Compréhension du Langage Audio

Chapitre 3 : Architectures Neuronales pour les Langues Africaines

Master - Traitement Automatique du Langage Naturel Avancé

Fréjus A. A. Laleye

7 février 2026

Plan du cours

Du Texte à la Parole : Le Défi de la Continuité

Rappel des chapitres précédents

- Chapitre 1 : Architecture Transformer, self-attention, multi-head attention
- Chapitre 2 : Tokenisation BPE, embeddings — données **discrètes**

Le défi de l'audio

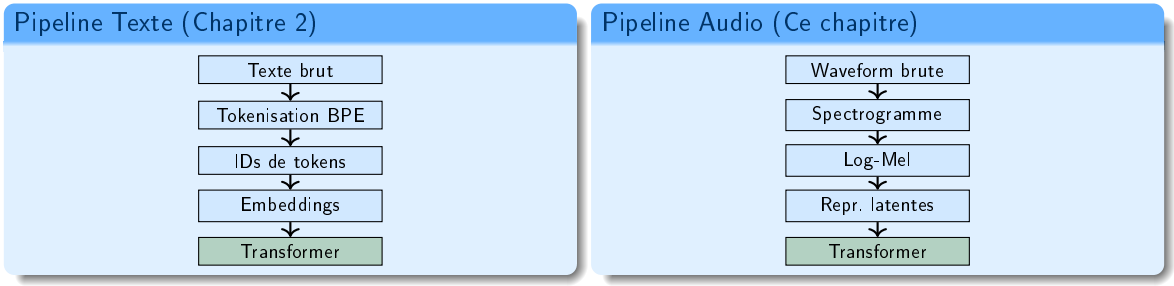
La parole est un signal **continu** échantillonné à haute fréquence :

- 16 000 échantillons/seconde (vs quelques dizaines de tokens textuels)
- Comment adapter les Transformers à ce signal ?

Trois architectures majeures

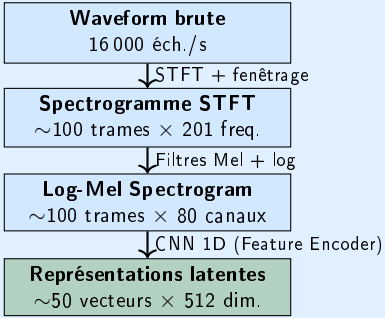
1. **Wav2vec 2.0** (Meta, 2020) — Auto-supervisé contrastif
2. **Whisper** (OpenAI, 2022) — Supervision faible encodeur-décodeur
3. **MMS** (Meta, 2023) — 1 100+ langues via adaptateurs

Comparaison des pipelines NLP textuel et audio



La chaîne de transformation du signal audio

De 16 000 échantillons/s à 50 vecteurs/s



Analogie clé

BPE (chapitre 2) : texte continu $\rightarrow$ tokens discrets

Feature Encoder : signal audio continu $\rightarrow$ “tokens acoustiques” discrets

La STFT : du temps à la fréquence

Transformée de Fourier à Court Terme

$$X(t,f) = \sum_{n=0}^{N-1} x(n) \cdot w(n - t \cdot H) \cdot e^{-j2\pi fn/N}$$

Interprétation

- $x(n)$: signal audio échantillonné
- $w(\cdot)$: fenêtre d'analyse (Hann)
- $e^{-j2\pi fn/N}$: noyau de Fourier
- $|X(t,f)|$: énergie à la fréquence f, instant t

Fenêtre de Hann

$$w(n) = 0.5 \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

Élimine les fuites spectrales aux bords de la fenêtre.

Principe d'incertitude temps-fréquence

Fenêtre courte (10 ms) $\rightarrow$ bonne résolution temporelle, mauvaise fréquentielle
Fenêtre longue (100 ms) $\rightarrow$ bonne résolution fréquentielle, mauvaise temporelle
Compromis standard : 25 ms ($n_{fft} = 400$ à 16 kHz)

L'échelle Mel et le Log-Mel Spectrogram

Conversion Hz → Mel

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$$

- $f \ll 700$ Hz : échelle quasi-linéaire
- $f \gg 700$ Hz : échelle logarithmique
- Mime la perception de la cochlée

Filtres triangulaires Mel

- Étroits en basses fréquences (haute résolution)
- Larges en hautes fréquences (basse résolution)
- 80 filtres (Whisper) ou 128 (Whisper v3)

Log-Mel Spectrogram

$$\text{Log-Mel}(t, m) = \log\left(\max\left(\epsilon, \sum_f |X(t,f)|^2 \cdot H_m(f)\right)\right)$$

Pourquoi le log ?

1. Compression de la dynamique
2. Séparation source-filtre :
   $\log(S \cdot F) = \log S + \log F$
3. Alignement perceptuel (loi de Stevens)

L'évolution vers le End-to-End

Systèmes hybrides (1990-2010)

1. Extraction MFCC
2. Modèle acoustique (HMM-GMM/DNN)
3. Modèle de langage (n-grammes)
4. Décodeur (Viterbi)

Chaque composant entraîné séparément.

Systèmes End-to-End (2018+)

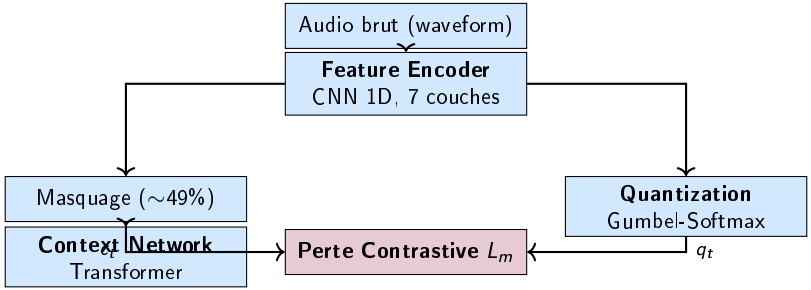
Un seul modèle neuronal :
Audio brut $\rightarrow$ Transcription

- Optimisation conjointe
- Moins d'ingénierie manuelle
- Meilleure généralisation
- Transfert multilingue

Attention

End-to-End $\neq$ pas de prétraitement. Le Log-Mel reste utilisé par Whisper. La différence : le modèle *apprend* à extraire l'information pertinente.

Architecture de wav2vec 2.0



Trois blocs principaux

1. **Feature Encoder** : waveform → représentations latentes (réduction $\times 320$)
2. **Context Network** : contextualisation via Transformer
3. **Quantization Module** : création de pseudo-tokens acoustiques discrets

Feature Encoder : CNN 1D multi-couches

Architecture : 7 couches Conv1D + LayerNorm + GELU

Couche	Ch. in	Ch. out	Kernel	Stride	Réduction
1	1	512	10	5	$\times 5$
2	512	512	3	2	$\times 10$
3	512	512	3	2	$\times 20$
4	512	512	3	2	$\times 40$
5	512	512	3	2	$\times 80$
6	512	512	2	2	$\times 160$
7	512	512	2	2	$\times 320$

Réduction temporelle

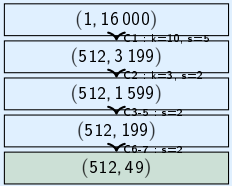
Stride total $= 5 \times 2^6 = 320$. Pour 1s d'audio à 16 kHz :
$T = 16\,000/320 = 50$ vecteurs latents (comparable au nombre de tokens BPE pour une phrase)

La convolution 1D : pourquoi et comment ?

Pourquoi la convolution ?

- **Invariance par translation** : un phonème “a” a les mêmes caractéristiques quelle que soit sa position
- **Détection de motifs locaux** : chaque kernel est un détecteur de motif acoustique

Réduction progressive



Formule

$$y(t) = \sum_{n=0}^{k-1} \sum_{c=0}^{C_{\text{in}}-1} x(c, t \cdot s + n) \cdot w(c, n)$$

Le **stride** s contrôle le sous-échantillonnage.

Implémentation du Feature Encoder

From scratch (NumPy) — Convolution 1D

```
def conv1d_from_scratch(x, kernel, stride=1):
    channels_in, length = x.shape
    channels_out, k_cin, kernel_size = kernel.shape
    output_length = (length - kernel_size) // stride + 1
    output = np.zeros((channels_out, output_length))
    for t in range(output_length):
        start = t * stride  # Le stride controle le pas
        segment = x[:, start:start + kernel_size]
        for c_out in range(channels_out):
            output[c_out, t] = np.sum(kernel[c_out] * segment)
    return output
```

PyTorch (nn.Module)

```
class FeatureEncoder(nn.Module):
    LAYER_CONFIGS = [(10,5),(3,2),(3,2),(3,2),(3,2),(2,2),(2,2)]
    def __init__(self, d_model=512):
        super().__init__()
        layers, in_ch = [], 1
        for k, s in self.LAYER_CONFIGS:
            layers.append(nn.Sequential(
                nn.Conv1d(in_ch, d_model, k, stride=s, bias=False),
                nn.GELU()))
```

Module de Quantification (Gumbel-Softmax)

Pourquoi discrétiser ?

- Empêcher le modèle de "tricher" (copie triviale)
- Créer un vocabulaire fini de "mots acoustiques"
- Analogue à K-means, mais apprenable

Product Quantization

- $G = 2$ groupes, $V = 320$ entrées/groupe
- Vocabulaire acoustique : $320^2 = 102\,400$ tokens
- Comparable aux 50 257 tokens BPE de GPT-2

Gumbel-Softmax

$$y_i = \frac{\exp\left((\log \pi_i + g_i)/\tau\right)}{\sum_j \exp\left((\log \pi_j + g_j)/\tau\right)}$$

- $g_i \sim \text{Gumbel}(0, 1)$: bruit stochastique
- τ : température
  - $\tau \to 0$: sélection dure (one-hot)
  - $\tau \to \infty$: sélection douce
- Straight-through estimator : argmax en forward, gradients softmax en backward

Perte Contrastive L_m

Formule

$$L_m = -\log \frac{\exp\left(\text{sim}(c_t, q_t)/\kappa\right)}{\exp\left(\text{sim}(c_t, q_t)/\kappa\right) + \sum_{\tilde{q} \sim Q_t} \exp\left(\text{sim}(c_t, \tilde{q})/\kappa\right)}$$

Similarité cosinus

$$\text{sim}(a, b) = \frac{a \cdot b}{\|a\| \times \|b\|}$$

- Invariante à la norme (énergie)
- Bornée $[-1, 1]$
- Discriminante en haute dimension

Interprétation

- c_t : contexte (sortie Transformer)
- q_t : cible quantifiée correcte
- $\tilde{q}$: distracteurs négatifs
- $\kappa = 0.1$: température
- Équivalent à une cross-entropy sur $K + 1$ classes

Lien InfoNCE

$I(c_t; q_t) \geq \log(K + 1) - L_m$ — Maximise l'information mutuelle contexte-cible

Perte de Diversité L_d et Perte Totale

Perte de diversité

$$L_d = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v}$$

- Empêche le **mode collapse**
- $\bar{p}_{g,v}$: fréquence d'utilisation de l'entrée v dans le groupe g
- Entropie maximale $\rightarrow$ utilisation uniforme du codebook

Perte totale

$$L = L_m + \alpha \cdot L_d \quad (\alpha = 0.1)$$

- L_m : guide l'apprentissage des représentations
- L_d : assure l'utilisation efficace du codebook

Mode collapse

Sans L_d, le quantificateur n'utilise que quelques entrées sur 640, rendant les représentations inutiles.

XLS-R : Extension Cross-Lingue

Statistiques

Caractéristique	Valeur
Volume de données	436 000 heures
Couverture linguistique	128 langues
Tailles de modèle	300M, 1B, 2B paramètres

Hypothèse : inventaire phonétique universel

- L'IPA recense $\sim$600 consonnes et voyelles distinctes
- Chaque langue n'en utilise qu'un sous-ensemble (français $\sim$36, Zoulou $\sim$59)
- XLS-R apprend des représentations couvrant cet espace partagé
- Facilite le transfert vers de nouvelles langues

Adaptation

XLS-R + couche CTC + fine-tuning sur petit corpus étiqueté $\rightarrow$ ASR pour une nouvelle langue

Whisper : Supervision Faible à Grande Échelle

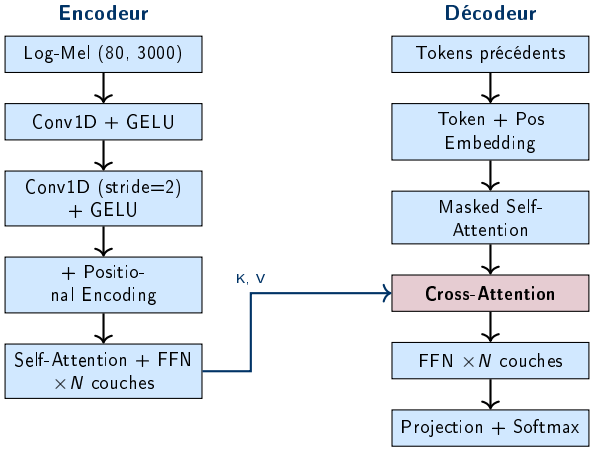
Approche

- **680 000 heures** de données audio faiblement étiquetées (Internet)
- Architecture **encodeur-décodeur Transformer** classique
- Approche radicalement différente de wav2vec 2.0 (supervisé vs auto-supervisé)

Prétraitement Log-Mel

Paramètre	**Whisper**	**Whisper v3**
Fréquence d'échantillonnage	16 000 Hz	16 000 Hz
Canaux Mel (n_{mels})	80	128
Fenêtre FFT (n_{fft})	400 (25 ms)	400 (25 ms)
Hop length	160 (10 ms)	160 (10 ms)
Durée max segment	30 s	30 s

Architecture Encodeur-Décodeur de Whisper



Cross-Attention : le lien encodeur-décodeur

$\text{CrossAttn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ où Q vient du décodeur, K, V de l'encodeur

Tokens Spéciaux et Capacités Multitâche

Séquence de tokens spéciaux

Capacités

- Transcription multilingue
- Traduction vers l'anglais
- Détection de langue
- Timestamps au niveau mot

Stratégies de décodage

- **Temperature Fallback** : augmente τ si confiance faible
- **Décodage glouton** : argmax à chaque pas
- **Anti-hallucination** : log-prob, compression Gzip

Hallucinations

Whisper peut halluciner sur les langues peu dotées. Les heuristiques anti-hallucination sont cruciales.

Évaluation : WER et CER

Distance de Levenshtein (programmation dynamique)

$$\text{dp}[i][j] = \begin{cases} \text{dp}[i-1][j-1] & \text{si ref}[i] = \text{hyp}[j] \\ 1 + \min(\text{dp}[i-1][j],\ \text{dp}[i][j-1],\ \text{dp}[i-1][j-1]) & \text{sinon} \end{cases}$$

Opérations : substitution (S), suppression (D), insertion (I)

WER (Word Error Rate)

$$\text{WER} = \frac{S + D + I}{N_{\text{ref\_mots}}}$$

Standard, niveau mot.

CER (Character Error Rate)

$$\text{CER} = \frac{S_c + D_c + I_c}{N_{\text{ref\_chars}}}$$

Niveau caractère, plus adapté aux langues agglutinantes.

Implémentation

From scratch : matrice de programmation dynamique $O(n \times m)$
Optimisé : bibliothèque `jiwer` en 2 lignes

MMS : Massively Multilingual Speech

Stratégie de collecte de données

Utilisation de **textes religieux** (traductions de la Bible) comme source de données audio alignées.

Avantages

- Couverture : 1 100+ langues
- Alignement texte-audio naturel (versets)
- Bonne qualité acoustique

Limites

- Domaine restreint (registre religieux)
- Souvent un seul locuteur/langue
- Biais de genre (voix masculines)

Impact pour les langues africaines

Premiers systèmes ASR pour le Fon (Bénin), Bambara (Mali), Lingala (RDC), Twi (Ghana) — langues sans aucune ressource ASR auparavant.

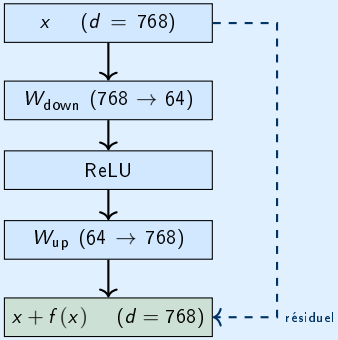
Architecture Bottleneck Adapter

Formule

$$\text{Adapter}(x) = x + W_{\text{up}} \cdot \text{ReLU}(W_{\text{down}} \cdot x)$$

- $W_{\text{down}} : d \to r$ $(768 \to 64)$
- ReLU : non-linéarité
- $W_{\text{up}} : r \to d$ $(64 \to 768)$
- $x + \ldots$: connexion résiduelle

Schéma

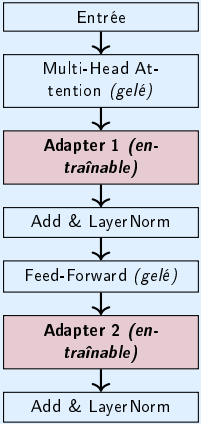


Paramètres

- Par adaptateur : $\sim$100K params
- 2 par couche $\times$ 12 couches = 2.4M
- Ratio : $\sim$0.8% du modèle base

Insertion des Adaptateurs dans le Transformer

Schéma d'une couche



Procédure

1. **Geler** le modèle de base (`requires_grad = False`)
2. **Insérer** adaptateurs après attention et FFN
3. **Entraîner** uniquement les adaptateurs

Efficacité de stockage

- 1 100 langues $\times$ 2.4M = 2.6G params adaptateurs
- + 300M modèle base partagé
- vs 1 100 $\times$ 300M = 330G (modèles complets)
- Gain : $\sim$**100**$\times$ plus efficace

Pipeline MMS complet

La Perte CTC : Alignement Automatique

Problème

Feature Encoder : $\sim$50 vecteurs/s. Transcription : $\sim$20 caractères. Comment aligner sans supervision ?

Formalisme CTC

$$P(\mathrm{y}|\mathrm{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathrm{y})} \prod_{t=1}^{T} P(\pi_t|\mathrm{x})$$

Marginalise sur **tous les alignements possibles** (programmation dynamique).

Exemple : cible “ab”, $T = 5$ trames

Alignement π	**Après suppression blanks/répétitions**
a-a-ϵ-b-b	ab
ϵ-a-ϵ-b-ϵ	ab
a-ϵ-ϵ-ϵ-b	ab

CTC

Tonalité : quand le pitch change le sens

Yoruba (Nigeria, ∼45M locuteurs)

3 tons : haut (´), moyen, bas (‘)

Mot	Ton	Sens
owó	haut-haut	argent
owò	haut-bas	respect
ọwọ	bas-haut	main

Igbo (Nigeria, ∼30M locuteurs)

2 tons principaux + tons modulés

Mot	Ton	Sens
ákwá	haut-haut	pleurs
àkwà	bas-bas	tissu
àkwá	bas-haut	œuf
ákwà	haut-bas	pont

Impact sur l’ASR

Les modèles entraînés sur langues non tonales n’exploitent pas le pitch pour la discrimination lexicale. Solutions : features F0 explicites, fine-tuning sur données tonales, métriques TER/FER.

Morphologie Agglutinante

Swahili : *hatutawasiliana* = “nous ne communiquerons pas les uns avec les autres”

Morphème	Fonction
ha-	Négation
tu-	Sujet 1ère pers. pluriel
-ta-	Temps futur
-wa-	Objet 3ème pers. pluriel
-sili-	Racine (“communiquer”)
-an-	Réciprocité
-a	Voyelle finale

Impact sur la tokenisation

- Vocabulaire combinatoire explosif
- BPE peut découper incorrectement
- WER artificiellement élevé

Zoulou : *ngiyakuthanda*

ngi- (je) + -ya- (présent) + -ku- (te) + -thand- (aimer) + -a (fin)
= “je t’aime”

Code-Switching : alternance de langues

Exemples

- **Swahili-Anglais :** "*Nilienda* ***shopping*** *jana na* ***friend*** *yangu*"
  ("Je suis allé faire du shopping hier avec mon ami")
- **Yoruba-Anglais :** "*Mo ti* ***download app*** *náà, ṣùgbọn kò* ***work***"
  ("J'ai téléchargé l'application, mais ça ne marche pas")

Défis

- Transitions entre langues imprévisibles
- Phonèmes des deux langues coexistent
- Vocabulaire doit couvrir les deux langues

Approches

- Données avec code-switching naturel
- Modèles multilingues (XLS-R, Whisper)
- Combinaison d'adaptateurs

Métriques d'Évaluation Adaptées

Comparaison des métriques

Métrique	Formule	Usage
WER	$\frac{S+D+I}{N_{\text{ref}}}$	Standard, niveau mot
CER	$\frac{S_c+D_c+I_c}{N_{\text{ref\_chars}}}$	Langues agglutinantes
TER	$\frac{\text{Err. tonales}}{N_{\text{syllabes}}}$	Langues tonales
FER	$\frac{\text{Trames F0 err.}}{N_{\text{trames}}}$	Précision du pitch

Exemple Swahili : *hatutawasiliana* → *hatutawasilian*

- **WER** : 1 mot erroné / 3 = **33.3%** (pénalise lourdement)
- **CER** : 1 caractère manquant / 30 = **3.3%** (reflète mieux la qualité)

Exemple Yoruba : *owó* (argent) → *owò* (respect)

WER = 100%, CER = 33%, **TER = 50%** (capture l'erreur tonale spécifiquement)

Récapitulatif : Trois Architectures Complémentaires

Comparaison

	Wav2vec 2.0	**Whisper**	**MMS**
Paradigme	Auto-supervisé	Supervisé faible	Transfer learning
Entrée	Waveform brute	Log-Mel	Waveform brute
Architecture	Encodeur seul	Enc.-Décodeur	Enc. + Adaptateurs
Données	960h (LS)	680 000h	Textes religieux
Langues	1 (+ XLS-R)	99	1 100+
Sortie	Représentations	Texte	Texte (CTC)

Points clés

- Wav2vec 2.0 : apprend des représentations universelles sans étiquettes
- Whisper : robuste grâce à la quantité massive de données supervisées
- MMS : efficace en paramètres ($\sim$0.8%) pour couvrir 1 100+ langues

Conclusion et Perspectives

Ce que nous avons appris

1. Chaîne de transformation : waveform $\rightarrow$ spectrogramme $\rightarrow$ Log-Mel $\rightarrow$ représentations latentes
2. **Wav2vec 2.0** : Feature Encoder (CNN $\times 320$), quantification Gumbel-Softmax, perte contrastive $L_m + \alpha L_d$
3. **Whisper** : encodeur-décodeur, tokens spéciaux multitâche, 680K heures
4. **MMS** : adaptateurs bottleneck ($\sim$0.8% params), 1 100+ langues via CTC
5. Défis africains : tonalité, morphologie agglutinante, code-switching

Compétences acquises

- ✓ Implémenter from scratch : STFT, filtres Mel, Conv1D, perte contrastive, adaptateurs
- ✓ Utiliser Whisper pour la transcription multilingue
- ✓ Évaluer avec WER, CER, TER, FER
- ✓ Concevoir des adaptateurs pour de nouvelles langues

Références

Articles fondateurs

- Baevski et al. (2020). *wav2vec 2.0 : A Framework for Self-Supervised Learning of Speech Representations*. NeurIPS.
- Radford et al. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision*. arXiv.
- Pratap et al. (2023). *Scaling Speech Technology to 1,000+ Languages*. arXiv.
- Conneau et al. (2020). *Unsupervised Cross-lingual Representation Learning for Speech Recognition*. arXiv.

Questions ?

Merci de votre attention !