🎧 Sujet global — Speech-To-Text, Audio Deep Learning & Agents Vocaux

2 sections : TP + Projet/Hackathon

🔹 SECTION 1 — Travaux Pratiques (TP)

🗓️ Jour 1 — Architecture From Scratch : de MFCC à Transformers

> Objectif : comprendre et construire un pipeline STT de A à Z, en explorant plusieurs architectures.

🚀 Partie 1 — MLP + MFCC + CTC

1. Implémenter un script Python utilisant **Keras** ou **PyTorch**.
2. Pipeline minimal :
    - Extraction des **MFCC** (TorchAudio ou Librosa).
    - Encodage des transcriptions caractère par caractère.
    - Architecture **MLP** simple.
    - Loss : **Connectionist Temporal Classification (CTC)**.
3. Ressources :
    - MFCC vs MelSpec : https://vtiya.medium.com/mfcc-vs-mel-spectrogram-8f1dc0abbc62
    - Exemple Keras CTC ASR : https://keras.io/examples/audio/ctc_asr/
    - Comprendre la CTC : https://distill.pub/2017/ctc/

      https://harald-scheidl.medium.com/intuitively-understanding-connectionist-tem

poral-classification-3797e43a86c

- HF Audio CTC : https://huggingface.co/learn/audio-course/chapter3/ctc

---

Partie 2 — CNN + Spectrogrammes

1. Remplacer les MFCC par un **Spectrogramme** (ou MelSpectrogram).
2. Ajouter des **couches convolutionnelles** pour extraire les features.
3. Comparer performances / stabilité / convergence.

Partie 3 — RNN (LSTM / GRU / BiLSTM)

- Remplacer le MLP ou la tête intermédiaire par :
  - **LSTM**
  - **GRU**
  - **Bi-LSTM**
- Étudier l'impact sur :
  - la capacité temporelle
  - la stabilité de la CTC
  - la vitesse d'entraînement

Partie 4 — Approche Transformers

- Implémenter une variante Transformer pour ASR :
  - Self-attention sur frames audio
  - Positional encoding

- Tutoriel Keras recommandé :
  https://keras.io/examples/audio/transformer_asr/

Partie 5 — Hyper-paramètres (Grid / Optuna / Hyperopt)

Objectif : construire un script complet de tuning

Hyper-paramètres à explorer :

- nb de couches
- nb de neurones
- type d'extracteur (MFCC vs MelSpec)
- learning rate
- augmentation audio

Grid Search CV :

- https://keras.io/keras_tuner/api/tuners/grid/
- https://medium.com/@4AInsights/hyperparameter-tuning-with-keras-and-gridsearchcv-a-comprehensive-guide-46214cc0d999
- https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

Optuna : https://github.com/optuna/optuna
Hyperopt : https://github.com/hyperopt/hyperopt

🗓️ Jour 2 — Fine-tuning de modèles pré-entraînés (HuggingFace)

Objectif :

Fine-tuner un modèle STT state-of-the-art sur vos données.

Modèles recommandés :

- **wav2vec 2.0**
- **HuBERT**
- **Whisper**
- **Faster-Whisper**
- **Insanely-Fast-Whisper**

Tutoriels officiels :

- HF Audio Course :
  https://huggingface.co/learn/audio-course/chapter5/asr_models
  https://huggingface.co/learn/audio-course/fr/chapter5/fine-tuning
- Fine-tuning wav2vec2 :
  https://huggingface.co/blog/fine-tune-wav2vec2-english
- Fine-tuning Whisper :
  https://huggingface.co/blog/fine-tune-whisper
- Faster-Whisper :
  https://github.com/SYSTRAN/faster-whisper
- Insanely Fast Whisper :
  https://github.com/Vaibhavs10/insanely-fast-whisper

🔷 SECTION 2 — Projet / Hackathon

🎤 Construire un Agent Vocal Téléphonique IA

🎯 Objectif :

Créer un agent vocal capable de :

- répondre au téléphone
- écouter en streaming

- transcrire l’appel
- répondre avec un modèle IA
- parler grâce à un TTS temps réel

🔧 Outils possibles :

OpenAI Realtime API & Agents

- https://openai.github.io/openai-agents-python/
- https://github.com/twilio-samples/speech-assistant-openai-realtime-api-python/blob/main/main.py
- https://www.twilio.com/code-exchange/ai-voice-assistant-openai-realtime-api
- https://www.twilio.com/en-us/blog/voice-ai-assistant-openai-realtime-api-python
- JS Extensions :
  https://openai.github.io/openai-agents-js/extensions/twilio/

Twilio (voix)

- Très simple mais **qualité audio faible**
- Débugger en enregistrant le flux brut (PCMU μ-law → PCM16)

Alternatives à tester :

- WhatsApp Cloud API (via VoIP)
- Google Meet
- Jambonz + DID Logic (fournisseur de numéros)
  → meilleure qualité audio & plus de contrôle sur le streaming

---

🔬 Extensions avancées possibles :

🔹 Voice cloning & chant

- RVC :
  https://huggingface.co/spaces/Clebersla/RVC_V2_Huggingface_Version
- XTTS-v2 :
  https://huggingface.co/coqui/XTTS-v2
- OpenVoice :
  https://huggingface.co/myshell-ai/OpenVoice
- LLASA TTS :
  https://huggingface.co/blog/srinivasbilla/llasa-tts

🔹 Moshi (Kyutai)

- https://github.com/kyutai-labs/moshi
- https://huggingface.co/collections/kyutai/moshi-v01-release

---

🗂️ Annexes — Améliorations STT avancées

🔸 Speaker Diarization

→ détecter les locuteurs
→ filtrer par locuteur

Datasets : LibriSpeech
Librairies : Torchaudio, SpeechBrain

🔸 Speaker Identification (SI)

→ embeddings, comparaison cosinus, seuils personnalisés

🔸 Voice Activity Detection (VAD)

Dataset Mulan
Torchaudio :
https://pytorch.org/audio/stable/tutorials/audio_data_augmentation_tutorial.html

---

🔸 Noise Reduction

- Facebook Denoiser
  https://github.com/facebookresearch/denoiser

- SepFormer SpeechBrain
  https://huggingface.co/speechbrain/sepformer-wham-enhancement

---

🔸 Emotion Recognition

- https://huggingface.co/speechbrain/emotion-recognition-wav2vec2-IEMOCAP

- Notebook :

  https://colab.research.google.com/github/m3hrdadfi/soxan/blob/main/notebooks/Emotion_recognition_in_Greek_speech_using_Wav2Vec2.ipynb

- IEMOCAP dataset :
  https://zenodo.org/records/1478765

🔸 Filtrage de voix (séparation)

- Looking to Listen (Google)
  https://blog.research.google/2018/04/looking-to-listen-audio-visual-speech.html

- VoiceFilter
  https://google.github.io/speaker-id/publications/VoiceFilter/

- Sound Separation
  https://github.com/google-research/sound-separation

📌 À retenir pour tous les TP

Toujours se poser :

- **Quelles données je manipule ?** (MFCC, MelSpec, audio brut)
- **Quel est mon input ?**
- **Quel est mon output ?**
- **Quelle loss ?** (CTC souvent)
- **Quelle architecture ?** (MLP, CNN, RNN, Transformer, wav2vec…)