LONDE Tristan

👋 Hello!

I'm an Artificial Intelligence engineering student at **ENIB** (École Nationale d'Ingénieurs de Brest), passionate about **speech synthesis**, **signal processing**, **computer vision**, and building **offline intelligent systems**.

In my free time, I design and develop **fully local voice assistants**, **machine learning applications for voice**, and innovative projects at the intersection of **cybersecurity**, **computer vision**, and **AI-generated content detection** (like **deepfake audio**).

I specialize in creating **privacy-focused local voice solutions** and explore open-source LLMs like **GPT4All**, **Nous Hermes**, and **Mistral**. I also enjoy working with tools like **FastAPI**, **Streamlit**, and **OpenCV** to build **real-time AI-powered interfaces**.

🌎 Interests

- 💻 **AI & Machine Learning**: Real-time models, voice AI, local-first systems
- 🎤 **Speech Synthesis**: Custom voice assistants, voice-to-voice pipelines
- 👁️ **Computer Vision**: Fatigue detection, facial analysis, eye tracking
- 🛡️ **Cybersecurity**: Deepfake detection, audio anti-spoofing

---

EmotionAI-voice

An AI-powered application for detecting human emotions

EmotionAI Voice is an open-source deep learning project that classifies vocal emotions using raw `.wav` audio.
It's designed for applications in mental health monitoring, UX analysis, and intelligent speech interfaces.

🔬 The model is trained **from scratch**, using spectrogram-based audio features, and aims to recognize 8 core emotions.

🎯 Features

- 🧠 Emotion recognition: `neutral`, `calm`, `happy`, `sad`, `angry`, `fearful`, `disgust`, `surprised`
- 🎧 Accepts `.wav` audio inputs (from RAVDESS dataset)
- 📊 CNN and CNN+GRU models implemented in PyTorch
- 🔍 Real-time evaluation with confusion matrix and accuracy tracking
- 🛠️ Fully open-source and customizable (no pre-trained models)
- 🧪 Includes **SpecAugment** for data augmentation (frequency/time masking)

📚 Dataset — RAVDESS

We use the **RAVDESS** dataset, which includes:

- 🎭 24 professional actors (balanced male/female)
- 🎙️ 1440 `.wav` files (16-bit, 48kHz)
- 8 labeled emotions:
  `neutral`, `calm`, `happy`, `sad`, `angry`, `fearful`, `disgust`, `surprised`

Each `.wav` file is preprocessed into a **Mel spectrogram** and stored as `.npy` format.

🧠 Model Architectures

2 different models

✅ CNN (Best Performance)

- 3x Conv1D + ReLU + MaxPool
- Fully connected layers
- Dropout regularization (adjustable)

🔁 CNN + GRU

- CNN front-end for spatial encoding
- GRU (recurrent layers) to capture temporal dynamics
- Lower accuracy than CNN-only model

🧪 SpecAugment: Data Augmentation

To improve generalization, we implemented `SpecAugmentTransform` which applies:

- 🕒 **Time masking**: hides random time intervals
- 📡 **Frequency masking**: hides random mel frequency bands

📈 Training Results

- Best Validation Accuracy: **~49.6%**
- Training set: Actors 1–20
- Validation set: Actors 21–24

Confusion Matrix Example:

🔍 Key Observations:

- Surprised, calm, and disgust are the most accurately predicted emotions.
- Neutral, happy, and sad tend to be confused with each other, which is common due to subtle acoustic variations.
- The model struggles with fearful and angry in some cases — suggesting those may share overlapping vocal characteristics in this dataset.
- Emotion classes like happy and fearful are often misclassified due to variability in expression intensity among different actors.

📈 Interpretation

While the model captures general emotion cues, it suffers from class overlap and limited generalization. The accuracy remains significantly above random (12.5% for 8 classes), but there is still room for improvement.

🚀 Getting Started

1. Install dependencies

```
pip install -r requirements.txt
```

2. Download dataset from Kaggle

Follow the instructions in the README.md located in the data folder

3 . Train the model

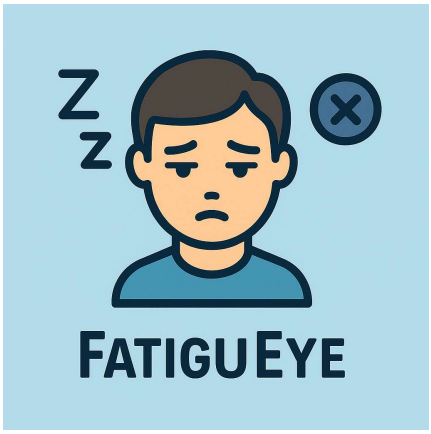
```
python src/train.py
```

4. Evaluation the performances with a confusion matrix

```bash python src/confusion_matrix.py
```

FatiguEye

Detection of fatigue using a webcam



A smart computer vision system that detects signs of fatigue, eye strain, and microsleep using a standard webcam.

🎯 Purpose

FatiguEye is a real-time fatigue detection system based on eye tracking and facial landmark analysis.
It helps identify early signs of drowsiness by measuring:

- 👁️ Eye Aspect Ratio (EAR)
- 🔁 Blink frequency
- ⏱️ Prolonged eyelid closure
- ⚠️ Microsleep events

Ideal for driver monitoring, industrial safety, or ergonomic fatigue prevention.

🧠 How It Works

FatiguEye uses MediaPipe Face Mesh to extract eye landmarks, and computes the **EAR (Eye Aspect Ratio)** on each video frame.

📡 Processing pipeline:

1. 🎥 Webcam feed is captured in real-time
2. 🧠 Facial landmarks (eyes) are detected with Mediapipe
3. 📏 EAR is calculated per eye
4. 🪟 Blink count and eye closure duration are analyzed
5. 🔔 Fatigue alerts are raised (visual + audio)

🚀 Demo Preview

💻 Technologies Used

Tech	Description
Python	Core language
OpenCV	Webcam video processing + overlays
MediaPipe	Face mesh & eye landmark detection
NumPy	EAR computation
Streamlit	Live web dashboard

Tech	Description
winsound	Audio alert (Windows only)

Installation

```bash git clone https://github.com/Tirovo/fatigueye.git cd fatigueye python -m venv venv source venv/bin/activate # Or venv\Scripts\activate on Windows
```

VATN-WaterGame

💧 A strategic game on water management!

VATN (from the Old Norse word for “water”) is a game designed to raise awareness about water management. Players must make critical daily decisions to address major water-related issues in their country, influencing key parameters that determine the nation’s survival.

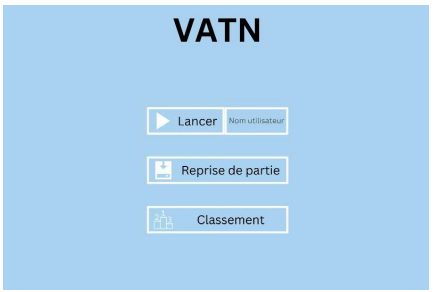
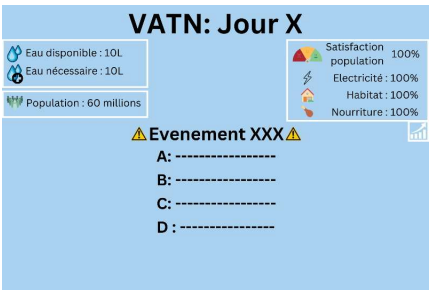
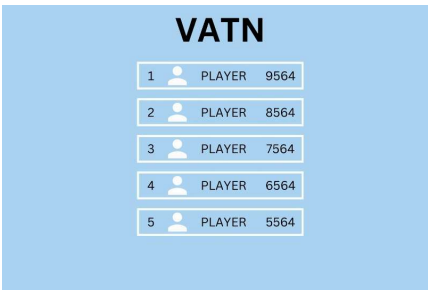
🎯 Purpose

- 💧 **Water Management Awareness**: Educate players on the importance of sustainable water policies.
- 🎮 **Engaging Decision-Making**: Every day presents a new challenge that affects the nation’s status.
- 🏆 **Survival & Strategy**: Keep your country alive as long as possible by maintaining stability.
- 📊 **Progress Tracking**: Visualize the evolution of key parameters through in-game graphs.

Features

🏷️ Feature	🔍 Description
🌍 **Game Type**	Strategic Decision-Making Simulation
📅 **Daily Choices**	Players make decisions each day affecting country parameters
📉 **Dynamic Statistics**	Key indicators fluctuate based on player actions
💀 **Game Over**	The country collapses if the population reaches 0
📂 **Save & Load**	Resume previous games using saved files
🏅 **Rankings**	Compare results with previous local games
📊 **Graphical Summary**	Track country status evolution over time

Gameplay Overview

🎮 Main Menu	📊 In-Game Statistics	🏆 Endgame Rankings
		

AntiVuvuzelaFilter

🎵 Noise filter for clear audio 🎙️

Anti-Vuvuzela Filter is an open-source project dedicated to **second-order analog filters** and beyond. This project was initially developed to design an **"anti-vuvuzela" filter**, aiming to attenuate the distinctive and persistent sound of vuvuzelas while preserving the clarity of commentators' voices during the **2010 FIFA World Cup**.

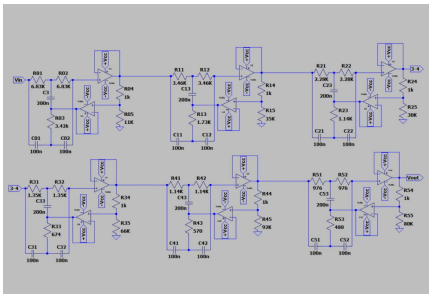
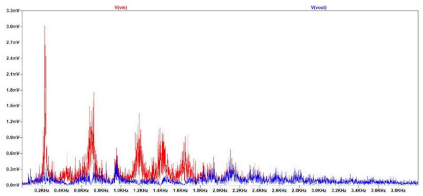
🎯 Purpose

- 🔇 **Targeted Noise Reduction**: Specifically designed to **attenuate vuvuzela noise** while maintaining the intelligibility of speech.
- 🎚️ **Second-Order Analog Filtering**: Utilizing advanced filtering techniques for efficient noise cancellation.
- 🛠️ **Open-source and Customizable**: Modify and adapt the design for other audio filtering applications.

📝 Features

🏷️ Feature	🔍 Description
🎼 **Filter Type**	Second-order analog filter
🎯 **Target Frequency**	233 Hz (typical vuvuzela frequency)
🎙️ **Voice Preservation**	Maintains speech clarity
🔧 **Components**	Resistors, capacitors, and operational amplifiers
🖥️ **Simulation Tools**	Jupyter Notebook, LTSpice
🛠 **Real-world Testing**	Assembled and tested in real conditions
🎤 **Input**	Analog audio signal
🔊 **Output**	Cleaned audio signal with reduced vuvuzela noise
🌍 **Use Cases**	Audio signal processing, speech enhancement, noise reduction

📐 Simulation & Testing

🛠️ LTSpice Circuit	📜 Simulation
	

MotorControlShield

🔄 Arduino shield for single DC motor control

The **Motor Control Shield** is an open-source project designed for controlling DC motors. It comes in the form of an Arduino shield mounted on an STM32 Nucleo board. The shield enables motor control via an NMOS transistor, current measurement with a shunt resistor, and rotation tracking using data from an incremental encoder.

🎯 Purpose

- 🔄 **Motor Control**: Provides precise control over DC motors.
- 📉 **Current Measurement**: Monitors the current consumed by the motor.
- 🔄 **Rotation Tracking**: Uses an incremental encoder to track motor rotation.
- 🛠️ **Open-source & Customizable**: Modifiable and adaptable for various projects.

📝 Features

🏷️ Feature	🔍 Description
🔄 **Motor Control**	Uses an NMOS transistor to control motor speed and direction
📉 **Current Measurement**	Shunt resistor for measuring the current consumed by the motor
🔄 **Rotation Tracking**	Incremental encoder to track motor position and rotation speed
⭕ **Compatibility**	Arduino shield compatible with STM32 Nucleo boards
🖥️ **PCB Design**	Open-source and customizable
🌍 **Use Cases**	robots, embedded systems, and motor control applications

📐 PCB Design Preview

📜 Functional diagram	📜 Schematic	🖥️ PCB Layout	3D