Unified Explainable AI Platform

Unified Interface for Deepfake Audio Detection
and Lung Cancer Diagnosis

Technical Report

Authors:

Adrien Servas
Alexandre Francony
Léonard Seidlitz
Raphael Roux
Romain Requena

January 2025

GitHub: https://github.com/MwaRPZ/xai_project
Explainable Artificial Intelligence (XAI) Project

Contents

1 Introduction

This project aims to integrate two Explainable Artificial Intelligence (XAI) systems into a single interactive platform capable of processing both audio and medical image data. The goal is to provide a unified interface allowing users to explore, compare, and understand decisions made by different classification models.

1.1 Context

Explainability of deep learning models has become a major concern, particularly in sensitive domains such as deepfake detection and medical diagnosis. This project combines two complementary use cases:

- **Deepfake Audio Detection**: Identification of synthetic vs. authentic audio files
- **Lung Cancer Detection**: Analysis of chest X-rays to identify malignant tumors

1.2 Objectives

The main objectives of the project are:

1. Refactor and integrate two existing repositories into a coherent interface
2. Support audio and image classification using pre-trained models
3. Implement multiple XAI techniques (LIME, SHAP, Grad-CAM, Integrated Gradients)
4. Create a comparison tab for side-by-side evaluation of XAI methods
5. Automate compatibility checks between modalities and XAI methods

2 Architecture and Technical Decisions

2.1 Framework Choice

We chose **PyTorch** as the deep learning backend rather than TensorFlow, primarily for:

- Compatibility with Python 3.14+
- Flexibility for implementing XAI techniques (especially Grad-CAM and Integrated Gradients)
- Native gradient management for attribution methods

The user interface is built with **Streamlit**, offering:

- Rapid development of interactive web interfaces
- Native file upload handling
- Model caching system (`@st.cache_resource`)
- Tab-based organization for easy navigation

2.2 Project Structure

```
XAI_Project/
|-- app.py                      # Main Streamlit application
|-- requirements.txt            # Python dependencies
|-- download_data.py            # Dataset download script
|-- modules/
|   |-- audio_detector/         # Audio detection module
|   |   |-- train.py            # Training (5 epochs, validation)
|   |   |-- preprocess.py       # Spectrogram generation
|   |   +-- lime_explainer.py   # LIME implementation for audio
|   |-- image_detector/         # Image detection module
|   |   |-- train.py            # Training (10 epochs, validation)
|   |   +-- grad_cam.py         # Grad-CAM implementation
|   +-- common/
|       |-- shap_explainer.py   # SHAP for both modalities
|       +-- ig_explainer.py     # Integrated Gradients
+-- models/                     # Trained model weights
```

3 Classification Models

3.1 Common Architecture: MobileNetV2

For both classification tasks, we use **MobileNetV2** with transfer learning from ImageNet. This choice is motivated by:

- **Lightweight**: Parameter-efficient architecture
- **Performance**: Good performance on various classification tasks
- **XAI Compatibility**: Convolutional structure suitable for Grad-CAM and gradient-based methods

The final classification layer is modified to produce 2 classes:

```
model.classifier[1] = nn.Linear(model.last_channel , 2)
```

3.2 Audio Model

Parameter	Value
Architecture	MobileNetV2
Classes	Real (0), Fake (1)
Input	Mel Spectrogram (224×224×3)
Epochs	5
Train/Val Split	80/20
Optimizer	Adam (lr=0.0001)

Table 1: Audio detection model configuration

Audio preprocessing converts `.wav` files to Mel spectrograms using the `librosa` library, then resizes them to 224×224 images compatible with MobileNetV2.

3.3 Image Model (X-rays)

Parameter	**Value**
Architecture	MobileNetV2
Classes	Normal (0), Malignant (1)
Input	Normalized RGB image (224×224)
Epochs	10
Train/Val Split	80/20
Optimizer	Adam (lr=0.0001)
Normalization	ImageNet (mean, std)

Table 2: Lung cancer detection model configuration

4 Explainability Techniques (XAI)

Our platform implements four complementary XAI techniques, each offering a different perspective on model decisions.

4.1 LIME (Local Interpretable Model-agnostic Explanations)

LIME generates local explanations by perturbing the input and observing the impact on predictions. For images (and audio spectrograms), LIME segments the image and identifies the most influential regions.

Application: Audio (via spectrograms)

Implementation: Uses `lime.lime_image` with automatic segmentation and masking perturbation.

4.2 SHAP (SHapley Additive exPlanations)

SHAP uses Shapley values from game theory to attribute importance to each feature. We use `GradientExplainer` for efficient gradient-based approximation.

Application: Audio and Image

Specifics:

- Baseline: black image (zero tensor)
- Visualization: heatmap with "hot" colormap
- Optimized memory management with `gc.collect()`

4.3 Grad-CAM (Gradient-weighted Class Activation Mapping)

Grad-CAM produces activation maps by combining gradients of the target class with activations from the last convolutional layer. This method is particularly suited for CNNs.

Application: Images (X-rays)

Implementation:

```
target_layer = image_model.features[-1]
cam = GradCAM(model=image_model, target_layer=target_layer)
heatmap = cam(input_tensor, class_idx=prediction)
```

4.4 Integrated Gradients

This method computes the integral of gradients along a linear path between a baseline (black image) and the input. It satisfies important axiomatic properties such as completeness and sensitivity.

Application: Audio and Image
Parameters: 30 interpolation steps (precision/performance balance)

4.5 Automatic Method Filtering

The system automatically filters available XAI methods based on input type:

XAI Method	**Audio**	**Image**
LIME	✓	–
SHAP	✓	✓
Grad-CAM	–	✓
Integrated Gradients	✓	✓

Table 3: XAI method compatibility by modality

5 User Interface

The Streamlit interface is organized into three main tabs:

5.1 Deepfake Audio Tab

Allows users to:

1. Upload a `.wav` audio file
2. Play the audio file with an integrated player
3. Select XAI methods to apply
4. Get prediction (REAL/FAKE) with confidence level
5. View explanations in expandable panels

5.2 Lung Cancer Tab

Similar to the audio tab, with:

1. Upload of `.jpg`/`.png` images
2. Display of original X-ray
3. Prediction (NORMAL/MALIGNANT)
4. Grad-CAM visualization overlaid on original image

5.3 Comparison Tab

This tab enables side-by-side comparison of different XAI methods:

- Selection of data type (Audio/Image)
- Column-based display of visualizations
- Clear identification of each method used
- Use of session-stored results to avoid recalculations

6 Improvements Over Original Repositories

6.1 Unification and Modularity

- Merger of two separate projects into a coherent interface
- Modular architecture with clear separation of responsibilities
- Shared code for common XAI methods (SHAP, IG)

6.2 Technical Improvements

- **PyTorch Migration**: Python 3.14+ compatibility, TensorFlow removed
- **OpenCV Removal**: Replaced with PIL/Matplotlib for better portability
- **Enhanced Training**: 80/20 train/validation split, best model saving
- **Memory Management**: Explicit cleanup with `gc.collect()` and `torch.cuda.empty_cache()`

6.3 New Features

- Added **Integrated Gradients** method
- XAI methods **comparison tab**
- **Automatic filtering** of incompatible methods
- **Model caching** for optimal performance
- **State management** (session state) for result persistence

7 Datasets

7.1 Fake or Real Dataset (Audio)

- **Source**: Kaggle (mohammedabdeldayem/the-fake-or-real-dataset)
- **Content**: Original and re-recorded audio files
- **Classification**: "original" → Real, "rerecorded" → Fake

7.2 CheXpert Dataset (Images)

- **Source**: Kaggle (ashery/chexpert)
- **Content**: Annotated chest X-rays
- **Classification**: "No Finding" = 1 → Normal, otherwise → Abnormal

8 Generative AI Usage Statement

In accordance with project requirements, we declare the use of generative AI tools in this work.

8.1 Tools Used

- **ChatGPT** (OpenAI)
- **Gemini** (Google)

8.2 Purposes

1. **Code Implementation**: Assistance with XAI model code
2. **Project Structure**: Advice on module organization
3. **Debugging**: Resolution of technical issues and errors

9 Conclusion

This project successfully created a unified explainable AI platform combining two distinct application domains: deepfake audio detection and medical imaging diagnosis. The main achievements include:

- An intuitive user interface with Streamlit
- Integration of four complementary XAI methods
- A comparison system allowing evaluation of different explainability approaches
- A modular and extensible architecture

The platform demonstrates the importance of explainability in AI systems, particularly for sensitive applications where understanding model decisions is crucial for establishing user trust.