Intel® Unnati 2025 Industrial Training Project

Title :AI-Powered Interactive Learning Assistant for Classrooms

Team Name:EduNova
Team Members:

1. **Adya Gangwal 230962084**
2. **Rajdeep Sahu 230962138**

---

1. Introduction

In recent years, the field of artificial intelligence has witnessed a shift from narrow, single-task systems to more general-purpose, multi-modal assistants capable of understanding and processing diverse forms of input. This project presents a practical implementation of such an assistant capable of handling image classification, object detection, image captioning, speech-to-text, and text generation. The system is built using a modular design and optimized for real-time performance using OpenVINO, making it suitable for deployment on edge devices and low-resource environments.

The project highlights the orchestration of powerful pre-trained models using a streamlined backend and a user-friendly web interface, enabling intuitive interaction with the AI system.

---

2. Objectives

- To build a unified assistant capable of understanding and processing image, audio, and text inputs.
- To leverage open-source pre-trained models for five key tasks:
    - **Image classification** (ViT)
    - **Object detection** (YOLOv8)
    - **Image captioning** (BLIP)
    - **Speech recognition** (Whisper)
    - **Language generation** (Mistral)
- To optimize these models using **OpenVINO** for efficient CPU inference.

- To integrate all functionalities into a **React-based frontend** and a **FastAPI backend** for seamless user experience.

---

3. Architecture Overview

a. Models

Each model is responsible for a specific modality:

- **ViT**: Classifies objects in images.
- **YOLOv8**: Detects and labels objects with bounding boxes.
- **BLIP**: Generates meaningful captions for images.
- **Whisper**: Transcribes speech into text.
- **Mistral**: Generates human-like text responses from prompts.
- **Tesseract**: Extracts printed or handwritten text from images.

b. Backend (FastAPI)

The backend acts as the middleware between the models and the frontend. It handles:

- File uploads and preprocessing
- Calling the appropriate model based on input type
- Formatting and returning the output
- Managing concurrent requests

c. Frontend (React)

The user interface is built using React and supports:

- Uploading images and audio
- Entering text prompts
- Viewing outputs such as captions, detections, and transcriptions
- Switching between tasks with ease

---

4. Development Process

Phase 1: Model Selection and Testing

Robust, open-source models were selected and tested independently to ensure reliability.

Phase 2: Backend Setup

Created FastAPI endpoints for each task. Inputs were preprocessed (e.g., image resizing, audio normalization), and outputs formatted.

Phase 3: Frontend Interface

Developed an interactive UI in React allowing users to upload files, input text, and view results. Axios was used for communication with the backend.

Phase 4: OpenVINO Optimization

Some models were exported to ONNX format and converted to OpenVINO Intermediate Representation (IR), which significantly improved inference performance.

Phase 5: Integration and Testing

Optimized models were integrated into the backend. The full pipeline was tested to ensure accuracy and consistency.

---

5. OpenVINO Integration

Intel's **OpenVINO toolkit** was leveraged to improve performance on CPUs and edge devices. Key benefits observed:

- Faster inference across all tasks
- Lower memory consumption
- Increased responsiveness
- CPU and VPU compatibility

This makes the solution highly suitable for deployment on local desktops, embedded systems, and private environments where GPU availability is limited.

---

6. User Interaction Flow

1. User selects a task (e.g., captioning, transcription).
2. Uploads a file or enters a prompt.
3. Frontend sends the data to the FastAPI backend.

4. Backend preprocesses the input and runs the corresponding OpenVINO-optimized model.
5. Output is sent back and displayed on the UI.

This flow is designed for simplicity and accessibility, even for non-technical users.

---

7. Retrieval-Augmented Generation (RAG) with Supabase

To enhance the quality and contextuality of text responses, we implemented a Retrieval-Augmented Generation (RAG) workflow. When a user submits a query, the system:

1. Converts the query into an embedding vector.
2. Performs a semantic search over previously stored queries using Supabase's vector database.
3. Retrieves the most relevant historical context.
4. Passes that context along with the user's query into the language model (Mistral) to generate a richer, more informed response.

We created a `user_queries` table in Supabase with a `VECTOR` column and indexed it using `ivfflat` for fast cosine similarity search. This setup ensures efficient retrieval while maintaining low latency, even as the number of stored queries grows.

By combining semantic memory with generative AI, the assistant delivers more coherent and context-aware responses, especially during multi-turn interactions.

---

8. Drowsiness Detection with Mediapipe and Face Landmarks

In addition to core multi-modal AI tasks, we developed a **Drowsiness Detection Module** using **Mediapipe** and facial landmark detection to ensure safety and attention in real-time applications (e.g., driving assistance).

How it works:

- **Mediapipe's Face Mesh** detects facial landmarks, especially around the eyes.
- We monitor **eye aspect ratio (EAR)** to detect blinking and eye closure.

- If both eyes remain **closed for more than 50 consecutive frames**, the system triggers a **drowsiness alert**.
- This real-time monitoring is lightweight and suitable for edge deployment.

This module adds an important **real-world safety feature** to our assistant and demonstrates the extensibility of the platform beyond standard vision and language tasks.

---

9. Conclusion

This project demonstrates the potential of **multi-modal AI systems** when combined with intelligent deployment strategies. With the help of **OpenVINO**, we significantly improved performance, enabling real-time responsiveness even on CPU-only systems.

The addition of **RAG-based semantic search** and **drowsiness detection** further showcases the adaptability and extensibility of our platform. A unified React + FastAPI architecture ensures clean modularity and ease of scaling. This assistant is not only functional but also practical for real-world edge applications.

---

10. Limitations

While the system demonstrates strong multi-modal capabilities, there are a few limitations to note:

- **Limited Visual Categories**: Currently, the image classification and captioning modules are fine-tuned only on a small set of educational categories like **history, landmarks, and lab apparatus**. For broader domains or general-purpose images, additional training would be required to ensure accuracy.

- **Educational Scope**: The assistant is primarily designed for **students from Grade 1 to 10**. Higher education levels often demand domain-specific knowledge and more advanced image and document understanding, which are not yet supported in this version.

- **RAG Context Constraints**: Our RAG setup relies on previously stored user queries. If a query is entirely novel or out-of-distribution, the retrieved context may not significantly improve the generation quality.

- **Hardware Assumptions**: Although optimized for CPUs using OpenVINO, the system's performance on extremely low-resource devices may vary based on available memory and processing power.

11 . Individual Contribution

We worked closely together throughout the project, regularly sharing code, testing different approaches, and combining the best results from both sides. Instead of splitting tasks strictly, we each explored different components — like trying out different model setups or optimizing inference — and then merged what worked best.