Architecture et Composants du Modèle

Ce document fournit un aperçu détaillé de l'architecture de notre modèle et des composants utilisés pour la classification d'images, incluant le fine-tuning, l'extraction de caractéristiques, et les prédictions finales utilisant un modèle d'apprentissage automatique.

1. Configuration de l'Environnement

- Bibliothèques Python : PyTorch, torchvision, scikit-learn, imbalanced-learn, XGBoost, joblib, PIL, numpy, pandas, matplotlib.
- Appareil configuré pour l'entraînement : GPU si disponible, sinon CPU.

2. Gestion des Données Personnalisées

- La classe CustomDataset gère le chargement et la transformation des images pour l'entraînement et le test.
- Les images et les étiquettes sont chargées à partir de fichiers CSV et traitées avec des transformations spécifiées pour augmenter les données pour une meilleure généralisation du modèle.

3. Chargement et Augmentation des Données

- Les ensembles de données d'entraînement, de validation et de test sont créés en utilisant CustomDataset.
- Techniques d'augmentation des données incluent le recadrage aléatoire, le retournement, la rotation, les ajustements de couleur, et l'effacement pour l'ensemble d'entraînement.
- La normalisation et le redimensionnement sont appliqués à l'ensemble de test pour le préparer à l'évaluation du modèle.

4. Architecture du Modèle

- Modèle de base : **EfficientNet B0** pré-entraîné sur ImageNet.
- **Fine-tuning** impliquant le remplacement de la couche classificateur par une nouvelle pour s'adapter au nombre de classes cibles.
- Fine-tuning de couches spécifiques tout en gelant les autres pour tirer parti des caractéristiques pré-apprises tout en s'adaptant à notre ensemble de données spécifique.

5. Processus de Fine-Tuning

- L'entraînement implique la rétropropagation et l'optimisation utilisant AdamW pour mettre à jour les poids.
- La fonction de perte utilisée est CrossEntropyLoss pour la classification multiclasses.
- L'arrêt précoce (Early Stopping) est mis en œuvre pour prévenir le surajustement basé sur la perte de validation.

6. Extraction des Caractéristiques

- Après l'entraînement, les caractéristiques sont extraites de l'ensemble de données en utilisant la base convolutive pré-entraînée d'EfficientNet.
- Les caractéristiques et les étiquettes sont préparées pour l'entraînement avec le modèle d'apprentissage automatique.

7. Intégration de l'Apprentissage Automatique

- Le classificateur **XGBoost** est utilisé pour la prédiction finale, exploitant les caractéristiques extraites du modèle de deep learning.
- La technique de suréchantillonnage ADASYN est appliquée pour aborder le déséquilibre des classes.

- Les hyperparamètres pour XGBoost sont optimisés en utilisant GridSearchCV.

8. Évaluation et Métriques du Modèle

- La performance du modèle est évaluée en utilisant l'exactitude, la précision, le rappel et le score F1.
- Un rapport de classification détaillé fournit des insights sur la performance à travers toutes les classes.

9. Déploiement et Utilisation

- Les modèles (EfficientNet et XGBoost) ainsi que le scaler de données sont sauvegardés sous forme de fichiers .pkl pour un chargement et une inférence faciles.
- Des instructions pour charger les modèles et réaliser des prédictions sur de nouvelles données sont fournies.

10. Exemples de Code et Utilisation

- Des scripts d'exemple pour l'entraînement, le fine-tuning, l'extraction de caractéristiques et l'évaluation sont inclus.
- Des commandes exemples pour exécuter les scripts et charger les modèles sont fournies pour assurer une utilisation aisée.

Détails du Fine-Tuning

- **Niveau de Fine-Tuning** : Seules les couches supérieures de l'EfficientNet sont fine-tunées tandis que les couches antérieures sont gelées. Cette

approche aide à adapter le réseau pré-entraîné à de nouvelles tâches tout en évitant un entraînement extensif.

Techniques de Deep Learning Utilisées

- **Augmentation des Données** : Pour rendre le modèle robuste face à différentes orientations et conditions d'éclairage.
- **Apprentissage par Transfert** : Utilisation d'EfficientNet pré-entraîné sur ImageNet pour exploiter des caractéristiques déjà apprises.
- **Extraction de Caractéristiques** : Utilisation de la base convolutive d'EfficientNet pour extraire des caractéristiques significatives des images.

Techniques d'Apprentissage Automatique

- **XGBoost pour la Prédiction** : Utilisation des techniques de boosting par gradient pour la classification finale basée sur les caractéristiques extraites par le modèle de deep learning.
- **Réglage des Hyperparamètres** : Utilisation de GridSearchCV pour trouver les paramètres optimaux pour le classificateur XGBoost.
- **Gestion du Déséquilibre des Classes** : Mise en œuvre de ADASYN pour un échantillonnage synthétique adaptatif de la classe minoritaire.