# Explication du Code :

Dans cette première partie du code, nous définissons plusieurs classes essentielles qui servent de fondation pour l'entraînement de notre modèle de segmentation dentaire 3D. Chaque classe encapsule des propriétés et des méthodes spécifiques qui facilitent la gestion des données, la configuration du modèle et l'évaluation des performances.

## 1. Classe Dataset

La classe `Dataset` est responsable de la gestion des données d'entrée. Elle permet de charger, prétraiter et fournir les échantillons nécessaires pour l'entraînement et la validation du modèle.

**Attributs :**

- **data_path :** Chemin d'accès aux fichiers de données.
- **labels :** Étiquettes associées aux données pour l'apprentissage supervisé.

**Méthodes :**

- **`__init__` :** Initialise la classe avec les chemins de données et les étiquettes.
- **`__getitem__` :** Effectue le prétraitement des images 3D selon plusieurs étapes :
    - **Chargement des Données :** Lecture du fichier VTK contenant le maillage et extraction des étiquettes de matériau.
    - **Centrage des Points :** Déplacement des points du maillage vers l'origine en soustrayant le centre de masse.
    - **Récupération des Cellules :** Extraction des indices des cellules et formation d'un tableau contenant les coordonnées des cellules.
    - **Calcul des Normales :** Calcul des normales des cellules pour obtenir des informations sur leur orientation.
    - **Normalisation des Données :** Calcul des valeurs maximales, minimales, moyennes et écarts-types pour normaliser les coordonnées.
    - **Formation des Entrées et Étiquettes :** Création de la matrice d'entrées `X` et stockage des étiquettes dans `Y`.
    - **Sélection des Échantillons :** Sélection aléatoire d'un sous-ensemble équilibré de cellules pour le batch d'entraînement.
    - **Calcul des Matrices de Similarité :** Création de matrices de similarité pour aider le modèle à apprendre les relations locales entre les points.
    - **Préparation du Dictionnaire de Sortie :** Rassemblement des cellules, des étiquettes et des matrices de similarité pour l'entraînement.

## 2. Classe MeshSegNet

La classe `Model` encapsule la définition du modèle de réseau de neurones utilisé pour la segmentation.

**Attributs :**

- **architecture :** Décrit la structure du modèle (couches, fonctions d'activation, etc.).
- **optimizer :** Algorithme utilisé pour l'optimisation (par exemple, Adam ou SGD).

**Méthodes :**

- **`build_model` :** Construit le modèle en fonction de l'architecture définie.
- **`compile_model` :** Compile le modèle avec la fonction de perte et les métriques appropriées.

## 3. Fonctions de Métriques

Les fonctions suivantes sont utilisées pour évaluer les performances des modèles de segmentation :

- **`weighting_DSC` :** Calcule le Dice Score Coefficient (DSC) pondéré pour chaque classe.
- **`weighting_SEN` :** Calcule la sensibilité (ou rappel) pondérée pour chaque classe.
- **`weighting_PPV` :** Calcule la valeur prédictive positive (PPV) pondérée pour chaque classe.
- **`Generalized_Dice_Loss` :** Calcule la perte de Dice généralisée.
- **`DSC`, `SEN`, `PPV` :** Calculent respectivement le Dice Score, la sensibilité et la valeur prédictive positive pour chaque classe.

## 4. Étapes de Prétraitement des Données

Étant donné que nous disposons de seulement 20 échantillons de fichiers `.vtp`, nous avons opté pour l'augmentation de données, enrichissant notre jeu de données de maillages 3D.

**Types de Transformations Appliquées :**

- **Rotations :** Appliquées aléatoirement autour des axes X, Y et Z.
- **Translations :** Déplacement aléatoire le long des axes X, Y et Z.
- **Mises à l'Échelle :** Transformation d'échelle selon les axes X, Y et Z.

## 5. Split Data

L'étape de partitionnement des données prépare les échantillons augmentés pour l'entraînement, la validation et le test du modèle.

**Processus :**

- **Initialisation :** Définition des chemins pour les données d'augmentation et création des répertoires si nécessaire.
- **Échantillons Valides :** Création d'une liste d'échantillons existants.
- **K-Fold :** Utilisation de la validation croisée avec K-Fold pour diviser les échantillons.

- **Création des Listes :** Génération des noms de fichiers correspondants et enregistrement dans des fichiers CSV.

## 6. Entraînement du Modèle

Le processus d'entraînement inclut :

**Étapes :**

- **Configuration de l'environnement :** Détection de la disponibilité du GPU.
- **Chargement des données :** Création des jeux de données et des DataLoaders.
- **Définition du modèle :** Initialisation du modèle MeshSegNet et de l'optimiseur Adam.
- **Chargement des checkpoints :** Vérification des checkpoints précédents.
- **Entraînement :** Boucle sur les époques, mise à jour des poids et affichage des statistiques.
- **Validation :** Évaluation sur l'ensemble de validation.
- **Sauvegarde :** Enregistrement des poids et des métriques.

## 7. Test du Modèle

Pour chaque image de test, une étape de test est essentielle. Nous générons des images segmentées en utilisant le modèle déjà entraîné.

**Processus :**

- **Prétraitement :** Chargement et prétraitement du maillage (centrage, calcul des normales, normalisation).
- **Calcul des Matrices de Similarité :** Construction des matrices `A_S` et `A_L` pour segmenter le maillage.
- **Préparation des Données :** Transformation des données en tenseurs.
- **Prédiction :** Exécution du modèle pour obtenir les probabilités de segmentation.
- **Conversion :** Transformation des probabilités en étiquettes de classe.
- **Sauvegarde des Résultats :** Création d'un nouveau maillage avec les étiquettes prédictives et sauvegarde.