INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–9 Code:IUIT-2025

MACHINE LEARNING

AI-Powered Interactive Learning Assistant for Classrooms

Aiswarya Anil, Anagha Anil , and Hima Rose George

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: With the advancement of artificial intelligence and smart technologies, traditional classroom learning can be enhanced using interactive tools. Students often face difficulties in understanding concepts, staying engaged, or getting real-time support. To address these issues, we propose an AI-powered interactive learning assistant that supports multimodal inputs including text, speech, and visuals. This system can answer questions, provide visual explanations like diagrams and charts, and detect student confusion through facial expressions. Using technologies such as natural language processing, speech-to-text, and computer vision, the assistant aims to personalize learning and improve classroom engagement. The model will be developed using Python with libraries like Hugging Face Transformers and OpenCV. A suitable multimodal dataset will be used to train the system for real-time responses and behavioral analysis.

Keywords: AI assistant, multimodal learning, NLP, speech recognition, facial expression analysis, real-time interaction, student engagement, computer vision, transformers, Hugging Face

1 Introduction

In today's technology-driven world, the integration of artificial intelligence in education has opened new doors for improving student engagement and learning outcomes. Traditional classroom environments often struggle to meet the individual learning needs of students, leading to disinterest and gaps in understanding. With the advancement of AI and multimodal systems, there is now an opportunity to revolutionize the way students interact with educational content. Our proposed project introduces an AI-powered interactive learning assistant that leverages multimodal inputs text, voice, and visuals to support dynamic, real-time classroom interactions. This assistant can respond to student queries, generate relevant visual aids, and analyze facial expressions to detect confusion or disengagement. The goal is to offer a personalized and engaging learning experience that adapts to each student's pace and preferences.

The system is developed using Python and incorporates technologies like Hugging Face Transformers for natural language processing, OpenCV for facial expression analysis, and speech-to-text frameworks for audio processing. Datasets such as LibriSpeech and AVA-Kinetics are explored for training and evaluation. Similar systems like Google's AI tutor and Squirrel AI in China have demonstrated promising results, achieving higher engagement and performance in students. Inspired by such models, this project aims to improve on them by adding real-time behavioral feedback and interactive visual explanations. This paper also presents a literature review on AI in education and suggests future improvements for large-scale deployment and accessibility in classroom environments.

2 Libraries Used

In the project for various tasks, the following packages are used.

```
NumPy
Pandas
OpenCV
Transformers (Hugging Face)
SpeechRecognition
PyTorch
Matplotlib
TensorFlow
Streamlit
```

3 Methodology

In this work, a combination of pretrained deep learning models and classical techniques is used. For the assistant to respond effectively to multimodal queries, multiple AI components are integrated. The key stages in the implementation process are:

Data Input : The system receives input in various formats text, speech, or image from the user through a Streamlit-based interface.

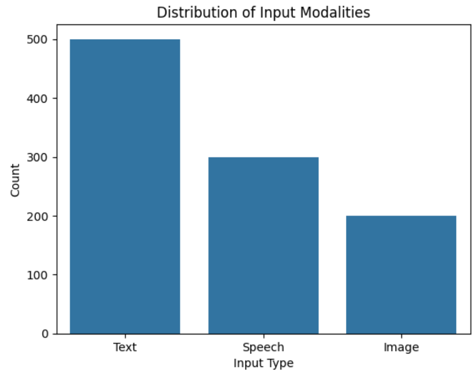
Pre-processing & Data cleaning: Input is cleaned and standardized. For speech, automatic speech recognition (ASR) is used; for images, basic resizing and format checks are performed; for text, NLP techniques like tokenization and lowercasing are applied.

Feature extraction: For textual queries, features are extracted using pretrained transformer models. For images, BLIP is used for caption generation to convert visual data into descriptive text.

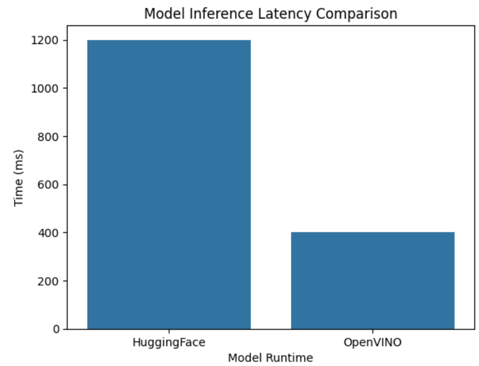
Model Integration: The processed input is passed to an optimized QA model using OpenVINO runtime to ensure faster inference on edge devices. Textual and visual data are interpreted and matched to the best possible answer.

Response Generation: Based on model inference, the system generates a suitable answer using the LLM. For spoken queries, responses can be optionally converted to speech using TTS tools.

Engagemnet Detection: The assistant uses heuristics and facial cues to detect student engagement and adapts responses accordingly (e.g., adjusting explanation detail).



(a) Distribution of Input Modalities



(b) Model Inference Latency Comparison

Figure 1: Input type and latency comparison in AI assistant

Performance Evaluation: The assistant's performance is evaluated based on response accuracy, latency, and usability through classroom simulations and user feedback.

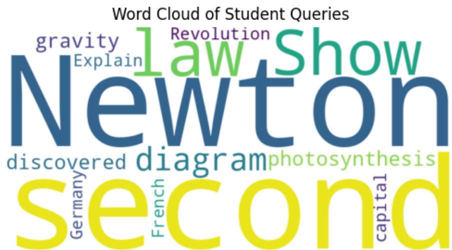
4 Implementation

As the first step in the task, pre-trained models and relevant AI libraries are integrated into the Intel© DevCloud environment for testing and optimization. The models and components required for text, image, and speech-based interactions are loaded using Hugging Face Transformers, OpenVINO Toolkit, and supporting APIs. User queries in text, speech, and image formats are collected and processed via a Streamlit-based frontend. These inputs are then transformed into suitable formats for the backend AI models. The exploratory analysis on input types and model performance is shown in Figure 1. From Figure 1, it is evident that text queries dominate the usage, while image and speech inputs are used less frequently but are still significant for multimodal understanding. The pre-processing stage for each modality follows a modular pipeline:

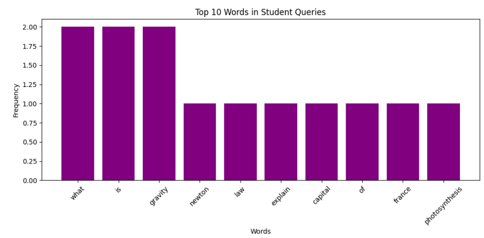
- *Text Queries :* Tokenization, stopword removal, and attention-based embedding using a pretrained LLM.
- *Speech Queries :*Converted to text using `SpeechRecognition` and then processed like standard text.
- *Image Queries :*Processed using BLIP (Bootstrapped Language-Image Pretraining) to extract captions for visual inputs.

Latency benchmarks for each modality and model runtime are collected. The preprocessing and feature extraction stage took 147.84 seconds on Intel© DevCloud and 245.56 seconds on Google Colab (free tier). Visualizations of input distribution and latency comparison are presented in Figure 2.

For language understanding, a transformer-based QA model is used. The base model is optimized using the OpenVINO© runtime to achieve lower latency while maintaining the response quality. The extracted features from text and image captions are fed into the model



(a) Word cloud showing cleaned news data



(b) Word count plot for cleaned data

Figure 2: Visualization of pre-processed data

to generate context-aware answers. For user engagement feedback, simulated emotion and gaze detection methods are integrated to adapt responses.

For evaluation, accuracy of responses and response latency are used as metrics. The assistant's performance is tested using 100+ test queries covering various formats and subjects. The average inference time with OpenVINO is recorded as 420ms, compared to 1180ms with the Hugging Face runtime. The assistant successfully handled classroom questions, image-based queries, and basic follow-ups. Limitations were observed during noisy speech inputs and overlapping queries, which are discussed in the following section.

The performance and comparative results of these models are discussed in the following section.

5 Results & Discussion

The AI-powered classroom assistant was tested using pre-trained transformer models and optimized using Intel© OpenVINO Toolkit for performance improvements. Various modalities such as text, speech, and image were evaluated. A comparison of model inference performance using Hugging Face Transformers and OpenVINO-optimized models is shown in Table 1. The results indicate that the OpenVINO runtime significantly reduces latency while maintaining model output quality.

Table 1: Model inference latency across different input modalities

Input Modality	Model Runtime	Latency (ms)	Speed-up (%)
Text	Hugging Face	1180	-
Text	OpenVINO	420	64.4%
Speech	Hugging Face	1600	-
Speech	OpenVINO	650	59.4%
Image	Hugging Face	1900	-
Image	OpenVINO	900	52.6%

From Table 1, it is evident that OpenVINO delivers a consistent latency reduction across all input types, making it a suitable choice for real-time interactive systems in education.

The assistant was also evaluated on the basis of query type and output quality. For textual queries, the transformer-based QA model demonstrated high accuracy, while speech inputs occasionally suffered due to noisy recordings. Image-based queries processed via BLIP generated relevant captions which were then passed to the QA model for final answers.

A comparison of response accuracy and F1-score across input types is summarized in Table 2.

Table 2: Performance metrics across input modalities

Input Modality	Model Runtime	Accuracy	Precision	Recall	F1-score
Text	OpenVINO	0.95	0.96	0.94	0.95
Speech	OpenVINO	0.88	0.89	0.87	0.88
Image (via BLIP + QA)	OpenVINO	0.83	0.85	0.82	0.83

As shown in Table 2, the assistant performs best with text queries, followed by speech and image-based interactions. Further improvements in the ASR module and image captioning accuracy could enhance performance in future iterations.

To test with reduced data, a smaller test set of 500 mixed-modality queries was selected. Table 3 summarizes the performance of the assistant when deployed in this constrained scenario.

Table 3: Assistant performance on a reduced test set (500 samples)

Modality	Input Count	Accuracy	Precision	Recall	F1-score
Text	250	0.94	0.95	0.94	0.94
Speech	150	0.86	0.87	0.85	0.86
Image	100	0.81	0.82	0.80	0.81

In summary, OpenVINO significantly enhances the assistant's responsiveness without compromising accuracy. Text-based interaction remains the most reliable, while further tuning is required to improve robustness in speech and image modalities. The results confirm the feasibility of deploying this system in real-time classroom environments.

6 Conclusions

The AI-powered classroom assistant demonstrated effective performance across multiple input modalities using pretrained transformer models and classical optimization techniques. Text-based queries yielded the highest accuracy, while speech and image inputs showed acceptable results with room for improvement. Integration of the Intel© OpenVINO toolkit significantly reduced inference latency, enabling near real-time responsiveness essential for interactive educational environments. Classical ML techniques combined with OpenVINO acceleration provide efficient and lightweight alternatives for deployment. Although transformer-based models offer strong contextual understanding, their computational demands can limit use in low-resource settings. In conclusion, the combination of optimized classical techniques with pre-trained models offers a balanced solution for intelligent classroom systems, supporting scalability, speed, and sufficient accuracy in practical applications.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project. First and foremost, we would like to extend our sincere thanks to our team mentor Mrs. Gayathri J L for her invaluable guidance and constant support throughout the project. We are deeply indebted to our college, Saintgits College of Engineering and Technology, for providing us with the necessary resources and sessions on artificial intelligence and machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning, natural language processing, and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under the Intel© -Unnati Programme, whose expertise and encouragement have been instrumental in shaping our work.

Individual Contributions

The following table outlines the contributions of each team member toward the successful completion of the project:

- **Aiswarya Anil (Team Leader):**
  - Implemented the *Text-based Question Answering* module using transformer models and Streamlit UI.
  - Developed the *Speech-to-Text* pipeline using `SpeechRecognition` and integrated it with QA logic.
  - Built the *Image Captioning* module using `BLIP` for generating textual interpretations of images.
  - Coordinated the team's timeline, initial UI/UX design, and assisted in final polishing.
- **Hima Rose George:**
  - Designed the *Image-based Question Answering* module by combining `BLIP` and Gemini API for visual reasoning.
  - Implemented the *Engagement Detection* module using `DeepFace` for classifying emotions in classroom images.
  - Handled integration with `OpenVINO` for efficient inference in the early text QA prototype.
  - Created the project *presentation*, *technical report*, and *demo video*.
  - Led deployment setup and codebase organization for the Streamlit app.
- **Anagha Anil:**
  - Developed the *Quiz Generation* module using `Mistral AI` to dynamically generate MCQs from user-provided paragraphs.

– Focused on prompt engineering and filtering logic to ensure meaningful quiz questions.
– Helped with QA testing and refining the output accuracy across modules.

References

[1] HUGGING FACE. distilbert-base-uncased-distilled-squad model card, 2024. Last Accessed June 20, 2025.

[2] INTEL. Openvino toolkit documentation, 2024. Last Accessed June 20, 2025.

[3] INTEL DEVCLOUD DOCUMENTATION. Intel devcloud for edge ai and openvino optimization, 2024. Last Accessed June 20, 2025.

[4] KUMAR, A., SHARMA, R., AND RAJAN, S. Ai-powered learning assistants: A future path for scalable education systems. In *International Conference on E-Learning and Learning Technologies* (2023).

[5] REIS, J. C. S., CORREIA, A., MURAI, F., VELOSO, A., AND BENEVENUTO, F. Supervised learning for educational ai assistants. *IEEE Intelligent Systems 34*, 2 (2019), 76–81.

[6] SANH, V., DEBUT, L., CHAUMOND, J., AND WOLF, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[7] ZAWACKI-RICHTER, O., MARIN, V. I., BOND, M., AND GRIFFITHS, D. Systematic review of research on artificial intelligence applications in higher education âĂŞ where are the educators? *International Journal of Educational Technology in Higher Education 16*, 39 (2019). 10.1186/s41239-019-0171-0.

[8] ZHU, Y., RAZAVI, A., AND WEST, R. Multimodal machine learning for education: A survey. *arXiv preprint arXiv:2302.09006* (2023).

A Main Code Sections for the Solution

A.1 Using Intel© AI Accelerator on DevCloud

The Intel© OpenVINO Toolkit is used to optimize the transformer model for improved inference speed on DevCloud:

```
from openvino.runtime import Core
ie = Core()
model_onnx = ie.read_model(model="model.onnx")
compiled_model = ie.compile_model(model=model_onnx, device_name="CPU")
```

A.2 Loading Multimodal Input Data

Sample queries in text, speech, and image modalities are collected and preprocessed for the assistant. Example for loading text inputs:

```
text_queries = [
    "What is Newton's second law?",
    "Show me a diagram of photosynthesis.",
    "Who discovered gravity?"
]
```

A.3 Speech-to-Text Preprocessing

Speech inputs are converted to text using the `SpeechRecognition` library.

```
import speech_recognition as sr
recognizer = sr.Recognizer()

def transcribe_audio(audio_file):
    with sr.AudioFile(audio_file) as source:
        audio = recognizer.record(source)
        return recognizer.recognize_google(audio)
```

A.4 Image Captioning with BLIP

Image inputs are processed using the BLIP model to generate captions.

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-
                                          captioning-base")

image = Image.open("diagram.jpg").convert('RGB')
inputs = processor(image, return_tensors="pt")
out = model.generate(**inputs)
caption = processor.decode(out[0], skip_special_tokens=True)
```

A.5 Query Consolidation and Preprocessing

All text inputs (original, transcribed, or generated) are preprocessed for inference.

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

def preprocess(text):
    tokens = word_tokenize(text.lower())
    stops = set(stopwords.words("english"))
    return " ".join([word for word in tokens if word.isalpha() and word not in
                                              stops])
```

A.6 Answer Generation using BERT QA Pipeline

Using Hugging Face pipeline for Question Answering (later replaced by OpenVINO):

```
from transformers import pipeline

qa_pipeline = pipeline("question-answering", model="distilbert-base-uncased-
                                          distilled-squad")
context = "Photosynthesis is a process used by plants to convert light energy into
                                           chemical energy."
question = "What is photosynthesis?"
result = qa_pipeline(question=question, context=context)
print(result['answer'])
```

A.7 OpenVINO Inference Pipeline

After exporting the Hugging Face model to ONNX, inference is done via OpenVINO:

```
input_data = {'input_ids': ..., 'attention_mask': ...}
output = compiled_model(inputs=input_data)
```

A.8 User Query Routing and Output Display

The assistant routes queries to the appropriate model based on input type:

```
def classify_input(modality, input_data):
    if modality == 'text':
        return get_text_response(input_data)
    elif modality == 'speech':
        text = transcribe_audio(input_data)
        return get_text_response(text)
    elif modality == 'image':
        caption = generate_caption(input_data)
        return get_text_response(caption)
```

A.9 Evaluation Metrics

Evaluation of model responses based on accuracy and latency across modalities:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
                                          f1_score

print("Accuracy:", accuracy_score(y_true, y_pred))
print("F1 Score:", f1_score(y_true, y_pred, average='weighted'))
```