Technical Review Explainable AI

Explainable AI Platform for
Deepfake Audio Detection and Lung Cancer Classification

Team Members

Pierre Briand
Tom Delahaye
Alexandre Laroudie
Kentin Guillemot
Gabriel Carlotti

January 6, 2026

Contents

1 Introduction and Context

The objective of this project is to deliver a unified **Explainable Artificial Intelligence (XAI)** interface serving two distinct modalities:

- Deepfake audio detection
- Lung cancer classification from chest X-ray images

The platform integrates multiple explainability techniques — **LIME**, **Grad-CAM**, and **SHAP** — and enables side-by-side comparison of explanations.

Technology Stack Rationale

- **FastAPI**: lightweight, asynchronous REST API with simple CORS management.
- **TensorFlow / Keras**: retained for audio processing to preserve compatibility with existing spectrogram-based models.
- **PyTorch**: used for image models to leverage `torchvision` and native Grad-CAM hooks.
- **React + Vite (TanStack Router)**: responsive single-page application capable of orchestrating multi-step uploads and XAI requests.

Scope

The platform:

- Accepts audio (`.wav`) and image files,
- Exposes pretrained models,
- Enforces modality-dependent XAI usage,
- Renders explanations in a comparison-oriented interface,
- Provides a runnable demo and documentation.

2 Architecture Overview

2.1 API Service

The API service (`api.py`) exposes:

- Health and discovery endpoints (`/api/models`, `/api/xai-techniques`),
- Analysis endpoints for audio and image modalities.

Uploads are stored in temporary locations using file-type allowlists:

- Audio: `wav`, `mp3`, `m4a`
- Images: `png`, `jpg`, `jpeg`, `bmp`

A maximum file size constant exists but is not yet enforced. Lazy loading ensures fast startup by importing modality modules only when required.

2.2 Audio Backend

The audio backend (`backend/audio_deepfake`) includes:

- Input validation and spectrogram standardization,
- Deepfake detection models (MobileNet, VGG16, ResNet50),
- XAI methods (Grad-CAM, LIME, SHAP) applied to spectrograms,
- A command-line pipeline producing JSON predictions and PNG explanations.

Original architectures were preserved to maintain continuity with prior training.

2.3 Image Backend

The image backend (`backend/lung_cancer`) is based on:

- A ResNet50 predictor with optional custom weights,
- A three-class classification setup (Adenocarcinoma, Benign, Squamous),
- Grad-CAM and LIME explainability methods.

SHAP is intentionally omitted for images due to the lack of a stable implementation in the original repository.

2.4 Frontend

The frontend provides:

- Drag-and-drop uploads,
- Media-type detection,
- Dynamic model selection,
- XAI method selection,
- Display of predictions and confidence scores.

Visual rendering of heatmaps is planned but not yet implemented.

3 Selected Models and XAI Methods

3.1 Audio

Models:

- MobileNet
- VGG16
- ResNet50

All models operate on 224×224 spectrogram inputs.

XAI Methods:

- Grad-CAM: spatial saliency localization,
- LIME: local perturbation-based explanations,
- SHAP: feature attribution using Shapley values.

3.2 Image

Model: ResNet50 (PyTorch), with ImageNet weights by default.

XAI Methods:

- Grad-CAM,
- LIME,
- SHAP: not available (explicit placeholder response).

Model and method availability is partially filtered by modality; unsupported combinations currently return backend error messages.

4 Improvements over Original Repositories

Key enhancements include:

- Unified API surface and discovery endpoints,
- Centralized upload handling and structured logging,
- Shared CLI pipeline ensuring reproducibility,
- A single frontend orchestrating all workflows.

These improvements enable scalability, maintainability, and a smoother user experience.

5 Future Enhancements

Planned improvements:

- Enforce XAI compatibility rules at UI and API levels,
- Return and render explanation visualizations,
- Provide documented pretrained weights and model metadata,
- Enforce file size, duration, and dimension constraints,
- Improve performance through caching, offloading, and GPU support,
- Add automated tests and monitoring.

6 Run Notes

API

```
python api.py
```

Default URL: `http://localhost:5000` Swagger UI available at `/docs`.

Frontend

```
pnpm install
pnpm dev
```

CLI Demo

```
python main.py audio <file.wav> --model mobilenet
python main.py image <scan.png> --output ./results
```

These commands generate prediction JSON files and corresponding XAI visualizations.