Google Cloud

Vertex AI Gemini

Multimodal prompting

April 2025 Edition

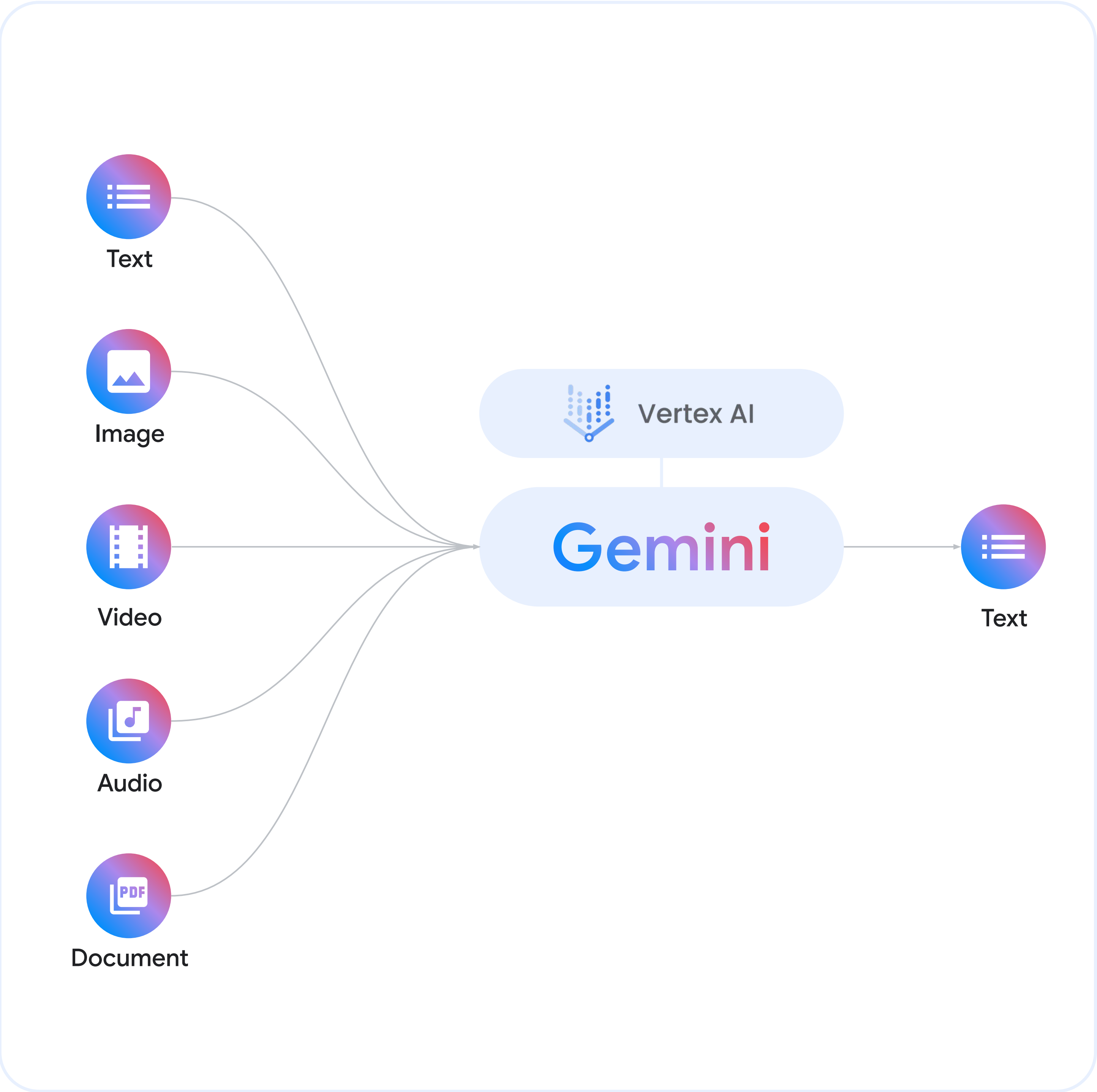
Gemini Models on Vertex AI

	Gemini 1.5 Pro	Gemini 1.5 Flash	Gemini 2.0 Flash	Gemini 2.0 Flash Lite
Input modalities	text, documents, image, video, audio, pdf	text, documents, image, video, audio, pdf	text, documents, image, video, audio, pdf	text, documents, image, video, audio, pdf
Output modalities	text	text	Text, audio (exp)	text
Context window, total token limit	2,097,152	1,048,576	1,048,576	1,048,576
Output context length	8,192	8,192	8,192	8,192
Grounding with Search	Y	Y	Y	N
Context Caching	Y	Y	Y	N
Function calling	Y	Y	Y	Y
Max Limits	Images: 7200 Image size: 20MB Videos: 10 Video length: ~1.8 hrs Audio: 1 Audio length: ~19 hrs Pages per PDF: 1000 Max PDF size: 30 MB	Images: 3000 Image size: 20MB Videos: 10 Video length: 1 hr Audio: 1 Audio length: ~8.4hr Pages per PDF: 1000 Max PDF size: 30 MB	Images: 3000 Image size: 7MB Videos: 10 Video length: ~45mins (with audio)/1hr (without audio) Audio: 1 Audio length: ~8.4hr Pages per PDF: 1000 Max PDF size: 50 MB	Images: 3000 Image size: 7MB Videos: 10 Video length: ~45mins (with audio)/1hr (without audio) Audio: 1 Audio length: ~8.4hr Pages per PDF: 1000 Max PDF size: 50 MB
Code execution	N	N	Y	N
Live API	N	N	Y	N
Latency	Most capable in 1.5 family	Fastest in 1.5 family	More capable but slower than Lite	Fastest+cost efficient
Fine tuning	Y	Y	Y	N
SDK	Vertex AI SDK	Vertex AI SDK	Gen AI SDK	Gen AI SDK
Pricing units	Character	Character	Token	Token

Contents

Gemini Multimodal Prompting

Multimodal prompts combine multiple input formats - text, image, video, audio, document - **and generate a text-only output.**



01.

General tips for Multimodal prompting

Multimodal.Tip #1

Craft clear and concise instructions

- As with general prompting strategies, remember to be **specific and explicit** with your instructions.
- Use **prompt components** as needed (persona, task) with structured or well-formatted **instructions**.

```
You are a zoologist whose job is to name
animals in the image. You should always
provide an english and latin name.
```

```
1. American Bison, Bison bison

2. Lesser Mouse Deer, Tragulus javanicus

3. Giraffe, Giraffa camelopardalis

4. Reindeer, Rangifer tarandus

5. Chinese Water Deer, Hydropotes inermis

6. Pronghorn, Antilocapra americana
```

Multimodal.Tip #2

Add your media first for single-media prompts

- If your **prompt contains a single media file** - image, video or audio or document - **place the media file before the text prompt**.
- While Gemini can handle media and text inputs in any order, for prompts containing a single media, it might perform better if that media file is placed before the text prompt.
- For **prompts** with **highly interleaved media files with texts, order** them in the most **logical way**.

```
input_image = Part.from_data(
   data=base64.b64decode(encoded_image),
   mime_type="image/jpeg")

input_prompt = "..."

MODEL_ID = "gemini-2.0-flash"
model = GenerativeModel(MODEL_ID)

responses = model.generate_content(
  contents=[input_image, input_prompt],
  generation_config=generation_config,
  safety_settings=safety_settings
)

for response in responses:
  print(response.text, end="")
```

Multimodal.Tip #3

Use System Instructions to shape the models response

- System instructions guide the model's behavior before it addresses end user prompts, allowing you to define model roles and context, and desired output format.
- By setting these instructions, you effectively steer the model's responses and ensure more consistent behavior across multiple turns. This approach is particularly useful for establishing a specific persona or maintaining a consistent style.

```
system_prompt = """You are good at looking at pictures and
uncovering the full story within a visual scene. Your task is to
provide a rich and insightful description of the image"""

input_image = Part.from_data(
   data=base64.b64decode(encoded_image),
   mime_type="image/jpeg")

input_prompt = "Describe what is depicted on the image"

MODEL_ID = "gemini-2.0-flash"
model = GenerativeModel(model_name=MODEL_ID,
                        system_instruction=system_prompt)

responses = model.generate_content(
  contents=[input_image, input_prompt],
  generation_config=generation_config,
  safety_settings=safety_settings
)

for response in responses:
  print(response.text, end="")
```

Multimodal.Tip #4

Add examples to the prompt

- Use **realistic few-shot examples** to illustrate how you would like the task to be accomplished with the **example prompt followed by the example response.**
- The illustrated examples steer the model towards the response style or format you like.

```
You are an expert nutritionist. Your task is to read the label and extract
all entities about the product in the label. Follow the example below:

# EXAMPLE 1
{
  "ingredients": ["Sugar",
      ...],
  "allergen_information": {
     "contains": ["Added ..."],
     "may_contain": []
   }
}
```


```
# EXAMPLE 2
{
  "ingredients": ["Cow's Milk", ...],
  "allergen_information": {
    "contains": ["Milk … "],
    "may_contain": []
  }
}
```


```
Now your turn!
```


```
{
  "ingredients": [
    "Maida", "Sugar", "Eggs", "Ghee",…"Permitted Additive (E 422)"
  ],
  "allergen_information": {
    "contains": ["Dryfruits", "Nuts"],
    "may_contain": ["dairy", "tree nuts", "peanuts", "gluten" …]
  }
}
```

Multimodal.Tip #5

Break it down step-by-step

Divide complex tasks into manageable sub-tasks, guiding the model through the process.

```
Analyze the following image containing fruits.
```

Image

```
How much should I pay for the fruits given the following
price list?
```

List

```
Answer the question through these steps:

1. Identify all kind of fruits in the image.

2. Count the quantity of each fruit.

3. For each fruit identified, check the price
from the price list.

4. Calculate subtotal price for each type of fruit.

5. Calculate total price of fruits using subtotals.


Answer:
```

	A	B
1	Fruit	Price per item
2	Apples	$1.50
3	Bananas	$0.80
4	Kiwi	$1.25
5	Mangoes	$2.00
6	Oranges	$0.99
7	Peaches	$1.75
8	Watermelon	$4.00

```
Let's calculate the total price:

1. Fruits in the image: Bananas, Apples

2. Quantity of each fruit: Bananas: 2, Apples: 3

3. Price per fruit: Bananas: $0.80 each, Apples: $1.50 each

4. Subtotal price for each fruit:

     Bananas: 2 bananas * $0.80/banana = $1.60

     Apples: 3 apples * $1.50/apple = $4.50

5. Total price: $1.60 (bananas) + $4.50 (apples) = $6.10

Therefore, you should pay $6.10 for the fruits.
```

Multimodal.Tip #6

Specify the output format

Ask for the output to be in the format you want - Markdown, JSON, HTML and more.

This is referred to as Controlled Generation, which can be used directly in the prompt, Vertex AI Studio or through the API **with response_schema** defined.

```
Analyze the following image containing fruits.
```

Image

```
How much should I pay for the fruits given the following
price list?
```

List

```
Answer the question through these steps:
1. Identify all kind of fruits in the image.
2. Count the quantity of each fruit.
3. For each fruit identified, check the price
from the price list.
4. Calculate subtotal price for each type of fruit.
5. Calculate total price of fruits using subtotals.

Return answer as JSON

{"items": [{"name": string, "quantity": int,
"subtotal_price": numeric}], "total_price": number}

Answer:
```

	A	B
1	Fruit	Price per item
2	Apples	$1.50
3	Bananas	$0.80
4	Kiwi	$1.25
5	Mangoes	$2.00
6	Oranges	$0.99
7	Peaches	$1.75
8	Watermelon	$4.00

```
{
  "items": [
  {
      "name": "Apples",
      "quantity": 2,
      "subtotal_price": 3.00
    },
    {
      "name": "Bananas",
      "quantity": 2,
      "subtotal_price": 1.60
    }
  ],
  "total_price": 4.60
}
```

Multimodal.Tip #7

Use Context Caching for repeated queries

Ideal Use Cases:

- Chatbots with lengthy instructions
- Recurring video or document analysis
- Frequent code repository analysis

How it Works:

- Create a cache of your long input.
- Reference this cache by its resource name in subsequent prompt requests.
- Optionally set a cache expiration time.

Benefits:

- **Cost Savings:** Avoid reprocessing large inputs in every request.
- **Reduced Latency:** Faster processing when asking multiple questions about the same long content.
- **Significant Savings:** Particularly beneficial for longer media.

Consider enabling context caching when asking multiple questions about long videos, audio, or documents to optimize costs and potentially reduce processing time.

Multimodal.Tip #7

Using Context Caching

You can use Vertex AI SDK to implement context caching with a long video when prompting a Gemini model. The below sample illustrates creating a cache and then using it in a subsequent request to analyze the video content. You can replace video with other multimodal input types.

```python
from vertexai.generative_models import GenerativeModel, Part
from vertexai import caching

MODEL_ID = "gemini-2.0-flash"

# Prepare contents
video_file = Part.from_uri("gs://bucket/.../long_video.mp4",
                           mime_type="video/mp4")

# Create cached content
cache = caching.CachedContent.create(
    model_name=MODEL_ID,
    contents=[video_file],
    ttl=datetime.timedelta(minutes=60))
print(cache.name)

# Call Gemini with cached content
model = GenerativeModel.from_cached_content(cached_content=cache)

responses = model.generate_content("""Describe what happens in the
beginning, in the middle and in the end of the video. Also, list the
name of the main character and if he has any problems, list them as
well""")

for response in responses:
  print(response.text, end="")
```

Multimodal.Tip #8

Leveraging Semantic Cues For Effective LLM Instructions

- Enhance your prompting by including natural language cues that are specific to the task.

- These act as powerful semantic guides, much like nuanced phrasing influences human comprehension.

- For instance, in a "Yes/No" classification task where uncertainty might arise, explicitly prompting for an **"unmistakable"** answer can encourage the model to bias uncertain responses towards "No," reducing potential hallucinations.

02.

Working with Images

Image Requirements

Supported image types	JPEG, PNG, WEBP
Image Resolution	**Maximum image size:** 7MB
Image Token Usage	**High/Medium/Default Resolutions:** US/Asia: 40M | EU: 10M **Low Resolutions:** US/Asia: 10M | EU: 3M
Image Limit	Gemini 2.0 Flash allow up to 3000 images in a prompt request

General recommendations

Images with higher resolution yield better results.

Rotate images to their proper orientation before adding them to the prompt.

Avoid blurry images.

MM.Image.Tip #1

Add your media first for single-media prompts

- When there are multiple images in the prompt, **enumerate each image with an index,** like "Image 1:" and "Image 2:".
- You can **refer to them later** in your prompt or have the model refer to them in the model response.

```
image 1 <piano_recital.jpg>
image 2 <family_dinner.jpg>
image 3 <coffee_shop.jpg>

Write a blogpost about my day using image 1 and image 2.
Then, give me ideas
for tomorrow based on image 3.
```

MM.Image.Tip #2

Detecting Text in Images

For text detection within an image, **prompts using a single image generally yield better results** compared to those with multiple images.

IMPORTANT: Higher image resolution generally leads to more accurate text detection.

```
Your task is to analyze the
given document and extract
structured data from bills,
receipts, and invoices using
following instructions.
```

PURCHASE ORDER

ACME, INC	
456 Model Garden	
Codey City, BY, 67890	
Ph: (222) - 345 - 6666	
Fax: (222) - 345 - 6000	
Email: buyer1@acmeinc.com	

DATE	9/1/2023
PO NO	PO-2023-A123

Vendor:	Ship To:
LLM, INC	Attn: BERT SIMPSON
123 Bison Street	ACME, INC
Gecko City, ST 12345	456 Model Garden St
Phone: (123) 456-7890	Codey City, BY, 67890
Fax: (123) 456 - 7800	Ph: (222) - 345 - 6666
Email: langchain@llminc.com	Fax: (222) - 345 - 6000
	Email: buyer1@acmeinc.com

Department	Requested By	Payment Terms	Delivery Date
Engineering	Bert Simpson	Net 15 Days	9/25/2023

Item #	Description	Qty	Unit Price	Total
A233	15" LED Monitor	1	$200.00	$200.00
B124	Vertical Mounting Stand	2	$100.00	$200.00
			Tax Rate	10%
			Taxes	$40
			Shipping & Handling	$0
			Total Due	**$440.00**

```
<instructions>
1. Analyze and Determine document type
(bill, receipt, etc.). Identify key sections
(header, itemized list, footer)
2. Extract the following elements
  - Vendor/Merchant Name
  - Transaction/Issue Date
  - Transaction Details:
   - Items, Quantities, Prices
   - Subtotals (if available)
   - Total Amount
   - Payment Method (if specified)
  - Other relevant details (customer info, order numbers, taxes,
discounts, terms, etc.) under a separate section.
3. Handle the challenges in the document input:
  - Adapt to various document formats & layouts
  - Double-check extracted data
  - Use context for unclear information, note uncertainties
</instructions>
Return extracted information in JSON format.
```

MM.Image.Tip #3

Detecting Objects in Images with 2D bounding boxes

You can prompt Gemini for **object detection** and **localization** in images and videos using **bounding boxes.** The model outputs bounding box coordinates in the `[ymin,xmin,ymax,xmax]` format.

For best results, follow these prompt engineering guidelines:

- Strong prompts also **include examples** with a few-shot prompt format. Specify output format directly if needed.
- Use explicit object references and specify the desired bounding box format.
- Clearly state the objects you want detected and provide clear output format instructions in your prompt to ensure correct triggering.
- Use specific keywords like **"detect"** and **"pinpoint the bounding boxes"** in the prompt.

Handling the bounding box output from Gemini

- Use JSON for output, by default it returns a list of objects with each entry containing **"box_2d"** and **"label"**.
- Predicted bounding boxes are in **normalized coordinates [1000, 1000]**. Rescale the coordinates to outlay bounding boxes on the original image.
- Output formats should include suffixes like **"Is it possible to use (bbox), (predicted_label), and (detected_mask) as the keys?"**

MM.Image.Tip #3

Detecting Objects in Images with 2D bounding boxes

Let’s put the prompting guidelines for bounding boxes together with an example below:

```
Detect the bounding boxes of the
handwritten terms and identify the objects
on the castle. If there are more than one
instance of an object, add them to the JSON
list of objects with keys as “box_2d” and
“labels”.

Follow the example below:

[{"box_2d" : [ymin, xmin, ymax, xmax],
“label”: “name of detected object},
{"box_2d" : [ymin, xmin, ymax, xmax],
“label”: “name of detected object}]
```


```
[
  {"box_2d": [850, 243, 904, 488], "label": "drawbridge?"},

  {"box_2d": [806, 511, 863, 744], "label": "portcullis?"},

  {"box_2d": [894, 335, 966, 455], "label": "turret?"},

  {"box_2d": [894, 501, 972, 708], "label": "Battlements?"}
]
```

MM.Image.Tip #4

Detecting Points and Spatial Relationships

Precise entity reference via pointing is a key capability for Gemini 2.0 Flash, which features **improved spatial understanding with point prediction and reasoning in 3D space.**

For best results, follow these prompt engineering guidelines:

- Points are less cluttered than bounding boxes and can be sufficient for some use cases.
- Use temperatures greater than 0 to reduce repetition e.g. 0.5
- Limit the number of items (e.g., to 10) to prevent looping and speed up coordinate decoding.
- Gemini can search and reason about different views of the same 3D scene. When searching for objects within a scene, be as specific as possible to help guide the model.

Handling the points output from Gemini

- Use JSON for output, by default it returns a list of objects with each entry containing **"point"** and **"label"**. The points are in [y, x] format normalized to (0, 1000).
- Use advanced reasoning to combine points by adding the following to your prompt; **"Explain how to use each part, put them in the label field, remove duplicated parts and instructions"**

MM.Image.Tip #4

Detecting Points and Spatial Relationships

Let's put the prompting guidelines for 3D points and spatial relationships together with an example below:

```
Point to no more than 10 items in the
image, include spill.

The answer should follow the JSON format
[{“point”:, “label”:}]. The points are in
[y,x] format normalized to 0 - 1000.

Explain how to use each part, put them in
the label field, remove duplicated parts
and instructions.
```


```
[
  {"point": [374, 768], "label": "battlement, a defensive parapet at
the top of a wall or tower, typically with alternating indentations or
embrasures for shooting through"},
  {"point": [588, 449], "label": "portcullis, a strong, heavy grating
sliding up and down in vertical grooves, typically at the entrance to
a castle"},
  {"point": [748, 227], "label": "drawbridge, a bridge of which the
whole or a section can be raised or drawn aside to prevent access"},
  {"point": [273, 334], "label": "turret, a small tower on top of a
larger tower or at the corner of a building or wall, typically of a
castle"}
]
```

MM.Image.Tip #5

Detecting 3D Bounding Boxes

3D bounding boxes is a new experimental feature from Gemini 2.0 that will continue to improve in future models. It is represented by nine numbers:

- **Metric center -** `(x, y, z)`
- **Metric size -** `(l, w, h)`
- **Orientation -** `(roll, pitch, yaw) all in degrees`

For best results, follow these prompt engineering guidelines:

- **Using system Instructions is highly encouraged.** As an experimental feature, system instructions will steer the model to returning the right bounding boxes
- **Be specific.** Tell the model exactly what you need for the output format (recommended JSON).
- Use temperatures greater than 0 (e.g., 0.5) to reduce repetition.
- Limit the number of items (e.g. up to 10) to prevent looping and speed up decoding.
- **Do not be afraid to experiment.** You can experiment with these parameters to find what works best for your use-case.

Handling the 3D bounding box output from Gemini

- Use JSON for output, by default it returns a list of objects with each entry containing **"box_3d"** and **"label"**. The bounding boxes are represented by 9 numbers; center, size and orientation.
- Specify the output format in your prompt. For example: **"The 3D bounding box format should be [x_center, y_center, z_center, x_size, y_size, z_size, roll, pitch, yaw]"**

MM.Image.Tip #5

Detecting 3D Bounding Boxes

Let's put the prompting guidelines for 3D bounding box detection together with an example below:

System Instruction: You are an expert at detecting 3D bounding boxes. Your output format is nine numbers represented as [x_center, y_center, z_center, x_size, y_size, z_size, roll, pitch, yaw].

Prompt: Detect the 3D bounding boxes of the handwritten terms and identify the objects on the castle. Output a json list where each entry contains the object name in "label" and its 3D bounding box in "box_3d". The 3D bounding box format should be [x_center, y_center, z_center, x_size, y_size, z_size, roll, pitch, yaw].


```
[
  {"label": "turret", "box_3d":
[-0.4,2.14,0.23,0.68,0.73,1.35,-36,-2,0]},
  {"label": "turret", "box_3d":
[0.61,2.18,0.28,0.73,0.69,1.38,-36,-4,-3]},
  {"label": "drawbridge?", "box_3d":
[-0.28,1.71,-0.56,0.53,0.22,0.34,-36,0,2]},
  {"label": "battlements?", "box_3d":
[0.14,1.72,-0.61,0.74,0.24,0.39,-36,-1,0]},
  {"label": "portcullis?", "box_3d":
[0.04,1.74,-0.56,0.48,0.21,0.31,-36,0,3]}
]
```

MM.Image.Tip #6

Guiding the models' attention

Specify the **relevant parts of the image** you want the prompt to draw information from to ensure accurate and targeted results.

```
How long will these diapers last before
I run out?

Use the weight shown on the box to
determine the child's age, and use the
total number of diapers in the box.

Divide the total number by how many
diapers the child goes through per day.
```


```
The box contains 198 diapers. The box states the diapers
are for a child weighing 22-37 lbs.

The average child in this weight range is likely between
1 and 2 years old.

A 2-year-old typically goes through 6 diapers per day,
therefore, the diapers will last around 33 days (198/6).
```

MM.Image.Tip #7

Optimize outputs by asking for detailed analysis

Begin your prompt by asking the model to **describe and extract relevant details** from the image/video. This improves understanding and ensures the model incorporates visual information into its tailored and specific output.

Things to know

Limitations with Images

Spatial Reasoning

Models may struggle with precise text or object location in images, and may only provide approximate object counts. Be as specific as possible to guide the model.

Medical Uses

Models are not suitable for interpreting medical images (e.g., x-rays, CT scans) or offering medical advice.

People Recognition

Models are not designed to identify non-celebrities in images.

Content Moderation

Models will not respond to images that violate safety policies.

Accuracy

Models may hallucinate or make errors with low-quality, rotated, or low-resolution images, or when interpreting handwritten text.

These limitations will be updated periodically to reflect ongoing model improvements.

03.

Working with Videos

Video Requirements

Supported formats	FLV, MOV, MPEG, MPEGPS, MPG, MP4, WEBM, WMV, 3GPP
Sampling frame rate	Videos are sampled at 1fps. Each video frame accounts for 258 tokens.
Audio Encoding	The audio track is encoded with video frames. The audio is split into 1s chunks (32 tokens each) and combined with video frames and timestamps (7 tokens each).

MM.Video.Tip #1

Timestamp localization

- For video timestamp localization, use timestamps in **MM:SS format** (minutes:seconds).
- Use the **same format for questions** that ask about a timestamp.

```
Break the video into segments based on
similar concepts. For each segment
provide a start and end time (MM:SS), a
brief description, and a concise
summary under 15 words.


Output format:
<start_timestamp-end_timestamp>:
<description> - <summary>
```


```
Here are the segments:


0:00-0:13: Rio de Janeiro Introduction - Welcome to the city and its beauty.

0:13-0:20: Famous Landmarks - Explore the well-known beaches and neighborhoods.

0:20-0:35: Exploring the Favelas - Unveiling the hidden community beyond headlines.

0:35-0:49: Favela Beyond the Stereotype - Beyond the struggles, find the life in
favelas.

0:49-0:57: Join Google's Expedition - Experience the favelas through Google's 360°
view.
```

MM.Video.Tip #2

Ask Gemini to focus on visual content

When summarizing well-known video clips, instruct the model to **"focus on the visual content, rather than outside knowledge"** to avoid hallucinations caused by relying on external knowledge.

MM.Video.Tip #3

Process long videos in segments for dense outputs

- When performing tasks with dense output on long videos (e.g., dense captioning, chapterization), **split the video into smaller segments** and process each segment with separate API calls.
- This **prevents hitting token limits** and potentially oversimplified results.
- Experiment to determine the **optimal segment length** for your specific task.

Things to know

Limitations with Video

Content moderation

The models refuse to provide answers on videos that violate our safety policies.

Non-speech sound recognition

The models might make mistakes recognizing sound that's not speech (background noise).

High-speed motion

Because of the fixed 1 frame per second (fps) sampling rate, the models might make mistakes understanding high-speed motion in video.

Transcription punctuation:

While Gemini 2.0 Flash offers significantly improved transcriptions compared to previous models, it's still important to verify that all punctuation is included.

These limitations will be updated periodically to reflect ongoing model improvements.

04.

Working with Audio

MM.Audio.Tip #1

For better audio transcription

"Transcribe this audio" can sometimes return a summary instead of the full transcription. Adding **"do not summarize"** to your prompt can help.

MM.Audio.Tip #2

Add examples for effective speaker diarization.

When segmenting audio by speaker labels, **provide few-shot examples** for better results.

```
audio <podcast_recording.mp4>


Your task is to analyze the podcast recording and provide
an exact transcription of all of the audio. Be concise and
short, no need to provide chapter summaries.

Format should be as follows:
[speaker 1][00:00-00:10] This is what the speaker said.
[speaker 2][00:11-00:20] This is what the second speaker
said.
[speaker 1][00:21-00:30] This is what the speaker 1 said
next.
```

Things to know

Limitations with Audio

Non-speech sound recognition

The models might make mistakes recognizing sound that's not speech (background noise).

These limitations will be updated periodically to reflect ongoing model improvements.

05.

Working with Documents

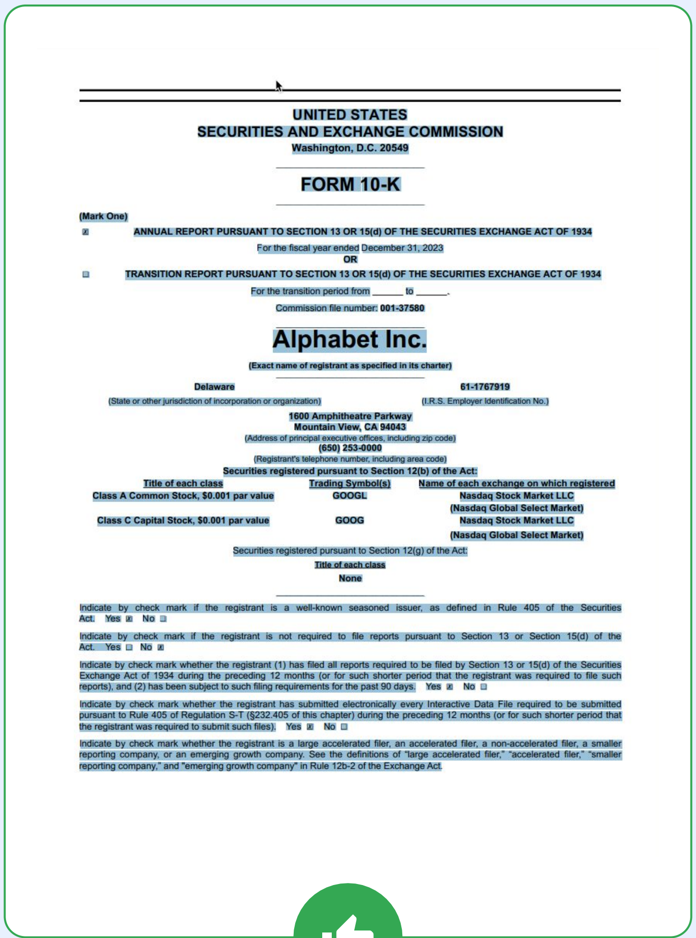
Document Requirements

Supported formats	PDF, TXT
Tokenization	Each page in PDF is tokenized as an image and the cost follows Gemini image pricing.
Size limit	Up to 1000 pages per PDF is supported (max PDF size allowed is 50MB per file) and number of pages allowed is limited to number of images the model supports. Maximum number of of files per prompt is 3000.

MM.Documents.Tip #1

Use Digital PDFs Over Scanned PDFs

- Gemini can process both native digital and scanned PDFs. **Using native digital PDFs is generally preferred.** For example, when working with text-heavy documents like contracts, use PDFs created digitally.
- Native digital PDFs often have **higher image quality**, leading to more accurate text extraction and reducing potential errors.



UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

FORM 10-K

(Mark One)

☒ ANNUAL REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934

For the fiscal year ended December 31, 2023

OR

☐ TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934

For the transition period from ______ to ______.

Commission file number: 001-37580

Alphabet Inc.

(Exact name of registrant as specified in its charter)

Delaware
(State or other jurisdiction of incorporation or organization)

61-1767919
(I.R.S. Employer Identification No.)

1600 Amphitheatre Parkway
Mountain View, CA 94043
(Address of principal executive offices, including zip code)
(650) 253-0000
(Registrant's telephone number, including area code)

Securities registered pursuant to Section 12(b) of the Act:

Title of each class	Trading Symbol(s)	Name of each exchange on which registered
Class A Common Stock, $0.001 par value	GOOGL	Nasdaq Stock Market LLC (Nasdaq Global Select Market)
Class C Capital Stock, $0.001 par value	GOOG	Nasdaq Stock Market LLC (Nasdaq Global Select Market)

Securities registered pursuant to Section 12(g) of the Act:

Title of each class

None

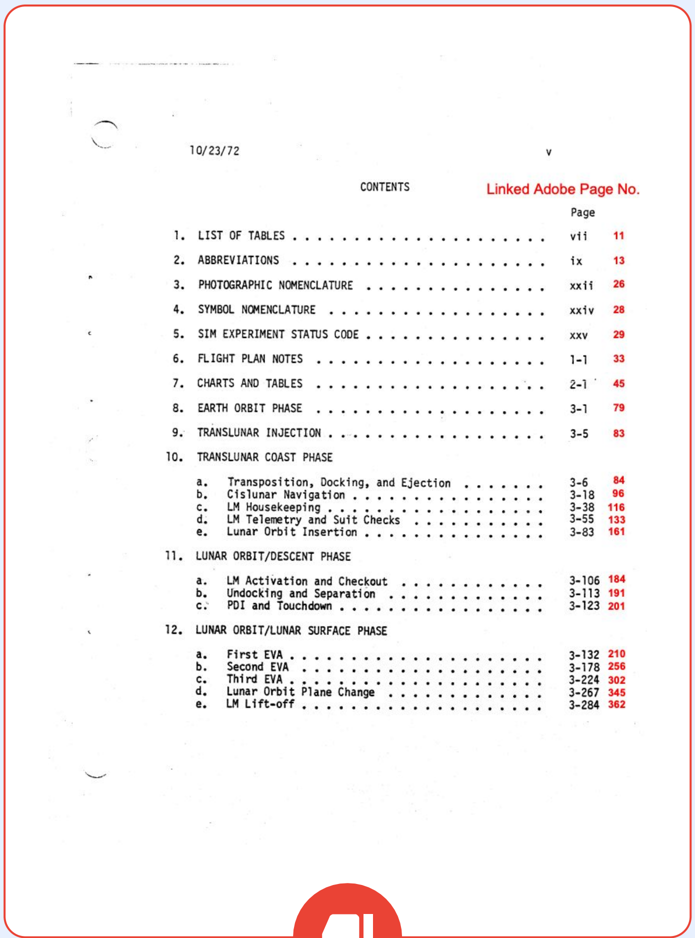
Indicate by check mark if the registrant is a well-known seasoned issuer, as defined in Rule 405 of the Securities Act. Yes ☒ No ☐

Indicate by check mark if the registrant is not required to file reports pursuant to Section 13 or Section 15(d) of the Act. Yes ☐ No ☒

Indicate by check mark whether the registrant (1) has filed all reports required to be filed by Section 13 or 15(d) of the Securities Exchange Act of 1934 during the preceding 12 months (or for such shorter period that the registrant was required to file such reports), and (2) has been subject to such filing requirements for the past 90 days. Yes ☒ No ☐

Indicate by check mark whether the registrant has submitted electronically every Interactive Data File required to be submitted pursuant to Rule 405 of Regulation S-T (§232.405 of this chapter) during the preceding 12 months (or for such shorter period that the registrant was required to submit such files). Yes ☒ No ☐

Indicate by check mark whether the registrant is a large accelerated filer, an accelerated filer, a non-accelerated filer, a smaller reporting company, or an emerging growth company. See the definitions of "large accelerated filer," "accelerated filer," "smaller reporting company," and "emerging growth company" in Rule 12b-2 of the Exchange Act.



10/23/72 v

CONTENTS

Linked Adobe Page No.

Google Cloud

MM.Documents.Tip #2

Split long documents

Gemini supports PDF files in the prompt. Each PDF can have **up to 1000 pages** with a maximum file size of 50MB. For larger documents, split them into multiple files.

Things to know

Limitations with Documents

Bounding Box Detection

The models aren't precise at locating text or objects in PDFs.
They might only return the approximated counts of objects.

Accuracy

The models might hallucinate when interpreting handwritten text in PDF documents.

These limitations will be updated periodically to reflect ongoing model improvements.

Summary

Multimodal Prompting with Gemini on Vertex AI

General Multimodal Prompting Strategies

1. Craft clear and concise instructions.
2. Add your media first for single-media prompts.
3. Add examples to the prompt.
4. Break down the task step-by-step.
5. Specify the output format.
6. Use context caching for repeated queries.
7. Use semantic cues in your prompts

Working with Images

1. Enumerate when prompt has multiple images.
2. Use a single image for optimal text detection.
3. You can detect objects in images with bounding boxes in 2D and 3D.
4. Detect 3D points and spatial relationships.
5. Guiding models' attention by adding hints.
6. Ask for detailed analysis for optimizing output.

Working with Videos

1. Specify timestamp format when localizing videos.
2. Ask Gemini to focus on visual content for well-known video clips.
3. Process long videos in segments for dense outputs.

Working with Audio

1. Ask Gemini to avoid summarizing for transcription.
2. Add examples for effective speaker diarization.

Working with Documents

1. Prefer digital PDFs over scanned PDFs.
2. Split long documents.

Key things to remember!

Every LLM is unique with its strengths and weaknesses!

Prompting **strategies evolve** (just like LLMs).

Get ready to **experiment!**

These techniques are **a great place to start.**

Vertex AI Gemini: Prompting Strategies

Last updated: Apr 25, 2025

Intended Audience	GenAI practitioners and prompt engineers using Generative AI on Vertex AI
Scope	The scope of this deck is to guide you through a typical prompt engineering workflow, demonstrating how to create effective prompts and leverage proven strategies when working with Vertex AI's Gemini 2.0 family of models. This content will be updated periodically to reflect ongoing improvements.
Additional Resources	Generative AI on Vertex AI: Prompt design and engineering GitHub Repo: Gen AI on Vertex AI GitHub Repo: Vertex AI Gemini Migrate to Gemini 2

Release Notes

04/10/2024	General prompting strategies with PaLM and Gemini 1.0 models
10/03/2024	Major updates with Gemini 1.5 including multimodal prompts, and other features e.g. system instruction, context caching
03/23/2025	Major updates with Gemini 2.0 • Multimodal Live API • **Native tool use**: Search as a tool and Code execution • New **modalities**: Image generation and speech generation • **Thinking mode**: Generate the "thinking process" for stronger reasoning capabilities. • The Gen AI **SDK** provides a unified interface to Gemini 2.0 and Gemini 1.5 through both the Gemini Developer API and the Gemini API on Vertex AI. The SDK is generally available in Python. Support for Go and TypeScript/JavaScript are in Preview, and Java support is in experimental.
ROADMAP	Updates to prompting with long context, function calling, grounding

Google Cloud

Stay up to date

[Vertex AI Studio]

[Vertex AI Prompting Strategies]