Unified Explainable AI Interface for Multi-Modal Classification

Explainability AI Project – ESILV A5 DIA2

Élèves :
Hugo BONNELL
Mathieu COWAN

Enseignants :
Walid KHERIJI

6 janvier 2026

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

Table des matières

1 Introduction

Abstract - This report presents a unified Explainable AI (XAI) interface that combines audio deepfake detection and chest X-ray classification into a single Streamlit application. Users can upload an image or audio file, select a compatible pretrained model, and generate explanations with Grad-CAM, LIME, or SHAP. The design emphasizes modular preprocessing, a centralized model registry, and reusable explainability wrappers to ensure consistent behavior across modalities and a transparent user experience.

The problem addressed is the fragmentation of XAI demonstrations across separate repositories and modalities. Students and practitioners often need to compare explanation methods on different media types, yet workflows, formats, and tool interfaces vary widely. The objective of this project is to provide a single, coherent interface that supports inference and comparative explainability for both image and audio models, with minimal setup and clear visual outputs.

The report focuses on the technical architecture, design rationale, and key implementation details of this unified system. It outlines how the application routes inputs, performs preprocessing, executes model inference, and generates explanations in a reproducible manner, while keeping the interface accessible for classroom and demo contexts.

2 Mission and Goals

The mission is to deliver an interactive, explainable inference interface that is simple to run, consistent across modalities, and informative for users. Key goals include :

- Unified workflow for image and audio classification.
- Comparable XAI outputs across Grad-CAM, LIME, and SHAP.
- Clear visualization of predictions and confidence scores.
- Minimal operational overhead for local demos or coursework.

From a delivery standpoint, the system must support different pretrained models, automatically enforce modality compatibility, and expose explanations with consistent semantics. The project is also designed for reproducibility, meaning that structure and configuration are explicit and the code can be understood and extended by students without specialized infrastructure.

3 Architecture Overview

The system follows a modular monolithic architecture centered on a Streamlit application. The user interface, preprocessing, model inference, and XAI generation run within a single Python process, while responsibilities are separated into dedicated modules. This design favors simplicity and rapid iteration, which is appropriate for exploratory explainability work.

1. A user uploads an image or audio file in the UI.

2. The application detects the input modality and filters compatible models.
3. Preprocessing converts raw input into tensors (and spectrograms for audio).
4. A selected model produces class probabilities and a prediction.
5. Grad-CAM, LIME, or SHAP generates an explanation for the same input.
6. Visualizations are rendered side-by-side in the browser.

The architecture emphasizes separation of concerns rather than distributed services. Streamlit pages handle user interaction, while preprocessing and XAI modules encapsulate modality-specific logic. This modularization makes it easier to add models or new explainability techniques without changing the UI contract.

4 Backend and Agentic Design

Implementation is organized around a small number of focused modules and a registry that connects the UI to the model and XAI layers. The main runtime is the Streamlit app in `streamlit/app.py`, which provides navigation and delegates work to the two pages in `streamlit/pages/1_Inference.py` and `streamlit/pages/2_Comparison.py`.

Key modules and directory structure are as follows :

- `models/` stores pretrained model loaders and label metadata, with a single `models/registry.py` used to expose model capabilities.
- `preprocessing/` contains modality-specific pipelines : `preprocessing/image.py` for resizing and normalization, and `preprocessing/audio.py` for waveform-to-spectrogram conversion.
- `xai/` implements explanation methods (`gradcam.py`, `lime.py`, `shap.py`) behind a consistent interface for the UI layer.
- `configs/` holds per-model configuration and example cases in `cases.json`, which are used for reference and evaluation.

The following excerpt summarizes the inference and explanation flow used in both pages (see Listing 1) :

Listing 1 – Unified inference and explanation flow.

```
input_type = detect_input_type(uploaded_file)
model = MODEL_REGISTRY[input_type][model_key]["loader"](device=
    device)
x, image, transform = preprocess(input_type, uploaded_file,
    device)
pred_idx, confidence = predict(model, x)
explanation = explain(xai_method, model, x, image, pred_idx,
    transform)
```

5 Development Choices

Python was selected for its mature machine learning ecosystem and the availability of robust XAI libraries. It supports rapid prototyping while keeping the codebase accessible to students and researchers who may extend the project.

Streamlit provides a lightweight web UI without requiring a separate frontend stack, making it well suited for interactive demos and coursework. PyTorch and Torchvision are used for model definition and inference because of their flexible tensor API and pretrained model support. Librosa handles audio loading and spectrogram extraction, while Pillow and OpenCV provide image manipulation utilities.

The explainability stack relies on Grad-CAM, LIME, and SHAP because they represent complementary attribution paradigms and are widely cited baselines in the XAI literature. This combination enables side-by-side comparison of explanation behaviors across different model architectures.

6 Operations and Limits

The application runs locally via `streamlit run` and does not require external services. Latency is dominated by model inference and explanation generation, with SHAP typically being the slowest method. GPU acceleration is optional : the system automatically selects CUDA when available, otherwise it runs on CPU.

Limitations are primarily computational and data-related. Explanations are only as good as the pretrained models and the preprocessing choices. Audio explanations depend on spectrogram quality, and filename-based ground truth inference in the comparison page is heuristic. Large inputs can also increase memory usage and rendering time.

7 Knowledge and Data Strategy

The project operates on local assets and configuration rather than a centralized database. Model weights are stored under `models/`, while `configs/` provides per-model `cases.json` files used to document representative inputs and expected classes. These files help structure demonstrations and make manual evaluation repeatable.

Input data is treated as transient user uploads. Images are normalized to match the target model's expected input, while audio is converted into spectrogram images prior to inference. This strategy keeps the pipeline consistent with the image-based XAI methods, enabling visual explanations for both modalities.

8 Quality, Evaluation, and Maintenance

Evaluation is primarily qualitative and exploratory. The comparison page enables visual inspection of Grad-CAM, LIME, and SHAP on the same input, which helps identify consistency or disagreement between methods. The `cases.json` fixtures offer a lightweight way to repeat tests and observe regressions after code changes.

Maintenance focuses on updating models, ensuring preprocessing aligns with model expectations, and adjusting the explanation parameters for clarity. Because the codebase is modular, replacing a model or adding a new method only requires updating the registry and adding a new implementation in `xai/`.

9 Roadmap and Future Work

Future improvements include adding quantitative metrics for explainability stability, integrating caching to reduce repeated computation, and supporting batch analysis for datasets. A model management layer could also automate the registration of new checkpoints and ensure consistent metadata.

Another extension is richer reporting, such as exporting visual explanations and predictions into PDF summaries for class projects or audits. These features remain compatible with the current modular structure.

10 Usage of AI

AI was used for the following purposes :

- English grammar-correctness and clarity.
- Working with specialized python libraries. AI agents with web search capacities were able to find the tools that we'd need to use inside of libraries' docs faster than us, to gain time and implement right away the correct methods for what we'd be trying to do.
- Code optimizations. Before reviewing we asked an AI to polish our code, removing unused functions/variables and simplifying certain mechanisms.

11 Conclusion

This project delivers a unified XAI interface that bridges image and audio classification within a single, accessible Streamlit application. The modular architecture clarifies responsibilities between UI, preprocessing, model inference, and explainability, while enabling side-by-side comparison of popular attribution methods. The result is a practical platform for teaching, analysis, and future research extensions.