# Vision

Copy page

Learn how to use vision capabilities to understand images.

Several OpenAI models have vision capabilities, meaning the models can take images as input and answer questions about them. Historically, language models were limited to a single input modality: text.

Currently, models that can take images as input include `o1`, `gpt-4o`, `gpt-4o-mini`, and `gpt-4-turbo`.

## 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.

Analyze the content of an image

python

```python
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/Gf
                    },
                },
            ],
        }
    ],
    max_tokens=300,
)

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

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

Keep model limitations in mind as you explore use cases for visual understanding.

Video understanding with vision
Learn how to use use GPT-4 with Vision to understand videos in the OpenAI Cookbook

## Uploading Base64 encoded images

If you have an image or set of images locally, pass them to the model in Base64 encoded format:

```
import base64
from openai import OpenAI

client = OpenAI()

# 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)

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "What is in this image?",
                },
                {
                    "type": "image_url",
                    "image_url": {"url": f"data:image/jpeg;base64,{base64_image}"},
                },
            ],
        }
    ],
)
```

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

## Multiple image inputs

The Chat Completions API is capable of taking in and processing multiple image inputs, in Base64 encoded format or as an image URL. The model processes each image and uses information from all images to answer the question.

Multiple image inputs python

```python
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/Gf
                    },
                },
                {
                    "type": "image_url",
                    "image_url": {
                        "url": "https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gf
                    },
                },
            ],
        }
    ],
    max_tokens=300,
)
print(response.choices[0])
```

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

## Low or high fidelity image understanding

The `detail` parameter—which has three options, `low`, `high`, and `auto`—gives you control over how the model processes the image and generates its textual understanding. By default, the model will use the `auto` setting, which looks at the image input size and decides if it should use the `low` or `high` setting.

> `low` enables the "low res" mode. The model receives a low-resolution 512px x 512px version of the image. It represents 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` enables "high res" mode, which first lets the model see the low-resolution image (using 85 tokens) and then creates detailed crops using 170 tokens for each 512px x 512px tile.

Choosing the detail level

python

```python
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/Gf
                        "detail": "high",
                    },
                },
            ],
        }
    ],
    max_tokens=300,
)

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

## Managing images

Unlike the Assistants API, the Chat Completions API isn't stateful. That means you have to manage messages (including images) you pass to the model. To pass the same image to the model multiple times, you have to pass the image each time you make a request to the API.

For long-running conversations, we suggest passing images via URLs instead of Base64. The latency of the model can also be improved by downsizing your images ahead of time to less than the maximum size.

### Image size guidelines

We restrict image uploads to 20MB per image. Here are our image size expectations.

| MODE | EXPECTED IMAGE SIZE |
|---|---|
| Low-res | 512px x 512px |
| High res | Short side: less than 768px<br><br>Long side: less than 2,000px |

After an image has been processed by the model, it's deleted from OpenAI servers and not retained. We do not use data uploaded via the OpenAI API to train our models.

## Limitations

While models with vision capabilities are powerful and can be used in many situations, it's important to understand the limitations of these models. Here are some known limitations:

**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 or upside-down text and 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**: The model may give approximate counts for objects in images.

**CAPTCHAS**: For safety reasons, our system blocks the submission of CAPTCHAs.

## Calculating costs

Image inputs are metered and charged in tokens, just as text inputs are. The token cost of an image is determined by two factors: size and detail.

### Low res cost

Any image with `detail: low` costs 85 tokens.

### High res cost

To calculate the cost of an image with `detail: high`, we do the following:

- Scale to fit within a 2048px x 2048px square, maintaining original aspect ratio
- Scale so that the image's shortest side is 768px long
- Count the number of 512px squares in the image—each square costs **170 tokens**
- Add **85 tokens** to the total

### Cost calculation examples

- A 1024 x 1024 square image in `detail: high` mode costs 765 tokens
  - 1024 is less than 2048, so there is no initial resize.
  - The shortest side is 1024, so we scale the image down to 768 x 768.
  - 4 512px square tiles are needed to represent the image, so the final token cost is `170 * 4 + 85 = 765`.
- A 2048 x 4096 image in `detail: high` mode costs 1105 tokens
  - We scale down the image to 1024 x 2048 to fit within the 2048 square.
  - The shortest side is 1024, so we further scale down to 768 x 1536.
  - 6 512px tiles are needed, so the final token cost is `170 * 6 + 85 = 1105`.
- A 4096 x 8192 image in `detail: low` most costs 85 tokens
  - Regardless of input size, low detail images are a fixed cost.

## FAQ

### Can I fine-tune the image understanding capabilities of a model?

Vision fine-tuning is available for some models. Learn more.

### Can I use a model with vision capabilities to generate images?

No, you can use `dall-e-3` to generate images and `o1`, `gpt-4o`, `gpt-4o-mini`, or `gpt-4-turbo` to understand images.

### What type of files can I upload?

We support PNG (.png), JPEG (.jpeg and .jpg), WEBP (.webp), and non-animated GIF (.gif).

### Is there a limit to the size of the image I can upload?

Yes, we restrict image uploads to 20MB per image.

### Can I delete an image I uploaded?

No, we will delete the image for you automatically after it has been processed by the model.

### Where can I learn more about the considerations of vision capabilities?

You can find details about our evaluations, preparation, and mitigation work in the GPT-4 with Vision system card.

We have further implemented a system to block the submission of CAPTCHAs.

### How do rate limits for models with vision capabilities work?

We process images at the token level, so each image we process counts towards your tokens per minute (TPM) limit. See the calculating costs section for details on the formula used to determine token count per image.

### Can a model with vision capabilities understand image metadata?

No, the model does not receive image metadata.

### What happens if my image is unclear?

If an image is ambiguous or unclear, the model will do its best to interpret it. However, the results may be less accurate. A good rule of thumb is that if an average human cannot see the info in an image at the resolutions used in low/high res mode, the model cannot either.