garment Full body","Accessories","Garment Lower body"]

for large in large_datasets:
  if os.path.exists("dataset/"+large):
    files=os.listdir("dataset/"+large)

    for image in files:
      if len(os.listdir("dataset/"+large))>5000: #On supprime des images jusqu'a chaque classe contient un max de 5000
        os.remove("dataset/"+large+"/"+image)
```

➔ *Distribution après sous-échantillonnage:*



➔ *Nombre d'image par classes après sous-échantillonnage:*

```
Underwear  :  5462
Swimwear  :  3125
Shoes  :  5156
Nightwear  :  1898
Garment Lower body  :  5000
Accessories  :  5000
Garment Full body  :  5000
Socks & Tights  :  2431
Garment Upper body  :  5000
```

Maintenant que notre dataset est bien traité, on peut commencer la partie de séparation des données, normalisation, encoding , etc

IV. Séparation des données:

Dans cette section, on va s'intéresser par la séparation des données en deux échantillons, un pour l'apprentissage et l'autre pour évaluation du modèle.

Pour faire ca, on va utiliser la fonction, qui un outil de keras cree specifiquement pour manipuler des dataset des images:

`tensorflow.keras.utils.image_dataset_from_directory()`

Cette fonction est très puissante puisqu'il s'occupe non pas juste de la séparation des données, mais aussi à :

- ***Encoding*** (Convertir le nom des classes en nombre entier)
- ***Création des Batch*** (mini-échantillon des données de traitement et évaluation)
- ***Redimensionnement des images***

➔ ***Création des données d'apprentissage:***

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  "dataset", #dossier contenant les classes et leurs images
  validation_split=0.2, #pourcentage de donnees pour validation
  subset="training", #Specifier que cette data est pour l'entrainement
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Training data :
Found 38072 files belonging to 9 classes.
Using 30458 files for training.
```

"***dataset***" *: dossier contenant les classes et les images.*

"***subset***"*: spécifie le type d'échantillon qu'on veut créer, dans ce cas, training.*

*"**image_size**": nouvelle dimension des images (redimensionnement).*

*"**validation_split**": % de données pour évaluation de chaque batch.*

➔ *Création des données d'évaluation:*

```
val_ds = tf.keras.utils.image_dataset_from_directory(
  "dataset",
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Testing data :
Found 38072 files belonging to 9 classes.
Using 7614 files for validation.
```

On voit qu'on 9 classes final pour entraîner le modèle.

V. Normalisation:

Dans cette section, on s'intéresse à normaliser les images, afin de faire ca, on va diviser la matrice d'image par 255.

Puisque chaque pixel value de notre matrice d'image est comprise entre 0 et 255 (RGB), Pour la normaliser on la divise par 255.

Le resultat est que chaque image se convertit a une matrice des pixel comprises entre 0 et 1.

```
[ ]  normalization_layer = layers.Rescaling(1./255)

     normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
     image_batch, labels_batch = next(iter(normalized_ds))
     first_image = image_batch[0]
     # Notice the pixel values are now in `[0,1]`.
     print(np.min(first_image), np.max(first_image))

     0.17641416 0.9399995
```

le resultat nous donne des valeurs entre 0.176 et 0.939 qui est inclut [0,1]

VI. Réseau Neuronal Convulsif (CNN)

Dans cette partie, nous allons nous focaliser sur un des algorithmes les plus performants du Deep Learning, les Convolutional Neural Network ou CNN : Réseaux de neurones convolutifs en français, ce sont des modèles de programmation puissants permettant notamment la reconnaissance d'images en attribuant automatiquement à chaque image fournie en entrée(Une image numérique est une représentation binaire de données visuelles), une étiquette correspondant à sa classe d'appartenance.

1. ARCHITECTURE D'UN CONVOLUTIONAL NEURAL NETWORK-CNN

Leur mode de fonctionnement est à première vue simple : l'utilisateur fournit en entrée une image sous la forme d'une matrice de pixels.

Celle-ci dispose de 3 dimensions :

- Deux dimensions pour une image en niveaux de gris.
- Une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales (Rouge, Vert, Bleu).

Contrairement à un modèle MLP (Multi Layers Perceptron) classique qui ne contient qu'une partie classification, l'architecture du Convolutional Neural Network dispose en amont d'une partie convolutive et comporte par conséquent deux parties bien distinctes :

➢ **Une partie convolutive :** Son objectif final est d'extraire des caractéristiques propres à chaque image en les compressant de façon à réduire leur taille initiale. En résumé, l'image fournie en entrée passe à travers une succession de filtres, créant par la même occasion de nouvelles images appelées cartes de convolutions. Enfin, les cartes de convolutions obtenues sont concaténées dans un vecteur de caractéristiques appelé code CNN.

Tout d'abord, à quoi sert la convolution ?

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Sur une image, son effet s'assimile à un filtrage dont voici le fonctionnement :

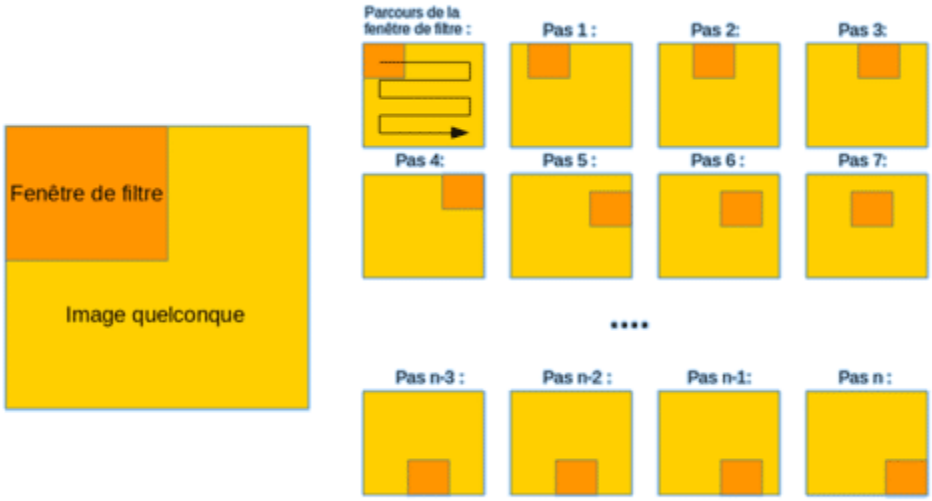


Schéma du parcours de la fenêtre de filtre sur l'image

- Dans un premier temps, on définit la taille de la fenêtre de filtre située en haut à gauche.
- La fenêtre de filtre, représentant la feature, se déplace progressivement de la gauche vers la droite d'un certain nombre de cases défini au préalable (le pas) jusqu'à arriver au bout de l'image.
- À chaque portion d'image rencontrée, un calcul de convolution s'effectue permettant d'obtenir en sortie une carte d'activation ou feature map qui indique où est localisées les features dans l'image : plus la feature map est élevée, plus la portion de l'image balayée ressemble à la feature.

➢ **Une partie classification :** Le code CNN obtenu en sortie de la partie convolutive est fourni en entrée dans une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche (MLP

pour Multi Layers Perceptron). Le rôle de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image. Pour revenir sur cette partie, n'hésitez pas à consulter l'article sur le sujet .

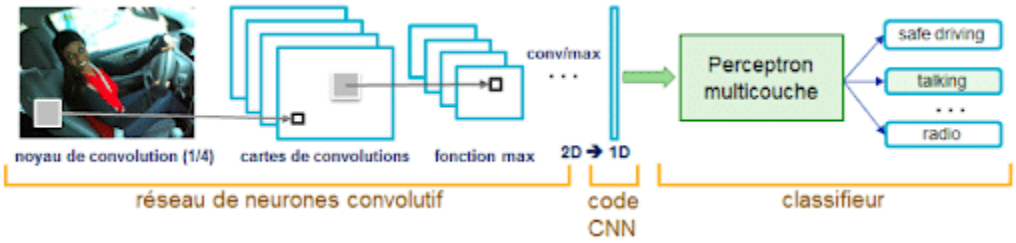
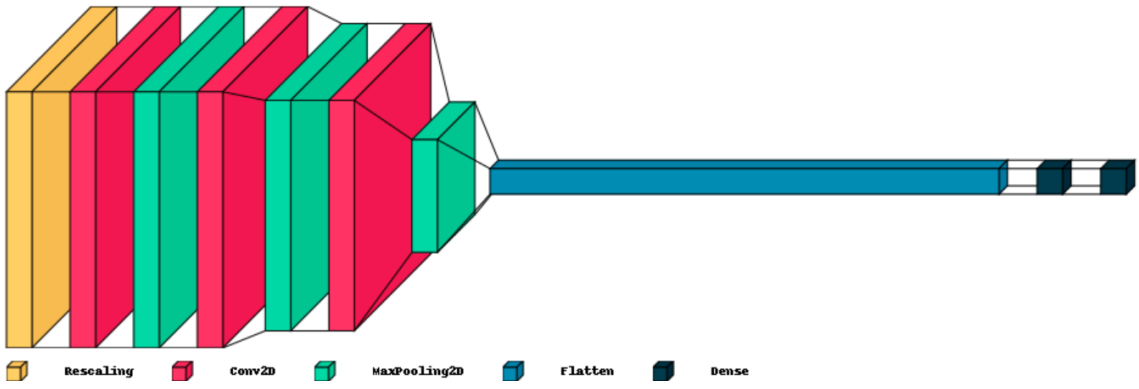


Schéma représentant l'architecture d'un CNN

2. ARCHITECTURE DE NOTRE MODEL:

Notre reseau est construit par les couches suivants:

- 3 couches de convolution (filtrage d'image)
- 3 couches de maxpooling (Reduction de dimensions)
- 1 couche pour transformer la matrice 2D d'output en list 1D
- 1 couche Dense, represente les classes des images input.



```
num_classes = len(class_names)

model = Sequential([
  # data_augmentation,
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)), #couche pour normalisation

  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),

  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),

  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),

  layers.Flatten(),

  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)

])
```

Pour l'optimizer, on a choisit Adam Optimizer

```
[ ]  model.compile(optimizer='adam',
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])
```

resultats:

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_5 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_15 (Conv2D)          (None, 180, 180, 16)      448

 max_pooling2d_15 (MaxPoolin  (None, 90, 90, 16)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 90, 90, 32)        4640

 max_pooling2d_16 (MaxPoolin  (None, 45, 45, 32)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 45, 45, 64)        18496

 max_pooling2d_17 (MaxPoolin  (None, 22, 22, 64)       0
 g2D)

 flatten_2 (Flatten)         (None, 30976)             0

 dense_10 (Dense)            (None, 128)               3965056

 dense_11 (Dense)            (None, 9)                 1161

=================================================================
Total params: 3,989,801
Trainable params: 3,989,801
Non-trainable params: 0
_________________________________________________________________
```

3. APPRENTISSAGE:

on a utiliser 5 epochs pour entrainer notre dataset, en a trouver comme valeur de training accuracy : **0.9568**

Et une precision de test de valeur : 0.7977

```
[ ]  epochs=5 #Iteration sur dataset complete
     history = model.fit(
       train_ds,
       validation_data=val_ds,
       epochs=epochs
     )

     Epoch 1/5
     952/952 [==============================] - 1354s 1s/step - loss: 1.0337 - accuracy: 0.6606 - val_loss: 0.7026 - val_accuracy: 0.7734
     Epoch 2/5
     952/952 [==============================] - 1334s 1s/step - loss: 0.5642 - accuracy: 0.8187 - val_loss: 0.6051 - val_accuracy: 0.8021
     Epoch 3/5
     952/952 [==============================] - 1329s 1s/step - loss: 0.3584 - accuracy: 0.8839 - val_loss: 0.6591 - val_accuracy: 0.8068
     Epoch 4/5
     952/952 [==============================] - 1323s 1s/step - loss: 0.2136 - accuracy: 0.9307 - val_loss: 0.9345 - val_accuracy: 0.7993
     Epoch 5/5
     952/952 [==============================] - 1322s 1s/step - loss: 0.1372 - accuracy: 0.9568 - val_loss: 1.0475 - val_accuracy: 0.7980
```

4. EVALUATION :

On observe que notre modele a des precisions d'apprentissage et d'evaluation de proche valeurs:

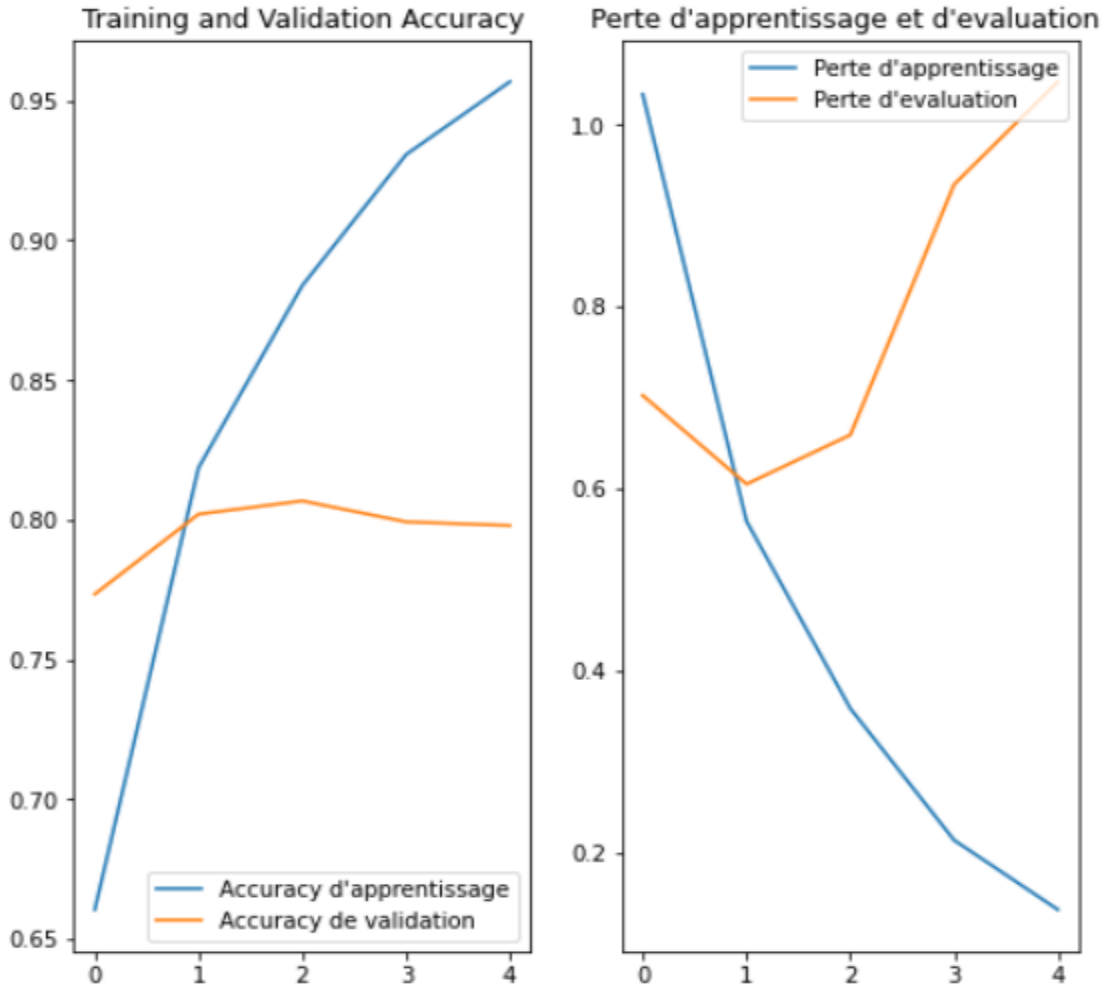
Training accuracy : *0.9826*

Evaluation accuracy : *0.7977*

La similarite entre les deux precisions signifie la bonne conception du modele et que notre modele a bien performe au niveau d'evaluation.

Ce bon performance qui est grace au etape precedent du traitement des donnees (normalisation, sous-echantillonage, etc)

```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label="Accuracy d'apprentissage")
plt.plot(epochs_range, val_acc, label="Accuracy de validation")
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label="Perte d'apprentissage")
plt.plot(epochs_range, val_loss, label="Perte d'evaluation")
plt.legend(loc='upper right')
plt.title("Perte d'apprentissage et d'evaluation ")
plt.show()
```



Conclusion :

Au début de notre projet, on a débuté avec des données qui ont été très bruité , après une analyse des données on a été capables de détecter des problèmes qui auront eu une mauvaise influence sur la précision et la performance de notre modèle.

Après avoir traiter ces problèmes dans la partie traitement des données on était capable d'avoir une dataset bien créé qui a permet le modèle d'apprendre sans les dégâts des bruit.

La partie nécessaire d'un projet machine learning, c'est pas l'entrainement mais c'est tous qui le précèdent, c'est l'analyse et la compréhension des données qui permettent la création d'un bon modèle.

Références :

https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations

https://fastapi.tiangolo.com/

https://www.youtube.com/watch?v=zfiSAzpy9NM

https://www.analyticsvidhya.com/blog/2021/06/image-processing-using-cnn-a-beginners-guide/

https://analyticsindiamag.com/how-to-visualize-deep-learning-models-using-visualkeras/

https://keras.io/api/utils/model_plotting_utils/