# **An Architectural Blueprint for a Billion-Scale Multimodal Video Search Engine**

## **Section 1: The Heart of the System: Selecting the Optimal Multimodal Embedding Model**

The foundational component of any large-scale semantic search system is the embedding model. This model is responsible for translating raw, unstructured data—in this case, video frames—into dense numerical vectors, or embeddings. The quality and characteristics of these embeddings directly dictate the performance, accuracy, and capabilities of the entire search pipeline. This section provides a detailed analysis of the current state-of-the-art in multimodal embedding models, culminating in a definitive recommendation tailored to the specific requirements of this project. A critical examination of the relationship between semantic search embeddings and copy-detection embeddings is also presented, resolving a key architectural question.

### **1.1 The Evolution from CLIP to Unified Architectures: A Pragmatic Explanation**

To understand the current landscape of multimodal models, it is essential to first understand the technology that defined the field: OpenAI's CLIP (Contrastive Language-Image Pre-training). The core innovation of CLIP was its method for learning a shared representational space for both images and text from vast quantities of image-caption pairs scraped from the web.

Architecturally, CLIP employs a dual-encoder pattern. It uses two distinct neural networks: a Vision Transformer (ViT) to process images and a separate text Transformer to process natural language descriptions. During training, the model is given a batch of (image, text) pairs. Its objective, known as a contrastive loss, is to maximize the cosine similarity of the embeddings for correct pairs (e.g., the vector for an image of a cat and the vector for the text "a photo of a cat") while minimizing the similarity for all incorrect pairs in the batch. This approach proved remarkably effective, enabling powerful zero-shot capabilities where the model could classify images into categories it had never explicitly been trained on, simply by comparing the image embedding to the text embedding of a class name. This dual-encoder, contrastive pre-training paradigm became the foundation for many subsequent multimodal models, including Google's ALIGN.

However, this pioneering dual-encoder architecture has a significant and well-documented limitation known as the "modality gap". Because the image and text data are processed through separate, architecturally distinct encoders, they learn different data transformations. Even though they are trained to map to the same vector space, the embeddings for each modality tend to form their own distinct clusters within that space. One region of the space becomes the "image cluster," and another becomes the "text cluster." Consequently, an embedding for a text query might be semantically closer to an unrelated text embedding than it is to the embedding of a highly relevant image. This phenomenon degrades the performance of cross-modal retrieval tasks, which are central to this project's requirements (text-to-image and image-to-text search).

To address this fundamental flaw, a new generation of state-of-the-art models has emerged, moving away from the dual-encoder pattern toward a unified architecture. Models like Voyage AI's voyage-multimodal-3 utilize a single, unified vision-language transformer encoder. In this architecture, both text and image data (represented as tokens) are fed into the

*same* transformer, allowing the model to learn a truly unified representation where the contextual relationship between visual and textual information is deeply intertwined. This unified approach effectively eliminates the modality gap, ensuring that a concept is represented by a similar vector regardless of whether its source was an image or a piece of text. For a system that must robustly handle searches across different modalities, this architectural evolution represents a critical leap forward in performance and reliability.

### **1.2 Head-to-Head Analysis of State-of-the-Art Models**

The selection of an embedding model involves a trade-off between performance, cost, flexibility, and ease of deployment. The following analysis compares the leading proprietary and open-source models relevant to this project's goals.

* **OpenAI CLIP:** As the foundational model, CLIP remains a strong open-source baseline. It excels at zero-shot generalization and is supported by an active community. Its variants, like clip-vit-large-patch14-336, are widely used in research and are available under a permissive MIT license, making them ideal for local testing and validation without incurring API costs. However, its dual-encoder architecture suffers from the modality gap, and benchmark comparisons show it is consistently outperformed by newer models.
* **Google ALIGN:** Google's ALIGN (A Large-scale Image and Noisy-text embedding) follows the same dual-encoder principle as CLIP but was trained on a significantly larger and noisier dataset of over one billion image-text pairs from the web. This allows it to handle real-world, imperfect data effectively and demonstrates strong performance on image-text retrieval benchmarks. While powerful, it shares the same architectural limitations as CLIP.
* **Cohere Embed v3:** This is a state-of-the-art commercial multimodal model designed for enterprise applications. It is engineered to place image and text embeddings into a tightly integrated vector space, making it highly effective for use cases like searching e-commerce product catalogs or libraries of design files. A key advantage for massive-scale deployment is its support for embedding quantization. It can produce
* int8 and binary embeddings, which dramatically reduce memory and storage requirements by 4x and 32x, respectively, while also speeding up search performance significantly. This feature is extremely valuable when dealing with billions of vectors. The model is available via API and through cloud providers like AWS and Azure.
* **Google Vertex AI multimodalembedding@001:** This is Google Cloud's managed multimodal embedding service. It is a highly capable model that generates large, 1408-dimension vectors for text, images, and video segments, all within the same semantic space. Its primary advantage is its seamless integration with the GCP ecosystem, particularly Vertex AI Vector Search. This can simplify the development pipeline, especially given the available GCP credits. The API allows for embedding entire video files by specifying a GCS URI, and it can return embeddings for specific intervals within the video. However, this convenience comes with the significant risks of vendor lock-in and potentially unpredictable, high costs at production scale. Pricing is complex, based on the type and amount of data processed (e.g., per 1,000 characters, per image, per second of video), in addition to the costs of hosting the search index itself.
* **Voyage AI voyage-multimodal-3:** This model currently represents the pinnacle of multimodal embedding performance. Its unified vision-language transformer architecture fundamentally solves the modality gap that plagues CLIP-like models. It uniquely supports interleaved text and image inputs, allowing it to vectorize complex documents like PDFs or screenshots with exceptional accuracy. Benchmarks show it dramatically outperforms competitors; for example, on table/figure retrieval tasks, it improves upon OpenAI CLIP large and Cohere multimodal v3 by over 40%. It also boasts a very large 32,000-token context length, compared to 512 for Cohere and 8,192 for OpenAI models. This combination of superior architecture and benchmarked performance makes it the leading candidate for accuracy-critical applications.
* **Open-Source Alternatives (MetaCLIP, OpenCLIP):** Beyond the original CLIP, the open-source community has produced powerful alternatives. MetaCLIP, for example, is a model trained by Meta AI on a massive, curated dataset of 400 million image-text pairs, achieving performance competitive with the original CLIP models. OpenCLIP is a community-driven effort to replicate and improve upon CLIP, providing a variety of pre-trained model weights under permissive licenses. These models are invaluable for local development, pipeline testing, and deployments where cost control and infrastructure ownership are paramount.

The following table synthesizes these findings into a direct comparison.

**Table 1: Comparative Analysis of State-of-the-Art Multimodal Embedding Models**

| Model Name | Architecture Type | Key Features | Embedding Dimension | Accuracy Insights (from Benchmarks) | Licensing & Deployment | Inference Speed / Cost |
| --- | --- | --- | --- | --- | --- | --- |
| **Voyage voyage-multimodal-3** | Unified Vision-Language Transformer | Interleaved text/image input, 32k context length, no modality gap | 1024 | SOTA on table/figure/screenshot retrieval; outperforms CLIP & Cohere by >25% on document retrieval | Commercial API | 75ms for a query up to 200 tokens; 57M tokens/hr on ml.g6.xlarge |
| **Cohere embed-multimodal-v3.0** | Dual-Encoder Contrastive (Advanced) | Quantization (int8, binary), optimized for e-commerce, design files, multilingual (100+ languages) | 1024 | Strong on domain-specific benchmarks (e-commerce, charts); int8 maintains 99.9% quality, binary 90-98% | Commercial API (Cohere, AWS, Azure) | Optimized for low latency; binary search up to 40x faster |
| **Google multimodalembedding@001** | Vision-Language Model (Proprietary) | Native video segment embedding, deep GCP integration, OCR capabilities | 1408 | High-quality general-purpose model; performance depends on ranking, not absolute score | GCP Managed Service | Priced per 1k chars, per image, or per second of video |
| **OpenAI CLIP ViT-L/14** | Dual-Encoder Contrastive | Foundational zero-shot model, strong generalization, large open-source community | 768 | Strong baseline, but outperformed by newer models; suffers from modality gap and spurious correlations | Open Source (MIT License) | Self-hosted; cost depends on hardware. |
| **MetaCLIP** | Dual-Encoder Contrastive | Trained on massive curated dataset (400M pairs), strong open-source performance | 512 / 1024 | Performance is competitive with original CLIP models. | Open Source (CC BY-NC 4.0) | Self-hosted; cost depends on hardware. |

### **1.3 The "Deep Vision" Dilemma: A Two-Embedding Strategy**

A central question in the project brief is whether the embeddings generated for semantic search can be reused for the "Deep Vision" copy detection stage, which is based on a Self-Supervised Copy Detection (SSCD) model. The answer is an unequivocal **no**. The two tasks are not just different; their objectives are fundamentally opposed, necessitating a two-embedding strategy.

The goal of **semantic search** is to abstract away from low-level features to capture high-level meaning. An embedding for an image of a "golden retriever playing fetch" should be located very close in the vector space to an embedding for the text "a yellow dog chasing a ball". This requires the model to create dense clusters of semantically related concepts, where different instances of the same concept are grouped together.

The goal of **SSCD**, or image copy detection, is the precise opposite. It must identify *perceptual duplicates*, not semantic ones. An SSCD model must recognize that a heavily compressed, cropped, and color-shifted version of an image is a

*copy*, while simultaneously recognizing that two different photographs of the Eiffel Tower, taken seconds apart from slightly different angles, are *not* copies.

To achieve this, SSCD models employ specific architectural choices and training objectives that are actively detrimental to semantic clustering. The SSCD paper highlights several key modifications to a standard self-supervised learning pipeline :

1. **Differential Entropy Regularization:** This is the most critical factor. SSCD adds an entropy regularization term to its loss function. The explicit purpose of this term is to force the embedding distribution to be as uniform as possible, effectively pushing all non-identical image vectors apart from each other. This prevents the "collapse" of embeddings into a few points but also destroys the semantic clusters needed for search. The authors note that this regularization term is "not helpful for classification" because it discourages the very clustering that classification relies on.
2. **Specialized Augmentations:** The model is trained with augmentations like MixUp and CutMix, which combine parts of different images. This trains the model to detect partial copies, forcing it to focus on exact feature correspondence rather than holistic scene understanding.
3. **Perceptual Focus:** The entire objective is to create a descriptor that is invariant to transformations like compression and color shifts but highly sensitive to changes in the actual image content. This is a model of *perceptual similarity*, not semantic similarity.

Therefore, the system architecture must be designed to generate and store **two distinct embeddings** for each video frame:

* **semantic\_embedding**: A vector generated by a state-of-the-art multimodal model (like voyage-multimodal-3). This vector is optimized for semantic clustering and will be used for the initial, broad search to find conceptually related content.
* **sscd\_embedding**: A vector generated by the project's existing pre-trained SSCD model. This vector is optimized for perceptual matching. It will be retrieved by its ID *after* the initial semantic search and used only for the final, high-precision copy detection comparison.

Attempting to use one embedding for both tasks would result in a system that fails at both: the semantic search would be unable to find related concepts, and the copy detection would be unable to distinguish between different scenes.

### **1.4 Final Recommendation and Rationale**

Based on the comprehensive analysis of model architectures, benchmark performance, and feature sets, the following recommendations are made for the semantic embedding model.

* **Primary Recommendation: Voyage AI voyage-multimodal-3**
  + **Rationale:** This model offers the highest accuracy and robustness. Its unified transformer architecture directly solves the modality gap, a critical flaw in older models for cross-modal search. Its documented superior performance across a range of challenging retrieval tasks provides high confidence that it will deliver the most accurate results for text-to-image, image-to-image, and video-to-video search. While it is a commercial API, its state-of-the-art performance justifies the cost for a production system where accuracy is paramount.
* **Secondary (GCP-Centric) Recommendation: Google Vertex AI multimodalembedding@001**
  + **Rationale:** If the primary project driver is to maximize the use of the $300k in GCP credits, this is a powerful and viable alternative. Its native integration with the GCP ecosystem, especially Vertex AI Vector Search, provides a streamlined path to deployment. Its ability to handle video as a native input type is also a significant advantage. However, this path comes with a higher risk of vendor lock-in and requires careful cost modeling to ensure long-term financial viability at a scale of billions of vectors.
* **Testing & Validation Recommendation: OpenCLIP or MetaCLIP**
  + **Rationale:** For all local development, prototyping, and pipeline testing, a high-quality open-source model is the most pragmatic choice. Models like OpenCLIP and MetaCLIP are available under permissive licenses (MIT, CC BY-NC 4.0) and can be self-hosted at no API cost. This allows for the complete end-to-end development and testing of the ingestion and search pipeline before switching to the production-grade commercial API for final deployment and performance tuning. This approach de-risks the project and contains development costs.

## **Section 2: The Foundation: Choosing a Vector Database for Massive Scale**

Having selected the embedding strategy, the next critical decision is the choice of infrastructure for storing, indexing, and querying the resulting billions of vectors. The stated requirement for an open-source, cloud-agnostic solution that can scale to billions of items, combined with a negative experience regarding the cost of managed services like OpenSearch, clearly points away from proprietary platforms or simple database extensions. A purpose-built, distributed, native vector database is required to meet the performance and scalability demands of this project.

### **2.1 The Landscape of Vector Databases: Native vs. Add-on**

Vector search solutions can be broadly categorized into two classes, and understanding the distinction is crucial for making a sound architectural choice.

1. **Add-on Solutions:** This category includes extensions to traditional databases that add vector search capabilities. The most popular example is pgvector, an extension for PostgreSQL. Other examples include k-NN plugins for search engines like Elasticsearch and OpenSearch. While these solutions are convenient for smaller-scale projects or for teams heavily invested in a particular database ecosystem, they are not architected for the extreme performance and scale this project demands. They often treat vectors as just another data type, without the specialized storage formats, indexing algorithms, and distributed query execution engines necessary for billion-scale workloads.
2. **Native Vector Databases:** These are systems designed from the ground up with the sole purpose of managing and searching massive collections of high-dimensional vectors. Their architectures are fundamentally different, built around concepts like Approximate Nearest Neighbor (ANN) search algorithms, custom data sharding strategies, and disaggregated compute and storage. For a project targeting "billions of videos," a native vector database is not just recommended; it is a prerequisite for success.

### **2.2 The Open-Source Titans: Milvus vs. Weaviate vs. Qdrant**

Within the native open-source landscape, three databases have emerged as the leading contenders: Milvus, Weaviate, and Qdrant. Each offers a powerful, scalable solution, but they differ in their architectural philosophies and core strengths.

* **Milvus:**
  + **Scalability Architecture:** Milvus is widely regarded as the most mature and proven solution for massive-scale vector search. Its key differentiator is its highly decoupled, cloud-native, microservices-based architecture. It disaggregates storage and computing, and further breaks down compute into four distinct layers: an access layer (load balancing), a coordinator service (cluster management), worker nodes (for query, data ingestion, and indexing), and a storage layer (for metadata and logs). This design allows each component to be scaled independently. For example, during a heavy data ingestion phase, the data nodes can be scaled up, while during a query-heavy phase, the query nodes can be scaled. This granular control is essential for optimizing performance and cost in a dynamic, billion-scale environment. Milvus is designed to be deployed on Kubernetes, making it inherently cloud-agnostic. Recent versions have even replaced external dependencies like Kafka with a custom, lightweight Write-Ahead Log (WAL) system called Woodpecker, further reducing operational complexity and resource overhead at scale.
  + **Indexing & Filtering:** Milvus is highly flexible, supporting a variety of ANN index types, including HNSW, IVF\_FLAT, and the disk-based DiskANN, allowing for fine-tuned performance across different hardware profiles. Its metadata filtering is powerful. The standard approach is "pre-filtering," where Milvus first uses indexes on scalar fields (e.g.,
  + platform) to identify matching records, creates a bitset of allowed IDs, and then performs the ANN search only on that subset. For more complex scenarios, it also supports advanced strategies like iterative filtering and metadata-aware subgraphs to maintain performance even with highly selective filters.
* **Weaviate:**
  + **Scalability Architecture:** Weaviate is also a cloud-native, open-source vector database designed for Kubernetes deployment. Its architecture is highly modular, with a standout feature being its ability to integrate machine learning models directly for on-the-fly vectorization. While this project will provide its own embeddings, this modularity speaks to the system's flexibility. Weaviate's core data model is object-oriented and centers around a knowledge graph concept, allowing it to represent relationships between data objects in addition to vector similarity. This makes it exceptionally powerful for complex semantic applications but may introduce unnecessary complexity for the more straightforward search-and-filter use case of this project. It exposes its functionality primarily through a GraphQL API.
  + **Indexing & Filtering:** Weaviate uses its own custom implementation of the HNSW algorithm, which notably offers full CRUD (Create, Read, Update, Delete) support. It supports robust filtered search, enabling hybrid queries that combine traditional keyword-based filters with vector similarity search.
* **Qdrant:**
  + **Scalability Architecture:** Qdrant's primary calling card is performance, driven by its implementation in Rust, a language known for speed and memory safety. It supports distributed deployment and horizontal scaling, and its public roadmap includes "Project Nebula," a plan for a highly scalable sharded HNSW implementation, signaling a strong commitment to future-proofing its architecture for massive scale.
  + **Indexing & Filtering:** Qdrant's most significant architectural advantage is its advanced and highly efficient payload filtering mechanism. Unlike systems that might apply filters after retrieving an initial set of vector candidates, Qdrant builds separate, dedicated indexes for vector payloads (metadata). This allows it to perform extremely fast filtering on metadata
  + *before* initiating the vector search, even with complex filtering logic. For applications where queries are consistently and heavily constrained by metadata, this design can lead to superior performance.

The following table provides a direct technical comparison of these three leading open-source databases.

**Table 2: Comparative Analysis of Open-Source Vector Databases**

| Database | Scalability Architecture | Indexing (HNSW) | Metadata Filtering | Deployment & Cloud-Agnosticism | Ideal Use Case |
| --- | --- | --- | --- | --- | --- |
| **Milvus** | Disaggregated compute/storage, 4-layer microservices architecture designed for independent scaling | Supports multiple indexes (HNSW, DiskANN, etc.). Highly tunable M, efConstruction | Advanced pre-filtering, iterative filtering, and metadata-aware subgraphs for efficiency at all filter ratios | Kubernetes-native via Helm charts; offers Distributed, Standalone, and Lite modes | Massive-scale (1B+) deployments requiring granular, independent scaling of components. |
| **Qdrant** | Rust-based, distributed deployment with sharding. Built for performance and memory safety | Filtrable HNSW; standout feature is modular payload indexing for ultra-fast filtering | Optimized for payload filtering with separate, dedicated payload indexes, providing a significant performance advantage | Kubernetes support via Helm charts; available as a managed cloud service | High-performance search with frequent, complex, and rich metadata filtering requirements. |
| **Weaviate** | Cloud-native, modular, knowledge-graph-oriented architecture with built-in vectorization modules | Custom HNSW implementation with full CRUD support | Supports robust filtered search, often combined with keyword search for hybrid retrieval scenarios | Kubernetes support via Helm charts; available as a managed cloud service | Semantic search applications with integrated vectorization needs and complex data relationships (knowledge graphs). |

### **2.3 The GCP Alternative: Analyzing Vertex AI Vector Search**

Given the availability of significant GCP credits, it is prudent to analyze Google's native offering, Vertex AI Vector Search.

* **Underlying Technology:** Vertex AI Vector Search is a fully managed service. Crucially, it does not use HNSW. Instead, it is built upon Google's proprietary **ScaNN (Scalable Nearest Neighbors)** algorithm. ScaNN is a state-of-the-art ANN algorithm developed by Google Research that often employs techniques like vector quantization and anisotropic scoring to achieve high performance at massive scale. It is the same core technology that powers similarity search across many of Google's own products.
* **Features & Integration:** As a first-party GCP service, its integration with the Vertex AI ecosystem is its greatest strength. It connects seamlessly with the multimodalembedding API for embedding generation and with other services like BigQuery and Cloud Storage for data management. It supports hybrid search, allowing queries to combine dense vectors for semantic meaning with sparse vectors for keyword matching. The deployment process is managed through the GCP console or APIs: one creates an index from embedding files in Cloud Storage, creates an index endpoint, and then deploys the index to that endpoint for serving.
* **Cost and Lock-in Analysis:** The GCP credits make the initial adoption cost-free, but a long-term analysis is critical. The pricing model for Vertex AI Vector Search has two main components: a one-time cost for building or updating an index (priced per GiB of data, e.g., $3/GiB) and a continuous cost for serving the index (priced per node, per hour). A single, small
* e2-standard-16 serving node can cost over $500 per month. For a database of billions of high-dimensional vectors, the data size would be in the terabytes, leading to substantial index building and storage costs, and requiring a large cluster of serving nodes. These costs could quickly become prohibitive and may exceed the expense of running a self-managed open-source cluster on equivalent GKE nodes. Most importantly, building the core of the application on this service creates a hard dependency on a proprietary GCP product, which is in direct conflict with the stated goal of being cloud-agnostic.

### **2.4 Final Recommendation and Rationale**

The choice of database is a long-term commitment that will define the scalability, cost, and flexibility of the entire system.

* **Primary Recommendation: Milvus**
  + **Rationale:** For a system explicitly designed to scale to "billions" of vectors, Milvus's distributed, microservices-based architecture is the most robust and proven choice among the open-source options. Its fundamental design principle of disaggregating compute and storage provides the necessary flexibility to manage performance and cost at extreme scale, a feature unmatched by more monolithic designs. Its widespread adoption by major enterprises (like Salesforce, PayPal, and eBay) provides strong evidence of its production-readiness and stability. As a Kubernetes-native application, it is perfectly suited for a cloud-agnostic deployment on GKE, allowing full utilization of the available GCP compute credits without vendor lock-in.
* **Strong Contender: Qdrant**
  + **Rationale:** Qdrant is an exceptionally strong alternative, particularly if the majority of search queries will involve complex metadata filters. Its architecture, which prioritizes fast payload filtering through dedicated indexes, could offer a decisive performance edge in such scenarios. The choice between Milvus and Qdrant is a choice between Milvus's superior architectural flexibility for scaling heterogeneous workloads and Qdrant's specialized optimization for filter-heavy queries. A practical next step would be to conduct a small-scale benchmark using representative data and query patterns to compare the two directly.
* **Final Guidance:** The recommendation is to begin development with **Milvus**. Its architectural maturity for extreme scale, proven track record in production, and unparalleled flexibility make it the safest and most scalable foundation for meeting the ambitious goals of this project.

## **Section 3: The Blueprint: Storing and Indexing Billions of Video Frames**

This section translates the high-level architectural decisions into a practical implementation plan. It provides a concrete data model, a guide to configuring the vector index for optimal performance, and a strategy for formulating effective search queries. The recommendations will use **Milvus** as the reference database, in line with the conclusion of the previous section.

### **3.1 Data Modeling for Multimodal Video Search**

A well-designed data schema is the foundation of a high-performance database. While Milvus supports dynamic schemas, defining an explicit schema for a production system is a best practice that ensures data integrity and clarity. The core entity in this system is a single video frame. Each document, or record, in the Milvus collection will represent one frame.

The proposed collection schema is designed to store all necessary information for both the initial semantic search and the subsequent "Deep Vision" analysis, while being optimized for efficient filtering and retrieval.

**Proposed Milvus Collection Schema:**

* frame\_id (Type: INT64, is\_primary=True, auto\_id=False): This will serve as the unique primary key for each frame record. It is recommended to generate these IDs externally to maintain a clear link to the source data.
* video\_code (Type: VARCHAR, max\_length=64, index=True): This field stores the unique identifier for the parent video (e.g., jkknas32da). Creating a scalar index on this field is **critical**. It will allow for the extremely fast retrieval of all frames belonging to a specific video, which is necessary for the "Deep Vision" stage.
* platform (Type: VARCHAR, max\_length=32, index=True): Stores the source platform (e.g., 'instagram', 'tiktok', 'youtube'). This field should also be indexed to enable efficient filtering by platform.
* frame\_number (Type: INT32): The sequential number of the frame within its video (e.g., 1, 2, 3...). This can be useful for ordering or for queries that target specific parts of a video.
* s3\_uri (Type: VARCHAR, max\_length=1024): The full URI pointing to the original frame image file in the S3 bucket (e.g., s3://oriane-frames/instagram/jkknas32da/1.png). This allows for retrieval of the original high-resolution image if needed.
* **semantic\_embedding** (Type: FLOAT\_VECTOR, dim=1024): This field will store the 1024-dimensional vector generated by the recommended Voyage AI model. This is the vector that will be used for all semantic search operations. The primary HNSW index will be built on this field.
* **sscd\_embedding** (Type: FLOAT\_VECTOR, dim=...): This field will store the vector generated by the project's custom SSCD model. The dimension should match the output of that model. Critically, **this field will not have an HNSW index built on it**. It is purely for storage and will be retrieved via frame\_id or video\_code lookups after the initial semantic search is complete. This saves significant memory and indexing resources.

### **3.2 Mastering HNSW: A Practical Guide to Indexing**

The Hierarchical Navigable Small World (HNSW) algorithm is the de facto standard for high-performance Approximate Nearest Neighbor (ANN) search in memory. It constructs a multi-layered graph structure that allows for extremely fast navigation through the vector space, providing an excellent trade-off between search speed and accuracy (recall). The HNSW index will be created exclusively on the

semantic\_embedding field.

Understanding and tuning HNSW involves three key parameters in Milvus :

* M: This parameter defines the maximum number of bidirectional links (edges) each node can have on a single layer of the graph. A higher value of M creates a denser, more connected graph. This generally leads to higher search accuracy because there are more potential paths to the true nearest neighbors. However, it comes at the cost of increased memory consumption and longer index-building times. For large-scale datasets, a value between 16 and 48 is a common and effective range.
* efConstruction: This parameter controls the size of the dynamic list of candidate neighbors that are considered when inserting a new node into the graph. A larger efConstruction value allows the algorithm to explore more potential connections, resulting in a higher-quality, more optimal graph structure. This directly translates to better search recall. The trade-off is a significantly longer index build time. Since index building is an offline process, it is generally advisable to use a generous value here (e.g., 200-500) to create the best possible index.
* ef (or search\_param['ef']): This is a **query-time** parameter that determines the size of the candidate list during a search. It is the most important parameter for tuning the live performance of the system. A higher ef value forces the search algorithm to explore more nodes and traverse more edges in the graph, which increases the probability of finding the true nearest neighbors (higher recall) but also increases query latency. A lower ef value results in a faster, less exhaustive search, which may slightly reduce recall. This parameter allows for a dynamic trade-off between speed and accuracy on a per-query basis.

**Practical Indexing Strategy:**

1. **Build a High-Quality Index Offline:** When creating the index on the semantic\_embedding field, prioritize quality. Use robust parameters like M=32 and efConstruction=400. The computational cost of building the index is paid once per data batch, so investing in a high-quality graph structure will pay dividends in search performance and accuracy later.
2. **Tune ef Dynamically at Query Time:** Expose the ef search parameter in the application's query logic. For standard user-facing searches where low latency is critical, a lower ef (e.g., 64 or 128) might be appropriate. For internal, high-recall analytical tasks, a much higher ef (e.g., 512 or more) could be used. This provides the flexibility to meet different performance requirements without rebuilding the index.

### **3.3 The Power of Metadata Filtering**

A pure vector search across a corpus of billions of frames would be computationally expensive and often return irrelevant results. For instance, a user might search for an image but only be interested in videos from TikTok. Metadata filtering is the mechanism that makes such queries efficient.

**Milvus Filtering Strategy:**

1. **Index Scalar Fields:** As defined in the schema, ensure that all fields intended for filtering (platform, video\_code) have a scalar index created on them. Milvus uses these indexes to rapidly identify matching records without having to scan the entire dataset.
2. **Utilize Pre-filtering in Queries:** Every search request sent to Milvus should include a boolean expression in the filter argument of the search() call. This expression can combine multiple conditions using standard logical operators.
   * Example: filter="platform == 'tiktok' and frame\_number > 100"
   * Example: filter="platform in ['instagram', 'youtube']"
3. **Mechanism:** When Milvus receives such a query, it executes a two-step process. First, it uses the fast scalar indexes to evaluate the filter expression and generate a bitmask that flags all records matching the criteria. Second, the HNSW ANN search is performed
4. *only* on the subset of vectors corresponding to the 'true' values in this bitmask. This dramatically prunes the search space *before* any computationally expensive vector distance calculations are performed, leading to a massive improvement in both speed and relevance.

### **3.4 A Practical Guide to Search Query Formulation**

The following Python code examples, using the pymilvus client, demonstrate how to implement the required search functionalities.

**Setup (Assumed):**

Python

from pymilvus import MilvusClient

# Configure your Milvus client

client = MilvusClient(uri="http://<your-milvus-ip>:19530")

COLLECTION\_NAME = "video\_frames\_db"

# Assume 'embedding\_model' is an object that can generate embeddings

# e.g., embedding\_model.embed(image=img\_data, text=txt\_data)

**Search by Image:**

Python

def search\_by\_image(image\_path, top\_k=10, platform\_filter=None):

# 1. Load and generate semantic embedding for the query image

with open(image\_path, "rb") as f:

image\_bytes = f.read()

query\_vector = embedding\_model.embed(image=image\_bytes)['image\_embedding']

# 2. Build search parameters

search\_params = {"metric\_type": "COSINE", "params": {"ef": 128}}

# 3. Execute the search with an optional metadata filter

results = client.search(

collection\_name=COLLECTION\_NAME,

data=[query\_vector],

limit=top\_k,

search\_params=search\_params,

filter=platform\_filter, # e.g., "platform == 'tiktok'"

output\_fields=["frame\_id", "video\_code", "platform"]

)

return results

**Search by Text:**

Python

def search\_by\_text(text\_query, top\_k=10, filter\_expression=None):

# 1. Generate semantic embedding for the query text

query\_vector = embedding\_model.embed(text=text\_query)['text\_embedding']

# 2. Build search parameters

search\_params = {"metric\_type": "COSINE", "params": {"ef": 128}}

# 3. Execute the search

results = client.search(

collection\_name=COLLECTION\_NAME,

data=[query\_vector],

limit=top\_k,

search\_params=search\_params,

filter=filter\_expression,

output\_fields=["frame\_id", "video\_code", "s3\_uri"]

)

return results

**Search by Video (Example using frame averaging):**

Python

import numpy as np

def search\_by\_video(video\_frames, top\_k=10):

# 1. Generate embeddings for a list of representative frames

embeddings =

for frame in video\_frames:

embedding = embedding\_model.embed(image=frame)['image\_embedding']

embeddings.append(embedding)

# 2. Average the embeddings to create a single query vector

if not embeddings:

return

query\_vector = np.mean(np.array(embeddings), axis=0).tolist()

# 3. Build search parameters and execute search

search\_params = {"metric\_type": "COSINE", "params": {"ef": 128}}

results = client.search(

collection\_name=COLLECTION\_NAME,

data=[query\_vector],

limit=top\_k,

search\_params=search\_params,

output\_fields=["frame\_id", "video\_code"]

)

# Post-process to get unique video codes

unique\_videos = {}

for hit in results:

video\_code = hit['entity']['video\_code']

if video\_code not in unique\_videos:

unique\_videos[video\_code] = hit['distance']

return sorted(unique\_videos.items(), key=lambda item: item)

## **Section 4: The Complete Pipeline: From Ingestion to Retrieval**

This final section integrates all the preceding components into a cohesive, end-to-end system architecture. It outlines the data flow for both ingesting new video content and processing user search queries, providing a clear blueprint for building the production system. The section concludes with a practical strategy for local development and cloud deployment.

### **4.1 The Ingestion Pipeline Architecture**

The ingestion pipeline is responsible for processing new videos, generating the necessary embeddings, and populating the vector database. A robust, scalable, and asynchronous architecture is essential to handle a high volume of incoming content without creating bottlenecks.

The proposed ingestion architecture follows a message-driven, event-based pattern:

1. **Video Upload/Discovery Trigger:** The process begins when a new video's frames are added to their respective folder in the oriane-frames S3 bucket. An S3 Event Notification is configured to fire upon object creation (e.g., for the last frame of a video, or a manifest file). This event publishes a message to a Simple Queue Service (SQS) queue. The message payload contains essential information, such as the platform and video\_code.
2. **Scalable Worker Pool:** A pool of ingestion workers consumes messages from the SQS queue. This pool can be implemented using a variety of scalable compute services, such as AWS Lambda functions or a Kubernetes Deployment running on GKE. Using a queue decouples the upload process from the processing, ensuring that spikes in video uploads can be handled gracefully by the workers at their own pace.
3. **Parallel Embedding Generation:** For each message (representing one video), a worker performs the following steps: a. It lists all frame files associated with the video\_code in the S3 bucket. b. It iterates through each frame. For every frame, it makes **two parallel API calls** to maximize throughput: i. **Semantic Embedding:** A call is made to the commercial embedding model's API (e.g., Voyage AI) to generate the semantic\_embedding. ii. **SSCD Embedding:** A call is made to a self-hosted endpoint for the project's custom SSCD model to generate the sscd\_embedding. This model should be deployed as a scalable service (e.g., on GKE with GPU nodes) to handle the load. c. The worker collects the results for a batch of frames (e.g., 1,000 frames at a time) to prepare for bulk insertion.
4. **Bulk Insertion into Milvus:** Once a batch is ready, the worker constructs the data records according to the schema defined in Section 3.1. It then performs a single, efficient bulk insert() operation into the Milvus collection. Bulk insertion is vastly more performant than inserting records one by one. The worker continues this process until all frames for the video are ingested, after which it deletes the message from the SQS queue.

### **4.2 The Search and Retrieval Workflow**

The search workflow is the core of the application, implementing the two-stage filtering process to efficiently find video copies.

1. **User Query Submission:** A user submits a query to the application's backend API. The query can be a text string, an image file, or a video file.
2. **Stage 1: Semantic Search (Broad Filtering):** a. The backend first processes the query to generate a semantic\_embedding. If the query is an image or text, this is a single API call. If it's a video, it involves extracting representative frames and generating an aggregate embedding (as shown in Section 3.4). b. A search() request is sent to Milvus. This request contains the query semantic\_embedding, any user-specified metadata filters (e.g., platform, date range), and a limit parameter (top\_k) to retrieve a large but manageable set of initial candidates (e.g., k=10,000). c. Milvus executes the filtered HNSW search on the semantic\_embedding field and returns a ranked list of the top\_k most semantically similar frame\_ids.
3. **Intermediate Step: Candidate Video Aggregation:** a. The backend receives the list of frame\_ids from Milvus. b. Using the video\_code returned for each frame, the backend aggregates these results to create a unique, de-duplicated list of candidate video\_codes. This is the crucial pre-filtered list that the "Deep Vision" stage will operate on. This list is significantly smaller than the entire database, achieving the primary goal of the semantic search layer.
4. **Stage 2: "Deep Vision" (High-Precision Matching):** a. The filtered list of unique video\_codes is passed to the existing SSCD pipeline. b. Instead of fetching and processing raw frames, the SSCD pipeline now performs a highly efficient lookup in Milvus. It executes a query() operation to retrieve the pre-computed sscd\_embeddings for all frames belonging to the candidate videos. This is done using a WHERE clause on the indexed video\_code field (e.g., client.query(collection\_name=COLLECTION\_NAME, filter="video\_code in ['video1', 'video2',...]", output\_fields=["sscd\_embedding", "frame\_id"])). c. With the relevant sscd\_embeddings now in memory, the SSCD model can perform its final, high-precision frame-by-frame comparison to identify exact copies, remakes, or other transformations. This two-stage process ensures that the computationally expensive SSCD analysis is only performed on a small, highly relevant subset of the total video library.

### **4.3 Local Testing and Cloud Deployment Strategy**

A phased approach to development and deployment is recommended to ensure stability, manage costs, and maintain flexibility.

**Local Development Environment (Docker):**

* **Database:** For initial development and simple notebook-based experiments, **Milvus Lite** is an excellent choice. It is a lightweight, pip-installable version of Milvus that runs entirely within a Python process, requiring no external dependencies. For more comprehensive local testing that mirrors a production environment, use the official Milvus
* docker-compose.yml file to run a **Milvus Standalone** instance on a local machine. This bundles all necessary components into a single Docker container.
* **Models:** The open-source embedding models (e.g., OpenCLIP) and the custom SSCD model should be run in their own Docker containers. This allows for the entire application stack—backend, workers, models, and database—to be orchestrated locally using Docker Compose.

**Cloud Deployment (Kubernetes on GCP):**

The goal is to build a cloud-agnostic, production-grade system on GCP, leveraging the available credits without being locked into proprietary services. Kubernetes is the ideal platform for this.

1. **Provision GKE Cluster:** Create a Google Kubernetes Engine (GKE) cluster in the desired region. Use a mix of node pools, including standard CPU-based nodes for most services and GPU-equipped nodes for serving the SSCD model.
2. **Deploy Milvus Distributed:** The most critical step is to deploy a production-grade Milvus cluster. This is achieved using the official **Milvus Helm chart** for Kubernetes. The Helm chart automates the deployment of the entire Milvus distributed architecture, including all coordinator services and worker nodes, and configures the necessary persistent storage using GCP Persistent Disks.
3. **Deploy Application Services:** Containerize the application backend, the ingestion workers, and the SSCD model inference server. Deploy these as separate services (Deployments and Services) within the same GKE cluster.
4. **Configure Networking:** Ensure all services are deployed within the same Virtual Private Cloud (VPC). Use Kubernetes internal DNS for service discovery, allowing the application backend and workers to communicate with the Milvus cluster using its internal service name.
5. **Leverage GCP Credits:** This entire infrastructure—GKE nodes, Persistent Disk storage, and any associated networking—runs on Google Cloud compute and storage resources. Therefore, its operational costs will be covered by the $300k in GCP credits. This strategy achieves the best of both worlds: it leverages the financial advantage of the GCP credits to build and operate a massive-scale system, while the use of open-source, containerized components like Milvus and Kubernetes ensures the entire stack remains fully cloud-agnostic and can be migrated to any other cloud provider or on-premises environment in the future.

Sources used in the report:



[researchgate.net](https://www.researchgate.net/figure/Zero-shot-performance-comparison-of-CLIP-ALIGN-BLIP-and-CoCa-models-across_tbl2_391974842)

[Zero-shot performance comparison (%) of CLIP, ALIGN, BLIP, and CoCa models across datasets. - ResearchGate](https://www.researchgate.net/figure/Zero-shot-performance-comparison-of-CLIP-ALIGN-BLIP-and-CoCa-models-across_tbl2_391974842)

[Opens in a new window](https://www.researchgate.net/figure/Zero-shot-performance-comparison-of-CLIP-ALIGN-BLIP-and-CoCa-models-across_tbl2_391974842)

[](https://arxiv.org/html/2403.11497v1)

[arxiv.org](https://arxiv.org/html/2403.11497v1)

[Do CLIPs Always Generalize Better than ImageNet Models? - arXiv](https://arxiv.org/html/2403.11497v1)

[Opens in a new window](https://arxiv.org/html/2403.11497v1)

[](https://www.byteplus.com/en/topic/516148)

[byteplus.com](https://www.byteplus.com/en/topic/516148)

[Best Multimodal Embedding Models for Scalable Tech Tools - BytePlus](https://www.byteplus.com/en/topic/516148)

[Opens in a new window](https://www.byteplus.com/en/topic/516148)

[](https://www.researchgate.net/publication/391629733_Asymmetric_Semantic_Search_Using_Multi-Dimensional_Vector_Text_Data_Representation_Open_Semantic_Technologies_for_Intelligent_Systems_ISSN_2415-7740)

[researchgate.net](https://www.researchgate.net/publication/391629733_Asymmetric_Semantic_Search_Using_Multi-Dimensional_Vector_Text_Data_Representation_Open_Semantic_Technologies_for_Intelligent_Systems_ISSN_2415-7740)

[Asymmetric Semantic Search Using Multi-Dimensional Vector Text Data Representation [Open Semantic Technologies for Intelligent Systems (ISSN: 2415-7740)] - ResearchGate](https://www.researchgate.net/publication/391629733_Asymmetric_Semantic_Search_Using_Multi-Dimensional_Vector_Text_Data_Representation_Open_Semantic_Technologies_for_Intelligent_Systems_ISSN_2415-7740)

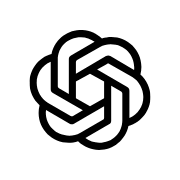
[Opens in a new window](https://www.researchgate.net/publication/391629733_Asymmetric_Semantic_Search_Using_Multi-Dimensional_Vector_Text_Data_Representation_Open_Semantic_Technologies_for_Intelligent_Systems_ISSN_2415-7740)

[](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/multimodal-embeddings-api)

[cloud.google.com](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/multimodal-embeddings-api)

[Multimodal embeddings API | Generative AI on Vertex AI - Google Cloud](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/multimodal-embeddings-api)

[Opens in a new window](https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/multimodal-embeddings-api)

[](https://openai.com/index/clip/)

[openai.com](https://openai.com/index/clip/)

[CLIP: Connecting text and images - OpenAI](https://openai.com/index/clip/)

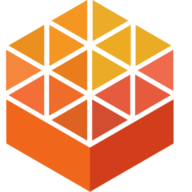
[Opens in a new window](https://openai.com/index/clip/)

[](https://roboflow.com/model-feature/image-embedding)

[roboflow.com](https://roboflow.com/model-feature/image-embedding)

[Top Image Embedding Models - Roboflow](https://roboflow.com/model-feature/image-embedding)

[Opens in a new window](https://roboflow.com/model-feature/image-embedding)

[](https://synthesis.ai/2023/05/16/generative-ai-iv-clip-and-multimodal-retrieval/)

[synthesis.ai](https://synthesis.ai/2023/05/16/generative-ai-iv-clip-and-multimodal-retrieval/)

[Generative AI IV - CLIP and multimodal retrieval - Synthesis AI](https://synthesis.ai/2023/05/16/generative-ai-iv-clip-and-multimodal-retrieval/)

[Opens in a new window](https://synthesis.ai/2023/05/16/generative-ai-iv-clip-and-multimodal-retrieval/)

[](https://openaccess.thecvf.com/content/CVPR2022/papers/Pizzi_A_Self-Supervised_Descriptor_for_Image_Copy_Detection_CVPR_2022_paper.pdf)

[openaccess.thecvf.com](https://openaccess.thecvf.com/content/CVPR2022/papers/Pizzi_A_Self-Supervised_Descriptor_for_Image_Copy_Detection_CVPR_2022_paper.pdf)

[A Self-Supervised Descriptor for Image Copy ... - CVF Open Access](https://openaccess.thecvf.com/content/CVPR2022/papers/Pizzi_A_Self-Supervised_Descriptor_for_Image_Copy_Detection_CVPR_2022_paper.pdf)

[Opens in a new window](https://openaccess.thecvf.com/content/CVPR2022/papers/Pizzi_A_Self-Supervised_Descriptor_for_Image_Copy_Detection_CVPR_2022_paper.pdf)

[](https://weaviate.io/developers/weaviate/concepts/vector-index#:~:text=Hierarchical%20Navigable%20Small%20World%20(HNSW)%20is%20an%20algorithm%20that%20works,vectors%20can%20be%20resource%20intensive.)

[weaviate.io](https://weaviate.io/developers/weaviate/concepts/vector-index#:~:text=Hierarchical%20Navigable%20Small%20World%20(HNSW)%20is%20an%20algorithm%20that%20works,vectors%20can%20be%20resource%20intensive.)

[weaviate.io](https://weaviate.io/developers/weaviate/concepts/vector-index#:~:text=Hierarchical%20Navigable%20Small%20World%20(HNSW)%20is%20an%20algorithm%20that%20works,vectors%20can%20be%20resource%20intensive.)

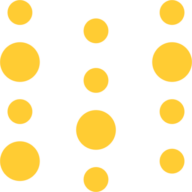
[Opens in a new window](https://weaviate.io/developers/weaviate/concepts/vector-index#:~:text=Hierarchical%20Navigable%20Small%20World%20(HNSW)%20is%20an%20algorithm%20that%20works,vectors%20can%20be%20resource%20intensive.)

[](https://www.timescale.com/blog/vector-database-basics-hnsw)

[timescale.com](https://www.timescale.com/blog/vector-database-basics-hnsw)

[Vector Database Basics: HNSW - Timescale](https://www.timescale.com/blog/vector-database-basics-hnsw)

[Opens in a new window](https://www.timescale.com/blog/vector-database-basics-hnsw)

[](https://wandb.ai/byyoung3/ml-news/reports/Voyage-Multimodal-3-A-new-Multimodal-Embedding-Model--VmlldzoxMDIxNDc2Mg)

[wandb.ai](https://wandb.ai/byyoung3/ml-news/reports/Voyage-Multimodal-3-A-new-Multimodal-Embedding-Model--VmlldzoxMDIxNDc2Mg)

[Voyage-Multimodal-3: A new Multimodal Embedding Model | ml-news - Wandb](https://wandb.ai/byyoung3/ml-news/reports/Voyage-Multimodal-3-A-new-Multimodal-Embedding-Model--VmlldzoxMDIxNDc2Mg)

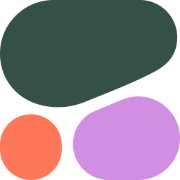
[Opens in a new window](https://wandb.ai/byyoung3/ml-news/reports/Voyage-Multimodal-3-A-new-Multimodal-Embedding-Model--VmlldzoxMDIxNDc2Mg)

[](https://milvus.io/ai-quick-reference/how-does-indexing-work-in-a-vector-db-ivf-hnsw-pq-etc)

[milvus.io](https://milvus.io/ai-quick-reference/how-does-indexing-work-in-a-vector-db-ivf-hnsw-pq-etc)

[How does indexing work in a vector DB (IVF, HNSW, PQ, etc.)? - Milvus](https://milvus.io/ai-quick-reference/how-does-indexing-work-in-a-vector-db-ivf-hnsw-pq-etc)

[Opens in a new window](https://milvus.io/ai-quick-reference/how-does-indexing-work-in-a-vector-db-ivf-hnsw-pq-etc)

[](https://cohere.com/blog/multimodal-embed-3)

[cohere.com](https://cohere.com/blog/multimodal-embed-3)

[Introducing multimodal Embed 3: Powering AI search - Cohere](https://cohere.com/blog/multimodal-embed-3)

[Opens in a new window](https://cohere.com/blog/multimodal-embed-3)

[](https://en.wikipedia.org/wiki/Hierarchical_navigable_small_world)

[en.wikipedia.org](https://en.wikipedia.org/wiki/Hierarchical_navigable_small_world)

[Hierarchical navigable small world - Wikipedia](https://en.wikipedia.org/wiki/Hierarchical_navigable_small_world)

[Opens in a new window](https://en.wikipedia.org/wiki/Hierarchical_navigable_small_world)

[](https://www.googlecloudcommunity.com/gc/AI-ML/Estimating-Vertex-AI-Vector-Search-Costs-Seeking-Cost-Effective/m-p/796972)

[googlecloudcommunity.com](https://www.googlecloudcommunity.com/gc/AI-ML/Estimating-Vertex-AI-Vector-Search-Costs-Seeking-Cost-Effective/m-p/796972)

[Re: Estimating Vertex AI Vector Search Costs: Seek... - Google Cloud Community](https://www.googlecloudcommunity.com/gc/AI-ML/Estimating-Vertex-AI-Vector-Search-Costs-Seeking-Cost-Effective/m-p/796972)

[Opens in a new window](https://www.googlecloudcommunity.com/gc/AI-ML/Estimating-Vertex-AI-Vector-Search-Costs-Seeking-Cost-Effective/m-p/796972)

[](https://neuroflash.com/blog/google-vertex-ai-pricing-pros-cons/)

[neuroflash.com](https://neuroflash.com/blog/google-vertex-ai-pricing-pros-cons/)

[Google Vertex AI Pricing: The Pros and Cons - neuroflash](https://neuroflash.com/blog/google-vertex-ai-pricing-pros-cons/)

[Opens in a new window](https://neuroflash.com/blog/google-vertex-ai-pricing-pros-cons/)

[](https://www.langchain.ca/blog/top-10-vector-databases-2024/)

[langchain.ca](https://www.langchain.ca/blog/top-10-vector-databases-2024/)

[Top 10 Vector Databases in 2024 - LLM & Langchain Blogs](https://www.langchain.ca/blog/top-10-vector-databases-2024/)

[Opens in a new window](https://www.langchain.ca/blog/top-10-vector-databases-2024/)

[](https://www.kdnuggets.com/the-5-best-vector-databases-you-must-try-in-2024)

[kdnuggets.com](https://www.kdnuggets.com/the-5-best-vector-databases-you-must-try-in-2024)

[The 5 Best Vector Databases You Must Try in 2024 - KDnuggets](https://www.kdnuggets.com/the-5-best-vector-databases-you-must-try-in-2024)

[Opens in a new window](https://www.kdnuggets.com/the-5-best-vector-databases-you-must-try-in-2024)

[](https://cloud.google.com/vertex-ai/docs/vector-search/overview)

[cloud.google.com](https://cloud.google.com/vertex-ai/docs/vector-search/overview)

[Vector Search | Vertex AI - Google Cloud](https://cloud.google.com/vertex-ai/docs/vector-search/overview)

[Opens in a new window](https://cloud.google.com/vertex-ai/docs/vector-search/overview)

[](https://www.googlecloudcommunity.com/gc/Community-Blogs/Building-Enterprise-Ready-Generative-AI-Applications-with-Vertex/ba-p/723378)

[googlecloudcommunity.com](https://www.googlecloudcommunity.com/gc/Community-Blogs/Building-Enterprise-Ready-Generative-AI-Applications-with-Vertex/ba-p/723378)

[Building Enterprise-Ready Generative AI Applications with Vertex AI Vector Search](https://www.googlecloudcommunity.com/gc/Community-Blogs/Building-Enterprise-Ready-Generative-AI-Applications-with-Vertex/ba-p/723378)

[Opens in a new window](https://www.googlecloudcommunity.com/gc/Community-Blogs/Building-Enterprise-Ready-Generative-AI-Applications-with-Vertex/ba-p/723378)

[](https://cloud.google.com/vertex-ai/docs/vector-search/quickstart)

[cloud.google.com](https://cloud.google.com/vertex-ai/docs/vector-search/quickstart)

[Vector Search quickstart | Vertex AI - Google Cloud](https://cloud.google.com/vertex-ai/docs/vector-search/quickstart)

[Opens in a new window](https://cloud.google.com/vertex-ai/docs/vector-search/quickstart)

[](https://ai.google.dev/gemini-api/docs/pricing)

[ai.google.dev](https://ai.google.dev/gemini-api/docs/pricing)

[Gemini Developer API Pricing | Gemini API | Google AI for Developers](https://ai.google.dev/gemini-api/docs/pricing)

[Opens in a new window](https://ai.google.dev/gemini-api/docs/pricing)

[](https://blog.voyageai.com/2025/01/07/voyage-3-large/)

[blog.voyageai.com](https://blog.voyageai.com/2025/01/07/voyage-3-large/)

[voyage-3-large: the new state-of-the-art general-purpose embedding model](https://blog.voyageai.com/2025/01/07/voyage-3-large/)

[Opens in a new window](https://blog.voyageai.com/2025/01/07/voyage-3-large/)

[](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb)

[github.com](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb)

[generative-ai/embeddings/intro\_multimodal\_embeddings.ipynb at main - GitHub](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb)

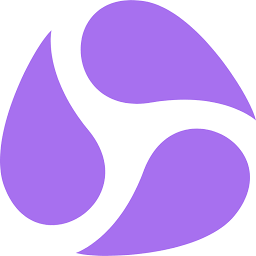
[Opens in a new window](https://github.com/GoogleCloudPlatform/generative-ai/blob/main/embeddings/intro_multimodal_embeddings.ipynb)

[](https://github.com/shanyu-sys/Awesome-multi-modal-embeddings)

[github.com](https://github.com/shanyu-sys/Awesome-multi-modal-embeddings)

[shanyu-sys/Awesome-multi-modal-embeddings - GitHub](https://github.com/shanyu-sys/Awesome-multi-modal-embeddings)

[Opens in a new window](https://github.com/shanyu-sys/Awesome-multi-modal-embeddings)

[](https://liquidmetal.ai/casesAndBlogs/vector-comparison/)

[liquidmetal.ai](https://liquidmetal.ai/casesAndBlogs/vector-comparison/)

[Vector Database Comparison: Pinecone vs Weaviate vs Qdrant vs FAISS vs Milvus vs Chroma (2025) | LiquidMetal AI](https://liquidmetal.ai/casesAndBlogs/vector-comparison/)

[Opens in a new window](https://liquidmetal.ai/casesAndBlogs/vector-comparison/)

[](https://milvus.io/ai-quick-reference/what-vector-databases-support-multimodal-search-effectively)

[milvus.io](https://milvus.io/ai-quick-reference/what-vector-databases-support-multimodal-search-effectively)

[What vector databases support multimodal search effectively? - Milvus](https://milvus.io/ai-quick-reference/what-vector-databases-support-multimodal-search-effectively)

[Opens in a new window](https://milvus.io/ai-quick-reference/what-vector-databases-support-multimodal-search-effectively)

[](https://pureinsights.com/blog/2024/comparing-vector-search-solutions-2024/)

[pureinsights.com](https://pureinsights.com/blog/2024/comparing-vector-search-solutions-2024/)

[Comparing Vector Search Solutions 2024 - Pureinsights](https://pureinsights.com/blog/2024/comparing-vector-search-solutions-2024/)

[Opens in a new window](https://pureinsights.com/blog/2024/comparing-vector-search-solutions-2024/)

[](https://dev.to/mongodb/building-multimodal-ai-applications-with-mongodb-voyage-ai-and-gemini-49g3)

[dev.to](https://dev.to/mongodb/building-multimodal-ai-applications-with-mongodb-voyage-ai-and-gemini-49g3)

[Building Multimodal AI Applications With MongoDB, Voyage AI, and Gemini](https://dev.to/mongodb/building-multimodal-ai-applications-with-mongodb-voyage-ai-and-gemini-49g3)

[Opens in a new window](https://dev.to/mongodb/building-multimodal-ai-applications-with-mongodb-voyage-ai-and-gemini-49g3)

[](https://qdrant.tech/articles/what-is-a-vector-database/)

[qdrant.tech](https://qdrant.tech/articles/what-is-a-vector-database/)

[An Introduction to Vector Databases - Qdrant](https://qdrant.tech/articles/what-is-a-vector-database/)

[Opens in a new window](https://qdrant.tech/articles/what-is-a-vector-database/)

[](https://weaviate.io/developers/weaviate/concepts/vector-index)

[weaviate.io](https://weaviate.io/developers/weaviate/concepts/vector-index)

[Vector Indexing | Weaviate](https://weaviate.io/developers/weaviate/concepts/vector-index)

[Opens in a new window](https://weaviate.io/developers/weaviate/concepts/vector-index)

[](https://milvus.io/id/blog/introduce-milvus-2-6-built-for-scale-designed-to-reduce-costs.md)

[milvus.io](https://milvus.io/id/blog/introduce-milvus-2-6-built-for-scale-designed-to-reduce-costs.md)

[Introducing Milvus 2.6: Affordable Vector Search at Billion Scale](https://milvus.io/id/blog/introduce-milvus-2-6-built-for-scale-designed-to-reduce-costs.md)

[Opens in a new window](https://milvus.io/id/blog/introduce-milvus-2-6-built-for-scale-designed-to-reduce-costs.md)

[](https://www.truefoundry.com/blog/best-vector-databases)

[truefoundry.com](https://www.truefoundry.com/blog/best-vector-databases)

[7 Best Vector Databases in 2025 - TrueFoundry](https://www.truefoundry.com/blog/best-vector-databases)

[Opens in a new window](https://www.truefoundry.com/blog/best-vector-databases)

[](https://milvus.io/docs/filtered-search.md)

[milvus.io](https://milvus.io/docs/filtered-search.md)

[Filtered Search | Milvus Documentation](https://milvus.io/docs/filtered-search.md)

[Opens in a new window](https://milvus.io/docs/filtered-search.md)

[](https://milvus.io/ai-quick-reference/how-does-a-vector-database-handle-scaling-up-to-millions-or-billions-of-vectors-and-what-architectural-features-enable-this-scalability)

[milvus.io](https://milvus.io/ai-quick-reference/how-does-a-vector-database-handle-scaling-up-to-millions-or-billions-of-vectors-and-what-architectural-features-enable-this-scalability)

[How does a vector database handle scaling up to millions or billions of vectors, and what architectural features enable this scalability? - Milvus](https://milvus.io/ai-quick-reference/how-does-a-vector-database-handle-scaling-up-to-millions-or-billions-of-vectors-and-what-architectural-features-enable-this-scalability)

[Opens in a new window](https://milvus.io/ai-quick-reference/how-does-a-vector-database-handle-scaling-up-to-millions-or-billions-of-vectors-and-what-architectural-features-enable-this-scalability)

[](https://dataaspirant.com/popular-vector-databases/)

[dataaspirant.com](https://dataaspirant.com/popular-vector-databases/)

[Most Popular Vector Databases You Must Know in 2025 - Dataaspirant](https://dataaspirant.com/popular-vector-databases/)

[Opens in a new window](https://dataaspirant.com/popular-vector-databases/)

[](https://milvus.io/blog/how-to-filter-efficiently-without-killing-recall.md)

[milvus.io](https://milvus.io/blog/how-to-filter-efficiently-without-killing-recall.md)

[Vector Search in the Real World: How to Filter Efficiently Without Killing Recall - Milvus Blog](https://milvus.io/blog/how-to-filter-efficiently-without-killing-recall.md)

[Opens in a new window](https://milvus.io/blog/how-to-filter-efficiently-without-killing-recall.md)

[](https://milvus.io/docs/v2.4.x/overview.md)

[milvus.io](https://milvus.io/docs/v2.4.x/overview.md)

[What is Milvus Milvus v2.4.x documentation](https://milvus.io/docs/v2.4.x/overview.md)

[Opens in a new window](https://milvus.io/docs/v2.4.x/overview.md)

[](https://www.cloudraft.io/blog/top-5-vector-databases)

[cloudraft.io](https://www.cloudraft.io/blog/top-5-vector-databases)

[Top 5 Vector Databases in 2025 - CloudRaft](https://www.cloudraft.io/blog/top-5-vector-databases)

[Opens in a new window](https://www.cloudraft.io/blog/top-5-vector-databases)

[](https://milvus.io/docs/architecture_overview.md)

[milvus.io](https://milvus.io/docs/architecture_overview.md)

[Milvus Architecture Overview](https://milvus.io/docs/architecture_overview.md)

[Opens in a new window](https://milvus.io/docs/architecture_overview.md)

[](https://milvus.io/docs/hnsw.md)

[milvus.io](https://milvus.io/docs/hnsw.md)

[HNSW | Milvus Documentation](https://milvus.io/docs/hnsw.md)

[Opens in a new window](https://milvus.io/docs/hnsw.md)

[](https://milvus.io/)

[milvus.io](https://milvus.io/)

[Milvus | High-Performance Vector Database Built for Scale](https://milvus.io/)

[Opens in a new window](https://milvus.io/)

[](https://github.com/milvus-io/milvus/discussions/28135)

[github.com](https://github.com/milvus-io/milvus/discussions/28135)

[Filtering support · milvus-io milvus · Discussion #28135 - GitHub](https://github.com/milvus-io/milvus/discussions/28135)

[Opens in a new window](https://github.com/milvus-io/milvus/discussions/28135)

[](https://docs.llamaindex.ai/en/stable/examples/vector_stores/MilvusOperatorFunctionDemo/)

[docs.llamaindex.ai](https://docs.llamaindex.ai/en/stable/examples/vector_stores/MilvusOperatorFunctionDemo/)

[Milvus Vector Store - Metadata Filter - LlamaIndex](https://docs.llamaindex.ai/en/stable/examples/vector_stores/MilvusOperatorFunctionDemo/)

[Opens in a new window](https://docs.llamaindex.ai/en/stable/examples/vector_stores/MilvusOperatorFunctionDemo/)

[](https://roboflow.com/licensing)

[roboflow.com](https://roboflow.com/licensing)

[Roboflow Licensing](https://roboflow.com/licensing)

[Opens in a new window](https://roboflow.com/licensing)

[](https://aws.amazon.com/marketplace/pp/prodview-b4mpgdxvpa3v6)

[aws.amazon.com](https://aws.amazon.com/marketplace/pp/prodview-b4mpgdxvpa3v6)

[AWS Marketplace: Cohere Embed Model 3 - Multilingual - Amazon.com](https://aws.amazon.com/marketplace/pp/prodview-b4mpgdxvpa3v6)

[Opens in a new window](https://aws.amazon.com/marketplace/pp/prodview-b4mpgdxvpa3v6)

[](https://github.com/Azure/azure-search-vector-samples/blob/main/demo-python/code/e2e-demos/azure-ai-search-cohere-embed-v3.ipynb)

[github.com](https://github.com/Azure/azure-search-vector-samples/blob/main/demo-python/code/e2e-demos/azure-ai-search-cohere-embed-v3.ipynb)

[Azure AI Search with Cohere Embed V3 Int8 and Binary Embeddings Support - GitHub](https://github.com/Azure/azure-search-vector-samples/blob/main/demo-python/code/e2e-demos/azure-ai-search-cohere-embed-v3.ipynb)

[Opens in a new window](https://github.com/Azure/azure-search-vector-samples/blob/main/demo-python/code/e2e-demos/azure-ai-search-cohere-embed-v3.ipynb)

[](https://cohere.com/blog/int8-binary-embeddings)

[cohere.com](https://cohere.com/blog/int8-binary-embeddings)

[Cohere int8 & binary Embeddings - Scale Your Vector Database to Large Datasets](https://cohere.com/blog/int8-binary-embeddings)

[Opens in a new window](https://cohere.com/blog/int8-binary-embeddings)

[](https://learn.microsoft.com/en-us/azure/ai-studio/how-to/deploy-models-cohere-embed)

[learn.microsoft.com](https://learn.microsoft.com/en-us/azure/ai-studio/how-to/deploy-models-cohere-embed)

[How to use Cohere Embed V3 models with Azure AI Foundry - Learn Microsoft](https://learn.microsoft.com/en-us/azure/ai-studio/how-to/deploy-models-cohere-embed)

[Opens in a new window](https://learn.microsoft.com/en-us/azure/ai-studio/how-to/deploy-models-cohere-embed)

[](https://aws.amazon.com/marketplace/pp/prodview-hrid2zxusacxy)

[aws.amazon.com](https://aws.amazon.com/marketplace/pp/prodview-hrid2zxusacxy)

[AWS Marketplace: voyage-multimodal-3 Embedding Model - Amazon.com](https://aws.amazon.com/marketplace/pp/prodview-hrid2zxusacxy)

[Opens in a new window](https://aws.amazon.com/marketplace/pp/prodview-hrid2zxusacxy)

[](https://docs.anthropic.com/en/docs/build-with-claude/embeddings)

[docs.anthropic.com](https://docs.anthropic.com/en/docs/build-with-claude/embeddings)

[Embeddings - Anthropic API](https://docs.anthropic.com/en/docs/build-with-claude/embeddings)

[Opens in a new window](https://docs.anthropic.com/en/docs/build-with-claude/embeddings)

[](https://blog.voyageai.com/2024/11/12/voyage-multimodal-3/)

[blog.voyageai.com](https://blog.voyageai.com/2024/11/12/voyage-multimodal-3/)

[voyage-multimodal-3: all-in-one embedding model for interleaved text, images, and screenshots](https://blog.voyageai.com/2024/11/12/voyage-multimodal-3/)

[Opens in a new window](https://blog.voyageai.com/2024/11/12/voyage-multimodal-3/)

[](https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings)

[cloud.google.com](https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings)

[Get multimodal embeddings | Generative AI on Vertex AI - Google Cloud](https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings)

[Opens in a new window](https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal-embeddings)

[](https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/multimodalembedding)

[console.cloud.google.com](https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/multimodalembedding)

[Embeddings for Multimodal – Vertex AI - Google Cloud Console](https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/multimodalembedding)

[Opens in a new window](https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/multimodalembedding)

[](https://www.youtube.com/watch?v=YXp6obGHy4E)

[youtube.com](https://www.youtube.com/watch?v=YXp6obGHy4E)

[How to use GCP Vertex AI multimodal embedding - YouTube](https://www.youtube.com/watch?v=YXp6obGHy4E)

[Opens in a new window](https://www.youtube.com/watch?v=YXp6obGHy4E)

Sources read but not used in the report