

Rapport — Adapter-Based Fine-Tuning (Whisper-small) — ASR Fellowship Challenge

1. Informations personnelles

- Nom : NONO NGANSOP NAHOMIE MADELEINE
- Email / Contact : nahomie.nono@facscience-uy1.cm

2. Description de l'expérience

- Modèle de base : openai/whisper-small
- Dataset : DigitalUmuganda/ASR_Fellowship_Challenge_Dataset (Afrivoice_Kinyarwanda health subset)

3. Architecture des adaptateurs

- Topologie : bottleneck linear ($d_{\text{model}} \rightarrow r \rightarrow d_{\text{model}}$), activation ReLU, up init = 0.
- Emplacement : adaptateurs ajoutés après FFN de chaque couche d'encodeur et de décodeur.

4. Stratégie d'entraînement

- Poids gelés du modèle base.
- Paramètres entraînables : adaptateurs uniquement.
- Hyperparamètres : lr=3e-4, epochs=3, batch=8, bottleneck_dim=64, optim=AdamW.

5. Résultats attendus

- WER modèle base : ...
- WER modèle affiné : ...
- Différence WER : ...
- Nombre de paramètres entraînables : ...

6. Reproductibilité & Instructions (pas à pas)

1. Installer les dépendances: `pip install -r requirements.txt`
2. Télécharger le dataset (si besoin) et exécuter l'entraînement: `python src/train.py --model_name openai/whisper-small --batch_size 8 --num_epochs 3 --adapter_dir ./adapters`
3. Générer transcriptions: `python src/evaluate.py --model_name openai/whisper-small --adapter_path None --split test --out base_transcriptions.txt`
`python src/evaluate.py --model_name openai/whisper-small --adapter_path ./adapters/adapter_weights.pth --split test --out finetuned_transcriptions.txt`

7. Conclusion & prochaines étapes

- Suggestions: data augmentation, LoRA, PEFT integration, validation split, scheduler tuning.