Vision

Groq API offers fast inference for multimodal models with vision capabilities for understanding and interpreting visual data from images. By analyzing the content of an image, multimodal models can generate human-readable text for providing insights about given visual data.

Supported Model(s)

Groq API supports powerful multimodal model(s) that can be easily integrated into your applications to provide fast and accurate image processing for tasks such as visual question answering, caption generation, and Optical Character Recognition (OCR):

Llama 3.2 11B Vision (Preview)

- **Model ID:** llama-3.2-11b-vision-preview
- **Description:** Llama 3.2 11B Vision is a powerful multimodal model capable of processing both text and image inputs. It supports **multilingual, multi-turn conversations, tool use,** and **JSON mode**.
- **Context Window:** 8,192 tokens
- **Limitations:**
    - **Preview Model:** Currently in preview and should be used for experimentation.
    - **Image Size Limit:** The maximum allowed size for a request containing an image URL as input is 20MB. Requests larger than this limit will return a 400 error.
    - **Single Image per Request:** Only one image can be processed per request in the preview release. Requests with multiple images will return a 400 error.
    - **System Prompt:** The model does not support system prompts and images in the same request.

How to Use Vision

Use Groq API vision features via:

- GroqCloud Console Playground: Select llama-3.2-11b-vision-preview or llava-v1.5-7b-4096-preview as the model and upload your image.
- **Groq API Request:** Call the chat.completions API endpoint (i.e. https://api.groq.com/openai/v1/chat/completions) and set model_id to llama-3.2-11b-vision-preview or llava-v1.5-7b-4096-preview. See code examples below.

How to Pass Images from URLs as Input

The following are code examples for passing your image to the model via a URL:

```
const Groq = require('groq-sdk');


const groq = new Groq();
async function main() {
  const chatCompletion = await groq.chat.completions.create({
    "messages": [
     {
       "role": "user",
       "content": [
        {
          "type": "text",
          "text": "What's in this image?"
        },
        {
          "type": "image_url",
          "image_url": {
            "url": "${IMAGE_DATA_URL}"
          }
        }
      ]
     },
     {
       "role": "assistant",
       "content": ""
     }
    ],
    "model": "llama-3.2-11b-vision-preview",
    "temperature": 1,
```

```
    "max_tokens": 1024,
    "top_p": 1,
    "stream": false,
    "stop": null
  });

  console.log(chatCompletion.choices[0].message.content);
}

main();
```

How to Pass Locally Saved Images as Input

To pass locally saved images, we'll need to first encode our image to a base64 format string before passing it as the image_url in our API request as follows

```
from groq import Groq
import base64


# 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)
```

```
client = Groq()


chat_completion = client.chat.completions.create(
  messages=[
    {
      "role": "user",
      "content": [
        {"type": "text", "text": "What's in this image?"},
        {
          "type": "image_url",
          "image_url": {
            "url": f"{base64_image}",
          },
        },
      ],
    }
  ],
  model="llava-v1.5-7b-4096-preview",
)


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

Tool Use with Images

The llama-3.2-11b-vision-preview model supports tool use! The following cURL example defines a get_current_weather tool that the model can leverage to answer a user query that contains a question about the weather along with an image of a location that the model can infer location (i.e. New York City) from:

```
curl https://api.groq.com/openai/v1/chat/completions -s \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $GROQ_API_KEY" \
```

```
-d '{
"model": "llama-3.2-11b-vision-preview",
"messages": [
{
  "role": "user",
  "content": [{"type": "text", "text": "Whats the weather like in this state?"}, {"type": "image_url", "image_url": { "url": "https://cdn.britannica.com/61/93061-050-99147DCE/Statue-of-Liberty-Island-New-York-Bay.jpg"}}]
}
],
"tools": [
{
  "type": "function",
  "function": {
  "name": "get_current_weather",
  "description": "Get the current weather in a given location",
  "parameters": {
    "type": "object",
    "properties": {
    "location": {
      "type": "string",
      "description": "The city and state, e.g. San Francisco, CA"
    },
    "unit": {
      "type": "string",
      "enum": ["celsius", "fahrenheit"]
    }
    },
    "required": ["location"]
```

```
    }
    }
}
],
"tool_choice": "auto"
}' | jq '.choices[0].message.tool_calls'
```

The following is the output from our example above that shows how our model inferred the state as New York from the given image and called our example function:

```
[
 {
   "id": "call_q0wg",
   "function": {
    "arguments": "{\"location\": \"New York, NY\",\"unit\": \"fahrenheit\"}",
    "name": "get_current_weather"
   },
   "type": "function"
 }
]
```

JSON Mode with Images

The llama-3.2-11b-vision-preview model supports JSON mode! The following Python example queries the model with an image and text (i.e. "Please pull out relevant information as a JSON object.") with response_format set for JSON mode:

```
from groq import Groq

client = Groq()
completion = client.chat.completions.create(
   model="llama-3.2-11b-vision-preview",
```

```
  messages=[
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "Please pull out relevant information as a JSON object. "
        },
        {
          "type": "image_url",
          "image_url": {
            "url": "${IMAGE_DATA_URL}"
          }
        }
      ]
    }
  ],
  temperature=1,
  max_tokens=1024,
  top_p=1,
  stream=False,
  response_format={"type": "json_object"},
  stop=None,
)

print(completion.choices[0].message)
```

Multi-turn Conversations with Images

The llama-3.2-11b-vision-preview model supports multi-turn conversations! The following Python example shows a multi-turn user conversation about an image:

```
from groq import Groq


client = Groq()

completion = client.chat.completions.create(
  model="llama-3.2-11b-vision-preview",
  messages=[
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "what companies are listed in this image?"
        },
        {
          "type": "image_url",
          "image_url": {
            "url": "${IMAGE_DATA_URL}"
          }
        }
      ]
    },
    {
      "role": "user",
      "content": "what is groq?"
    }
  ],
  temperature=1,
```

```
    max_tokens=1024,
    top_p=1,
    stream=False,
    stop=None,
)

print(completion.choices[0].message)
```

Venture Deeper into Vision

Use Cases to Explore

Vision models can be used in a wide range of applications. Here are some ideas:

Accessibility Applications: Develop an application that generates audio descriptions for images by using a vision model to generate text descriptions for images, which can then be converted to audio with one of our audio endpoints.

E-commerce Product Description Generation: Create an application that generates product descriptions for e-commerce websites.

Multilingual Image Analysis: Utilize the multilingual capabilities of Llama 3.2 to create applications that can describe images in multiple languages.

Multi-turn Visual Conversations: Leverage the capavilities of Llama 3.2 to develop interactive applications that allow users to have extended conversations about images.

These are just a few ideas to get you started. The possibilities are endless, and we're excited to see what you create with vision models powered by Groq for low latency and fast inference!