Projet

Contexte : Vous disposez d'un prototype fonctionnel d'analyse de sentiment multimodal (texte, audio, vision). Ce prototype utilise des méthodes de *fusion précoce* et des encodages unimodaux simplifiés (Mean Pooling, MFCC Mean).

Objectif Principal : Transformer ce prototype en une architecture d'état de l'art, en vous inspirant des principes de l'article **DEVA (Deep Enhanced Vision-Audio-Text Alignment)**. L'objectif est de passer d'une fusion non-interactive (concaténation) à une fusion **Text-Guided** par des mécanismes d'Attention Croisée.

Les résultats finaux devront être évalués en utilisant les métriques académiques standards (**Acc-2, F1, MAE, Corr**)

II. Le Code de Base : Limites à Adresser

Votre code de départ, bien que fonctionnel, présente trois faiblesses majeures que l'architecture DEVA vise à corriger :

Domaine	**Code Actuel (Prototype)**	**Problème**	**Solution Requise (DEVA)**
Encodage Texte (X_t)	bert.last_hidden_state.mean(dim=1)	Perte de la structure séquentielle.	Utiliser un **Transformer Encoder** pour créer une séquence X_t riche de $T=8$ tokens.
Richesse Sémantique	MFCC Mean, Mini-CNN	Manque de descripteurs émotionnels fins.	Introduire des **Descriptions Émotionnelles Textuelles** (D_a et D_v) encodées par BERT.
Fusion	torch.cat([t, a, v], dim=-1)	Fusion non-interactive (les modalités ne s'influencent pas).	Utiliser le mécanisme d'**Attention Croisée** (MFU) guidé par le texte.

III. Étapes Détaillées d'Amélioration

Vous devez suivre ces cinq étapes pour mettre à niveau l'architecture.

Étape 1 : Amélioration de l'Encodage Textuel (X_t)

Le texte est la modalité centrale qui doit guider la fusion.

1. **Modification de l'encodage :** Au lieu de prendre la moyenne de la séquence BERT, vous devez implémenter la classe **TextEncoder**.
2. **Mécanisme Requis :** Le TextEncoder doit prendre la séquence brute de BERT, lui ajouter un **token spécial E_m** (un vecteur learnable ou initialisé à zéro) en tête, appliquer un **nn.TransformerEncoderLayer** sur la séquence complète, puis ne conserver que les **$T=8$ premiers tokens** pour former la séquence enrichie X_t.

Étapes 2 & 3 : Intégration des Descriptions Émotionnelles (D_a et D_v)

L'article DEVA exige l'ajout d'une couche sémantique aux modalités Audio et Vision.

1. **Audio Emotional Description (AED - D_a) :**
    - Implémentez la fonction **extract_audio_features(audio)** (pour Pitch, Loudness, Jitter, Shimmer, même si simplifiés).
    - Implémentez la fonction **audio_description(...)** qui convertit ces 4 valeurs en une phrase textuelle descriptive (e.g., "The speaker used low pitch...").
    - **Encodage D_a :** Cette phrase doit être encodée en utilisant votre TextEncoder pour obtenir un embedding D_a.
2. **Visual Emotional Description (VED - D_v) :**
    - Implémentez la fonction **visual_description(aus)** qui prend une liste d'Action Units (AUs simulées) et génère une phrase descriptive (e.g., "The person shows signs of: raised inner brow...").
    - **Encodage D_v :** Cette phrase doit être encodée de la même manière pour obtenir un embedding D_v.

Étape 4 : Le Cœur de la Fusion (MFU et Attention Croisée)

Le texte (X_t) doit maintenant activement interagir avec X_a et X_v pour les enrichir.

1. **CrossModalAttention :** Créez la classe **CrossModalAttention(nn.Module)** pour implémenter l'Attention Croisée Scaled Dot-Product.

2. **MFU (Minor Fusion Unit) :** Créez la classe **MFU(nn.Module)** qui implémente la fusion résiduelle text-guided.
    - Elle utilise deux modules CrossModalAttention : un pour $T \to A$ et un pour $T \to V$.
    - Sa formule clé est : $\text{Output} = \text{prev\_fusion} + \alpha \cdot \text{Att}_{T \to A}(X_a, X_t) + \beta \cdot \text{Att}_{T \to V}(X_v, X_t)$
    - α et β doivent être des **nn.Parameter** (scalaires) learnables.
3. **Alignement :** Utilisez la classe **AudioVisualFeatureProjector** pour transformer les embeddings de prototype en séquences alignées X_a et X_v de taille **$[B, T, D]$** avant la fusion.

Étape 5 : Assemblage Final et Métriques Académiques

1. **DEVANet Final :** Créez la classe principale **DEVANet(nn.Module)** qui orchestre le flux complet, intégrant tous les composants.
2. **Fonction d'Évaluation :** Implémentez la fonction **evaluate_metrics(labels, preds)** qui doit calculer et retourner les quatre métriques exigées pour la publication :
    - **Acc-2** (Accuracy Binaire : Positif vs. Négatif, seuil à 0)
    - **F1-score** (Binaire, average='weighted')
    - **MAE** (Mean Absolute Error, régression)
    - **Corrélation de Pearson (Corr)** (régression)

IV. Livrables et Critères de Succès

Livrables Attendus

1. Un **Notebook Colab** contenant l'intégralité du code final.
2. La classe **DEVANet** complètement implémentée, utilisant les composants d'attention croisée.

Critères de Succès

- Le code s'exécute sans erreur avec la nouvelle boucle d'entraînement.
- Le modèle intègre clairement les concepts de **séquence X_t**, **descriptions textuelles (D_a, D_v)**, et **fusion par MFU**.

- Le notebook doit afficher les résultats finaux de la fonction **evaluate_metrics** (Acc-2, F1, MAE, Corr), démontrant l'adoption des standards d'évaluation académique.