Programmation mobile

Android SDK – React Native

Projet Final

SORBONNE UNIVERSITÉ

Concept

Le but de ce projet est de réaliser une application de transfert de timbre en ReactNative. Le transfert de timbre sera réalisé par un réseau de neurones, le modèle RAVE développé par Antoine Caillon à l'ircam.

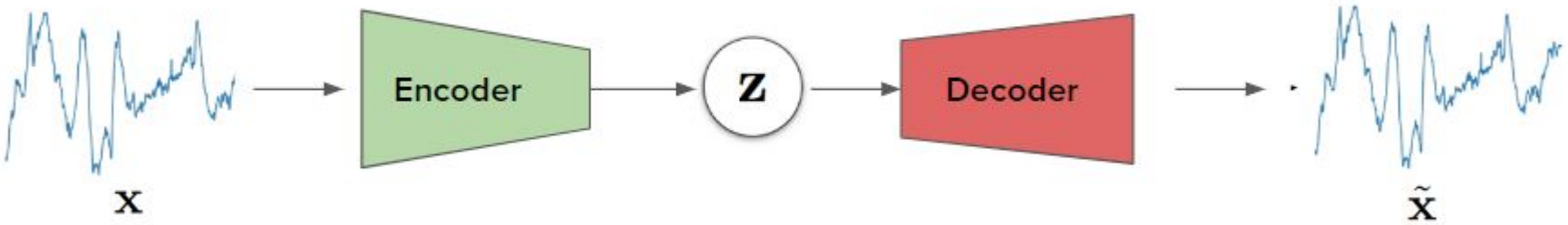
Vous pouvez trouvez une démonstration très similaire à cette application à l'adresse suivante : https://caillonantoine.github.io/ravejs/.

Puisqu'il est difficile de faire réaliser les calculs du modèle au téléphone directement, un serveur python sera mis en place pour la partie calcul et renverra les clips audio transformés.

En résumé l'application permettra :

- De se connecter au serveur
- D'enregistrer et de sauvegarder des clips avec le micro du téléphone
- D'envoyer ces clips et d'écouter / sauver le résultat sur le téléphone

Le transfert de timbre



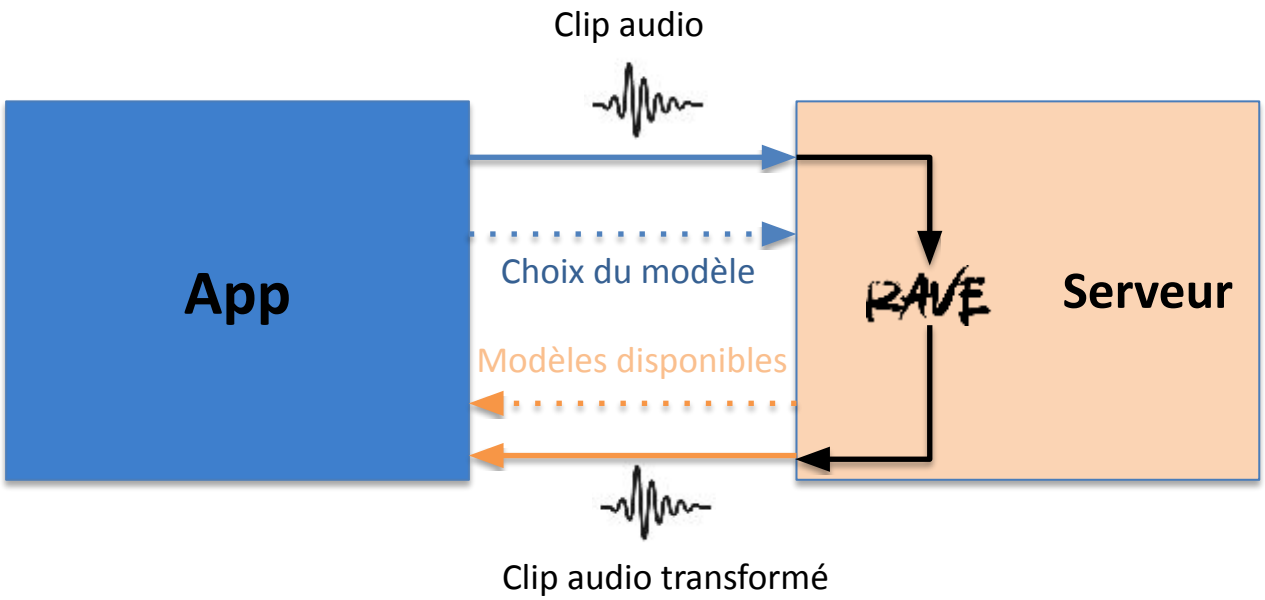
Un modèle de type VAE est entrainé sur un ensemble d'exemples de façon de façon à être capable d'encoder et de décoder un échantillon audio. La représentation intermédiaire, appelée représentation latente, capture les caractéristiques haut-niveau du son. Lorsque l'on utilise le modèle avec un type de son sur lequel le modèle n'a pas été entrainé, il transfert ses caractéristiques vers le domaine d'apprentissage.

5 modèles sont mis à votre disposition : Jazz, Darbouka, Parole, Chats, Chiens.
Les modèles sont entrainés en python (Pytorch) et ici converti pour simplicité en onnx.

Application

Votre application devra comporter 3 vues différentes

- Une vue « Home » qui permettra de se connecter au serveur
- Une vue « Record » qui permet d'enregistrer des clips audio
- Une vue « RAVE » qui permet de choisir un clip audio et de le transférer au serveur pour écouter le résultat.



API du serveur

Le serveur fourni est un serveur flask Python (voir installation en annexe). Les différentes requêtes disponibles sont les suivantes :

- « / »
  Requête par défaut. Retourne un message « Connexion success ! ».
- « /upload »
  - Requête POST pour envoyer le fichier audio à transformer. Le calcul est effectué automatiquement et le serveur renvoie un message de confirmation quand le calcul est terminé. Fonctionne sur des fichiers « .wav » ou « .m4a » générés par android.
- « /download »
  - Permet de télécharger le fichier audio transformé
- « /getmodels »
  - Renvoie la liste de modèle sous forme JSON
- « /selectModel/<modelname>»
  - Requête pour sélectionner le modèle à utiliser parmi la liste de modèle

Les fonctions JS react-native pour télécharger et uploader un fichier vous sont fournies en annexe.

Vue Home

Cette vue permet d'entrer les informations du serveur et de s'y connecter

Elle devra contenir :

- Une entrée utilisateur pour l'adresse IP
- Une entrée pour le port
- Un bouton qui permet de tester la connexion en envoyant une requête au serveur. L'application devra notifier l'utilisateur via un toast ou un message.

Vue Record

Cette vue permet d'enregistrer des clips audios et de les stocker dans le téléphone**.**

Elle devra contenir :

- Un bouton pour commencer et terminer l'enregistrement
- Une lecture play/pause de cet enregistrement
- Un bouton pour sauver l'enregistrement (en lui donnant un nom).
- Un affichage des enregistrements déjà effectués (de type FlatList ou autre), avec la possibilité de supprimer ou de réécouter les enregistrements.

Pour la lecture et l'enregistrement audio, il est recommandé d'utiliser la librairie expo : https://docs.expo.dev/versions/latest/sdk/audio/

Pour la gestion des fichiers : https://docs.expo.dev/versions/latest/sdk/filesystem/

Vue RAVE

Cette vue permet de sélectionner un clip audio, de l'envoyer au serveur, de télécharger le résultat puis de l'écouter.

Elle devra contenir :

- Un système de vue (Tab par exemple) permettant de choisir entre :
  - Charger un son par défaut (stocké dans les assets de l'application)
  - Sélectionner un clip parmi les enregistrements de la vue « Record »
  - Sélectionner un son dans les fichiers du téléphone (de la musique par exemple)

  Une solution pour afficher des tabs dans une seule et même vue : https://github.com/satya164/react-native-tab-view

- Un bouton pour envoyer le clip au serveur. Un widget indiquera que le modèle est en train de calculer (par ex: https://reactnative.dev/docs/activityindicator). Lorsque le calcul est terminé, le son transformé sera téléchargé automatiquement

- Deux boutons pour lire le fichier audio original et le transformé.

Consignes générales

- L'ensemble des variables/callbacks partagés entre les différentes vues devront être gérées avec un Store de type Redux
- Seules les variables utilisées uniquement par les composants seront donc gérées avec useState.
- Les enregistrements effectués dans la vue records devront être sauvegardées de façon à être retrouvées à l'ouverture de l'application (avec donc des fichiers en mémoire, et un state qui conserve le lien vers leur emplacement sauvé via le store *redux-persist* par exemple). Pour les enregistrements audio, ils devront être copiés du cache vers le dossier de stockage persistant de l'app.
- La navigation se fera entre les vues en swipant (avec des *Tabs react-navigation*)

Evaluation

Rendu

Vous devez publier votre application sur expo pour qu'elle puisse être lancée via un QR-Code.

Votre code complet sera rendu via un repo github. Si vous souhaitez faire des modifications au serveur, vous pouvez rendre un code python, mais n'uploadez pas les modèles (gitignore).

Evaluation

Testez bien votre appli ! Vous serez pénalisés si certaines combinaison d'actions provoque des erreurs.

Une grande attention sera portée à la lisibilité du code, qui doit être commenté. L'ergonomie de votre application sera également évalué (facilité de prise en main, design (utilisation d'icones par ex https://ionic.io/ionicons).

Annexes : Serveur Python

Installation du serveur python

- Pour utiliser le serveur python, vous devez utiliser un environnement anaconda. Si vous n'avez pas encore anaconda, vous pouvez installer miniconda (version réduite de anaconda) à l'adresse suivante https://docs.conda.io/en/latest/miniconda.html
- Vous devez ensuite installer les packages nécessaires (dans un terminal avec votre environnement miniconda activé (Anacomda Prompt sur Windows, terminal sur Mac/Linux) :

```
cd <serverPath>
pip install -r requirements.txt
```

- Installer **ffmpeg** pour la conversion des fichiers audios reçus du téléphone.

Windows

Suivre ce lien

Mac/Linux

```
sudo apt-get install ffmpeg
```

Lancer le serveur

```
python server.py
```

L'adresse IP et le port du serveur s'affiche dans le terminal

Annexe : Fonctions Up/DL

```
import * as FileSystem from 'expo-file-system';
import { Asset, useAssets } from 'expo-asset';


// Download file to document directory
const downloadFile = async () => {
    // Create a directory in the app document directory
    let directory = FileSystem.documentDirectory + "my_directory"
    await FileSystem.makeDirectoryAsync(directory);

    // Download file
    const { uri } = await FileSystem.downloadAsync(serverAdress + "/download", directory + "/hey.wav")
}

// Get the asset file local uri
const [assets, error] = useAssets([require('../assets/audio.wav')]);

// Send file function
const sendFile = async () => {
    let fileUri = assets[0].localUri // The uri of the file you want to upload
    resp = await FileSystem.uploadAsync(serverAdress + "/upload", uri, {
        fieldName: 'file',
        httpMethod: 'POST',
        uploadType: FileSystem.FileSystemUploadType.MULTIPART,
        headers: { filename: uri }
    })
    console.log(resp.body)
}
```