ChatMoonVLM Report

Okan Berhoğlu

January 11, 2026

Contents

Abstract

This report details the design and implementation of ChatMoonVLM, a web-based Visual Question Answering (VQA) system powered by the efficient Moondream vision-language model. Addressing the need for accessible and privacy-preserving AI solutions, the project leverages a lightweight model architecture capable of running on local hardware without reliance on external APIs. The application is built using the Streamlit framework to provide an intuitive user interface for image uploading and interactive querying. Furthermore, the entire system is containerized with Docker to ensure consistent deployment and reproducibility across different environments. Key features include persistent chat history, resource-aware model loading, and a modular architecture. This work demonstrates the practical viability of deploying efficient, open-source vision-language models for real-world applications on consumer-grade infrastructure.

1 Introduction

Recent advancements in Artificial Intelligence have led to the development of powerful Vision Language Models (VLMs) capable of understanding and generating text based on visual inputs. Models such as GPT-4V and Gemini have demonstrated remarkable capabilities in tasks ranging from image captioning to complex visual reasoning. Despite their performance, these state-of-the-art models often present significant challenges for practical deployment. Many are proprietary, limiting accessibility and research transparency. Furthermore, their substantial computational requirements make them unsuitable for resource-constrained environments, such as edge devices or local applications where latency and privacy are critical concerns. This necessitates the exploration of efficient, open-source alternatives that balance performance with resource utilization.

Moondream emerges as a compelling solution in this landscape. It is a small, open-source vision language model designed to run efficiently on consumer-grade hardware and even mobile devices. By leveraging a compact architecture, Moondream significantly reduces the computational overhead associated with traditional VLMs without sacrificing essential capabilities. Consequently, Moondream offers a practical pathway for deploying visual understanding capabilities in real-world scenarios where resource efficiency and accessibility are paramount.

To ensure reproducibility and streamline the deployment process, Docker is utilized to containerize the application. Docker provides a lightweight and portable environment that encapsulates the application and its dependencies, eliminating environment inconsistency issues. By defining the runtime environment in a Dockerfile, it is ensured that the model behaves consistently across different computing platforms. This approach simplifies the setup process for other researchers and developers, allowing them to easily deploy and experiment with software applications or models without worrying about conflicting libraries or system configurations.

In this work, the Moondream model is implemented for Visual Question Answering (VQA) tasks. The model processes natural language questions about input images and generates corresponding natural language answers. To ensure consistency and ease of deployment, the application is containerized using Docker. The implementation is validated using a set of test images and questions. The application is a web application that developed by using Streamlit.

This report is structured as follows; after the introduction, the methodology section describes the technical details of the implementation, an example usage of the application is given in the results section and the usage of the application is given in the usage section. Lastly the conclusion section provides a summary of the work done.

2 Methodology

The application is designed as a web-based Visual Question Answering (VQA) system using the Streamlit framework. This approach significantly enhances user interaction and accessibility. The core of the system remains the Moondream model, but the surrounding architecture is redesigned to support stateful user sessions, image management, and a responsive user interface.

2.1 Application Architecture

The application follows a modular structure, separating the user interface, business logic, and model interaction. The entry point is the `main.py` script, which serves as the main entry point for the Streamlit application. It initializes the session state, manages the navigation between different pages, and ensures the Moondream model is loaded. It directs the user to either the Main Page or the Chat Page based on the application state.

The User Interface, located in the `ui/` directory, is built using Streamlit's component-based design. It consists of the `MainPage` and `ChatPage` components. The `MainPage` handles the initial user landing, providing functionality for users to upload new images to start a chat session and displaying a history of recent chat sessions. The `ChatPage` manages the active chat interface, displaying the uploaded image, rendering the chat history (questions and answers), and providing an input field for new queries to interact with the model effectively.

To maintain clean code and separation of concerns, service modules are implemented in the `services/` directory. The `ChatService` manages the storage and retrieval of chat history, creating new sessions, and generating chat names. The `ImageService` handles image file operations, including saving uploaded images and loading them for display or processing.

2.2 Model Integration

The `Model` class in `model/model.py` encapsulates the interaction with the Moondream model. It is responsible for conditionally loading the model on GPU or CPU based on hardware availability, processing the input image into embeddings that the model can understand, and taking a natural language question and the encoded image to generate a text response during inference.

A dedicated script, `model_installer.py`, is used to download the specific revision of the Moondream model from the Hugging Face hub. This ensures that the correct model files are present before the application starts, which is particularly important for the Dockerized environment.

2.3 Data Management and Optimization

To ensure a seamless user experience and efficient resource utilization, the application implements specific strategies for data persistence and performance optimization. The application maintains a persistent record of user interactions, where chat sessions, including the image reference and the dialogue history, are serialized and stored in a JSON file. The `ChatService` module handles the reading and writing of this data, ensuring that users can access their previous conversations even after restarting the application. This feature allows for a continuity of work and reviewing past insights. Furthermore, to address the computational cost of the Visual Question Answering task, the `Model` class implements an in-memory caching mechanism. When an image is uploaded, it is encoded once, and the resulting embeddings are stored in the `self.enc_image` attribute. Subsequent questions about the same image utilize this cached representation, significantly reducing latency and improving the responsiveness of the chat interface.

2.4 Containerization

The application is containerized using Docker to ensure a consistent execution environment. The `Dockerfile` is updated to support the web application. It uses `python:3.10-slim` as a base image for a lightweight footprint and installs system dependencies and Python packages listed in `requirements.txt`, which now includes `streamlit`.

A crucial step in the methodology is the execution of `model_installer.py` during the Docker build phase. This bakes the large model weights into the Docker image, preventing the need to download them every time the container starts.

Finally, the container exposes port 8501, the default for Streamlit, and uses `streamlit run main.py` as the entry point command.

3 Usage

The resulting image is published in the docker hub. The application can be found in this link: https://hub.docker.com/r/okanberhoglu/chat_moon_vlm

After downloading the image, the application can be run using the following command:

```
docker run -p 8501:8501 okanberhoglu/chat_moon_vlm:latest
```

The given command runs the Docker image in a container and maps the host's port 8501 to the container's port 8501. This allows the user to access the web application by navigating to `http://localhost:8501` in their web browser. The Docker image "okanberhoglu/chat_moon_vlm:latest" encapsulates all required dependencies, the Streamlit server, and the inference environment. Once the container is running, the application interface will be available for uploading images and asking questions.

4 Results

The main page of the application is shown in the figure 1.

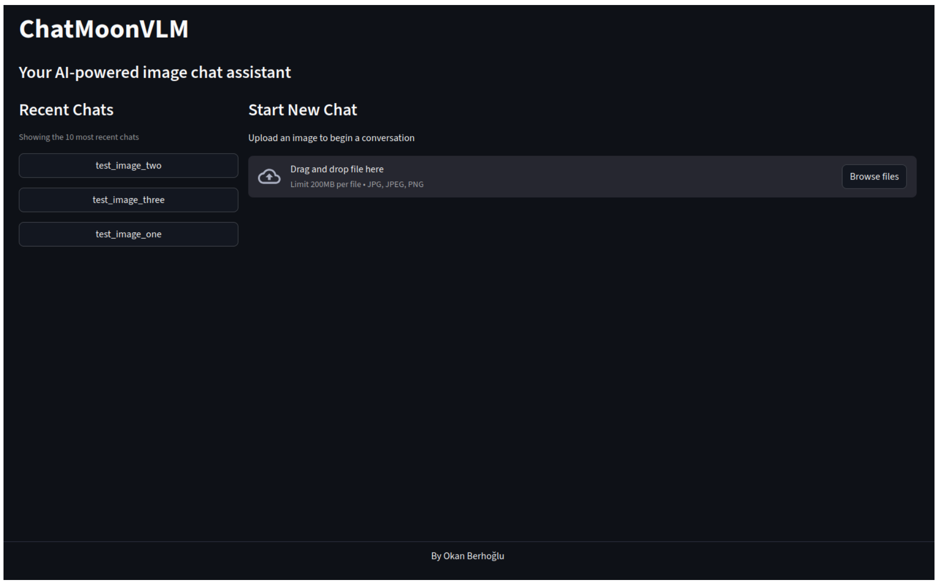

ChatMoonVLM
Your AI-powered image chat assistant
Recent Chats
Showing the 10 most recent chats
test_image_two
test_image_three
test_image_one
Start New Chat
Upload an image to begin a conversation
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
By Okan Berhoğlu

Figure 1: Main page of the application.

In the main page user can upload an image and start a new chat. Besides, user can go to the previous chats in this page. The figure 2 shows the main page with an uploaded image. By clicking the "Start New Chat" button, user can start a new chat. And the chat screen of for the image that is given in the figure 2 is shown in the figure 3.

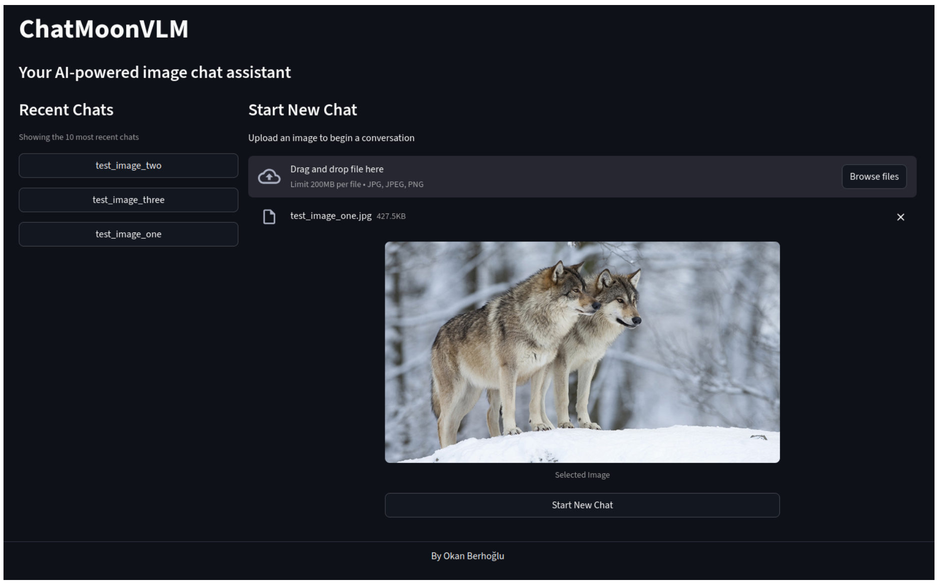
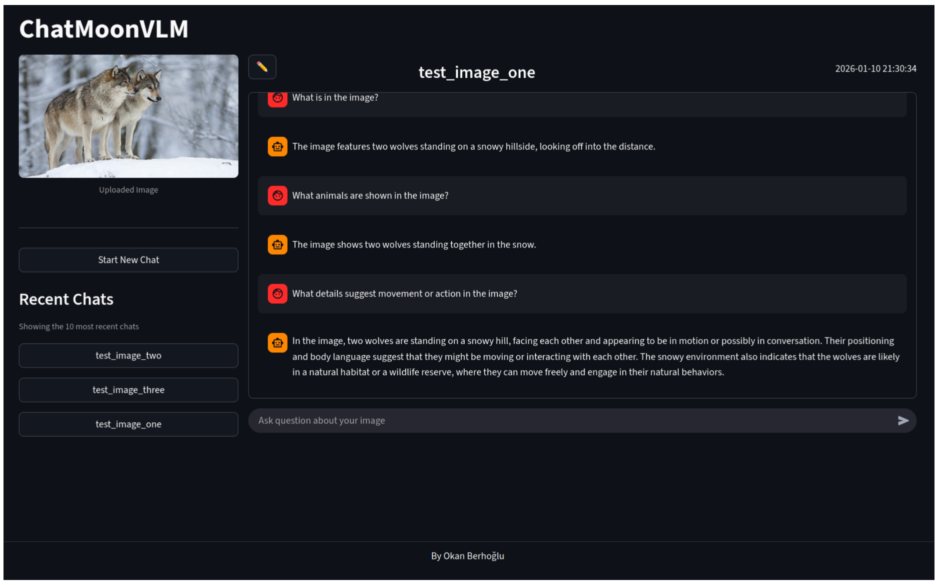

ChatMoonVLM
Your AI-powered image chat assistant
Recent Chats
Showing the 10 most recent chats
test_image_two
test_image_three
test_image_one
Start New Chat
Upload an image to begin a conversation
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
test_image_one.jpg 427.5KB
Selected Image
Start New Chat
By Okan Berhoğlu

Figure 2: Example input image.



ChatMoonVLM

Uploaded Image

Start New Chat

Recent Chats

Showing the 10 most recent chats

test_image_two

test_image_three

test_image_one

test_image_one

2026-01-10 21:30:34

What is in the image?

The image features two wolves standing on a snowy hillside, looking off into the distance.

What animals are shown in the image?

The image shows two wolves standing together in the snow.

What details suggest movement or action in the image?

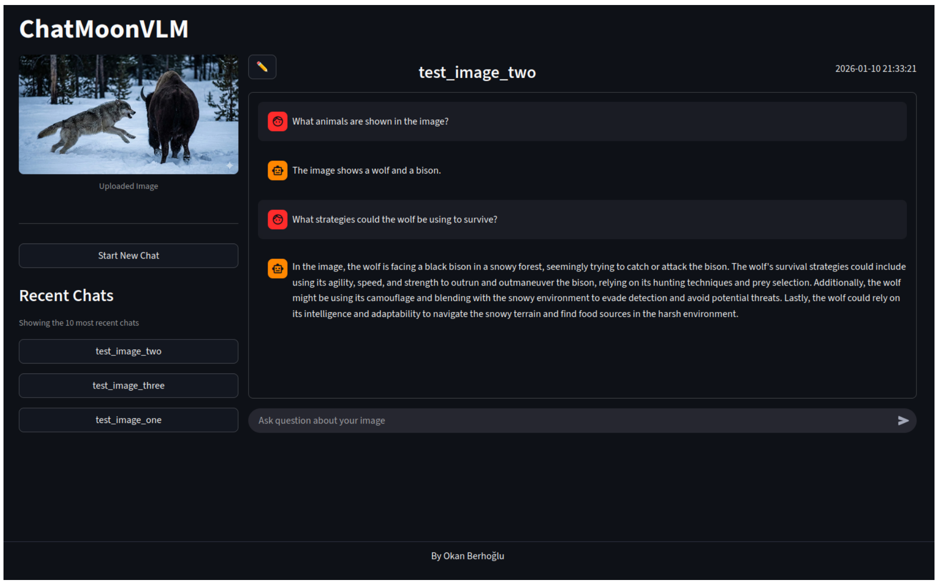
In the image, two wolves are standing on a snowy hill, facing each other and appearing to be in motion or possibly in conversation. Their positioning and body language suggest that they might be moving or interacting with each other. The snowy environment also indicates that the wolves are likely in a natural habitat or a wildlife reserve, where they can move freely and engage in their natural behaviors.

Ask question about your image

By Okan Berhoğlu

Figure 3: Example usage with a test image.

In the figures 3, 4 and 5 the chat screen with example questions are shown. The user can ask questions about the image and the model will answer them in the chat page.



ChatMoonVLM

Uploaded Image

Start New Chat

Recent Chats

Showing the 10 most recent chats

test_image_two

test_image_three

test_image_one

test_image_two

2026-01-10 21:33:21

What animals are shown in the image?

The image shows a wolf and a bison.

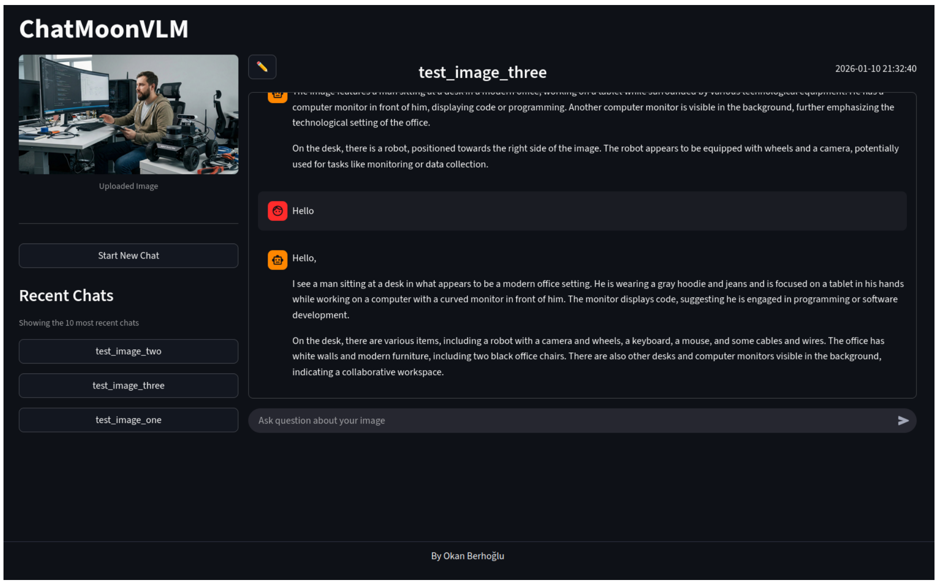
What strategies could the wolf be using to survive?

In the image, the wolf is facing a black bison in a snowy forest, seemingly trying to catch or attack the bison. The wolf's survival strategies could include using its agility, speed, and strength to outrun and outmaneuver the bison, relying on its hunting techniques and prey selection. Additionally, the wolf might be using its camouflage and blending with the snowy environment to evade detection and avoid potential threats. Lastly, the wolf could rely on its intelligence and adaptability to navigate the snowy terrain and find food sources in the harsh environment.

Ask question about your image

By Okan Berhoğlu

Figure 4: Example usage with a test image.



test_image_three

2026-01-10 21:32:40

The image features a man sitting at a desk in a modern office, working on a tablet while surrounded by various technological equipment. He has a computer monitor in front of him, displaying code or programming. Another computer monitor is visible in the background, further emphasizing the technological setting of the office.

On the desk, there is a robot, positioned towards the right side of the image. The robot appears to be equipped with wheels and a camera, potentially used for tasks like monitoring or data collection.

Hello

Hello,

I see a man sitting at a desk in what appears to be a modern office setting. He is wearing a gray hoodie and jeans and is focused on a tablet in his hands while working on a computer with a curved monitor in front of him. The monitor displays code, suggesting he is engaged in programming or software development.

On the desk, there are various items, including a robot with a camera and wheels, a keyboard, a mouse, and some cables and wires. The office has white walls and modern furniture, including two black office chairs. There are also other desks and computer monitors visible in the background, indicating a collaborative workspace.

Figure 5: Example usage with a test image.

5 Conclusion

In this project, a web-based Visual Question Answering (VQA) system was successfully developed and deployed using the Moondream model and the Streamlit framework. The application provides an intuitive interface for users to upload images, manage chat history, and interact with the vision-language model in real-time. By containerizing the entire system with Docker, a consistent and reproducible environment was established, simplifying deployment across different platforms. Furthermore, the implementation of chat history persistence and in-memory image embedding caching significantly enhances the user experience and system performance. This work demonstrates that efficient, open-source vision-language models can be effectively integrated into user-friendly applications and run on local hardware, offering a practical solution for privacy-conscious and resource-constrained scenarios.

References

[1] Moondream.ai. *Moondream: A tiny vision language model*. Available at: `https://moondream.ai/` (Accessed: January 11, 2026).

[2] Hugging Face. *Hugging Face: The AI community building the future*. Available at: `https://huggingface.co/` (Accessed: January 11, 2026).

[3] Docker. *Docker Documentation*. Available at: `https://docs.docker.com/` (Accessed: January 11, 2026).

[4] Streamlit. *Streamlit: A faster way to build and share data apps*. Available at: `https://streamlit.io/` (Accessed: January 11, 2026).