Classification des Déchets par Deep Learning
Analyse et Comparaison des Modèles

N'Dje Jules Geraud Odje

4 avril 2025

Table des matières

1 Introduction

Ce rapport présente un projet de deep learning visant à classifier des images de déchets en différentes catégories. Les systèmes automatisés de tri des déchets représentent un enjeu majeur pour améliorer l'efficacité du recyclage et réduire l'impact environnemental. L'utilisation de l'intelligence artificielle, en particulier des réseaux de neurones convolutifs (CNN), offre des solutions prometteuses pour cette tâche.

Notre projet utilise le jeu de données RealWaste, qui contient des images de différents types de déchets dans des conditions réelles. L'objectif est de développer et comparer deux approches de classification :

— Un réseau entièrement connecté (Fully Connected Network)

— Un réseau de neurones convolutif basé sur l'architecture MobileNetV2 (transfer learning)

Ce rapport détaille l'ensemble du processus, de l'exploration des données à l'évaluation des modèles, en passant par l'implémentation des différentes architectures et l'analyse des résultats.

2 Exploration et Préparation des Données

2.1 Structure du jeu de données

Le projet commence par l'exploration du jeu de données RealWaste. Ce jeu de données contient des images de déchets réels dans différentes catégories. Nous commençons par monter le Google Drive et extraire le contenu de l'archive zip.

Les commandes exécutées nous permettent de comprendre la structure des dossiers et d'identifier les différentes catégories de déchets présentes dans le jeu de données. Chaque catégorie correspond à un type spécifique de déchet.

2.2 Distribution des images

Une analyse approfondie du jeu de données nous permet de connaître la distribution des images dans chaque catégorie. Cette étape est cruciale pour comprendre la répartition des données et identifier d'éventuels déséquilibres de classes qui pourraient affecter l'entraînement du modèle.

Pour chaque catégorie, nous comptons le nombre d'images disponibles et calculons le total. Cette information est essentielle pour planifier la division des données en ensembles d'entraînement, de validation et de test.

2.3 Visualisation des échantillons

Avant de procéder à la préparation des données, nous visualisons des échantillons d'images pour chaque catégorie. Cette étape permet de :

— Observer la variabilité des images au sein de chaque catégorie

— Identifier les caractéristiques distinctives des différents types de déchets

— Vérifier la qualité et la résolution des images

— Comprendre les défis potentiels pour la classification (variations d'éclairage, d'angle, etc.)

Cette visualisation nous aide également à déterminer les prétraitements nécessaires et les stratégies d'augmentation de données à mettre en place.

2.4 Division des données

Pour évaluer correctement les performances des modèles, nous divisons le jeu de données en trois ensembles :

— Entraînement (70%) : utilisé pour ajuster les paramètres du modèle

— Validation (15%) : utilisé pour ajuster les hyperparamètres et surveiller l'overfitting

— Test (15%) : utilisé pour l'évaluation finale des performances du modèle

Cette division est réalisée pour chaque catégorie, en veillant à maintenir la même proportion d'images. Les fichiers sont copiés dans des répertoires séparés pour faciliter l'utilisation des générateurs de données de Keras.

2.5 Prétraitement et augmentation de données

Pour améliorer la robustesse des modèles et éviter le surapprentissage, nous mettons en place les techniques suivantes :

— **Redimensionnement** : Toutes les images sont redimensionnées à 224×224 pixels, une taille standard pour de nombreuses architectures CNN.

— **Normalisation** : Les valeurs des pixels sont normalisées entre 0 et 1 en divisant par 255.

— **Augmentation de données** : Pour l'ensemble d'entraînement uniquement, nous appliquons diverses transformations :

  — Rotations (±20°)

  — Translations horizontales et verticales (±20%)

  — Cisaillement (±20%)

  — Zoom (±20%)

  — Retournement horizontal

Ces techniques permettent d'enrichir artificiellement le jeu de données et d'améliorer la capacité de généralisation des modèles en les exposant à des variations des images d'origine.

3 Développement des Modèles

3.1 Modèle entièrement connecté (Fully Connected)

Notre première approche consiste à développer un réseau de neurones entièrement connecté (FCN). Bien que généralement moins performants que les CNN pour les tâches de vision par ordinateur, les FCN servent de base de comparaison.

L'architecture du modèle FCN est la suivante :

— Aplatissement de l'entrée (224×224×3)

— Couche Dense avec 512 neurones et activation ReLU

— Dropout (50%)

— Couche Dense avec 256 neurones et activation ReLU

— Dropout (40%)

— Couche Dense avec 128 neurones et activation ReLU

— Dropout (30%)

— Couche de sortie avec activation softmax (nombre de neurones = nombre de catégories)

Les couches de dropout sont incluses pour réduire le surapprentissage en désactivant aléatoirement une partie des neurones pendant l'entraînement.

3.2 Modèle CNN avec Transfer Learning (MobileNetV2)

Notre deuxième approche utilise le transfer learning avec MobileNetV2, un réseau préentraîné sur ImageNet. MobileNetV2 est choisi pour son bon équilibre entre performance et légèreté.

L'architecture de ce modèle est composée de :

— Base MobileNetV2 préentraînée (avec poids ImageNet)
— Couche GlobalAveragePooling2D
— Couche Dense avec 128 neurones et activation ReLU
— Dropout (30%)
— Couche de sortie avec activation softmax

Pour l'entraînement, nous avons gelé les couches de la base MobileNetV2 pour conserver les caractéristiques générales apprises sur ImageNet, et nous n'avons entraîné que les couches ajoutées pour adapter le modèle à notre tâche spécifique.

3.3 Stratégie d'entraînement

Pour les deux modèles, nous avons utilisé :

— Optimiseur Adam avec un taux d'apprentissage initial de 0.001
— Fonction de perte : entropie croisée catégorielle éparse
— Métrique d'évaluation : précision
— Callbacks pour améliorer l'entraînement :
  — EarlyStopping : arrêt anticipé si la performance ne s'améliore pas
  — ReduceLROnPlateau : réduction du taux d'apprentissage si plateau
  — ModelCheckpoint : sauvegarde du meilleur modèle

Nous avons également expérimenté avec différents optimiseurs (Adam, SGD avec momentum, RMSprop) pour comparer leurs performances.

4 Évaluation et Résultats

4.1 Comparaison des courbes d'apprentissage

Les courbes d'apprentissage fournissent des informations cruciales sur le comportement des modèles pendant l'entraînement. Nous avons visualisé :

— L'évolution de la précision sur les ensembles d'entraînement et de validation
— L'évolution de la perte (loss) sur les ensembles d'entraînement et de validation

Ces courbes permettent de détecter le surapprentissage (écart important entre performance d'entraînement et de validation) et d'évaluer si le modèle continue à apprendre ou a atteint un plateau.

4.2 Performances sur l'ensemble de test

L'évaluation finale des modèles est réalisée sur l'ensemble de test, qui contient des données jamais vues pendant l'entraînement. Les métriques clés sont :

— Précision globale
— Matrice de confusion normalisée
— Précision, rappel et F1-score par catégorie

La matrice de confusion est particulièrement utile pour identifier les catégories qui posent le plus de difficultés au modèle et comprendre les types d'erreurs commises.

4.3 Comparaison des modèles

Une analyse comparative des performances des deux modèles révèle les points suivants :

— Le modèle CNN basé sur MobileNetV2 surpasse le modèle entièrement connecté

— L'amélioration est significative, démontrant l'importance des architectures convolutives pour les tâches de vision

— La différence de performance est particulièrement marquée pour certaines catégories spécifiques

4.4 Influence des optimiseurs

Les expériences avec différents optimiseurs montrent que :

— Adam offre généralement la convergence la plus rapide et les meilleures performances

— SGD avec momentum nécessite plus d'époques mais peut atteindre des performances similaires

— RMSprop se situe entre les deux en termes de vitesse de convergence

Le choix de l'optimiseur affecte non seulement la vitesse de convergence mais aussi la performance finale du modèle.

5 Analyse des Prédictions

5.1 Visualisation des prédictions

Pour mieux comprendre le comportement des modèles, nous visualisons des exemples de prédictions sur l'ensemble de test. Ces visualisations montrent :

— Des exemples correctement classifiés

— Des exemples incorrectement classifiés

— Les différences de performance entre les deux modèles

Cette analyse qualitative complète l'évaluation quantitative et aide à identifier les cas difficiles et les limitations des modèles.

5.2 Analyse des erreurs

L'analyse des erreurs de classification révèle plusieurs facteurs qui peuvent affecter la performance :

— Similarité visuelle entre certaines catégories

— Variations d'éclairage, d'angle ou d'arrière-plan

— Déséquilibre dans le nombre d'exemples par catégorie

— Présence de plusieurs types de déchets dans une même image

Cette analyse aide à identifier les domaines d'amélioration potentiels pour les futures itérations du modèle.

6 Conclusion et Perspectives

6.1 Principaux enseignements

Ce projet démontre l'efficacité des techniques de deep learning, en particulier des CNN avec transfer learning, pour la classification des déchets. Les principales conclusions sont :

— Les CNN surpassent significativement les réseaux entièrement connectés pour cette tâche
— L'augmentation de données est cruciale pour améliorer la robustesse des modèles
— Le transfer learning permet d'obtenir de bonnes performances même avec un ensemble de données relativement limité
— Certaines catégories de déchets restent difficiles à distinguer, suggérant la nécessité d'approches plus sophistiquées

6.2 Applications potentielles

Les modèles développés pourraient être appliqués dans divers contextes :

— Systèmes automatisés de tri des déchets dans les centres de recyclage
— Applications mobiles éducatives pour sensibiliser au tri des déchets
— Poubelles intelligentes capables de trier automatiquement les déchets
— Systèmes de surveillance pour vérifier la qualité du tri

6.3 Améliorations futures

Plusieurs pistes d'amélioration peuvent être envisagées :

— Augmentation de la taille et de la diversité du jeu de données
— Essai d'autres architectures CNN (EfficientNet, ResNet, etc.)
— Techniques d'apprentissage semi-supervisé pour exploiter des données non étiquetées
— Approches d'ensemble combinant plusieurs modèles
— Détection d'objets pour localiser et identifier plusieurs déchets dans une même image

7 Annexes

7.1 Annexe A : Configuration expérimentale

— Environnement : Google Colab avec GPU
— Framework : TensorFlow 2.x et Keras
— Taille des images : 224×224 pixels
— Taille de batch : 32
— Nombre maximum d'époques : 20 (avec early stopping)

7.2 Annexe B : Exemple de code

```
def build_transfer_learning_model():
    # Chargement de la base MobileNetV2 pr entra n e
    base_model = MobileNetV2(
        weights='imagenet',
        include_top=False,
        input_shape=(img_height, img_width, 3)
    )

    # Gel des couches de la base
    base_model.trainable = False

    # Construction du mod le complet
    model = models.Sequential([
        base_model,
```

```
        layers.GlobalAveragePooling2D(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(num_classes, activation='softmax')
    ])

    return model
```

Listing 1 – Construction du modèle CNN avec transfer learning