```
!pip install google-generativeai tqdm python-dotenv aiohttp requests beautifulsoup4 huggingface_hub
```

```
Requirement already satisfied: google-generativeai in c:\users\sumit\anaconda3\lib\site-packages (0.8.5)
Requirement already satisfied: tqdm in c:\users\sumit\anaconda3\lib\site-packages (4.67.1)
Requirement already satisfied: python-dotenv in c:\users\sumit\anaconda3\lib\site-packages (1.1.0)
Requirement already satisfied: aiohttp in c:\users\sumit\anaconda3\lib\site-packages (3.11.10)
Requirement already satisfied: requests in c:\users\sumit\anaconda3\lib\site-packages (2.32.5)
Requirement already satisfied: beautifulsoup4 in c:\users\sumit\anaconda3\lib\site-packages (4.12.3)
Requirement already satisfied: huggingface_hub in c:\users\sumit\anaconda3\lib\site-packages (0.36.0)
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in c:\users\sumit\anaconda3\lib\site-packages (from google-g
Requirement already satisfied: google-api-core in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai) (2.28.1)
Requirement already satisfied: google-api-python-client in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai)
Requirement already satisfied: google-auth>=2.15.0 in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai) (2.4
Requirement already satisfied: protobuf in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai) (5.29.3)
Requirement already satisfied: pydantic in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai) (2.12.4)
Requirement already satisfied: typing-extensions in c:\users\sumit\anaconda3\lib\site-packages (from google-generativeai) (4.15.
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in c:\users\sumit\anaconda3\lib\site-packages (from google-ai-genera
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in c:\users\sumit\anaconda3\lib\site-packages (from googl
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\sumit\anaconda3\lib\site-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sumit\anaconda3\lib\site-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sumit\anaconda3\lib\site-packages (from requests) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sumit\anaconda3\lib\site-packages (from requests) (2025.11.12)
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in c:\users\sumit\anaconda3\lib\site-packages (from google-api-core[grpc]!=
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in c:\users\sumit\anaconda3\lib\site-packages (from google-api-core[
Requirement already satisfied: cachetools<7.0,>=2.0.0 in c:\users\sumit\anaconda3\lib\site-packages (from google-auth>=2.15.0->g
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\sumit\anaconda3\lib\site-packages (from google-auth>=2.15.0->go
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\sumit\anaconda3\lib\site-packages (from google-auth>=2.15.0->google-gen
Requirement already satisfied: pyasn1>=0.1.3 in c:\users\sumit\anaconda3\lib\site-packages (from rsa<5,>=3.1.4->google-auth>=2.1
Requirement already satisfied: colorama in c:\users\sumit\anaconda3\lib\site-packages (from tqdm) (0.4.6)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (2.4.4)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (1.2.0)
Requirement already satisfied: attrs>=17.3.0 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (24.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in c:\users\sumit\anaconda3\lib\site-packages (from aiohttp) (1.18.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\sumit\anaconda3\lib\site-packages (from beautifulsoup4) (2.5)
Requirement already satisfied: filelock in c:\users\sumit\anaconda3\lib\site-packages (from huggingface_hub) (3.17.0)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\sumit\anaconda3\lib\site-packages (from huggingface_hub) (2025.3.2)
Requirement already satisfied: packaging>=20.9 in c:\users\sumit\anaconda3\lib\site-packages (from huggingface_hub) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\users\sumit\anaconda3\lib\site-packages (from huggingface_hub) (6.0.2)
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in c:\users\sumit\anaconda3\lib\site-packages (from google-api-python-cli
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in c:\users\sumit\anaconda3\lib\site-packages (from google-api
Requirement already satisfied: uritemplate<5,>=3.0.1 in c:\users\sumit\anaconda3\lib\site-packages (from google-api-python-clien
Requirement already satisfied: pyparsing<4,>=3.0.4 in c:\users\sumit\anaconda3\lib\site-packages (from httplib2<1.0.0,>=0.19.0->
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\sumit\anaconda3\lib\site-packages (from pydantic->google-gener
Requirement already satisfied: pydantic-core==2.41.5 in c:\users\sumit\anaconda3\lib\site-packages (from pydantic->google-genera
Requirement already satisfied: typing-inspection>=0.4.2 in c:\users\sumit\anaconda3\lib\site-packages (from pydantic->google-gen
```

```python
import os, json, time, asyncio
from tqdm import tqdm
import google.generativeai as genai
from huggingface_hub import hf_hub_download
import requests
from dotenv import load_dotenv
load_dotenv()

api_key = os.getenv("GEMINI_API_KEY")
genai.configure(api_key=api_key)

print("API Loaded Successfully!")
```

```
API Loaded Successfully!
```

```python
def safe_parse_json(text: str):
    import re
    text = text.strip()
    try: return json.loads(text)
    except: pass
    m = re.search(r'(\{(?:.|\n)*?\})', text)
    if m:
        try: return json.loads(m.group(1))
        except: pass
    m = re.search(r'(\[(?:.|\n)*?\])', text)
    if m:
        return json.loads(m.group(1))
    return text
```

```
class GeminiLLM:
    def __init__(self, model='models/gemini-2.5-pro'):
        self.model = genai.GenerativeModel(model)

    def generate(self, prompt):
        try:
            resp = self.model.generate_content(prompt)
            return resp.text
        except Exception as e:
            return f'[ERROR: {e}]'
```

```
class BaseAgent:
    def __init__(self, name, llm):
        self.name = name
        self.llm = llm
    async def run_task(self, task): raise NotImplementedError
```

```
class WebResearcherAgent(BaseAgent):
    async def run_task(self, task):
        topic = task["instruction"]
        prompt = f"""
Create a well-formatted Markdown article explaining:

    {topic}

Formatting rules:
- Use ### for main headings
- Use #### for subheadings
- Use bullet points
- Use spacing
- No JSON, no code blocks
- Make it look like a nicely formatted blog post
- Include links to good sources

Example format:

### **Topic Title**

#### **Section 1**
* point 1
* point 2

#### **Section 2**
* explanation
* link: https://example.com

Now write the formatted output:
"""
        response = self.llm.generate(prompt)
        return {
            "id": task["id"],
            "status": "done",
            "result": response
        }
```

```
class Controller:
    def __init__(self):
        self.llm = GeminiLLM()
        self.workers = {
            'web_researcher': WebResearcherAgent('web_researcher', self.llm)
        }

    def plan(self, goal):
        return [{"id": "t1", "role": "web_researcher", "instruction": goal, "context": {}}]

    async def execute(self, tasks):
        results = []
        for t in tasks:
            agent = self.workers[t['role']]
            results.append(await agent.run_task(t))
        return results
```

```
from IPython.display import Markdown, display

controller = Controller()
goal = input("Enter your goal: ")

tasks = controller.plan(goal)
print("TASKS:\n", tasks)

results = await controller.execute(tasks)

print("\nRESULTS:\n")
for r in results:
    display(Markdown(r["result"]))
```

```
Enter your goal:  image compression using deep learning and their models and datasets
TASKS:
 [{'id': 't1', 'role': 'web_researcher', 'instruction': 'image compression using deep learning and their models and datasets',

RESULTS:
```

Revolutionizing Pixels: An Introduction to Deep Learning for Image Compression

For decades, algorithms like JPEG, PNG, and WebP have been the unsung heroes of the internet, shrinking image files to make websites load faster and photo sharing possible. These traditional methods rely on handcrafted mathematical transforms, like the Discrete Cosine Transform (DCT), to remove redundant information. While effective, they often introduce noticeable artifacts like blocking and blurring at high compression rates.

Today, a new revolution is underway, powered by deep learning. Instead of using fixed mathematical formulas, neural networks learn to compress and decompress images in a way that is optimized for human perception, often achieving superior quality at the same or even smaller file sizes.

The Core Idea: Learning to Compress with Autoencoders

At the heart of most learned image compression models is a structure called an **autoencoder**. It's an artificial neural network trained to reconstruct its own input. For compression, this process is broken down into three key stages.

The Encoder

- This is a neural network (typically a Convolutional Neural Network or CNN) that takes a full-resolution image as input.
- Its job is to analyze the image and transform it into a compact, dense set of numbers called a **latent representation**. This representation is the compressed version of the image.

The Quantizer

- The latent representation produced by the encoder consists of floating-point numbers, which are difficult to store efficiently.
- The quantizer's role is to round these continuous values to the nearest integers. This is a crucial step that makes the data compressible but also introduces a loss of information.
- This quantized representation is then losslessly encoded (e.g., using arithmetic coding) to create the final bitstream, or compressed file.

The Decoder

Start coding or generate with AI.

- It takes the quantized latent representation as input and attempts to reconstruct the original image with the highest possible fidelity.
- The encoder and decoder are trained together, end-to-end, to minimize the difference between the original and the reconstructed image while keeping the size of the latent representation as small as possible.

Key Models and Architectures

The field has evolved rapidly, with several key architectural breakthroughs leading to state-of-the-art performance.

Variational Autoencoders (VAEs)

- Early pioneering work used VAEs to create a probabilistic model of the latent space.
- These models, notably from Ballé et al., introduced specialized layers like Generalized Divisive Normalization (GDN) that mimic processing in the human visual system, helping the network learn more efficient representations.
- Link: [End-to-end Optimized Image Compression]

Hyperprior Models

- This was a major leap forward. A hyperprior model uses a second, smaller autoencoder to compress the latent representation itself.
- This secondary network learns the spatial dependencies and statistical distribution within the latent representation, allowing for much more efficient entropy coding. This means a smaller file size for the same quality.
- This architecture is the foundation of many modern, high-performance learned compression models.
- Link: [Variational Image Compression with a Scale Hyperprior]

Autoregressive and Context-Adaptive Models

- To further improve compression, these models add a context-aware component.