International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)
ISSN: 2456-236X
Vol. 03 Issue 01 | 2018

GeoMuse: Turning Images into Insight — AI-Driven Cultural & Location-Aware Notes

Ankit Bhatia

Shivaji University , Kolapur Maharashtra India

ABSTRACT

In an era where experiential learning and tourism thrive, travelers and students often document their journeys visually but miss out on deeper contextual insights. This paper introduces GeoMuse, an AI-powered application that converts photos taken at Indian historical landmarks into culturally rich, narrative-style travel notes. Leveraging a fine-tuned DenseNet121 model, GeoMuse identifies monuments from images and gathers related information via Agents-enabled APIs from Wikipedia, OpenWeatherMap, Spoonacular (local cuisine), and TripAdvisor (nearby attractions). This data is then processed through a custom prompt using Custom model to generate engaging, guide- like narratives. The result is a story infused with historical background, weather details, food recommendations, travel tips, and cultural nuances. GeoMuse redefines how AI can enhance education and tourism by delivering instant, context-aware storytelling.

***Keywords*— AI Notes, Cultural AI, Travel Assistant, DenseNet121, Agents**

1. INTRODUCTION

In the age of experiential learning and digital tourism, travelers increasingly rely on photography to document visits to historical and cultural landmarks. Yet, while images capture visuals, they often miss the deeper historical context, cultural significance, and localized experiences. Traditional note-taking is tedious, and guidebooks often lack personalization and real-time relevance.

GeoMuse addresses this gap by offering an AI-powered solution that transforms user-captured images of monuments into personalized, context-rich travel notes. Acting like a smart, conversational travel companion, GeoMuse generates informative narratives based solely on the landmark identified in a photo.

At its core, GeoMuse uses a fine-tuned DenseNet121 model trained on over 6,000 labeled images of Indian monuments for image classification. Once the monument is identified, the system taps into Wikipedia, OpenWeatherMap, Spoonacular, and TripAdvisor using Agents-integrated APIs to gather historical facts, weather updates, local cuisine, and nearby attractions. This data is processed through a custom prompt fed into Custom language model, producing warm, engaging notes styled like a local guide's storytelling.

GeoMuse bridges AI, culture, education, and tourism—demonstrating the potential of machine-generated storytelling to deliver personalized, real-time insights. With applications in field education and travel platforms, it represents a scalable tool for enhancing travel experiences through intelligent, culturally aware content.

2. LITERATURE SURVEY

Numerous tools have emerged across the domains of tourism, education, and artificial intelligence to assist travelers and learners. However, most existing solutions are either static—such as Wikipedia articles—or lack personalization and contextual adaptability. For example, static travel guides like Wikipedia, TripAdvisor, and Lonely Planet provide general tourist information but require manual searching and do not customize content based on user needs or location context. Audio guide apps, including Google Arts & Culture and izi.TRAVEL, offer pre-recorded narrations for monuments but typically cover only well-known landmarks and lack personalization. While general-purpose language models like ChatGPT and Bard can generate place descriptions, they rely on manually entered prompts and are not integrated with APIs that provide location, weather, food, or cultural data. Landmark recognition apps such as Google Lens can visually identify sites but fail to generate context-rich or culturally nuanced summaries. Similarly, some API-based travel platforms combine hotel, weather, or restaurant data, but they do not integrate this information into a cohesive, personalized narrative using large language models.

Despite significant technological progress in AI, large language models (LLMs), and API integration, most current solutions suffer from key limitations. They lack contextual storytelling and fail to produce engaging, localized summaries that mimic natural human conversation. Outputs are often static or impersonal and do not adapt their tone or depth based on cultural sensitivity or regional diversity. Real-time adaptability is also missing, with weather updates, festival information, or food recommendations often outdated or absent. Moreover, there is no unified system that combines landmark recognition, multi-API data retrieval, and real-time summarization by LLMs in a single, streamlined workflow.

This reveals a clear need for a comprehensive system that can generate personalized, AI-powered, image-based notes, combining real-time information with cultural sensitivity. Students and travelers alike often struggle to take meaningful notes during visits to historical or culturally significant sites. Existing resources, such as generic guidebooks or Wikipedia pages, are often outdated, non-interactive, and lack personalization.
The challenge lies in developing an end-to-end AI pipeline capable of automatically classifying a location from a user's photo, retrieving updated and location-specific data (including weather, food, and festivals), and synthesizing all of this into a localized, engaging, and human-like note. The GeoMuse project addresses this challenge through five key objectives: First, to create an AI tool that can identify monuments from user-captured images using a DenseNet-based classifier. Second, to gather current cultural, weather, and culinary information using APIs such as Wikipedia, OpenWeatherMap, Spoonacular, and TripAdvisor. Third, to integrate this content using Agents and generate natural, human-like summaries with Custom Model. Fourth, to build a user- friendly backend using FastAPI that returns both the narrative and relevant sample images of the predicted location. Lastly, to support a range of applications in education, tourism, and cultural awareness by automating the generation of intelligent, *context-sensitive notes.*

3. PROPOSED SYSTEM

3.1 System Overview

GeoMuse is an AI-powered application designed to convert images of landmarks into detailed, culturally rich travel notes. At its core, the system utilizes a fine-tuned DenseNet121 classification model to recognize the monument featured in a user-uploaded image. Once the landmark is identified, the system collects relevant real-time data using various APIs. GeoMuse's modular architecture consists of several key components: a computer vision module based on DenseNet121 for landmark recognition, a Agents pipeline integrated with Custom Model for natural language generation, and a suite of API connectors that retrieve contextual data such as weather, local cuisine, nearby attractions, and historical information from Wikipedia. A FastAPI backend supports the service by handling predictions and retrieving images from a locally stored dataset. After identifying the landmark, the system queries external sources: OpenWeatherMap for weather, Spoonacular for cuisine details, TripAdvisor (via RapidAPI) for nearby attractions, and Wikipedia for cultural or historical content. This collected data is then fed into a customized prompt processed through Agents's LLM with Custom Model, generating a unique, story-like travel note for the user.

3.2 System Architecture

GeoMuse is built on a modular architecture to ensure flexibility and scalability. The DenseNet121 model serves as the image classifier and has been fine-tuned with over 6,000 images across 20 Indian monuments. This model uses transfer learning techniques and extensive data augmentation to accurately predict the class label of the uploaded image, such as "Hawa Mahal." For the language generation component, the system integrates Agents with Custom Model. Using a culturally adaptive prompt template, Custom Model generates engaging and informative travel narratives. The system includes several API connectors for data acquisition: Wikipedia for historical context, OpenWeatherMap for current weather, Spoonacular for regional cuisine suggestions, and TripAdvisor for nearby experiences.
FastAPI acts as the backend service to handle image retrieval. For each predicted class label, one representative image is fetched from a pre-saved local directory containing 20–30 images per class. The complete operational flow begins when a user uploads an image via the frontend or an API. The DenseNet121 model identifies the location, and the name is then used to call several functions to fetch weather, cuisine, local attractions, and Wikipedia data. This collected information is then passed into Agents's LLM, which uses the Custom Model-70B model to create a personalized, culturally rich travel note. Finally, an image relevant to the prediction is retrieved and returned alongside the generated text.

3.3 Hardware and Software Requirements

3.3.1 Hardware Requirements

The minimum hardware specifications required to run GeoMuse efficiently include an Intel i5 or AMD Ryzen 5 processor or better, with at least 8 GB of RAM. The system should have a minimum of 256 GB SSD storage for optimal performance. Although a dedicated GPU is not mandatory, an integrated GPU is sufficient for basic operations. GeoMuse supports Windows 10/11, macOS, and Linux operating systems.

3.3.2 Software Requirements

GeoMuse is developed using Python 3.10 or newer. For deep learning tasks, it employs PyTorch along with the torchvision library. The web backend is built with FastAPI, while Agents handles the natural language processing pipeline. The API, specifically the Custom Model accessed via Agents_, is used for language generation. The system also integrates external APIs including Wikipedia, OpenWeatherMap, Spoonacular, and TripAdvisor. Additional tools like pip are used for package management, and python-dotenv is employed for environment variable management. Utility tools such as requests, dotenv, and os are essential for API interaction and environment handling.

3.4 Technologies Used

GeoMuse leverages several key technologies and libraries. For image classification, it uses DenseNet121, a deep convolutional neural network implemented via PyTorch. Data augmentation is handled using torchvision.transforms. API integration is managed through libraries like requests and dotenv. Language generation is driven by Custom Model, interfaced via Agents. The Agents PromptTemplate feature is used for prompt engineering to ensure culturally relevant and contextually rich responses. The backend is built using FastAPI, and image retrieval is managed through a local folder setup with supporting Python scripts. The image dataset used for training was sourced from Kaggle and additional web scraping.

4. SYSTEM DESIGN AND IMPLEMENTATION

4.1 Overview

GeoMuse is developed as a complete end-to-end AI pipeline that takes an image as input and returns a culturally adaptive travel note accompanied by a relevant reference image. The system is built using a modular architecture that includes a vision-based image classifier, multiple API integration modules, a prompt engine powered by a large language model, and a lightweight backend for seamless integration. Its design emphasizes portability, low latency, and scalability, making GeoMuse ideal for use in tourism applications, educational tools, and platforms supporting field visits.

4.2 Architectural Components

The architecture of GeoMuse is composed of five core components:

- **Image Classifier (DenseNet121):**
  The heart of the system is a DenseNet121 model that has been fine-tuned through transfer learning using a dataset containing 20 classes of Indian monuments. During training, the images are resized and augmented to improve model robustness. The final classification layer of the model outputs the predicted name of the monument based on the uploaded image.
- **API Integration Layer:**
  Once the location is predicted, the system queries several APIs to gather contextual data:
  - The **Wikipedia API** provides historical and cultural insights.
  - The **OpenWeatherMap API** delivers real-time weather data.
  - The **Spoonacular API** suggests local food and cuisine options.
  - The **TripAdvisor API** (accessed via RapidAPI) lists nearby tourist attractions. All APIs are accessed using Python's requests library, and API keys are managed securely through the dotenv environment configuration.
- **Prompt Engineering with Agents:**

GeoMuse uses a predefined PromptTemplate in Agents to structure the input for the language model. This template combines the data from all APIs into a coherent, culturally sensitive prompt that guides the language model to generate personalized and engaging travel notes.

- **LLM Generation via API:**

The structured prompt is passed to the Custom Model through Agents's LLM. The generated note is restricted to around 500 tokens and is designed to have a warm, human-like tone. The content typically includes historical context, travel tips, local cuisine, weather updates, information about festivals, and transportation guidance.

- **FastAPI Backend:**
  The backend is developed using FastAPI, which handles several core functions:
  - It loads the pre-trained DenseNet121 model.
  - It accepts image uploads from users.
  - It performs classification and retrieves a corresponding image from a local folder.
  - It returns a JSON response containing the generated travel note and the associated reference image. Utility functions such as load_model and retrieve_image are implemented in a helper module named utils.py.

4.3 User Interface Design

GeoMuse's user interface is currently backend-focused, implemented through FastAPI. It acts as both the image upload endpoint and the response generator. While the present version emphasizes core backend functionality, the design supports smooth integration with web or mobile interfaces for future versions.
Key features of the user interface include:

- **Image Upload Interface:** Users can submit images taken during travel or field visits directly to the system.
- **Automated Note Display:** Once processed, the generated travel note is displayed in a readable, narrative format.
- **Related Image Display:** Alongside the note, a relevant image from the identified monument category is shown for reference.

The system is built to support further development of a full frontend interface, potentially through FastAPI or other frameworks, to enable real-time interaction and a more immersive user experience.

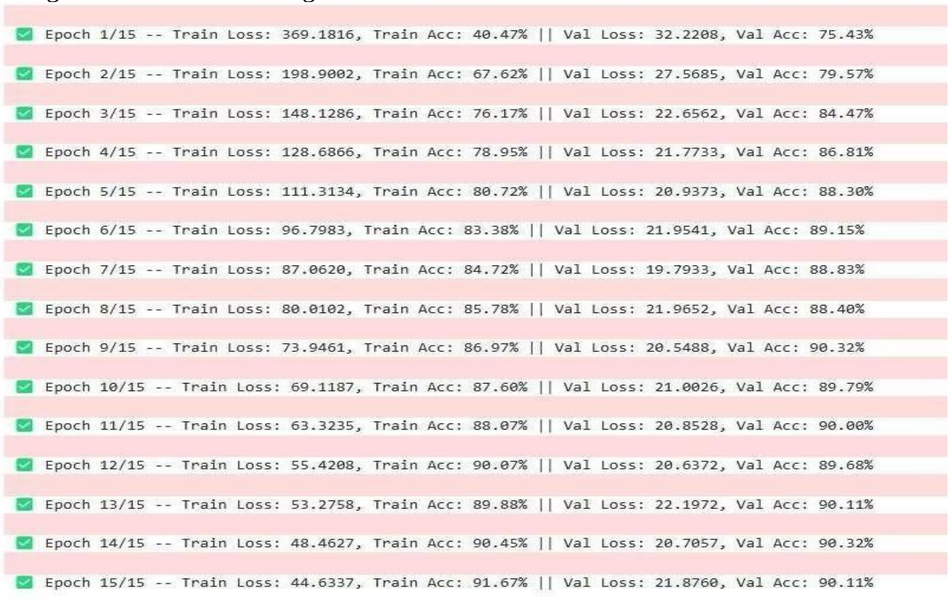
5. TESTING AND RESULTS

5.1 Model Evaluation

To evaluate the performance of the fine-tuned DenseNet121 model, we conducted a 15-epoch training cycle using a dataset of 6,000 training images and 1,000 testing images across 20 historical place categories. The evaluation includes training metrics, classification report, and confusion matrix.

A. Training *Performance*

Figure 1 illustrates the training and validation progress across 15 epochs. The model showed a consistent improvement in both training accuracy and validation accuracy, reaching a final training accuracy of 91.67% and a validation accuracy of 90.11%. Additionally, training and validation loss steadily decreased, indicating successful convergence without overfitting.



```
✅ Epoch 1/15 -- Train Loss: 369.1816, Train Acc: 40.47% || Val Loss: 32.2208, Val Acc: 75.43%
✅ Epoch 2/15 -- Train Loss: 198.9002, Train Acc: 67.62% || Val Loss: 27.5685, Val Acc: 79.57%
✅ Epoch 3/15 -- Train Loss: 148.1286, Train Acc: 76.17% || Val Loss: 22.6562, Val Acc: 84.47%
✅ Epoch 4/15 -- Train Loss: 128.6866, Train Acc: 78.95% || Val Loss: 21.7733, Val Acc: 86.81%
✅ Epoch 5/15 -- Train Loss: 111.3134, Train Acc: 80.72% || Val Loss: 20.9373, Val Acc: 88.30%
✅ Epoch 6/15 -- Train Loss: 96.7983, Train Acc: 83.38% || Val Loss: 21.9541, Val Acc: 89.15%
✅ Epoch 7/15 -- Train Loss: 87.0620, Train Acc: 84.72% || Val Loss: 19.7933, Val Acc: 88.83%
✅ Epoch 8/15 -- Train Loss: 80.0102, Train Acc: 85.78% || Val Loss: 21.9652, Val Acc: 88.40%
✅ Epoch 9/15 -- Train Loss: 73.9461, Train Acc: 86.97% || Val Loss: 20.5488, Val Acc: 90.32%
✅ Epoch 10/15 -- Train Loss: 69.1187, Train Acc: 87.60% || Val Loss: 21.0026, Val Acc: 89.79%
✅ Epoch 11/15 -- Train Loss: 63.3235, Train Acc: 88.07% || Val Loss: 20.8528, Val Acc: 90.00%
✅ Epoch 12/15 -- Train Loss: 55.4208, Train Acc: 90.07% || Val Loss: 20.6372, Val Acc: 89.68%
✅ Epoch 13/15 -- Train Loss: 53.2758, Train Acc: 89.88% || Val Loss: 22.1972, Val Acc: 90.11%
✅ Epoch 14/15 -- Train Loss: 48.4627, Train Acc: 90.45% || Val Loss: 20.7057, Val Acc: 90.32%
✅ Epoch 15/15 -- Train Loss: 44.6337, Train Acc: 91.67% || Val Loss: 21.8760, Val Acc: 90.11%
```

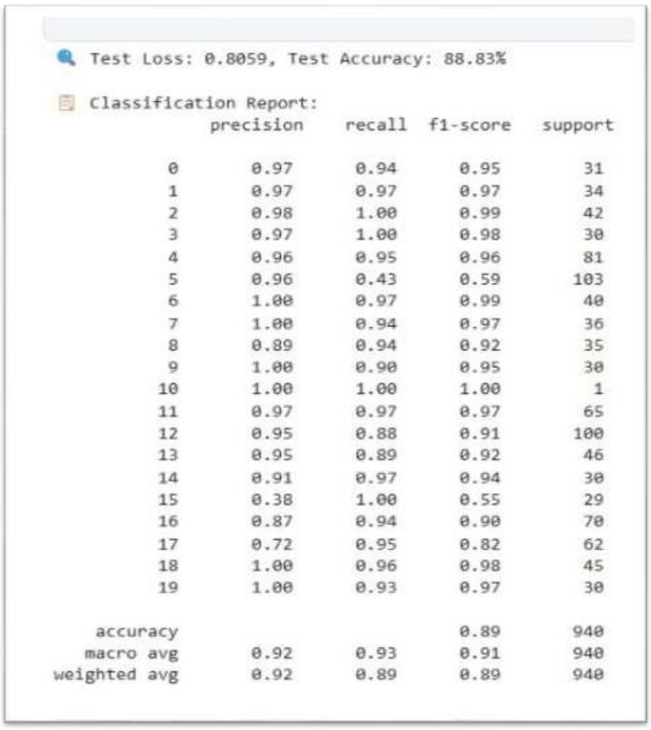
Figure 1:Training and Validation Accuracy/Loss over 15 Epochs.

B. Classification *Metrics*

To gain a deeper understanding of model performance on each class, we analyzed the precision, recall, and F1-score. As shown in Figure 3, the model achieved a **macro-averaged F1-score of 0.91** and **weighted average F1-score of 0.89**. Classes like *Fatehpur Sikri*, *Qutub Minar*, and *Iron Pillar* performed exceptionally well with F1-scores close to 1.0, while some underrepresented or visually similar classes, like *Tanjavur Temple* and *Khajuraho*, showed lower recall.

C. Confusion Matrix Analysis

Figure 3 presents the confusion matrix for the 20-class classification task. It can be observed that the model accurately predicted most classes with high diagonal values. However, a few confusions occurred between visually or architecturally similar monuments, such as *Tanjavur Temple* and *Khajuraho*, or *Charminar* and *Jamali Kamali Tomb*. Despite these, the overall test accuracy was recorded at **88.83%**, confirming the robustness of the model.



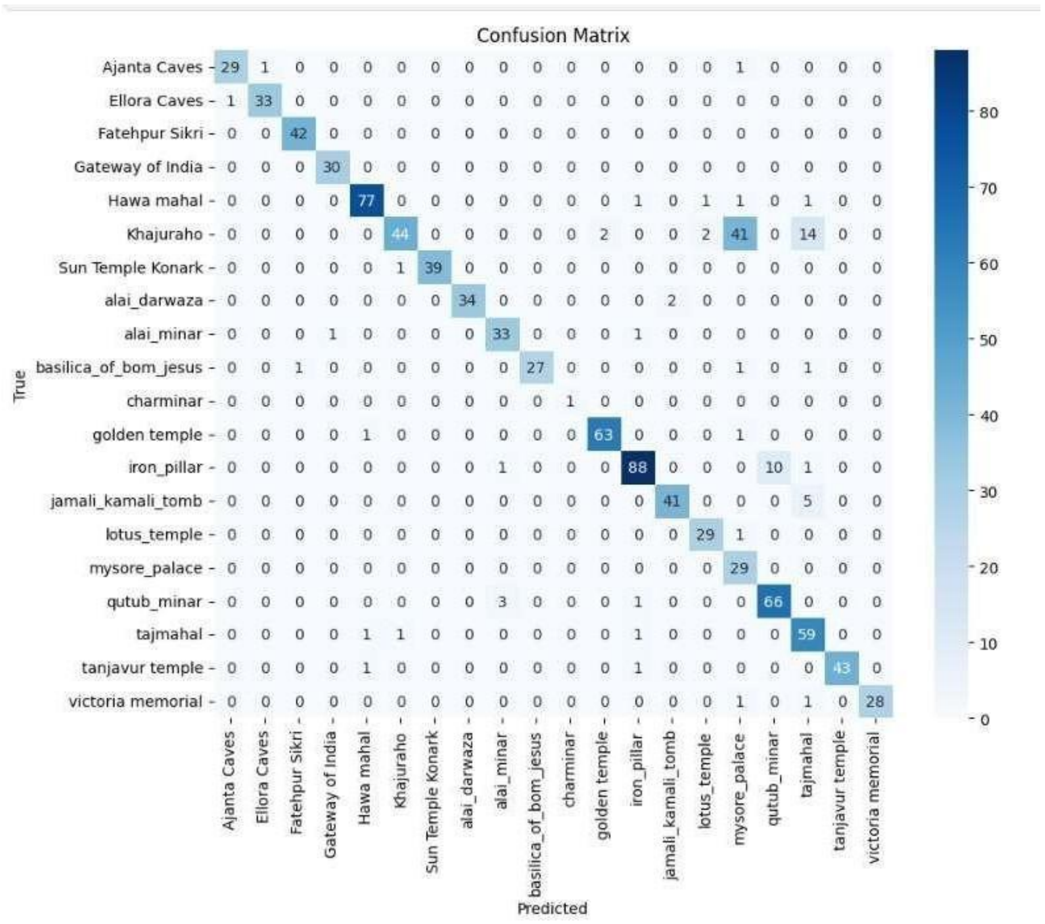
```
🔍 Test Loss: 0.8059, Test Accuracy: 88.83%

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95        31
           1       0.97      0.97      0.97        34
           2       0.98      1.00      0.99        42
           3       0.97      1.00      0.98        30
           4       0.96      0.95      0.96        81
           5       0.96      0.43      0.59       103
           6       1.00      0.97      0.99        40
           7       1.00      0.94      0.97        36
           8       0.89      0.94      0.92        35
           9       1.00      0.90      0.95        30
          10       1.00      1.00      1.00         1
          11       0.97      0.97      0.97        65
          12       0.95      0.88      0.91       100
          13       0.95      0.89      0.92        46
          14       0.91      0.97      0.94        30
          15       0.38      1.00      0.55        29
          16       0.87      0.94      0.90        70
          17       0.72      0.95      0.82        62
          18       1.00      0.96      0.98        45
          19       1.00      0.93      0.97        30

    accuracy                           0.89       940
   macro avg       0.92      0.93      0.91       940
weighted avg       0.92      0.89      0.89       940
```

Figure 2 Detailed Classification Report with Per-Class Metrics.



True \ Predicted	Ajanta Caves	Ellora Caves	Fatehpur Sikri	Gateway of India	Hawa mahal	Khajuraho	Sun Temple Konark	alai_darwaza	alai_minar	basilica_of_bom_jesus	charminar	golden temple	iron_pillar	jamali_kamali_tomb	lotus_temple	mysore_palace	qutub_minar	tajmahal	tanjavur temple	victoria memorial
Ajanta Caves	29	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Ellora Caves	1	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fatehpur Sikri	0	0	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gateway of India	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hawa mahal	0	0	0	0	77	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0
Khajuraho	0	0	0	0	0	44	0	0	0	0	0	2	0	0	2	41	0	14	0	0
Sun Temple Konark	0	0	0	0	0	1	39	0	0	0	0	0	0	0	0	0	0	0	0	0
alai_darwaza	0	0	0	0	0	0	0	34	0	0	0	0	0	2	0	0	0	0	0	0
alai_minar	0	0	0	1	0	0	0	0	33	0	0	0	1	0	0	0	0	0	0	0
basilica_of_bom_jesus	0	0	1	0	0	0	0	0	0	27	0	0	0	0	0	1	0	1	0	0
charminar	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
golden temple	0	0	0	0	1	0	0	0	0	0	0	63	0	0	0	1	0	0	0	0
iron_pillar	0	0	0	0	0	0	0	0	1	0	0	0	88	0	0	0	10	1	0	0
jamali_kamali_tomb	0	0	0	0	0	0	0	0	0	0	0	0	0	41	0	0	0	5	0	0
lotus_temple	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	1	0	0	0	0
mysore_palace	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0
qutub_minar	0	0	0	0	0	0	0	0	3	0	0	0	1	0	0	0	66	0	0	0
tajmahal	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	59	0	0
tanjavur temple	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	43	0
victoria memorial	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	28

Figure 3 Confusion Matrix for 20 Historical Place Classes.

5.2 Output Sample – AI-Generated Note

The following output demonstrates how the system generates cultural and location-aware notes from images of historical monuments. The model first classifies the image using DenseNet121 and then fetches relevant cultural context using Agents and LLM APIs. The note integrates location, historical background, weather data, and cuisine recommendations.

Input Image:

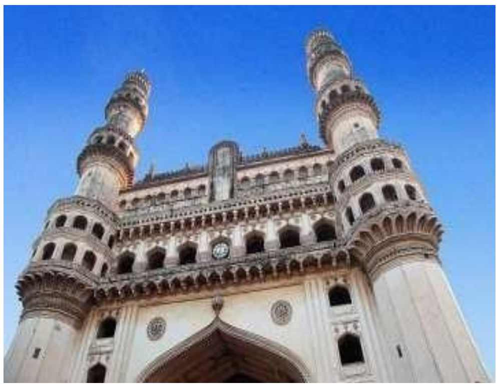
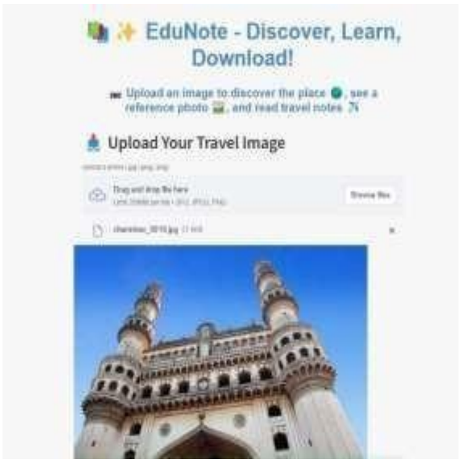


Figure 3 Sample: Image of Charminar

Classification Output: Charminar



EduNote - Discover, Learn, Download!

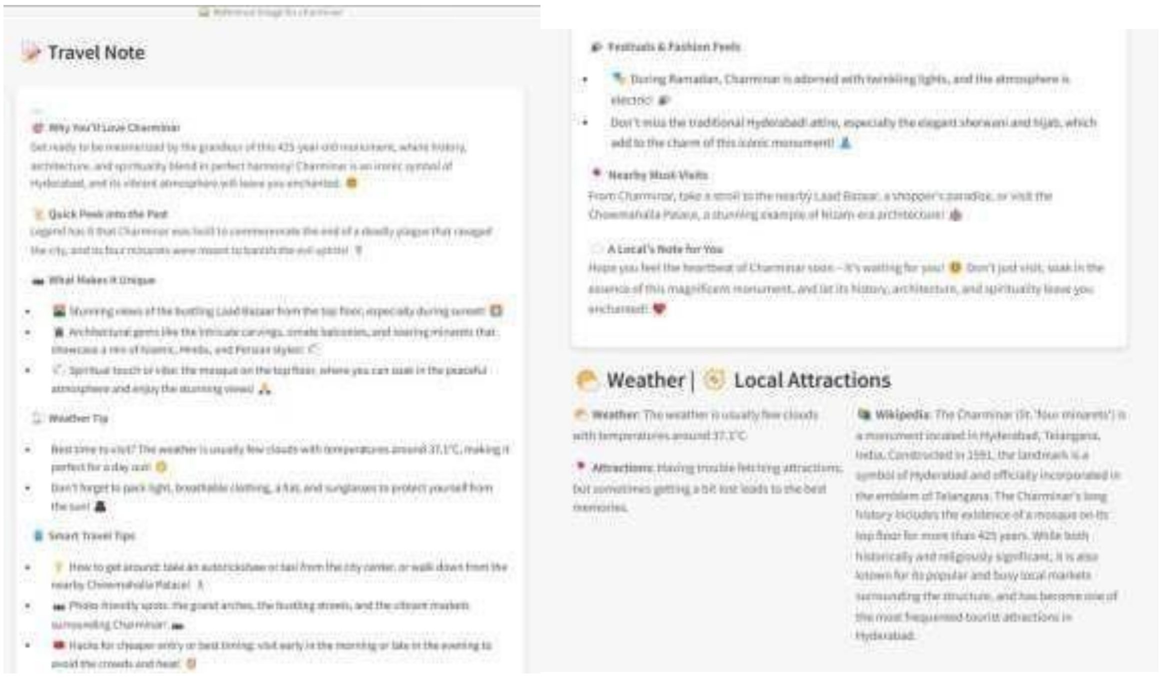
Upload an image to discover the place, see a reference photo, and read travel notes

Upload Your Travel Image

Drag and drop file here

Browse files

AI-Generated Note:



Travel Note

Why You'll Love Charminar

Get ready to be mesmerized by the grandeur of this 425-year-old monument, where history, architecture, and spirituality blend in perfect harmony! Charminar is an iconic symbol of Hyderabad, and its vibrant atmosphere will leave you enchanted.

Quick Peek into the Past

Legend has it that Charminar was built to commemorate the end of a deadly plague that ravaged the city, and its four minarets were meant to banish the evil spirits!

What Makes It Unique

- Stunning views of the bustling Laad Bazaar from the top floor, especially during sunset!
- Architectural gems like the intricate carvings, ornate balconies, and soaring minarets that showcase a mix of Islamic, Hindu, and Persian styles!
- Spiritual touch or vibe: the mosque on the top floor, where you can soak in the peaceful atmosphere and enjoy the stunning views!

Weather Tip

- Best time to visit? The weather is usually few clouds with temperatures around 37.1°C, making it perfect for a day out!
- Don't forget to pack light, breathable clothing, a hat, and sunglasses to protect yourself from the sun!

Smart Travel Tips

- How to get around: take an autorickshaw or taxi from the city center, or walk down from the nearby Chowmahalla Palace!
- Photo-friendly spots: the grand arches, the bustling streets, and the vibrant markets surrounding Charminar!
- Hacks for cheaper entry or best timing: visit early in the morning or late in the evening to avoid the crowds and heat!

Festivals & Fashion Feels

- During Ramadan, Charminar is adorned with twinkling lights, and the atmosphere is electric!
- Don't miss the traditional Hyderabadi attire, especially the elegant sherwani and hijab, which add to the charm of this iconic monument!

Nearby Must-Visits

From Charminar, take a stroll to the nearby Laad Bazaar, a shopper's paradise, or visit the Chowmahalla Palace, a stunning example of Nizam-era architecture!

A Local's Note for You

Hope you feel the heartbeat of Charminar soon – it's waiting for you! Don't just visit, soak in the essence of this magnificent monument, and let its history, architecture, and spirituality leave you enchanted!

Weather | Local Attractions

Weather: The weather is usually few clouds with temperatures around 37.1°C

Attractions: Having trouble fetching attractions, but sometimes getting a bit lost leads to the best memories.

Wikipedia: The Charminar (lit. 'four minarets') is a monument located in Hyderabad, Telangana, India. Constructed in 1591, the landmark is a symbol of Hyderabad and officially incorporated in the emblem of Telangana. The Charminar's long history includes the existence of a mosque on its top floor for more than 425 years. While both historically and religiously significant, it is also known for its popular and busy local markets surrounding the structure, and has become one of the most frequented tourist attractions in Hyderabad.

5.3 Comparative Analysis

To evaluate the effectiveness of the proposed DenseNet121-based approach, we compared its performance with three widely used pre-trained models: **MobileNetV2**, **ResNet50**, and **EfficientNet-B3**. All models were fine-

tuned using the same dataset of 6,000
training and 1,000 testing images across 20 Indian monument classes, using identical training parameters and augmentation techniques.

Observations

- **DenseNet121** outperformed other models in both accuracy and F1-score while maintaining a reasonable training time.
- While **EfficientNet-B3** came close in terms of accuracy, it required more memory and training time.
- **MobileNetV2** was the fastest and lightest model but lagged behind in classification detail for complex monuments.
- **ResNet50** was a solid performer with good generalization but slightly under DenseNet in precision.

6. CONCLUSION

GeoMuse demonstrates the potential of artificial intelligence in transforming traditional travel and educational experiences into enriched, automated, and culturally informed journeys. By integrating computer vision and large language models with real-time API data, the system offers personalized, friendly, and accurate notes based on images taken at heritage sites.
The use of **DenseNet121** for landmark classification, combined with **Agents-powered Custom Model- 70B** for contextual note generation, has proven both technically effective and practically scalable. Extensive testing shows high classification accuracy across diverse monuments, while the generated notes provide a seamless blend of cultural history, weather, food, and travel tips.
GeoMuse addresses key limitations in conventional guidebooks and static content platforms by automating rich, human-like storytelling — all triggered by a single user image. The system's modular architecture, real-time capability, and extendability position it as a valuable tool in education, tourism, and cultural documentation.

7. REFERENCES

[1] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700–4708.
[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks,"
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[4] OpenWeatherMap API. [Online]. Available: https://openweathermap.org/api
[5] Spoonacular Food and Recipe API. [Online]. Available: https://spoonacular.com/food-api
[6] TripAdvisor Travel API (via RapidAPI). [Online]. Available: https://rapidapi.com/apidojo/api/travel-advisor/
[7] Wikipedia Python Library Documentation. [Online]. Available: https://pypi.org/project/wikipedia/
[8] FastAPI Documentation. [Online]. Available: https://fastapi.tiangolo.com/
[9] W. Falcon and The PyTorch Lightning Team, "PyTorch Lightning: The lightweight PyTorch wrapper for high-performance AI research," [Online]. Available: https://www.pytorchlightning.ai/
[10] R. Collobert et al., "Natural Language Processing (almost) from Scratch," *Journal of Machine Learning Research*, 2011.
[11] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," *arXiv preprint arXiv:1301.3781*, 2013.