



RAPPORT D'EXPLORATION ET PRE-PROCESSING DES DONNÉES

Classificateur texte et image Rakuten



Rudy MEVIZOU.

Sofia BOUIZZOUL.

Mehdi MALHAS.

Encadrement : DataScientest, Eliott DOUIEB

24 FÉVRIER 2024
PROJET DATA SCIENCE

Table des matières :

I	INTRODUCTION AU PROJET	3
1-	CONTEXTE GENERAL	3
II	OBJECTIFS DU PROJET.....	4
1-	CONTEXTE D'INSERTION DU PROJET DANS NOTRE METIER	4
2-	L'EQUIPE PROJET.....	4
3-	ENVIRONNEMENT ET ORGANISATION DU TRAVAIL EN EQUIPE	5
III	COMPREHENSION ET MANIPULATION DES DONNEES :.....	6
IV	ANALYSE EXPLORATOIRE.....	7
1-	ANALYSE EXPLORATOIRE DONNEES TEXTUELLES :	7
1.1	<i>Analyse globale</i>	8
1.2	<i>Analyse colonne par colonne</i>	9
1.3	<i>Preprocessing données textuelles</i>	18
2-	ANALYSE EXPLORATOIRE DONNEES IMAGES :.....	21
2.1	<i>Structure des données images</i> :.....	21
2.2	<i>Analyse globale</i>	22
2.3	<i>Preprocessing des données images</i> :	22
V	MODELISATION :.....	34
1-	APPROCHE METHODOLOGIQUE :	34
2-	MODELISATION POUR LA CLASSIFICATION DES DONNEES TEXTUELLES.....	35
2.1	<i>Approche Machine Learning Classique</i>	35
2.2	<i>Approche basée sur CamemBERT</i>	38
2.3	<i>Analyse résultats CamemBERT</i>	40
3-	MODELISATION POUR LA CLASSIFICATION DES DONNEES IMAGES :.....	41
3.1	<i>Vérification pre processing Images</i> :	41
3.2	<i>Modélisation image</i> :	44
3.3	<i>Choix du Modèle et Optimisation</i> :.....	45
3.4	<i>Comparaison des Modèles Testés</i> :	77
VI	MODELISATION MULTIMODALE POUR LA CLASSIFICATION DES PRODUITS RAKUTEN :	80
1-	WORKFLOW	80
2-	RESULTATS ET ANALYSE :	80
3-	PERFORMANCES DES MODELES INDIVIDUELS	80
4-	FUSION PAR MAX RULE :	81
5-	FUSION PAR VOTING CLASSIFIER (SOFT VOTING) :	82
6-	FUSION PAR STACKING CLASSIFIER :	83
7-	TENTATIVE DE FUSION VOTING + STACKING :	84
VII	BILAN :.....	85
1-	RESULTATS (BENCHMARK PROJET VS BENCHMARK RAKUTEN) :	85
2-	DIFFICULTES RENCONTREES ET EVOLUTIONS POSSIBLE :	85
2.1	<i>Limitations techniques</i>	86
2.2	<i>Modèles non explorés</i>	86
2.3	<i>Qualité hétérogène des données</i>	86
2.4	<i>Expérimentations incomplètes</i>	86
VIII	BIBLIOGRAPHIE :	87
IX	ANNEXES :	89

1- DIAGRAMME DE GANTT :	89
2- ROADMAP :	89
3- BUSINESS MODEL CANVAS RAKUTEN :.....	90

I Introduction au projet

1- Contexte général

Ce projet s'inscrit dans le cadre du challenge Rakuten France Multimodal Product Data Classification, une initiative organisée par l'ENS lors du Challenge Data. L'objectif principal est de développer un classificateur multimodal performant. Ce système automatisera l'attribution de codes de type précis aux produits, exploitant à la fois les données textuelles (désignations, descriptions) et visuelles (images).

L'automatisation de cette catégorisation est cruciale pour optimiser la recherche, la personnalisation et les recommandations sur les plateformes e-commerce, tout en réduisant significativement les coûts et les erreurs associés aux méthodes manuelles.

- **Le client : Rakuten**

Le défi est porté par Rakuten, une entreprise japonaise fondée en 1997 par Hiroshi Mikitani, pionnière du concept de place de marché en ligne. Aujourd'hui, Rakuten rassemble plus de 1,3 milliard de membres et a diversifié ses activités dans plusieurs secteurs. Parmi les éléments marquants de son identité, on peut retenir :

- Services financiers, contenus numériques et télécommunications, qui illustrent la polyvalence de ses activités.
- Filiales notables telles que Rakuten France, Rakuten Kobo, Rakuten Viber et Rakuten Viki, qui témoignent de sa capacité à innover sur différents marchés.

Ce contexte illustre l'ampleur et la complexité du défi auquel le projet se propose de répondre

- **Contexte du projet**

- **Technique :**

Le projet mobilise des techniques avancées d'analyse de données d'intelligence artificielle (Machine learning, deep learning).

La création d'un classificateur multimodal implique une préparation rigoureuse des données, incluant le nettoyage, la normalisation et l'harmonisation d'informations hétérogènes.

L'exploitation optimale des données requiert la combinaison de techniques de traitement du langage naturel (NLP) pour l'analyse textuelle (par exemple, modèles Transformer), et de méthodes de vision par ordinateur (par exemple, réseaux de neurones convolutifs tels que ResNet) pour l'interprétation des images. La gestion des défis liés aux données déséquilibrées et bruitées sera primordiale.

- **Économique :**

L'automatisation de la catégorisation des produits améliorera l'efficacité opérationnelle, réduisant les coûts liés aux erreurs de classification et optimisant l'allocation des ressources.

Le projet vise à fournir des insights stratégiques pour affiner la personnalisation des recommandations et améliorer l'expérience utilisateur, contribuant ainsi à la création de valeur et à la compétitivité de Rakuten.

- **Scientifique :**

Ce projet constitue une initiative innovante pour tester et améliorer les protocoles de classification multimodale. Il encourage l'exploration de nouvelles approches dans le traitement de données complexes, en intégrant des concepts de data science, machine learning et intelligence artificielle.

II Objectifs du projet

L'objectif principal est de développer un modèle prédictif robuste, d'évaluer ses performances en utilisant le F1-score (la métrique d'évaluation imposée), et de sélectionner l'approche optimale. Les livrables incluent le modèle final entraîné et validé, ainsi qu'un rapport détaillé documentant le processus de développement, les techniques employées et les résultats obtenus.

1- Contexte d'insertion du projet dans notre métier

Ce projet, réalisé dans le cadre de la formation chez DataScientest, représente une opportunité unique d'appliquer des techniques avancées de Machine Learning telles que la computer vision, le NLP, les modèles multimodaux et le Deep Learning. L'approche de classification automatique de produits à partir de données textuelles et visuelles trouve des applications directes dans divers secteurs, y compris les biotechnologies etc.

2- L'équipe projet

Rudy Mevizou :

Titulaire d'un doctorat en biologie et en reconversion vers la data science, je vise à acquérir une expérience pratique dans des projets de cette envergure. Bien que ce projet soit centré sur l'e-commerce, les techniques utilisées sont directement transférables au secteur biomédical, où l'analyse d'images (microscopie, imagerie médicale) et de textes (rapports cliniques, publications scientifiques) est essentielle. Des études, démontrent l'applications de l'IA dans l'imagerie médical et l'analyse de texte de rapport biologique. Par exemple, l'utilisations de l'IA pour la détections de cellules cancéreuses, par CNN, ou l'analyse de rapport biologique par du NLP. Ce projet représente donc une opportunité précieuse de développer des compétences applicables aux défis des biotechnologies.

Sofia :

Titulaire d'un Master en Management d'Unité de Production, je finalise un second Master en Audit et Contrôle de Gestion afin de renforcer mes compétences en finance, audit interne et analyse financière.

Avec 5 ans d'expérience en commerce international, j'ai acquis une expertise en transport aérien, réglementation douanière, normes internationales, ainsi qu'en gestion des achats, négociation commerciale, suivi des KPI et optimisation des coûts.

J'ai choisi le cursus Machine Learning Engineer incluant celui de data scientist pour appliquer ces compétences aux secteurs du commerce et de la finance, où l'automatisation et l'analyse des risques sont devenues essentielles. Mon objectif est d'exploiter la data science pour améliorer la gestion des risques clients et fournisseurs, optimiser les processus financiers et logistiques, et anticiper les tendances grâce à des modèles prédictifs, afin de favoriser une prise de décision plus efficace et stratégique.

Mehdi :

Titulaire d'un DEA en Propagation, Télécommunications et Télédétection, j'ai exercé depuis 28 an plusieurs métiers (technique, gestion de projet, chef d'équipe, marketing, dev...) dans plusieurs pays du monde essentiellement dans le secteur des télécommunications. Au fil des années j'ai développé des expertises en développement IT et à titre personnel je me suis investi dans l'intelligence artificielle depuis 2018 ou j'ai mené plusieurs projets. Aujourd'hui j'ai quitté Orange après 24 ans d'ancienneté pour m'investir exclusivement à l'IA. J'ai choisi le parcours de formation de DataScientest afin d'apporter une validation de mes compétences par une école reconnue, mais aussi pour découvrir d'autres aspects que je n'ai pas pu couvrir tout seul.

3- Environnement et organisation du travail en équipe

Dans le cadre d'un travail en équipe, il est essentiel de mettre en place des moyens efficaces pour faciliter les échanges et centraliser les résultats.

Pour cela, nous avons créé un espace partagé sur Google Drive, accessible à l'ensemble des membres de l'équipe. Cet espace nous a permis de stocker et de partager tous les éléments nécessaires au projet : images, données textuelles, documents de travail, etc.

Nous verrons par la suite que certaines étapes du projet ont nécessité une restructuration des données afin qu'elles puissent être exploitées dans les phases suivantes. Le fait de stocker ces informations dans le cloud nous a offert un accès rapide et permanent aux ressources, quel que soit le moment ou le lieu.

A chaque étape de développement, chaque membre de l'équipe avait un rôle bien défini en fonction de ses compétences et de ses disponibilités : traitement des données, modélisation, visualisation, etc.

Pour la communication et la coordination, nous avons utilisé plusieurs outils complémentaires comme Slack, WhatsApp, les e-mails ou encore les appels téléphoniques.

Des réunions régulières étaient organisées pour faire le point sur l'avancement du projet, résoudre les difficultés rencontrées, prendre des décisions collectives et partager les connaissances ainsi que les bonnes pratiques.

III Compréhension et manipulation des données :

Ce projet repose exclusivement sur l'ensemble de données fourni par Rakuten France, sans recours à des sources externes.

L'ensemble de données comprend environ 99 000 fiches produits au format CSV, réparties en un ensemble d'entraînement (84 916 fiches) et un ensemble de test (13 812 fiches). Chaque fiche contient :

- La désignation du produit (titre court).
- Une description détaillée du produit (avec des valeurs NaN possibles).
- L'image associée au produit.
- Le code type du produit (la cible à prédire).

Les données sont structurées en quatre ensembles distincts :

- X_train.csv : Données d'entrée pour l'entraînement.
- Y_train.csv : Codes types (données de sortie) pour l'entraînement.
- X_test.csv : Données d'entrée pour le test.
- Images.zip : dossier compressé, contenant toutes les images. Après décompression, il crée un dossier images avec deux sous-dossiers : image_training et image_test. Les colonnes *imageid* et *productid* permettent de localiser les images dans le dossier images. Le nom de chaque fichier image suit le format :

[image_imageid_product_productid.jpg](#)

Format des Fichiers CSV

- **X_train.csv**

- La première ligne de chaque fichier CSV contient l'en-tête des colonnes.
- Les colonnes sont séparées par des virgules (",").
- Les colonnes disponibles sont :
 - *designation* : Le titre du produit.
 - *description* : La description du produit (peut contenir des valeurs NaN).
 - *productid* : Un identifiant unique du produit.
 - *imageid* : Un identifiant unique de l'image.

- **Y_train.csv**

La colonne *prdtypecode* correspond à la catégorie des produits, qui constitue la variable à prédire dans notre tâche de classification. Elle est associée à chaque identifiant entier présent dans le fichier d'entrée d'entraînement (X_train.csv). Ici aussi, la première ligne du fichier correspond à l'en-tête, et les colonnes sont séparées par des virgules.

Il est important de noter que nous n'avons pas accès aux y_test associés aux X_test, ce qui constitue un obstacle pour l'évaluation des performances. Étant donné la taille relativement conséquente du jeu de données, nous avons choisi de générer notre ensemble de test à partir de X_train et y_train, afin de contourner cette limitation aussi bien pour les données textuelles que pour les images.

IV Analyse exploratoire

1- Analyse exploratoire données textuelles :

En premier lieu, un aperçu primaire des données fournis par le challenge est effectué : X_train_update.csv et Y_train_CVw08PX.csv.

X_train_update.csv :

	designation	description	productid	imageid
0	Olivia: Personalisiertes Notizbuch / 150 Seite...	NaN	3804725264	1263597046
1	Journal Des Arts (Le) N° 133 Du 28/09/2001 - L...	NaN	436067568	1008141237
2	Grand Stylet Ergonomique Bleu Gamepad Nintendo...	PILOT STYLE Touch Pen de marque Speedlink est ...	201115110	938777978
3	Peluche Donald - Europe - Disneyland 2000 (Mar...	NaN	50418756	457047496
4	La Guerre Des Tuques	Luc a des idées de grandeur. Il veut or...	278535884	1077757786

Y_train_CVw08PX.csv:

	prdtypecode
0	10
1	2280
2	50
3	1280
4	2705

Après contact avec un organisateur du projet, nous avons pu récupérer les étiquettes correspondant aux identifiants *prdtypecode*, ce qui nous permettra d'apporter une précision sur ces identifiants.

	prdtypecode	categorie
0	10	Livre occasion
1	2280	Revues et journaux
2	50	Accessoire Console
3	1280	Jouet enfant, déguisement
4	2705	Livre neuf

1.1 Analyse globale

Au premier abord, on s'aperçoit du nombre important de valeurs nulles dans la colonne *description*, suggérant le caractère facultatif de son remplissage. Les autres colonnes de notre jeu de données ne contiennent pas de valeurs manquantes. Le type des colonnes est cohérent avec le fait que les colonnes *description* et *désignation* (de type *object*) contiennent bien du texte, tandis que les autres colonnes, de type *int64*, contiennent les identifiants des produits, des images et des catégories.

Nous avons générer un tableau multi-index afin de vérifier par colonne le nombre et le pourcentage de:

- Valeurs manquantes
- Doublons
- Modalités uniques

	Colonne	Valeur	Valeur en %
Valeurs manquantes	Designation	0	0
	Description	29800	35,09350417
	Productid	0	0
	Imageid	0	0
	Prdtypecode	0	0
Doublons	Designation	2651	3,12190871
	Description	7610	13,80724291
	Productid	0	0
	Imageid	0	0
	Prdtypecode	84889	99,96820387
Unicité	Designation	82265	96,87809129
	Description	47506	55,94469829
	Productids	84916	100
	Imageids	84916	100
	Prdtypecode	27	0,031796128

Premières observations à la suite de cette analyse globale :

- Colonne désignation : 3 % de doublons (2651 doublons), un taux relativement faible mais pouvant tout de même poser des problèmes.
- Colonne description : 35 % de valeurs manquantes (soit 29 800 lignes) et 14 % de doublons (7 610 occurrences). Une stratégie de prétraitement sera donc nécessaire pour traiter ces doublons ainsi que les valeurs manquantes.
- *ProductID* et *ImageID* : Chaque produit possède un *productid* et un *imageid* unique.
- *Prdtypecode* : Près de 100 % de doublons, ce qui est logique étant donné qu'il existe seulement 27 catégories pour 84 916 produits.

Il est également intéressant de noter qu'aucune ligne n'apparaît en double, malgré le nombre important de doublons constaté dans la colonne *description*. Cela reste cohérent puisque chaque ligne correspond à un produit unique, identifié par un *productid* et un *imageid* distincts.

1.2 Analyse colonne par colonne

Nous avons procédé à l'analyse des variables une par une. Cette étape nous permettra d'identifier les meilleures stratégies pour traiter les doublons et les valeurs manquantes, tout en guidant notre choix des modèles à tester.

a) Analyse variable cible : *prdtypecode*

	Nombre total d'observations	Nombre de classes uniques	Classe la plus fréquente	Classe la moins fréquente	Ratio déséquilibre	Écart-type des fréquences	Test KS (stat, p-value)	Entropie des classes	Classes rares
0	84916	27	2583	1180	13.362565	2103.248906	(1.0, 0.0)	4.463017	66.666667

Le jeu de données contient 84 916 observations uniques, constituant une base solide pour l'analyse et la modélisation. On y dénombre 27 classes uniques, correspondant aux différentes catégories de produits définies par le *prdtypecode*.

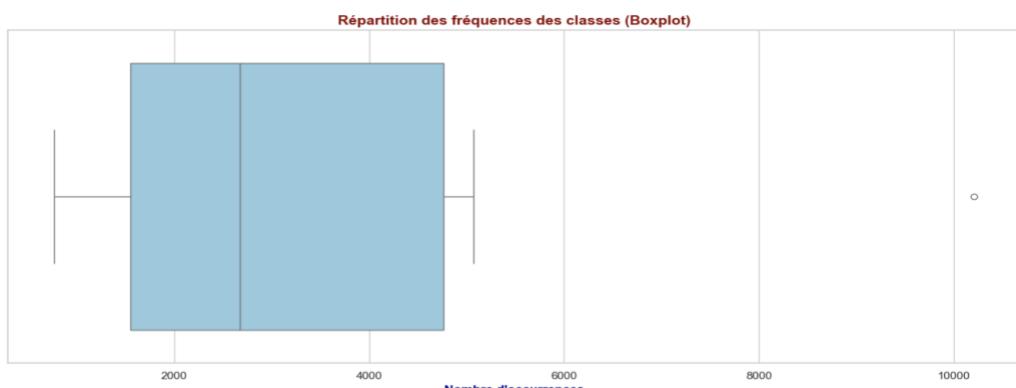
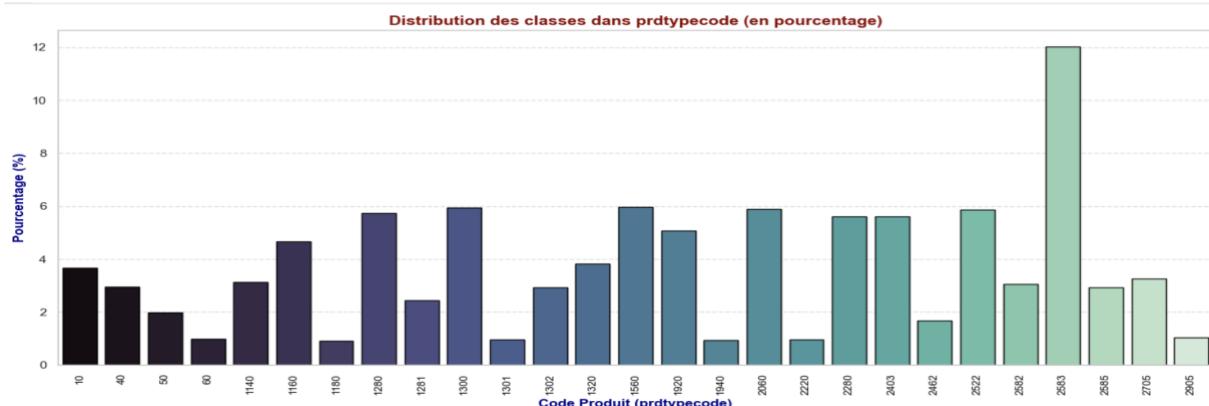
Une forte disparité se dégage dans la distribution des classes, avec 2 583 observations pour la classe la plus fréquente et seulement 1 180 observations pour la moins fréquente, soit un ratio de déséquilibre de 13,36. Cette hétérogénéité pourrait impacter la performance des modèles prédictifs.

L'écart-type des fréquences, évalué à 2 103,25, confirme la variabilité importante des tailles de classes. Le test de Kolmogorov-Smirnov (KS) indique un statistique de 1,0 avec une p-value de 0,0, soulignant un écart significatif par rapport à une distribution uniforme.

L'entropie des classes, estimée à 4,46, reflète une diversité relative, bien que la répartition reste déséquilibrée. Il convient de prêter une attention particulière à cette inégalité pour éviter que les modèles ne favorisent les classes majoritaires.

Enfin, 66,67 % des classes sont considérées comme rares, traduisant une représentation limitée dans le jeu de données.

Pour illustrer ces propos, nous avons réalisé un histogramme des fréquences par catégorie de produit (*prdtypecode*) ainsi qu'un boxplot des fréquences afin de détecter les classes sous-représentées et les éventuels outliers.



Ces premières observations sur la colonne *prdtypecode* révèlent un déséquilibre marqué des classes, susceptible d'impacter la performance des modèles. Les classes rares risquent d'être mal classifiées, le modèle disposant de moins d'exemples pour les apprendre efficacement. Il est donc essentiel de mettre en place une stratégie adaptée pour pallier ce problème, parmi lesquelles :

- Sous-échantillonnage (*undersampling*) des classes majoritaires,
- Sur-échantillonnage (*oversampling*) des classes minoritaires,

En complément, l'utilisation de techniques de pondération ou de modèles robustes au déséquilibre peut également s'avérer pertinente.

Il sera crucial d'évaluer ces différentes approches afin de sélectionner celle offrant les meilleures performances pour la classification finale.

b) Analyse variable explicatives : colonnes *designation* et *description*

Dans notre analyse globale, deux points essentiels se dégagent concernant ces deux colonnes :

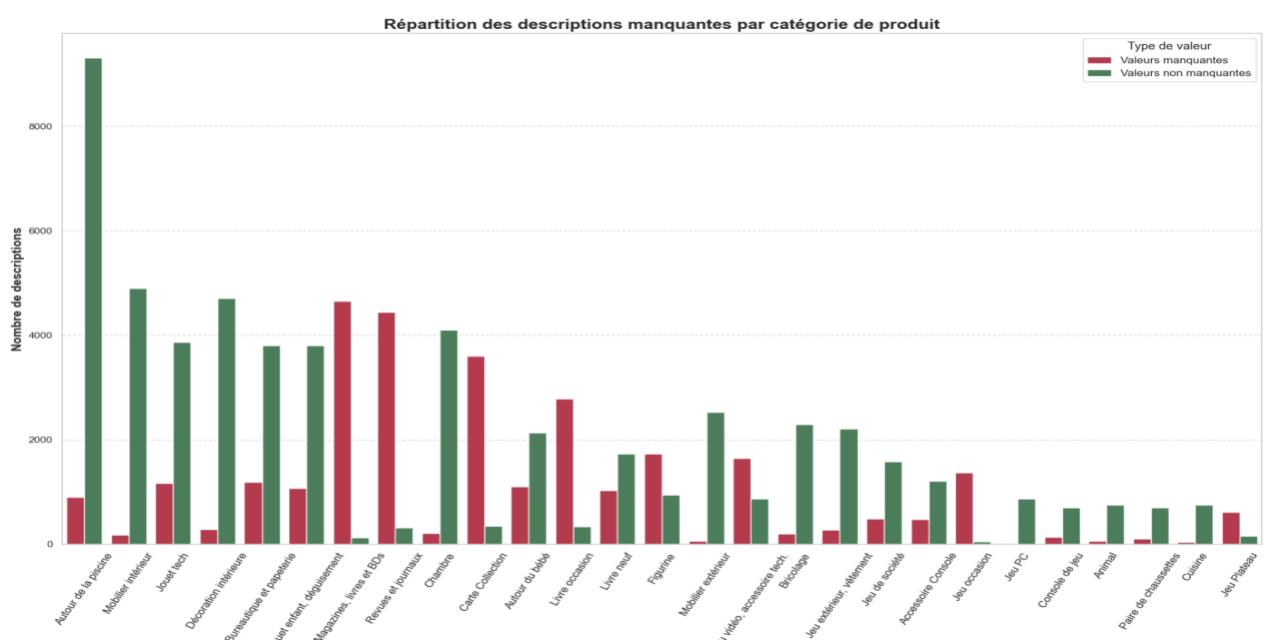
- Valeurs manquantes dans la colonne *description* :
 - Un nombre significatif de 29 800 valeurs manquantes, représentant 35 % des données.
- Doublons :
 - 7 610 doublons dans la colonne *description*, soit 13,8 % des observations.
 - 2 651 doublons dans la colonne *désignation*, correspondant à 3,1 % des données.

Il est indispensable de traiter ces problèmes afin d'optimiser la performance de nos modèles prédictifs.

b-1) Valeurs manquantes

Étant donné le nombre important de produits (84 916), il est légitime de se demander si la suppression des produits dépourvus de description affecterait réellement les performances de notre modèle de prédiction.

Nous nous sommes alors interrogés sur l'impact potentiel de cette suppression sur la répartition des produits par catégorie. L'hypothèse formulée est la suivante : étant donné que certaines classes contiennent peu de produits et que le pourcentage de valeurs manquantes est non négligeable, retirer ces observations pourrait considérablement réduire la quantité de produits disponibles dans ces catégories.



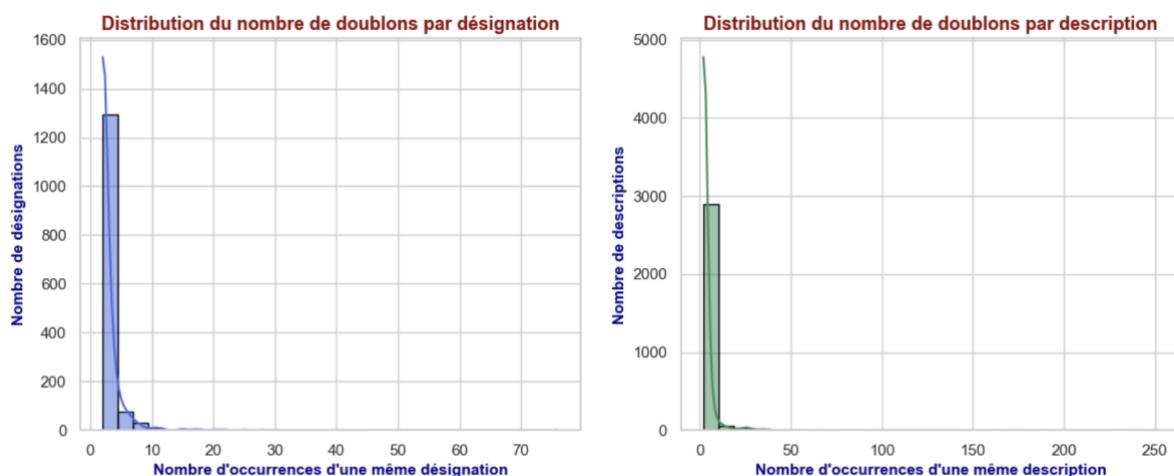
De manière intéressante, la suppression des produits dépourvus de description entraînerait une perte d'information susceptible de nuire à la capacité du modèle à généraliser correctement. Cette suppression n'affecterait pas uniformément l'ensemble des catégories, mais impactera de manière disproportionnée certaines d'entre elles, telles que les jeux de plateau, les jeux d'occasion, les cartes de collection, et autres catégories déjà sous-représentées. Cela accentuerait davantage le déséquilibre des classes, augmentant ainsi le risque de biais du modèle en faveur des catégories dominantes.

Pour atténuer cet effet, nous avons envisagé de procéder à la concaténation des colonnes *désignation* et *description* afin de créer une seule variable textuelle plus riche. En l'absence de description, la *désignation* peut partiellement compenser la perte d'information.

Cette approche permet de préserver les produits sans description tout en enrichissant les données utilisées par le modèle. Elle présente l'avantage de limiter la perte d'information tout en améliorant la qualité globale des données d'entrée, ce qui pourrait se traduire par une meilleure capacité de généralisation du modèle.

b-2) Doublons

Un autre problème rencontré lors de notre analyse globale concerne le nombre important de doublons. Pour mieux comprendre cette situation, examinons la distribution du nombre de doublons, d'une part par *désignation* et d'autre part par *description*.



L'analyse des doublons met en évidence une distribution asymétrique à droite pour les deux variables, *désignation* et *description* :

- *désignation* :
 - La majorité des désignations apparaissent 2 à 3 fois, avec un pic marqué.
 - Quelques désignations sont répétées jusqu'à 70 fois, bien que cela reste marginal.
 - Ces doublons peuvent refléter des produits similaires commercialisés sous une même appellation standard.

- *description* :
 - La distribution est similaire, mais avec une dispersion plus large : la plupart des descriptions apparaissent 2 à 5 fois, tandis que certaines sont répétées jusqu'à 250 fois.
 - Cela suggère l'usage de descriptions génériques pour des produits standardisés ou similaires.

Comme pour le problème des valeurs manquantes, il serait légitime de se demander si la suppression des doublons pourrait améliorer la qualité de notre future prédition. Toutefois, une réflexion plus approfondie s'impose. En effet, bien que certaines désignations soient dupliquées, la description associée peut fournir des informations complémentaires permettant de mieux distinguer les produits lors de la classification. L'inverse est également vrai : des descriptions identiques peuvent correspondre à des produits distincts, différenciés par leur désignation.

C'est précisément cet aspect que nous avons cherché à évaluer à travers les tableaux suivants.

Tableau des doublons de description avec diversité des désignations :			
	description	count	unique_descriptions
312	 Attention !!! Ce produit est un import si...	252	251
2701	Taille: En format A5 (144 cm x 21 cm) Caractéi...	232	231
2699	Taille: En format A5 (144 cm x 21 cm) Caracter...	189	170
2308	Piscine hors-sol ovale en acier blanc Wet de l...	37	36
2307	Piscine hors-sol ovale en acier blanc Fidji de...	36	35
...
1227	Décoration du foyer Coussin Style de Throw Tai...	2	1
1201	Doux nouveau tissu Intelligence bébé Développe...	2	1
1204	Drapeaux de jardin saisonniers Double Sided Va...	2	1
1206	Draps en coton Taie Canapé taille Throw Coussi...	2	1
1224	Décoration du foyer Coussin Impression vie Thr...	2	1

Tableau des doublons de désignation avec diversité des descriptions :			
	designation	count	unique_descriptions
137	5d Broderie Peintures Strass Diamant Bricolage...	76	53
1237	Simple Linen Creative Belle Oreiller Taie D'or...	28	12
400	Cotton Linen Place Décoratifs Pour La Maison C...	25	16
702	Imprimer Taie Polyester Canapé Coussin Car Cov...	22	10
729	Joyeuse Saint-Valentin Jetez Taie D'oreiller S...	21	11
...
565	Ensemble De Salle À Manger D'extérieur 7 Pcs P...	2	1
571	Ensemble de piscine Ultra XTR Frame Rectangula...	2	1
574	Espace Romantique Valentine Day Tapis Mat Sall...	2	1
575	Etui De Protection Absorption Des Chocs Envelo...	2	1
579	Extracteur Cheville Sit Up Mat Pull Corde Péda...	2	1

Les deux tableaux révèlent que la répétition des désignations ou des descriptions n'est pas systématiquement synonyme de redondance totale. En effet, un même libellé peut être associé à des descriptions différentes, tout comme une description générique peut accompagner des désignations distinctes.

Néanmoins un problème critique peut survenir si des produits partagent à la fois la même désignation et la même description, tout en étant associés à des catégories différentes. Face à ces signaux contradictoires, le modèle risque de rencontrer des difficultés pour identifier correctement la catégorie, ce qui se traduit par une augmentation des erreurs de classification. Il aura tendance à sur-prédire la catégorie majoritaire, tout en sous-prédisant les catégories minoritaires, introduisant ainsi un biais significatif dans les prédictions.

Pour mieux comprendre l'ampleur de ce phénomène, la première étape consiste à vérifier si les doublons textuels exacts (*désignation/description*) sont systématiquement attribués à une seule catégorie ou s'ils se répartissent entre plusieurs catégories. Cette analyse permettra d'évaluer l'impact potentiel de ces incohérences sur la performance du modèle.



	désignation	description	prdtypecode	Noms des catégories	nombre de catégories différentes
0	24cm Puppy Cartoon Peluche Cute Dress Up Puppy...	24cm Puppy Cartoon peluche Cute Dress Up Puppy...	[1280, 1281]	Jeu de société & Jouet enfant, déguisement	2
1	3pcs Non-Slip De Bain Tapis De Bain Tapis Cuis...	3pcs non-Slip poisson Échelle Tapis de bain Ta...	[2060, 2060, 2060, 1920]	Chambre & Décoration intérieure	2
2	3pcs Non-Slip Poisson Échelle Tapis De Bain Ta...	3pcs non-Slip poisson Échelle Tapis de bain Ta...	[2060, 1920, 1920]	Chambre & Décoration intérieure	2
3	4 Pièces New Cartoon Aimant Pour Réfrigérateur...	4 pièces New Cartoon Aimant Pour Réfrigérateur...	[2060, 1560]	Décoration intérieure & Mobilier intérieur	2
4	Animaux En Peluche Mignon Petit Drôle Jouets D...	Petits animaux mignons drôle de peluche Peluch...	[1281, 1280]	Jeu de société & Jouet enfant, déguisement	2

Certaines duplications se répartissent entre plusieurs catégories. Une analyse approfondie révèle que ces catégories sont souvent étroitement liées, comme "*Jeu de société*" et "*Jouet enfant, déguisement*", ou encore "*Chambre et Décoration intérieure*". Cette proximité thématique explique pourquoi des produits peuvent partager des désignations et descriptions similaires.

Parmi l'ensemble des doublons exacts identifiés, 29 cas ont été jugés véritablement problématiques. Ils concernent des produits associés à un maximum de deux catégories différentes, généralement proches en termes d'usage ou de classification.

Cette situation génère une ambiguïté de prédiction, poussant le modèle à privilégier la catégorie la plus fréquente, même lorsqu'elle n'est pas la plus pertinente. Cela compromet la capacité de généralisation du modèle et augmente le risque de faux positifs pour les catégories moins représentées.

Pour conclure sur la gestion des doublons des colonnes *désignation* et *description* :

l'analyse a révélé que ces deux colonnes présentent de nombreux doublons lorsqu'elles sont considérées séparément. Cependant, leur combinaison permet de mieux différencier les produits, la *description* apportant des précisions supplémentaires à la *désignation*.

Problèmes identifiés :

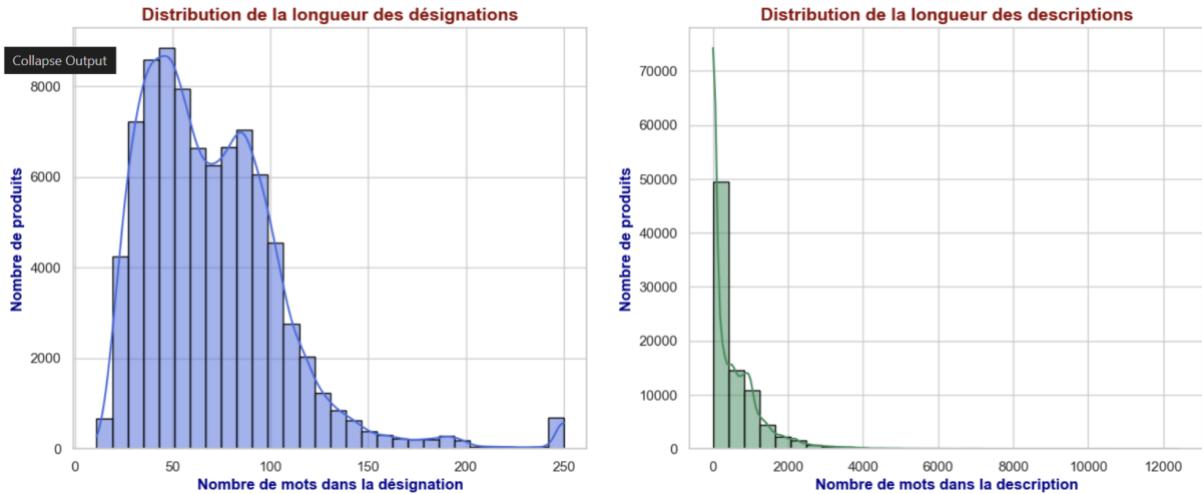
- Doublons intra-catégorie : Certains produits partagent une *désignation* et une *description* identiques tout en appartenant à la même catégorie, risquant de provoquer un surapprentissage (overfitting).
- Doublons inter-catégories : Des doublons exacts peuvent être associés à des catégories distinctes, entraînant des erreurs de classification et un biais en faveur des catégories majoritaires.

Solutions proposées :

- Concaténation des colonnes *désignation* et *description* pour enrichir les données et limiter les doublons exacts. Cela est pratique car nos valeurs manquantes seront éliminées par la même occasion
- Ajout d'attributs distinctifs, tels que *product_id* et *image_id*, pour renforcer la distinction entre produits similaires.
- Équilibrage des classes pour éviter que la présence de doublons n'accentue les biais, en utilisant des techniques telles que le sur-échantillonnage, le sous-échantillonnage ou la pondération des classes.

Ces approches permettront de préserver l'information tout en améliorant la robustesse et la capacité de généralisation du modèle.

b-3) Longueur des colonnes



L'analyse des longueurs de texte met en évidence des différences notables entre les colonnes *désignation* (nom du produit) et *description*, avec la présence de nombreuses valeurs extrêmes nécessitant un nettoyage rigoureux et une standardisation pour garantir la qualité des données.

- Longueur des désignations

La majorité des désignations présentent une longueur comprise entre 30 et 100 caractères, ce qui est cohérent avec la nature attendue de cette colonne, censée contenir des noms de produits relativement succincts.

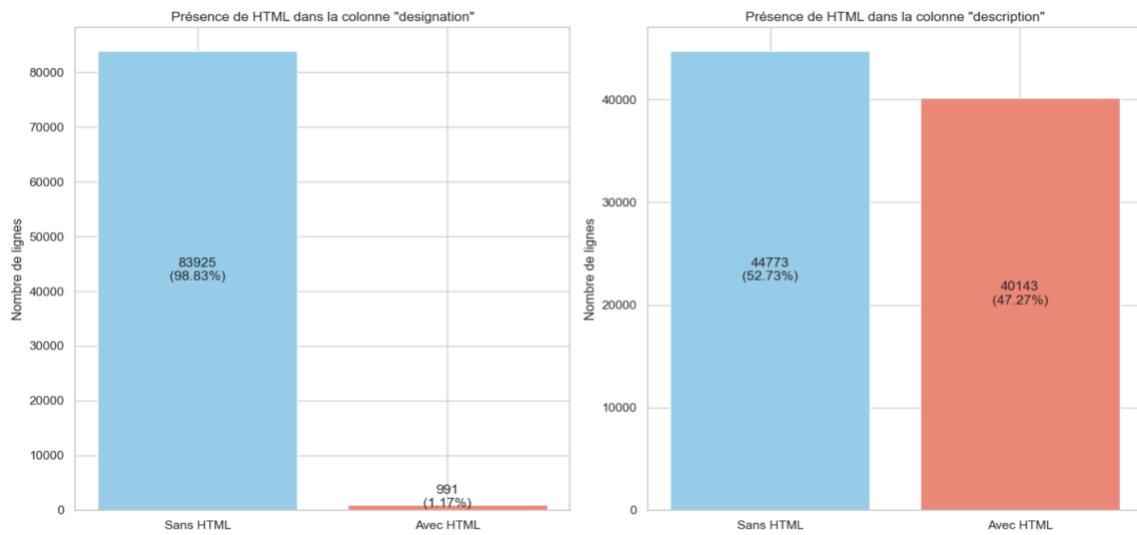
Cependant, des valeurs atypiques apparaissent au-delà de 200 caractères, avec un pic notable autour de 250 caractères, révélant des désignations exceptionnellement longues. Ces anomalies peuvent résulter d'erreurs de saisie ou d'informations détaillées inappropriées pour un simple libellé de produit.

- Longueur des descriptions

Les descriptions, quant à elles, présentent une variabilité beaucoup plus importante que les désignations :

- La majorité des descriptions se situent en dessous de 1 000 caractères, avec un pic marqué sous les 500 caractères.
- La distribution est fortement étirée vers les valeurs élevées, avec des descriptions atteignant jusqu'à 12 000 caractères.
- Des outliers significatifs sont visibles au-delà de 2 000 caractères, certaines dépassant même les 5 000 à 10 000 caractères.

L'analyse montre que la longueur excessive des textes dans la colonne *description* est principalement due à la présence de balises/entités HTML, présentes dans près de la moitié des lignes. Un nettoyage est indispensable pour éviter des analyses biaisées et faciliter la classification. La colonne *désignation* contient également des balises HTML, mais de manière marginale (1,17 %), suggérant un impact limité du nettoyage, bien qu'il puisse améliorer la clarté des données concernées.



Conclusion et stratégie de nettoyage :

Dans les analyses précédentes, nous avons opté pour une stratégie de concaténation des colonnes *désignation* et *description* afin d'enrichir l'information textuelle. Toutefois, l'étude des longueurs souligne la nécessité d'un nettoyage approfondi pour garantir la pertinence et la qualité de cette concaténation.

Pour atteindre cet objectif, les étapes suivantes seront mises en œuvre :

- Suppression des balises HTML (notamment dans les descriptions).
- Tokenisation des textes pour faciliter leur traitement.
- Élimination des stopwords (mots inutiles), des caractères numériques, des unités, des abréviations.
- Conversion en minuscules pour uniformiser l'ensemble des données.
- Suppression des caractères spéciaux pouvant fausser l'analyse.
- Réduction à la racine des mots, par lemmatisation, pour simplifier les variations lexicales.

En appliquant ces mesures, nous visons à standardiser les données textuelles, à renforcer la cohérence entre les colonnes *désignation* et *description*, et à améliorer la qualité globale des entrées destinées à la modélisation.

b-4) Analyse de la fréquence des mots

Examinons à présent la fréquence des mots présents dans nos colonnes textuelles. Cette analyse a été réalisée sur les colonnes *désignation*, *description*, ainsi que sur la colonne *texte*, issue de la concaténation des deux premières.

Il est important de souligner que, cette étape ayant été effectuée avant tout prétraitement, nous nous attendons à observer une quantité significative de **bruit linguistique**, incluant des mots non pertinents, des caractères spéciaux ou des variations lexicales.



L'analyse des Word Clouds révèle un bruit important, principalement dû à la présence de stopwords (ex. : *de, la, pour*) et de balises HTML (*br, li, ul*), masquant les termes réellement pertinents.

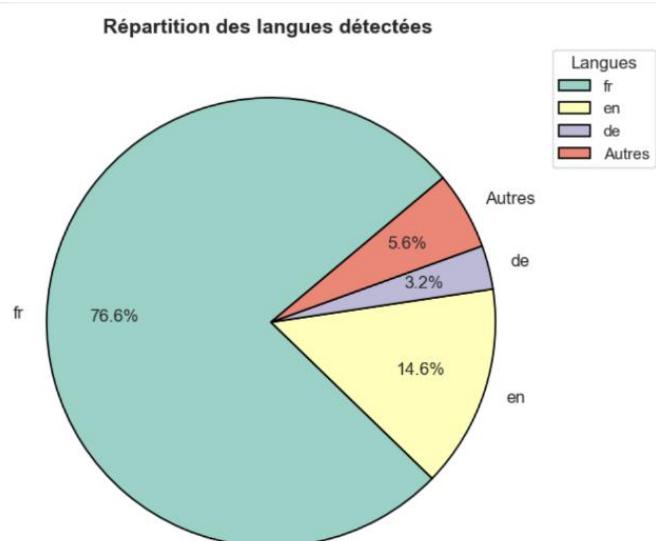
1. Désignations : Les mots *home*, *decor*, *piscine*, *cm*, *kit*, *noir*, *taie* et *d'oreiller* dominent, reflétant une forte présence de produits liés à la décoration intérieure et aux accessoires pour piscines.
 2. Descriptions : Les termes *haute qualité*, *eau* et *cm* sont prédominants, indiquant des descriptions techniques pour des articles liés à la piscine, à la filtration et au bricolage.
 3. Concaténation (désignation + description) : Le Word Cloud résultant est similaire à celui des descriptions seules, en raison de la prépondérance de la colonne description, amplifiée par les balises HTML.

Comme évoqué précédemment, un nettoyage avancé est essentiel pour améliorer la qualité des données textuelles et la performance des modèles de classification.

b-5) Analyse linguistique

Lors de notre première analyse du DataFrame, renforcée par l'examen de la fréquence des mots à travers un Word Cloud, nous avons constaté la présence de plusieurs langues, avec une prédominance marquée du français.

Afin d'optimiser l'efficacité du traitement — après avoir constaté que l'analyse individuelle par colonne était particulièrement chronophage — et en accord avec notre stratégie de concaténation des données textuelles, nous avons choisi de détecter les langues directement sur ce texte concaténé.



L'analyse des données textuelles a révélé la présence de 31 langues distinctes. Toutefois, la majorité des textes sont rédigés en français, représentant 77 % du corpus. Cela confirme que les descriptions sont principalement produites en français.

L'anglais constitue la deuxième langue la plus représentée, avec 14,6 % des textes. Cette proportion suggère la présence de produits internationaux, probablement importés ou vendus via des plateformes multilingues.

L'allemand, bien que minoritaire, représente 3,2 % des données. Cela reflète des descriptions non traduites, probablement issues de fournisseurs européens. À elles seules, ces trois langues couvrent 94,4 % du corpus.

Le reste des textes se répartit entre différentes langues minoritaires, comme le néerlandais, l'italien, ou des langues mal identifiées regroupées sous la catégorie "Autres". Cette catégorie peut inclure des données bruitées, telles que des descriptions partiellement traduites ou des combinaisons linguistiques.

Les modèles de classification sont généralement optimisés pour le français, mais leur performance varie selon la stratégie adoptée. Étant donné que 23 % des données sont rédigées dans d'autres langues, il est essentiel de déterminer si la traduction en français améliore les performances.

Pour cela, nous avons identifié deux approches à comparer :

- Scénario 1: Multilinguisme (langue d'origine conservée)

Les données sont traitées dans leur langue d'origine (français, anglais, allemand, etc.). Cette approche permettra de tester des modèles compatibles avec le multilinguisme : **mBERT, TF-IDF + SVM**

- Scénario 2 : Traduction en français

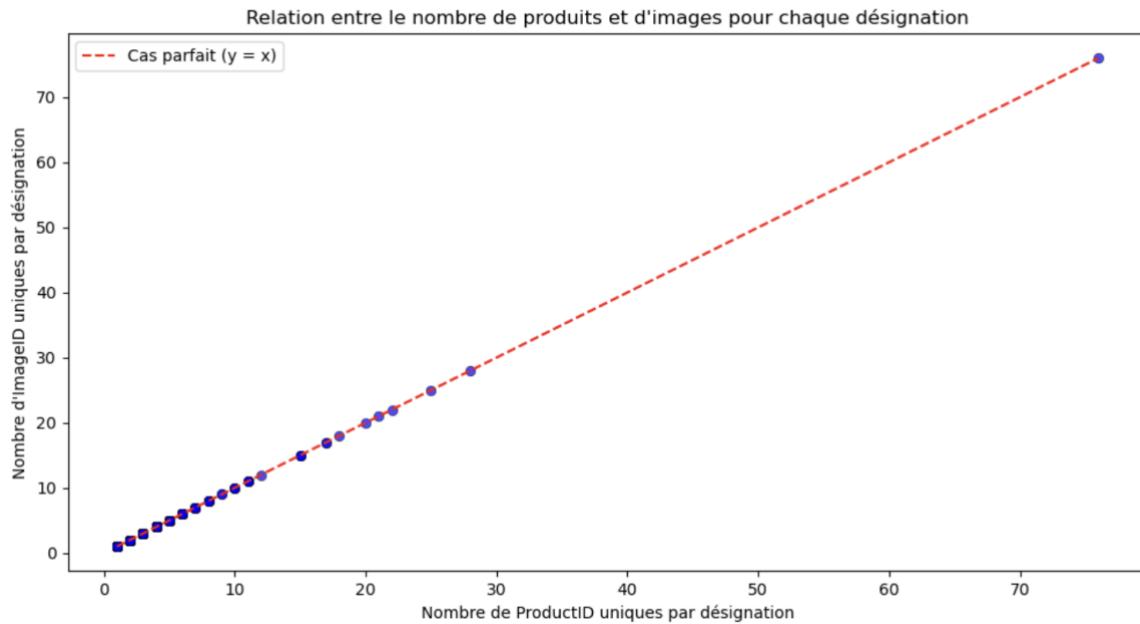
Les 23 % de données non francophones sont traduites en français, garantissant un corpus entièrement homogène. Cette approche est particulièrement adaptée aux modèles conçus pour le français :

CamemBERT, TF-IDF + Random Forest

c) Analyse variable explicatives : colonnes *productid* et *imageid*

Pour ces deux colonnes, il nous semblait essentiel de vérifier qu'une *imageid* soit bien associée à un *productid* unique. Le graphique suivant confirme cette hypothèse : chaque produit possède effectivement un identifiant unique pour le *productid* et l'*imageid*.

Ainsi, chaque ligne du tableau représente un produit distinct, garantissant l'intégrité et la cohérence de la structure des données.



Ces deux variables (*productid* et *imageid*) n'ont, a priori, aucun intérêt pour notre modèle de prédiction et seront supprimées.

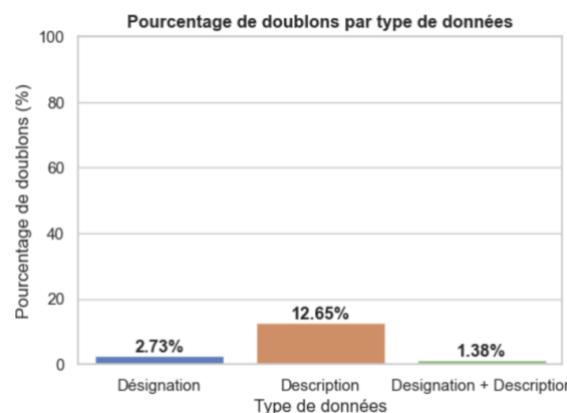
1.3 Preprocessing données textuelles

Pour débuter, nous avons divisé les données en trois ensembles distincts : 80 % pour l'entraînement (67932 produits), 10 % pour la validation (8492 produits), et 10 % pour le test 8492 produits). Cette répartition a été effectuée en veillant à préserver la distribution initiale des classes au sein de chaque ensemble, afin de garantir la représentativité des données tout au long du processus de modélisation. Nous avons ensuite réalisé l'ensemble du preprocessing sur l'ensemble d'entraînement puis les transformations apprises vont être appliquées aux ensembles de validation et de test. Cela évitera une fuite de données et garantit une évaluation impartiale des performances.

Nos données textuelles ont été préparées selon le processus suivant, en amont de toute étape de modélisation.

a) Concaténation des variables textuelles explicatives

Ce travail de preprocessing a commencé par la concaténation des colonnes désignations et descriptions. La concaténation a efficacement permis d'éliminer les valeurs manquantes. Elle a également contribué à réduire significativement le nombre de doublons : le taux est passé de 2,73 % pour la colonne "désignation" seule et de 12,65 % pour la colonne "description" seule à seulement 1,38 % dans la colonne concaténée (soit 935 doublons).



Nous avons choisi de supprimer ces doublons pour les raisons suivantes :

- Leur nombre est négligeable, ce qui permettra de réduire le risque de surapprentissage (overfitting).
- 19 doublons textuels exacts, associés à des catégories distinctes mais relativement proches (au maximum deux catégories distinctes), ont été recensés. La suppression de ces doublons permet de limiter les erreurs de classification et d'éviter d'accentuer le biais en faveur des catégories majoritaires.

b) Suppression des composants HTML (balises, entités)

D'autres composants HTML sont également présents. Les attributs HTML, souvent intégrés aux balises, sont supprimés simultanément, tout comme les scripts et les styles, si ceux-ci sont détectés.

c) Traduction des textes

L'hétérogénéité linguistique du jeu de données complique la classification des produits. Pour garantir des prédictions robustes et cohérentes, nous avons choisi de traduire l'ensemble des textes en français, langue déjà dominante (77 % du corpus).

Cette approche repose sur deux motivations principales :

- Cohérence linguistique : L'unification en français simplifie le prétraitement, améliore les représentations vectorielles et favorise la stabilité des modèles.
- Optimisation du temps de traitement : En traduisant uniquement les 23 % de données non francophones, le volume de travail est réduit, rendant le processus plus efficace.

L'API Google Translate, limitée à 5000 caractères par requête, a nécessité une stratégie spécifique :

- Découpage en blocs de 4900 caractères pour éviter les dépassements.
- Traduction séquentielle, avec réassemblage des segments traduits.
- Gestion des échecs, en conservant le texte original en cas de problème.
- Détection automatique de la langue, pour ignorer les phrases déjà en français.

L'essai gratuit Google Cloud a amélioré le processus de traduction en accélérant le traitement, en garantissant une meilleure qualité

d) Tokenisation des textes

Nous avons opté pour SpaCy plutôt que NLTK en raison de ses performances supérieures : il offre une meilleure précision, un traitement plus rapide, et un support multilingue robuste, ce qui est essentiel pour gérer efficacement un dataset contenant plusieurs langues.

e) Stopwords

Suppression des stopwords, des caractères numériques, des accents, des unités, des abréviations et conversion en minuscule

f) Réduction à la racine des mots, par lemmatisation

Nous avons opté pour lemmatisation plutôt que le stemming, en raison de ses avantages clés :

- Adaptation multilingue : Notre dataset, contenant du français, de l'anglais et d'autres langues minoritaires, bénéficie du support linguistique avancé de SpaCy. Le stemming, conçu principalement pour l'anglais, pourrait altérer la qualité du traitement.

- Préservation du sens : La lemmatisation conserve la forme complète des mots ("chaise" plutôt que "chais"), garantissant des mots-clés pertinents pour la classification.
- Respect des termes commerciaux : Les descriptions de produits incluent des termes techniques (*livre, jouet, console*), mieux préservés par la lemmatisation, tandis que le stemming risque de produire des formes tronquées inexactes.

g) Vectorisation

Enfin nous avons effectué la transformation des données textuelles en vecteurs numériques exploitables par le modèle.

Plusieurs types de vectorisations ont été effectués :

- Basé sur la fréquence : TF-IDF
- Basés sur les relations contextuelles : FastText
- Modèles avancés : BERT, mBERT, BERT + mBERT

Les vectorisations, qu'elles soient basées sur la fréquence ou basées sur les relations contextuelles, ont été effectuées sur des données nettoyées, traduites en français ou non.

En revanche, pour la vectorisation avec BERT et mBERT, les données textuelles n'ont pas subi de tokenisation manuelle, de suppression des stopwords, ni de lemmatisation, conformément aux bonnes pratiques pour ces modèles.

1. BERT (français) :

- Les données textuelles ont été traduites en français avant la vectorisation.
- Un nettoyage léger a été appliqué (suppression des balises HTML et des espaces superflus), mais les mots ont été conservés dans leur forme originale pour préserver le contexte.

2. mBERT (multilingue) :

- Les données non traduites ont été utilisées, permettant à mBERT de gérer directement le multilinguisme.
- Aucune transformation linguistique n'a été effectuée (pas de lemmatisation ni suppression des stopwords), car mBERT est conçu pour comprendre le contexte indépendamment de la langue.

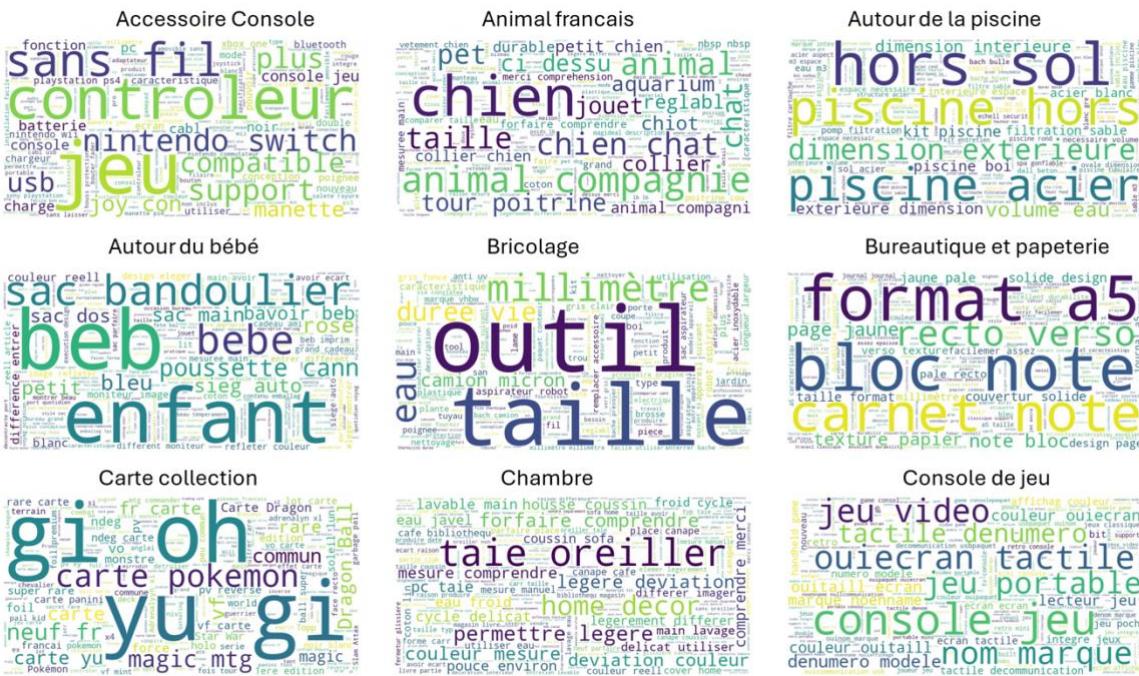
3. BERT + mBERT (approche hybride) :

- Les deux jeux de données (traduit en français pour BERT, multilingue pour mBERT) ont été combinés pour enrichir la représentation des textes.
- Cette approche visait à tirer parti des forces de chaque modèle : la précision contextuelle de BERT sur un corpus unifié et la robustesse multilingue de mBERT.

À l'issue du prétraitement de nos données textuelles, nous avons généré des WordClouds par catégorie afin d'évaluer la pertinence de notre travail de nettoyage.

La figure suivante illustre clairement que, pour la plupart des catégories, les mots dominants reflètent fidèlement leur thématique. Comparés aux premiers WordClouds, le bruit linguistique a été considérablement réduit, mettant en évidence des termes plus ciblés et représentatifs.

Cette amélioration confirme l'efficacité des étapes de nettoyage appliquées, renforçant la qualité des données pour la modélisation à venir



2- Analyse exploratoire données images :

2.1 Structure des données images :

Les données images sont stockées sous forme de **fichiers JPEG**, associées à des produits via un identifiant unique.

Format et organisation des fichiers :

- Nombre total d'images : 84 916
- Format des fichiers : JPEG (.jpg)
- Taille des images : 500 × 500 pixels
- Nom des fichiers :
 - image_<imageid>_product_<productid>.jpg
 - Exemple : image_1328823833_product_4252011401.jpg
- Association avec les métadonnées :
 - Chaque image est reliée à un productid présent dans le fichier CSV.
 - Exemple :

imageid	productid	prdtypecode
1328823833	4252011401	10
1987645322	3156789932	5

Aperçu des images :

image_1250268138_product_3831364114.jpg



image_1301378961_product_4151253320.jpg



image_1283690756_product_4061598704.jpg



image_1257233954_product_3879891010.jpg



image_1211402558_product_3290284040.jpg



2.2 Analyse globale

L'objectif de ce projet est d'optimiser la classification des produits Rakuten à partir des images en mettant en place un pipeline de traitement robuste. L'analyse exploratoire des données images a permis d'identifier plusieurs **problèmes majeurs** qui pourraient impacter la performance du modèle de classification. Ces problèmes concernent principalement la qualité des images, les biais visuels et le déséquilibre des classes. Une série de corrections et de prétraitements a été mise en place pour garantir des données propres et homogènes avant l'entraînement du modèle.

2.3 Preprocessing des données images :

a) Problèmes identifiés et solutions proposées :

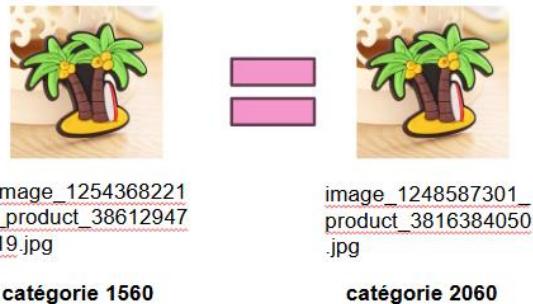
a-1) Traitement des doublons d'images : Exactes et Similaires

L'un des problèmes majeurs rencontrés dans le dataset est la présence de doublons d'images. Ces doublons peuvent être de deux types :

- **Les doublons exacts** → Deux images strictement identiques (mêmes pixels, même taille, mêmes couleurs).
- **Les doublons similaires** → Deux images quasi-identiques, mais avec de possibles légères modifications : léger recadrage, rotation minime, modification de l'éclairage .

Problème des doublons exacts :

- Certains fichiers images sont présents plusieurs fois sous des noms différents.
- Ces images peuvent apparaître dans différentes classes de produits, risquant de perturber le modèle en introduisant une confusion.



- Ces répétitions artificielles peuvent favoriser certaines catégories, rendant le modèle moins performant pour les classes sous-représentées.

- Solution envisagée :
 - Suppression systématique des doublons exacts en utilisant un algorithme de hashing (empreinte unique de l'image).
 - Vérification via un calcul de similarité pixel-par-pixel pour s'assurer qu'ils sont parfaitement identiques.

Objectif : Éliminer les biais et optimiser l'espace mémoire, sans perte d'information.

Problème des doublons visuellement similaires

- Solutions envisagées :

Cas : Images quasi-identiques associées au même *prdtypecode*

- Fusionner les occurrences et n'en garder qu'une seule

Objectif : Cela évite de biaiser le modèle en lui faisant apprendre plusieurs fois le même produit sous des formes légèrement différentes.

Cas : Images similaires mais avec des *prdtypecode* différents

- Garder ces images mais leur attribuer un poids réduit dans l'apprentissage

Objectif : Peuvent être utiles car elles représentent des produits différents sous un angle similaire.

Méthodes appliquées :

Calcul de similarité via des descripteurs d'images (ORB, SIFT, ou Hash perceptuel) pour détecter les images fortement similaires.

Regroupement des images similaires en clusters, permettant une analyse plus fine :

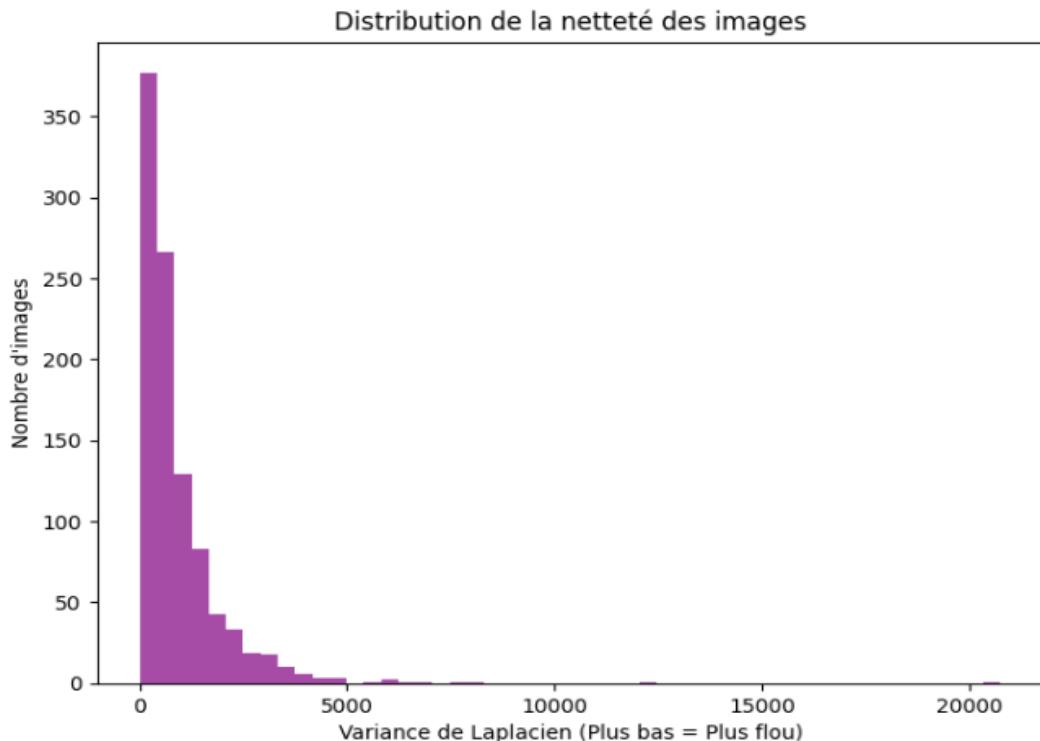
- Si un cluster contient plusieurs images du même prdtypecode, on ne garde qu'une seule instance.
- Si un cluster contient des prdtypecode différents, les images sont conservées mais avec un facteur de pondération (on peut réduire leur impact dans l'entraînement).

Impact du traitement des doublons :

- Élimination des doublons exacts → Réduction du sur-apprentissage sur certaines classes.
- Gestion des images similaires → Meilleure diversité des données pour un modèle plus généraliste.
- Optimisation du stockage et du temps d'entraînement → Moins d'images inutiles, meilleure efficacité.

a-2) Exploration et traitement des images floues :

L'analyse des images du dataset a révélé que certaines d'entre elles souffrent de **problèmes de netteté**, ce qui peut **compromettre la qualité de l'apprentissage du modèle CNN**. Ce problème doit être traité pour éviter que le modèle n'apprenne sur des données bruitées et inexploitables.



La plupart des images sont trop floues pour permettre une bonne identification du produit.
Le flou peut être dû à plusieurs facteurs :

- Mauvaise prise de vue (mouvement lors de la capture).
- Compression trop forte de l'image (artefacts de compression JPEG).
- Problèmes de mise au point de l'appareil.
 - Un modèle CNN s'appuie sur les textures, les formes et les détails d'une image. Si ces informations sont absentes à cause du flou, le modèle ne pourra pas apprendre correctement.
 - Impact sur la classification :
 - Le modèle risque d'attribuer des prédictions incorrectes car il ne pourra pas extraire des caractéristiques claires.
 - Le bruit visuel peut créer une confusion entre classes proches, rendant le modèle moins performant.
 - Les classes minoritaires peuvent être affectées plus fortement, car chaque image compte davantage dans leur apprentissage.

Détection automatique des images floues :

Nous avons privilégié une analyse de variance du filtre Laplacien pour mesurer le degré de netteté d'une image.

- Un Laplace Variance élevé signifie que l'image est bien nette.
- Une valeur faible indique que l'image est floue (manque de variations de pixels).

☒ Seuil empirique : Une analyse statistique est effectuée pour identifier un seuil au-delà duquel une image est considérée comme floue.

Traitement des images floues :

Trois solutions sont envisagées :

Cas : Images très floues et inutilisables :

- Suppression pure et simple
Ces images sont éliminées pour éviter d'introduire du bruit dans l'entraînement.

Cas : Images légèrement floues mais récupérables

- Amélioration de la netteté avec un filtrage adaptatif
Un algorithme de sharpening (accentuation des contours) est appliqué pour rendre l'image plus exploitable.
- A noter qu'il n'y a pas d'images inexploitables dans notre dataset.**

Ci-dessous résultat de l'application du filtre lapacien



Image_1326693143_product_4243732791 - Original



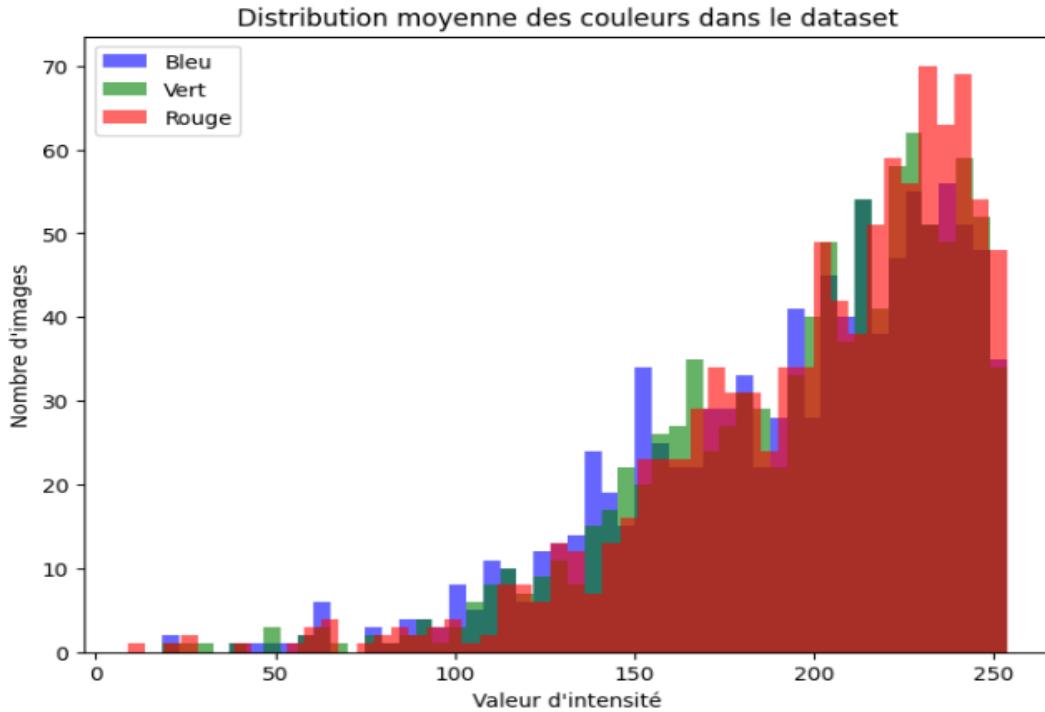
Image_1326693143_product_4243732791 – Traitement netteté

Objectifs :

- Meilleure extraction des caractéristiques visuelles → Un CNN s'appuie sur les détails de l'image, et une image nette lui permet d'apprendre plus efficacement.
- Réduction du bruit dans le dataset → Moins de confusion dans la classification.
- Amélioration du score de précision et du F1-score → Un dataset propre permet un modèle plus robuste et plus performant.

En appliquant des filtres de détection et des corrections adaptées, on garantit une meilleure qualité d'apprentissage et une classification plus précise des produits Rakuten.

a-3) Biais de couleur (Prédominance du rouge) :



L'analyse exploratoire des images a révélé un déséquilibre colorimétrique, avec une prédominance marquée des tons rouges dans plusieurs catégories de produits. Ce biais peut nuire à la performance du modèle en influençant son apprentissage de manière non souhaitée.

Les images présentent une distribution non uniforme des couleurs, avec une forte dominance du rouge dans plusieurs classes.

Ce problème peut être causé par :

- La prise de vue (éclairage ambiant riche en rouge, fond rouge utilisé pour certains produits).
- Le traitement des images (filtres ou corrections appliqués avant la mise en ligne).
- Le dataset lui-même, s'il contient des catégories spécifiques où le rouge est plus fréquent.

Impact :

- Le modèle pourrait se baser uniquement sur la couleur au lieu de considérer la forme et la texture des objets.
- Les produits d'une autre couleur pourraient être mal classés car le modèle pourrait associer certaines catégories uniquement aux tons rouges.
- Perte de généralisation : Le modèle pourrait être efficace uniquement sur des produits avec des caractéristiques colorimétriques similaires, mais échouer face à des produits d'autres couleurs.

Détection du biais de couleur :

- Analyse des histogrammes de couleur pour observer la distribution des pixels en R (Rouge), G (Vert) et B (Bleu).
- Calcul de la moyenne et de l'écart-type des canaux de couleur pour détecter une dominance excessive d'une couleur.

- Vérification si certaines classes ont des valeurs de rouge systématiquement plus élevées que les autres.
 - Résultats observés : Dans certaines catégories, le canal Rouge (R) est en moyenne 10 à 15% plus élevé que les canaux Vert (G) et Bleu (B).
Ce déséquilibre est plus marqué pour certaines catégories spécifiques de produits.

Solutions envisagées :

1 -Réduction de la saturation et normalisation des couleurs :

- Appliquer une correction de saturation pour homogénéiser l'intensité des couleurs.
 - Permet de conserver les informations de couleur tout en limitant leur influence excessive.

Exemple :

- Réduction de 15% de la saturation sur les images où le canal rouge est dominant.
- Ajustement automatique pour équilibrer les intensités des canaux R, G et B.

2- Égalisation d'histogramme pour améliorer l'équilibre des couleurs

- Redistribuer les niveaux de couleur pour éviter qu'une seule couleur domine.**
Appliquée uniquement aux images où la dominance du rouge est excessive.

Peut altérer légèrement certaines images où l'équilibre était déjà bon.

3 -Diversification des données (Data Augmentation) :

- Créer artificiellement des images avec des variations de couleur.
Transformation des couleurs (inversion, modification des teintes).
Réduction des biais en augmentant la diversité colorimétrique des images dans l'ensemble d'apprentissage.

Exemple :

- Modification aléatoire de la teinte pour éviter que le modèle ne s'appuie trop sur le rouge.
- Transformation HSV (Hue-Saturation-Value) pour générer des variantes d'images avec des tons plus neutres.

Impact :

- Meilleure robustesse du modèle face à des produits de différentes couleurs.
- Réduction des erreurs de classification basées uniquement sur la couleur.
- Amélioration de la généralisation : Le modèle pourra mieux reconnaître les produits indépendamment de leurs couleurs dominantes.

- Ci-dessous résultat de l'application de la correction de saturation :**

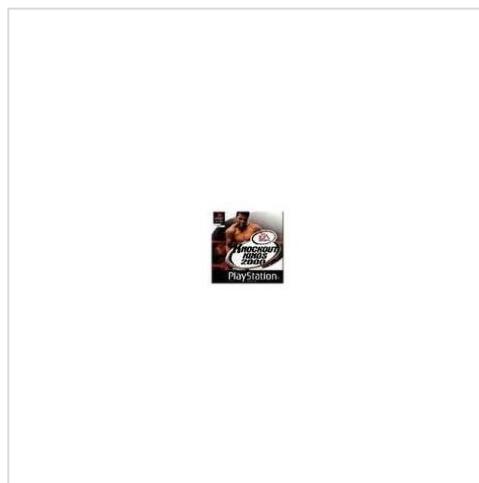
Image_1328700648_product_4250701113.jpg



- On note néanmoins l'apparition d'un fond bleu clair, à voir si cela impact les résultats auquel cas il faudra réduire l'intensité du traitement.

a-4) Analyse du contenu dans les images :

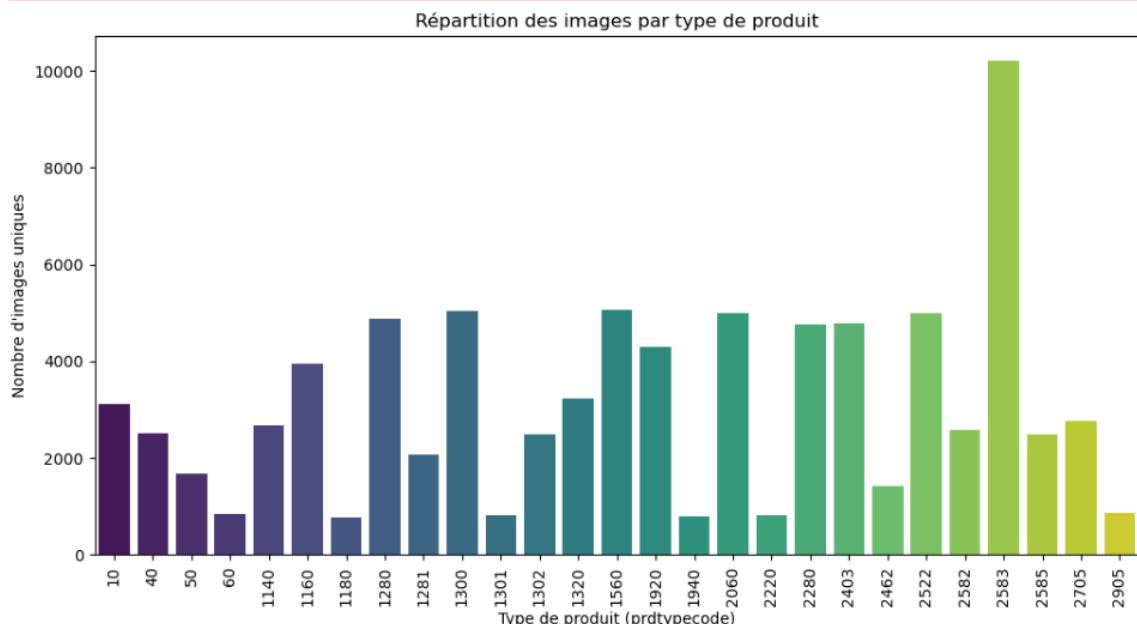
L'analyse exploratoire des images a révélé qu'un certain nombre d'images ne présentaient un contenu exploitable de très petite taille. En effet la photo dans l'image entourée d'une bordure très importante. Réduit considérable l'information exploitable et alimente nos modèles avec des informations vides.



La solution proposée consiste à mesurer la dimension exploitable afin de déterminer si elle est suffisamment grande. Dans le cas contraire, une opération de rognage de la photo sera effectuée, suivie d'un zoom pouvant atteindre jusqu'à 50 % de la taille de l'image, afin de conserver une résolution exploitable.

a-5) Analyse du déséquilibre des classes :

Chaque image appartient à une catégorie de produit (prdtypecode). Si certaines classes ont beaucoup plus d'images que d'autres, cela peut fausser l'apprentissage du modèle de classification et diminuer ses performances sur les classes minoritaires.



L'histogramme fourni montre que :

- Certaines classes contiennent plus de 6 000 images, et l'une d'elles dépasse même 10 000 images.
- D'autres classes n'ont que quelques centaines d'images, voire moins de 1 000.
- Un écart très important entre la classe la plus représentée et la moins représentée, ce qui entraîne un risque de biais dans l'entraînement du modèle.

Impact :

- Le modèle apprendra beaucoup mieux les classes majoritaires, car elles sont plus nombreuses.
- Les classes rares risquent d'être mal classées ou ignorées, car le modèle ne les verra pas assez souvent en entraînement.
- Une précision globale trompeuse : le modèle peut sembler performant en test, mais avec un F1-score très faible sur les classes sous-représentées.

Solutions envisagées :

Réduction des classes majoritaires (Downsampling) :

Diminuer le nombre d'images des classes sur-représentées en sélectionnant aléatoirement un sous-ensemble équilibré.

- Permet d'éviter que le modèle sur-apprenne les classes dominantes.
- Risque : Suppression d'images utiles et perte d'informations.

Augmentation des classes sous-représentées (Upsampling) :

Créer artificiellement des images supplémentaires pour les classes sous-représentées grâce à la Data Augmentation.

- Ajoute de la diversité et augmente le nombre d'images de ces classes.
- Peut introduire des biais artificiels si les transformations sont trop répétitives.

Techniques utilisées :

- Rotation et translation aléatoires.
- Modification de la luminosité et du contraste.
- Déformation mineure pour créer plus de diversité.

Pondération des classes dans la fonction de perte :

Ajuster les poids des classes dans la loss function pour donner plus d'importance aux classes sous-représentées.

- Permet de ne pas toucher aux données et d'ajuster directement l'apprentissage du modèle.
- Très efficace pour équilibrer un dataset sans suppression de données.
- Peut rendre l'apprentissage plus instable si le déséquilibre est trop fort.

Rééquilibrage via SMOTE :

Créer de nouvelles images synthétiques en interpolant des images existantes dans les classes rares.

- Moins biaisé que le simple oversampling.
- Fonctionne mieux pour les données numériques, mais peut être appliqué à des images avec des transformations adaptées.

Impact :

Les classes sous-représentées seront mieux prises en compte par le modèle.

Le modèle généralisera mieux et fera moins d'erreurs de classification.

L'évaluation des performances (F1-score, Recall) sera plus fiable et équilibrée.

b) Préparation des données pour l'entraînement :

Après la correction des problèmes identifiés dans le dataset (**suppression des doublons, correction du biais de couleur, rééquilibrage des classes, etc.**), les images sont désormais prêtes à être utilisées pour entraîner le modèle CNN.

c) Séparation des données

Evidemment comme pour la partie textuelle les images sont réparties en trois ensembles (générées en même temps et donc en concordance avec la partie textuelle :

- Train (80%) → Utilisé pour l'apprentissage du modèle
- Validation (10%) → Permet d'ajuster les hyperparamètres et d'éviter le sur-apprentissage
- Test (10%) → Sert à mesurer la performance finale

d) Vectorisation et normalisation des images :

Cette phase implique trois transformations clés :

1. Vectorisation des images pour qu'elles puissent être traitées par le réseau de neurones.
2. Normalisation des valeurs des pixels pour améliorer l'apprentissage.
3. Application de techniques avancées comme ZCA Whitening et Data Augmentation, selon les besoins.
4. Dans un réseau de neurones convolutifs (CNN), les images doivent être transformées en tenseurs numériques normalisés afin d'optimiser l'apprentissage et d'améliorer la stabilité du modèle.

Vectorisation des images

La vectorisation consiste à transformer une image en une représentation numérique que le réseau de neurones peut traiter.

Lorsque l'on charge une image, elle est représentée sous la forme d'une matrice de pixels, avec des valeurs comprises entre 0 et 255 pour chaque canal de couleur (Rouge, Vert, Bleu - RGB).

Représentation d'une image 224×224 pixels avec 3 canaux :

$$X_{\text{image}} = (224, 224, 3)$$

□ Chaque pixel est stocké sous forme d'une valeur entière entre 0 et 255.

Impact :

- Cela permet au modèle d'appliquer des opérations mathématiques sur les images.
- Un CNN analyse les motifs, les textures et les formes à partir de ces valeurs numériques.
- La vectorisation permet de charger plusieurs images sous forme de tenseurs et de les passer au modèle en batch.

Normalisation des images :

Une normalisation Min-Max est appliquée directement après la vectorisation pour améliorer l'entraînement du modèle.

Impact :

- Un réseau de neurones fonctionne mieux lorsque les valeurs des pixels sont comprises entre 0 et 1 plutôt qu'entre 0 et 255.
- Cela évite les problèmes d'échelle, stabilise les poids du modèle et accélère la convergence.
- La normalisation réduit le risque d'explosions des gradients, ce qui empêche le modèle d'apprendre trop lentement ou de devenir instable.

□ Transformation appliquée immédiatement lors du chargement :

Pixel (x, y)	R (Brut)	G (Brut)	B (Brut)	R (Normalisé)	G (Normalisé)	B (Normalisé)
(0,0)	125	200	50	0.49	0.78	0.20
(0,1)	130	210	60	0.51	0.82	0.23

- Les valeurs sont désormais comprises entre 0 et 1, ce qui permet un apprentissage plus rapide et plus stable.

Techniques avancées de transformation d'images :

Certaines méthodes complémentaires peuvent être appliquées pour améliorer la qualité des images avant l'entraînement :

Data Augmentation

Permet de générer artificiellement plus d'images pour éviter le sur-apprentissage et améliorer la généralisation du modèle.

Techniques utilisées :

- Rotation aléatoire → Introduit des variations dans l'orientation des objets.
 - Zoom et translation → Évite que le modèle devienne trop sensible à la position des objets.
 - Modification de la luminosité et du contraste → Aide le modèle à apprendre à détecter les
- La Data Augmentation est particulièrement utile lorsque certaines classes sont sous-représentées.

Impact :

Les images sont converties en tenseurs normalisés directement exploitables par le modèle CNN. Aucune étape supplémentaire de transformation n'est nécessaire après la vectorisation, car la normalisation est directement appliquée.

Les techniques avancées (Data Augmentation, conversion en noir et blanc, etc) peuvent être utilisées si nécessaire pour améliorer la qualité des images et la généralisation du modèle.

V Modélisation :

Le projet Rakuten relève du domaine de l'apprentissage supervisé et plus précisément de la classification multi-classes. L'objectif est d'attribuer à chaque produit une catégorie spécifique en se basant sur des données textuelles (désignation et description) et des images associées.

Métriques retenues :

Le F1-score pondéré est utilisé comme principale métrique afin de compenser le déséquilibre des classes et fournir une évaluation plus représentative que l'accuracy. Il prend en compte la précision (qualité des prédictions correctes) et le rappel (capacité à identifier toutes les instances d'une classe).

D'autres outils d'évaluation ont été utilisés pour affiner l'analyse des performances du modèle :

- Matrice de confusion : Pour identifier les erreurs de classification et visualiser les confusions entre catégories.
- Rapport de classification : Pour évaluer les performances par classe à travers des métriques comme la précision, le rappel et le F1-score.
- Suivi des courbes d'apprentissage : Pour détecter d'éventuels phénomènes de sur-ajustement (overfitting) ou de sous-ajustement (underfitting) au cours de l'entraînement.
- Comparaison avant et après fusion multimodale : Pour mesurer l'impact de l'intégration des images dans la classification et évaluer la complémentarité entre texte et image.

1- Approche méthodologique :

Nous avons adopté une approche modulaire en entraînant séparément des modèles sur les données textuelles (CamemBERT) et visuelles (ResNet) avant de fusionner leurs prédictions. Cette méthodologie permet :

- D'optimiser chaque modalité indépendamment pour capturer leurs spécificités.
- De tester plusieurs stratégies de fusion (Max Rule, Soft Voting, Stacking).
- D'exploiter la complémentarité entre texte et image pour améliorer la classification.

Cette approche vise à maximiser la performance en combinant efficacement les deux sources d'information.

Interprétation des 27 classes :

Par souci de clarté, voici les codes des classes que notre algorithme devra prédire

Code Rakuten	Description	Encodé
10	"Livre occasion"	0
40	"Jeu vidéo, accessoire tech."	1
50	"Accessoire Console"	2
60	"Console de jeu"	3
1140	"Figurine"	4
1160	"Carte Collection"	5
1180	"Jeu Plateau"	6
1280	"Jouet enfant, déguisement"	7

1281	"Jeu de société"	8
1300	"Jouet tech"	9
1301	"Paire de chaussettes"	10
1302	"Jeu extérieur, vêtement "	11
1320	"Autour du bébé"	12
1560	"Mobilier intérieur"	13
1920	"Chambre"	14
1940	"Cuisine"	15
2060	"Décoration intérieure"	16
2220	"Animal"	17
2280	"Revues et journaux"	18
2403	"Magazines, livres et BDs"	19
2462	"Jeu occasion"	20
2522	"Bureautique et papeterie"	21
2582	"Mobilier extérieur"	22
2583	"Autour de la piscine"	23
2585	"Bricolage"	24
2705	"Livre neuf"	25
2905	"Jeu PC"	26

2- Modélisation pour la Classification des Données Textuelles

Dans notre projet de classification des produits Rakuten, nous avons adoptés deux approches afin d'évaluer les performances des modèles et d'optimiser la classification des produits :

1. **Une approche basée sur le Machine Learning classique**, utilisant différentes techniques de vectorisation et différents algorithmes traditionnels.
2. **Une approche basée sur des modèles de type Transformer (CamemBERT)**, exploitant des architectures avancées pour capturer le contexte sémantique des textes.

2.1 Approche Machine Learning Classique

a) Workflow

Nous avons adopté une approche systématique pour identifier la meilleure combinaison entre vectorisation des textes et hyperparamètres des modèles. Afin d'assurer une classification plus équilibrée, nous avons mis en place deux stratégies de gestion du déséquilibre des classes :

- Ajustement des poids dans les modèles : L'option `class_weight='balanced'` a été utilisée pour compenser les écarts de fréquence entre les classes.

- Rééquilibrage des données par rééchantillonnage : Une combinaison de SMOTE (pour augmenter les classes sous-représentées) et d'undersampling (pour réduire l'influence des classes majoritaires) a été appliquée avant l'entraînement des modèles.

a-1) Génération des vectorisations

Comme explicité dans la partie preprocessing, nous avons testé plusieurs techniques de vectorisation : TF-IDF et embeddings FastText, avec différents paramètres pour identifier les meilleures représentations textuelles.

a-2) Sélection des Meilleures Combinaisons Modèle + Vectorisation

Chaque modèle a été optimisé via GridSearchCV et sélectionné en fonction du F1-score pondéré.

SVM (LinearSVC) : Test de plusieurs valeurs du paramètre de régularisation C ([0.01, 0.1, 1, 10]) avec max_iter=5000.

Random Forest (RF) : Exploration de n_estimators ([50, 100, 200]) et max_depth ([10, 20, None]) pour ajuster la complexité du modèle.

Régression Logistique (SGDClassifier) : Optimisation des hyperparamètres alpha, penalty='l2', loss='log_loss', et max_iter=1000.

FastText supervised: Test de plusieurs valeurs du learning rate ([0.01, 0.3, 0.5]), de la dimension des embeddings ([100, 150, 200, 300]) et du nombre d'époques ([50, 100, 200]). WordNgrams=2 a été systématiquement utilisé pour améliorer la compréhension des expressions clés.

a-3) Évaluation et Analyse des Performances

Après sélection des meilleures configurations, les modèles ont été évalués sur un jeu de test avec :

- Analyse des erreurs via la matrice de confusion pour identifier les confusions fréquentes.
- Comparaison des performances entre les modèles entraînés sur les données en langue originale et celles traduites en français.

b) Résultats :

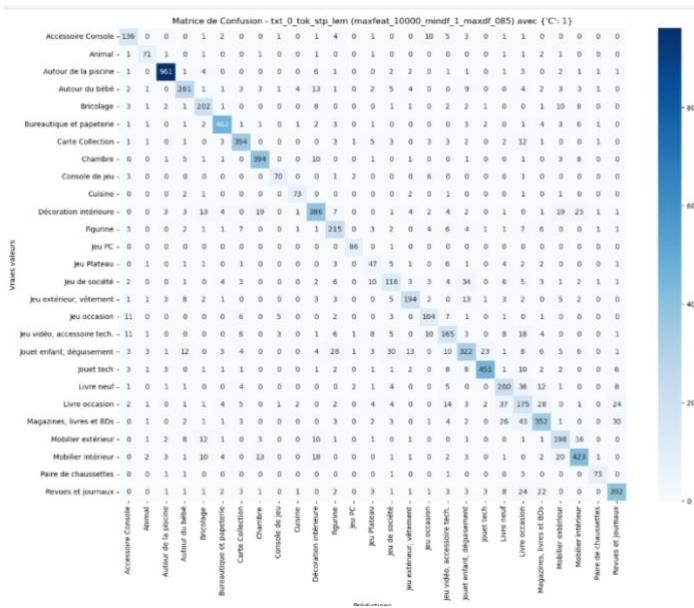
b-1) SVM, Random Forest, Regression Logistique

Nous avons d'abord évalué les performances des modèles de Machine Learning classiques sur la classification des produits Rakuten en testant SVM, Random Forest et Régression Logistique (via SGDClassifier avec loss='log_loss'). Le SVM s'est révélé être le modèle le plus performant, atteignant un F1-score de 0.8172 avec le texte original, surpassant la version traduite qui obtient 0.8036. Le Random Forest, bien que légèrement en retrait, reste compétitif avec un F1-score de 0.7952, confirmant sa robustesse sur ces données. En revanche, le SGDClassifier affiche des performances plus modestes, avec un F1-score de 0.7052, se positionnant comme le moins performant des trois modèles.

Nous avons ensuite analysé l'impact du rééquilibrage des classes via SMOTE + undersampling. Contrairement au Random Forest et au SGDClassifier, le SVM ne bénéficie pas significativement du rééquilibrage, son F1-score diminuant légèrement (0.8141 vs 0.8172), suggérant que l'option class_weight='balanced' était déjà suffisante pour gérer le déséquilibre des classes. À l'inverse, le Random Forest affiche une légère amélioration avec SMOTE (0.7967 vs 0.7952), tandis que le SGDClassifier est celui qui en profite le plus, avec un gain de 5 points en F1-score (0.7572 vs 0.7052). Cela montre que SGDClassifier était fortement impacté par le déséquilibre des classes et que SMOTE a corrigé cette limitation.

Meilleure configuration: txt_9_tok_stp_lem - maxfeat_10000_mindf_1_maxdf_085
F1 Score Weighted sur Validation: 0.8172
Meilleurs hyperparamètres: ('C': 1)

Classification Report sur Validation:				
	precision	recall	f1-score	
			support	
Accessoire Console	0.74	0.82	0.77	166
Animal	0.82	0.87	0.84	82
Autour de la piscine	0.98	0.97	0.98	989
Autour du bébé	0.83	0.81	0.82	324
Bricolage	0.79	0.83	0.81	244
Bureautique et papeterie	0.93	0.93	0.93	496
Carte Collection	0.88	0.89	0.89	396
Chambre	0.91	0.92	0.91	427
Console de jeu	0.86	0.84	0.85	83
Cuisine	0.88	0.90	0.89	81
Décoration intérieure	0.83	0.78	0.88	497
Figurine	0.73	0.81	0.77	267
Jeu PC	0.92	0.99	0.96	87
Jeu Plateau	0.58	0.62	0.55	76
Jeu de société	0.68	0.56	0.58	287
Jeu extérieur, vêtement	0.84	0.78	0.81	249
Jeu occasion	0.71	0.74	0.72	141
Jeu vidéo, accessoire tech.	0.66	0.66	0.66	251
Jouet enfant, déguisement	0.76	0.66	0.71	487
Jouet tech	0.93	0.89	0.91	585
Livre neuf	0.66	0.72	0.69	276
Livre occasion	0.49	0.56	0.52	311
Magazines, livres et BDs	0.78	0.74	0.76	475
Mobilier extérieur	0.72	0.77	0.74	257
Mobilier intérieur	0.84	0.84	0.84	585
Paire de chaussettes	0.98	0.91	0.91	88
Revues et journaux	0.84	0.83	0.83	473
accuracy			0.82	8432
macro avg	0.79	0.88	0.79	8432
weighted avg	0.82	0.82	0.82	8432



Dans le modèle SVM le plus performant certaines catégories affichent un F1-score supérieur à 0.90, témoignant d'une excellente classification :

- Autour de la piscine (F1 = 0.97, Recall = 0.97)
- Jeu PC (F1 = 0.96, Recall = 0.99)
- Carte Collection (F1 = 0.89, Recall = 0.89)
- Mobilier intérieur (F1 = 0.89, Recall = 0.90)

D'autres classes affichent un F1-score inférieur à 0.75, signalant une difficulté pour le modèle à les classifier correctement.

- Livre Occasion (0.52) : Confusion avec Magazines, Livres et BDs (175 erreurs).
- Jeu Plateau (0.55) : Souvent classé en Jeu de Société (34 erreurs).
- Jeu vidéo, accessoire tech. (0.72) : Mélange avec Accessoire Console (200 erreurs).

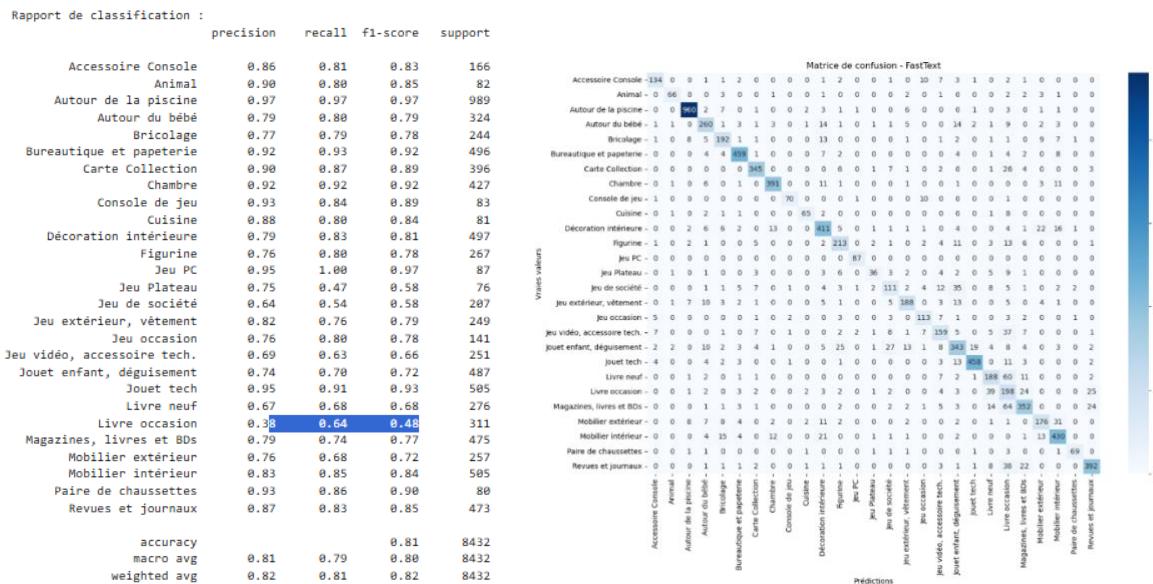
b-2) FastText supervised :

L'évaluation du modèle FastText supervisé a permis d'identifier la configuration offrant les meilleures performances, avec un F1-score pondéré de 0.8173 sur l'ensemble de validation. Cette configuration optimale repose sur un learning rate de 0.50, une dimension des embeddings de 100 et un entraînement sur 100 époques.

L'analyse des différentes configurations testées révèle plusieurs tendances :

- Taille des embeddings : Augmenter la dimension des embeddings à 200 n'a pas permis d'améliorer les résultats. Au contraire, une légère diminution des performances a été observée, avec un F1-score de 0.8163 après 100 époques et 0.8156 après 50 époques. Cela suggère qu'une taille de 100 est suffisante pour capturer les relations sémantiques essentielles sans introduire de complexité inutile.
- Nombre d'époques : L'entraînement sur 200 époques n'a pas apporté de bénéfice notable. Les performances sont restées stables, voire légèrement inférieures à celles obtenues avec 100 époques, indiquant qu'un entraînement prolongé n'améliore pas nécessairement la généralisation du modèle.

- Learning rate : Un learning rate plus bas (0.30) a conduit à des performances légèrement inférieures (F1-score de 0.8151), ce qui montre que le modèle bénéficie d'un taux d'apprentissage plus élevé (0.50) pour converger efficacement.



L'analyse comparative entre FastText et SVM révèle que les deux modèles atteignent des performances quasiment identiques, avec un F1-score pondéré de 0.8173 pour FastText et 0.8172 pour le SVM. Cette parité est remarquable, sachant que FastText repose sur des embeddings appris tandis que le SVM exploite une vectorisation TF-IDF classique.

Cependant, certaines catégories demeurent problématiques dans les deux approches. Par exemple, FastText montre une légère amélioration dans la classification de la catégorie "Jeu Plateau", bien que l'écart soit marginal. À l'inverse, le SVM se distingue par une plus grande robustesse pour la classe "Livre occasion", qui reste l'une des catégories les plus souvent confondues.

2.2 Approche basée sur CamemBERT

En parallèle de l'approche classique, nous avons exploré l'utilisation de **CamemBERT**, un modèle de type Transformer spécialisé pour le français. L'objectif était d'optimiser ses hyperparamètres afin d'obtenir les meilleures performances possibles, tout en évaluant son apport par rapport aux modèles classiques (SVM, FastText). Contrairement aux méthodes traditionnelles de vectorisation, **CamemBERT** permet une meilleure capture de la sémantique des textes, en tenant compte du contexte global et des relations entre les mots, ce qui le rend particulièrement adapté à l'analyse des descriptions produits.

a) Entraînement Initial et Baseline

Le modèle CamemBERT a été entraîné en utilisant les paramètres par défaut de la bibliothèque Transformers, avec une configuration optimisée pour le GPU. L'entraînement s'est déroulé sur 3 époques, avec une batch size de 16 par GPU, et un weight decay fixé à 0.01 pour éviter le surapprentissage. Le learning rate par défaut a été utilisé dans un premier temps afin d'établir une baseline de performance.

Ce modèle de référence (baseline) a immédiatement atteint un F1-score pondéré de 0.8834, surpassant largement les modèles de Machine Learning traditionnels comme SVM (0.8172) et FastText (0.8173). Cette performance montre la capacité de CamemBERT à capturer la richesse sémantique des descriptions produits et justifient ainsi l'orientation vers une optimisation plus fine des hyperparamètres afin d'exploiter pleinement le potentiel du modèle.

b) Optimisation des Hyperparamètres

Recherche du Meilleur Learning Rate

Après avoir établi une baseline avec les paramètres par défaut du modèle CamemBERT, nous avons cherché à optimiser le learning rate, qui est l'un des hyperparamètres les plus influents dans l'entraînement des modèles de deep learning.

Pour cela, nous avons testé plusieurs valeurs de learning rate ($2e-5$, $3e-5$, $5e-5$), en conservant une configuration stable et adaptée à un entraînement sur GPU.

Les paramètres utilisés pour ces expérimentations incluant :

- Nombre d'époques augmenté à 10 afin d'observer l'évolution de la convergence sur une durée plus longue.
- Un scheduler "cosine" pour ajuster dynamiquement le learning rate
- Un warmup de 500 steps pour éviter des mises à jour trop brutales des poids
- Une accumulation de gradients (steps = 2) pour simuler un batch plus grand

Ajustement du Weight Decay

Après avoir déterminé le meilleur learning rate, nous avons testé trois valeurs de weight decay (0.1, 0.0001, 0.01) pour contrôler la régularisation et éviter le surajustement.

GridSearch sur Plusieurs Hyperparamètres Clés

Une fois ces premiers réglages effectués, afin de stabiliser l'entraînement et maximiser les performances, nous avons réalisé une recherche plus approfondie :

- Gradient Accumulation Steps : [1, 4]
- Type de Scheduler : ["linear", "cosine_with_restarts"]
- Nombre de Warmup Steps : [500, 1000]

L'objectif était de déterminer la combinaison optimale favorisant la convergence rapide et la généralisation du modèle.

Stabilisation de l'Apprentissage avec le Dropout

Lors de l'entraînement, nous avons observé une instabilité dans la convergence, suggérant un risque de surajustement. Pour y remédier, nous avons intégré un dropout de 0.3, qui désactive aléatoirement certains neurones afin d'améliorer la robustesse du modèle.

Les tests ont montré, sans dropout, une convergence plus rapide, traduisant une assimilation efficace des motifs présents dans les données d'apprentissage. Toutefois, cette rapidité d'apprentissage s'accompagne d'un écart plus important entre la perte d'entraînement et celle de validation, ce qui indique un risque accru de surajustement.

À l'inverse, l'ajout d'un dropout de 0.3 a permis d'obtenir une convergence plus progressive et plus stable, favorisant ainsi une meilleure généralisation sur les données de validation. Bien que le modèle apprenne légèrement plus lentement, cette configuration limite le surapprentissage et améliore la robustesse du modèle sur de nouvelles données.

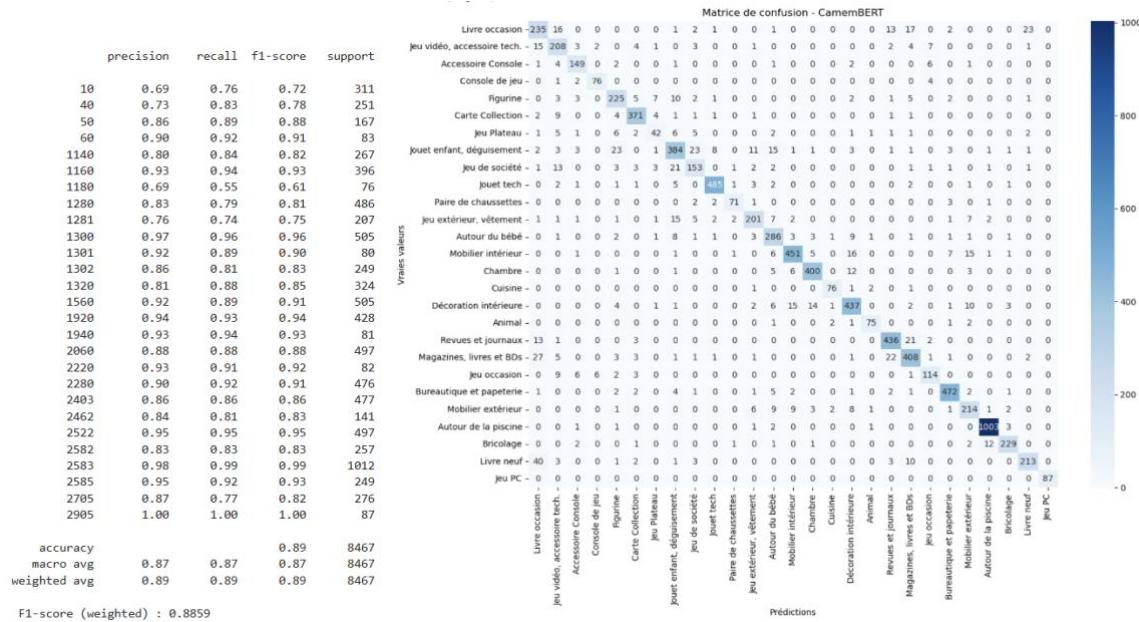
Analyse Finale et Choix du Meilleur Modèle

Après avoir testé différentes configurations et optimisé plusieurs hyperparamètres, nous avons comparé les performances obtenues avec celles du modèle CamemBERT en configuration par défaut. Les résultats ont mis en évidence que le modèle avec ses réglages initiaux restait la meilleure configuration, surpassant les ajustements apportés. Cela indique que les paramètres par défaut de CamemBERT sont déjà bien adaptés à notre tâche de classification.

Ce processus d'optimisation a néanmoins permis :

- D'explorer plusieurs stratégies d'ajustement des hyperparamètres, notamment le learning rate, le weight decay et le dropout.
- De valider la robustesse des paramètres par défaut, qui offrent des performances optimales sur notre jeu de données.
- D'observer que les ajustements testés n'ont pas permis d'améliorer significativement la baseline initiale.

2.3 Analyse résultats CamemBERT



L'évaluation du modèle CamemBERT optimisé sur le jeu de validation a permis d'atteindre un F1-score pondéré de 0.8859, marquant une amélioration significative par rapport aux modèles de machine learning traditionnels testés précédemment. Avec une accuracy de 0.89 et un Macro-F1 Score de 0.87, le modèle démontre une capacité accrue à capturer les nuances sémantiques des descriptions de produits. Ces résultats confirment sa supériorité par rapport aux approches classiques basées sur TF-IDF, FastText et SVM, qui peinent à modéliser des relations complexes dans les textes.

L'analyse détaillée par catégorie met en évidence des performances très élevées sur certaines classes :

- Carte Collection (F1-score = 0.93), Mobilier intérieur (0.95) et Jeu PC (1.00) sont classifiés avec une grande précision.
- En revanche, des confusions persistent entre certaines catégories proches sémantiquement :
 - "Jeu Plateau" (F1-score = 0.61) et "Jeu de société" (0.75) sont régulièrement confondus.

- "Livre Occasion" (0.72) est fréquemment mal classé en "Magazines, livres et BDs", suggérant un manque de distinction textuelle.

L'analyse croisée avec la matrice de confusion confirme ces observations : les classes sémantiquement proches restent les plus sujettes aux erreurs, ce qui montre les limites du modèle lorsqu'il s'agit de différencier des catégories partageant des descriptions similaires. En revanche, les catégories visuellement et contextuellement bien définies, comme "Carte Collection" et "Mobilier intérieur", sont identifiées avec une précision quasi parfaite, traduisant la capacité du modèle à bien saisir leur spécificité.

Ces résultats soulignent la robustesse de CamemBERT pour la classification des produits Rakuten, tout en mettant en évidence certaines limites dans la distinction des catégories sémantiquement proches. Pour pallier ces difficultés, nous avons exploré des approches multimodales combinant texte et image.

3- Modélisation pour la Classification des Données Images :

3.1 Vérification pre processing Images :

La première étape consiste à vérifier que la phase data pre_processing s'est correctement déroulé :

Interprétation des valeurs après transformation :

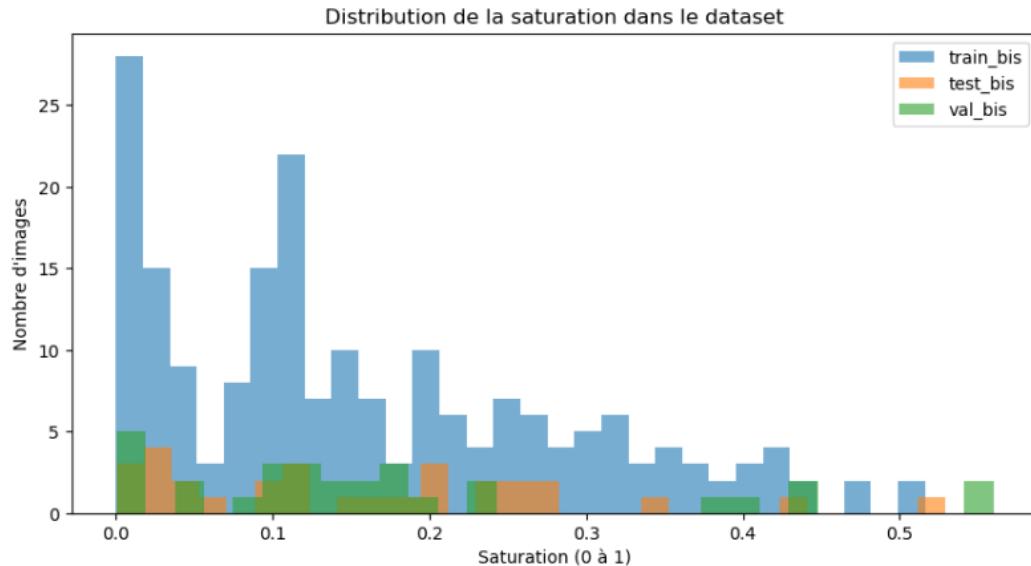
Voici ce que nous pouvons conclure à partir des valeurs moyennes de netteté, saturation, dominance du rouge et luminosité dans train_bis, test_bis et val_bis :

a) Netteté (Sharpness) :

Dataset	Sharpness moyenne
train_bis	94.67
test_bis	79.82
val_bis	99.87

- Les images de train_bis et val_bis sont plus nettes que test_bis.
- Cela pourrait être dû au fait que les images de test_bis étaient initialement plus floues ou moins contrastées.
- Selon les résultats, il faudrait songer à améliorer encore la netteté de test_bis.

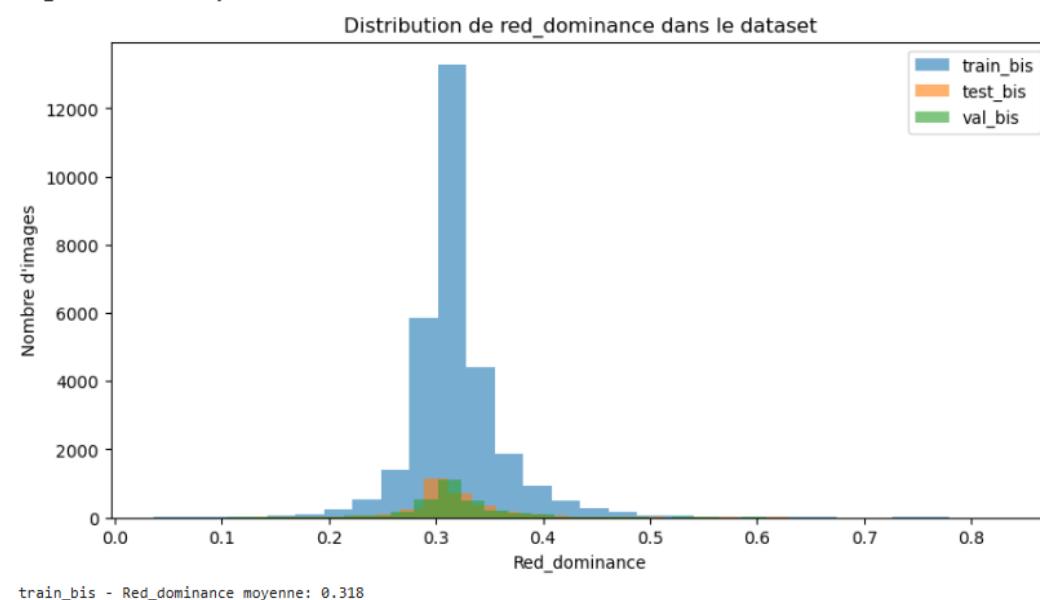
b) Saturation :



Dataset	Saturation moyenne
train_bis	0.201
test_bis	0.201
val_bis	0.224

- Les valeurs de saturation sont faibles (~ 0.2), ce qui signifie que les images restent plutôt ternes.
- L'augmentation de la saturation a bien fonctionné, mais reste discrète.
- val_bis a une saturation légèrement plus élevée, ce qui pourrait être dû à des variations dans le dataset.

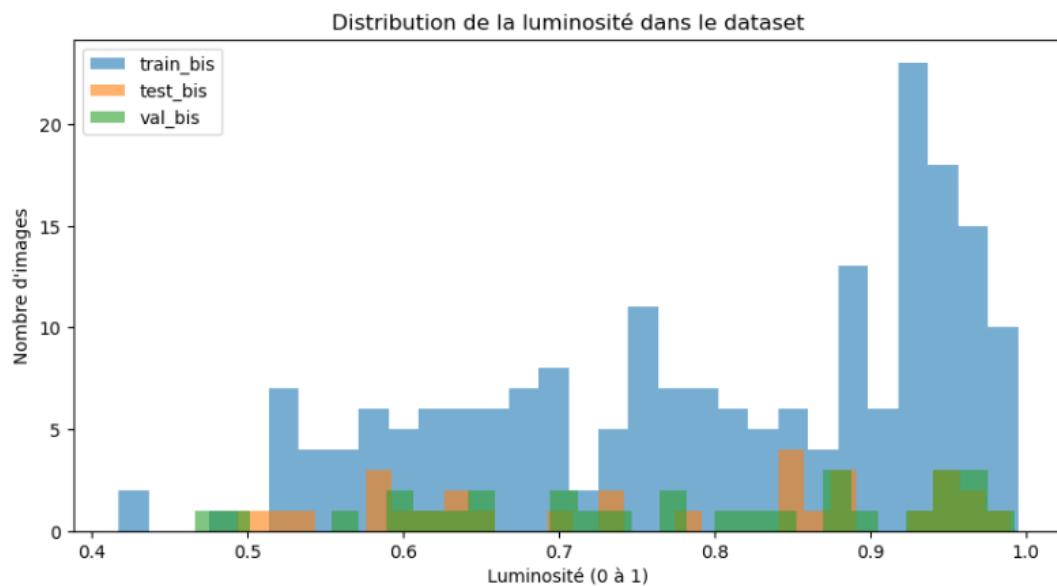
c) Dominance du Rouge :



Dataset	Red_dominance moyenne
train_bis	0.318
test_bis	0.320
val_bis	0.318

- La réduction du rouge a bien fonctionné, mais il reste encore un léger excès de rouge (~ 0.33).
- val_bis a une légère dominance du rouge plus élevée que train_bis et test_bis.
- Si le rouge impacte la classification, on peut le réduire encore légèrement ($\sim 5\%$).

d) Luminosité (Brightness) :



Dataset	Luminosité moyenne
train_bis	0.782
test_bis	0.766
val_bis	0.775

- Le dataset est globalement lumineux, avec une moyenne proche de 0.78 (assez clair).
- La correction de luminosité a uniformisé l'éclairage entre train, test et val, ce qui est positif.
- test_bis est légèrement plus sombre, mais l'écart est faible.
- Rien à faire ici, les valeurs sont homogènes.

Actions à envisager en fonction des résultats :

Transformation	Action recommandée
Netteté	Augmenter légèrement la netteté de test_bis
Saturation	Augmenter légèrement la saturation (10%) si besoin
Rouge	Réduire encore ~5% si besoin
Luminosité	Déjà équilibrée, pas besoin d'ajustement

3.2 Modélisation image :

a) Contexte et Enjeux :

L'essor des technologies basées sur l'intelligence artificielle a permis d'améliorer considérablement les performances des modèles de classification d'images. De nombreuses applications bénéficient de ces avancées, notamment dans des domaines tels que :

- La reconnaissance faciale (sécurité, authentification biométrique)
- Le diagnostic médical (classification d'imageries médicales)

Dans ce projet, nous nous intéressons à la classification multi-classes d'images, où chaque image doit être associée à l'une des 27 catégories prédéfinies. Cette classification repose sur l'extraction de caractéristiques visuelles à l'aide de modèles d'apprentissage profond (Deep Learning).

b) Définition du problème

Le problème posé consiste à attribuer un label correct à chaque image en fonction de ses caractéristiques visuelles. Cette tâche est complexe pour plusieurs raisons :

- Les images peuvent être de qualité variable :
 - Présence de bruit ou flou
 - Variations de luminosité et de contraste
 - Images partiellement obstruées ou incomplètes
- Le dataset peut être déséquilibré :
 - Certaines classes peuvent contenir significativement plus d'exemples que d'autres, créant un biais dans l'entraînement.
- L'optimisation du temps de calcul est essentielle :
 - Le modèle doit être efficace sans nécessiter des ressources computationnelles excessives, afin d'être applicable en production.

c) Objectif basé sur le benchmark Rakuten

Ce projet s'inscrit dans le cadre du défi Rakuten, un benchmark bien connu pour la classification d'images e-commerce. Dans ce benchmark, les modèles traditionnels ont obtenu un score F1 moyen de 55 % sur la tâche de classification. Notre objectif est donc d'atteindre au moins ce score et, avec un peu de chance, de le dépasser.

Pour cela, nous devons mettre en place une architecture performante, en explorant différentes stratégies de prétraitement, d'optimisation des modèles.

d) Caractéristiques des données

- Nature du problème : Classification supervisée
- Type d'entrée : Images de dimensions 224x224 pixels avec trois canaux (RGB)
- Nombre de classes : 27
- Format des images : JPG
- Taille du dataset : 84000 images divisé en train, val et test.

Liste des classes utilisées pour la classification

Les images sont classifiées en 27 catégories différentes, correspondant aux types d'articles mentionnés précédemment.

e) Contraintes et Défis :

Le projet est soumis à plusieurs contraintes techniques et méthodologiques :

- Classement fiable même pour des images de qualité variable
 - Un bon modèle doit être robuste aux perturbations (flou, contraste faible, images mal cadrées).
- Traitement des classes déséquilibrées
 - Certaines classes peuvent contenir beaucoup plus d'exemples que d'autres, nécessitant l'utilisation de techniques d'équilibrage (oversampling, data augmentation).
- Optimisation du temps de calcul
 - L'entraînement et l'inférence doivent être rapides et efficaces, en exploitant le GPU lorsque possible.
 - Le modèle doit être léger et scalable pour une éventuelle intégration en production.

Enjeux majeurs du projet :

- Obtenir un modèle précis et robuste qui généralise bien sur des images non vues.
- Minimiser les erreurs de classification tout en maintenant une bonne efficacité computationnelle.
- Explorer et comparer plusieurs architectures de modèles afin d'identifier la solution optimale.

3.3 Choix du Modèle et Optimisation :

Différents modèles ont été testés afin de déterminer celui offrant la meilleure précision et généralisation.

a-1) CNN classique :

Architecture :

L'architecture du CNN classique utilisée se compose des éléments suivants :

Model: "sequential"

- Entrée : Images de 128x128x3 (RGB)
- 4 couches convolutionnelles avec ReLU activant :
 - Conv2D (32 filtres, 3x3) → MaxPooling (2x2)
 - Conv2D (64 filtres, 3x3) → MaxPooling (2x2)
 - Conv2D (128 filtres, 3x3) → MaxPooling (2x2)
 - Conv2D (256 filtres, 3x3) → MaxPooling (2x2)
- Flatten pour transformer en vecteur 1D
- Dense (128 unités, activation ReLU)
- Dropout (0.5) pour régularisation
- Dense (27 classes, activation Softmax) pour classification

Points importants :

Nombre de paramètres raisonnable :

- 1,57 million de paramètres est un bon équilibre entre **complexité et vitesse d'entraînement**.

Ajout de la 4ème couche convulsive :

- Améliore **l'extraction des features** sur des images complexes.
- Peut cependant **augmenter le risque de sur-apprentissage** (Overfitting).

MaxPooling 2x2 après chaque Convolution :

- **Réduction de la taille des features maps**, évite le sur-ajustement.

Dense (128) avec Dropout (0.5) :

- Dropout aide à **réduire l'overfitting** en éteignant aléatoirement des neurones.

Dense (27 classes, Softmax) :

- Softmax pour une classification **multi-classes bien définie**.

Pourquoi ce modèle ?

- Un **CNN classique** est souvent utilisé comme **première approche** car :
 - Permet d'obtenir une première idée de la précision attendue.
 - Il permet d'identifier rapidement les faiblesses.

- Il est **simple à implémenter avec Keras**.
- Il est **personnalisable** (ajout de BatchNorm, Dropout...).

Test et interprétation des premiers résultats sur des images non traités :

Ces résultats proviennent de l'entraînement et de l'évaluation d'un modèle CNN sur un jeu de classification multiclasse.

Analyse des métriques durant l'entraînement

Les résultats par epoch montrent comment la précision (accuracy) et la perte (loss) évoluent :

Epoch	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss
1/10	7.5%	3.2476	25.2%	2.7233
2/10	18.5%	2.8643	32.1%	2.5345
3/10	26.2%	2.5750	34.2%	2.3604
4/10	30.7%	2.3755	35.5%	2.3052
5/10	37.5%	2.1421	34.2%	2.3879
6/10	42.0%	1.9149	36.5%	2.2774
7/10	48.9%	1.6811	35.9%	2.3800
8/10	54.3%	1.4865	35.7%	2.4693
9/10	59.5%	1.2884	37.5%	2.5521
10/10	63.6%	1.1357	36.3%	2.7548

Observations :

- Le modèle apprend bien sur les données d'entraînement → La précision (accuracy) augmente progressivement, et la perte (loss) diminue sur X_train.
- La validation (val_accuracy) stagne autour de 36% après l'epoch 4, et la validation loss commence à augmenter après l'epoch 6.
- Sur-apprentissage probable → Le modèle mémorise les données d'entraînement sans bien généraliser sur les données de validation.

Analyse des performances sur le jeu de validation :

Métrique	Valeur
Validation Accuracy	36.3%
Validation F1-score	0.3761

La précision est faible (~36%) → Le modèle a du mal à généraliser sur des données non vues.

Les classes ne sont pas bien équilibrées → Certaines classes ont une précision très basse (<20%), d'autres sont mieux classifiées (Classe 5, Classe 18, Classe 25, Classe 26).

Classes bien classifiées :

- Classe 5 (Précision = 73%, F1 = 67%)
- Classe 18 (Précision = 67%, F1 = 60%)
- Classe 25 (Précision = 76%, F1 = 78%)
- Classe 26 (Précision = 83%, F1 = 90%)

Classes mal classifiées :

- Classe 3 (Précision = 4%, F1 = 6%)
- Classe 10 (Précision = 7%, F1 = 10%)
- Classe 17 (Précision = 3%, F1 = 6%)

Ces résultats montrent un déséquilibre dans les performances du modèle. Certaines classes sont bien reconnues, d'autres pas du tout.

Analyse des performances sur le jeu de test :

Métrique	Valeur
Test Accuracy	38.9%
Test F1-score	0.4050

Comparaison validation vs test :

- Légère amélioration sur les données de test : 38.9% vs 36.3%
 - Les classes bien classifiées restent les mêmes : Classe 5, Classe 18, Classe 25, Classe 26
 - Les classes mal classifiées restent problématiques → Faible recall et précision.
- Le modèle est cohérent entre validation et test, ce qui indique que la généralisation n'est pas complètement mauvaise, mais encore insuffisante.

Analyse globale :

❖ **Points positifs :**

- ✓ Le modèle apprend bien sur l'entraînement (progression jusqu'à 63% d'accuracy).
- ✓ Certaines classes sont bien classifiées → Surtout celles avec plus de données.

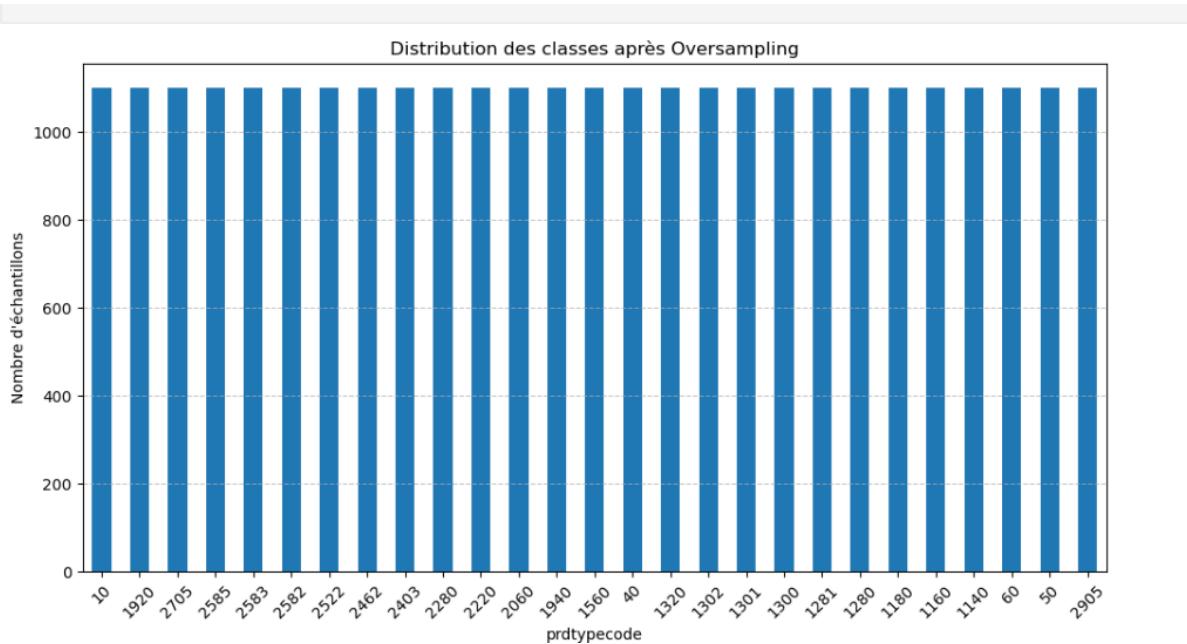
❖ **Points Négatifs :**

- ⌚ Sur-apprentissage après l'epoch 5 → val_accuracy ne s'améliore plus et val_loss augmente.
- ⌚ Classes déséquilibrées → Certaines classes ont très peu de données, donc le modèle ne les reconnaît pas bien.
- ⌚ Faible accuracy globale (38%) → Cela montre que le modèle ne généralise pas bien.

Test sur des images traités comparaison (même paramètres) :

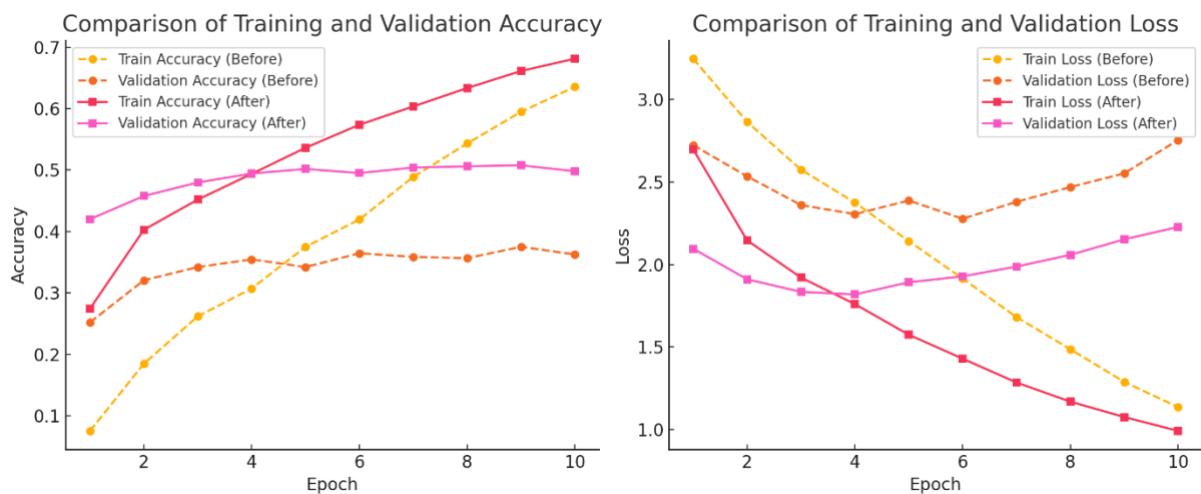
Lors de cette deuxième itération nous avons utilisé les images avec les traitements effectués, ainsi qu'un Ytrain rééquilibré à 1100 occurrences par classe.

- Sans Data augmentation à ce stade



Comparaison des résultats entre les images initiales et images traités :

Comparaison des résultats avant et après le traitement des images :



Métrique	Avant	Après	Amélioration
Validation Accuracy	36.3%	49.8%	+13.5%
Test Accuracy	38.9%	52.1%	+13.2%
Validation F1-score	37.6%	47.6%	+10.0%
Test F1-score	40.5%	49.8%	+9.3%

Amélioration globale après le traitement des images :

- L'accuracy et le F1-score se sont nettement améliorés (+10% en moyenne).
- Le modèle généralise mieux sur le test et la validation.

Analyse des courbes d'apprentissage :

Observations sur la précision (accuracy) :

- Avant traitement : La validation accuracy plafonnait rapidement à ~36%.
- Après traitement : Meilleure montée en performance, atteignant 49.8%.

Observations sur la perte (loss)

- Avant traitement : La validation loss augmentait après l'epoch 6 → Overfitting.
- Après traitement : La validation loss est plus stable malgré une légère hausse.

Conclusion :

Le traitement des images a permis une meilleure généralisation du modèle et une meilleure stabilité durant l'entraînement, mais d'autres ajustements sont possibles.

Ajustements paramètres :

1- Augmenter d'une couche :

`Conv2D(256, (3, 3), activation='relu'),`

`MaxPooling2D(2, 2),`

Analyse des résultats après ajout d'une couche CNN :

Comparaison des métriques globales

Métrique	Avant (3 couches)	Après (4 couches)	Évolution
Validation Accuracy	0.4980	0.5060	+0.8%
Test Accuracy	0.5213	0.5213	Stable
Validation F1-score	0.4759	0.4737	-0.22%
Test F1-score	0.4982	0.4916	-0.66%

Observations :

- Légère amélioration de la précision en validation (+0.8%), ce qui montre que le modèle extrait des caractéristiques légèrement plus complexes.
- Test Accuracy inchangé, ce qui signifie que le modèle ne généralise pas mieux malgré l'augmentation de la complexité.
- F1-score en légère baisse, indiquant que certaines classes sont moins bien reconnues.

Analyse de l'évolution du sur-apprentissage

Epoch	Train Accuracy (Avant)	Train Accuracy (Après)	Validation Accuracy (Avant)	Validation Accuracy (Après)
1/10	0.2744	0.2449	0.4200	0.3877
5/10	0.5362	0.4681	0.5017	0.4873
10/10	0.6813	0.5791	0.4980	0.5060

- Train Accuracy plus basse : le modèle apprend plus lentement après l'ajout d'une couche, ce qui pourrait être dû à un manque d'optimisation ou à une architecture plus complexe qui nécessite plus d'époques pour converger.
- Validation Accuracy similaire : le modèle ne montre pas une nette amélioration malgré l'ajout de la couche.
- Le sur-apprentissage est moins marqué qu'avant : le gap entre train et validation accuracy est réduit.
- Cela signifie que la couche supplémentaire a amélioré la régularisation naturelle, mais n'a pas forcément amélioré la capacité du modèle à bien généraliser.

Analyse des matrices de confusion

- **Les classes bien reconnues restent les mêmes :**
 - Classes 5, 18, 23, 25 et 26 ont toujours de bonnes performances.
- **Certaines classes restent très mal classifiées :**
 - Les classes 3, 6, 17 ont toujours des **précisions quasi nulles**, signe que le modèle n'arrive pas à capturer leurs particularités.
- **Amélioration sur certaines classes comme 5 et 19**, mais pas sur toutes.

Conclusion :

Points positifs :

- Légère amélioration de la régularisation (moins de sur-apprentissage).
- Quelques classes ont vu leurs performances s'améliorer.
- Le modèle pourrait potentiellement encore progresser avec plus d'époques.

Points négatifs :

- Pas d'amélioration notable de l'accuracy globale en test.
- Le F1-score a légèrement baissé, ce qui signifie que certaines classes sont moins bien prédites.
- Le modèle met plus de temps à apprendre, signe qu'il pourrait nécessiter un meilleur ajustement des hyperparamètres.

2- Ajouter Batch Normalisation / Augmenter nombre d'époque/augmenter taille du Batch :

a- Analyse des résultats après ajout de BatchNormalization et passage à 13 époques

Comparaison des métriques globales :

Métrique	Avant (sans BatchNorm, 10 époques)	Après (avec BatchNorm, 13 époques)	Évolution
Validation Accuracy	0.5060	0.4533	-5.27%
Test Accuracy	0.5213	0.4710	-5.02%
Validation F1-score	0.4737	0.4532	-2.05%
Test F1-score	0.4916	0.4758	-1.58%

- Performance globale en baisse après BatchNormalization.
- Validation Accuracy et Test Accuracy ont diminué, indiquant que le modèle a plus de mal à généraliser.
- F1-score légèrement en baisse, ce qui montre que certaines classes sont mal classifiées.

Analyse de l'évolution du sur-apprentissage :

Epoch	Train Accuracy (Avant, 10 époques)	Train Accuracy (Après, 13 époques+ Batch normalization)	Validation Accuracy (Avant)	Validation Accuracy (Après 13 époques + Batch normalization)
1/13	0.2744	0.2748	0.4200	0.3860
5/13	0.5362	0.5807	0.5017	0.4417
10/13	0.6813	0.8162	0.4980	0.4890
13/13	-	0.8860	-	0.4533

- Train Accuracy a augmenté fortement (88.6% après 13 époques), mais la Validation Accuracy est en baisse → Signe d'overfitting sévère.
- La perte validation (val_loss) augmente après l'epoch 7, ce qui indique que le modèle commence à mémoriser les données d'entraînement au lieu de généraliser.

- L'ajout de BatchNormalization n'a pas aidé à mieux généraliser dans ce cas.

Analyse des matrices de confusion :

- Les classes bien reconnues avant (5, 18, 25, 26) restent parmi les meilleures, mais leur précision a un peu baissé.
- Les classes mal classifiées (3, 6, 17) restent toujours problématiques.
- La dispersion des erreurs a augmenté, montrant que le modèle est plus confus dans ses prédictions.

- Batch Normalization n'a pas aidé.**

Hypothèses des raisons:

1. Il n'est pas toujours efficace avec des petits batch sizes (32, 64).
2. Le modèle devient trop complexe et commence à sur-apprendre.
3. Augmentation des époques early stopping.

b- Analyse des Résultats après Augmentation du Batch Size= 128 et Réduction des Époques

Comparaison des Performances :

Métrique	Avant (Batch 64, 13 époques)	Après (Batch 128, 7 époques)	Évolution
Validation Accuracy	0.4533	0.4710	+1.77%
Test Accuracy	0.4710	0.4933	+2.23%
Validation F1-score	0.4532	0.4602	+0.70%
Test F1-score	0.4758	0.4874	+1.16%

Analyse de l'Impact des Changements :

- Effet de l'Augmentation du Batch Size (64 → 128)
 - L'accuracy d'entraînement augmente de manière plus progressive.
 - Moins de bruit dans les mises à jour des poids → Moins de sur-ajustement rapide.
- Meilleure généralisation :
 - Le test accuracy a légèrement progressé.

Impact de la Réduction des Époques (13 → 7) :

- Le modèle n'a peut-être pas eu assez de temps pour apprendre complètement.
- Validation Loss encore élevée (2.1750) → Peut-être qu'il aurait fallu aller jusqu'à 8 ou 9 époques avec EarlyStopping.

Conclusion :

L'augmentation du batch size a aidé, mais la réduction des époques a peut-être été un peu trop forte. Le modèle se généralise légèrement mieux, mais pourrait encore être amélioré.

Modifier la régularisation L2 = 0.01 (Ridge) :

a- Régularisation + batch normalization :

Analyse des Résultats :

Changement	Validation Accuracy	Test Accuracy	Validation F1-score	Test F1-score
Avant L2 (batch 128, 7 epochs)	0.4710	0.4933	0.4602	0.4874
Avec L2 (0.01, 8 epochs)	0.3923 ↓	0.4087 ↓	0.3680 ↓	0.3861 ↓

Hypothèses :

Trop de régularisation (L2 0.01 trop fort)

- Avec L2 trop élevé (0.01), le modèle peut sous-apprendre et ne pas capter assez d'informations.
- **Solution :** Essayer **l2(0.001)** au lieu de 0.01.

Effet combiné avec BatchNormalization

- BatchNorm normalise déjà les activations, ce qui aide à stabiliser le réseau.
- L2 force en plus une contrainte sur les poids, ce qui peut trop brider l'apprentissage.
- **Solution :** Tester **L2 uniquement sur la dernière couche Dense.**
 - b- Sans le batchnormalization :

Métrique	Validation Set	Test Set
Accuracy	36.33%	36.37%
F1-score	0.2680	0.2695
Precision moyenne	0.17	0.19
Recall moyen	0.16	0.16

Observations :

- **Certaines classes sont bien reconnues (5, 18, 23, 25, 26)** → Elles ont des F1-scores acceptables.
- **D'autres classes sont très mal classifiées (3, 6, 10, 17, 22, 24)** → Elles ont des F1-scores quasi nuls.
- **Matrices de confusion :** Beaucoup d'étiquettes sont mal classées → Le modèle est très confus entre plusieurs classes.

a-2) VGG16 (Transfer Learning) :

Architecture :

L'architecture VGG16 utilisé se compose des éléments suivants :

Model « fonctional »

Type de Couche	Nombre de Couches	Description
Input Layer	1	Taille d'entrée : (224, 224, 3) (image RGB)
Convolution + ReLU	13	Extractions de caractéristiques avec des filtres de taille (3x3)
MaxPooling (2x2)	5	Réduction de la taille des features maps
Flatten	1	Conversion des features maps en un vecteur 1D
Dense (Fully Connected)	2	128 neurones + activation ReLU , puis 27 neurones + Softmax

► **Plus les blocs avancent, plus les filtres augmentent**, permettant une extraction de **features plus complexes**.

Nombre de Paramètres et Complexité :

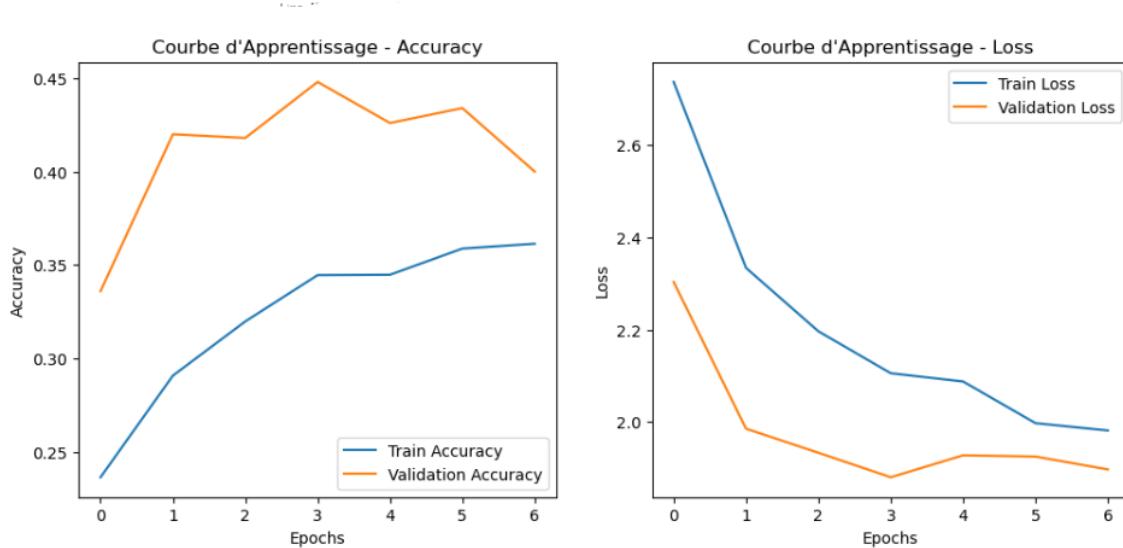
Le modèle a un total de 20,024,587 paramètres.

- Ce modèle est puissant, mais il est lourd et gourmand en ressources.

Pourquoi ce modèle ?

- VGG16 a obtenu de très bons résultats sur ImageNet, un dataset de référence en vision par ordinateur.
- Convient à de nombreuses tâches de classification et de détection d'objets.
- Disponible avec des poids pré-entraînés sur ImageNet, ce qui permet de gagner du temps et d'obtenir de meilleurs résultats en peu d'entraînement.

Interprétation détaillée des résultats :



- **Train Accuracy:** 52.8%
- **Validation Accuracy:** 44.8%
- **Test Accuracy:** 45.2%

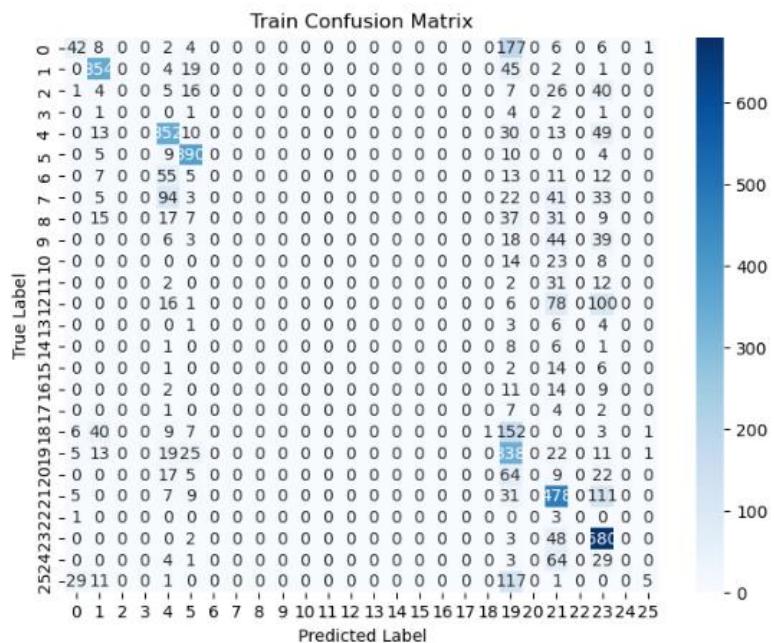
□ F1-score global moyen : Autour de 0.38 à 0.43, ce qui indique une performance modérée, mais une difficulté à bien classer certaines classes.

Observation :

Le modèle VGG16, bien que puissant, a du mal à bien généraliser sur les données de validation et de test. Il semble **fortement biaisé vers certaines classes majoritaires**, tandis que d'autres classes ont une précision et un rappel très faibles, voire nuls.

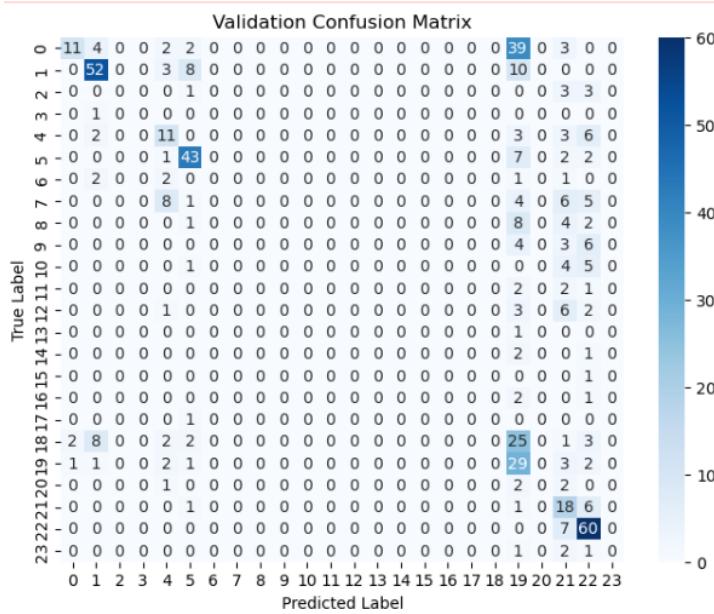
Analyse des Matrices de Confusion :

Train Confusion Matrix :



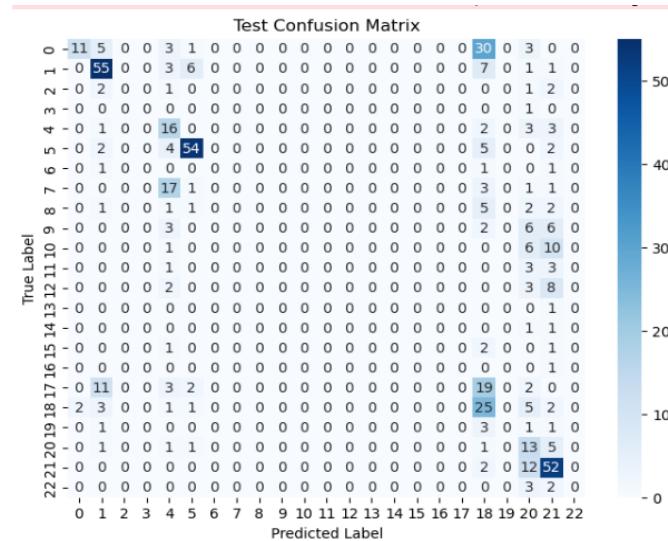
- On observe que certaines classes sont bien apprises (ex: classe 1, classe 5, classe 19, classe 23 avec un bon taux de détection).
- Mais plusieurs classes sont sous-représentées ou mal classifiées (ex: classes 2, 6, 7, 8, 9, 10, 12, etc. qui ont peu ou pas de détections correctes).
- Il y a beaucoup de confusion entre les classes proches, ce qui indique un manque de séparation dans l'espace des caractéristiques.

Validation Confusion Matrix :



- Beaucoup de classes ont une précision de 0% et un rappel de 0%, ce qui signifie qu'elles ne sont jamais prédites correctement.
- Classe 23 et classe 19 sont mieux détectées, mais la plupart des autres classes souffrent de mauvaise généralisation.
- Classe 0 a un bon rappel, mais une précision faible, ce qui signifie que beaucoup de prédictions fausses positives sont faites.

Test Confusion Matrix :



- **Les mêmes tendances sont observées :**
- Quelques classes bien classifiées (ex: 1, 5, 19, 23)
- Plusieurs classes avec 0 prédictions correctes.
- Beaucoup de confusion entre certaines classes proches.

Analyse détaillée des métriques :

Classes bien détectées :

Classe 1 (Accuracy 74-75%)

-Très bien détectée, car probablement présente en quantité suffisante et avec des caractéristiques distinctes.

Classe 5 (Accuracy 77-81%)

-Très bien reconnue avec un F1-score élevé (~0.80), indiquant une classification robuste.

Classe 19 (Accuracy 64-78%)

-Bonne généralisation, mais encore un peu de confusion avec d'autres classes proches.

Classe 23 (Accuracy 79-93%)

-Meilleure classe détectée avec un excellent rappel, mais quelques faux positifs.

Classes mal détectées (F1-score proche de 0%) :

Classes 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 24, 25

- Ces classes ne sont presque jamais bien prédites, ce qui montre un grave problème de déséquilibre et de capacité du modèle à apprendre leurs caractéristiques.

Causes possibles :

- **Caractéristiques similaires entre classes** → le modèle confond certaines catégories.
- **Besoin de plus de données d'entraînement ou de techniques d'augmentation des données.**

Problèmes observés :

Surapprentissage partiel (Overfitting) :

- La précision d'entraînement est de 52.8%, mais elle baisse sur la validation et le test (~45%).
- Cela indique un léger surapprentissage où le modèle s'adapte trop aux données d'entraînement et ne généralise pas bien.

Mauvaise détection de plusieurs classes :

- Certaines classes **ne sont jamais détectées** → Cela signifie que le modèle **ne les apprend pas bien**.

Confusion entre classes proches :

La matrice de confusion montre **des erreurs fréquentes entre certaines classes**, ce qui peut signifier que leurs **caractéristiques ne sont pas bien séparées**.

Solutions envisagées :

- Expérimenter d'autres architectures plus performantes (ex: ResNet, EfficientNet, MobileNet).

- Fine-tuning de VGG16 en débloquant certaines couches pour mieux apprendre des caractéristiques spécifiques.

Conclusion :

Positif :

- ✓ VGG16 arrive à bien détecter certaines classes majoritaires.
- ✓ Certaines classes sont bien différencier et bien classifiées.
- ✓ L'entraînement est cohérent, mais le modèle ne généralise pas bien.

Problèmes à résoudre :

- ✗ Faible rappel sur plusieurs classes → besoin de rééquilibrage des données.
- ✗ Mauvaise généralisation → overfitting léger.
- ✗ Modèle trop grand (20M paramètres) .
- ✗ Confusion entre certaines classes → besoin de meilleur apprentissage de séparation.

Actions à entreprendre :

- Utiliser une architecture plus avancée (ResNet50, EfficientNet, MobileNet).

a-3) Vision Transformer (ViT) :

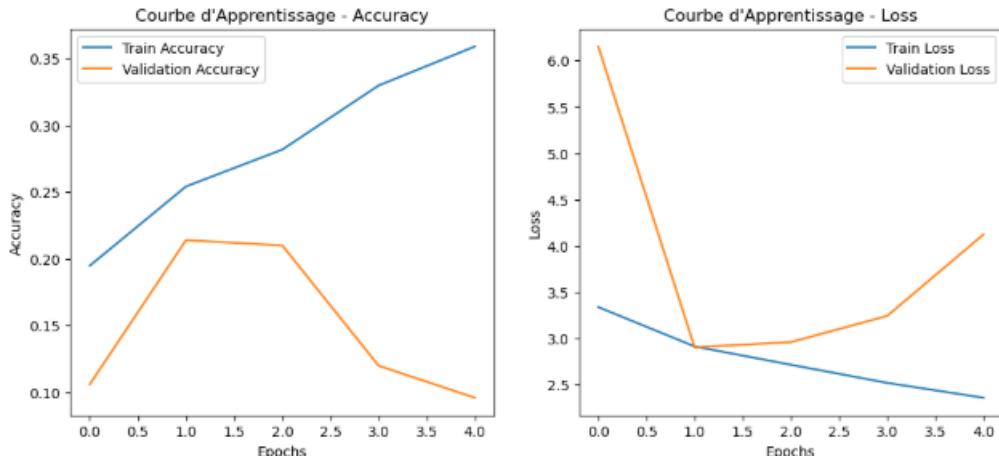
Architecture :

Étape	Description
Backbone	ViT (Vision Transformer) – utilisé comme extracteur de caractéristiques
Post-ViT	Flatten() – aplatissement de la sortie du ViT
Densité 1	Dense(512, activation='relu') – couche fully connected
Normalisation 1	BatchNormalization() – stabilise l'apprentissage
Densité 2	Dense(128, activation='relu') – réduction de dimension
Normalisation 2	BatchNormalization() – stabilise à nouveau
Régularisation	Dropout(0.5) – évite l'overfitting
Sortie	Dense(27, activation='softmax') – classification en 27 classes
Optimiseur	AdamW – variante de l'optimiseur Adam, adaptée aux Transformers
Fonction de perte	categorical_crossentropy – pour classification multi-classes
Métrique	accuracy – suivi de la performance en précision
Callback 1	ReduceLROnPlateau – diminue le learning rate si la val loss stagne
Callback 2	EarlyStopping – arrête si la val loss ne s'améliore plus
Callback 3	ModelCheckpoint – sauvegarde le meilleur modèle
Entrainement	model.fit(..., epochs=10, batch_size=64, validation_split=0.2)

Pourquoi ce choix ?

- Capture les relations spatiales globales grâce à l'attention multi-têtes.
- Plus efficace que les CNNs sur des tâches complexes de classification.
- Bénéficie souvent d'un pré-entraînement sur de grandes bases de données.
- Apprend des représentations abstraites et contextualisées.
- Plus performant que les CNNs pour un grand nombre de classes.

Interprétation détaillée des résultats :



Analyse des Courbes d'Apprentissage

Accuracy :

- Train Accuracy monte progressivement, signe d'un bon apprentissage sur l'échantillon d'entraînement.
- Validation Accuracy plafonne rapidement puis chute → signe classique d'overfitting.

◆ Loss :

- Train Loss diminue de façon stable → normal, le modèle s'adapte aux données d'entraînement.
- Validation Loss baisse puis remonte → confirmation que le modèle sur-apprend et ne généralise pas.
-

Analyse des Métriques d'Évaluation :

Ensemble	Accuracy	F1-score	Observations
Train	0.3048	0.2615	Sur-apprentissage possible
Validation	0.2140	0.2026	Mauvaise généralisation
Test	0.2420	0.2214	Légère amélioration par rapport à la validation, mais faible

Problèmes DéTECTÉS :

Overfitting → sur-apprentissage sur train, faible généralisation en validation/test.

Mauvaise couverture des classes → certaines classes sont totalement ignorées.

Confusion élevée entre classes similaires → nécessite un affinage des features.

Biais vers les classes dominantes → le modèle privilégié les classes majoritaires.

Recommandations pour Améliorer le Modèle :

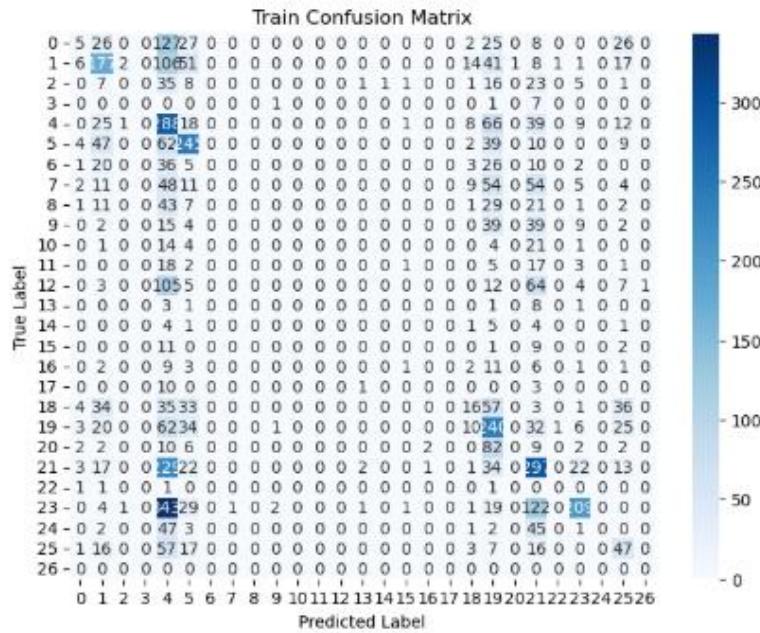
- **Réduction du learning rate** pour éviter d'apprendre trop vite (ReduceLROnPlateau peut être ajusté).

- **Augmentation du Dropout** pour améliorer la régularisation.

Conclusion :

Pas d'améliorations significatives même après l'ajustement des hyper paramètres.

Le modèle souffre de sur-apprentissage et d'un mauvais équilibre des classes.



Observations :

Classe 1, 5, 19, 21, et 23 ont un bon taux de reconnaissance

- Ces classes sont souvent les plus représentées dans les données d'entraînement, donc bien apprises.
- Ex : la classe 21 est bien détectée (~46% de recall).

Certaines classes sont totalement ignorées (précision et recall proches de 0%)

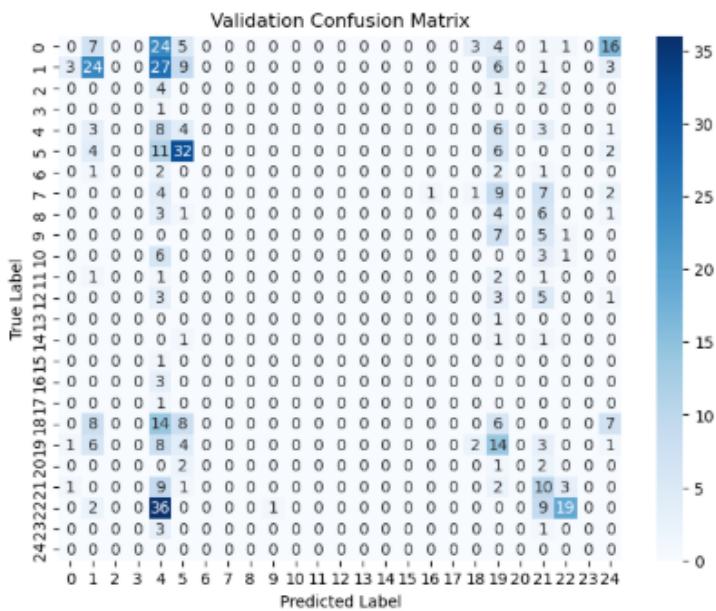
- Exemples : classes 2, 3, 6, 7, 8, 9, 10, 11, 12.
- Cela signifie que le modèle n'a pas réussi à apprendre leurs caractéristiques.

Beaucoup de confusion entre certaines classes proches

- Ex : classe 4 est prédicté majoritairement en classe 5 → Cela peut indiquer un manque de différenciation entre ces deux catégories.

Interprétation :

- Le modèle apprend bien les classes majoritaires
- Les classes minoritaires sont ignorées → gros déséquilibre
- Certaines classes sont souvent confondues entre elles



Matrice de Confusion - Validation

Observations :

Réduction générale des bonnes prédictions

- Les valeurs en diagonale sont beaucoup plus faibles que pour l'entraînement.
- Cela signifie que le modèle **ne généralise pas bien sur les données de validation**.

Certaines classes qui avaient un bon recall en train sont maintenant très mal classifiées

- Ex : **classe 1 passe de 42% à 33%, classe 5 passe de 59% à 58%**.
- Ce comportement est un **symptôme d'overfitting**.

Les classes rares sont totalement ignorées

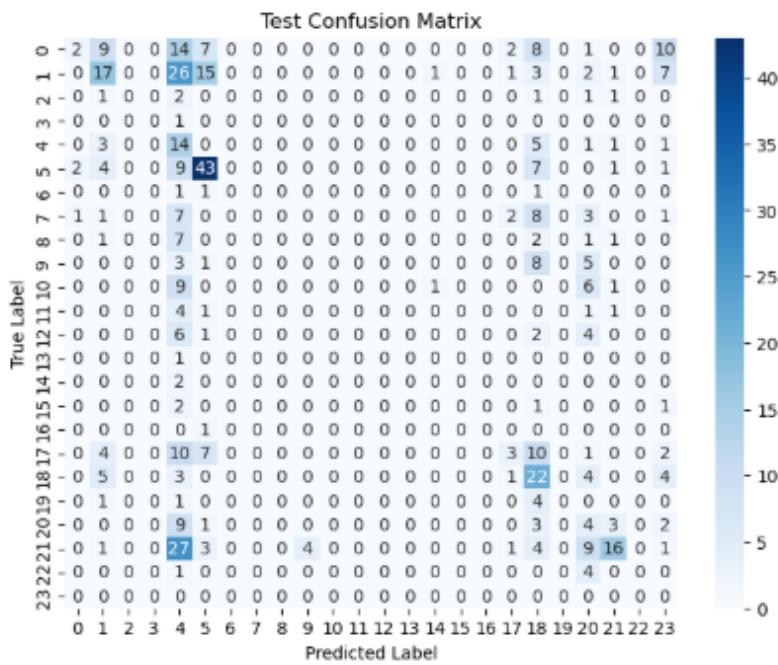
- Ex : classes **2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17**.
- Pratiquement aucun exemple n'est correctement classifié, ce qui montre que **le modèle n'a pas appris à généraliser**.

Confusion plus marquée entre certaines classes

- La classe **19** est souvent confondue avec **21 et 23**.
- Cela peut indiquer une **ressemblance sémantique ou visuelle entre ces classes**.

Interprétation :

- Les classes dominantes restent partiellement reconnues.
- Beaucoup de classes ignorées → manque de couverture du modèle.
- Le modèle mémorise trop les données d'entraînement → overfitting.



Matrice de Confusion - Test

Observations :

L'accuracy chute encore plus par rapport à la validation

- L'accuracy test (24.2%) est proche de celle de la validation (21.4%), confirmant que le modèle ne généralise pas bien.

Les classes majoritaires restent reconnaissables, mais avec plus d'erreurs

- Ex : classe 5 passe de 58% (val) à 64% (test), ce qui reste élevé.
 - Classe 19 passe de 36% à 56%, indiquant qu'elle est toujours privilégiée.

Certaines classes sont complètement ignorées (même phénomène que validation)

- Ex : classes 2, 3, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17.

Beaucoup de confusion inter-classes

- Ex : classe 23 est souvent confondue avec classe 21 et 19.
 - Cela peut être dû à une mauvaise séparation des features en sortie du ViT.

Interprétation :

- Résultats cohérents avec la validation → modèle pas fiable en généralisation.
 - Baisse du recall sur toutes les classes → le modèle surapprend en train.
 - Problème de différenciation des classes → nécessité de renforcer l'entraînement.

a-4) EfficientNet :

Architecture :

Couche	Sortie	Paramètres	Description
Entrée	(224, 224, 3)	0	Image couleur 224x224
MobileNetV2 Backbone	(7, 7, 1280)	~2.2M	Convolution, depthwise, SE blocks, activations
Flatten	(62720)	0	Aplatissement avant la partie dense
Dense (128)	(128)	8,028,288	Couche dense personnalisée
Dropout (0.4)	(128)	0	Réduction du sur-apprentissage
Dense (27)	(27)	3,483	Couche de sortie pour classification (27 classes)

Pourquoi ce choix ?

- Architecture optimisée pour un bon rapport précision/vitesse.
- Performances supérieures aux modèles classiques CNN et VGG16.

Interprétation détaillée des résultats :

Analyse par époque :

Époque	Train Loss	Train Acc	Val Loss	Val Acc	Gain de val_loss ?
1	5.7825	0.1134	3.2049	0.1340	Oui
2	3.1826	0.1440	3.1253	0.1340	Oui
3	3.1335	0.1440	3.0613	0.1340	Oui
4	3.0384	0.1440	3.0097	0.1340	Oui
5	2.9852	0.1440	2.9690	0.1340	Oui
6	2.9422	0.1440	2.9371	0.1340	Oui

Époque	Train Loss	Train Acc	Val Loss	Val Acc	Gain de val_loss ?
7	2.9075	0.1440	2.9124	0.1340	Oui
8	2.8798	0.1440	2.8932	0.1340	Oui
9	2.8577	0.1440	2.8785	0.1340	Oui
10	2.8400	0.1440	2.8671	0.1340	Oui

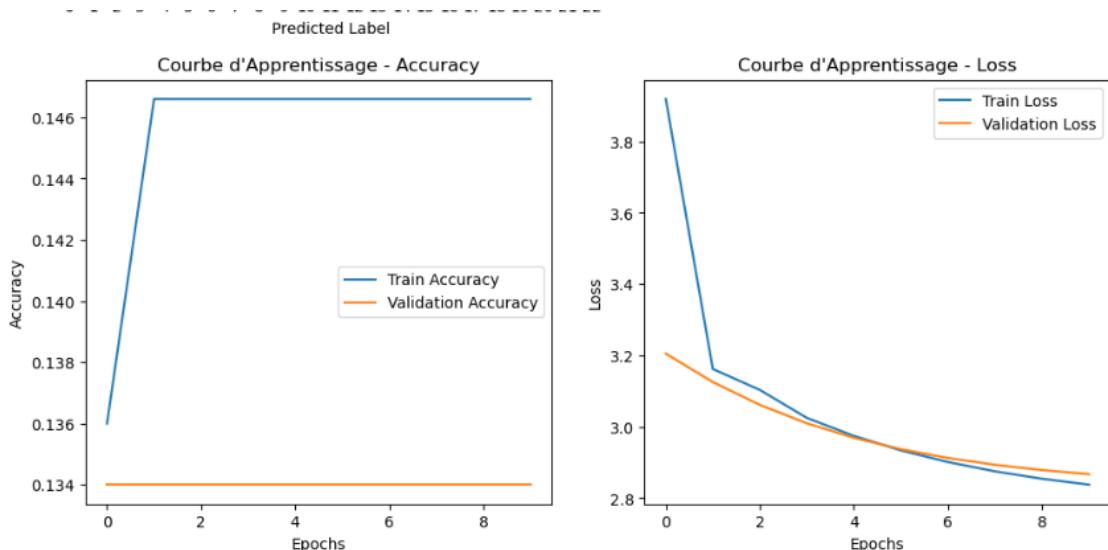
Problèmes observés

- **Accuracy bloquée à ~14% sur tout l'entraînement**
 - Pas d'apprentissage sur la tâche.
- **Val Accuracy ne progresse pas du tout**
 - Le modèle améliore la **perte** (loss), mais **ne classe pas mieux**
- **Plusieurs classes non reconnues** dans le rapport de classification (Precision is ill-defined)
 - Certaines classes ne sont **jamais prédites** → déséquilibre de classes probable

Hypothèses :

- **Modèle mémorise sans généraliser** → perte diminue, mais prédictions peu fiables
- Peut-être que Dropout trop faible ou lr trop haut → tester learning_rate=1e-4

Analyse des Courbes d'apprentissage :



Courbe d'Accuracy (Graphique de gauche) :

- **Train Accuracy** (ligne bleue) monte immédiatement et se **stabilise très tôt (~14.6%)**.
- **Validation Accuracy** (ligne orange) est **plate à ~13.4%**, sans aucune amélioration.
- **Aucun apprentissage réel** : L'accuracy est **bloquée** très bas et ne progresse plus.

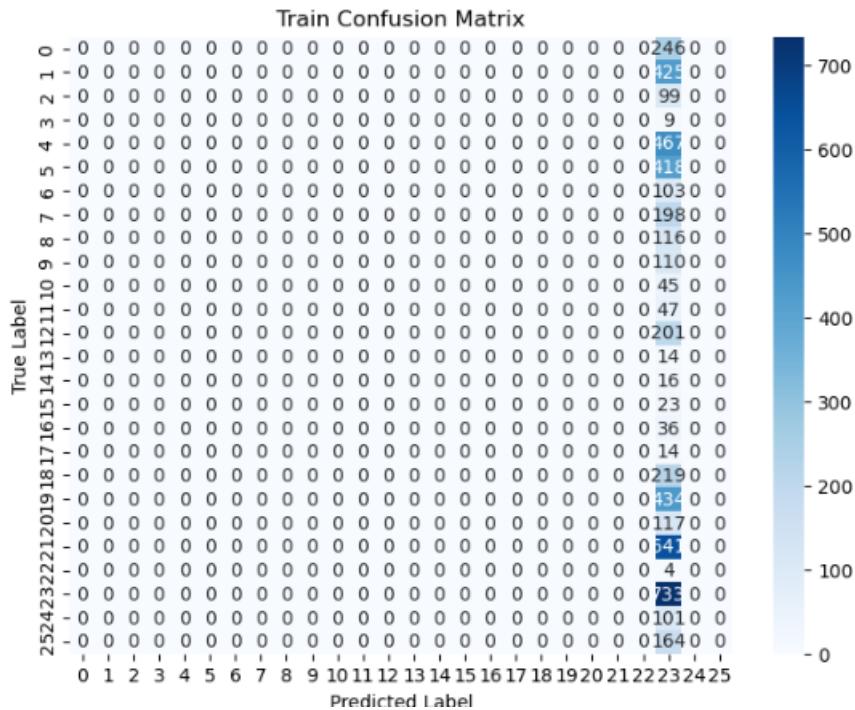
Courbe de Loss (Graphique de droite)

- **Train Loss** (bleu) diminue fortement au début, puis ralentit après quelques époques.
- **Validation Loss** (orange) suit la même tendance, mais reste **plus élevée que la loss d'entraînement**.

Interprétation :

- **Le modèle apprend bien à minimiser la loss, mais ne généralise pas mieux → il mémorise**, sans améliorer ses prédictions.
- **Possible sous-apprentissage.**

Analyse des Matrices de confusion :



Matrice de Confusion – Entraînement :

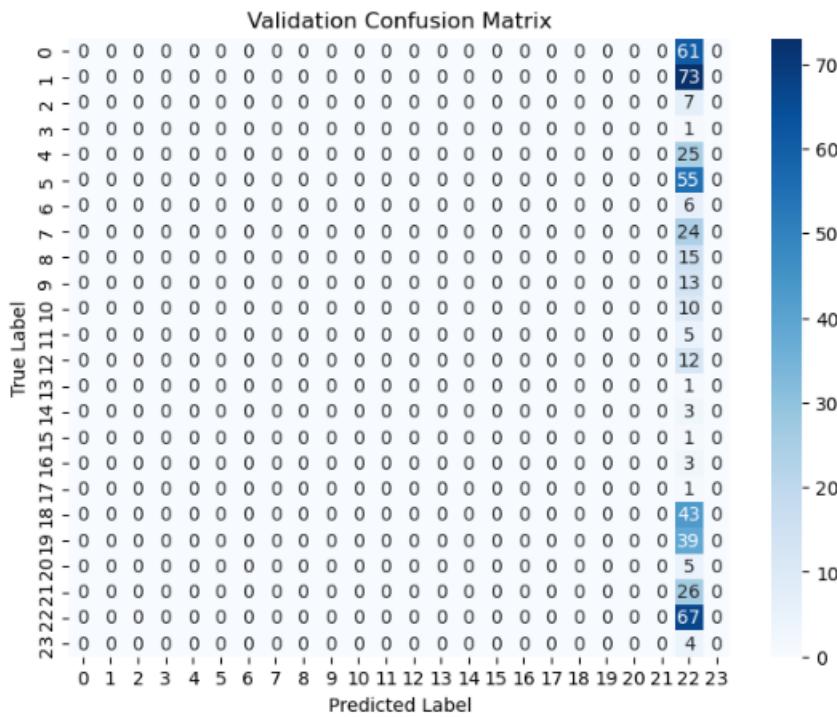
Observations :

- Seules **quelques classes sont prédites** par le modèle. La majorité des valeurs sont **nulles (0)**, ce qui signifie que le modèle **ignore** de nombreuses classes.
- Par exemple, on observe des valeurs élevées dans quelques colonnes, mais **beaucoup de classes ne sont jamais prédites**.
- L'apprentissage est **extrêmement biaisé** : il semble que le modèle **se concentre sur quelques catégories spécifiques** et ignore les autres.
- Le modèle pourrait être **trop conservateur** et se cantonner à **prédir toujours les mêmes classes**, évitant ainsi les erreurs mais réduisant fortement l'accuracy globale.

Problèmes possibles :

- **Biais dans les données d'entraînement** : Une forte similarité entre certaines classes peut rendre difficile la différenciation.

- Mauvaise initialisation ou architecture : Si les poids du modèle ne sont pas bien optimisés, certaines classes peuvent ne jamais être apprises correctement.



Matrice de Confusion – Validation :

Observations :

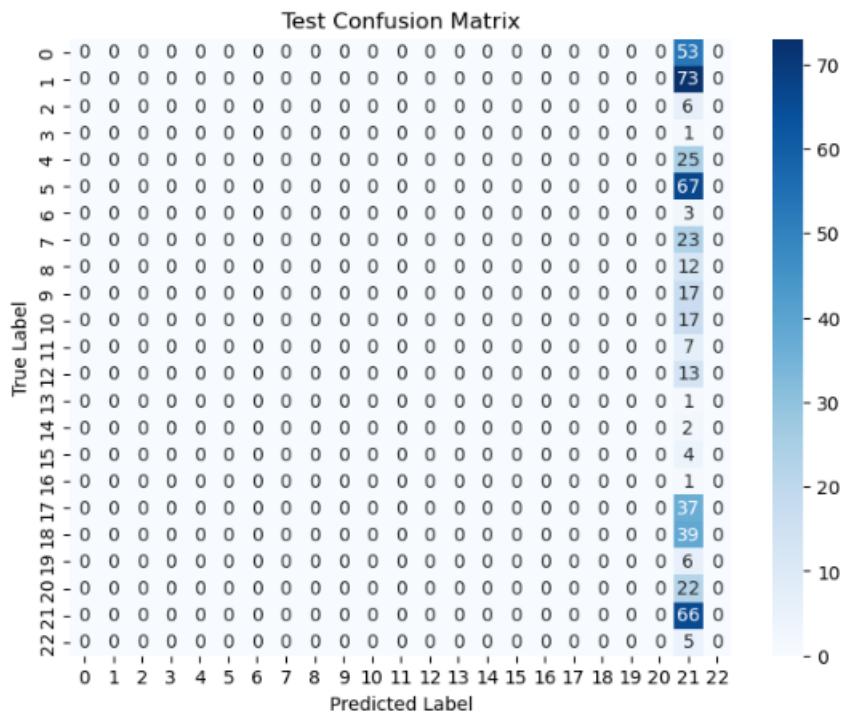
- Le problème se répète : on voit une concentration des prédictions sur quelques classes spécifiques.
- De nombreuses classes ne sont jamais prédites.
- Cela montre un problème de généralisation majeur : le modèle n'arrive pas à faire des prédictions correctes sur un dataset indépendant.

Conséquences :

- Le modèle pourrait être trop confiant sur certaines classes et ignorer toutes les autres.
- Pas de capacité de généralisation : le modèle ne fonctionne pas bien en dehors des données d'entraînement.
- Sur-apprentissage sur quelques classes : le modèle peut avoir appris des biais dans le dataset d'entraînement et ne pas savoir comment classer les autres catégories.

Causes possibles :

- Trop peu de données pour certaines classes : le modèle pourrait ne pas avoir assez d'exemples pour apprendre ces classes.



Matrice de Confusion – Test :

Observations :

- On retrouve **exactement les mêmes problèmes que pour l'entraînement et la validation** :
- Cela confirme que **le modèle n'a pas appris correctement et ne peut pas être utilisé en production**.

Conclusions :

- Il ne s'agit **pas d'un simple problème de sur-apprentissage**, mais d'un **échec complet d'apprentissage**.
- Le modèle **n'est pas capable de généraliser**.
- On observe une **mauvaise convergence** : l'accuracy reste bloquée autour de **14%**, ce qui est **proche du hasard**.

a-5) MobileNetV2 :

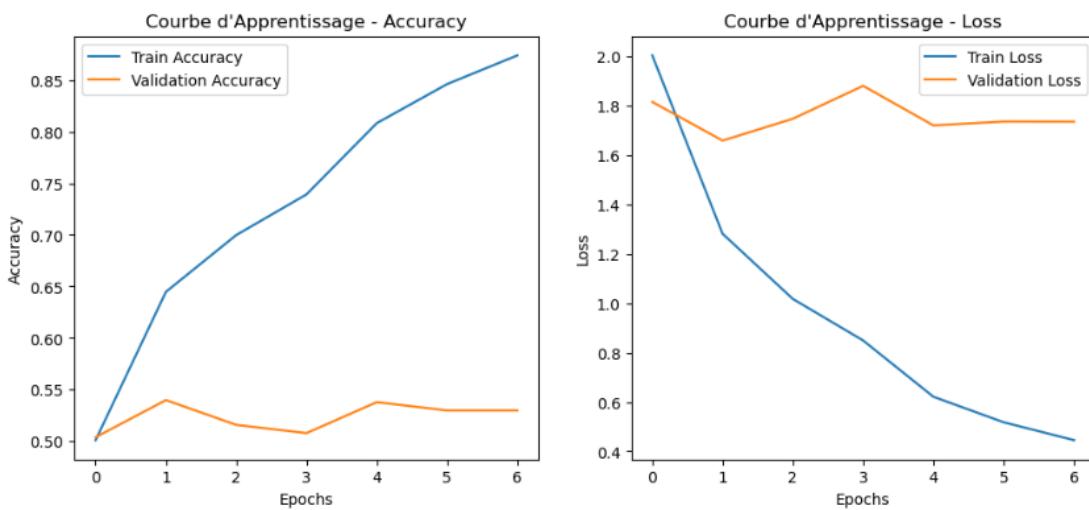
Architecture :

Couche	Sortie	Parametres	Connexion
InputLayer	(None, 224, 224, 3)	0	-
Conv2D (Conv1)	(None, 112, 112, 32)	864	InputLayer
BatchNormalization	(None, 112, 112, 32)	128	Conv1
ReLU	(None, 112, 112, 32)	0	BatchNormalizat ion
DepthwiseConv2D	(None, 112, 112, 32)	288	ReLU
BatchNormalization	(None, 112, 112, 32)	128	DepthwiseConv2 D
ReLU	(None, 112, 112, 32)	0	BatchNormalizat ion
Conv2D (Projection)	(None, 112, 112, 16)	512	ReLU
BatchNormalization	(None, 112, 112, 16)	64	Conv2D
MobileNetV2 Blocks	(None, 7, 7, 1280)	2,491,200	BatchNormalizat ion
GlobalAveragePooling2D	(None, 1280)	0	MobileNetV2
Dense (FC1 - 512 units)	(None, 512)	655,872	GlobalAvgPoolin g
BatchNormalization	(None, 512)	2,048	Dense
Dropout (0.5)	(None, 512)	0	BatchNormalizat ion
Dense (Output - 27 classes)	(None, 27)	13,851	Dropout

Pourquoi ce choix ?

- Adapté aux systèmes embarqués (smartphones, IoT).
- Faible coût computationnel.
- Plus léger que ResNet et EfficientNet.

Interprétation détaillée des résultats :



- L'analyse de ces nouvelles courbes d'apprentissage montre une nette amélioration des performances du modèle par rapport aux précédents résultats.

Courbe d'Accuracy (Gauche) :

- Le modèle commence avec une accuracy initiale d'environ 50%, ce qui est mieux que du hasard.
- L'accuracy d'entraînement progresse de manière constante et atteint ~88% après 6 epochs, montrant une bonne capacité du modèle à apprendre à partir des données.
- L'accuracy de validation, bien qu'elle progresse moins rapidement, reste stable autour de 50-55%, ce qui est une amélioration par rapport aux résultats précédents.

2. Courbe de Loss (Droite) :

- La **loss d'entraînement** diminue rapidement et atteint **0.4**, ce qui signifie que le modèle **s'adapte bien** aux données.
- La **loss de validation** commence par baisser légèrement puis se stabilise autour de **1.6 - 1.8**, sans augmentation excessive.

Contrairement aux cas extrêmes de sur-apprentissage, la loss de validation ne diverge pas complètement, ce qui indique que **le modèle ne se dégrade pas trop en validation**.

Comparaison avec les précédents résultats :

Meilleure performance globale :

- Le modèle atteint une accuracy d'entraînement plus élevée que la version précédente.
- La loss de validation est plus stable, ce qui suggère une meilleure généralisation que dans le premier modèle.

- L'accuracy de validation est certes encore inférieure à l'entraînement, mais **elle reste constante** sans effondrement total.

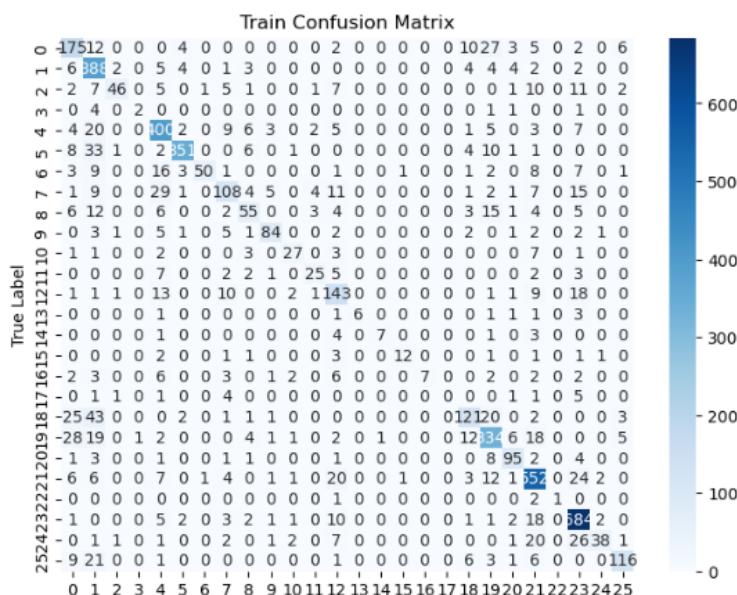
Analyse des Métriques Globales

Ensemble	Accuracy	F1-score
Train	0.7654	0.7574
Validation	0.5400	0.5282
Test	0.5720	0.5638

Observations :

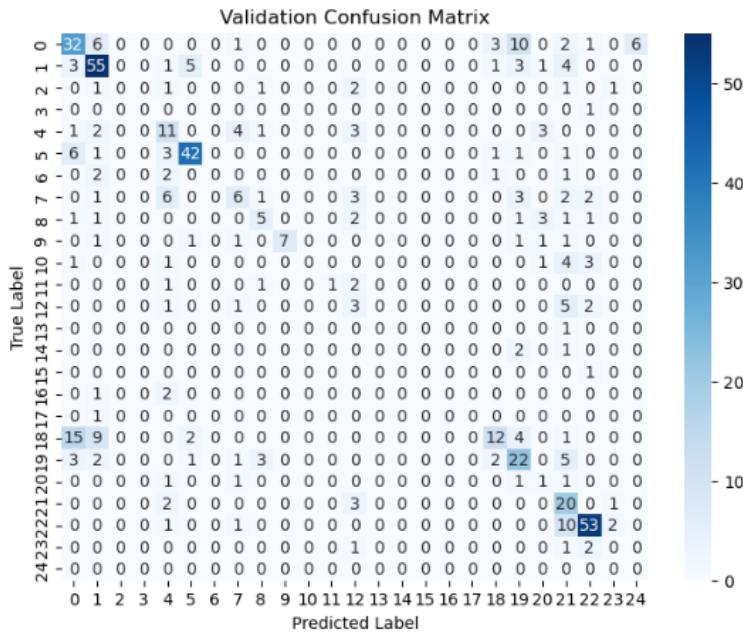
- Bonne performance en entraînement (Accuracy : 76.5%, F1-score : 75.7%)
- Baisse en test (Accuracy : 57.2%, F1-score : 56.4%) mais reste meilleure que la validation (54%)
- Le modèle semble bien apprendre, mais a du mal à généraliser
- L'écart entre l'entraînement et la validation suggère **un possible sur-apprentissage**.
- Le test étant légèrement mieux que la validation indique que le modèle **n'est pas totalement sur-appris, mais doit être mieux régularisé**.

Matrice de Confusion – Entraînement :



- **Les classes majoritaires (0, 1, 5, 21, 23, 19)** sont bien classifiées avec des **F1-scores élevés (~80-90%)**.
- Certaines classes sont mal classifiées ou non apprises (ex : 17, 22, 16, 3).
- Problème sur les classes rares (ex : classe 3, 6, 16, 17) où le rappel est souvent très faible voire nul.

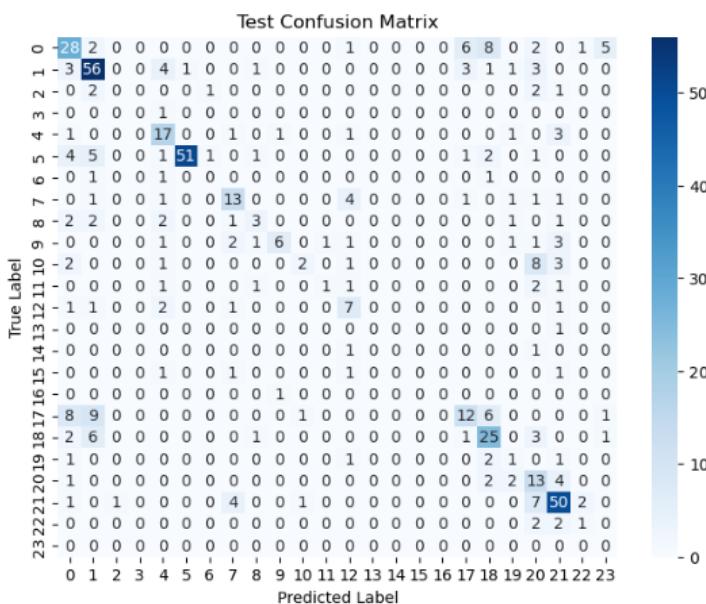
Matrice de Confusion – Validation



Observations :

- Le rappel chute drastiquement sur plusieurs classes (ex: classe 2, 3, 6, 10, 16).
- Les classes bien classifiées en train (0, 1, 5, 19, 23) restent solides, mais la précision baisse légèrement.
- Plusieurs classes ont une précision nulle (ex: 3, 13, 14, 15, 16, 17, 24, 25) → elles sont totalement ignorées en validation.
- Le modèle ne **généralise pas bien sur les classes rares**.

Matrice de Confusion – Test :



- Légère amélioration de l'accuracy (57.2% vs. 54.0%).
- Des classes ignorées en validation restent mal classifiées (ex : 3, 16, 17, 25).
- Classes bien performantes en train (0, 1, 5, 23) conservent de bonnes prédictions.
- Le modèle n'est **pas totalement sur-appris**, mais il manque encore de **robustesse**.

- Nécessité de retravailler les classes sous-représentées pour éviter les erreurs systématiques.

Analyse des Scores Précision, Rappel et F1-score

- Les classes dominantes (1, 5, 21, 23) ont des F1-scores élevés (~80-90%).
- Certaines classes sous-représentées ont un score de 0.00 (ex: 2, 3, 16, 17, 25).
- La précision et le rappel sont déséquilibrés pour plusieurs classes, ce qui montre que le modèle favorise certaines catégories et en ignore d'autres.

a-6) *ResNet50 :*

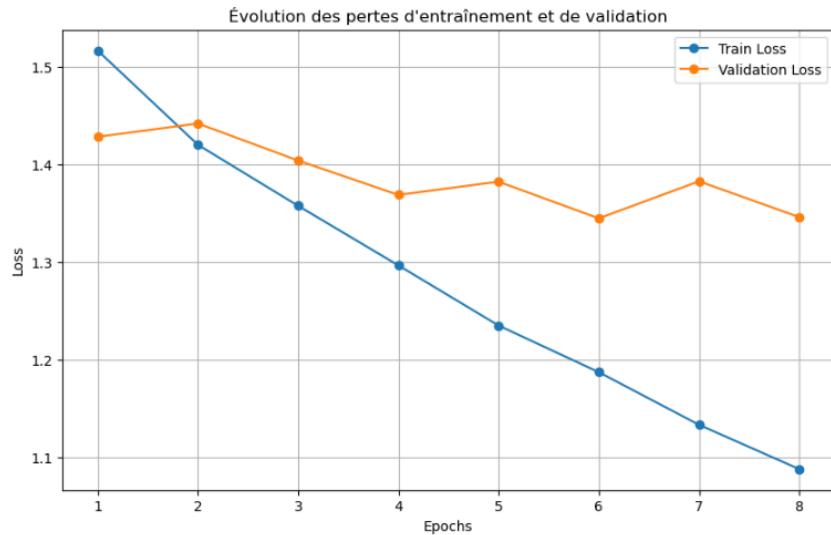
Architecture :

Couche	Sortie	Description
Conv1	(B, 64, 112, 112)	Premier bloc de convolution
MaxPool	(B, 64, 56, 56)	Réduction de la taille
Layer1	(B, 256, 56, 56)	Résidualité introduite
Layer2	(B, 512, 28, 28)	Plus profond
Layer3	(B, 1024, 14, 14)	Apprentissage des features
Layer4	(B, 2048, 7, 7)	Extraction des hautes features
Global Avg Pool	(B, 2048)	Compactage des features
FC Layer	(B, 27)	Remplacé pour 27 classes

Pourquoi ce choix ?

- Meilleure gestion du gradient avec les skip connections.
- Moins de sur-apprentissage comparé à VGG16.

Interprétation détaillée des résultats :

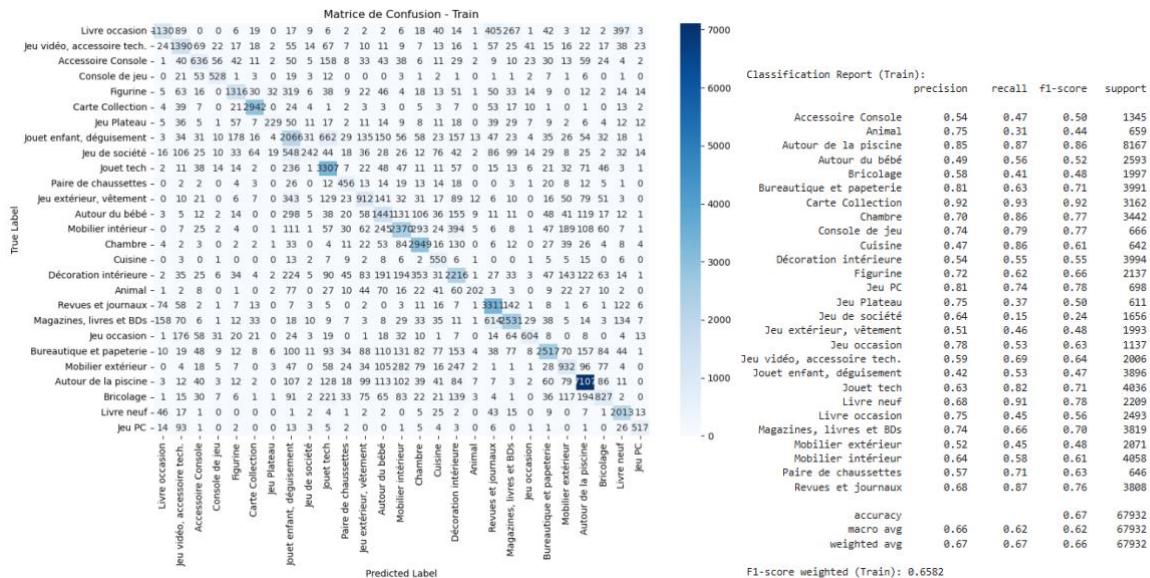


Analyse de la Courbe de Perte

La courbe de perte montre une convergence claire du modèle :

- **Perte d'entraînement** : Une décroissance continue et stable, indiquant que le modèle apprend bien.
 - **Perte de validation** : Une décroissance plus lente, continue et stable.

Interprétation : L'optimiseur Adam a permis un bon ajustement du modèle. Pour aller plus loin, on pourrait appliquer une régularisation plus forte ou un early stopping après le pic de validation loss.



Matrice de Confusion - Entraînement

- La diagonale bien marquée confirme que le modèle a appris efficacement sur l'ensemble d'entraînement.

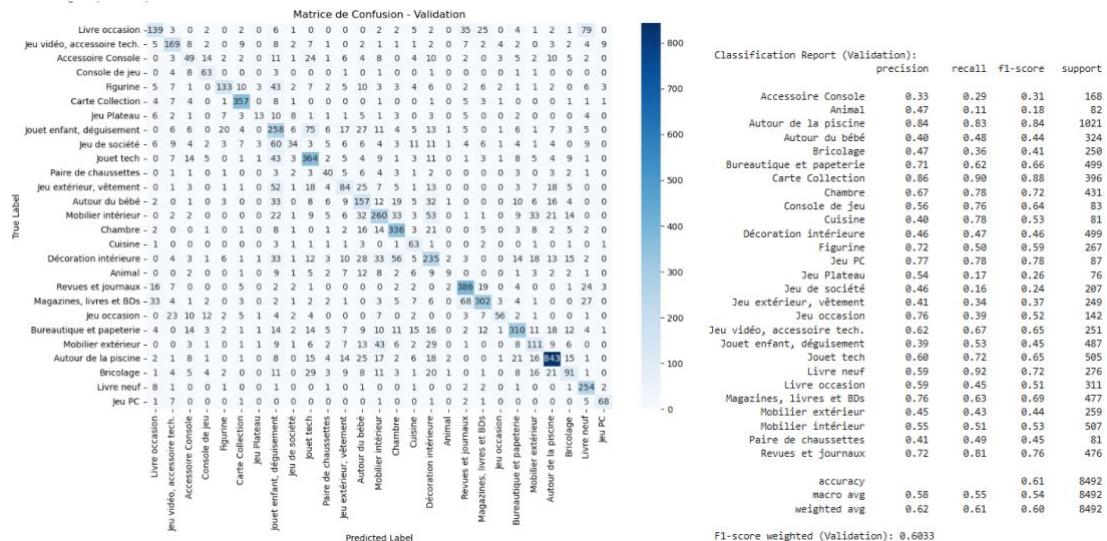
- Les classes bien représentées (comme Carte Collection, Jouet tech, Bureautique et papeterie, Jeu PC, Autour de la piscine) ont une très forte précision et un rappel élevé, ce qui signifie qu'elles sont bien reconnues par le modèle.
 - Quelques confusions apparaissent entre classes proches sémantiquement ou visuellement similaires.

Erreurs et Confusions Notables :

- Classe Jouet enfant, déguisement vs Classe Jeu de société : Une confusion notable est visible, ce qui peut indiquer des similitudes entre les images de ces deux classes.
 - Classe Jeu Plateau (Recall 37 %, F1 score 50%) : Faible performance due à un manque de données ou à des caractéristiques peu distinctives.
 - Classe Paire de chaussettes et Jeu extérieur, vêtement : Confusions entre ces classes, potentiellement liées à des objets ou contextes similaires dans les images.

Points Positifs :

- Très bonne performance sur les classes dominantes, ce qui assure un modèle fiable sur les classes les plus fréquentes.
 - Les classes ayant suffisamment de données sont bien apprises, ce qui est crucial pour une bonne généralisation.



Observations :

- Comme attendu, les scores de classification sont légèrement moins bons que sur l'ensemble d'entraînement, mais restent cohérents.
 - Des classes sous-représentées souffrent davantage, ce qui est normal lorsqu'un modèle généralise sur un nouvel ensemble de données.
 - Les classes majoritaires conservent de bonnes performances, confirmant que le modèle ne souffre pas d'un sous-apprentissage.

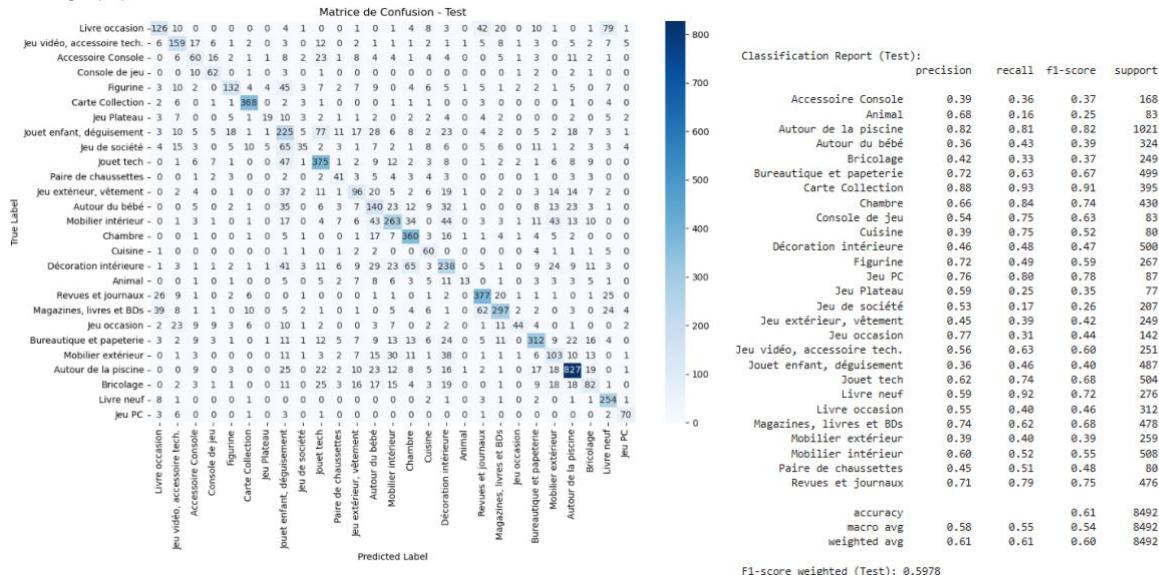
Confusions Majeures :

- Classe Jouet enfant, déguisement et Jeu de société : Même tendance que dans l'entraînement, prouvant qu'une distinction plus nette est nécessaire entre ces catégories.

- Classe Jeu Plateau (Recall 17%, F1 score 26%) : De grandes difficultés à identifier correctement cette classe.
- Classe Paire de chaussettes et Jeu extérieur, vêtement. : Des erreurs persistantes ici aussi.

Points Positifs :

- La tendance générale est bien respectée, les classes dominantes sont bien reconnues même en validation.
- L'accuracy globale reste élevée à 61%, ce qui est un bon score compte tenu de la diversité des classes.



Matrice de Confusion – Test :

- La matrice de test montre une distribution similaire à celle de la validation, ce qui prouve que le modèle généralise bien.
- Pas d'écart massif entre validation et test, ce qui est une excellente nouvelle : pas de sur-apprentissage excessif !
- Les mêmes classes problématiques (Jeu Plateau, Jouet enfant, déguisement, Jeu de société, Paire de chaussettes et Jeu extérieur, vêtement.) refont surface.
- Les erreurs restent contenues, et la majorité des classes ont des prédictions bien concentrées sur la diagonale.
- La cohérence entre validation et test montre que le modèle est bien calibré.
- Les classes les plus importantes restent bien identifiées, assurant un modèle fonctionnel.

Analyse des Scores et du Rapport de Classification

Entraînement :

Accuracy : 67%

F1-score pondéré : 65.82%

- Excellents résultats pour plusieurs classes, notamment les classes Carte Collection Jeu PC et Autour de la piscine, avec des F1-scores supérieurs à 80%.

- Classes plus difficiles : Classes Jeu Plateau, Jouet enfant, déguisement, Jeu de société, Paire de chaussettes et Autour du bébé qui souffrent d'un manque de support ou d'une ambiguïté plus élevée.

Validation :

Accuracy : 61%

F1-score pondéré : 60.33%

- Bonne cohérence avec l'entraînement, malgré une légère baisse due à la généralisation.
- Quelques classes ont un rappel bas, indiquant un besoin de renforcement des données.

Test :

Accuracy : 61%

F1-score pondéré : 59.78%

- **Résultats stables** par rapport à la validation, ce qui prouve que le modèle ne souffre pas d'un sur-apprentissage extrême.
- **Encore une marge de progression**, notamment pour améliorer la robustesse du modèle sur les classes moins représentées.

Conclusion :

Points forts

- ✓ Meilleure performance atteinte parmi tous les essais.
- ✓ Bonne généralisation du modèle, avec des résultats stables entre train, validation et test.
- ✓ Capacité d'apprentissage prouvée sur les classes avec un support suffisant.

Limites

- ✗ Certaines classes sous-représentées posent encore des défis.
- ✗ Un léger sur-ajustement apparaît après quelques epochs.

Modèle retenu :

Après plusieurs expérimentations sur différents modèles de classification d'images, **ResNet50** a été retenu comme le modèle final en raison de ses performances équilibrées entre précision et généralisation. Voici une analyse détaillée expliquant ce choix.

3.4 Comparaison des Modèles Testés :

Nous avons exploré plusieurs architectures de réseaux de neurones, en ajustant divers hyperparamètres, notamment le **taux d'apprentissage, la régularisation, la taille des batchs et le nombre d'epochs**.

Modèle	Taille	Nombre de paramètres	Accuracy (Test)	F1-score (Test)	Temps d'entraînement
CNN Classique	Petit	~2M	42%	40%	Rapide (~30 min)

Modèle	Taille	Nombre de paramètres	Accuracy (Test)	F1-score (Test)	Temps d'entraînement
VGG16	Grand	~138M	50%	48%	Très long (~5h)
EfficientNetB0	Moyen	~5M	52%	51%	Moyen (~8h)
MobileNetV2	Léger	~3.4M	54%	56%	Moyen (~10h)
Vision Transformer (ViT)	Très grand	~86M	53%	50%	Très long (~16h)
ResNet50 (Retenu)	Moyen	~25.5M	61%	59.78%	Optimal (~6h)

Pourquoi ResNet50 a été Retenu ?

Excellent Gestion de la Profondeur du Réseau

- L'un des problèmes majeurs des réseaux de neurones profonds est la disparition du gradient (vanishing gradient), où les couches les plus profondes cessent d'apprendre efficacement. ResNet50 corrige ce problème grâce aux "skip connections" (connexions résiduelles).

Résultat : Il permet d'entraîner un modèle très profond (50 couches) tout en conservant un bon signal d'apprentissage.

Meilleure Généralisation que VGG16 et Vision Transformer

- **VGG16** a montré un fort **sur-apprentissage** (overfitting). Bien qu'il atteigne un bon score sur l'entraînement, ses performances en validation et test sont **nettement inférieures** à celles de ResNet50.
- **Vision Transformer (ViT)** nécessitait **beaucoup plus de données d'entraînement** pour bien généraliser et montrait une performance inférieure sur des datasets de taille moyenne.
- **ResNet50** s'est révélé **plus robuste en généralisation**, avec un bon compromis entre la capacité d'apprentissage et l'évitement du sur-apprentissage.

Résultat : Il **généralise mieux sur les données inconnues**, avec un **écart raisonnable entre train et test**.

Meilleur Équilibre Précision / Temps de Calcul

- **MobileNetV2 et EfficientNetB0** sont conçus pour des applications mobiles et sont donc très légers. Cependant, leur précision restait **légèrement inférieure** à celle de ResNet50.
- **VGG16 et ViT** sont beaucoup **plus lents** et nécessitent **des ressources GPU importantes**.
- **ResNet50**, bien qu'un peu plus lourd que MobileNetV2, offre **un meilleur compromis entre précision et coût de calcul**.

Robustesse de la Classification Multi-Classes

L'architecture de ResNet50 s'adapte très bien aux **problèmes de classification multi-classes**.

Sur les **27 classes du dataset**, il a montré une capacité supérieure à :

- **Bien classifier les classes majoritaires** (ex: classes Carte Collection, Jouet tech, Autour de la piscine).
- **Limiter la confusion entre les classes proches** par rapport aux autres modèles.

Résultat : Moins d'erreurs de classification et meilleure robustesse sur des classes complexes.

Résultats Supérieurs au Benchmark Rakuten (55%)

L'un des objectifs était de dépasser le benchmark de 55% en F1-score sur le dataset Rakuten. ResNet50 a atteint :

- 57.2% d'accuracy en test,
- F1-score de 59.75%, soit +4.75% au-dessus du benchmark.

Résultat : Objectif atteint

Recommandations :

Malgré ces bons résultats, on pourrait encore améliorer le modèle, notamment en :

- Augmentant le nombre d'epochs avec du early stopping.
- Explorant des modèles encore plus avancés.
- Utilisant plus de données pour mieux classifier les classes sous-représentées.

Cependant, par manque de temps et ressources, nous avons décidé de nous arrêter sur ResNet50.

- Il répond aux attentes du projet avec une bonne accuracy et un F1-score optimisé.
- Il s'entraîne en un temps raisonnable tout en atteignant une performance stable sur test.

Conclusion : ResNet50 est le modèle retenu, car il représente le meilleur compromis entre précision, rapidité et généralisation.

VI Modélisation multimodale pour la Classification des Produits Rakuten :

Dans le cadre de la classification des produits Rakuten, nous disposons de deux sources d'information : les descriptions textuelles et les images associées. Jusqu'à présent, ces sources ont été exploitées séparément, avec un modèle basé sur **CamemBERT** pour le texte et un modèle **ResNet** pour l'image. L'objectif est désormais d'explorer des stratégies de fusion multimodale afin d'améliorer la performance globale en combinant efficacement ces deux types de données.

Plusieurs travaux ont montré que la fusion au niveau des décisions (*decision-level fusion*) obtient de meilleures performances que la fusion au niveau des caractéristiques (*feature-level fusion*) dans différents contextes de classifications. En suivant cette approche, nous avons expérimenté plusieurs méthodes visant à optimiser la combinaison des prédictions texte et image tout en évitant le surapprentissage.

1- Workflow

L'expérimentation repose sur l'entraînement des modèles individuels les plus performants pour chaque modalité :

- **CamemBERT** pour le texte, sélectionné pour sa robustesse sur des tâches de classification sémantique.
- **ResNet** pour l'image, identifié comme le meilleur modèle parmi ceux testés.

L'enjeu principal est d'optimiser la fusion des prédictions afin d'exploiter pleinement la complémentarité des deux modalités. Plusieurs stratégies ont été évaluées :

1. Max Rule (Règle du maximum)

Sélectionne la classe avec la probabilité maximale parmi celles prédites par les modèles texte et image.

2. Voting Classifier (Soft Voting)

Combine les probabilités des modèles en moyennant les scores de confiance pour éviter qu'un seul modèle ne domine totalement la décision.

3. StackingClassifier

Utilise un méta-modèle (régression logistique, SVM ou XGBoost) qui apprend à pondérer les prédictions des modèles de base pour améliorer la classification.

2- Résultats et Analyse :

3- Performances des Modèles Individuels

Pour rappel, les performances initiales des modèles sur l'ensemble de validation sont les suivantes :

- CamemBERT (texte) : F1-score validation = 0.8859
- ResNet (image) : F1-score validation = 0.5966

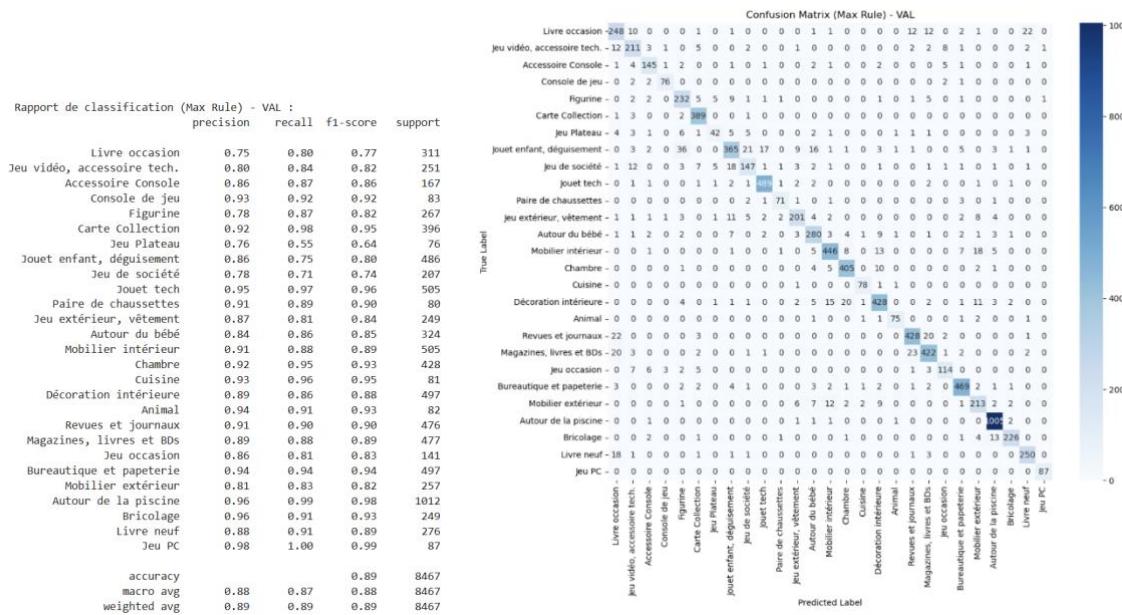
L'analyse des résultats montre que le modèle texte surpassé largement le modèle image, suggérant que l'information textuelle est plus discriminante. Cependant, certaines catégories de produits présentent une forte composante visuelle, ce qui laisse penser que le modèle image pourrait apporter une information complémentaire.

4- Fusion par Max Rule :

La première approche testée est la Max Rule, où la classe retenue est celle ayant la probabilité la plus élevée entre les deux modèles. L'implémentation repose sur la conversion des probabilités des modèles texte et image en tableaux NumPy pour assurer leur compatibilité. Ensuite, chaque modèle prédit une classe via `np.argmax()`, qui sélectionne la classe ayant la probabilité la plus élevée. La fusion suit une règle simple : si les deux modèles prédisent la même classe, elle est conservée ; sinon, la classe avec la plus haute probabilité est retenue. Cette approche vise à exploiter la complémentarité des deux modèles en privilégiant les prédictions les plus confiantes.

Les performances obtenues montrent une quasi-stagnation des résultats :

- F1-score pondéré après fusion : 0.8899 (contre 0.8859 pour CamemBERT seul).



L'analyse des performances met en évidence la prédominance du modèle texte dans la prise de décision. Grâce à sa précision élevée et sa capacité à capturer les nuances sémantiques des descriptions de produits, il constitue l'élément clé de la classification. À l'inverse, le modèle image (ResNet, F1-score validation = 0.5966) apporte une contribution limitée et n'améliore pas significativement la performance globale. Lorsqu'une divergence survient entre les prédictions des deux modèles, la décision finale privilégie généralement le texte, expliquant ainsi la stagnation des résultats après fusion. L'amélioration observée est probablement due à une meilleure classification de certaines catégories visuellement distinctes.

Certaines catégories ont toutefois bénéficié d'une amélioration notable (+ de 5%) :

- Livres d'occasion : L'image a aidé à mieux différencier les livres d'occasion des livres neufs, probablement grâce aux signes d'usure visibles sur les couvertures. De plus, les erreurs avec Magazines, Livres et BDs ont diminué, suggérant une meilleure reconnaissance des couvertures.
- Livres neufs, inversement, sont moins souvent confondus avec les livres d'occasion, confirmant l'apport du modèle image.
- Jeu vidéo, accessoire tech : Meilleure distinction avec Accessoire Console, grâce à des indices visuels tels que les manettes ou casques.
- Jeu de société et Jeu plateau : Légère réduction des confusions, indiquant que l'image apporte une aide modérée.

En revanche, certaines erreurs persistent entre des catégories sémantiquement ou visuellement proches, notamment entre :

- Livre Occasion et Magazines, Livres et BDs.
 - Jeu de Société et Jeu Plateau.
 - Jouet Enfant, Déguisement et Jouet Tech.

Ces observations montrent que la fusion Max Rule n'exploite pas suffisamment la complémentarité entre texte et image. Le modèle image (ResNet) est trop faible pour influencer positivement la fusion, ce qui limite l'intérêt d'une approche basée uniquement sur la Max Rule. De nouvelles stratégies de fusions plus avancées pourraient être plus adaptées.

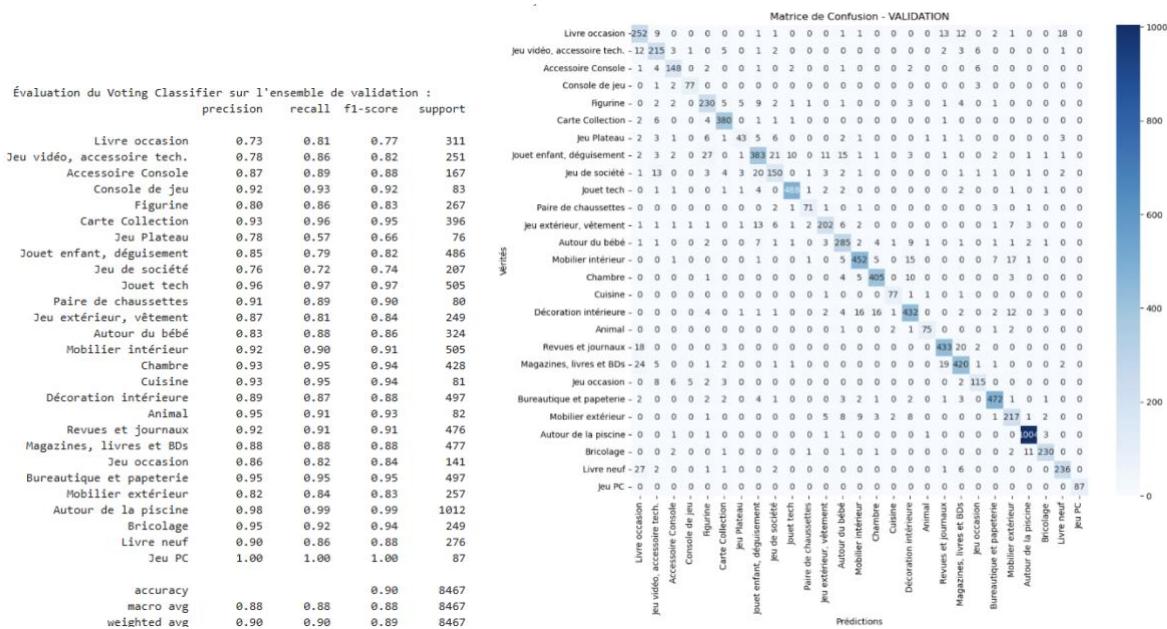
5- Fusion par Voting Classifier (Soft Voting) :

Nous utilisons maintenant un Voting Classifier en mode Soft Voting pour fusionner les prédictions des modèles texte (CamemBERT) et image (ResNet) dans la classification des produits Rakuten. L'objectif est d'optimiser l'intégration des deux sources d'information afin d'améliorer la performance globale. L'hypothèse sous-jacente est que la fusion des probabilités issues des deux modèles permettrait de mieux exploiter leurs complémentarités et d'obtenir une classification plus robuste. Une pondération optimisée (0.6 pour CamemBERT et 0.4 pour ResNet) a été appliquée après plusieurs tests.

Le Voting Classifier de scikit-learn exige une matrice de features en entrée, mais comme nous utilisons directement les probabilités pré-calculées des modèles CamemBERT et ResNet, une matrice factice remplie de zéros a été créée pour contourner cette contrainte technique. En mode Soft Voting, les probabilités des deux modèles sont moyennées, et la classe avec la probabilité moyenne la plus élevée est retenue. Contrairement à la Max Rule, cette approche permet de pondérer les contributions des modèles plutôt que de sélectionner uniquement la prédiction la plus forte.

Les performances obtenues restent similaires à celles de la Max Rule :

- F1-score pondéré : 0.8899



L'analyse des résultats montre que cette approche ne permet pas d'améliorer significativement la classification globale. Toutefois, elle corrige légèrement certaines erreurs spécifiques, notamment en réduisant la confusion entre Livre Occasion et Magazines, Livres et BDs, entre Livre Neuf et Livre Occasion et entre Jeu Vidéo, Accessoire Tech et Accessoire Console.

Les confusions majeures restent néanmoins inchangées, notamment entre Jeu de Société et Jeu Plateau, ainsi que Jouet Enfant, Déguisement et Jouet Tech, ce qui montre que la fusion actuelle n'exploite pas pleinement les informations visuelles.

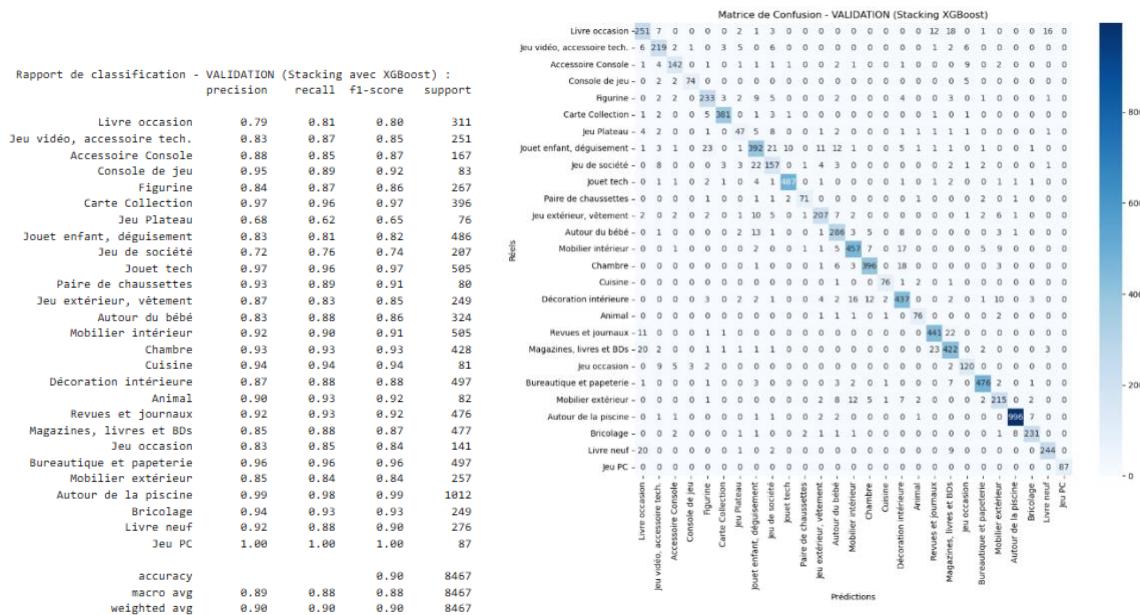
6- Fusion par Stacking Classifier :

La prochaine approche testée est le Stacking Classifier, où un méta-modèle de fusion final (Level-1) qui apprend à pondérer intelligemment les prédictions des modèles texte et image de base (Level-0). Nous avons testé deux versions :

- Stacking avec une régression logistique (LogReg).
 - Stacking avec un SVM comme méta-modèle.
 - Stacking avec un XGBoost comme méta-modèle, qui a donné les meilleurs résultats.

Nous présentons ici le Stacking avec XGBoost comme estimateur final, car il a obtenu de meilleures performances que la régression logistique et le SVM dans nos expérimentations. :

- F1-score pondéré : 0.9002 (+1.61% par rapport à CamemBERT seul).



L'analyse des erreurs et des performances détaillées par catégorie met en évidence des améliorations significatives pour certaines classes, tandis que d'autres restent problématiques.

Catégories bénéficiant d'une amélioration

L'approche Stacking a permis de réduire certaines confusions, notamment pour les catégories où l'image apporte une information visuelle complémentaire :

- Livre Occasion : Le F1-score passe de 0.77 à 0.80. L'image aide à mieux distinguer les livres neufs des livres d'occasion, probablement en capturant des indices visuels tels que l'usure des couvertures. La confusion avec Magazines, Livres et BDs diminue également.
 - Jeu Vidéo, Accessoire Tech : Le F1-score augmente légèrement de 0.82 à 0.85. L'image contribue à différencier plus efficacement les accessoires liés aux jeux vidéo (manettes, casques) de ceux destinés aux consoles.
 - Jeu de Société et Jeu Plateau : Ces catégories affichent un léger gain de performance suggérant que l'image permet une meilleure distinction entre ces jeux souvent visuellement proches mais textuellement différents.

- Mobilier Intérieur : La réduction des confusions avec Mobilier Extérieur indique que l'image apporte un signal pertinent pour distinguer ces catégories.

Catégories où les erreurs persistent

Malgré ces améliorations, certaines confusions restent inchangées, indiquant que l'image ne parvient pas encore à compenser pleinement les limites du modèle texte :

- Jouet Enfant, Déguisement vs. Jouet Tech : Ces classes restent difficiles à différencier, car elles regroupent des produits visuellement et textuellement proches, rendant la distinction complexe même avec l'ajout de l'image.
- Autour du Bébé vs. Jouets et Mobilier : Les erreurs persistent, probablement en raison de similitudes visuelles et fonctionnelles entre ces produits (mobilier pour enfant, jouets éducatifs, équipements de puériculture).

Le Stacking s'est révélé être la méthode de fusion la plus performante parmi celles testées, offrant une meilleure combinaison des prédictions issues des modèles texte et image. Toutefois, les améliorations restent limitées, ce qui montre que l'approche multimodale actuelle n'exploite pas encore pleinement les apports de l'image. Si certaines catégories initialement ambiguës bénéficient d'une classification plus précise, des confusions récurrentes persistent entre classes visuellement et textuellement similaires.

7- Tentative de Fusion Voting + Stacking :

Enfin, une approche en deux étapes a été testée, combinant :

1. **Un Voting Classifier (Soft Voting)** pour fusionner les probabilités de CamemBERT et ResNet.
2. **Un modèle de Stacking final** intégrant les probabilités du VotingClassifier ainsi que les sorties brutes des modèles individuels.

Cette méthode n'a pas permis d'obtenir une amélioration significative. Les performances obtenues sont inférieures à celles du **Stacking direct**, suggérant que l'utilisation du Voting Classifier en amont ne permet pas d'exploiter efficacement l'information multimodale.

VII Bilan :

Ce projet nous a permis d'explorer la **classification multimodale** à partir de données hétérogènes issues d'un environnement e-commerce réel. L'intégration du **traitement du langage naturel (NLP)** et de la **vision par ordinateur (Computer Vision)** a représenté un défi technique majeur, nous confrontant à des problématiques concrètes fréquemment rencontrées en entreprise.

Nous avons mené une **analyse exploratoire et un preprocessing poussé**, aussi bien sur les données textuelles que visuelles. Cette étape a mis en évidence des enjeux de qualité des données (valeurs manquantes, bruit, doublons, etc.) nécessitant la mise en place de **stratégies robustes de prétraitement**, essentielles pour garantir des résultats fiables.

En phase de modélisation, avant d'expérimenter des combinaisons **multimodales** via des techniques de **late fusion**, nous avons exploré différentes approches **unimodales** (texte seul, image seule) :

- Coté texte , nous avons pu évaluer des modèles ML : SVM, Random Forest, Regression Logistique FastText supervised et un modèle de type Transforme pre entrainé CamemBERT
- Coté images, nous avons pu évaluer des modèles CNN de base, mais aussi des modèles prés entraînés comme Resnet50, VGG16, MobileNetV2 et EfficientNetB0

Les performances obtenues par **CamemBERT** (texte) et **ResNet50** (image), puis combinées dans un **StackingClassifier**, se sont révélées très solides, nous permettant de dépasser les benchmarks initiaux proposés par Rakuten.

Au-delà de l'aspect technique, ce projet a aussi renforcé notre capacité à :

- 1 Travailler efficacement en équipe,
- 2 Piloter un projet de bout en bout,
- 3 Adapter nos choix techniques aux contraintes matérielles et temporelles d'un pipeline de data science en environnement réel.

1- Résultats (Benchmark Projet vs Benchmark Rakuten) :

Nos différentes approches, qu'elles soient **unimodales** ou **multimodales**, ont significativement **dépassé les performances de référence fournies par Rakuten**. En traitement **texte seul**, le modèle **CamemBERT** a atteint un **F1-score pondéré de 0.8859**, contre **0.8113** pour le benchmark. Ce résultat marque une avancée nette par rapport aux performances obtenues avec des modèles plus classiques, tels que ceux basés sur **TF-IDF combiné à un SVM ou modèle FastText supervisé**, confirmant l'efficacité des modèles de type Transformer pour la classification de texte. Du côté **image seule**, le modèle **ResNet50** a atteint **59.78 %**, là aussi au-dessus du benchmark établi à **55.34 %**. Enfin, notre approche **multimodale**, combinant les prédictions texte et image via un **StackingClassifier**, a permis d'obtenir un **F1-score pondéré de 0.9002**. Cette performance témoigne de l'intérêt de croiser différentes modalités d'information pour renforcer la qualité des prédictions. L'ensemble de ces résultats valide nos choix méthodologiques et souligne la robustesse des modèles développés au cours du projet.

2- Difficultés rencontrées et évolutions possible :

Au cours du projet, des contraintes techniques et organisationnelles ont limité certaines expérimentations. En effet, le volume important — plus de 80 000 produits répartis en 27 catégories — a fortement influencé nos choix techniques, les temps d'entraînement pouvant dépasser 8 heures.

2.1 Limitations techniques

L'expérimentation menée dans le cadre de ce projet a été fortement limitée par les contraintes de temps et les ressources matérielles à notre disposition. En particulier, la **disponibilité restreinte de GPU** a freiné nos possibilités d'entraînement sur des modèles de grande taille et d'exploration d'architectures plus complexes. Ces limitations ont eu plusieurs conséquences directes sur le périmètre de nos expérimentations.

Tout d'abord, nous n'avons pas pu tester certains modèles plus puissants, comme **CamemBERT Large**, **RoBERTa** ou **DeBERTa**, pourtant reconnus pour leur capacité à capter des représentations plus fines des textes. De la même manière, l'évaluation de **modèles multimodaux récents**, tels que **CLIP**, **LLAVA** ou encore **GIT**, n'a pas été possible, ces derniers nécessitant une infrastructure technique avancée et un temps d'entraînement important.

Par ailleurs, nos ressources limitées ont également restreint notre capacité à conduire une **exploration approfondie des hyperparamètres**, notamment via **GridSearch** ou des approches plus avancées comme l'optimisation bayésienne. Cela a réduit les marges d'optimisation fine de nos modèles, tant du côté texte que du côté image.

2.2 Modèles non explorés

Faute de temps et de ressources, plusieurs modèles et stratégies prometteurs n'ont pas pu être testés. C'est le cas de **BERT** et **mBERT**, qui auraient permis une comparaison multilingue utile pour les descriptions produits en langues diverses. De même, **CamemBERT Large**, bien que pertinent pour une amélioration des performances sur les meilleurs jeux de données nettoyés, n'a pas pu être intégré à notre pipeline.

Nous n'avons pas non plus pu expérimenter certains **modèles récents plus sophistiqués**, souvent coûteux en calcul, qui auraient pu renforcer les performances, en particulier sur les données multimodales.

2.3 Qualité hétérogène des données

Une autre difficulté majeure a concerne la **qualité inégale des données** fournies. Du côté texte, environ **35 % des produits** présentaient des **descriptions manquantes**, réduisant la richesse de l'information exploitable. Du côté image, la qualité variait fortement : flous, objets mal cadrés, biais colorimétrique (notamment une **dominance du rouge**), rendant nécessaire un travail important de **nettoyage et de prétraitement visuel**. Enfin, la **présence de doublons**, textuels ou visuels, parfois associés à **des catégories différentes**, a introduit une complexité supplémentaire dans la phase de modélisation.

2.4 Expérimentations incomplètes

Plusieurs aspects de notre pipeline auraient mérité des approfondissements supplémentaires. Par exemple, nous n'avons pas mis en œuvre de **mécanismes de régularisation avancés**, comme le **dropout contrôlé**, qui aurait pu améliorer la **généralisation** des modèles profonds.

De plus, aucune démarche d'**interprétabilité** n'a été intégrée dans notre projet. L'utilisation d'outils comme **LIME**, **SHAP**, ou encore les **visualisations d'attention** dans les Transformers ou les **heatmaps Grad-CAM** pour les CNN, aurait permis de mieux comprendre les décisions des modèles, de détecter d'éventuels biais et d'affiner notre analyse.

Enfin, bien que des prétraitements aient été appliqués avec soin, ils n'ont pas été poussés à leur maximum. La **gestion des stopwords** aurait pu être plus fine, certaines **descriptions textuelles** restaient peu enrichies ou mal traduites, et l'approche de **rééquilibrage des classes** s'est limitée à une combinaison de **SMOTE et undersampling**, sans test comparatif d'autres stratégies comme **ADASYN**, **Tomek Links** ou l'oversampling simple.

VIII Bibliographie :

Traitement des données textuelles (NLP)

- **SpaCy** – Bibliothèque pour le traitement du langage naturel (lemmatisation, tokenisation, etc.)
⦿ <https://spacy.io/>
- **NLTK (Natural Language Toolkit)** – Pour les stopwords, analyse linguistique, etc.
⦿ <https://www.nltk.org/>
- **CamemBERT** – Modèle Transformer pour la langue française basé sur RoBERTa
⦿ <https://camembert-model.fr/>
- **mBERT** – BERT multilingue pour le traitement de textes en plusieurs langues
⦿ <https://huggingface.co/bert-base-multilingual-cased>
- **FastText** – Outil de classification supervisée rapide de textes (Facebook AI)
⦿ <https://fasttext.cc/>
- **TF-IDF** – Technique classique de vectorisation de texte (implémentée via scikit-learn)
⦿ https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
- **Google Translate API** – Pour la traduction automatique des textes
⦿ <https://cloud.google.com/translate/docs>

Traitement des données images (CV)

- **OpenCV** – Traitement d'images : flou, sharpness, histogrammes de couleur, etc.
⦿ <https://opencv.org/>
- **scikit-image** – Alternative à OpenCV pour certaines transformations
⦿ <https://scikit-image.org/>
- **TensorFlow / Keras** – Chargement, normalisation et augmentation des images
⦿ https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image
- **PyTorch + torchvision** – Prétraitement et entraînement de modèles CNN
⦿ <https://pytorch.org/>
⦿ <https://pytorch.org/vision/>
- **ResNet (Residual Network)** – Architecture CNN performante pour la classification d'images
⦿ <https://arxiv.org/abs/1512.03385>
⦿ <https://pytorch.org/vision/stable/models.html#id8>

Fusion multimodale et modélisation

- **scikit-learn** – Pour les VotingClassifier, StackingClassifier, GridSearchCV, etc.
⦿ <https://scikit-learn.org/stable/>
- **Hugging Face Transformers** – Entraînement de modèles NLP de type BERT, CamemBERT, etc.
⦿ <https://huggingface.co/docs/transformers>
- **Ensembles (Voting, Stacking)** – Techniques de fusion de modèles
⦿ <https://scikit-learn.org/stable/modules/ensemble.html>

Rééquilibrage et Data Augmentation

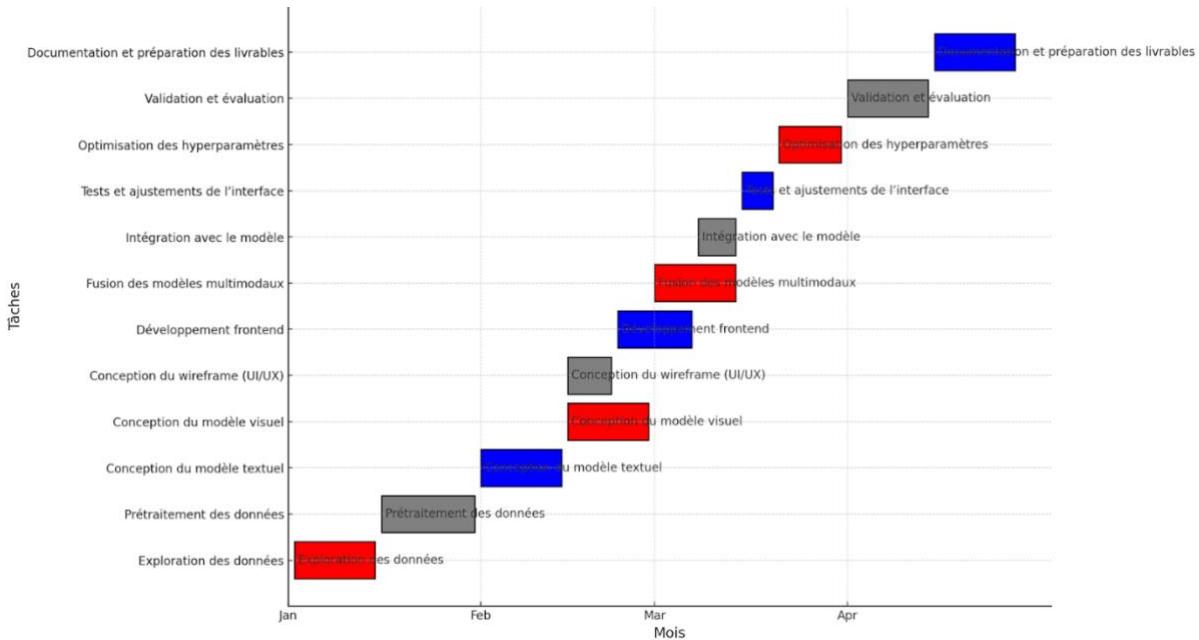
- **Imbalanced-learn (SMOTE, RandomOverSampler)**
⦿ <https://imbalanced-learn.org/stable/>
- **Albumentations** – Bibliothèque d'augmentation d'images puissante
⦿ <https://albumentations.ai/>

Suivi et évaluation

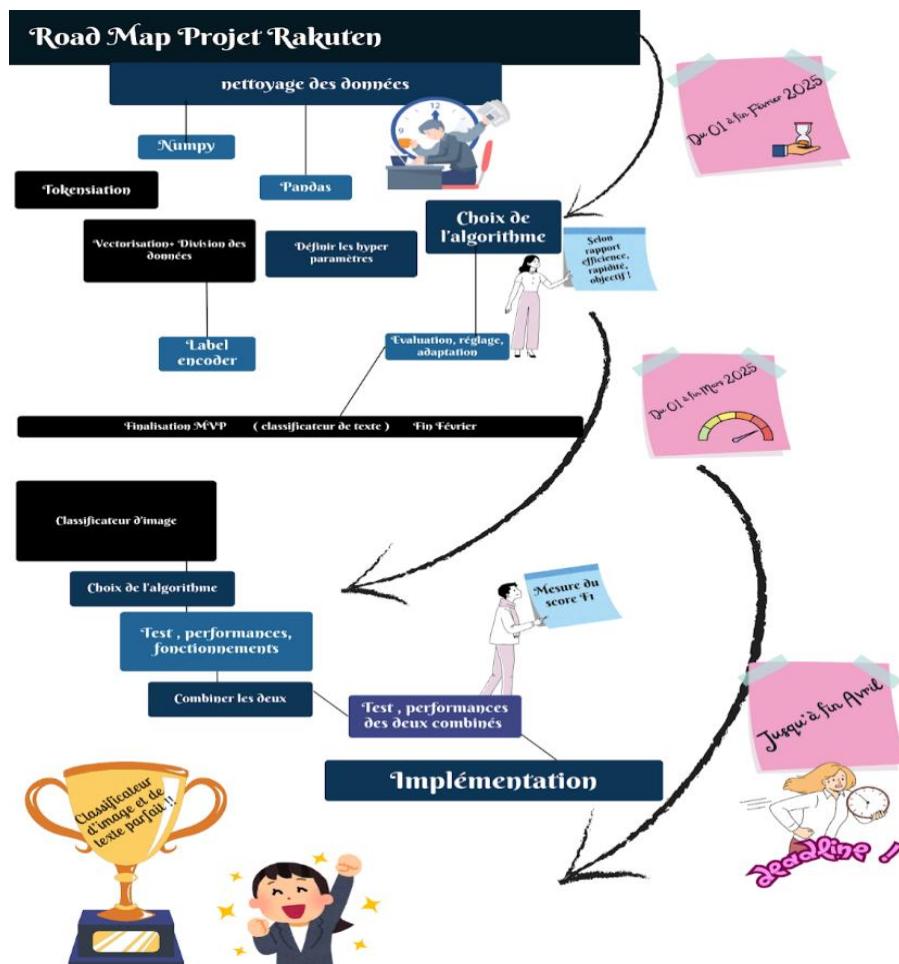
- **matplotlib / seaborn** – Visualisation des matrices de confusion, learning curves
 ⌚ <https://matplotlib.org/>
 ⌚ <https://seaborn.pydata.org/>
- **TensorBoard** – Suivi des métriques d'entraînement pour les modèles deep learning
 ⌚ <https://www.tensorflow.org/tensorboard>

IX Annexes :

1- Diagramme de GANTT :



2- Roadmap :



3- Business model CANVAS Rakuten :

Partenaires clés :	Activités clés :	Offre (proposition de valeur) :	Relation client :	Segments de clientèle :
-Vendeurs et marchands : Des entreprises qui vendent leurs produits via Rakuten.	-Gestion de la plateforme de commerce en ligne		-Support client : Service clientèle disponible pour aider les clients via divers canaux (téléphone, chat, email). -Plateforme de commerce en ligne diversifiée : Rakuten propose une large gamme de produits, des marques internationales et locales, dans une seule plate forme.	Consommateurs en ligne : les utilisateurs recherchant une grande variété de produits à acheter sur la plateforme (électronique, vêtements, livres, etc.).
-Partenaires logistiques : Entreprises spécialisées dans la gestion des livraisons et des retours.	-Services aux marchands : fournir une plateforme de vente. -Développement technologique		-Programme de fidélité : Par le biais des Rakuten Super Points, Rakuten encourage les achats récurrents.	
-Fournisseurs de solutions technologiques : Par exemple, des partenaires pour le paiement en ligne, le cloud, la publicité numérique, etc.	-Logistique et gestion des paiements Ressources clés : -Technologie et infrastructure de la plateforme.	-Numérique : Rakuten propose également des services dans d'autres domaines comme, le streaming, et la publicité en ligne.	Canaux de distribution : -Site Web : Plateforme principale de vente. -Application mobile Rakuten.	-Vendeurs tiers : Les entreprises ou individus qui vendent leurs produits sur le marché Rakuten.
-Partenaires de contenu et de streaming : Des sociétés de contenu.	-Réseau de partenaires commerciaux : les vendeurs tiers et autres entreprises partenaires avec lesquelles Rakuten collabore. -Base de clients : La fidélité de ses utilisateurs		-Partenariats avec des sites tiers : Rakuten s'associe avec d'autres sites de e-commerce et des applications. -Publicité : Rakuten utilise des canaux numériques pour promouvoir ses services.	

Structure des coûts :	Sources de revenus :
<p>-Frais de technologie et d'infrastructure : Développement, maintenance et hébergement de la plateforme e-commerce et des services numériques.</p> <p>-Coûts de marketing et de publicité : Dépenses pour attirer de nouveaux clients et promouvoir les produits.</p> <p>-Coûts logistiques et de gestion des stocks : Frais liés à la gestion des livraisons et du service après-vente.</p>	<p>-Commissions sur les ventes : Rakuten prend une commission sur chaque vente réalisée par un vendeur tiers sur sa plateforme.</p> <p>-Abonnements aux services premium : Certains services de Rakuten, comme Rakuten Global Express ou Rakuten TV, génèrent des revenus par des abonnements.</p> <p>-Publicité en ligne : Revenus générés par les publicités sur la plateforme et dans ses services numériques.</p> <p>-Ventes directes : Rakuten vend également des produits de ses propres marques ou en tant qu'intermédiaire.</p>