**[Vision](https://platform.openai.com/docs/guides/vision/vision)**

Learn how to use vision capabilities to understand images.

**[Introduction](https://platform.openai.com/docs/guides/vision/introduction)**

GPT-4o, GPT-4o mini, and GPT-4 Turbo have vision capabilities, meaning the models can take in images and answer questions about them. Historically, language model systems have been limited by taking in a single input modality, text.

**[Quickstart](https://platform.openai.com/docs/guides/vision/quickstart)**

Images are made available to the model in two main ways: by passing a link to the image or by passing the base64 encoded image directly in the request. Images can be passed in the user messages.

What's in this image?

python

Select librarypythoncurlnode

from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(

model="gpt-4o-mini",

messages=[

{

"role": "user",

"content": [

{"type": "text", "text": "What’s in this image?"},

{

"type": "image\_url",

"image\_url": {

"url": "https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gfp-wisconsin-madison-the-nature-boardwalk.jpg/2560px-Gfp-wisconsin-madison-the-nature-boardwalk.jpg",

},

},

],

}

],

max\_tokens=300,

)

print(response.choices[0])

The model is best at answering general questions about what is present in the images. While it does understand the relationship between objects in images, it is not yet optimized to answer detailed questions about the location of certain objects in an image. For example, you can ask it what color a car is or what some ideas for dinner might be based on what is in you fridge, but if you show it an image of a room and ask it where the chair is, it may not answer the question correctly.

It is important to keep in mind the [limitations of the model](https://platform.openai.com/docs/guides/vision/limitations) as you explore what use-cases visual understanding can be applied to.

[Video understanding with vision](https://cookbook.openai.com/examples/gpt_with_vision_for_video_understanding" \t "_blank)

[Learn how to use use GPT-4 with Vision to understand videos in the OpenAI Cookbook](https://cookbook.openai.com/examples/gpt_with_vision_for_video_understanding" \t "_blank)

**[Uploading base 64 encoded images](https://platform.openai.com/docs/guides/vision/uploading-base-64-encoded-images)**

If you have an image or set of images locally, you can pass those to the model in base 64 encoded format, here is an example of this in action:

import base64

import requests

# OpenAI API Key

api\_key = "YOUR\_OPENAI\_API\_KEY"

# Function to encode the image

def encode\_image(image\_path):

with open(image\_path, "rb") as image\_file:

return base64.b64encode(image\_file.read()).decode('utf-8')

# Path to your image

image\_path = "path\_to\_your\_image.jpg"

# Getting the base64 string

base64\_image = encode\_image(image\_path)

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {api\_key}"

}

payload = {

"model": "gpt-4o-mini",

"messages": [

{

"role": "user",

"content": [

{

"type": "text",

"text": "What’s in this image?"

},

{

"type": "image\_url",

"image\_url": {

"url": f"data:image/jpeg;base64,{base64\_image}"

}

}

]

}

],

"max\_tokens": 300

}

response = requests.post("https://api.openai.com/v1/chat/completions", headers=headers, json=payload)

print(response.json())

**[Multiple image inputs](https://platform.openai.com/docs/guides/vision/multiple-image-inputs)**

The Chat Completions API is capable of taking in and processing multiple image inputs in both base64 encoded format or as an image URL. The model will process each image and use the information from all of them to answer the question.

Multiple image inputs

python

Select librarypythoncurlnode

from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(

model="gpt-4o-mini",

messages=[

{

"role": "user",

"content": [

{

"type": "text",

"text": "What are in these images? Is there any difference between them?",

},

{

"type": "image\_url",

"image\_url": {

"url": "https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gfp-wisconsin-madison-the-nature-boardwalk.jpg/2560px-Gfp-wisconsin-madison-the-nature-boardwalk.jpg",

},

},

{

"type": "image\_url",

"image\_url": {

"url": "https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gfp-wisconsin-madison-the-nature-boardwalk.jpg/2560px-Gfp-wisconsin-madison-the-nature-boardwalk.jpg",

},

},

],

}

],

max\_tokens=300,

)

print(response.choices[0])

Here the model is shown two copies of the same image and can answer questions about both or each of the images independently.

**[Low or high fidelity image understanding](https://platform.openai.com/docs/guides/vision/low-or-high-fidelity-image-understanding)**

By controlling the detail parameter, which has three options, low, high, or auto, you have control over how the model processes the image and generates its textual understanding. By default, the model will use the auto setting which will look at the image input size and decide if it should use the low or high setting.

* low will enable the "low res" mode. The model will receive a low-res 512px x 512px version of the image, and represent the image with a budget of 85 tokens. This allows the API to return faster responses and consume fewer input tokens for use cases that do not require high detail.
* high will enable "high res" mode, which first allows the model to first see the low res image (using 85 tokens) and then creates detailed crops using 170 tokens for each 512px x 512px tile.

Choosing the detail level

python

Select librarypythoncurlnode

from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(

model="gpt-4o-mini",

messages=[

{

"role": "user",

"content": [

{"type": "text", "text": "What’s in this image?"},

{

"type": "image\_url",

"image\_url": {

"url": "https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gfp-wisconsin-madison-the-nature-boardwalk.jpg/2560px-Gfp-wisconsin-madison-the-nature-boardwalk.jpg",

"detail": "high"

},

},

],

}

],

max\_tokens=300,

)

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

**[Managing images](https://platform.openai.com/docs/guides/vision/managing-images)**

The Chat Completions API, unlike the Assistants API, is not stateful. That means you have to manage the messages (including images) you pass to the model yourself. If you want to pass the same image to the model multiple times, you will have to pass the image each time you make a request to the API.

For long running conversations, we suggest passing images via URL's instead of base64. The latency of the model can also be improved by downsizing your images ahead of time to be less than the maximum size they are expected them to be. For low res mode, we expect a 512px x 512px image. For high res mode, the short side of the image should be less than 768px and the long side should be less than 2,000px.

After an image has been processed by the model, it is deleted from OpenAI servers and not retained. [We do not use data uploaded via the OpenAI API to train our models](https://openai.com/enterprise-privacy).

**[Limitations](https://platform.openai.com/docs/guides/vision/limitations)**

While GPT-4 with vision is powerful and can be used in many situations, it is important to understand the limitations of the model. Here are some of the limitations we are aware of:

* Medical images: The model is not suitable for interpreting specialized medical images like CT scans and shouldn't be used for medical advice.
* Non-English: The model may not perform optimally when handling images with text of non-Latin alphabets, such as Japanese or Korean.
* Small text: Enlarge text within the image to improve readability, but avoid cropping important details.
* Rotation: The model may misinterpret rotated / upside-down text or images.
* Visual elements: The model may struggle to understand graphs or text where colors or styles like solid, dashed, or dotted lines vary.
* Spatial reasoning: The model struggles with tasks requiring precise spatial localization, such as identifying chess positions.
* Accuracy: The model may generate incorrect descriptions or captions in certain scenarios.
* Image shape: The model struggles with panoramic and fisheye images.
* Metadata and resizing: The model doesn't process original file names or metadata, and images are resized before analysis, affecting their original dimensions.
* Counting: May give approximate counts for objects in images.
* CAPTCHAS: For safety reasons, we have implemented a system to block the submission of CAPTCHAs.