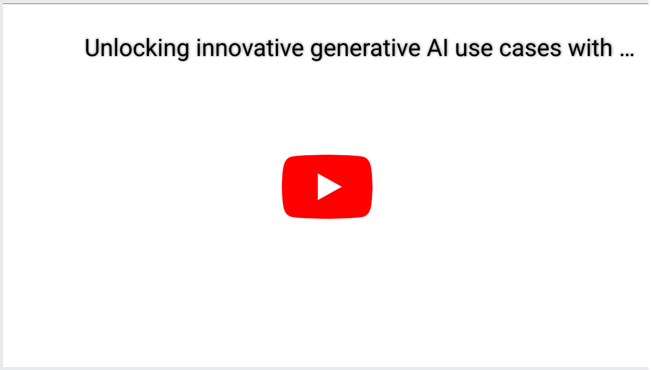
Get multimodal embeddings

To see an example of multimodal embeddings, run the "Intro to multimodal embeddings" Jupyter notebook in one of the following environments:

Open in Colab (https://colab.research.google.com/github/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb) | Open in Colab Enterprise (https://console.cloud.google.com/vertex-ai/colab/import/https%3A%2F%2Fraw.githubusercontent.com%2FGoogleCloudPlatform%2Fgenerative-ai%2Fmain%2Fembeddings%2Fintro_multimodal_embeddings.ipynb) | Open in Vertex AI Workbench user-managed notebooks (https://console.cloud.google.com/vertex-ai/workbench/deploy-notebook?download_url=https%3A%2F%2Fraw.githubusercontent.com%2FGoogleCloudPlatform%2Fgenerative-ai%2Fmain%2Fembeddings%2Fintro_multimodal_embeddings.ipynb) | View on GitHub (https://github.com/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb)



The multimodal embeddings model generates 1408-dimension vectors* based on the input you provide, which can include a combination of image, text, and video data. The embedding vectors can then be used for subsequent tasks like image classification or video content moderation.

The image embedding vector and text embedding vector are in the same semantic space with the same dimensionality. Consequently, these vectors can be used interchangeably for use cases like searching image by text, or searching video by image.

For text-only embedding use cases, we recommend using the Vertex AI text-embeddings API instead. For example, the text-embeddings API might be better for text-based semantic search, clustering, long-form document analysis, and other text retrieval or question-answering use cases. For more information, see Get text embeddings (/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings).

Supported models

You can get multimodal embeddings by using the following model:

- `multimodalembedding`

Best practices

Consider the following input aspects when using the multimodal embeddings model:

- **Text in images** - The model can distinguish text in images, similar to optical character recognition (OCR). If you need to distinguish between a description of the image content and the text within an image, consider using prompt engineering to specify your target content. For example: instead of just "cat", specify "picture of a cat" or "the text 'cat'", depending on your use case.

the text 'cat'

picture of a cat

Image credit: Manja Vitolic (https://unsplash.com/photos/gKXKBY-C-Dk) *on Unsplash* (https://unsplash.com/).

- **Embedding similarities** - The dot product of embeddings isn't a calibrated probability. The dot product is a similarity metric and might have different score distributions for different use cases. Consequently, avoid using a fixed value threshold to measure quality. Instead, use ranking approaches for retrieval, or use sigmoid for classification.

API usage

API limits

The following limits apply when you use the `multimodalembedding` model for text and image embeddings:

Limit	Value and description
	Text and image data
Maximum number of API requests per minute per project	120
Maximum text length	32 tokens (~32 words) The maximum text length is 32 tokens (approximately 32 words). If the input exceeds 32 tokens, the model internally shortens the input to this length.
Language	English
Image formats	BMP, GIF, JPG, PNG
Image size	Base64-encoded images: 20 MB (when transcoded to PNG) Cloud Storage images: 20MB (original file format) The maximum image size accepted is 20 MB. To avoid increased network latency, use smaller images. Additionally, the model resizes images to 512 x 512 pixel resolution. Consequently, you don't need to provide higher resolution images.
	Video data
Audio supported	N/A - The model doesn't consider audio content when generating video embeddings
Video formats	AVI, FLV, MKV, MOV, MP4, MPEG, MPG, WEBM, and WMV
Maximum video length (Cloud Storage)	No limit. However, only 2 minutes of content can be analyzed at a time.

Before you begin

1. In the Google Cloud console, on the project selector page, select or create a Google Cloud project.

   Go to project selector (https://console.cloud.google.com/projectselector2/home/dashboard)

2. Make sure that billing is enabled for your Google Cloud project (/billing/docs/how-to/verify-billing-enabled#confirm_billing_is_enabled_on_a_project).

3. Enable the Vertex AI API.

   Enable the API (https://console.cloud.google.com/flows/enableapi?apiid=aiplatform.googleapis.com)

★ Certain tasks in Vertex AI require that you use additional Google Cloud products besides Vertex AI. For example, in most cases, you must use Cloud Storage and Artifact Registry when you create a custom training pipeline. You might need to perform additional setup tasks to use other Google Cloud products.

4. Set up authentication for your environment.

   Select the tab for how you plan to use the samples on this page:

   Java (#java) Node.js (#node.js) Python (#python) REST (#rest)

   To use the Python samples on this page in a local development environment, install and initialize the gcloud CLI, and then set up Application Default Credentials with your user credentials.

   a. Install (/sdk/docs/install) the Google Cloud CLI.

   b. To initialize (/sdk/docs/initializing) the gcloud CLI, run the following command:

   ```
   gcloud init
   ```

   c. Update and install `gcloud` components:

   ```
   gcloud components update
   gcloud components install beta
   ```

   d. If you're using a local shell, then create local authentication credentials for your user account:

   ```
   gcloud auth application-default login
   ```

   You don't need to do this if you're using Cloud Shell.

   For more information, see Set up ADC for a local development environment (/docs/authentication/set-up-adc-local-dev-environment) in the Google Cloud authentication documentation.

5. To use the Python SDK, follow instructions at Install the Vertex AI SDK for Python (/vertex-ai/docs/start/install-sdk). For more information, see the Vertex AI SDK for Python API reference documentation (/python/docs/reference/aiplatform/latest).

6. Optional. Review pricing (/vertex-ai/generative-ai/pricing) for this feature. Pricing for embeddings depends on the type of data you send (such as image or text), and also depends on the mode you use for certain data types (such as Video Plus, Video Standard, or Video Essential).

Locations

A location is a region (/about/locations) you can specify in a request to control where data is stored at rest. For a list of available regions, see Generative AI on Vertex AI locations (/vertex-ai/generative-ai/docs/learn/locations-genai).

Error messages

Quota exceeded error

```
google.api_core.exceptions.ResourceExhausted: 429 Quota exceeded for
aiplatform.googleapis.com/online_prediction_requests_per_base_model with base
```

```
model: multimodalembedding. Please submit a quota increase request.
```

If this is the first time you receive this error, use the Google Cloud console to request a quota increase (/docs/quotas/view-manage#requesting_higher_quota) for your project. Use the following filters before requesting your increase:

- `Service ID: aiplatform.googleapis.com`
- `metric: aiplatform.googleapis.com/online_prediction_requests_per_base_model`
- `base_model:multimodalembedding`

Go to Quotas (https://console.cloud.google.com/iam-admin/quotas?service=aiplatform.googleapis.com&metric=aiplatform.googleapis.com/online_prediction_re

If you have already sent a quota increase request, wait before sending another request. If you need to further increase the quota, repeat the quota increase request with your justification for a sustained quota request.

Specify lower-dimension embeddings

By default an embedding request returns a 1408 float vector for a data type. You can also specify lower-dimension embeddings (128, 256, or 512 float vectors) for text and image data. This option lets you optimize for latency and storage or quality based on how you plan to use the embeddings. Lower-dimension embeddings provide decreased storage needs and lower latency for subsequent embedding tasks (like search or recommendation), while higher-dimension embeddings offer greater accuracy for the same tasks.

REST (#rest)PythonGo (#go)
(#python)

```python
import vertexai

from vertexai.vision_models import Image, MultiModalEmbeddingModel

# TODO(developer): Update & uncomment line below
# PROJECT_ID = "your-project-id"
vertexai.init(project=PROJECT_ID, location="us-central1")

# TODO(developer): Try different dimenions: 128, 256, 512, 1408
embedding_dimension = 128

model = MultiModalEmbeddingModel.from_pretrained("multimodalembedding@001")
image = Image.load_from_file(
    "gs://cloud-samples-data/vertex-ai/llm/prompts/landmark1.png"
)

embeddings = model.get_embeddings(
    image=image,
    contextual_text="Colosseum",
    dimension=embedding_dimension,
)

print(f"Image Embedding: {embeddings.image_embedding}")
print(f"Text Embedding: {embeddings.text_embedding}")

# Example response:
# Image Embedding: [0.0622573346, -0.0406507477, 0.0260440577, ...]
# Text Embedding: [0.27469793, -0.146258667, 0.0222803634, ...]
```

Send an embedding request (image and text)

Note: For text-only embedding use cases, we recommend using the Vertex AI text-embeddings API instead. For example, the text-embeddings API might be better for text-based semantic search, clustering, long-form document analysis, and other text retrieval or question-answering use cases. For more information, see Get text embeddings (/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings).

Use the following code samples to send an embedding request with image and text data. The samples show how to send a request with both data types, but you can also use the service with an individual data type.

Get text and image embeddings

REST (#rest)PythonNode.js (#node.js)Java (#java)Go (#go)
(#python)

To learn how to install or update the Vertex AI SDK for Python, see Install the Vertex AI SDK for Python (/vertex-ai/docs/start/use-vertex-ai-python-sdk). For more information, see the Python API reference documentation (/python/docs/reference/aiplatform/latest).

```
import vertexai
from vertexai.vision_models import Image, MultiModalEmbeddingModel

# TODO(developer): Update & uncomment line below
# PROJECT_ID = "your-project-id"
vertexai.init(project=PROJECT_ID, location="us-central1")

model = MultiModalEmbeddingModel.from_pretrained("multimodalembedding@001")
image = Image.load_from_file(
    "gs://cloud-samples-data/vertex-ai/llm/prompts/landmark1.png"
)

embeddings = model.get_embeddings(
    image=image,
    contextual_text="Colosseum",
    dimension=1408,
)
print(f"Image Embedding: {embeddings.image_embedding}")
print(f"Text Embedding: {embeddings.text_embedding}")
# Example response:
# Image Embedding: [-0.0123147098, 0.0727171078, ...]
# Text Embedding: [0.00230263756, 0.0278981831, ...]
```

Send an embedding request (video, image, or text)

When sending an embedding request you can specify an input video alone, or you can specify a combination of video, image, and text data.

Video embedding modes

There are three modes you can use with video embeddings: Essential, Standard, or Plus. The mode corresponds to the density of the embeddings generated, which can be specified by the `interval_sec` config in the request. For each video interval with `interval_sec` length, an embedding is generated. The minimal video interval length is 4 seconds. Interval lengths greater than 120 seconds might negatively affect the quality of the generated embeddings.

Pricing for video embedding depends on the mode you use. For more information, see pricing (/vertex-ai/generative-ai/pricing).

The following table summarizes the three modes you can use for video embeddings:

Mode	Maximum number of embeddings per minute	Video embedding interval (minimum value)
Essential	4	15 This corresponds to: `intervalSec` >= 15

Mode	Maximum number of embeddings per minute	Video embedding interval (minimum value)
Standard	8	8 This corresponds to: 8 <= `intervalSec` < 15
Plus	15	4 This corresponds to: 4 <= `intervalSec` < 8

Video embeddings best practices

Consider the following when you send video embedding requests:

- To generate a single embedding for the first two minutes of an input video of any length, use the following `videoSegmentConfig` setting:

  `request.json`:

```
// other request body content
"videoSegmentConfig": {
  "intervalSec": 120
}
// other request body content
```

- To generate embedding for a video with a length greater than two minutes, you can send multiple requests that specify the start and end times in the `videoSegmentConfig`:

  `request1.json`:

```
// other request body content
"videoSegmentConfig": {
  "startOffsetSec": 0,
  "endOffsetSec": 120
}
// other request body content
```

  `request2.json`:

```
// other request body content
"videoSegmentConfig": {
  "startOffsetSec": 120,
  "endOffsetSec": 240
}
// other request body content
```

Get video embeddings

Use the following sample to get embeddings for video content alone.

REST (#rest)Python (#python)Go (#go)

To learn how to install or update the Vertex AI SDK for Python, see Install the Vertex AI SDK for Python (/vertex-ai/docs/start/use-vertex-ai-python-sdk). For more information, see the Python API reference documentation (/python/docs/reference/aiplatform/latest).

```
import vertexai

from vertexai.vision_models import MultiModalEmbeddingModel, Video
from vertexai.vision_models import VideoSegmentConfig

# TODO(developer): Update & uncomment line below
# PROJECT_ID = "your-project-id"
vertexai.init(project=PROJECT_ID, location="us-central1")

model = MultiModalEmbeddingModel.from_pretrained("multimodalembedding@001")

embeddings = model.get_embeddings(
    video=Video.load_from_file(
        "gs://cloud-samples-data/vertex-ai-vision/highway_vehicles.mp4"
    ),
    video_segment_config=VideoSegmentConfig(end_offset_sec=1),
)

# Video Embeddings are segmented based on the video_segment_config.
print("Video Embeddings:")
for video_embedding in embeddings.video_embeddings:
    print(
        f"Video Segment: {video_embedding.start_offset_sec} - {video_embedding.end_offset_sec}"
    )
    print(f"Embedding: {video_embedding.embedding}")

# Example response:
# Video Embeddings:
# Video Segment: 0.0 - 1.0
# Embedding: [-0.0206376351, 0.0123456789, ...]
```

Get image, text, and video embeddings

Use the following sample to get embeddings for video, text, and image content.

REST (#rest)PythonGo (#go)
(#python)

To learn how to install or update the Vertex AI SDK for Python, see Install the Vertex AI SDK for Python (/vertex-ai/docs/start/use-vertex-ai-python-sdk). For more information, see the Python API reference documentation (/python/docs/reference/aiplatform/latest).

```
import vertexai

from vertexai.vision_models import Image, MultiModalEmbeddingModel, Video
from vertexai.vision_models import VideoSegmentConfig

# TODO(developer): Update & uncomment line below
# PROJECT_ID = "your-project-id"
vertexai.init(project=PROJECT_ID, location="us-central1")

model = MultiModalEmbeddingModel.from_pretrained("multimodalembedding@001")

image = Image.load_from_file(
    "gs://cloud-samples-data/vertex-ai/llm/prompts/landmark1.png"
)
video = Video.load_from_file(
    "gs://cloud-samples-data/vertex-ai-vision/highway_vehicles.mp4"
)

embeddings = model.get_embeddings(
    image=image,
    video=video,
    video_segment_config=VideoSegmentConfig(end_offset_sec=1),
    contextual_text="Cars on Highway",
)

print(f"Image Embedding: {embeddings.image_embedding}")

# Video Embeddings are segmented based on the video_segment_config.
print("Video Embeddings:")
for video_embedding in embeddings.video_embeddings:
    print(
        f"Video Segment: {video_embedding.start_offset_sec} - {video_embedding.end_offset_sec}"
    )
    print(f"Embedding: {video_embedding.embedding}")

print(f"Text Embedding: {embeddings.text_embedding}")
# Example response:
# Image Embedding: [-0.0123144267, 0.0727186054, 0.000201397663, ...]
# Video Embeddings:
# Video Segment: 0.0 - 1.0
# Embedding: [-0.0206376351, 0.0345234685, ...]
# Text Embedding: [-0.0207006838, -0.00251058186, ...]
```

What's next

- Read the blog "What is Multimodal Search: 'LLMs with vision' change businesses" (https://cloud.google.com/blog/products/ai-machine-learning/multimodal-generative-ai-search).
- For information about text-only use cases (text-based semantic search, clustering, long-form document analysis, and other text retrieval or question-answering use cases), read Get text embeddings (/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings).
- View all Vertex AI image generative AI offerings in the Imagen on Vertex AI overview (/vertex-ai/generative-ai/docs/image/overview).
- Explore more pretrained models in Model Garden (/vertex-ai/generative-ai/docs/model-garden/explore-models).
- Learn about responsible AI best practices and safety filters in Vertex AI (/vertex-ai/generative-ai/docs/learn/responsible-ai).