

# Classification automatique de biens de consommation

Place de marché



# Contexte et problématique

- L'entreprise « Place de marché » souhaite lancer une marketplace e-commerce.
- Sur cette marketplace anglophone, les vendeurs proposent des articles en postant une photo et une description.
  - ➡ L'attribution de la catégorie de l'article est effectuée **manuellement** par le vendeur, et est donc **peu fiable**.
- **Objectif** : automatiser la tâche de catégorisation des articles

## Mission 1

Etudier la faisabilité d'un moteur de classification des articles en différentes catégories, à partir (a) du texte de description et (b) de l'image

## Mission 2

Réaliser une classification supervisée à partir des images

Watches



Home Decor & Festive Needs



Kitchen & Dining



Baby Care



Beauty and Personal Care



# Présentation des données



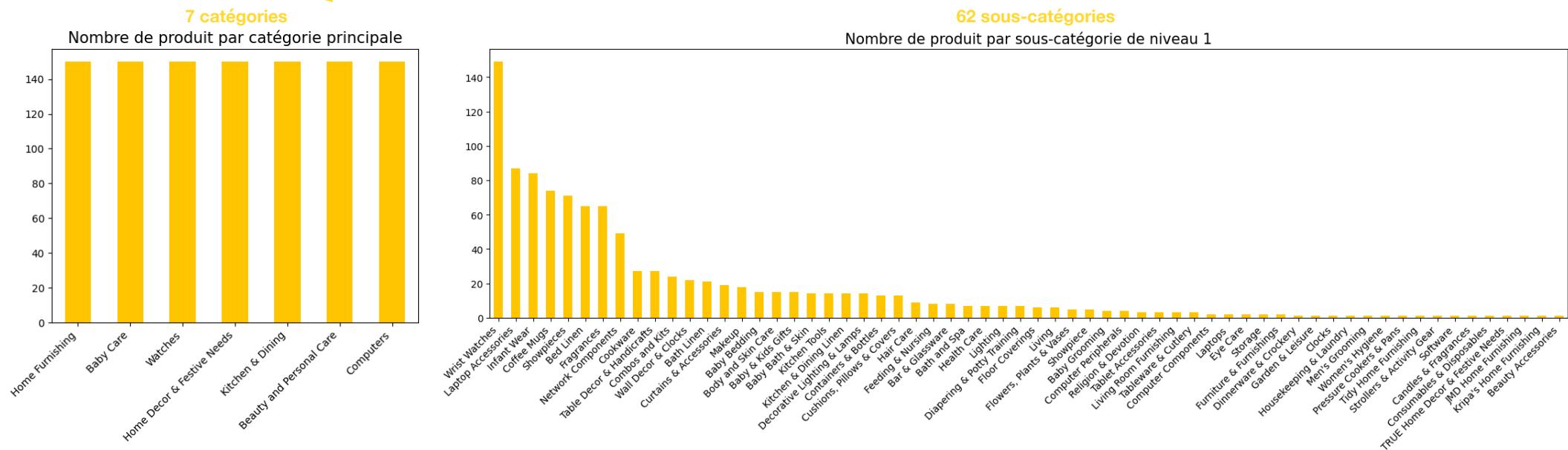
**A noter :** les textes et les images ne sont soumises à aucune contraintes de propriété intellectuelle

# Pré-traitement

- Nous avons extrait les catégories (et sous-catégories) d'appartenance de chaque article de la feature « product\_category\_tree » :

	product_name	product_category_tree	main_cat	sub_cat_1	sub_cat_2	sub_cat_3
0	Elegance Polyester Multicolor Abstract Eyelet ...	["Home Furnishing >> Curtains & Accessories >>...]	Home Furnishing	Curtains & Accessories	Curtains	Elegance Polyester Multicolor Abstract Eyelet ...

- Les articles sont uniformément répartis entre les 7 catégories principales (150 articles par catégories)



# Mission 1 : étude de faisabilité

Est-il envisageable de classifier les articles sur la base du texte qui les accompagne et/ou de leur photo ?

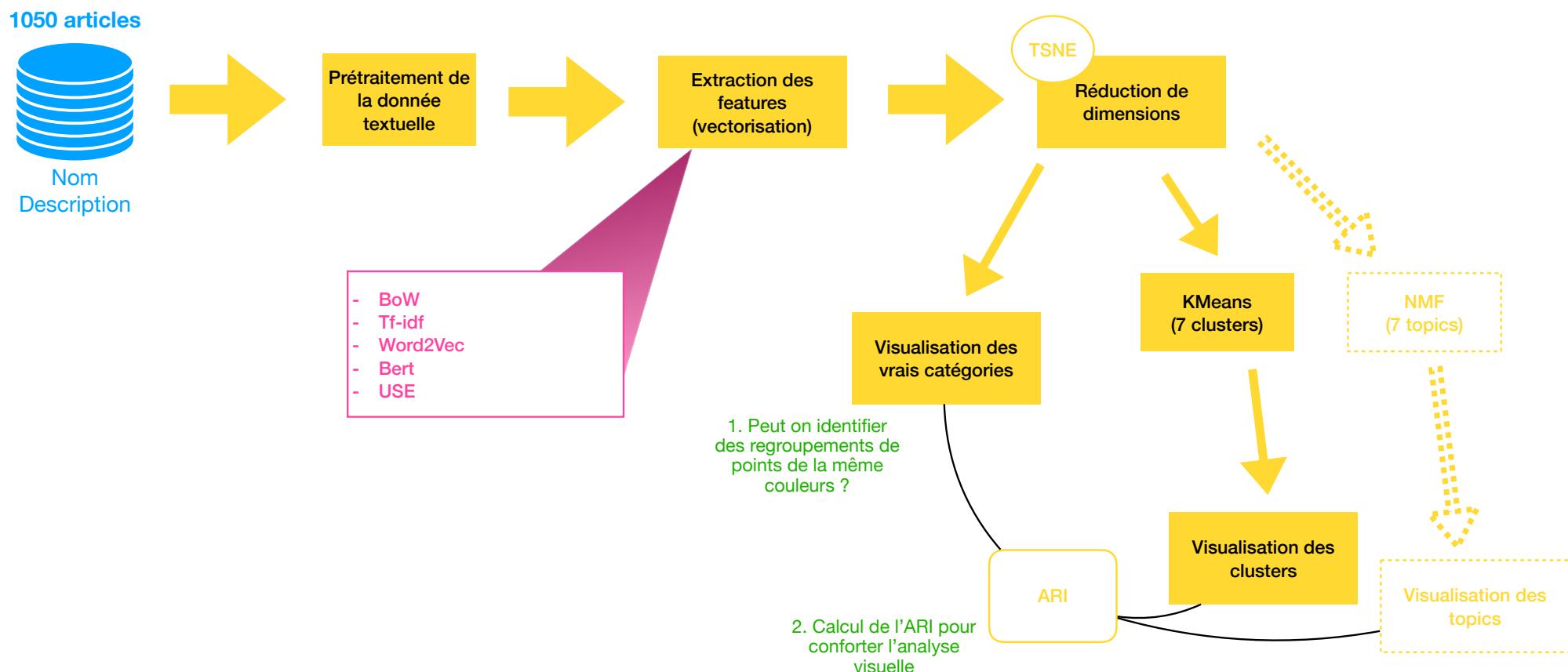
# Faisabilité

## A partir du texte



# Etude de faisabilité

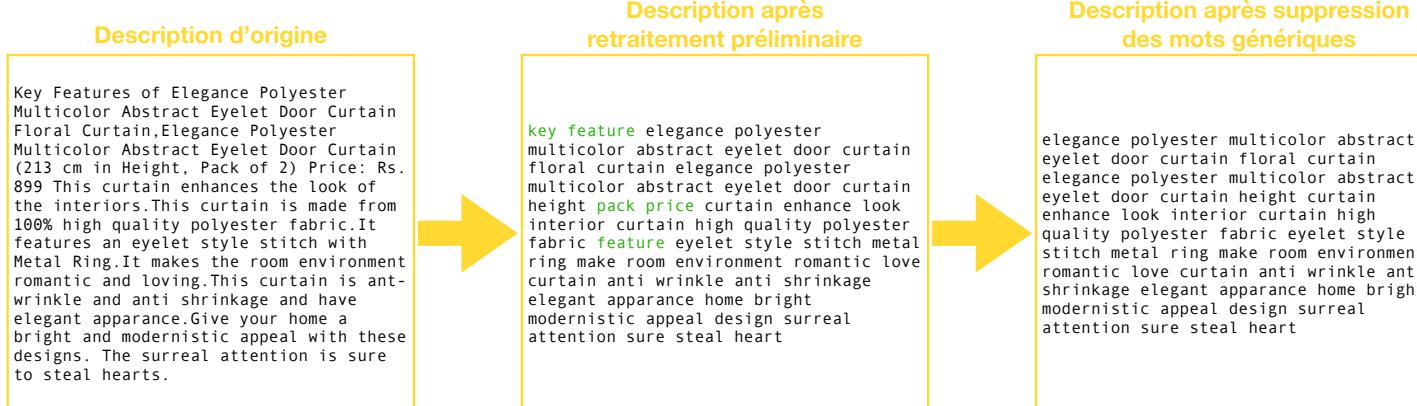
## Méthodologie



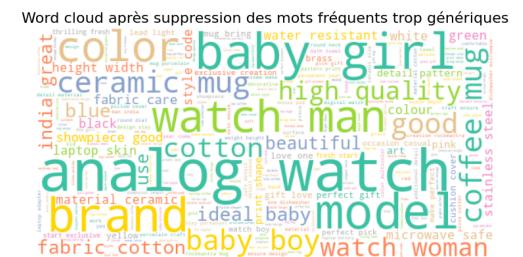
# Etude de faisabilité

## Prétraitement de la donnée textuelle

- Texte analysé : le nom du produit, sa description, la concaténation des deux
- Retraitements préliminaires:
  - mise en minuscule,
  - suppression des chiffres et de la ponctuation,
  - suppression des « stop words » et des mots trop courts ( $\leq 2$  caractères),
  - suppression des expressions spéciales (« 1x », « rs25 », « ant- »....),
  - Tokenisation & Lemmatisation.
- Création d'une liste de mots à supprimer sur la base des mots les plus fréquents (« product », « free », « genuine »...) : suppression des mots qui ne permettent pas de différencier les articles entre eux.



8



# Etude de faisabilité

## Feature extraction - type « Bag of Words »

Les différents algorithmes ont été testés 3 fois chacun : sur le nom de l'article, sur la description de l'article et sur la concaténation du nom des de la description\*

### BoW (Bag of Words)

L'algorithme compte le nombre de fois qu'un mot apparaît dans le document, sans considération de l'ordre ou de la structure des mots. Le nombre de colonne correspond au nombre de mot unique dans le corpus.

Les vecteurs sont « sparse ».

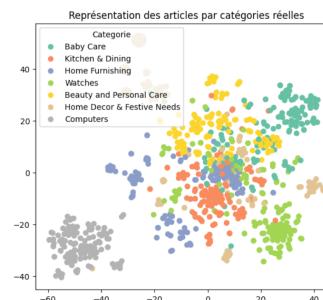
### Vecteurs

1050 x 4178

Réduction des vecteurs à 2 dimensions pour pouvoir les visualiser

TSNE

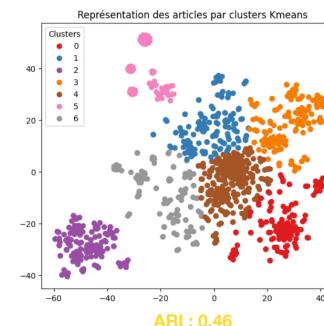
n\_components = 2  
perplexity = 30



Kmeans sur les données réduites pour comparer clusterisation avec vrais cat.

Kmeans

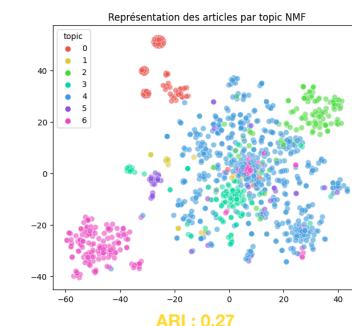
n\_clusters = 7



NMF sur les données réduites pour comparer les topics avec vrais cat.

NMF

n\_components = 7

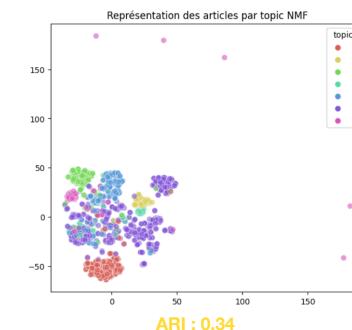
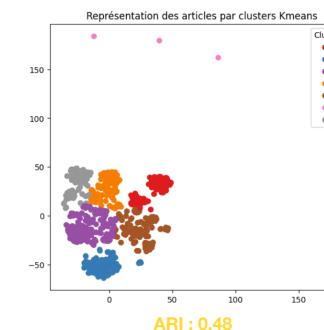
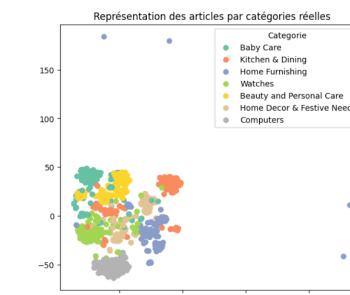


### Tf-iDf (Term Frequency - Inverse Document Frequency)

L'algorithme vectorise un mot en multipliant

- sa fréquence (nb de fois où le mot apparaît dans le document / nombre total de mot dans le document) par
- l'inverse de la proportion de document qui contient le mot (le log du nombre de document dans le corpus / nombre de document du corpus avec ce mot).

1050 x 4178



\* Sur ce slide, nous illustrons les résultats de la démarche adoptée sur *la description uniquement* (cf. Slide 12 pour le détail de tous les résultats obtenus)

**Légende taille des vecteurs**  
Taille qui dépend du corpus  
Taille choisie par l'utilisateur  
Taille qui dépend de l'architecture de l'algo

# Etude de faisabilité

## Détail du topic modeling (NMF)

Topic 0:	
mug	8.023586
coffee	3.328503
ceramic	2.455268
perfect	2.449351
gift	2.207575
design	1.735434
love	1.642410
bring	1.162455
tea	1.025262
material	0.937477

Kitchen & Dining

Topic 1:	
adapter	5.240313
vaiō	2.627981
vgn	2.627981
power	2.521210
laptop	2.478924
smartpro	2.248326
charger	2.004311
design	1.575760
output	1.324026
series	1.308471

??

Topic 2:	
baby	3.758510
girl	2.381774
detail	2.176432
cotton	1.943086
fabric	1.895619
dress	1.564874
boy	1.163322
sleeve	1.111109
print	1.079762
neck	1.037180

Baby Care

Topic 3:	
skin	3.798239
laptop	3.528815
print	1.692620
shape	1.604625
mouse	1.300438
pad	1.285131
combo	1.088329
multicolor	0.987842
size	0.787750
easy	0.733558

Computers

Topic 4:	
color	2.070613
cover	1.741055
material	1.503513
cotton	1.384283
design	1.230117
model	1.148642
width	1.141237
sheet	1.137382
bedsheet	1.110401
brand	0.986366

Home Furnishing

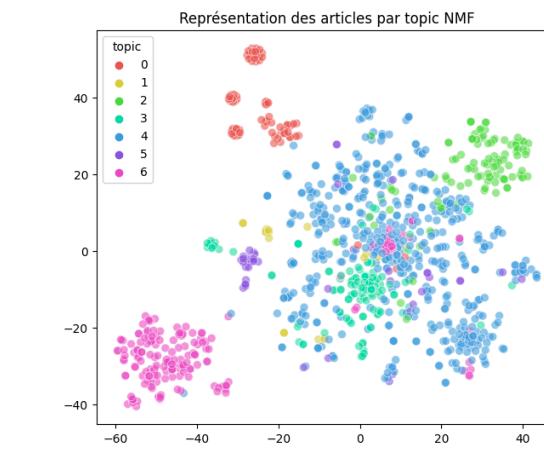
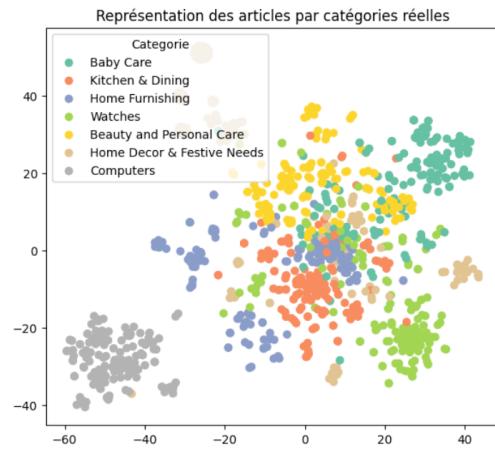
Topic 5:	
battery	4.606284
quality	3.105342
high	1.793687
lapguard	1.439176
laptop	1.362938
pass	1.169241
include	1.115284
cell	1.094130
label	0.831696
performance	0.682688

??

Topic 6:	
watch	5.093808
analog	3.236084
man	2.348541
india	1.664823
great	1.633338
woman	1.345691
dial	1.307241
strap	1.272502
water	0.892768
boy	0.810230

Watches

Résultats obtenus sur les vecteurs BoW issus de la description :



Exemple:  
La première description fait référence à un article de la catégorie Home Furnishing et a été affectée au topic 4

# Etude de faisabilité

## Feature extraction - Word embedding

### Word2vec

L'algorithme représente chaque mot par un vecteur unique, indépendamment de la nature et du contenu du corpus étudié.

Les vecteurs sont denses et il est possible de mesurer la similarité sémantique entre les mots.

### BERT (Bidirectional Encoder Representation from Transformers)

L'algorithme représente chaque mot par un vecteur qui est « context aware » (varie en fonction de son contexte), grâce à l'architecture transformer (encoder/decoder).

### USE (Universal Sentence Encoder)

L'algorithme modélise la signification de séquences de mots (phrase) en une représentation vectorielle de taille fixe.

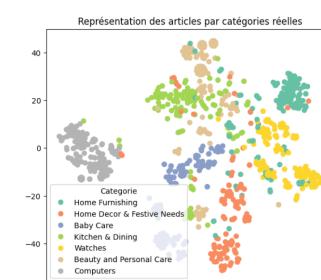
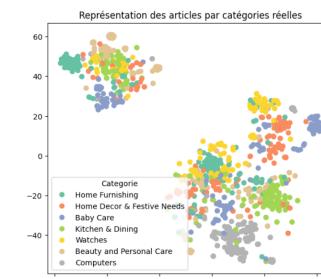
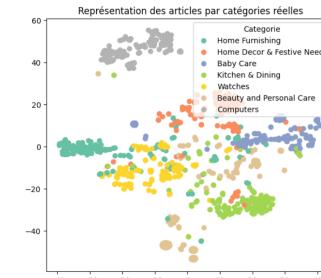
### Vecteurs

1050 x 100

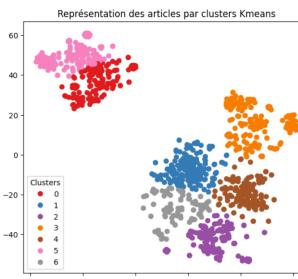
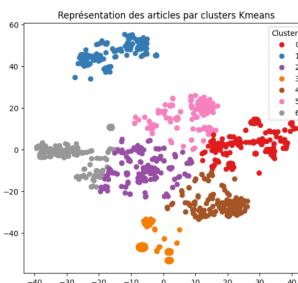
1050 x 768 (1024)

1050 x 512

Réduction des vecteurs à 2 dimensions pour pouvoir les visualiser



Kmeans sur les données réduites pour comparer clusterisation avec vrais cat.



ARI : 0.58

ARI : 0.31

ARI : 0.51

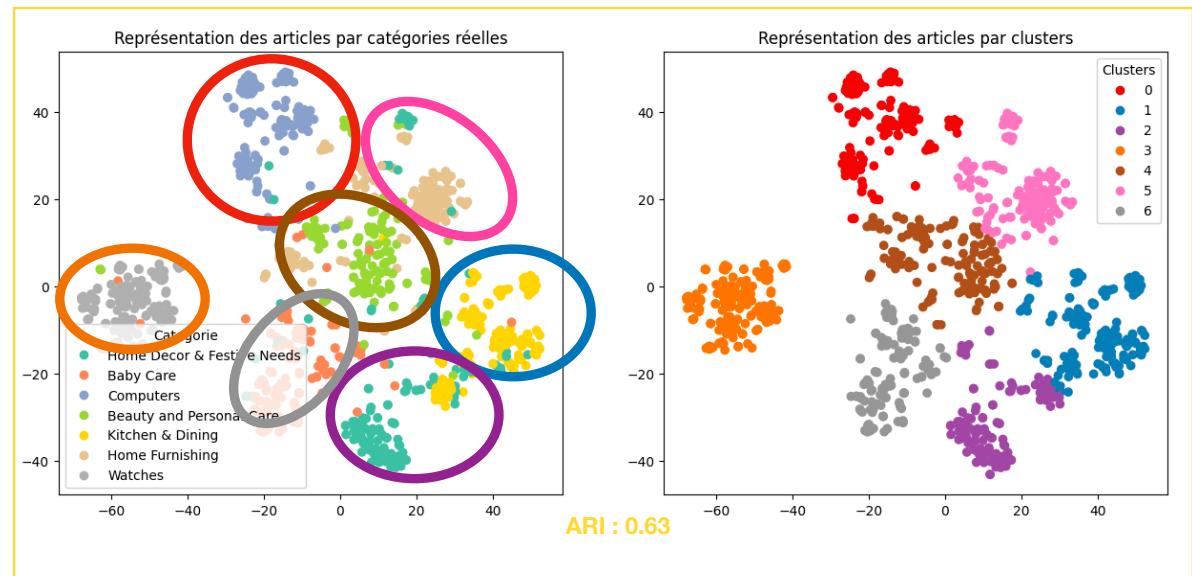
Sur ce slide, nous illustrons les résultats de la démarche adoptée sur *la description uniquement* (cf. Slide 12 pour le détail de tous les résultats obtenus)

# Etude de faisabilité

## Comparaison des résultats

Extraction_type	Data_source	ARI	time	Extraction_source
16	USE	name	0.6332	6.0
				USE -- name
8	W2Vec	name & description	0.5887	5.0
				W2Vec -- name & description
6	W2Vec	description	0.5806	5.0
				W2Vec -- description
17	USE	name & description	0.5780	6.0
				USE -- name & description
5	Tf-idf	name & description	0.5082	6.0
				Tf-idf -- name & description
15	USE	description	0.5078	5.0
				USE -- description
7	W2Vec	name	0.4997	5.0
				W2Vec -- name
3	Tf-idf	description	0.4827	6.0
				Tf-idf -- description
0	BOW	description	0.4557	6.0
				BOW -- description
11	BERT - base	name	0.4542	6.0
				BERT - base -- name
2	BOW	name & description	0.4539	6.0
				BOW -- name & description
1	BOW	name	0.4243	6.0
				BOW -- name
12	BERT - large	name	0.3648	7.0
				BERT - large -- name
14	BERT - large	name & description	0.3616	6.0
				BERT - large -- name & description
4	Tf-idf	name	0.3451	6.0
				Tf-idf -- name
13	BERT - base	name & description	0.3408	6.0
				BERT - base -- name & description
10	BERT - large	description	0.3239	6.0
				BERT - large -- description
9	BERT - base	description	0.3136	6.0
				BERT - base -- description

Détail des résultats obtenus avec le modèle USE  
(sur le nom de l'article)



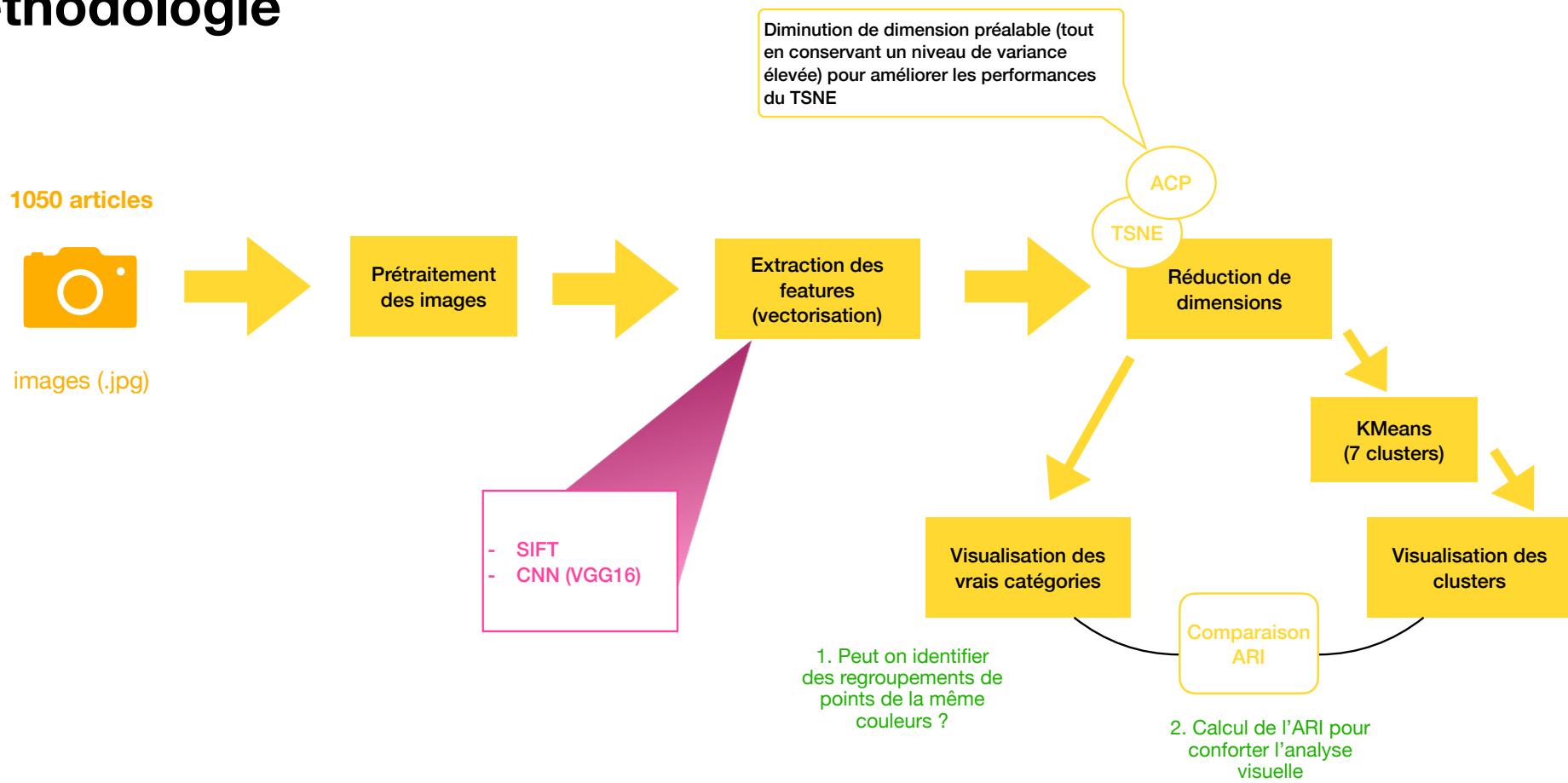
# Faisabilité

## A partir de l'image



# Etude de faisabilité

## Méthodologie



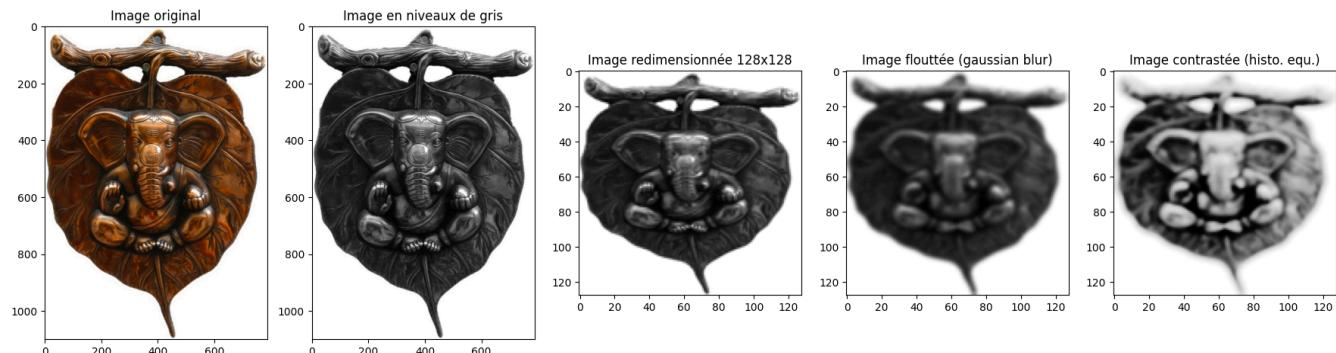
# Etude de faisabilité

## SIFT (« bag of images »)

**SIFT** : algorithme qui détecte les points distinctifs d'une image (ses features) : robustes au changements d'échelle, rotations, transformations affines... Des features qui ne sont pas affectées par la taille ou l'orientation de l'image.

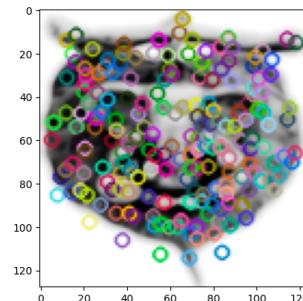
### 1. Prétraitement des images

- Conversion en niveau de gris
- Redimensionnement (128x128)
- Floutage (GaussianBlur 5)
- Égalisation



### 2. Extraction des keypoints / descripteurs par image :

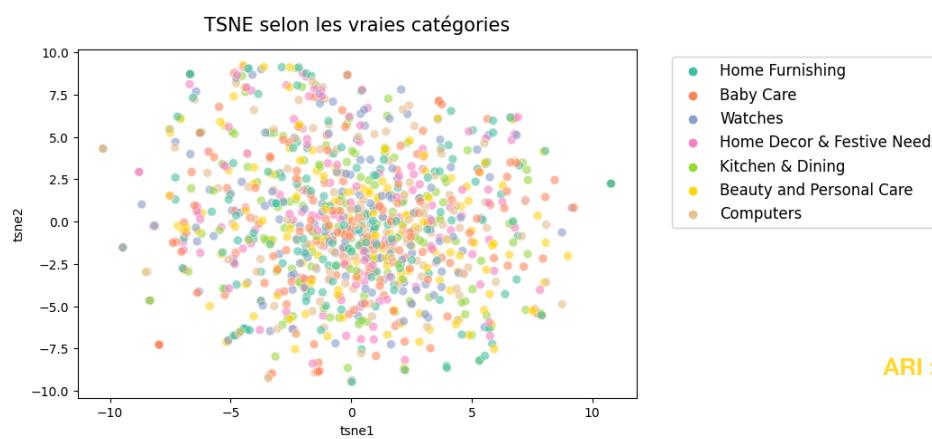
- Chaque image a un nombre de descripteurs qui lui est propre, tous de taille 128
- Les 1050 images sont représentées par un total de 134 689 keypoints



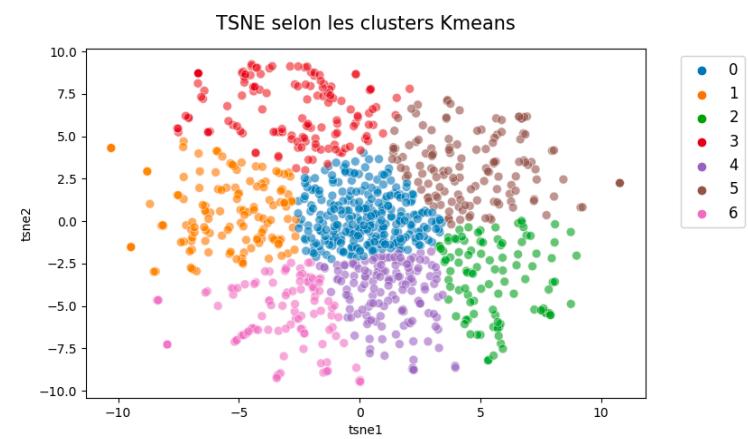
# Etude de faisabilité

## SIFT (« bag of images »)

3. Utilisation de l'algorithme Kmeans pour créer des clusters de descripteurs\* : regroupement des descripteurs proche entre eux
4. Création d'un histogramme par image (« feature vector ») : comptage du nombre de descripteur par cluster pour chaque image (matrice 1050 x 367\*)
5. Réduction de dimensionnalité
  - ACP préliminaire pour diminuer le nombre de features tout en conservant une variance élevé (99%) : 326 features au lieu de 367
  - TSNE à 2 composants pour pouvoir afficher en 2D les articles
6. Clusterisation des données réduites avec Kmeans (7 clusters) et calcul de l'ARI pour mesurer la similarité entre les catégories réelles et les clusters issus du Kmeans



ARI : 0



\* rule of thumb : 367 cluster, racine carré du nombre total de descripteurs

# Etude de faisabilité

## CNN (Transfer Learning)

**Transfer Learning** : réutilisation d'un modèle pré-entraîné pour capitaliser sur son apprentissage.

- La vision par ordinateur bénéficie particulièrement du transfer learning dans la mesure où entraîner de tels modèles requiert une puissance de calcul importante : il est fastidieux d'entraîner un modèle sur des jeux de données comportant des millions d'images.
- 38 modèles CNN pré-entraînés sur la base de données ImageNet sont disponibles via Keras.
- ImageNet est une base de données contenant plus de 14 millions d'images annotées appartenant à 1000 classes différentes.

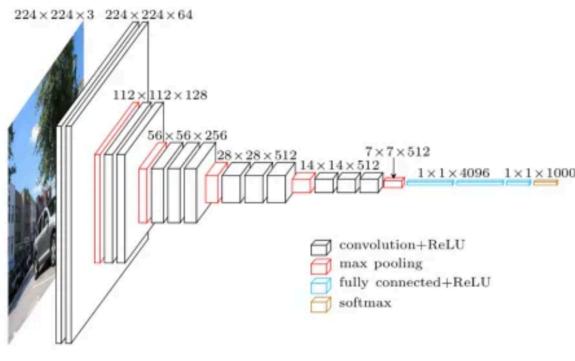
Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5

- VGG16 est une architecture de CNN populaire, proposée par l'Université de Oxford en 2014 (2ème prix de la classification d'image du ILSVRC cette année-là).

# Etude de faisabilité

## CNN - VGG16



```
base_model = VGG16(weights="imagenet")
model = Model(inputs=base_model.inputs, outputs=base_model.layers[-2].output)
```

Suppression de la dernière couche du modèle qui est adaptée à la tâche initiale du VGG16 : classification d'images dans 1000 classes différentes

### 1. Preprocessing des images :

- Redimensionnement  $224 \times 224$  (taille de l'input du VGG16)
- Ajout d'une dimension pour matcher le format de Keras (number, height, width, channel)
- Retraitements spécifiques à VGG16 via la fonction adaptée (« preprocess\_input ») : les images sont converties au format BGR (plutôt que RGB), puis chaque canal de couleur est centré sur 0 par rapport à l'ensemble des données ImageNet, sans mise à l'échelle

### 2. Utilisation du modèle pour créer un embedding uniquement (pas pour classifier) : matrice $1050 \times 4096$

\* source image : « Very Deep Convolutional Networks for Large-Scale Image Recognition », K. Simony et A. Zisserman, University of Oxford (2014)

Model: "vgg16"		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[None, 224, 224, 3]	0
block1_conv1 (Conv2D)	[None, 224, 224, 64]	1792
block1_conv2 (Conv2D)	[None, 224, 224, 64]	36928
block1_pool (MaxPooling2D)	[None, 112, 112, 64]	0
block2_conv1 (Conv2D)	[None, 112, 112, 128]	73856
block2_conv2 (Conv2D)	[None, 112, 112, 128]	147584
block2_pool (MaxPooling2D)	[None, 56, 56, 128]	0
block3_conv1 (Conv2D)	[None, 56, 56, 256]	295168
block3_conv2 (Conv2D)	[None, 56, 56, 256]	590080
block3_conv3 (Conv2D)	[None, 56, 56, 256]	590080
block3_pool (MaxPooling2D)	[None, 28, 28, 256]	0
block4_conv1 (Conv2D)	[None, 28, 28, 512]	1180160
block4_conv2 (Conv2D)	[None, 28, 28, 512]	2359808
block4_conv3 (Conv2D)	[None, 28, 28, 512]	2359808
block4_pool (MaxPooling2D)	[None, 14, 14, 512]	0
block5_conv1 (Conv2D)	[None, 14, 14, 512]	2359808
block5_conv2 (Conv2D)	[None, 14, 14, 512]	2359808
block5_conv3 (Conv2D)	[None, 14, 14, 512]	2359808
block5_pool (MaxPooling2D)	[None, 7, 7, 512]	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
<b>predictions (Dense)</b>	<b>(None, 1000)</b>	<b>4097000</b>

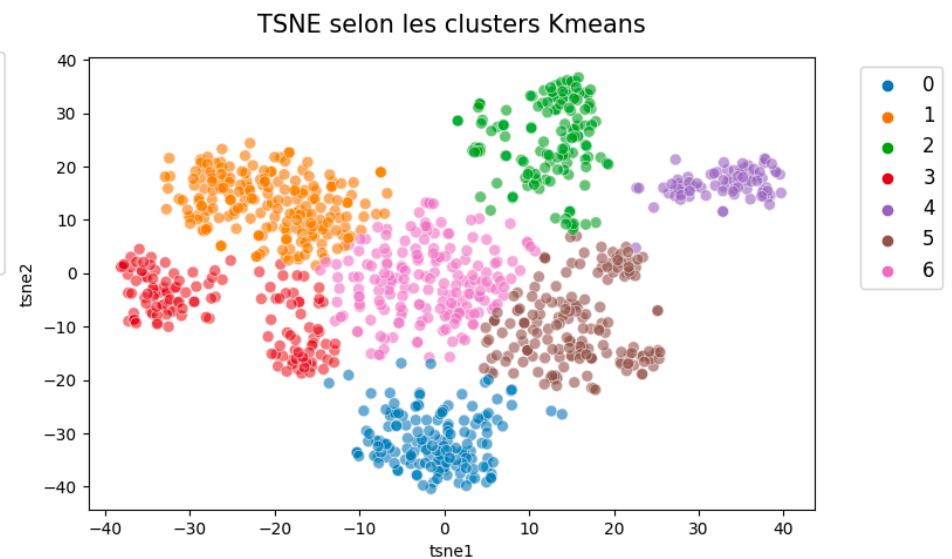
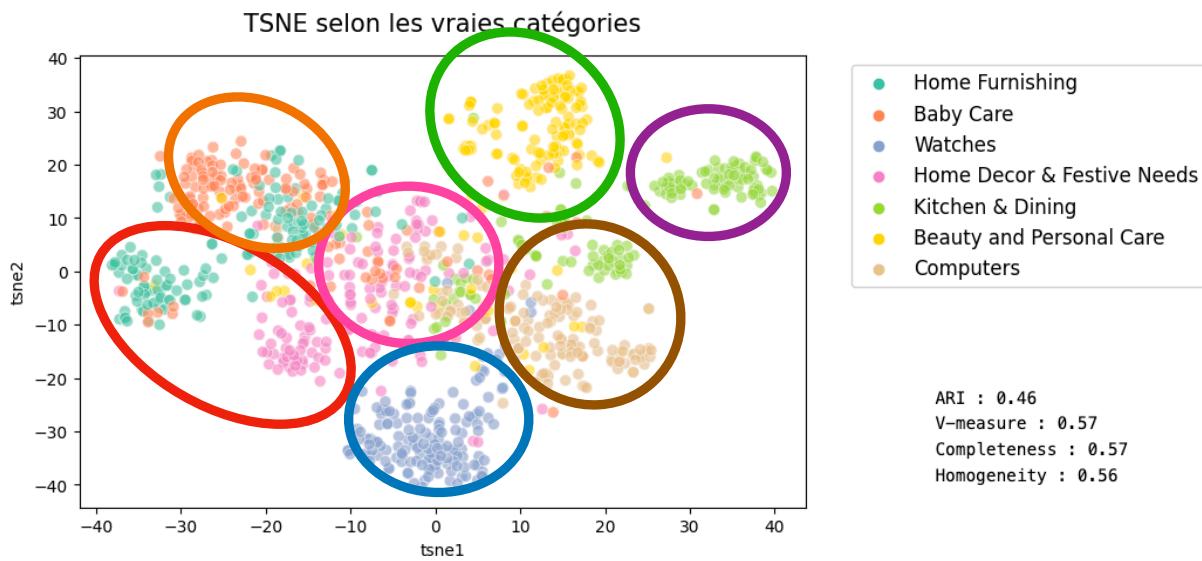
# Etude de faisabilité

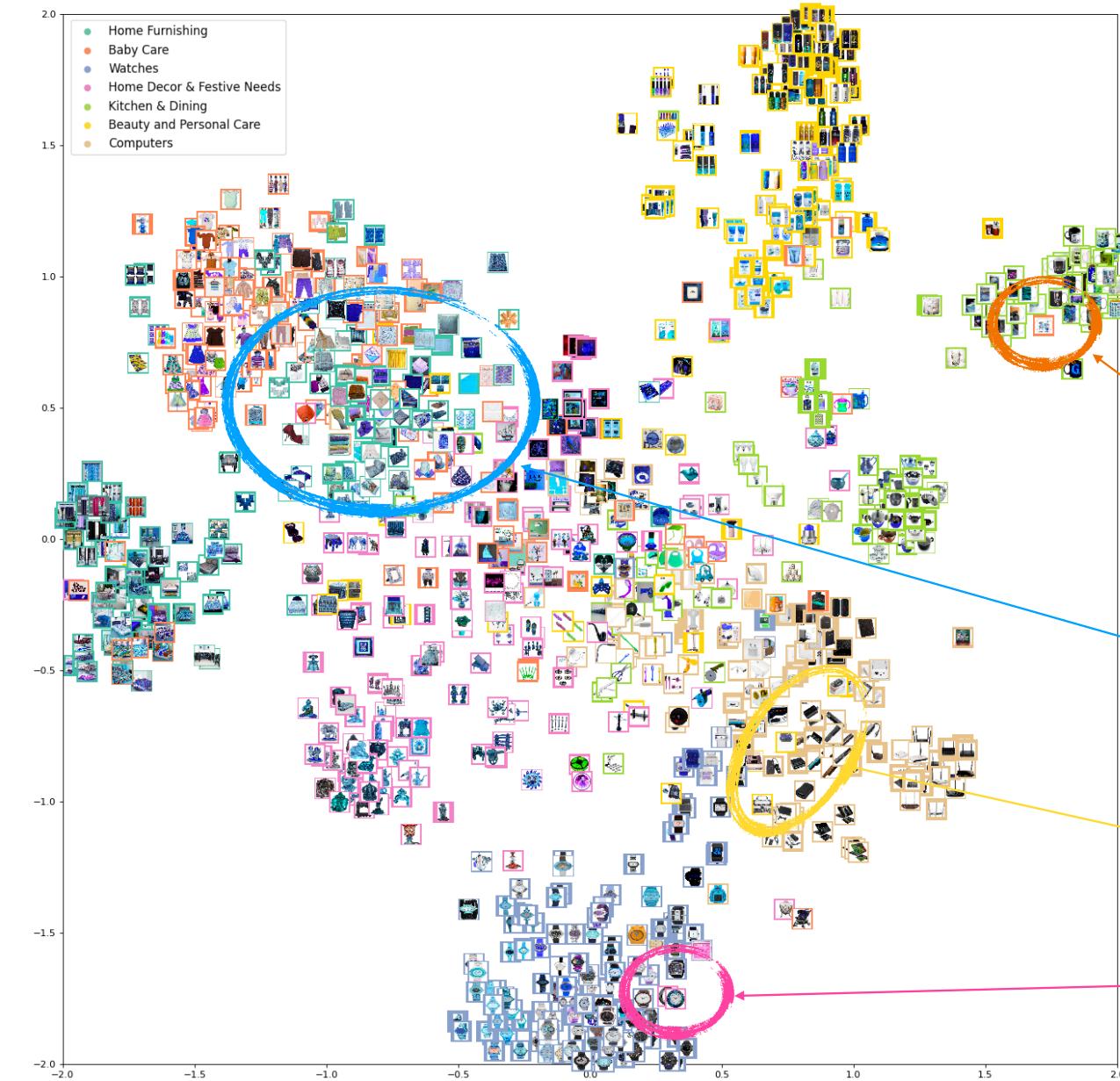
## CNN - VGG16

### 3. Réduction de dimensionnalité

- ACP préliminaire pour diminuer le nombre de features tout en conservant une variance élevé (99%) : 803 features au lieu de 4096
- TSNE à 2 composants pour pouvoir afficher en 2D les articles

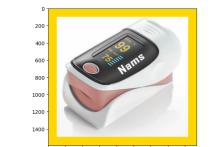
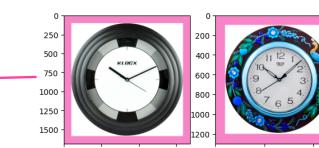
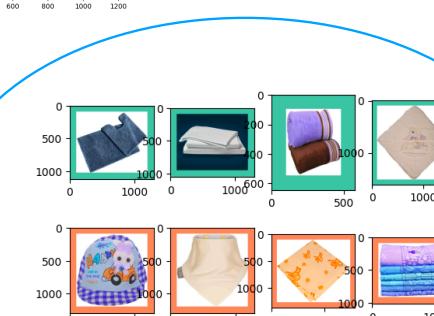
### 4. Clusterisation des données réduites avec Kmeans (7 clusters) et calcul de l'ARI pour mesurer la similarité entre les catégories réelles et les clusters issus du Kmeans (et conforter l'analyse visuelle)





Matrix de confusion de la classification à KMEANS

	1	2	3	4	5	6	7	8
Baby Care	110.0	5.0	1.0	24.0	8.0	1.0	1.0	1.0
Beauty and Personal Care	5.0	118.0	8.0	12.0	6.0	1.0	0.0	0.0
Computers	1.0	0.0	113.0	35.0	0.0	0.0	0.0	1.0
Home Decor & Festive Needs	2.0	1.0	6.0	81.0	52.0	0.0	0.0	8.0
Home Furnishing	79.0	0.0	0.0	4.0	67.0	0.0	0.0	0.0
Kitchen & Dining	0.0	12.0	34.0	25.0	0.0	79.0	0.0	0.0
Watches	0.0	0.0	11.0	2.0	0.0	0.0	0.0	137.0



# Mission 2 : classification supervisée

Classification des articles sur la base de leur image

# Classification supervisée

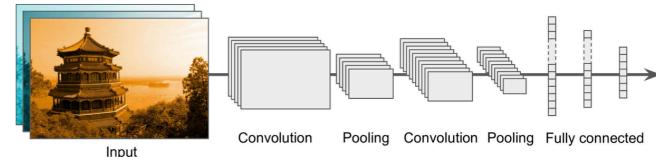
## Introduction

### Problématique machine learning

- Apprentissage supervisé : les observations du training set sont labélisées
- Classification en classe multiples : il s'agit de répartir les articles entre 7 catégories
- Donnée source : image (.jpg)

Les **Convolutional Neural Networks** (CNN), qui sont nés de l'étude du cortex visuel du cerveau, sont utilisés pour la reconnaissance d'image depuis les années 80.

- Architecture typique d'un CNN :



L'image devient de plus en plus petite au fil de sa traversé du réseau, mais également plus profondes (feature maps des couches convolutionnelles)

- Couche convolutionnelle
  - Les neurones de la 1ère couche convolutionnelle ne sont pas connectés à tous les pixels de l'image input, mais seulement aux pixels qui sont dans leur champs. Les neurones de la 2nde couche convolutionnelle sont connectées aux neurones de la 1ère couche qui sont localisés dans leur champs réduit.
  - Cette architecture permet au réseau de se concentrer sur des « low level features » dans la première couche, puis d'assembler ces features en « higher level features » dans la couche d'après, etc....
- Couche de pooling : permet de « subsample » l'input pour limiter la charge calculatrice, l'utilisation de la mémoire et le nombre de paramètres (limite l'overfitting). Chaque neurone est connecté à l'output d'un nombre limité de neurone de la couche précédente. Pas de poids, l'input est agrégé sur la base d'une fonction (max, mean)

\* source image : « Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow », A. Géron, O'Reilly (2019)

# Classification supervisée

## Introduction

- En s'appuyant sur la technique du transfert learning, nous avons testé 4 architectures CNN, entraînées sur **ImageNet** :

Modèle	Architecture	Size (MB)	Accuracy*
VGG-16	13 conv layers	528	90.1 %
VGG-19	16 conv layers	549	90.0%
Resnet50	49 conv layers	98	92.1%
EfficientNetB0	17 conv layers	29	93.3%

La dernière couche des 4 modèles a été modifiée pour s'adapter à notre problème de classification

- activation : « softmax »
- size : 7 (nombre de classes différentes)

Les résultats de ces modèles ont été comparés à un modèle construit « **from scratch** » (modèle de référence) :

- 3 ensembles conv / relu / pooling
- 1 fully connected layer (relu)
- output avec activation « softmax »

\* top-5 accuracy - performance du modèle sur le set de validation ImageNet

- **Loss function** : fonction utilisée par le GD pour entraîner le modèle
  - ➡ Cross-entropy dans le cas de la classification
- Les **metrics** sont utilisés pour évaluer le modèle.
  - ◆ **Accuracy** : le nombre de fois où le classifieur prédit correctement  
Un bon résultat peut être *misleading* et doit donc être analysé en conjonction avec
  - ◆ **F1 score** : moyenne harmonique de la précision et du rappel
    - La *précision* : taux de prédition correctes parmi les prédictions positives (mesure capacité à ne pas faire d'erreur lors d'une prédition positive)
    - Le *rappel* : taux d'observation positives détectées par le modèle (true positive rate)
  - ◆ **AUC** : mesure de la capacité d'un classifier à faire la distinction entre les classes
  - ◆ **Matrice de confusion** : récapitulatif des erreurs
  - ◆ **Temps d'entraînement**
  - ◆ Différence entre les performances du training set et du validation set pour évaluer la **capacité de généralisation du modèle**

# Classification supervisée

## Démarche et résultats

Split du dataset (1050 articles) en un **training set** (840 articles) et un **test set** (210 articles)

### Approche identique pour tous les modèles :

1. Récupération des couches entraînées et modification de la dernière couche uniquement :

```
predictions = Dense(7, activation='softmax')(x)
```

2. Modalité de compilation similaire :

- Optimizer : RMSprop (learning\_rate = 0.001)
- Loss : « categorical\_crossentropy »
- Metrics : accuracy, AUC, f1

3. Modalité d'entraînement similaire :

```
epochs=25  
batch_size=64  
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)
```

4. Résultats enregistrés avec et sans data augmentation

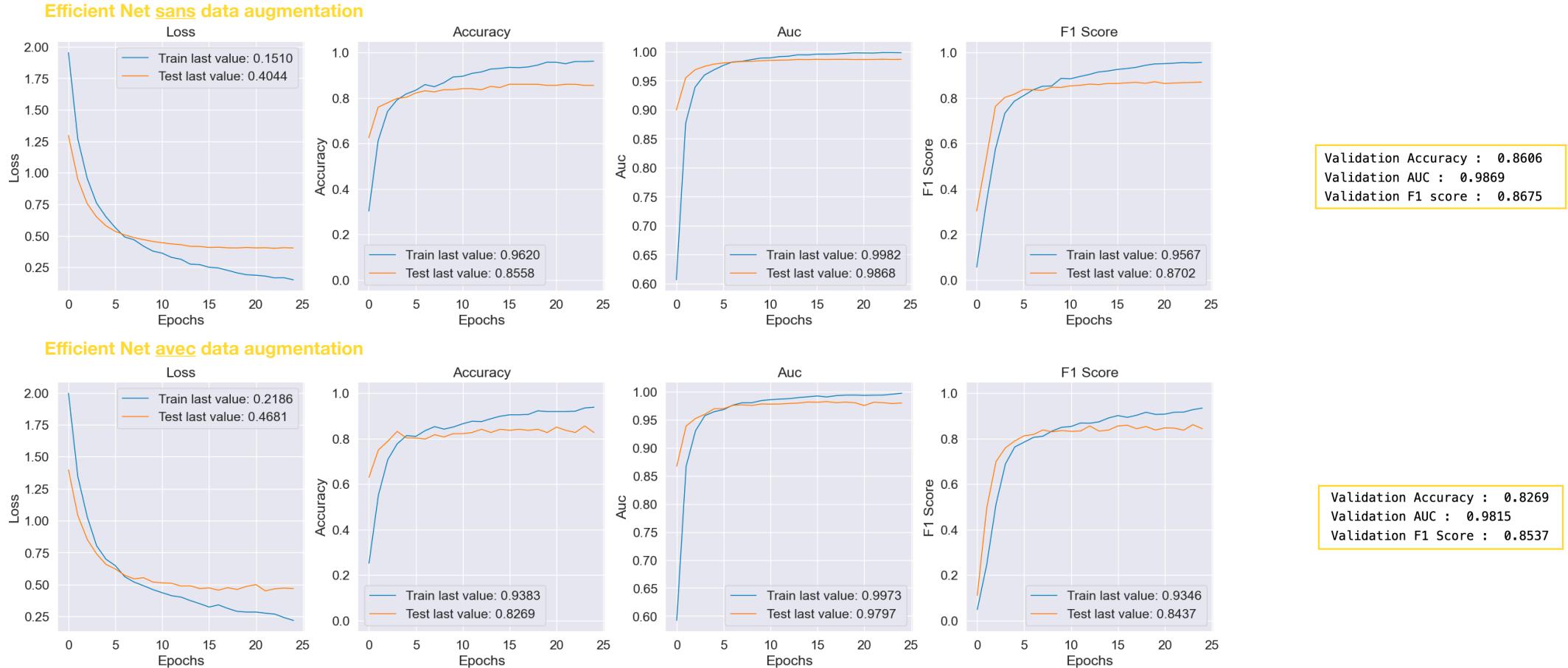
- Rotation, width shift, height shift, horizontal flip

Comparaison des résultats obtenus sur le validation set (25% du training set)

	Modèle	Accuracy	F1_score	Area Under the Curve	time
2	Efficient Net w/o data aug.	0.860577	0.867513	0.986945	366.0
0	VGG16 w/o data aug.	0.852381	0.866233	0.970183	1175.0
3	Efficient Net w data aug.	0.826923	0.853657	0.981465	462.0
5	VGG19 w data aug.	0.812500	0.819138	0.950617	2882.0
6	Resnet50 w/o data aug.	0.812500	0.827072	0.961400	489.0
7	Resnet50 w data aug.	0.793269	0.810812	0.954972	531.0
1	VGG16 w data aug.	0.783654	0.806596	0.942337	1530.0
4	VGG19 w/o data aug.	0.769231	0.788038	0.953173	1481.0
8	CNN from scratch w/o data aug.	0.566667	0.523578	0.839467	50.0
9	CNN from scratch w data aug.	0.423810	0.146477	0.777814	62.0

# Classification supervisée

## Meilleur résultat



# Classification supervisée

## Optimisation

Hyper-paramètres optimisés via Keras Tuner :

- Optimizer : `optimizers = ["SGD", "Adam", "RMSprop"]`
- Learning rate : `learning_rate = list(np.logspace(-4,-1,5))`
- Loss : `loss = ["categorical_crossentropy", "kullback_leibler_divergence"]`

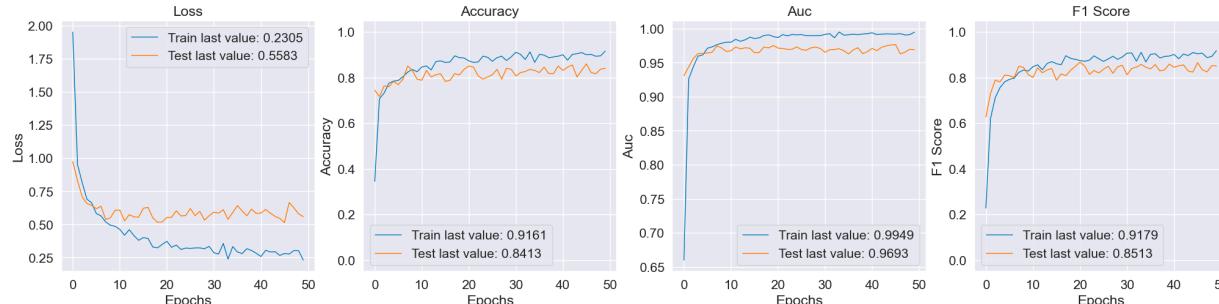
Hyper-paramètre optimisé « manuellement » :

- batch\_size : `batch_size = [16, 32, 64, 128]`

L'epoch a été fixée à 50

### Meilleurs paramètres

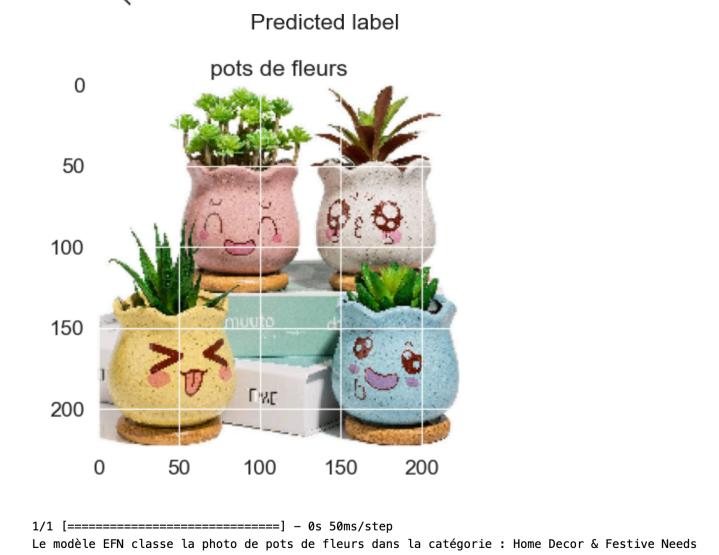
- Loss : `kullback_leibler_divergence`
- Optimizer : `RMSprop`
- Learning rate : `0.00316`
- Batch size : `32`



Accuracy sur le test set : 90 %

		Confusion matrix du modèle Efficient Net finetuné							Precision	Recall
									-35	72%
									-30	94%
		Baby Care	21	0	0	2	3	0	1	81%
		Beauty and Personal Care	1	17	0	2	1	0	0	100%
		Computers	0	0	37	0	0	0	1	97%
		Home Decor & Festive Needs	2	0	0	27	0	0	1	84%
		Home Furnishing	5	0	0	1	29	0	0	83%
		Kitchen & Dining	0	1	0	0	0	25	0	100%
		Watches	0	0	0	0	0	0	33	96%
									-5	100%
									0	100%
									-35	72%
									-30	94%
									-25	100%
									-20	84%
									-15	88%
									-10	100%
									-5	92%
									0	100%

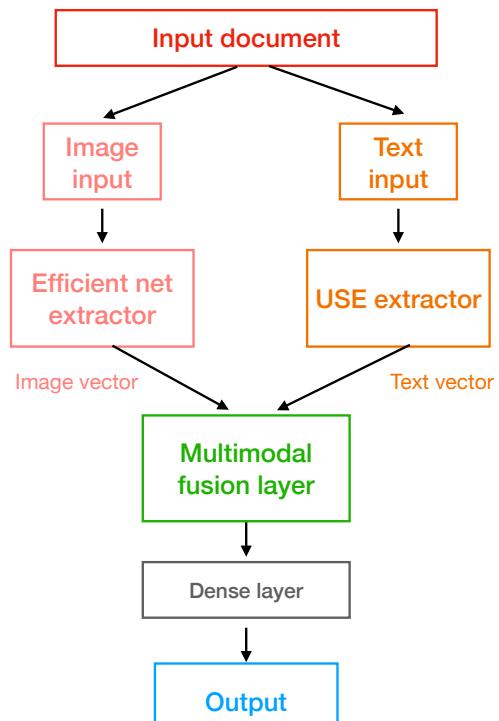
19 erreurs de classification



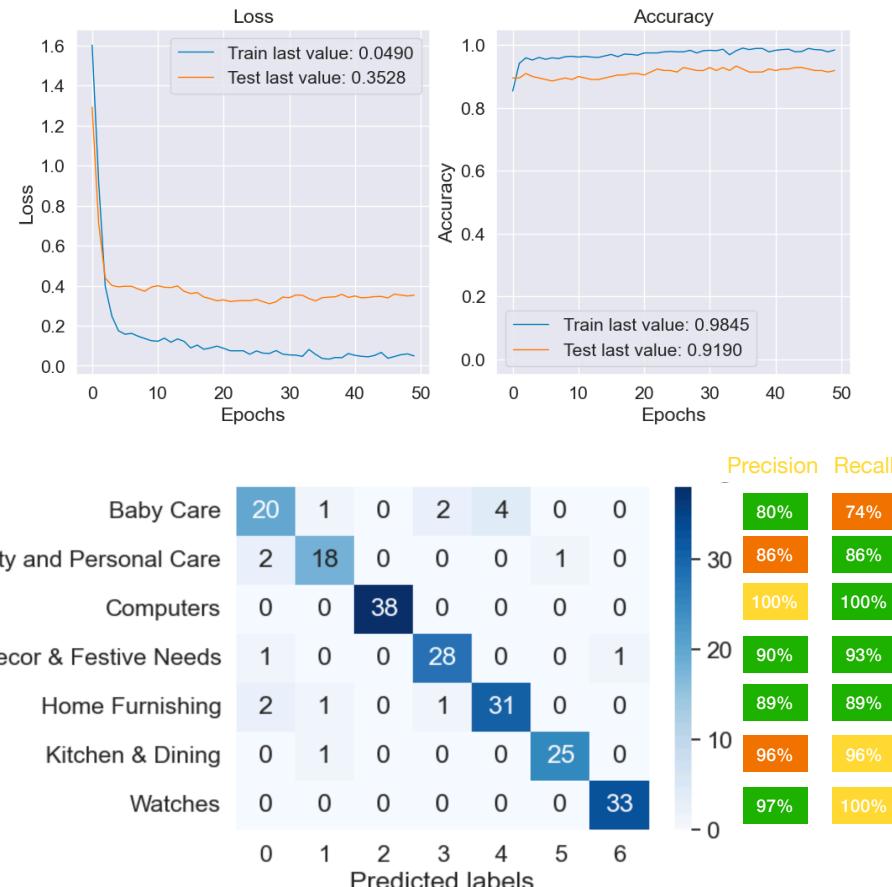
# Classification supervisée

## Modèle multimodal

Création d'un modèle multimodal qui s'appuie sur **le texte et l'image** pour catégoriser un article



Accuracy sur le test set : 92 %



**Element à considérer pour améliorer les performances du modèle:**

- Affiner le preprocessing des images et du texte
- Optimiser les opérations de data augmentation
- Finetuner les hyperparamètres à l'aide d'un autre tuner (Hyperband ou BayesianOptimization)
- Augmenter le nombre de données d'entraînement

# Test API

# Test de l'API

## Objectif

- « Place de marché » envisage d’élargir sa gamme de produit dans l’épicerie fine et souhaite tester la collecte de **produits à base de champagne** via une API.

## Présentation de l'API

- API (*Application Programming Interface*) : programme permettant à deux applications distinctes de communiquer entre elles et d'échanger des données.
- Nous avons utilisé une des API mises à disposition par **RapidAPI\***.
  - RapidAPI est un « API hub », une plateforme où il est possible de construire et consulter des API.
  - La plateforme recense ainsi des milliers d'API sur différents sujets (sport, finance, travel, food...)
- Edamam Food and Grocery Database** répertorie plus de 900,000 produits alimentaires de base, de restauration et de consommation courante emballés

Code Python pour récupérer les données via Rapidapi

```
url = "https://edamam-food-and-grocery-database.p.rapidapi.com/api/food-database/v2/parser"
querystring = {"ingr": "\\"champagne\\""}
headers = {
    "X-RapidAPI-Key": "d9e5a99e57msh2c0ac6e76aac869p12cd5ajsn50535546af72",
    "X-RapidAPI-Host": "edamam-food-and-grocery-database.p.rapidapi.com"
}
response = requests.get(url, headers=headers, params=querystring)
```

Nous avons fait l'exercice d'extraire 200 entrées : seuls 37% avaient une image

Extraction des 10 premiers aliments à base de champagne

	foodId	label	category	foodContentsLabel	image
0	food_a656mk2a5dmqb2adiamu6beihduu	Champagne	Generic foods	NaN	https://www.edamam.com/food-img/a71/a718cf3c52...
1	food_b753ithamdb8psbt0w2k9aquo06c	Champagne Vinaigrette, Champagne	Packaged foods	OLIVE OIL; BALSAMIC VINEGAR; CHAMPAGNE VINEGAR...	NaN
2	food_b3dyababjo54xbom6r8jzbghjqqe	Champagne Vinaigrette, Champagne	Packaged foods	INGREDIENTS: WATER; CANOLA OIL; CHAMPAGNE VINE...	https://www.edamam.com/food-img/d88/b64d973...
3	food_a9e0ghsamvoc45bw42ybsa3gken9	Champagne Vinaigrette, Champagne	Packaged foods	CANOLA AND SOYBEAN OIL; WHITE WINE (CONTAINS S...	NaN
4	food_an4jjucaucpus2a3utni8auhe7q9	Champagne Vinaigrette, Champagne	Packaged foods	WATER; CANOLA AND SOYBEAN OIL; WHITE WINE (CON...	NaN
5	food_apl44taoyv11r0lic1qa8xculi	Champagne Buttercream	Generic meals	sugar; butter; shortening; vanilla; champagne;...	NaN
6	food_byap67hab6evc3a0f9w1oag3s0qf	Champagne Sorbet	Generic meals	Sugar; Lemon juice; brandy; Champagne; Peach	NaN
7	food_am5egzbaq3fjlafl8xpdkdbc2asis	Champagne Truffles	Generic meals	butter; cocoa; sweetened condensed milk; vanil...	NaN
8	food_bc28rhiajk1fuva0vkfmeakbouc0	Champagne Vinaigrette	Generic meals	champagne vinegar; olive oil; Dijon mustard; s...	NaN
9	food_a79xmnya6toreaeukbroa0thhh0	Champagne Chicken	Generic meals	Flour; Salt; Pepper; Boneless, Skinless Chicke...	NaN

\* Autres API envisageable pour ce projet : apilayer (associé à Edamam), Nutritionix, Nutritifs API, FatSecret...

# Considération RGPD

- Une « donnée personnelle » est « toute information se rapportant à une **personne physique identifiée ou identifiable** ». Les données relatives aux aliments répertoriés dans la base Edamam Food and Grocery Database ne sont donc pas concernées.
- Rappel des 5 grands principes des règles de protection des données personnelles :
  - ❖ **Le principe de finalité** : le responsable d'un fichier ne peut enregistrer et utiliser des informations sur des personnes physiques que dans un but bien précis, légal et légitime ;
  - ❖ **Le principe de proportionnalité et de pertinence** : les informations enregistrées doivent être pertinentes et strictement nécessaires au regard de la finalité du fichier ;
  - ❖ **Le principe de durée de conservation limitée** : il n'est pas possible de conserver des informations sur des personnes physiques dans un fichier pour une durée indéfinie. Une durée de conservation précise doit être fixée, en fonction du type d'information enregistrée et de la finalité du fichier ;
  - ❖ **Le principe de sécurité et de confidentialité** : le responsable du fichier doit garantir la sécurité des informations qu'il détient. Il doit en particulier veiller à ce que seules les personnes autorisées aient accès à ces informations ;
  - ❖ **Les droits des personnes** : les personnes concernées sont en droit d'obtenir la communication des informations personnelles les concernant.

