DeepSeek

DeepSeek-V3 is a Mixture-of-Experts (MoE) language model with 671 billion total parameters, of which only 37 billion are active per token. It improves training efficiency and inference speed while maintaining state-of-the-art performance in natural language understanding, coding, and mathematical reasoning, including:

DeepSeek LLM (General-purpose language model)
DeepSeek Coder (AI for code generation)
DeepSeek Vision (Upcoming multimodal model)

Key Innovations:

- Multi-Head Latent Attention (MLA) for memory-efficient attention.
- DeepSeekMoE with auxiliary-loss-free load balancing.
- Multi-Token Prediction (MTP) for faster inference and training.
- FP8 mixed-precision training for efficient computation.
- Optimized cross-node communication for large-scale distributed training.

1. Basic Architecture

Transformer-Based Model with Mixture-of-Experts

DeepSeek-V3 follows the Transformer architecture, but instead of a dense model, it uses a Mixture-of-Experts approach, where only a subset of parameters are active at any time. This enables training large-scale models with lower computational costs.

- Total Parameters: 671 billion
- Active Parameters per Token: 37 billion
- Mixture-of-Experts (MoE) layers distribute computations across multiple expert subnetworks.

Multi-Head Latent Attention

MLA is a memory-efficient attention mechanism that reduces the Key-Value cache size without losing accuracy.

- Compresses attention keys and values to save memory.
- Uses Rotary Positional Embedding (RoPE) for long-context learning.
- Maintains performance similar to Multi-Head Attention (MHA) while reducing computational cost.
- MLA significantly reduces memory usage during inference, making DeepSeek-V3 faster and more efficient.

DeepSeekMoE: Optimized Mixture-of-Experts Training

DeepSeek-V3 uses DeepSeekMoE, an advanced MoE technique that improves load balancing and reduces training inefficiencies.

- Shared & Routed Experts: Some layers are shared across all tokens, while others are routed based on token characteristics.
- Auxiliary-Loss-Free Strategy: Unlike previous MoE models that use additional loss functions for balancing experts, DeepSeek-V3 dynamically adjusts expert loads without introducing performance penalties.
- Node-Limited Routing: Limits communication costs by restricting expert allocation across compute nodes.
- MoE allows DeepSeek-V3 to scale up massively while keeping training and inference computationally feasible.

Multi-Token Prediction

DeepSeek-V3 introduces Multi-Token Prediction, a novel approach where the model predicts multiple future tokens in parallel, instead of just the next token.

- Speeds up training by generating multiple tokens at once.
- Improves inference by enabling speculative decoding.
- Maintains complete causal relationships across token predictions.
- MTP reduces training costs and allows DeepSeek-V3 to generate text more quickly than traditional autoregressive models.

2. Infrastructure

DeepSeek-V3 was trained on a highly optimized computing infrastructure with NVIDIA H800 GPUs.

Compute Cluster

- 2048 NVIDIA H800 GPUs used for training.
- 8 GPUs per node connected via NVLink (fast intra-node communication).
- InfiniBand interconnects for communication between nodes.
- This setup enables high-speed distributed training while minimizing communication overhead.

Training Framework

- DeepSeek-V3 was trained using **HAI-LLM**, a custom training framework optimized for large-scale AI models.
    - Key Optimization
        - DualPipe Parallelism – A novel pipeline parallelism method that overlaps computation and communication, improving efficiency.
        - Efficient Cross-Node Communication – Uses custom all-to-all communication kernels that maximize InfiniBand and NVLink bandwidth.
        - Memory Optimization – Recomputed normalization & low-precision storage reduce memory usage during training.
        - DeepSeek-V3 can train on extremely large datasets while keeping computational costs low.

3. Pre-Training

Dataset & Training Objective

- Trained on 14.8 trillion diverse high-quality tokens.
- Supports long-context training up to 128K tokens.
- Uses FP8 mixed precision for more efficient training.

FP8 Mixed Precision Training

DeepSeek-V3 is one of the first large-scale models to use FP8 training, which reduces memory usage and improves computational efficiency.

- Reduces memory consumption without loss in accuracy.
- Speeds up training by reducing precision bottlenecks.
- FP8 training allows DeepSeek-V3 to be trained using fewer resources while maintaining high accuracy.

4. Post-Training

Supervised Fine-Tuning (SFT)

- Trained on human-annotated datasets.
- Improves instruction-following ability for chatbot applications.

Reinforcement Learning with Human Feedback (RLHF-like)

- Uses a Reward Model and Group Relative Policy Optimization (GRPO).
- Fine-tunes the model to align with human preferences.

Feature	DeepSeek-V3	GPT-4	LLaMA 3	Mistral
Total Parameters	671B	~1T	405B	7B, 13B
Active Parameters per Token	37B	~200B	405B	7B, 13B
Open-Source	Yes	No	Yes	Yes
Efficient MoE	Yes	No	No	No
Pre-Training Tokens	14.8T	Unknown	Unknown	Unknown

DeepSeek is a Chinese AI company specializing in foundation models for natural language processing (NLP), code generation, and multimodal AI. Their main goal is to create highly capable AI models.Including

DeepSeek LLM

DeepSeek LLM is a transformer-based AI model designed for various NLP tasks, similar to GPT-4, LLaMA, and Mistral. It can be used for:

- **Text generation**
- **Summarization**
- **Translation**
- **Question answering**
- **Sentiment analysis**
- **Conversational AI**

Key Features of DeepSeek LLM

- Large-scale dataset: Trained on a massive dataset covering web data, books, research papers, and code repositories.
- Multilingual support: Works across multiple languages, making it useful for global applications.
- Open-source availability: Some versions are freely available for research and development.

- Improved factual accuracy: Designed to reduce hallucinations and improve reliability.

- ☐ Available on **Hugging Face** and **GitHub** for **free use and fine-tuning**.
- ☐ API access for **developers and businesses**.

DeepSeek Coder

DeepSeek Coder is an AI coding assistant similar to GitHub Copilot and Code Llama. It helps developers by suggesting, completing, and debugging code in real-time.

Key Features of DeepSeek Coder

- Supports multiple programming languages: Python, Java, JavaScript, C++, ABAP, and more.
- Code completion & generation: Suggests full functions, classes, and even entire scripts.
- Intelligent debugging: Detects and fixes code errors.
- Optimized for software engineering: Can handle complex programming patterns.

- ☐ Open-source models are available on **Hugging Face**.
- ☐ Can be **integrated into IDEs** like VS Code.

DeepSeek Vision (Upcoming Model)

DeepSeek Vision is expected to be a multimodal AI model capable of processing text + images. This would make it an alternative to GPT-4V and Gemini.

Expected Features of DeepSeek Vision

- Image recognition
- Visual question answering (VQA)
- Image captioning
- Text + image reasoning

Availability & Access

DeepSeek provides open-source models as well as API-based access for businesses.

MODEL	AVAILABILITY	USAGE
DeepSeek LLM	Open-Source	Text-based AI, chatbot development, NLP tasks
DeepSeek Coder	Open-Source	Code generation, debugging, automation
DeepSeek Vision	Not Released Yet	Multimodal AI (Text + Image)
API Access	Available	Developers & businesses

- Hugging Face: Free access to model weights.
- GitHub: Open-source versions available.

- API Access: For enterprises and large-scale applications.

DeepSeek vs. Other AI Models

DeepSeek LLM vs. GPT-4, LLaMA, Mistral

Feature	**DeepSeek LLM**	**GPT-4**	**LLaMA 2**	**Mistral**
Open Source	(Partially)	(Closed)	(Open)	(Open)
Multimodal(Text + Image)	(Not Yet)	Yes	No	No
Performance	Fast	High	Moderate	Moderate
Fine-Tuning	Yes	No	Yes	Yes

DeepSeek Coder vs. Code Llama, GitHub Copilot

Feature	DeepSeek Coder	Code Llama	GitHub Copilot
Open-source	Yes	Yes	No
IDE Integration	Yes	Yes	Yes
Free Usage	Yes	Yes	No(paid)

Key Differences

- DeepSeek is partially open-source, while OpenAI’s GPT-4 is closed.

- DeepSeek is optimized for coding, whereas GPT-4 is a general AI model.
- DeepSeek provides free access, while GitHub Copilot and OpenAI APIs are paid.

Usage Of DeepSeek

For NLP Applications

AI chatbots
summarization tools
Sentiment analysis

For Coding

AI-powered code assistant (like Copilot)
Debugging and error detection
Code completion for developers

For Research

Experimenting with LLM architectures
Fine-tuning AI models for specific tasks

Creating RAG Agent using DeepSeek -R1 and Ollama Document Assisstent

In this I create end to end RAG application with the help of DeepSeek. This entire DeepSeek will be specifically installed in our local with the help of Ollama. In this I use Ollama embedding and show that How the vectors are created and how can we store that vector even in our local system

Accuracy

It's quite good when I chat with the RAG application. Performance wise also it's working really good.

First I download Ollama distilled Model and installed in my PC in My Command Prompt
This is the model I installed In my PC

```
ollama run deepseek-r1:1.5b
```

Models

DeepSeek-R1

```
ollama run deepseek-r1:671b
```

Distilled models

DeepSeek team has demonstrated that the reasoning patterns of larger models can be distilled into smaller models, resulting in better performance compared to the reasoning patterns discovered through RL on small models.

Below are the models created via fine-tuning against several dense models widely used in the research community using reasoning data generated by DeepSeek-R1. The evaluation results demonstrate that the distilled smaller dense models perform exceptionally well on benchmarks.

DeepSeek-R1-Distill-Qwen-1.5B

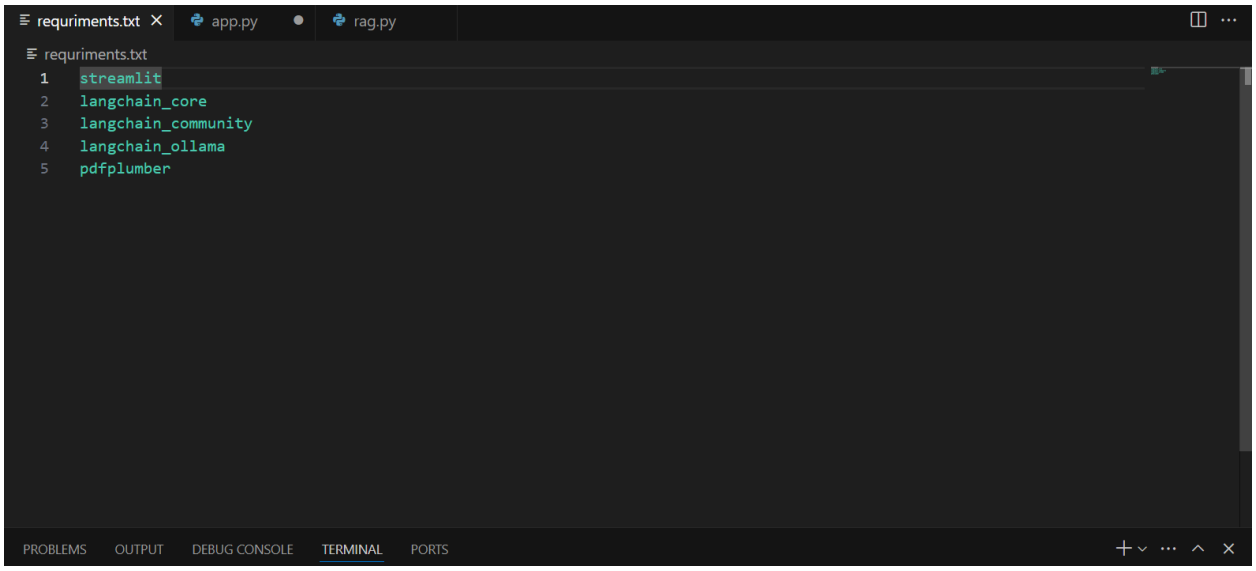
```
ollama run deepseek-r1:1.5b
```

DeepSeek-R1-Distill-Qwen-7B

```
ollama run deepseek-r1:7b
```

I created my RAG application in vscode and I push This in Github

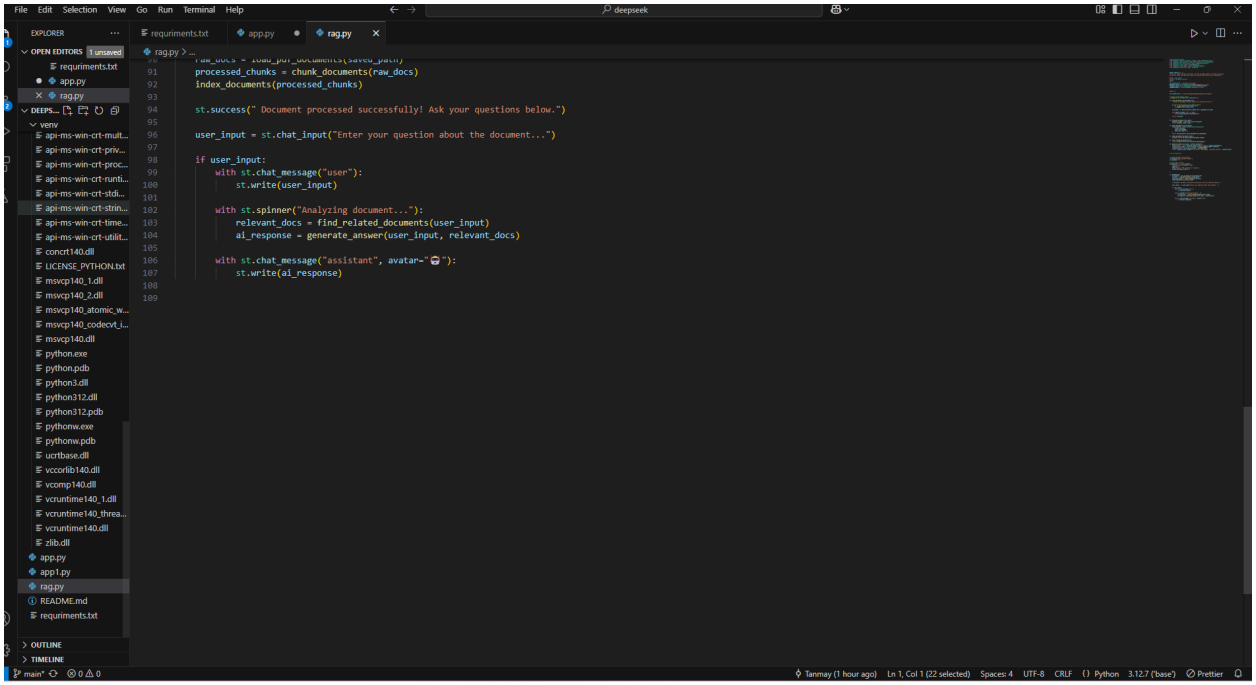
These are the requirements first we pip installed before working on code



Then I created a python file name rag.py where I start doing My code this is my code

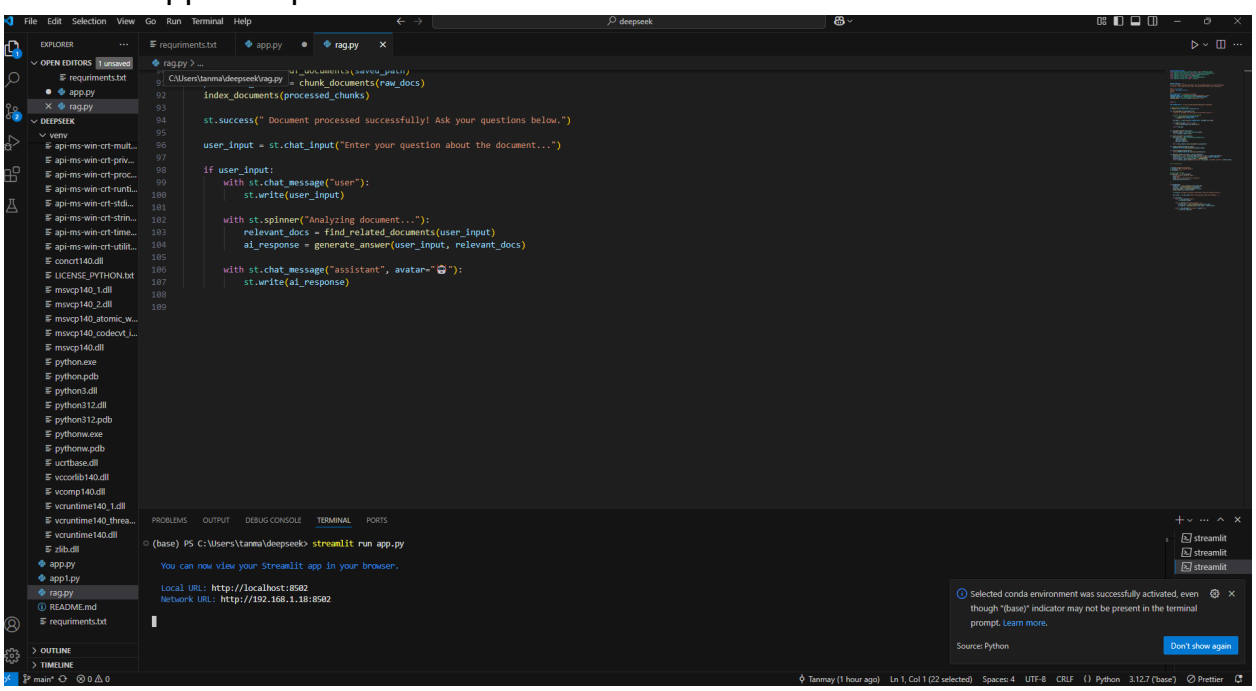
```python
import streamlit as st
from langchain_community.document_loaders import PDFPlumberLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain_core.vectorstores import InMemoryVectorStore
from langchain_ollama import OllamaEmbeddings
from langchain_core.prompts import ChatPromptTemplate
from langchain_ollama.llms import OllamaLLM



PROMPT_TEMPLATE = """
You are an expert research assistant. Use the provided context to answer the query.
If unsure, state that you don't know. Be concise and factual (max 3 sentences).

Query: {user_query}
Context: {document_context}
Answer:
"""
PDF_STORAGE_PATH = 'document_store/pdfs/'
EMBEDDING_MODEL = OllamaEmbeddings(model="deepseek-r1:1.5b")
DOCUMENT_VECTOR_DB = InMemoryVectorStore(EMBEDDING_MODEL)
LANGUAGE_MODEL = OllamaLLM(model="deepseek-r1:1.5b")


import os

PDF_STORAGE_PATH = "C:/Users/tanma/deepseek/document_store/pdfs/"

# Ensure the directory exists
os.makedirs(PDF_STORAGE_PATH, exist_ok=True)

def save_uploaded_file(uploaded_file):
    """Saves an uploaded file and ensures the directory exists."""

    # Ensure the directory exists before saving
    if not os.path.exists(PDF_STORAGE_PATH):
        os.makedirs(PDF_STORAGE_PATH)

    file_path = os.path.join(PDF_STORAGE_PATH, uploaded_file.name)

    with open(file_path, "wb") as file:
        file.write(uploaded_file.getbuffer())

    return file_path


def load_pdf_documents(file_path):
    document_loader = PDFPlumberLoader(file_path)
```

```python
def load_pdf_documents(file_path):
    document_loader = PDFPlumberLoader(file_path)
    return document_loader.load()

def chunk_documents(raw_documents):
    text_processor = RecursiveCharacterTextSplitter(
        chunk_size=1000,
        chunk_overlap=200,
        add_start_index=True
    )
    return text_processor.split_documents(raw_documents)

def index_documents(document_chunks):
    DOCUMENT_VECTOR_DB.add_documents(document_chunks)

def find_related_documents(query):
    return DOCUMENT_VECTOR_DB.similarity_search(query)

def generate_answer(user_query, context_documents):
    context_text = "\n\n".join([doc.page_content for doc in context_documents])
    conversation_prompt = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)
    response_chain = conversation_prompt | LANGUAGE_MODEL
    return response_chain.invoke({"user_query": user_query, "document_context": context_text})


# UI Configuration


st.title("Document Assisitent")
st.markdown("### Document Reader")
st.markdown("---")

# File Upload Section
uploaded_pdf = st.file_uploader(
    "Upload Research Document (PDF)",
    type="pdf",
    help="Select a PDF document for analysis",
    accept_multiple_files=False

)

if uploaded_pdf:
    saved_path = save_uploaded_file(uploaded_pdf)
    raw_docs = load_pdf_documents(saved_path)
    processed_chunks = chunk_documents(raw_docs)
    index_documents(processed_chunks)
```

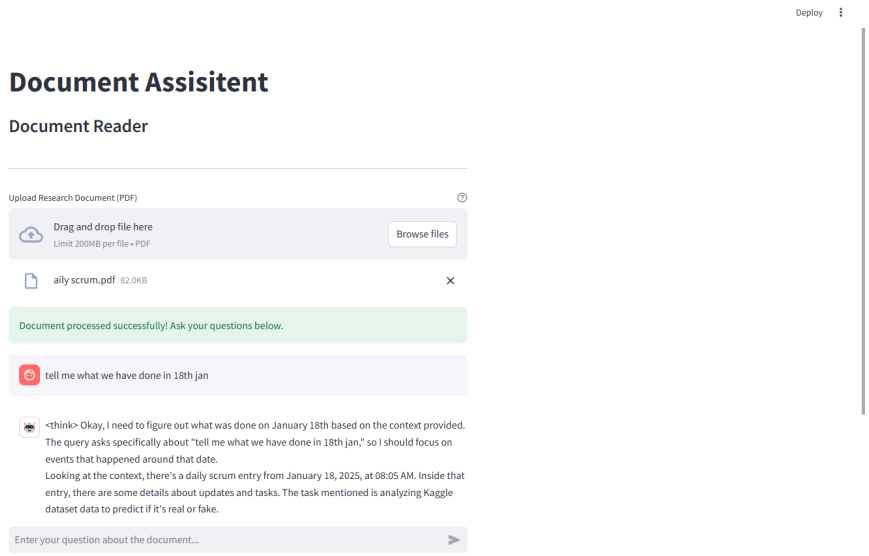


Then I run the code and in terminal I run streamlit run rag.py

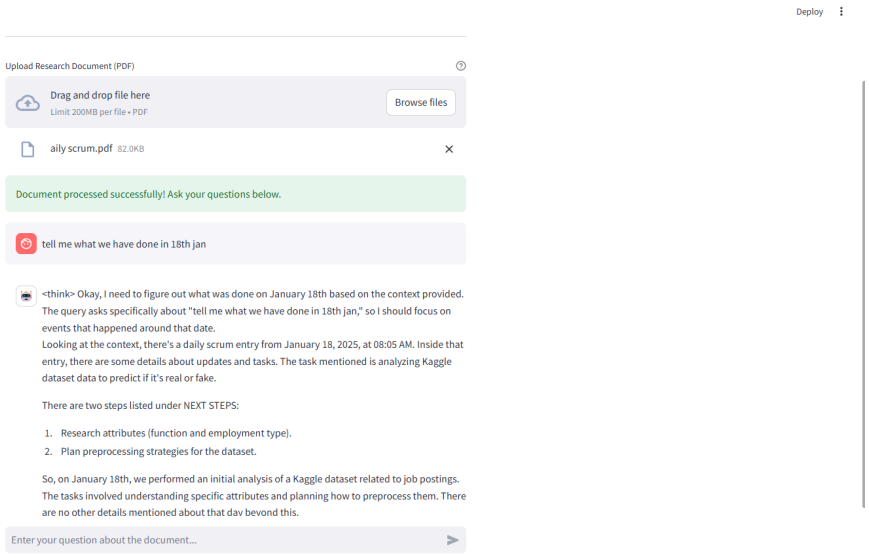
Streamlit app will open in browser



Streamlit app interface after uploading a file



After uploading a file the question bar appeared where you can ask anything about the PDF.It will exactly answer like the way DeepSeek answer.



The best part is that it will work locally, all information will be stored locally no one will access your file and information, not even China can access this only you can access this.