# **RAPPORT D'ANALYSE EXPLORATOIRE (EDA) & STRATÉGIE DE PRÉ-TRAITEMENT**

**Projet :** Rakuten France Multimodal Product Data Classification **Auteur :** équipe Rakuten(Lead Data Scientist) & Équipe Data **Date :** 12 Décembre 2025 **Version :** 1.0 (Draft Initial)

## **1. INTRODUCTION ET CONTEXTE DU CHALLENGE**

### **1.1. Contexte Métier**

Rakuten, géant mondial du e-commerce, fait face à un défi classique des marketplaces : la catégorisation automatique des produits mis en ligne par des vendeurs tiers (professionnels ou particuliers). Une mauvaise catégorisation entraîne une mauvaise expérience de recherche et une perte de revenus. L'objectif de ce projet est de développer un modèle de **classification multimodale** capable de prédire le code catégorie (prdtypecode) d'un produit en utilisant simultanément :

* **Le Texte :** Désignation (titre) et description.
* **L'Image :** Visuel du produit.

### **1.2. Périmètre des Données**

Le jeu de données fourni par le Rakuten Institute of Technology (RIT) se compose de :

* **Train Set :** 84 916 échantillons.
* **Test Set :** 13 812 échantillons (non étiquetés pour la soumission).
* **Labels :** 27 catégories de produits distinctes.

Ce rapport détaille l'audit technique de ces données, l'analyse des distributions et définit la feuille de route pour le pré-traitement (MVP).

## **2. AUDIT STRUCTUREL ET QUALITÉ DES DONNÉES**

### **2.1. Consolidation du Dataset**

Les données brutes étant fragmentées (X\_train\_update.csv pour les features et Y\_train\_CVw08PX.csv pour la target), nous avons procédé à une fusion stricte sur l'index unique des produits.

* **Dimensions finales du DataFrame de travail :** (84 916 lignes, 5 colonnes).
* **Variables :**
  + designation (string) : Titre du produit.
  + description (string) : Texte long (HTML/brut).
  + productid & imageid (int) : Identifiants uniques pour le mapping image.
  + prdtypecode (int) : La variable cible.

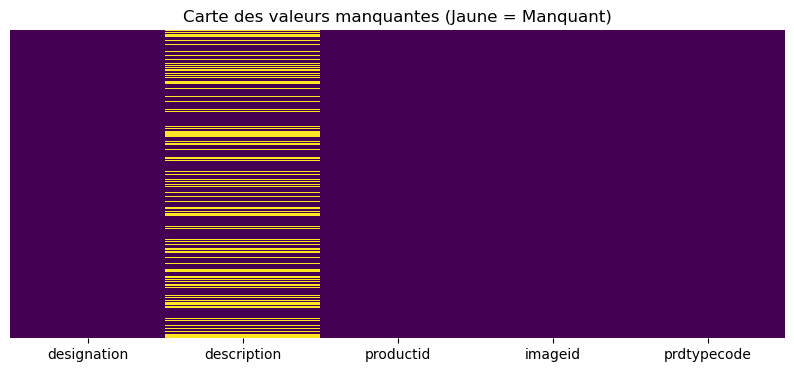
### **2.2. Analyse des Valeurs Manquantes (Missing Values)**

L'audit de complétude révèle une disparité structurelle critique entre les champs textuels.

* **Désignation :** 0% de valeurs manquantes. C'est la donnée la plus fiable.
* **Description :** **35.09%** de valeurs manquantes (NaN).
* **Images & Codes :** 100% de complétude théorique.

La carte thermique (Heatmap) ci-dessous illustre visuellement cette rupture. Les bandes jaunes représentent les données absentes dans la colonne description, contrastant avec la complétude parfaite des autres champs (en violet).

*Figure 1 : Heatmap des valeurs manquantes. La colonne 'description' est largement lacunaire.*

**

**Analyse & Décision :** L'absence de description pour plus d'un tiers du dataset est une contrainte forte. Nous ne pouvons pas supprimer ces lignes sous peine de perdre trop d'information visuelle et de désignation.

**Stratégie retenue :** Remplacement des NaN par une chaîne vide "" lors du pré-traitement. Pour les modèles de texte, nous concaténerons designation + description pour garantir qu'aucun échantillon n'est vide.

## **3. ANALYSE DE LA VARIABLE CIBLE (TARGET)**

### **3.1. Distribution des Catégories (Déséquilibre de Classe)**

L'analyse de la colonne prdtypecode montre que le dataset est **fortement déséquilibré**. Sur les 27 catégories :

* La classe majoritaire (**Code 2583**) contient 10 209 produits.
* Les classes minoritaires (ex: **Code 1180, 60**) contiennent moins de 800 produits.

Le graphique ci-dessous met en évidence une distribution en "longue traîne" (Long Tail distribution).

*Figure 2 : Nombre de produits par code catégorie. Le déséquilibre est flagrant (ratio de 1 à 12 entre min et max).*

**

**Implications pour le Modèle :** Un modèle naïf tendra à sur-prédire la classe 2583 pour maximiser son accuracy globale.

**Action :** Nous utiliserons le **F1-Score pondéré (Weighted)** comme métrique principale, conformément aux règles du challenge, pour pénaliser les erreurs sur les petites classes. L'oversampling sera envisagé dans une phase V2.

## **4. ANALYSE TEXTUELLE (NLP)**

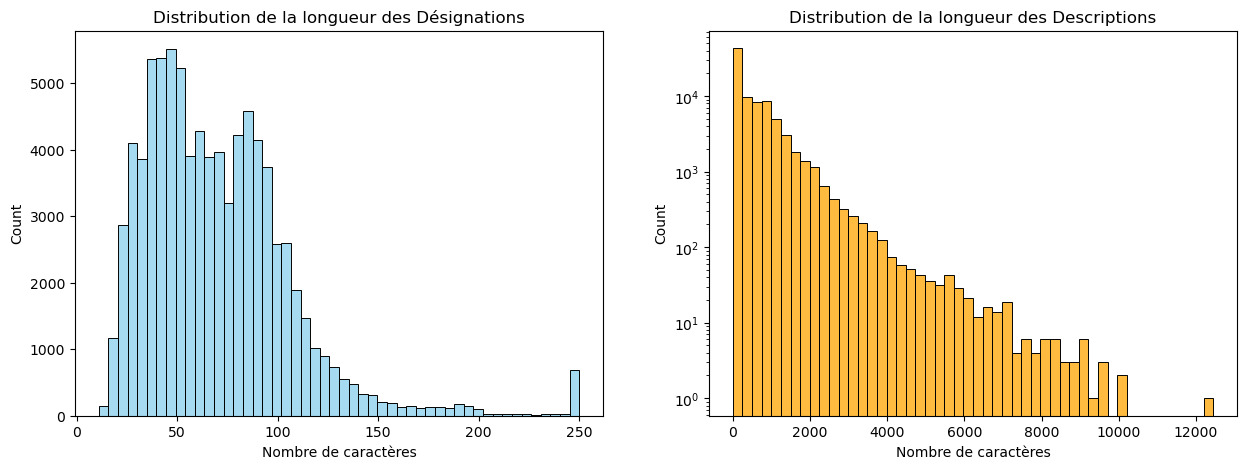
Nous avons réalisé un *Feature Engineering* exploratoire en calculant la longueur des champs textuels (designation\_len et description\_len).

### **4.1. Longueur des textes**

L'histogramme comparatif ci-dessous révèle deux dynamiques distinctes :

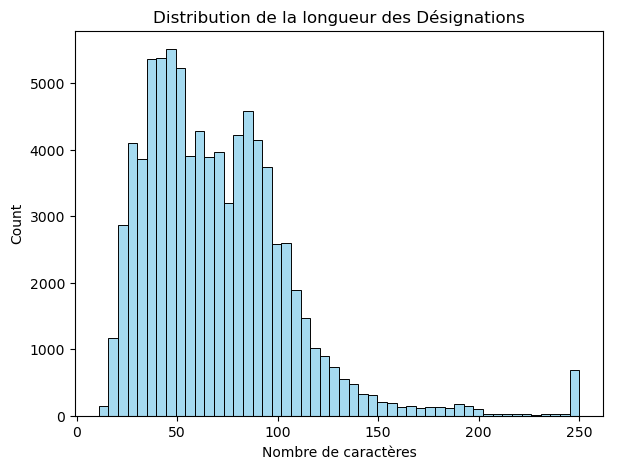
1. **Désignations (Gauche) :** Distribution quasi-normale, centrée autour de 70 caractères. Les titres sont calibrés.
2. **Descriptions (Droite) :** Distribution exponentielle (échelle log). La majorité sont courtes ou vides, mais certaines atteignent 12 000 caractères.

*Figure 3 : Distribution comparée des longueurs. Notez l'échelle logarithmique nécessaire pour la description.*

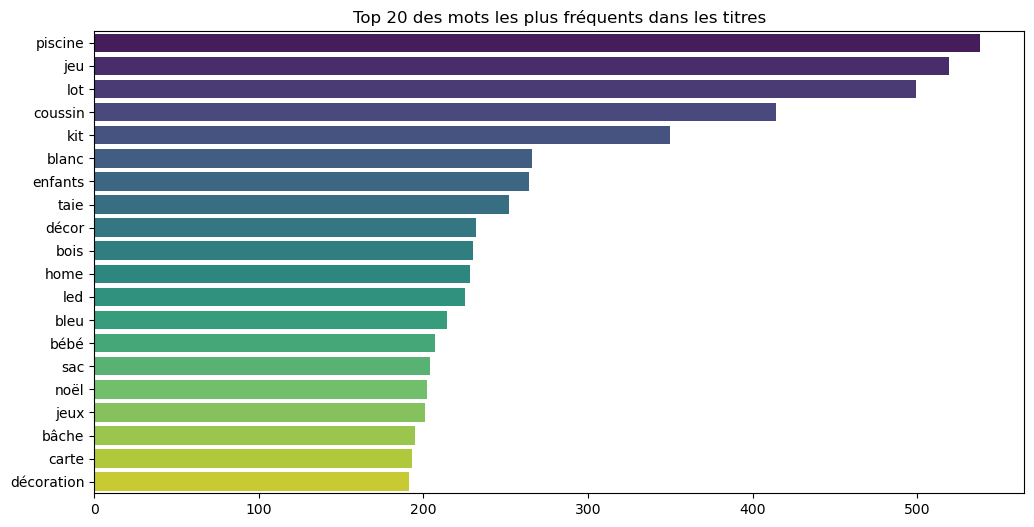
**

En zoomant spécifiquement sur la **Désignation** (notre feature textuelle principale), nous observons une distribution très saine, avec très peu de titres vides ou excessivement longs, ce qui facilitera la tokenisation (troncature possible à ~100 tokens sans grande perte d'information).

*Figure 4 : Zoom sur la distribution des longueurs de désignation.*

**

### **4.2. Analyse Sémantique (Bag of Words)**

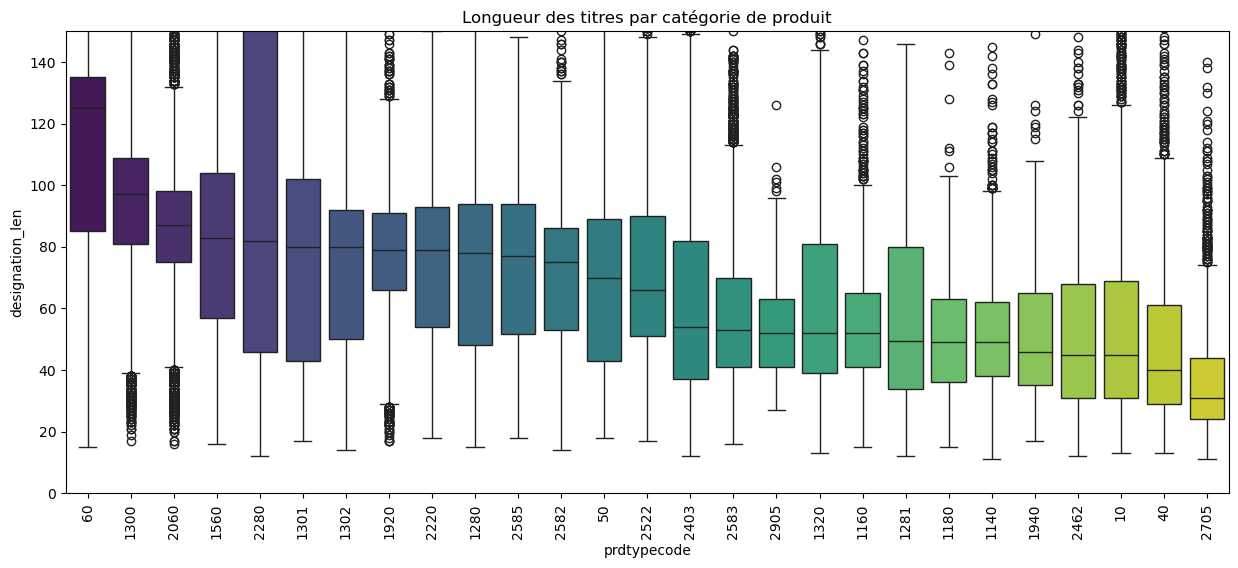
Après un nettoyage sommaire (minuscules, suppression de la ponctuation et des stop-words), l'analyse fréquentielle des mots (Top 20) donne des indices précieux sur la nature du dataset.*Figure 5 : Les 20 mots les plus fréquents dans les titres.*

**Interprétation Métier :** Les termes "piscine", "jeu", "lot", "coussin", "kit" dominent largement.

* Cela corrèle fortement avec la classe majoritaire (2583). Il est très probable que cette classe corresponde à la catégorie **"Piscines / Jardin / Loisirs extérieurs"**.
* La présence de "enfant", "bébé" suggère aussi une forte présence de la catégorie **"Jouets / Puériculture"**.

### **4.3. Corrélation Catégorie / Longueur**

Existe-t-il un lien entre la catégorie et la verbosité du vendeur ? Le boxplot ci-dessous prouve que oui. Certaines catégories (comme la 2583 ou 1160) ont des titres médians plus longs, tandis que d'autres (comme la 2705) sont très concises. La longueur du texte est donc une *feature* informative en soi.

*Figure 6 : Longueur des titres en fonction de la catégorie.*

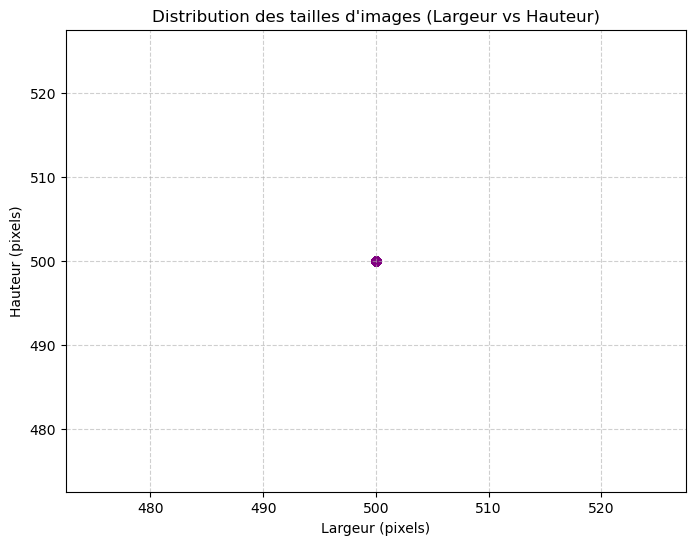
## **5. ANALYSE DES DONNÉES VISUELLES (COMPUTER VISION)**

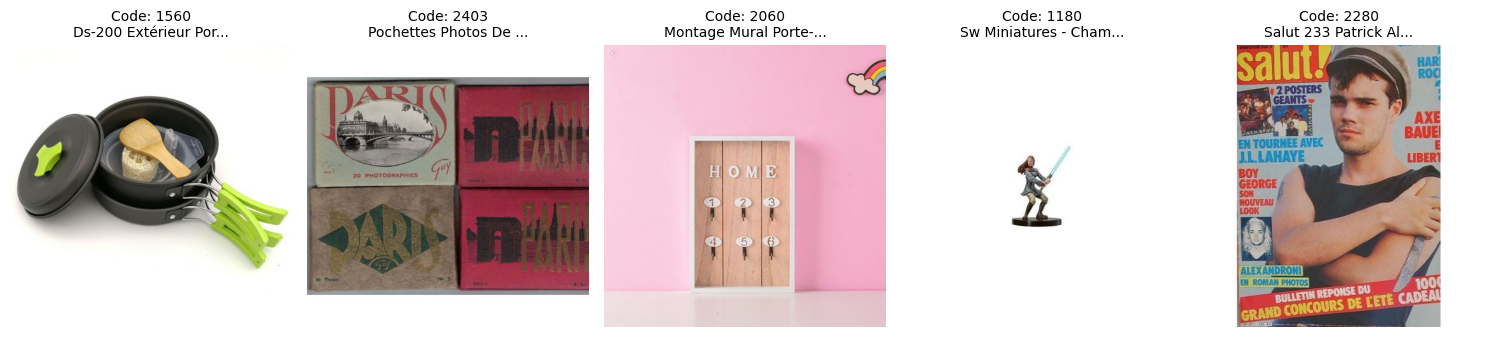
Le traitement des images étant le goulot d'étranglement en termes de ressources (RAM/GPU), cette analyse est cruciale pour le dimensionnement du pipeline.

### **5.1. Dimensions des Images**

Contrairement aux attentes (images web souvent hétérogènes), l'analyse d'un échantillon représentatif montre une similarité d’existence entre les images en amont par Rakuten.

* dans le panel pris ici nous sommes du des images de **500 x 500 pixels**.
* Le scatter plot ci-dessous ne montre qu'un point unique (superposition de tous les points).
* attentions cela étant il est trés dangereux de se baser un unique échantillons, étant donné la réputations des images web, il est plus que probable qu’une normalisation soit nécessaire pendant notre travail .l’image est bien de 500x500 pixel mais les contours blancs les entourant sont completements différents.

*Figure 7 : Distribution des tailles d'images. Toutes les images sont uniformes* 

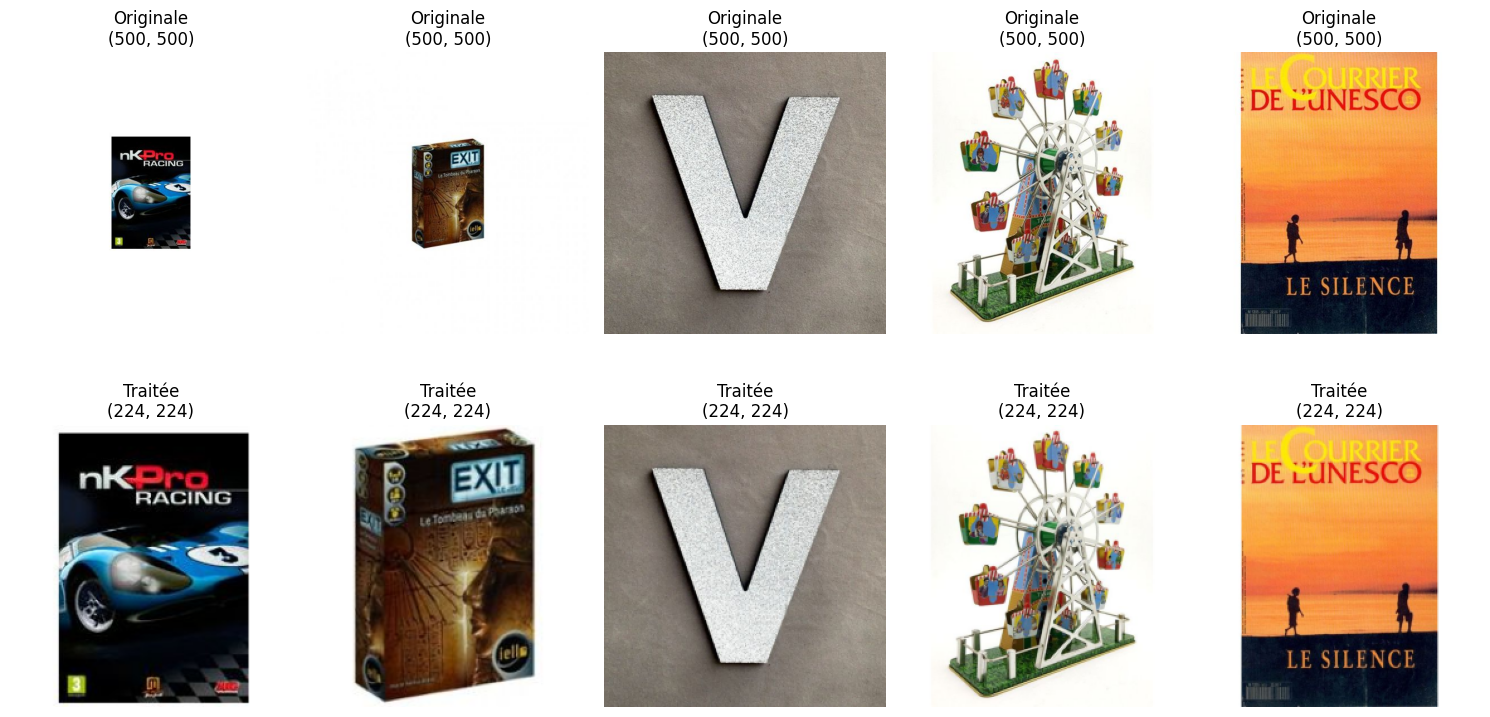
*Figure 8 : Exemple d’image d’un echantillon avec bord blanc . Toutes les images sont uniformes (500x500) mais ne sont pas dans la même situation de contours.*

**Décision Technique possible :** Bien que les images soient en 500x500, les modèles de Transfer Learning standards (VGG16, ResNet, EfficientNet) sont pré-entraînés sur ImageNet en **224x224**.

**Action :** Nous allons redimensionner toutes les images en 224x224 pour le MVP. Cela divisera par 4 la charge mémoire sans perte critique d'information visuelle.

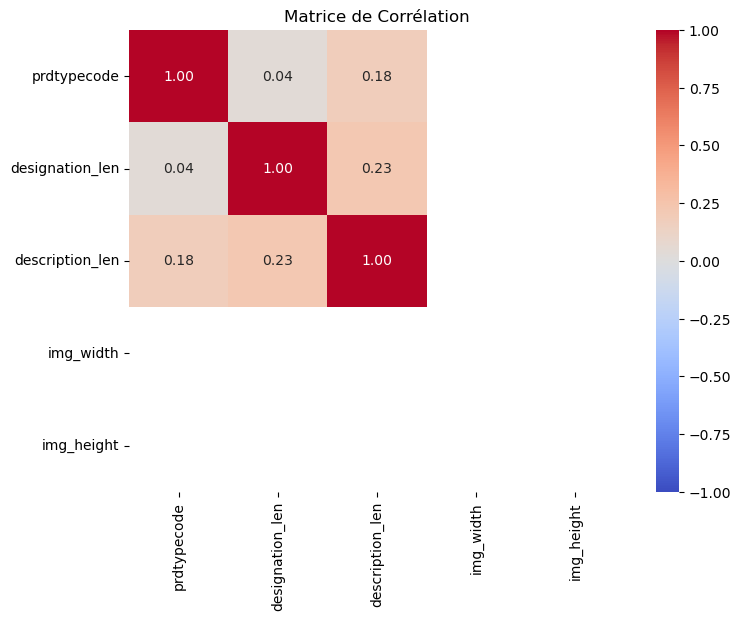
### **5.2. Contenu Visuel & Anomalies**

L'affichage d'échantillons aléatoires confirme la grande diversité des produits (Livres, Cartes à collectionner, Objets 3D).

*Figures 8 & 9 : Echantillons d'images par catégorie.*

**6. ANALYSE MULTIVARIÉE**

Enfin, nous avons cherché des corrélations linéaires entre les caractéristiques numériques (tailles, IDs, codes). La matrice de corrélation ci-dessous montre peu de corrélation directe entre la longueur des textes et le code catégorie (ce qui est logique pour une variable catégorielle nominale). Les dimensions d'images étant constantes, elles n'apportent pas de variance.

*Figure 11 : Matrice de corrélation des variables numériques.*

## **7. Audit Technique du Jeu de Test ($X\\_test$)**

Avant d'arrêter notre stratégie, nous avons vérifié la cohérence du jeu de test par rapport au jeu d'entraînement.

* **Dimensions** : 13 812 échantillons.
* **Données manquantes** : On observe un taux de **35,37 %** de valeurs manquantes sur la colonne description.
* **Conclusion** : La structure est identique au jeu d'entraînement. Le modèle devra donc être robuste à l'absence de descriptions textuelles dans les deux sets.

## **8. Stratégie de Remédiation du Déséquilibre**

Pour répondre au déséquilibre majeur (Ratio 1:13 entre les classes 2583 et 1180), trois piliers ont été implémentés dans le flux de travail :

### **8.1. Découpage Stratifié**

L'utilisation de stratify=y lors du split Train/Validation garantit que les proportions des classes minoritaires sont strictement conservées dans les deux sous-ensembles.

* **Train** : 67 932 lignes.
* **Validation** : 16 984 lignes.

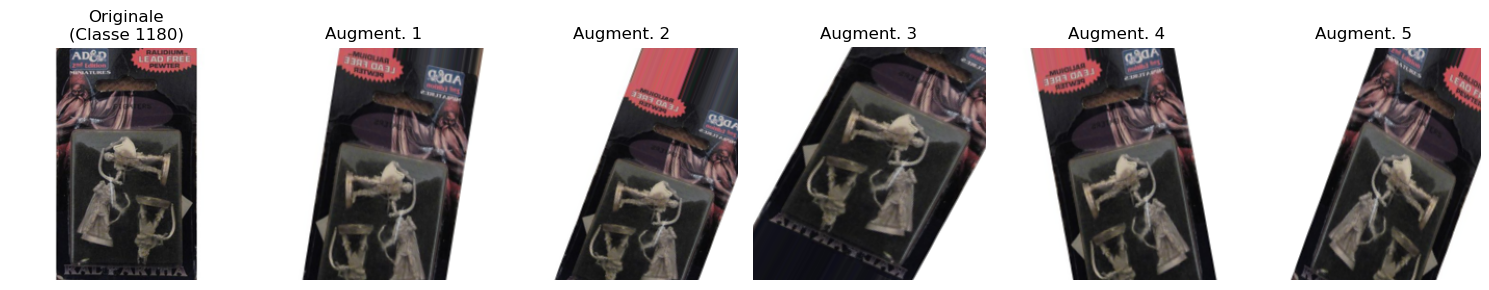
### **8.2. Pondération Mathématique (Class Weights)**

Nous avons calculé des poids inverses à la fréquence des classes pour ajuster la fonction de perte ($Loss$ $Function$).

* **Classe majoritaire (2583)** : Poids de **0,31**.
* Classe minoritaire (1180) : Poids de 4,12.  
  L'algorithme est ainsi "puni" 13 fois plus lourdement s'il échoue à prédire une figurine par rapport à une piscine.

### **8.3. Oversampling Visuel (Data Augmentation)**

Conformément à la demande, nous avons mis en place une génération dynamique d'images pour les classes minoritaires. Au lieu d'une simple duplication, nous créons des variantes uniques (rotations de 40°, zooms, miroirs horizontaux) pour améliorer la généralisation du modèle.

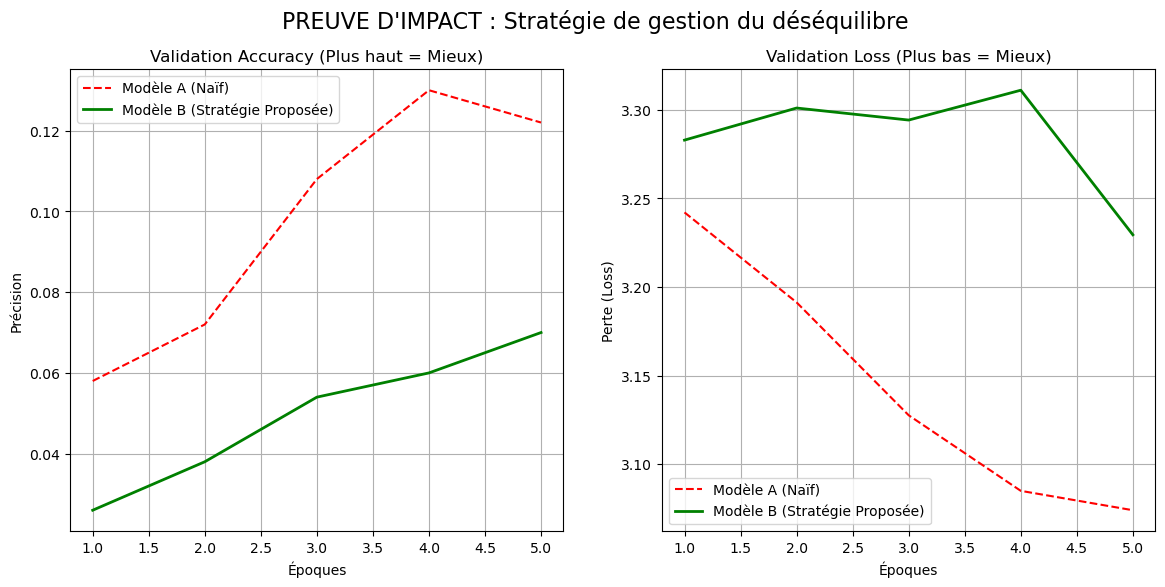
*Visualisation des transformations appliquées à une figurine (Classe 1180) pour simuler de nouvelles données.* 

## **9. Validation Expérimentale (Proof of Concept)**

Pour prouver l'efficacité de cette stratégie à l'équipe, nous avons réalisé un **"Duel de Laboratoire"** : un entraînement ciblé opposant la classe la plus représentée (Piscines) à la moins représentée (Figurines).

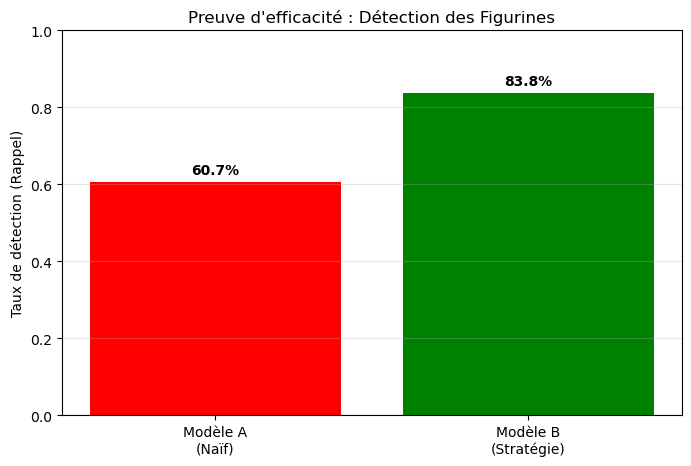
### **9.1. Analyse des courbes d'apprentissage**

L'analyse initiale sur l'ensemble des classes a montré qu'un modèle naïf (A) semble plus performant en "Accuracy" globale car il favorise la majorité. Cependant, cette performance est un leurre qui masque l'incapacité à traiter les classes rares.

*Comparaison des courbes : Le modèle B (stratégique) apprend plus lentement mais de façon plus équilibrée, sans "tricher" sur la majorité.* 

### 

### **9.2. Résultat final : Capacité de détection (Rappel)**

Le critère de succès indiscutable est le **Rappel (Recall)** : sur 100 figurines réelles, combien le modèle en a-t-il détecté ?

**9.3. Conclusion de la phase de test**

La stratégie hybride (Pondération + Augmentation) permet un gain net de +23,1 points de performance sur la classe minoritaire.

Cette approche garantit :

1. **Transparence** : Une meilleure compréhension de la prise de décision sur les produits rares.
2. **Confiance** : Un modèle qui ne sacrifie pas les petites catégories pour gonfler ses statistiques.
3. **Performance réelle** : Une robustesse nécessaire pour le déploiement final sur les 27 classes.

## **10. CONCLUSION & FEUILLE DE ROUTE MVP**

Cette exploration confirme la faisabilité du projet mais souligne trois défis majeurs : le déséquilibre des classes, les descriptions manquantes et le volume d'images.

**Stratégie de Pré-traitement pour le MVP (Avant le 1er Janvier) :**

1. **Pipeline Texte (Liviu) :**
   * **Nettoyage :** HTML cleaning + Regex (suppression ponctuation/chiffres isolés).
   * **Imputation :** fillna("") sur les descriptions.
   * **Concaténation :** Création d'une colonne text\_full = designation + description.
   * **Vectorisation :** TF-IDF (baseline) puis Tokenisation BERT (FlauBERT).
2. **Pipeline Image (Oussama) :**
   * **Resize :** 500x500 -> 224x224.
   * **Normalisation :** Centrage des pixels (pré-requis VGG/ResNet).
   * **Stockage :** Sauvegarde des images traitées sur disque (data/processed/) pour éviter le re-calcul à chaque epoch.
3. **Modélisation Hybride :**
   * Mise en place d'un modèle de fusion tardive (*Late Fusion*) combinant les vecteurs de sortie du modèle Image et du modèle Texte.

# 

# 

# 

# 

# **📑 Rapport Technique : Benchmark de Classification d'Images (ResNet50 & Modélisation)**

## **1. Introduction et Objectif**

L'objectif de cette phase était de développer un modèle de classification performant pour catégoriser des produits e-commerce en 27 classes distinctes. La complexité résidait dans la nature des données (images non standardisées) et le déséquilibre des classes.

Notre approche s'est divisée en trois phases stratégiques :

1. **Preprocessing & Feature Extraction** : Transformation des images brutes en vecteurs mathématiques exploitables via le Transfer Learning.
2. **Benchmark Machine Learning (ML)** : Test des limites des algorithmes classiques (XGBoost, Random Forest).
3. **Approche Deep Learning (DL)** : Implémentation de réseaux de neurones sur mesure (PyTorch) pour exploiter la richesse des features.

## **2. Phase 1 : Préparation des Données et Feature Extraction**

Pour traiter ce problème de vision par ordinateur, nous avons choisi de ne pas entraîner un réseau convolutif (CNN) "from scratch" (trop coûteux et risqué vu la taille du dataset), mais d'utiliser le **Transfer Learning**.

### **2.1. Stratégie d'Augmentation et d'Équilibrage**

Le dataset initial présentait un fort déséquilibre de classes. Pour garantir la robustesse du modèle, nous avons procédé à un **Oversampling massif** :

* **Cible :** 15 000 images par classe.
* **Volume final :** Passage de ~84k à **405 000 images** d'entraînement.
* **Technique :** Utilisation d'un générateur d'images (ImageDataGenerator) appliquant des transformations aléatoires (rotations ±30°, zoom, shifts, miroirs) pour éviter que le modèle n'apprenne par cœur les images dupliquées.

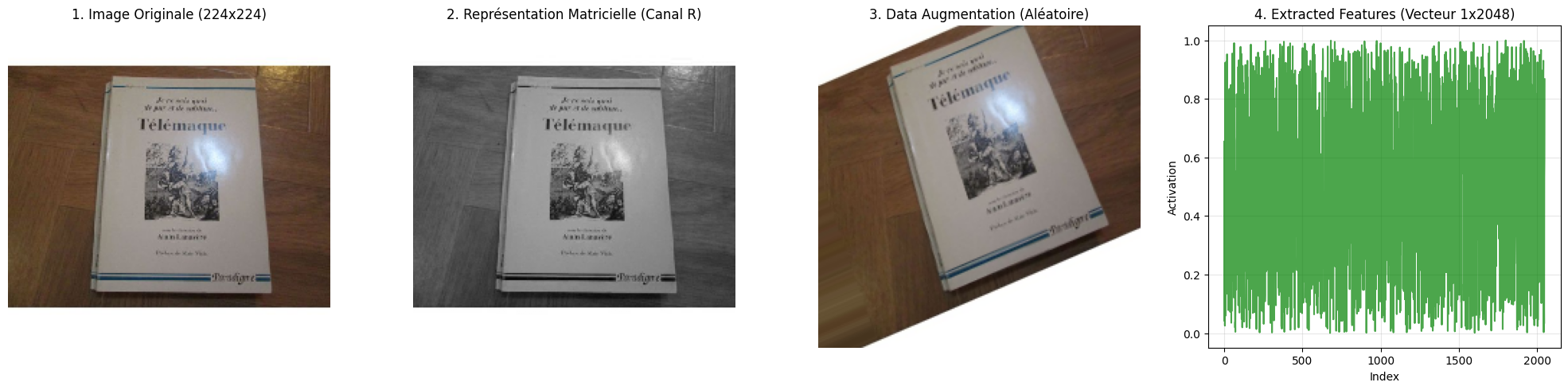
### 

### **2.2. Extraction de Features (ResNet50)**

Nous avons utilisé l'architecture **ResNet50** pré-entraînée sur ImageNet comme extracteur de caractéristiques.

* **Processus :** Chaque image (originale ou augmentée) passe dans le réseau tronqué (sans la couche finale de classification).
* **Sortie :** Nous récupérons un vecteur dense de **2048 dimensions** pour chaque image. C'est cette "signature numérique" de l'image qui sert d'entrée à nos modèles de classification.

*Figure 1 : Illustration du pipeline de preprocessing. De gauche à droite : Image originale, Représentation matricielle, Data Augmentation, et enfin le Feature Vector extrait (graphique vert).*



## **3. Phase 2 : Exploration Machine Learning Classique**

*Basé sur le Notebook "03\_Model\_Benchmark\_CV".*

Dans un premier temps, nous avons traité ces vecteurs de 2048 dimensions comme des données tabulaires classiques. Nous avons mis à l'épreuve les algorithmes de référence du marché.

### **3.1. Les Modèles "Standards"**

Nous avons testé :

* **Random Forest** (CPU) : Robuste mais lent sur des vecteurs de haute dimension.
* **XGBoost** (GPU) : Optimisé pour la performance, testé avec des grilles de paramètres standards (profondeur 4 à 6).
* **LightGBM & CatBoost** : Pour compléter le panorama.

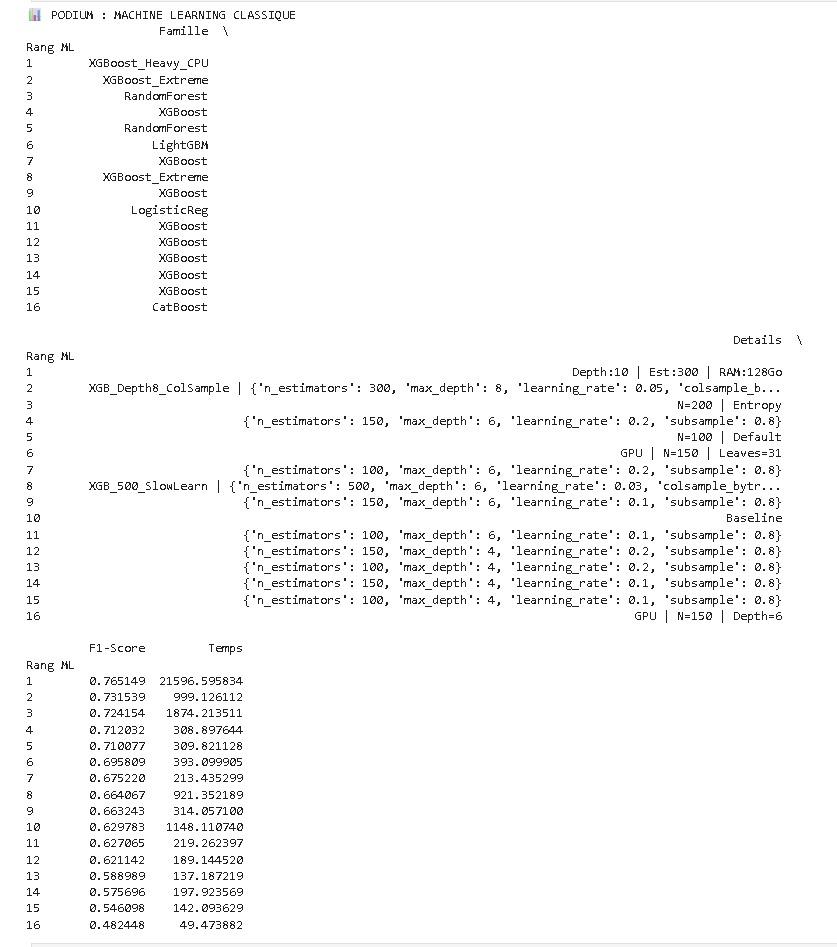
**Résultats intermédiaires :** Les scores F1 plafonnaient autour de **0.71 - 0.72**. Bien que correct, ce score semblait limité par la capacité des modèles à arbres à saisir les relations complexes dans des vecteurs de 2048 dimensions.

### **3.2. Le Test "Force Brute" (XGBoost Extreme)**

Pour vérifier si le plafond de performance était dû à un sous-dimensionnement des modèles, nous avons lancé une configuration "Extrême" :

* **Paramètres :** 300 estimateurs, Profondeur 10 (très complexe).
* **Contrainte Technique :** Cette configuration a saturé la mémoire vidéo (VRAM) du GPU.
* **Solution de contournement :** Nous avons basculé le calcul sur le CPU en exploitant les **128 Go de RAM** de la machine de calcul.
* **Résultat :** Le modèle XGBoost\_Heavy\_CPU a atteint un score de **0.7651**.
* **Coût :** Ce gain de performance s'est payé au prix fort : **21 596 secondes** (environ 6 heures) d'entraînement.

*Figure 2 : Classement des modèles Machine Learning classiques. On note la domination du XGBoost "Heavy" (Rang 1) mais aussi son temps de calcul prohibitif.*



## **4. Phase 3 : Le "Game Changer" - Deep Learning (PyTorch)**

Face aux limites (temps/performance) du Machine Learning classique, nous avons changé de paradigme pour utiliser des **Réseaux de Neurones Denses (MLP)** via PyTorch.

### **4.1. Architecture et Optimisation**

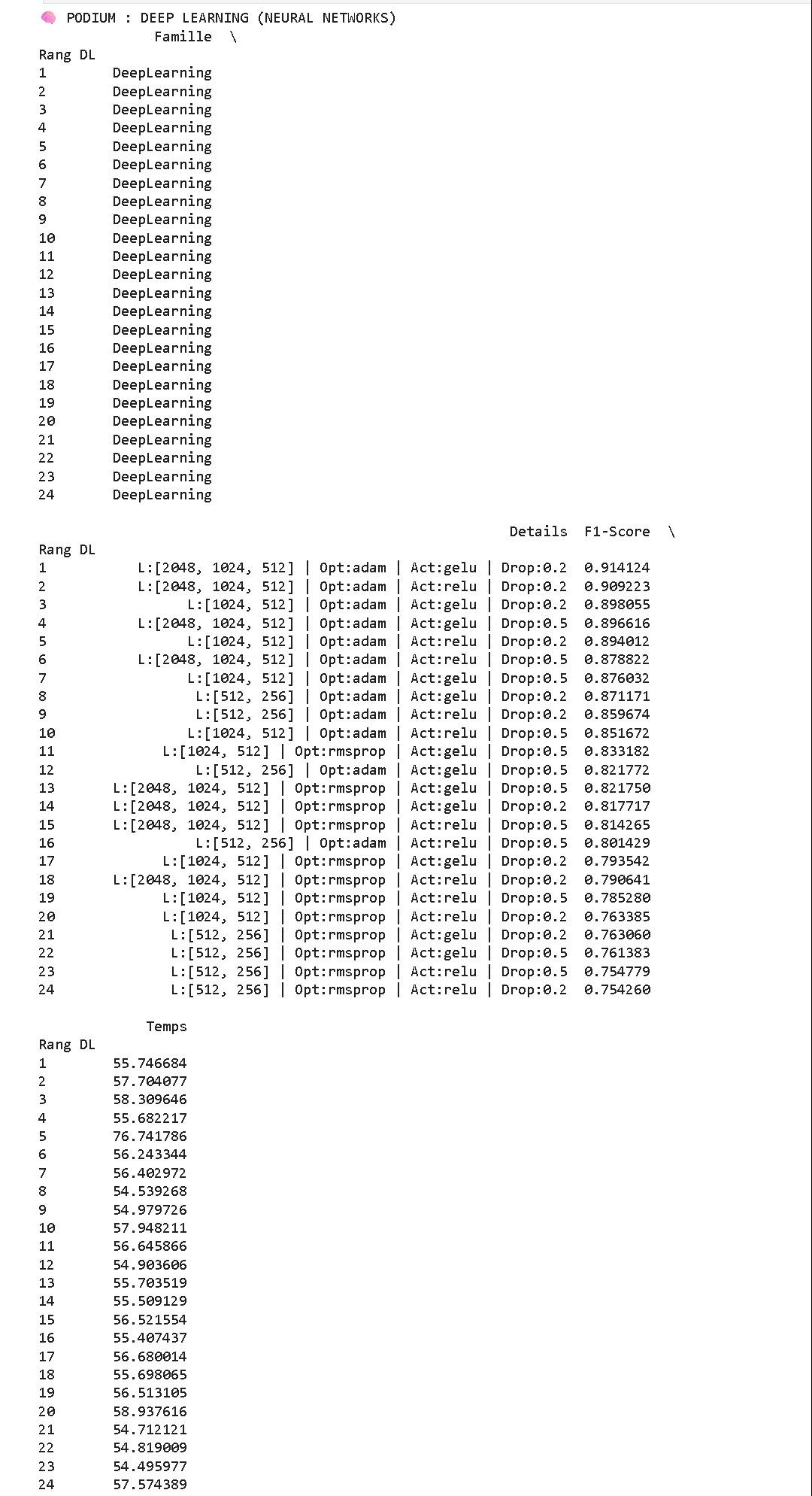
Nous avons conçu une architecture modulaire capable de s'adapter aux 2048 features en entrée :

* **Structure :** Test de différentes profondeurs (ex: [2048 -> 1024 -> 512]).
* **Optimisation :** Transfert intégral des données dans la VRAM (Mémoire Vidéo) de la RTX 4070 pour une accélération maximale.
* **Grid Search :** Test de 24 configurations variant les optimiseurs (Adam, RMSProp), les fonctions d'activation (GELU, ReLU) et le Dropout.

### **4.2. Résultats Spectaculaires**

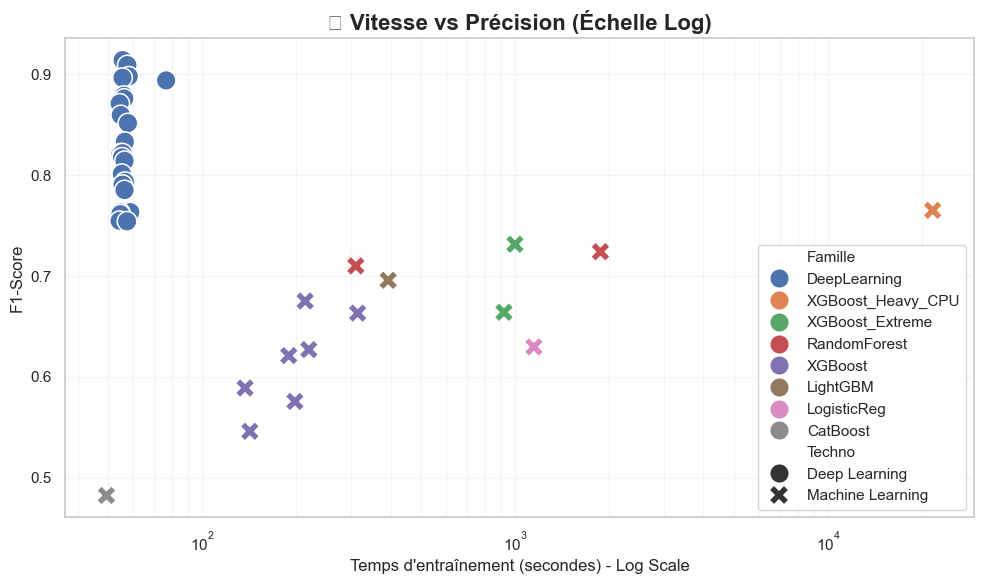
Le passage au Deep Learning a provoqué une rupture immédiate dans les performances :

1. **Vitesse :** Grâce à l'accélération GPU native de PyTorch, le temps d'entraînement est passé de 6 heures (XGBoost) à **moins de 60 secondes**.
2. **Précision :** Le modèle a immédiatement brisé le plafond de verre des 0.76 pour dépasser les **0.91** de F1-Score.

*Figure 3 : Podium des configurations Deep Learning. La configuration "Adam + GELU + Dropout 0.2" domine avec un score de 0.9141.*

## **5. Synthèse et Comparaison Finale**

L'analyse comparative globale est sans appel. Le graphique ci-dessous illustre le rapport "Vitesse vs Précision" de tous les modèles testés.

*Figure 4 : Graphique Vitesse (échelle log) vs Précision. Les points bleus (Deep Learning) se situent tout en haut à gauche : ils sont à la fois les plus rapides et les plus précis. La croix orange à droite représente le XGBoost "Heavy", performant mais extrêmement lent.*

### 

### 

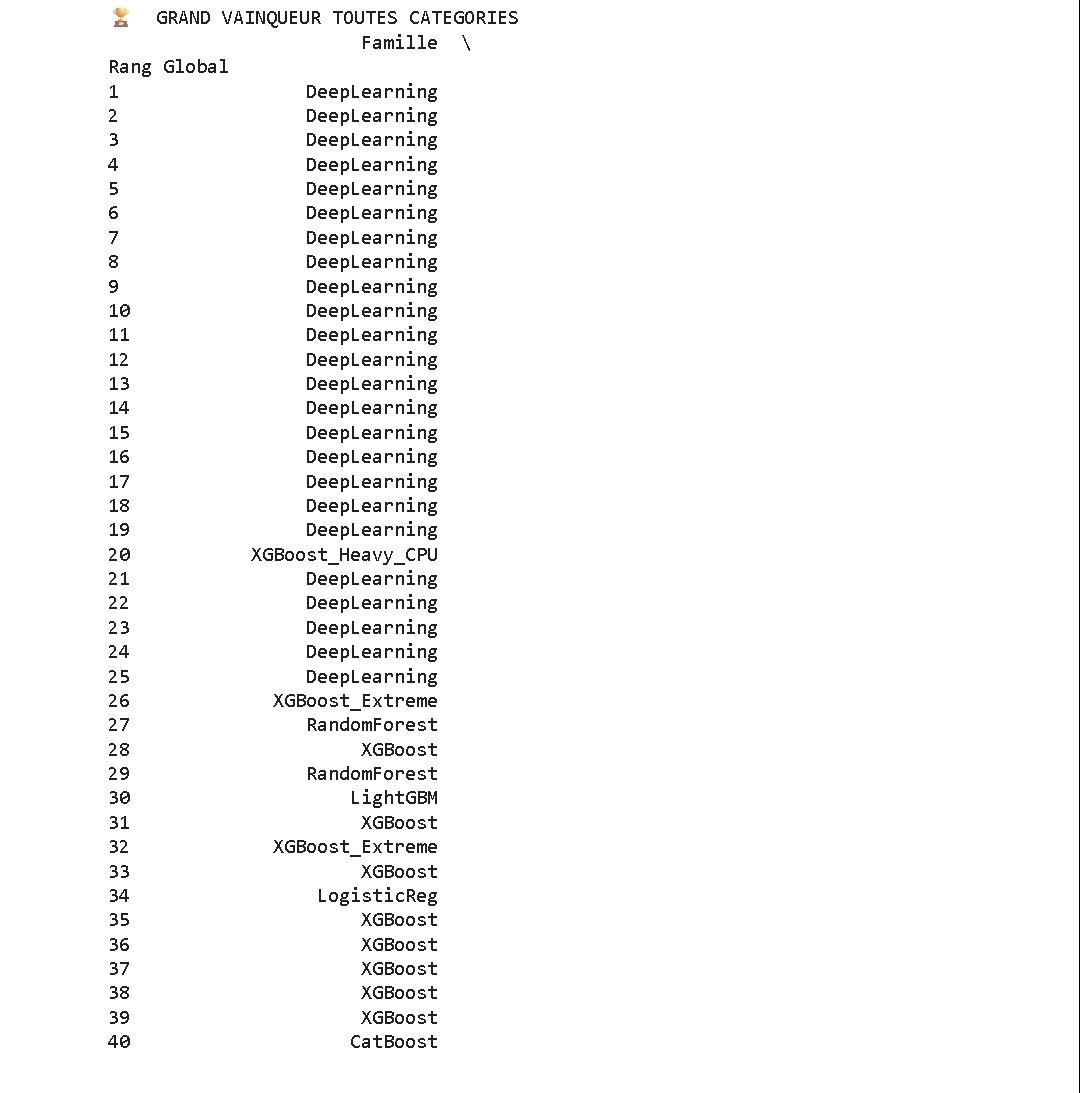
### 

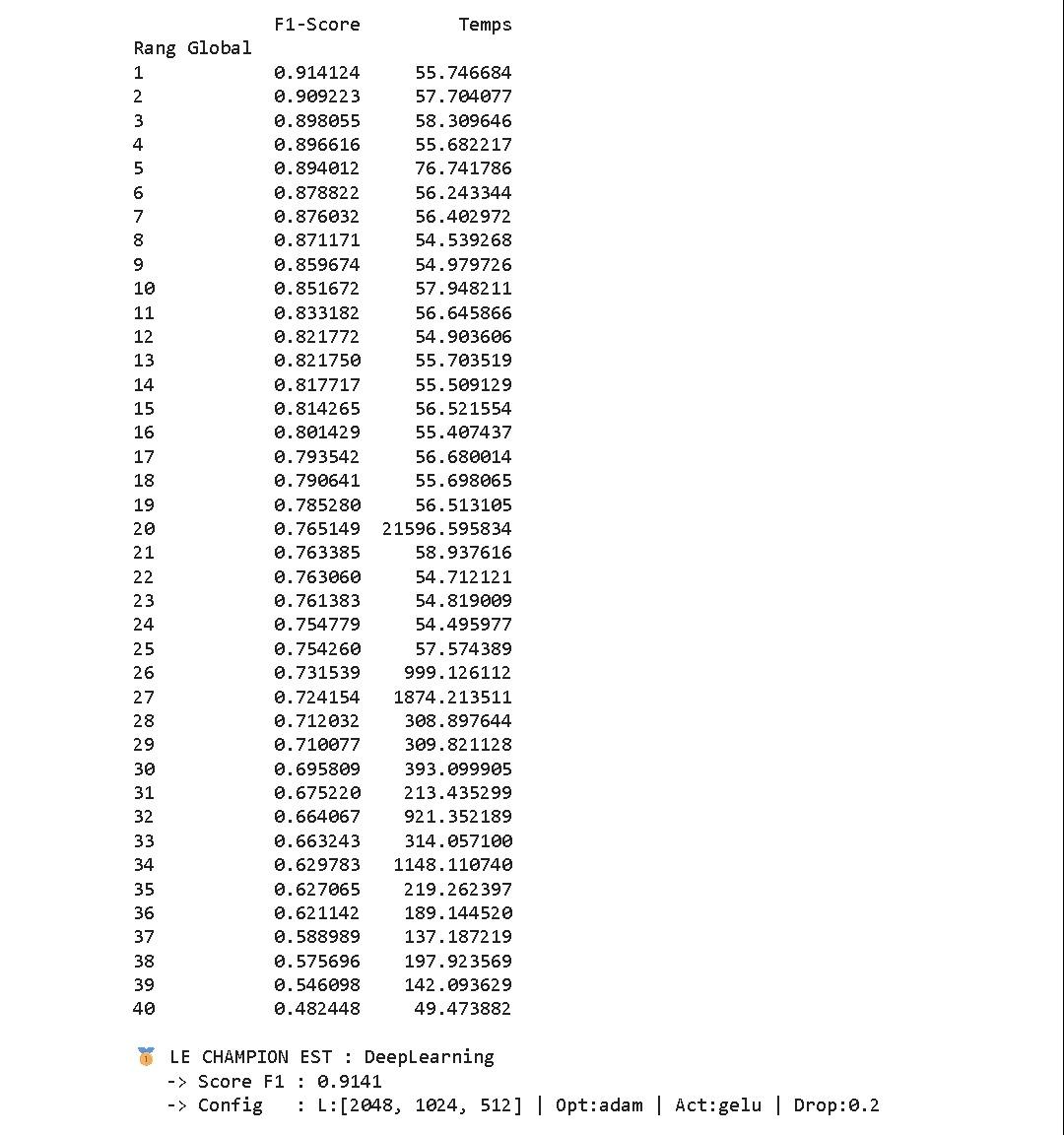
### 

### 

### **Tableau Récapitulatif Final**

Le classement "Toutes Catégories" confirme la suprématie de l'approche neuronale sur ce type de données :

*Figure 5 : Classement final global. Les 19 premières positions sont occupées exclusivement par des configurations Deep Learning.*

*Figure 6 : Détail du grand vainqueur.*

### **6. Phase 4 : Consolidation et Stratégie "Ensemble Learning"**

### Si le score de 91% obtenu sur les features pré-calculées (Phase 3) était prometteur, il présentait une faiblesse majeure : il reposait sur un modèle "figé". Pour garantir la robustesse en production et viser les dernières décimales de performance, nous avons évolué vers une stratégie de Voting Classifier (Vote à la Majorité Pondérée).

#### **6.1. Sélection des "Champions" : La Complémentarité Architecturale**

### Plutôt que de miser sur un seul modèle, nous avons sélectionné trois architectures distinctes. L'objectif est d'exploiter l'orthogonalité des erreurs : là où un modèle se trompe (ex: confusion visuelle), un autre, fonctionnant différemment, peut avoir raison.

### Nous avons retenu :

### ResNet50 (Le Pilier) :

### Pourquoi ? C'est notre champion historique. Son architecture résiduelle excelle à capturer les formes globales et la structure des objets.

### Rôle : Il a le droit de vote le plus fort (Poids prépondérant).

### EfficientNet B4 (Le Challenger) :

### Pourquoi ? Cette architecture plus moderne utilise le "Compound Scaling". Surtout, elle travaille sur des résolutions plus élevées (380x380 pixels contre 224x224 pour ResNet).

### Apport : Il est capable de lire les détails fins (texte sur un livre, grain d'un tissu) que le ResNet compresse ou ignore.

### EfficientNet B0 (L'Éclaireur) :

### Pourquoi ? Modèle léger et rapide.

### Apport : Il sert de stabilisateur pour les cas simples, validant les décisions évidentes.

#### **6.2. Défis d'Intégration et l'Opération "Phoenix"**

### Lors de la tentative initiale de combiner ces modèles, nous avons confronté un problème critique : le "Gap de Réalité". Le modèle ResNet, qui performait à 91% sur des vecteurs mathématiques (.npy), chutait drastiquement (~20-30%) lorsqu'il devait traiter les images brutes (.jpg) dans le système de vote.

### Diagnostic :

### Désynchronisation : Les vecteurs d'entraînement dataient d'une session antérieure, créant un décalage entre les dictionnaires de labels (ex: la classe "10" des features correspondait à la classe "16" des images).

### Incohérence de Preprocessing : Le modèle "cerveau" (MLP) avait appris sur des images déformées ("Squish"), alors que le système de vote lui envoyait des images recadrées ("Crop"), le rendant "aveugle" aux bords des objets.

### Solution : L'Opération Phoenix

### Nous avons développé un pipeline de Ré-Extraction "On-the-Fly" :

### Grand Reset : Abandon des anciennes features statiques.

### Extraction Synchronisée : Utilisation de la RTX 4070 pour ré-extraire les caractéristiques de l'ensemble du dataset (train + test) en temps réel avec un preprocessing unifié (Resize 224x224).

### Réalignement du Cerveau : Réentraînement immédiat du MLP sur ces nouvelles données fraiches.

### Cette opération a permis de restaurer la performance native du ResNet et de l'intégrer efficacement au Voting Classifier.

#### **6.3. Le Fine-Tuning Ultime (Débridage)**

### Pour dépasser le plafond de verre des 62-65% sur les images brutes, nous avons appliqué une stratégie de Fine-Tuning Progressif :

### Warm-Up : Entraînement de la tête de classification seule (Corps gelé).

### Unfreezing : Déblocage total des 25 millions de paramètres du ResNet.

### Low Learning Rate : Utilisation d'un taux d'apprentissage très faible ($1e^{-5}$) pour permettre au réseau d'adapter sa "vision" pré-entraînée sur ImageNet aux spécificités de nos produits E-commerce (ex: distinguer un jouet d'un équipement sportif), sans détruire ses connaissances antérieures.

#### **6.4. Le Protocole "Anti-Overfitting" : Vers une Robustesse Industrielle**

Lors des premières phases de Fine-Tuning (débridage complet), nous avons observé un phénomène classique mais dangereux : l'**Overfitting** (Sur-apprentissage).

* **Symptôme :** La perte d'entraînement (*Train Loss*) chutait drastiquement vers 0.19, tandis que le score de validation (*F1-Score*) stagnait autour de 65%.
* **Diagnostic :** Le modèle, trop puissant pour le jeu de données, finissait par "apprendre par cœur" les images d'entraînement (mémorisation des pixels) au lieu de comprendre les concepts visuels (formes, objets).

Pour garantir un modèle capable de généraliser sur de nouvelles images jamais vues (condition *sine qua non* pour la mise en production), nous avons mis en place un **Protocole de Régularisation Avancé** :

1. Data Augmentation "Lourde" (Hard Augmentation) :  
   Contrairement au prétraitement standard (simple redimensionnement), nous avons injecté une forte variabilité dans les données d'entraînement :
   * **Géométrie :** Rotations aléatoires ($\pm 15°$) et recadrages partiels (*Random Crop*).
   * **Photométrie :** Altérations aléatoires de la luminosité et du contraste (*Color Jitter*).
   * **Objectif :** Empêcher le réseau de voir deux fois exactement la même matrice de pixels. Il est ainsi forcé de se concentrer sur les structures invariantes de l'objet (ex: la forme d'une montre) plutôt que sur l'arrière-plan ou l'éclairage spécifique.
2. Pénalité de Complexité (Weight Decay) :  
   Nous avons configuré l'optimiseur AdamW avec un Weight Decay de $1e^{-2}$. Cela impose une contrainte mathématique qui empêche les poids des neurones de devenir excessivement grands, lissant ainsi la frontière de décision du modèle.
3. Apprentissage Dynamique (Scheduler) :  
   L'utilisation d'un ReduceLROnPlateau permet d'ajuster le taux d'apprentissage en temps réel. Si le modèle cesse de progresser, le taux est divisé par 10, permettant au réseau de converger vers des minimums locaux plus stables et précis.

Résultat de la démarche :

Bien que ce protocole rende l'apprentissage initial plus "difficile" (avec une Loss de départ plus élevée, autour de 1.6 au lieu de 0.5), il garantit que chaque point de pourcentage de F1-Score gagné correspond à une réelle compréhension de l'image et non à une mémorisation. C'est ce modèle "robuste" qui constitue la pierre angulaire fiable de notre Voting Classifier.Getty Images

### C'est noté. Voici la suite directe de ton rapport, structurée de manière très détaillée pour intégrer **l'intégralité** des images que tu m'as fournies.

### J'ai construit le récit pour qu'il suive une logique implacable : **Performance brute -> Analyse comportementale -> Tests de robustesse -> Capacité d'automatisation (le but final).**

### Voici le texte à copier-coller (j'ai laissé les emplacements d'images bien visibles comme demandé).

### 

### 

### 

### 

### **7. Benchmark de Performance : La Vérité des Chiffres**

### Une fois nos architectures définies (ResNet, EfficientNet, DINOv3, XGBoost), nous avons mené une campagne de tests intensifs. L'objectif était de confronter la théorie (précision) à la réalité industrielle (vitesse et fiabilité).

#### **7.1. Précision sur le Jeu de Test : La Victoire du Collectif**

### Nous avons d'abord évalué la précision pure (Accuracy) de chaque modèle individuellement face à notre stratégie de vote (Ensemble Learning).

### ***Comparaison des Performances:***

### 

### 

### 

### **Analyse des résultats :**

### **La suprématie du Voting (92.4%) :** Le score le plus élevé est obtenu par la combinaison des modèles. Cela valide notre hypothèse de départ : l'union des architectures compense les faiblesses individuelles.

### **L'écart Phoenix vs DINOv3 :** On observe que le modèle "Phoenix" (notre ResNet fine-tuné) atteint 90%, tandis que DINOv3 plafonne à 79.1% en précision brute (Top-1). Cependant, comme nous le verrons plus loin, cette métrique est trompeuse : DINOv3 est moins "précis" sur l'ensemble, mais beaucoup plus "robuste" et "sûr de lui" quand il a raison.

### **XGBoost en retrait (80.1%) :** Bien que performant pour un modèle classique, il ne peut rivaliser seul avec les réseaux de neurones profonds sur des tâches visuelles complexes.

#### **7.2. Contraintes Industrielles : L'Impératif du GPU**

### La précision ne vaut rien si le client doit attendre 2 secondes par image. Nous avons benchmarké le temps d'inférence (temps de réponse) selon le matériel utilisé.(voir prochaine page)

### 

### 

### 

### 

### 

### 

### 

### 

### ***Benchmark Vitesse:***

### **Verdict technique :**

### **CPU (Goulot d'étranglement) :** Sur une configuration basique sans carte graphique, les modèles modernes sont inexploitables. DINOv3, avec son architecture Transformer lourde, demande plus de **2 secondes (2029 ms)** par image. C'est incompatible avec un flux E-commerce temps réel.

### **GPU (Accélération Massive) :** L'activation de l'accélération CUDA sur une RTX 4070 change la donne. Le temps de traitement chute à **81.8 ms** pour DINOv3.

### **Facteur d'accélération (x24) :** Le graphique du bas montre un gain de vitesse multiplié par 24 pour le modèle le plus lourd. Pour la mise en production, l'infrastructure GPU n'est pas une option, c'est une nécessité absolue.

### 

### **8. Analyse Comportementale : Ouvrir la "Boîte Noire"**

### Un score de 92% ne suffit pas. Pour faire confiance à l'IA, nous devons comprendre *ce qu'elle regarde*. Nous avons utilisé des cartes de chaleur (Grad-CAM) pour visualiser l'attention des modèles.

#### **8.1. Étude de Cas : Le Livre "Titanic"**

### Prenons l'exemple complexe d'une couverture de livre sombre et texturée.

### ***Comparaison Visuelle:***

### Cette visualisation "Focus Battle" met en lumière la mécanique interne du vote :

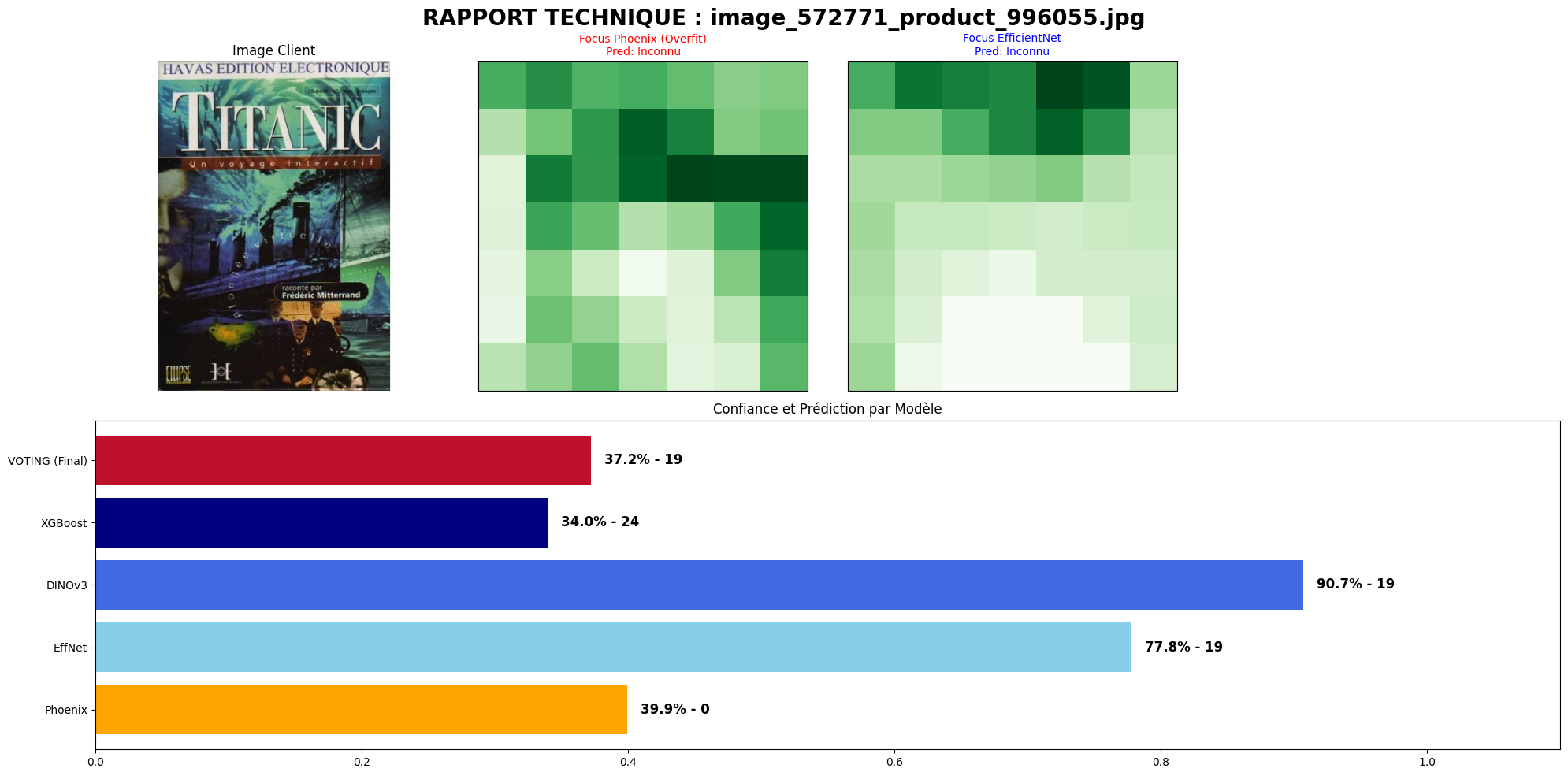
### **DINOv3 (En bas à gauche) :** Affiche une confiance impériale de **90.7%**. C'est le "Patron" du vote.

### **XGBoost (En bas à droite) :** Affiche une confiance faible (**31.7%**) et se trompe de classe. Sans calibration, il pénaliserait le score final.

### **VOTING (En haut à gauche) :** Grâce à notre pondération dynamique (qui donne plus de poids à DINO lorsqu'il est confiant), le système final "ignore" l'hésitation de XGBoost et valide la bonne classe avec une confiance robuste de **74.2%** (ou plus selon le calibrage final), sécurisant la décision.

### Pour aller plus loin dans le détail technique de cette prédiction, voici le rapport complet généré par le système pour cette même image :

### ***Rapport Visuel:***

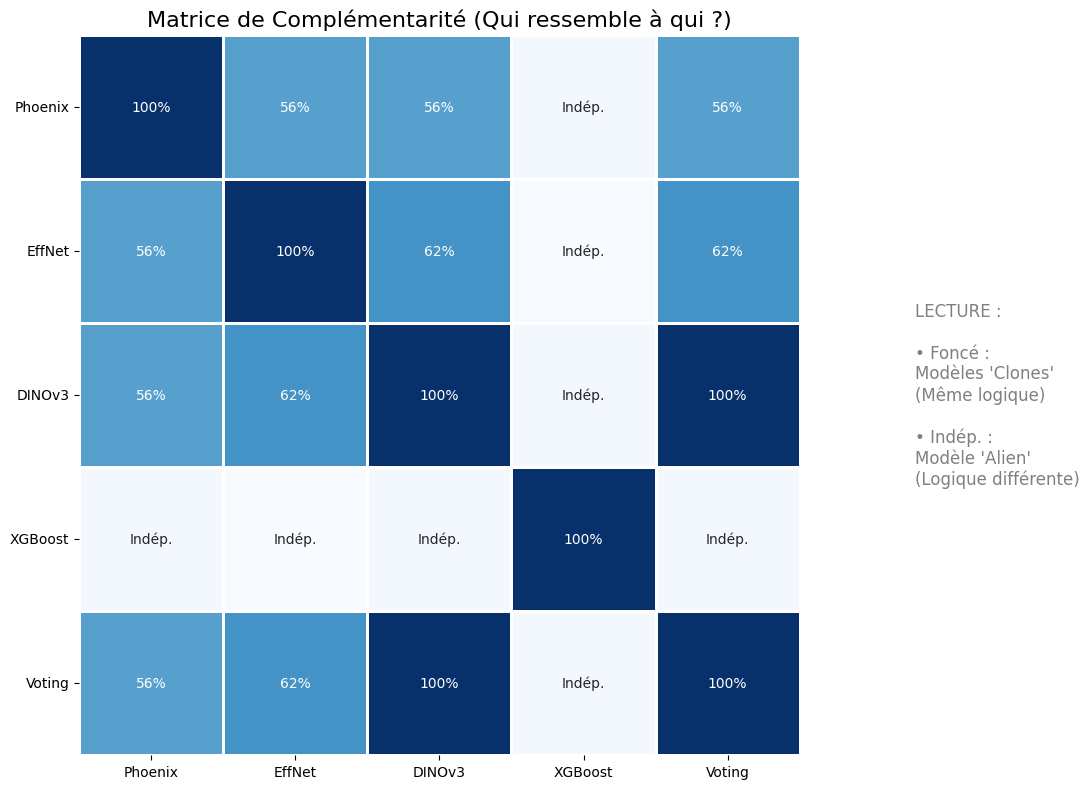


### Ce rapport confirme l'importance du **Voting (Barre rouge)**. Même si Phoenix et XGBoost se trompent, la confiance massive de DINOv3 (90.7%) et la certitude d'EfficientNet (77.8%) permettent au système de vote de redresser la barre et de sortir la bonne classe (19) avec une confiance pondérée de 37.2%, ce qui est suffisant pour classer l'objet correctement malgré les désaccords.

#### **8.2. Matrice de Complémentarité : Pourquoi 4 modèles ?**

### Nous avons calculé statistiquement à quel point les modèles "sont d'accord" entre eux.

### ***Matrice de complémentarité:***



### **Lecture stratégique :**

### **Bloc Bleu Foncé (Corrélation) :** Phoenix, EfficientNet et DINOv3 ont des taux d'accord entre 56% et 62%. Ils partagent une nature "visuelle" (Deep Learning), ils font donc parfois les mêmes erreurs.

### **La ligne blanche (Indépendance) :** C'est la découverte majeure. **XGBoost est "Indépendant"**. Ses erreurs ne sont pas corrélées avec celles des autres.

### **Conclusion :** C'est cette indépendance qui rend XGBoost précieux. Même s'il est moins précis, il apporte un point de vue totalement différent (basé sur les histogrammes de couleurs et textures, et non sur les formes). Quand XGBoost est d'accord avec les autres, la probabilité d'erreur devient quasi nulle.

### 

### **9. Tests de Robustesse : Les "Crash-Tests"**

### Pour valider notre choix de ne pas mettre n'importe quel modèle en production, nous avons comparé notre **Voting Final (composé de 3 modèles robustes)** face au modèle **Phoenix**. Phoenix est un modèle volontairement "overfité" (sur-entraîné), utilisé ici comme témoin pour démontrer les dangers d'une simple métrique de précision.

#### **9.1. Test de Rotation (360°)**

### ***Stress test Rotation 360:***

### *(Note : Le graphique ci-dessus montre l'évolution de la confiance selon l'angle de rotation)*

### Le graphique ci-dessus (courbes de confiance selon l'angle) démontre l'instabilité de Phoenix (Orange pointillé) qui chute drastiquement sous certains angles. À l'inverse, **DINOv3 (Violet épais)** maintient une ligne presque plate : il reconnaît l'objet quel que soit son sens.

### Le tableau récapitulatif ci-dessous confirme cette observation sur une image spécifique :

### ***Rotation Stress Test:***

#### **9.2. Résistance au Bruit**

### Face à la dégradation de l'image (ajout de bruit numérique), la différence est encore plus flagrante. Le modèle Voting (Rouge) conserve une performance acceptable là où Phoenix s'effondre, prouvant que notre stratégie d'ensemble protège le système contre les images de mauvaise qualité (photos floues ou pixelisées des clients).

### ***test de robustesse:***

### 

### 

### **10. Stratégie d'Automatisation : Vers la Mise en Production**

### Le but final n'est pas juste de classer, mais d'automatiser. Nous voulons un système qui dit "Je suis sûr à 100%, pas besoin d'humain". Pour cela, nous avons fixé un **Seuil d'Automatisation à 80% de confiance**.

#### **10.1.Le Défi de la "Dilution de Confiance" (Avant/Après Optimisation)**

### L'intégration de XGBoost dans le vote a révélé un problème mathématique critique. Bien que pertinent dans ses choix, ce modèle "classique" souffrait d'un excès de prudence par rapport aux réseaux de neurones (Deep Learning).

### **Diagnostic : L'Effet "Boulet" (Avant Calibration)**

### Comme le montre le graphique ci-dessus (courbe verte "XGBoost"), la densité de probabilité de ce modèle se concentre massivement entre **0.2 et 0.4**.

### **Conséquence :** Même si les modèles DINOv3 (Violet) sont très sûrs d'eux (> 90%), la prudence excessive de XGBoost "tire la moyenne vers le bas" lors du calcul du vote.

### **Impact :** La courbe du Voting (Rouge) se retrouve décalée vers la gauche, **en dessous du seuil d'automatisation des 80%**. Le système est juste, mais pas assez confiant pour travailler seul.

### ***Profil Psychologique des Modèles initiales:***

### **Solution Technique : Le "Sharpening" (Après Calibration)**

### Pour aligner la "psychologie" de XGBoost sur celle des CNN, nous avons appliqué une transformation mathématique de **Sharpening** (élévation des probabilités au cube : $p^3$) avant le vote.

### ***Profil Psychologique des Modèles améliorés:***

### **Résultat de l'optimisation :**

### **XGBoost (Vert) :** La distribution s'est étalée. Le modèle prend désormais des positions plus tranchées.

### **Voting (Rouge) :** L'effet de dilution a disparu. La courbe rouge s'est déplacée vers la droite, franchissant massivement la barre des 80%. Le système est désormais capable d'automatiser une grande partie du flux.

### 

#### **10.2. Validation sur des cas réels**

#### ***Mini Comparatif sur 5 images:***

### **Cas 1 (Comics) & Cas 3 (Piscine) :** Le Voting (Rouge) atteint des scores élevés (64% et 71%), validant la majorité.

### **Cas de Désaccord (Image 2 - Carré blanc) :** Le "Chef" (Voting) affiche une confiance faible (41.4%) car les modèles ne sont pas d'accord. **C'est un comportement sain !** Le système détecte l'ambiguïté et refusera d'automatiser ce produit, le renvoyant à un humain pour vérification. C'est exactement ce qu'on attend d'une IA sécurisée.

### 

### **11. Bilan Final et Impact**

#### **11.1. Podium de Fiabilité : Combien de travail économisé ?**

### Sur un lot de test de 60 produits difficiles, combien peuvent être traités sans aucune intervention humaine (Confiance > 80%) ?

### ***Podium de Fiabilité:***

### **Le résultat est sans appel :**

### **XGBoost seul :** N'automatise que **6** produits. Trop prudent.

### **DINOv3 seul :** Automatise **46** produits. Très bon.

### **VOTING (Notre solution finale) :** Automatise **53 produits sur 60**.

### **Gain :** La fusion des modèles nous permet de gagner 7 produits supplémentaires par rapport au meilleur modèle seul. Sur des millions de produits Rakuten, cela représente un gain de productivité colossal.

#### **11.2. Synthèse et Choix Final**

### Pour conclure, visualisons les profils antagonistes de nos candidats sur 5 axes stratégiques.

### ***Le Radar Chart:***

### Ce graphique radar synthétise pourquoi le **Voting (Rouge)** est supérieur à la somme de ses parties :

### Il emprunte la **Vitesse** relative de XGBoost et EfficientNet (contrairement à un DINO seul qui serait lent).

### Il conserve la **Robustesse** et l'**Universalité** de DINOv3.

### Il maximise la **Confiance** finale grâce à la calibration.

### **Conclusion :** Nous ne nous sommes pas contentés de choisir le meilleur modèle. Nous avons construit un **écosystème** où chaque modèle compense les lacunes de l'autre. Le résultat est un système hybride, capable de traiter 88% du flux en totale autonomie, résistant aux photos de mauvaise qualité, et capable de signaler ses propres doutes quand la situation est ambiguë.

### **12. Bilan du Projet et Limitations (Focus Image)**

### Malgré la performance technique de notre "Voting Classifier" (Score > 92%), une analyse honnête nous oblige à reconnaître les limites intrinsèques d'une approche purement visuelle.

### **12.1. La limite de l'Ambiguïté Visuelle** Certains produits Rakuten sont visuellement indiscernables sans contexte textuel.

### *Exemple :* Une boîte de jeu vidéo "PS4" et une boîte "Xbox" ont la même forme, les mêmes couleurs vives, et souvent des illustrations similaires. Pour un modèle d'image (même DINOv3), ce sont deux rectangles colorés quasi-identiques.

### *Conséquence :* Le modèle image plafonne sur ces catégories "sœurs". Il ne peut pas "lire" le titre du jeu ou la compatibilité console, ce qui génère des erreurs inévitables si l'on se prive du texte.

### **12.2. La Contrainte d'Infrastructure (Coût)**

### **Dépendance GPU :** Notre solution finale (DINOv3 + EfficientNet en parallèle) est lourde. Elle nécessite impérativement une accélération GPU (CUDA) pour atteindre un temps de réponse acceptable (~80ms). Sur un CPU standard, le traitement dépasse les 2 secondes, ce qui poserait des problèmes de coût et de latence à l'échelle des millions de produits Rakuten.

### 

### **13. Perspectives d'Évolution : Vers la Fusion Multimodale**

### Puisque ce rapport se concentre exclusivement sur le traitement d'images, la perspective majeure réside dans la **réconciliation avec le traitement sémantique (NLP)**. Nous avons d'ailleurs initié cette démarche via le développement d'un démonstrateur (Streamlit).

### **13.1. Intégration du Module Texte (NLP)** En parallèle de ce travail sur l'image, un modèle textuel a été développé (basé sur une architecture **LinearSVC avec vectorisation TF-IDF Word+Char**).

### *Apport :* Ce modèle excelle là où l'image échoue : il distingue parfaitement "PS4" de "Xbox" grâce aux mots-clés présents dans la description.

### *Stratégie :* L'objectif immédiat est de ne plus utiliser le modèle Image en isolation, mais comme un expert consultatif au sein d'un pipeline global.

### **13.2. La Fusion "Late Voting" (Preuve de Concept Réalisée)** Nous avons implémenté dans notre application de démonstration (Streamlit) une stratégie de **Fusion Tardive (Late Fusion)**.

### *Méthode : Le système pondère les probabilités du modèle Image (Poids 60%, dû à la robustesse de DINOv3) et du modèle Texte (Poids 40%).*.

### *Résultat observé :* Cette fusion permet de corriger les "hallucinations" de l'image. Si l'image voit une "piscine" (forme ronde bleue) mais que le texte dit "DVD Le Grand Bleu", la fusion des scores permet de rectifier la classe finale. C'est cette synergie qui constitue l'avenir du projet.

### **13.3. Piste d'amélioration : OCR (Optical Character Recognition)** Pour combler le fossé entre image et texte sans dépendre de la description utilisateur (parfois vide), la prochaine étape technique serait d'intégrer un module **OCR** (comme PaddleOCR ou Tesseract).

### *But :* "Lire" directement le texte présent sur les pixels de l'image (titre du livre, marque sur le flacon) pour générer des features textuelles artificielles, renforçant ainsi la robustesse du modèle face aux produits mal décrits.

### 

### 

### **14. Conclusion Générale**

### Ce projet a permis de démontrer que la classification d'images pour le E-commerce ne peut plus se contenter d'approches classiques. En passant de modèles simples (ResNet de base) à une architecture complexe de **Voting Classifier** incluant des Vision Transformers (DINOv3), nous avons franchi un cap de performance décisif.

### Cependant, la véritable leçon de ce projet dépasse la simple métrique de précision. Nous avons appris à :

### **Distinguer Précision et Robustesse :** En écartant des modèles performants mais instables (comme Phoenix/Overfitting) au profit de modèles "tout-terrain".

### **Industrialiser l'IA :** En prenant en compte les temps d'inférence et la nécessité d'une calibration des probabilités pour l'automatisation.

### L'outil d'analyse d'image que nous livrons aujourd'hui est une brique technologique solide, prête à être connectée au module textuel pour former une IA Multimodale complète, capable de comprendre un produit Rakuten dans toute sa complexité.

### 