

# How to use the DALL-E API



Ted Sanders  
Nov 3, 2022

Open in Github

This notebook shows how to use OpenAI's DALL-E image API endpoints.

There are three API endpoints:

- **Generations:** generates an image or images based on an input caption
- **Edits:** edits or extends an existing image
- **Variations:** generates variations of an input image

## Setup

- Import the packages you'll need
- Import your OpenAI API key: You can do this by running `export OPENAI\_API\_KEY="your API key"` in your terminal.
- Set a directory to save images to

```
# imports
from openai import OpenAI # OpenAI Python library to make API calls
import requests # used to download images
import os # used to access filepaths
from PIL import Image # used to print and edit images

# initialize OpenAI client
client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY", "<your OpenAI API key if not set as env var>"))
```

```
# set a directory to save DALL-E images to
image_dir_name = "images"
image_dir = os.path.join(os.curdir, image_dir_name)

# create the directory if it doesn't yet exist
if not os.path.isdir(image_dir):
    os.mkdir(image_dir)

# print the directory to save to
print(f"image_dir={image_dir}")

image_dir='./images'
```

## Generations

The generation API endpoint creates an image based on a text prompt. [API Reference](#)

**Required inputs:**

- **prompt** (str): A text description of the desired image(s). The maximum length is 1000 characters for dall-e-2 and 4000 characters for dall-e-3.

**Optional inputs:**

- **model** (str): The model to use for image generation. Defaults to dall-e-2
- **n** (int): The number of images to generate. Must be between 1 and 10. Defaults to 1.
- **quality** (str): The quality of the image that will be generated. hd creates images with finer details and greater consistency across the image. This param is only supported for dall-e-3.
- **response\_format** (str): The format in which the generated images are returned. Must be one of "url" or "b64\_json". Defaults to "url".
- **size** (str): The size of the generated images. Must be one of 256x256, 512x512, or 1024x1024 for dall-e-2. Must be one of 1024x1024, 1792x1024, or 1024x1792 for dall-e-3 models. Defaults to "1024x1024".
- **style** (str | null): The style of the generated images. Must be one of vivid or natural. Vivid causes the model to lean towards generating hyper-real and dramatic images. Natural causes the model to produce more natural, less hyper-real looking images. This param is only supported for dall-e-3.
- **user** (str): A unique identifier representing your end-user, which will help OpenAI to monitor and detect abuse. [Learn more](#).

```
# create an image

# set the prompt
prompt = "A cyberpunk monkey hacker dreaming of a beautiful bunch of bananas, digital art"

# call the OpenAI API
generation_response = client.images.generate(
    model="dall-e-3",
    prompt=prompt,
    n=1,
    size="1024x1024",
    response_format="url",
)

# print response
print(generation_response)
```

```
ImagesResponse(created=1701994117, data=[Image(b64_json=None, revised_prompt=None, url='https://oai...
```

```
# save the image
generated_image_name = "generated_image.png" # any name you like; the filetype should be .png
generated_image_filepath = os.path.join(image_dir, generated_image_name)
generated_image_url = generation_response.data[0].url # extract image URL from response
generated_image = requests.get(generated_image_url).content # download the image

with open(generated_image_filepath, "wb") as image_file:
    image_file.write(generated_image) # write the image to the file
```

```
# print the image
print(generated_image_filepath)
display(Image.open(generated_image_filepath))
```

## Variations

The variations endpoint generates new images (variations) similar to an input image. [API Reference](#)

Here we'll generate variations of the image generated above.

### Required inputs:

- `image` (str): The image to use as the basis for the variation(s). Must be a valid PNG file, less than 4MB, and square.

### Optional inputs:

- `model` (str): The model to use for image variations. Only dall-e-2 is supported at this time.
- `n` (int): The number of images to generate. Must be between 1 and 10. Defaults to 1.
- `size` (str): The size of the generated images. Must be one of "256x256", "512x512", or "1024x1024". Smaller images are faster. Defaults to "1024x1024".
- `response_format` (str): The format in which the generated images are returned. Must be one of "url" or "b64\_json". Defaults to "url".
- `user` (str): A unique identifier representing your end-user, which will help OpenAI to monitor and detect abuse. [Learn more](#).

```
# create variations

# call the OpenAI API, using `create_variation` rather than `create`
variation_response = client.images.create_variation(
    image=generated_image, # generated_image is the image generated above
    n=2,
    size="1024x1024",
    response_format="url",
)

# print response
print(variation_response)
```

```
ImagesResponse(created=1701994139, data=[Image(b64_json=None, revised_prompt=None, url='https://oai...
```

```
# save the images
variation_urls = [datum.url for datum in variation_response.data] # extract URLs
variation_images = [requests.get(url).content for url in variation_urls] # download images
variation_image_names = [f"variation_image_{i}.png" for i in range(len(variation_images))] # create names
variation_image_filepaths = [os.path.join(image_dir, name) for name in variation_image_names] # create filepaths
for image, filepath in zip(variation_images, variation_image_filepaths): # loop through the variations
    with open(filepath, "wb") as image_file: # open the file
        image_file.write(image) # write the image to the file
```

```
# print the original image
```

```

print(generated_image_filepath)
display(Image.open(generated_image_filepath))

# print the new variations
for variation_image_filepaths in variation_image_filepaths:
    print(variation_image_filepaths)
    display(Image.open(variation_image_filepaths))

```

## Edits

The edit endpoint uses DALL-E to generate a specified portion of an existing image. Three inputs are needed: the image to edit, a mask specifying the portion to be regenerated, and a prompt describing the desired image. [API Reference](#)

### Required inputs:

- `[image]` (str): The image to edit. Must be a valid PNG file, less than 4MB, and square. If mask is not provided, image must have transparency, which will be used as the mask.
- `[prompt]` (str): A text description of the desired image(s). The maximum length is 1000 characters.

### Optional inputs:

- `[mask]` (file): An additional image whose fully transparent areas (e.g. where alpha is zero) indicate where image should be edited. Must be a valid PNG file, less than 4MB, and have the same dimensions as image.
- `[model]` (str): The model to use for edit image. Only dall-e-2 is supported at this time.
- `[n]` (int): The number of images to generate. Must be between 1 and 10. Defaults to 1.
- `[size]` (str): The size of the generated images. Must be one of "256x256", "512x512", or "1024x1024". Smaller images are faster. Defaults to "1024x1024".
- `[response_format]` (str): The format in which the generated images are returned. Must be one of "url" or "b64\_json". Defaults to "url".
- `[user]` (str): A unique identifier representing your end-user, which will help OpenAI to monitor and detect abuse. [Learn more](#).

## Set Edit Area

An edit requires a "mask" to specify which portion of the image to regenerate. Any pixel with an alpha of 0 (transparent) will be regenerated. The code below creates a 1024x1024 mask where the bottom half is transparent.

```

# create a mask
width = 1024
height = 1024
mask = Image.new("RGBA", (width, height), (0, 0, 0, 1)) # create an opaque image mask

# set the bottom half to be transparent
for x in range(width):
    for y in range(height // 2, height): # only loop over the bottom half of the mask
        # set alpha (A) to zero to turn pixel transparent
        alpha = 0
        mask.putpixel((x, y), (0, 0, 0, alpha))

# save the mask
mask_name = "bottom_half_mask.png"
mask_filepath = os.path.join(image_dir, mask_name)
mask.save(mask_filepath)

```

## Perform Edit

Now we supply our image, caption and mask to the API to get 5 examples of edits to our image

```

# edit an image

# call the OpenAI API
edit_response = client.images.edit(
    image=open(generated_image_filepath, "rb"), # from the generation section
    mask=open(mask_filepath, "rb"), # from right above
    prompt=prompt, # from the generation section
    n=1,
    size="1024x1024",
    response_format="url",
)

# print response
print(edit_response)

```

```
ImagesResponse(created=1701994167, data=[Image(b64_json=None, revised_prompt=None, url='https://oai...
```

```

# save the image
edited_image_name = "edited_image.png" # any name you like; the filetype should be .png
edited_image_filepath = os.path.join(image_dir, edited_image_name)
edited_image_url = edit_response.data[0].url # extract image URL from response

```

```
edited_image_url = edit_response.json()['url'] # extract image URL from response
```

```
edited_image = requests.get(edited_image_url).content # download the image
```

```
# print the original image
```

```
print(generated_image_filepath)
```

```
display(Image.open(generated_image_filepath))
```

```
# print edited image
```

```
print(edited_image_filepath)
```

```
display(Image.open(edited_image_filepath))
```