Introduction to gpt-4o

Juston Forte
May 13, 2024

Open in Github

GPT-4o ("o" for "omni") is designed to handle a combination of text, audio, and video inputs, and can generate outputs in text, audio, and image formats.

Background

Before GPT-4o, users could interact with ChatGPT using Voice Mode, which operated with three separate models. GPT-4o will integrate these capabilities into a single model that's trained across text, vision, and audio. This unified approach ensures that all inputs—whether text, visual, or auditory—are processed cohesively by the same neural network.

Current API Capabilities

Currently, the API supports `{text, image}` inputs only, with `{text}` outputs, the same modalities as `gpt-4-turbo`. Additional modalities, including audio, will be introduced soon. This guide will help you get started with using GPT-4o for text, image, and video understanding.

Getting Started

Install OpenAI SDK for Python

```
%pip install --upgrade openai --quiet
```

Configure the OpenAI client and submit a test request

To setup the client for our use, we need to create an API key to use with our request. Skip these steps if you already have an API key for usage.

You can get an API key by following these steps:

1. **Create a new project**
2. **Generate an API key in your project**
3. (RECOMMENDED, BUT NOT REQUIRED) **Setup your API key for all projects as an env var**

Once we have this setup, let's start with a simple {text} input to the model for our first request. We'll use both `system` and `user` messages for our first request, and we'll receive a response from the `assistant` role.

```
from openai import OpenAI
import os

## Set the API key and model name
MODEL="gpt-4o"
client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY", "<your OpenAI API key if not set as an env
```

```
completion = client.chat.completions.create(
  model=MODEL,
  messages=[
    {"role": "system", "content": "You are a helpful assistant. Help me with my math homework!"}, #
    {"role": "user", "content": "Hello! Could you solve 2+2?"}  # <-- This is the user message for wl
  ]
)

print("Assistant: " + completion.choices[0].message.content)
```

```
Assistant: Of course!

\[ 2 + 2 = 4 \]

If you have any other questions, feel free to ask!
```

Image Processing

GPT-4o can directly process images and take intelligent actions based on the image. We can provide images in two formats:

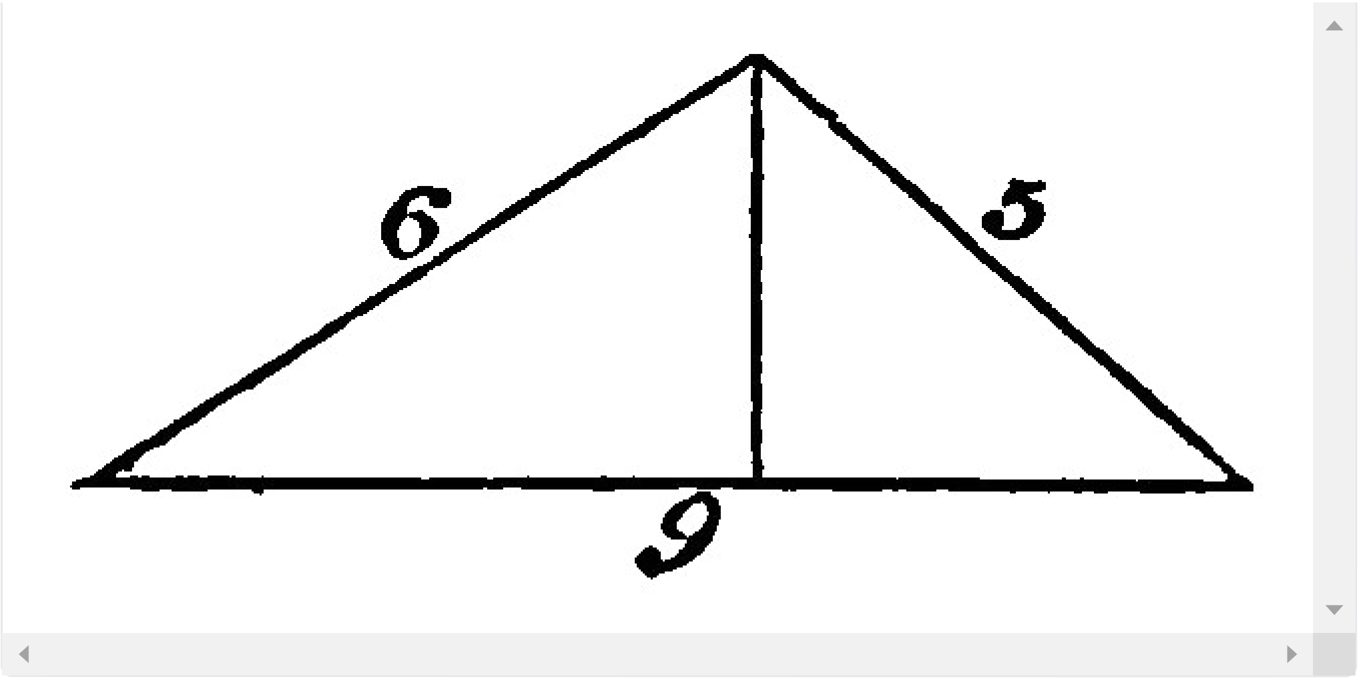
1. Base64 Encoded
2. URL

Let's first view the image we'll use, then try sending this image as both Base64 and as a URL link to the API

```
from IPython.display import Image, display, Audio, Markdown
import base64

IMAGE_PATH = "data/triangle.png"

# Preview image for context
display(Image(IMAGE_PATH))
```



Base64 Image Processing

```
# Open the image file and encode it as a base64 string
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode("utf-8")

base64_image = encode_image(IMAGE_PATH)

response = client.chat.completions.create(
    model=MODEL,
    messages=[
        {"role": "system", "content": "You are a helpful assistant that responds in Markdown. Help m
        {"role": "user", "content": [
            {"type": "text", "text": "What's the area of the triangle?"},
            {"type": "image_url", "image_url": {
                "url": f"data:image/png;base64,{base64_image}"}
            }
        ]}
    ],
    temperature=0.0,
)

print(response.choices[0].message.content)
```

```
To find the area of the triangle, we can use Heron's formula. First, we need to find the semi-

The sides of the triangle are 6, 5, and 9.

1. Calculate the semi-perimeter \( s \):
\[ s = \frac{a + b + c}{2} = \frac{6 + 5 + 9}{2} = 10 \]

2. Use Heron's formula to find the area \( A \):
\[ A = \sqrt{s(s-a)(s-b)(s-c)} \]

Substitute the values:
\[ A = \sqrt{10(10-6)(10-5)(10-9)} \]
\[ A = \sqrt{10 \cdot 4 \cdot 5 \cdot 1} \]
\[ A = \sqrt{200} \]
```

```
So, the area of the triangle is \( 10\sqrt{2} \) square units.
```

URL Image Processing

```
response = client.chat.completions.create(
    model=MODEL,
    messages=[
        {"role": "system", "content": "You are a helpful assistant that responds in Markdown. Help m
        {"role": "user", "content": [
            {"type": "text", "text": "What's the area of the triangle?"},
            {"type": "image_url", "image_url": {
                "url": "https://upload.wikimedia.org/wikipedia/commons/e/e2/The_Algebra_of_Mohammed_I
            }
        ]}
    ],
    temperature=0.0,
)

print(response.choices[0].message.content)
```

```
To find the area of the triangle, we can use Heron's formula. Heron's formula states that the

\[ \text{Area} = \sqrt{s(s-a)(s-b)(s-c)} \]

where \(s\) is the semi-perimeter of the triangle:
```

```
\[ s = \frac{a + b + c}{2} \]

For the given triangle, the side lengths are \(a = 5\), \(b = 6\), and \(c = 9\).

First, calculate the semi-perimeter \(s\):

\[ s = \frac{5 + 6 + 9}{2} = \frac{20}{2} = 10 \]

Now, apply Heron's formula:

\[ \text{Area} = \sqrt{10(10-5)(10-6)(10-9)} \]
\[ \text{Area} = \sqrt{10 \cdot 5 \cdot 4 \cdot 1} \]
\[ \text{Area} = \sqrt{200} \]
\[ \text{Area} = 10\sqrt{2} \]

So, the area of the triangle is \(10\sqrt{2}\) square units.
```

Video Processing

While it's not possible to directly send a video to the API, GPT-4o can understand videos if you sample frames and then provide them as images. It performs better at this task than GPT-4 Turbo.

Since GPT-4o in the API does not yet support audio-in (as of May 2024), we'll use a combination of GPT-4o and Whisper to process both the audio and visual for a provided video, and showcase two usecases:

1. Summarization
2. Question and Answering

Setup for Video Processing

We'll use two python packages for video processing - opencv-python and moviepy.

These require **ffmpeg**, so make sure to install this beforehand. Depending on your OS, you may need to run `brew install ffmpeg` or `sudo apt install ffmpeg`

```
%pip install opencv-python --quiet
%pip install moviepy --quiet
```

Process the video into two components: frames and audio

```python
import cv2
from moviepy.editor import VideoFileClip
import time
import base64

# We'll be using the OpenAI DevDay Keynote Recap video. You can review the video here: https://www.yo
VIDEO_PATH = "data/keynote_recap.mp4"
```

```python
def process_video(video_path, seconds_per_frame=2):
    base64Frames = []
    base_video_path, _ = os.path.splitext(video_path)

    video = cv2.VideoCapture(video_path)
    total_frames = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = video.get(cv2.CAP_PROP_FPS)
    frames_to_skip = int(fps * seconds_per_frame)
    curr_frame=0

    # Loop through the video and extract frames at specified sampling rate
    while curr_frame < total_frames - 1:
        video.set(cv2.CAP_PROP_POS_FRAMES, curr_frame)
        success, frame = video.read()
        if not success:
            break
        _, buffer = cv2.imencode(".jpg", frame)
        base64Frames.append(base64.b64encode(buffer).decode("utf-8"))
        curr_frame += frames_to_skip
    video.release()

    # Extract audio from video
    audio_path = f"{base_video_path}.mp3"
    clip = VideoFileClip(video_path)
    clip.audio.write_audiofile(audio_path, bitrate="32k")
    clip.audio.close()
    clip.close()

    print(f"Extracted {len(base64Frames)} frames")
    print(f"Extracted audio to {audio_path}")
    return base64Frames, audio_path

# Extract 1 frame per second. You can adjust the `seconds_per_frame` parameter to change the sampling
base64Frames, audio_path = process_video(VIDEO_PATH, seconds_per_frame=1)
```

```
MoviePy - Writing audio in data/keynote_recap.mp3
```

```
## Display the frames and audio for context
display_handle = display(None, display_id=True)
for img in base64Frames:
    display_handle.update(Image(data=base64.b64decode(img.encode("utf-8")), width=600))
    time.sleep(0.025)

Audio(audio_path)
```

0:00 / 3:31

Example 1: Summarization

Now that we have both the video frames and the audio, let's run a few different tests to generate a video summary to compare the results of using the models with different modalities.

We should expect to see that the summary generated with context from both visual and audio inputs will be the most accurate, as the model is able to use the entire context from the video.

1. Visual Summary
2. Audio Summary
3. Visual + Audio Summary

Visual Summary

The visual summary is generated by sending the model only the frames from the video. With just the frames, the model is likely to capture the visual aspects, but will miss any details discussed by the speaker.

```
response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content": "You are generating a video summary. Please provide a summary of tl
    {"role": "user", "content": [
        "These are the frames from the video.",
        *map(lambda x: {"type": "image_url",
                        "image_url": {"url": f'data:image/jpg;base64,{x}', "detail": "low"}}, base64
        ],
    }
    ],
    temperature=0,
)
print(response.choices[0].message.content)
```

```
## Video Summary: OpenAI DevDay Keynote Recap

The video appears to be a keynote recap from OpenAI's DevDay event. Here are the key points cc

1. **Introduction and Event Overview**:
   - The video starts with the title "OpenAI DevDay" and transitions to "Keynote Recap."
   - The event venue is shown, with attendees gathering and the stage set up.

2. **Keynote Presentation**:
   - A speaker, presumably from OpenAI, takes the stage to present.
   - The presentation covers various topics related to OpenAI's latest developments and annour

3. **Announcements**:
   - **GPT-4 Turbo**: Introduction of GPT-4 Turbo, highlighting its enhanced capabilities and
   - **JSON Mode**: A new feature that allows for structured data output in JSON format.
   - **Function Calling**: Demonstration of improved function calling capabilities, making int
   - **Context Length and Control**: Enhancements in context length and user control over the
   - **Better Knowledge Integration**: Improvements in the model's knowledge base and retrieva

4. **Product Demonstrations**:
   - **DALL-E 3**: Introduction of DALL-E 3 for advanced image generation.
   - **Custom Models**: Announcement of custom models, allowing users to tailor models to spec
   - **API Enhancements**: Updates to the API, including threading, retrieval, and code interp

5. **Pricing and Token Efficiency**:
   - Discussion on GPT-4 Turbo pricing, emphasizing cost efficiency with reduced input and out

6. **New Features and Tools**:
```

The results are as expected - the model is able to capture the high level aspects of the video visuals, but misses the details provided in the speech.

Audio Summary

The audio summary is generated by sending the model the audio transcript. With just the audio, the model is likely to bias towards the audio content, and will miss the context provided by the presentations and visuals.

`{audio}` input for GPT-4o isn't currently available but will be coming soon! For now, we use our existing `whisper-1` model to process the audio

```python
# Transcribe the audio
transcription = client.audio.transcriptions.create(
    model="whisper-1",
    file=open(audio_path, "rb"),
)
## OPTIONAL: Uncomment the line below to print the transcription
#print("Transcript: ", transcription.text + "\n\n")

response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content":"""You are generating a transcript summary. Create a summary of the
    {"role": "user", "content": [
        {"type": "text", "text": f"The audio transcription is: {transcription.text}"}
        ],
    }
    ],
    temperature=0,
)
print(response.choices[0].message.content)
```

```
### Summary

Welcome to OpenAI's first-ever Dev Day. Key announcements include:

- **GPT-4 Turbo**: A new model supporting up to 128,000 tokens of context, featuring JSON mode
- **New Features**:
  - **Dolly 3**, **GPT-4 Turbo with Vision**, and a new **Text-to-Speech model** are now avail
  - **Custom Models**: A program where OpenAI researchers help companies create custom models
  - **Increased Rate Limits**: Doubling tokens per minute for established GPT-4 customers and
- **GPTs**: Tailored versions of ChatGPT for specific purposes, programmable through conversat
- **Assistance API**: Includes persistent threads, built-in retrieval, a code interpreter, anc

OpenAI is excited about the future of AI integration and looks forward to seeing what users wi
```

The audio summary is biased towards the content discussed during the speech, but comes out with much less structure than the video summary.

Audio + Visual Summary

The Audio + Visual summary is generated by sending the model both the visual and the audio from the video at once. When sending both of these, the model is expected to better summarize since it can perceive the entire video at once.

```
## Generate a summary with visual and audio
response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content":"""You are generating a video summary. Create a summary of the prov
    {"role": "user", "content": [
        "These are the frames from the video.",
        *map(lambda x: {"type": "image_url",
                        "image_url": {"url": f'data:image/jpg;base64,{x}', "detail": "low"}}, base64
        {"type": "text", "text": f"The audio transcription is: {transcription.text}"}
        ],
    }
],
    temperature=0,
)
print(response.choices[0].message.content)
```

```
## Video Summary: OpenAI Dev Day

### Introduction
- The video begins with the title "OpenAI Dev Day" and transitions to a keynote recap.

### Event Overview
- The event is held at a venue with a sign reading "OpenAI Dev Day."
- Attendees are seen entering and gathering in a large hall.

### Keynote Presentation
- The keynote speaker introduces the event and announces the launch of GPT-4 Turbo.
- **GPT-4 Turbo**:
  - Supports up to 128,000 tokens of context.
  - Introduces a new feature called JSON mode for valid JSON responses.
  - Improved function calling capabilities.
  - Enhanced instruction-following and knowledge retrieval from external documents or database
  - Knowledge updated up to April 2023.
  - Available in the API along with DALL-E 3, GPT-4 Turbo with Vision, and a new Text-to-Speec
```

```
### Custom Models
- Launch of a new program called Custom Models.
  - Researchers will collaborate with companies to create custom models tailored to specific u
  - Higher rate limits and the ability to request changes to rate limits and quotas directly i

### Pricing and Performance
- **GPT-4 Turbo**:
  - 3x cheaper for prompt tokens and 2x cheaper for completion tokens compared to GPT-4.
  - Doubling the tokens per minute for established GPT-4 customers.
```

After combining both the video and audio, we're able to get a much more detailed and comprehensive summary for the event which uses information from both the visual and audio elements from the video.

Example 2: Question and Answering

For the Q&A, we'll use the same concept as before to ask questions of our processed video while running the same 3 tests to demonstrate the benefit of combining input modalities:

1. Visual Q&A
2. Audio Q&A
3. Visual + Audio Q&A

```
QUESTION = "Question: Why did Sam Altman have an example about raising windows and turning the radio
```

```
qa_visual_response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content": "Use the video to answer the provided question. Respond in Markdowr
    {"role": "user", "content": [
        "These are the frames from the video.",
        *map(lambda x: {"type": "image_url", "image_url": {"url": f'data:image/jpg;base64,{x}', "det
        QUESTION
        ],
    }
    ],
    temperature=0,
)
print("Visual QA:\n" + qa_visual_response.choices[0].message.content)
```

```
Visual QA:
Sam Altman used the example about raising windows and turning the radio on to demonstrate the
```

```
qa_audio_response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content":"""Use the transcription to answer the provided question. Respond ii
    {"role": "user", "content": f"The audio transcription is: {transcription.text}. \n\n {QUESTION}"
    ],
    temperature=0,
)
print("Audio QA:\n" + qa_audio_response.choices[0].message.content)
```

```
Audio QA:
The provided transcription does not include any mention of Sam Altman or an example about rais
```

```
qa_both_response = client.chat.completions.create(
    model=MODEL,
    messages=[
    {"role": "system", "content":"""Use the video and transcription to answer the provided question.
    {"role": "user", "content": [
        "These are the frames from the video.",
        *map(lambda x: {"type": "image_url",
                        "image_url": {"url": f'data:image/jpg;base64,{x}', "detail": "low"}}, base64
                        {"type": "text", "text": f"The audio transcription is: {transcription.text}"
        QUESTION
        ],
    }
    ],
    temperature=0,
)
print("Both QA:\n" + qa_both_response.choices[0].message.content)
```

```
Both QA:
Sam Altman used the example of raising windows and turning the radio on to demonstrate the imp
```

Comparing the three answers, the most accurate answer is generated by using both the audio and visual from the video. Sam Altman did not discuss the raising windows or radio on during the Keynote, but referenced an improved capability for the model to execute multiple functions in a single request while the examples were shown behind him.

Conclusion

Integrating many input modalities such as audio, visual, and textual, significantly enhances the performance of the model on a diverse range of tasks. This multimodal approach allows for more comprehensive understanding and interaction, mirroring more closely how humans perceive and process information.

Currently, GPT-4o in the API supports text and image inputs, with audio capabilities coming soon.