Vision

Our latest Mistral Small and our Pixtral models possess vision capabilities, enabling them to analyze images and provide insights based on visual content in addition to text. This multimodal approach opens up new possibilities for applications that require both textual and visual understanding.

Models with Vision Capacilities:

- Pixtral 12B (`pixtral-12b-latest`)
- Pixtral Large 2411 (`pixtral-large-latest`)
- Mistral Small 2503 (`mistral-small-latest`)

Passing an Image URL

If the image is hosted online, you can simply provide the URL of the image in the request. This method is straightforward and does not require any encoding.

python typescript curl

```
import { Mistral } from "@mistralai/mistralai";

const apiKey = process.env["MISTRAL_API_KEY"];

const client = new Mistral({ apiKey: apiKey });

const chatResponse = await client.chat.complete({
  model: "pixtral-12b",
  messages: [
    {
      role: "user",
      content: [
        { type: "text", text: "What's in this image?" },
        {
          type: "image_url",
          imageUrl: "https://tripfixers.com/wp-content/uploads/2019/11/eiffel-tower-with-snow.jpeg",
        },
      ],
    },
  ],
});
```

```
console.log("JSON:", chatResponse.choices[0].message.content);
```

Passing a Base64 Encoded Image

If you have an image or a set of images stored locally, you can pass them to the model in base64 encoded format. Base64 encoding is a common method for converting binary data into a text format that can be easily transmitted over the internet. This is particularly useful when you need to include images in API requests.

```
import base64
import requests
import os
from mistralai import Mistral

def encode_image(image_path):
    """Encode the image to base64."""
    try:
        with open(image_path, "rb") as image_file:
            return base64.b64encode(image_file.read()).decode('utf-8')
    except FileNotFoundError:
        print(f"Error: The file {image_path} was not found.")
        return None
    except Exception as e:  # Added general exception handling
        print(f"Error: {e}")
        return None

# Path to your image
image_path = "path_to_your_image.jpg"

# Getting the base64 string
base64_image = encode_image(image_path)

# Retrieve the API key from environment variables
api_key = os.environ["MISTRAL_API_KEY"]

# Specify model
model = "pixtral-12b-2409"

# Initialize the Mistral client
client = Mistral(api_key=api_key)

# Define the messages for the chat
messages = [
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "What's in this image?"
            },
```

```
            {
                "type": "image_url",
                "image_url": f"{base64_image}"
            }
        ]
    }
]

# Get the chat response
chat_response = client.chat.complete(
    model=model,
    messages=messages
)

# Print the content of the response
print(chat_response.choices[0].message.content)
```

Use cases

- Understand charts
- Compare images
- Transcribe receipts
- Transcribe old documents
- OCR with structured output

FAQ

- **What is the price per image?**

  The price is calculated using the same pricing as input tokens.

  Pixtral:

  For both Pixtral models, each image will be divided into batches of 16x16 pixels, with each batch converted to a token. As a rule of thumb, an image with a resolution of "ResolutionX"x"ResolutionY" will consume approximately `(ResolutionX/16) * (ResolutionY/16)` tokens. For example, a 720x512 image will consume approximately

`(720/16) * (512/16)` ≈ 1440 tokens. Note that all images with a resolution higher than 1024x1024 will be downscaled while maintaining the same aspect ratio. For instance, a 1436x962 image will be downscaled to approximately 1024x686, consuming around `(1024/16) * (686/16)` ≈ 2600 tokens.

Final Formula: `N of tokens ≈ (ResolutionX * ResolutionY) / 256`

Small 2503:

Small is similar; however, instead of batches of 16, it will be batched in 14 pixels. Instead of a maximum resolution of 1024x1024, it has a maximum resolution of 1540x1540. Due to its slightly different architecture, it also only uses 1/4 of that number of tokens as input to the text decoder. This means that in total, you can summarize the consumption approximately as `(ResolutionX/14) * (ResolutionY/14) * 1/4`, which is approximately 3x less than Pixtral models, making it use fewer tokens and be more efficient.

Final Formula: `N of tokens ≈ (ResolutionX * ResolutionY) / 784`

- **Can I fine-tune the image capabilities?**

  No, we do not currently support fine-tuning the image capabilities.

- **Can I use them to generate images?**

  No, they are designed to understand and analyze images, not to generate them.

- **What types of image files are supported?**

  We currently support the following image formats:

  - PNG (.png)
  - JPEG (.jpeg and .jpg)
  - WEBP (.webp)
  - Non-animated GIF with only one frame (.gif)

- **Is there a limit to the size of the image?**

  The current file size limit is 10Mb.

- **What's the maximum number images per request?**

  The maximum number images per request via API is 8.

- **What is the rate limit?**

  For information on rate limits, please visit https://console.mistral.ai/limits/.