

EmbeddingGemma 2 — reformatted model-card text

Reformatted model-card text, not a research paper. Google DeepMind; CC BY 4.0. Layout changed; site navigation, media and code widgets omitted. Tables reformatted. Consult the original for exact presentation.

[Original Google model card](#) · CC BY 4.0

[Hugging Face](#) | [GitHub](#) | [Launch Blog](#) | [Documentation](#) | License : Apache 2.0 | Authors : Google DeepMind

EmbeddingGemma 2 is an open multimodal embedding model built by Google DeepMind which maps text (incl. code), images, video, and audio inputs—and combinations thereof—into a single, unified 768-dimensional vector space. The model has 740M total parameters, combining a 270M parameter text model with modular vision (170M) and audio (300M) encoders.

Designed to run on consumer hardware such as mobile devices and laptops, EmbeddingGemma 2 delivers low-latency semantic representations for on-device applications, like search, retrieval-augmented generation (RAG), classification, and clustering.

EmbeddingGemma 2 builds upon the architectural and capability advancements of Gemma 4, offering several core features:

- Native multimodality: Unifies 4 modalities (text, images, video, and audio) in a single shared 768-dimensional embedding space.
- Multilinguality and code: EmbeddingGemma 2 understands 100+ languages, and achieves a ~14% improvement on code tasks relative to its predecessor.
- Flexible footprint: Combines a 270M parameter text backbone (130M transformer + 140M embedder) with selectively loadable vision (170M) and audio (300M) encoders, allowing developers to load only the modalities required for their use case.
- Matryoshka Representation Learning (MRL): Native support for truncated embeddings across 128d, 256d, 512d, and 768d, enabling up to a 6x reduction in vector storage costs with minimal impact on quality.
- Context length: 8K token context window, capable of processing minutes of audio or video.
- Task-steered representations: Uses lightweight text instruction prefixes to optimize embeddings for different tasks (search, classification, clustering, semantic similarity, etc.).

Model Overview

Parameters	Total	740M
	Backbone	130M
	Embedder	140M
	Modality Encoders	Vision: 170M Audio: 300M
Architecture	Layers	24
	Model Dimension	512
	Hidden Dimension	2048
	Sliding Window	1024 tokens
	Vocabulary Size	262,144
	# Heads	4
	# KV-Heads (Local/Global)	2/1
	Local:Global	5:1
	Attention	GQA/MQA
	Activation	Gated FFN with GELU
	Pooling	Mean Pooling
	Projection Layer	512→768
Input/Output	Supported Modalities	Text, Images, Video, Audio
	Context Window	8,192 tokens
	Native Output Dimension	768
	MRL Truncation Dimensions	128, 256, 512

Benchmark Results

EmbeddingGemma 2 was evaluated across text, code, vision, visual document, video, and audio embedding benchmarks. All results reported below use the full-precision checkpoint.

Overall Evaluation Results (768d)

Modality	Benchmark	Metric	EmbeddingGemma 2	EmbeddingGemma 1
Text	Massive Text Embedding Benchmark (MTEB, multilingual, v2)	Mean(Task), Multiple	61.36	61.15
	Massive Text Embedding Benchmark (MTEB, code, v1)	Mean(Task), NDCG@10	78.68	68.76
Image	Massive Image Embedding Benchmark (MIEB, lite)	Mean(TaskType), Multiple	64.64	-
	Massive Multimodal Embedding Benchmark (MMEB v2 - Image)	Mean(Task), Hit@1	57.28	-
	Massive Multimodal Embedding Benchmark (MMEB v2 - VisDoc)	Mean(Task), NDCG@5	67.84	-
Video	Massive Multimodal Embedding Benchmark (MMEB v2 - Video)	Mean(Task), Hit@1	50.67	-
Audio	Massive Sound Embedding Benchmark (MSEB, Retrieval)	Mean(Task), MRR@10	69.54	-
	Massive Audio Embedding Benchmark (MAEB) Hugging Face	Mean(Task), Multiple	49.39	-

Evaluation Results with Vector Truncation

With MRL, EmbeddingGemma 2 representations can be truncated below the native 768d to 128d, 256d, and 512d representations and re-normalized. With this, model users can reduce storage requirements, with minimal quality impact down to 256d. 128d is best suited to text-only workloads.

Output Dimension	Compression Ratio	MTEB (multilingual, v2) Mean(Task)	MTEB (eng, v2) Mean(Task)	MTEB (code, v1) Mean(Task)	MIEB (lite) Mean(TaskType)	MMEB (v2) Overall	MSEB (Retrieval) Mean(Task)	MAEB Mean(Task)
768d (Full)	1:1	61.36	68.46	78.68	64.64	59.01	69.54	49.39
512d	1:1.5	61.17	68.41	77.24	64.32	58.38	69.18	49.21
256d	1:3	60.41	67.78	76.18	63.13	56.24	66.76	48.91
128d	1:6	57.89	65.68	71.41	59.06	45.65	56.71	46.92

Quick Start

Install the sentence-transformers library:

Generate text embeddings: `python from sentence_transformers import SentenceTransformer`

```
model = SentenceTransformer("google/embeddinggemma-2")
```

```
query = "What causes the northern lights?" document = "The northern lights are caused by charged particles from the sun."
```

```
query_emb = model.encode(query, prompt_name="SearchQuery") doc_emb = model.encode(document, prompt_name="Document") print(model.similarity(query_emb, doc_emb)) ``
```

Best Practices

For optimal embedding quality and runtime efficiency, follow these configurations and best practices:

1. Task Instruction Prefixes

EmbeddingGemma 2 is trained with short task instruction prefixes prepended to text inputs. Using the right prefix improves quality; omitting it still works but reduces precision. Prefixes apply to text only. Pass images, video, and audio without any prefix.

Documents with a real title should be formatted as title: {title} | text: {content} . Use title: none when no title is available.

Prefix Notation & Usage

We offer two types of task prefixes, depending on how embeddings are used in the task. There are two types of tasks:

- Asymmetric Tasks (e.g. retrieval): Use a query prefix for queries and a document prefix for corpus items.
- Symmetric Tasks (e.g. classification, similarity) : Apply the same task prefix to all inputs being compared.

Use Case	Task Type	Prompt Name	Query Task Instruction	Document Task Instruction (use none if no title)
Web / document search	Asymmetric	SearchQuery	task: search result query: {query}	title: {title} text: {content}
Question answering	Asymmetric	QuestionAnswering	task: question answering query: {question}	title: {title} text: {passage}
Fact checking	Asymmetric	FactChecking	task: fact checking query: {claim}	title: {title} text: {evidence}
Code search	Asymmetric	CodeRetrieval	task: code retrieval query: {query}	title: {title or filename} text: {code}
Text classification	Symmetric	Classification	task: classification query: {content}	N/A
Clustering	Symmetric	Clustering	task: clustering query: {content}	N/A
Measuring similarity	Symmetric	SentenceSimilarity	task: sentence similarity query: {content}	N/A

Please note: prompt_name="Document" applies title: none ; titled documents must still be formatted manually, like model.encode(f"title: {title} | text: {document}")

2. Selective Encoder Loading

The vision and audio encoders are independent components. To reduce memory consumption when deploying text-only or single-modality pipelines, disable unused modality encoders via SentenceTransformer's `config_kwargs` :

Please note: configuring EmbeddingGemma 2 to load with omission of some encoders differs among model libraries; please refer to the appropriate documentation for more information on this.

Active Modalities	config_kwargs	Effective Size
Text only	<code>{"vision_config": None, "audio_config": None}</code>	270M
Text and image	<code>{"audio_config": None}</code>	440M
Text and audio	<code>{"vision_config": None}</code>	570M
Full multimodal	<code>{}</code>	740M

3. Matryoshka Dimension Truncation

EmbeddingGemma 2 is trained with Matryoshka Representation Learning, so the 768-dimensional output vector can be shortened by keeping only its leading dimensions. The supported dimensions are 768, 512, 256, and 128. Shorter vectors reduce storage and speed up similarity search at some cost to quality.

At runtime, please adhere to the following guidelines:

- Re-normalize after truncating: slicing a unit-length vector does not preserve unit length. The shortened vector must be L2-normalized before it is used for cosine similarity. Skipping this step degrades ranking quality silently—it produces plausible-looking scores rather than an error.
- Queries and documents must share a dimension. A 768-dimensional query cannot be scored against a 128-dimensional corpus.
- To avoid truncating and normalizing yourself, you should pass in the `truncate_dim` and `normalize_embeddings` fields when calling `model.encode()` :

Model quality at each dimension is reported in the truncation table in the Benchmark Results section above. Quality is close to lossless down to 256 dimensions. 128 dimensions degrades multimodal quality substantially and should be validated against your own workload before adoption.

4. Numerical Precision

Run inference in `bfloat16` or `float32` . Do not use `float16` .

EmbeddingGemma 2's activation range exceeds the dynamic range of `float16` . In `float16` the model returns NaN or silently degraded embeddings rather than raising an error, so the failure is easy to miss.

`bfloat16` is safe, and carries the same 8-bit exponent as `float32` . This is the recommended default on hardware with native support, and halves memory use relative to `float32` . Use `float32` elsewhere, including on most CPUs.

In SentenceTransformers, you can set this via `model_kwargs` , and determine it programmatically via `is_bf16_supported` :

5. Multimodal Input

Interleaving

A single input may mix text with images, video, and audio, using a single shared 8,192-token context.

The position of each media item within the sequence is marked in the text using placeholder tokens from the model's vocabulary:

- `<|image|>` marks the position of image input
- `<|video|>` marks the position of video input
- `<|audio|>` marks the position of audio input

For example, a product listing indexed for search might be encoded as:

Each placeholder in text is filled from the corresponding key, in order. The call returns a single embedding that represents the text, images and video together. This embedding can be compared directly against any other EmbeddingGemma 2 embedding; for example, a text-only query such as "waterproof shoes for trail running".

Context Limits

All modalities share a single 8,192-token context window. Each modality consumes that budget at a fixed rate:

Modality	Token Cost	Max Input
Text	1 token per subword	8,192 tokens
Image	280 tokens per image (default)	~29 images
Video	140 tokens per frame (default)	~58 frames
Audio	25 tokens per second	~327 seconds

Note max input for images and video can be as high as ~114 images or frames when using a lower vision token budget (described below).

The maximums above assume a single modality with no accompanying text. Interleaved inputs draw from the same budget, so mixing modalities reduces the amount of each that fits. Note also max input for images and video can be as high as ~114 images or frames when using a lower vision token budget (described below).

Vision Token Budget

Model users can forgo the default sequence length for input images / video frames to represent images with "soft token" amounts ranging from 70 to 1120. Increasing the vision budget for input images trades latency and token count for quality.

In general, higher input sequence lengths capture more information. Scaling up input sequence length means more expressive input representations; so, increasing input sequence length will improve fine-grained visual understanding, uplifting embedding quality and performance in downstream use cases.

Input Sampling Defaults

Audio and video inputs have the following default sampling rates:

- By default, video is processed as sampled frames through the vision encoder at 1 frame per second; FPS sampling rate is configurable.
- Audio should be supplied as mono at 16 kHz.

Model Data

Our pre-training dataset is a large-scale, diverse collection of data encompassing a wide range of domains and modalities, which includes web documents, code, images, video, and audio, with a cutoff date of January 2025. These include:

- Web Documents: A diverse collection of web text ensures the model is exposed to a broad range of linguistic styles, topics, and vocabulary. The training dataset includes content in over 140 languages.
- Code: Exposing the model to code helps it to learn the syntax and patterns of programming languages, which improves its ability to understand code-related semantics.
- Images: A wide range of images enables the model to perform image analysis and visual data extraction tasks.
- Video: Video sequences and clips covering diverse visual scenes, human actions, and temporal dynamics.
- Audio: Speech recordings across multiple languages, environmental sounds, and acoustic events.
- Cross-modality Samples: Paired text, image, video, and audio data to align representations across different modalities.

Data Processing

Several data cleaning and filtering methods were applied to the training data:

- CSAM Filtering : Rigorous CSAM (Child Sexual Abuse Material) filtering was applied at multiple stages in the data preparation process to ensure the exclusion of harmful and illegal content.
- Sensitive Data Filtering : As part of making Gemma pre-trained models safe and reliable, automated techniques were used to filter out certain personal information and other sensitive data from training sets.
- Additional Methods : Filtering based on content quality and safety in line with our policies .

Ethics & Safety

EmbeddingGemma 2 is a pre-trained embedding model. Unlike generative models, it does not undergo post-training alignment, safety tuning, or output-level moderation. Safety mitigations during development were focused on pre-training data filtering to reduce exposure to harmful content and severe biases in the learned embedding space, in alignment with Google's AI Principles .

Since embedding models produce representations rather than user-facing text, safety risks manifest downstream in how those representations are used.

Developers and deployers are responsible for evaluating and implementing application-level safeguards, like retrieval filtering and fairness testing, appropriate to their specific production use case.

Deployments must adhere to the Gemma Prohibited Use Policy .

Usage and Limitations

EmbeddingGemma 2 has certain limitations that users should be aware of.

Intended Usage

EmbeddingGemma 2 generates embeddings from input content, which can be used for a number of downstream applications.

The following list of potential uses is not comprehensive. The purpose of this list is to provide contextual information about the possible use-cases that the model creators considered as part of model training and development.

- Retrieval: Embeddings for semantic search across text, code, images, video, and audio, such as document search, RAG over enterprise knowledge bases, code search from natural language queries, or spoken-query search over audio archives.
- Classification: Embeddings to classify inputs according to preset labels, like sentiment analysis, content moderation, audio event detection, or image categorization.
- Clustering: Embeddings to group inputs based on semantic similarity, like organizing document collections, clustering customer feedback by theme, or discovering near-duplicate content across modalities.
- Semantic Similarity: Embeddings to measure pairwise similarity between inputs, such as recommendation systems, duplicate detection, paraphrase identification, or cross-lingual content alignment.
- Fact Verification: Embeddings for retrieving evidence documents given a claim, like automated fact-checking systems or source attribution pipelines.

Limitations

- Training Data: The quality and diversity of the training data significantly influence the model's capabilities. Biases or gaps in the training data can lead to limitations in the model's responses. The scope of the training dataset determines the subject areas the model can handle effectively. For example, while EmbeddingGemma 2 supports 100+ languages, the model may not exhibit equal performance across languages.
- Context and Task Complexity: Models perform well on tasks that can be framed with clear prompts and instructions. Open-ended or highly complex tasks might be challenging. A model's performance can be influenced by the amount of context provided (longer context generally leads to better outputs, up to a certain point).
- Language Ambiguity and Nuance: Natural language is inherently complex. Models might struggle to grasp subtle nuances, sarcasm, or figurative language.

- Task Instruction Usage: For text tasks, omitting the recommended task prefix may lead to sub-optimal embedding quality.

Ethical Considerations and Risks

In creating an open embedding model, we have carefully considered the following:

- Bias and Fairness Models trained on large-scale, real-world text and image data can reflect socio-cultural biases embedded in the training material. Training data used for EmbeddingGemma 2 underwent safety filtering to mitigate the risk of these biases.
- Misinformation and Misuse Embedding representations can be misused to retrieve, classify, or otherwise organize embedded content in false, misleading or harmful ways. Guidelines are provided for responsible use with the model, see the Responsible Generative AI Toolkit .
- Transparency and Accountability This model card summarizes details on the model's architecture, capabilities, limitations, and evaluation processes. A responsibly developed open model offers the opportunity to share innovation by making Vision-Language Model (VLM) technology accessible to developers and researchers across the AI ecosystem.

Risks Identified and Mitigations

- Misuse for malicious purposes: Technical limitations and developer and end-user education can help mitigate against malicious applications of embedding models. Educational resources and reporting mechanisms for users to flag misuse are provided.
- Privacy violations: Models were trained on data filtered for removal of certain personal information and other sensitive data. Developers are encouraged to adhere to privacy regulations with privacy-preserving techniques.
- Perpetuation of biases: It's encouraged to perform continuous monitoring (using evaluation metrics, human review) and the exploration of de-biasing techniques during model training, fine-tuning, and other use cases.

Benefits

EmbeddingGemma 2 is among the strongest multimodal embedding models under 1B parameters. We're excited to see how developers will use and adapt this model for on-device or edge AI applications.

The model is designed from the ground up for responsible AI development, like other Gemma-family models.