

Beyond Metadata Search

How to distribute embeddings for semantic search

New to geo embeddings?

Intro to Geo Embeddings and Geospatial Foundation Models

14:00 · this room

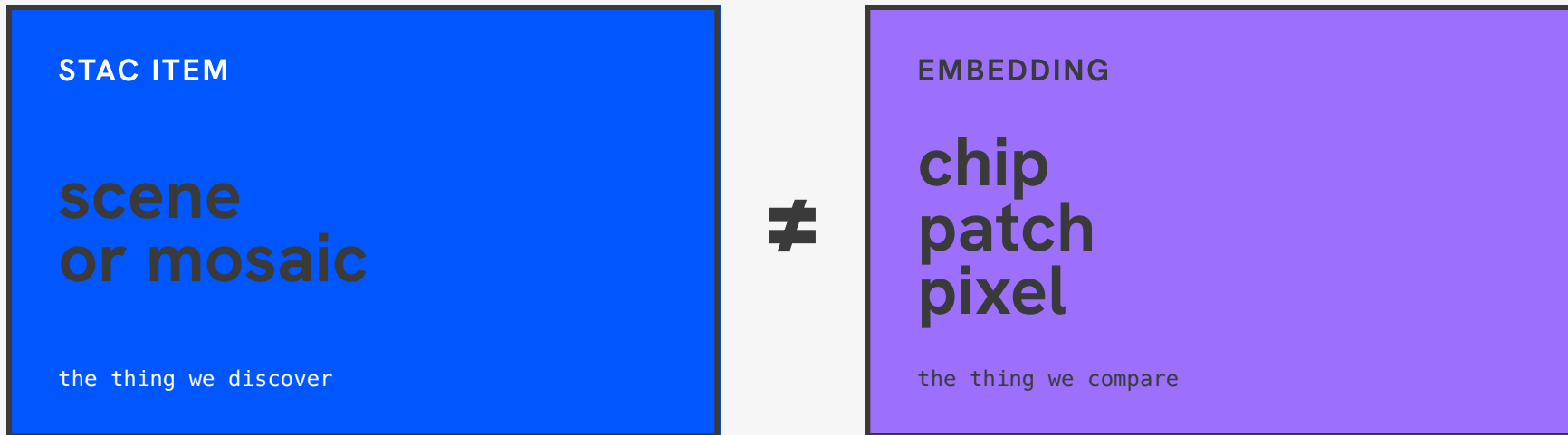
That talk covers what embeddings are, how foundation models produce them, and how to use them.

This talk starts with a narrower question: **how should embeddings fit into STAC for distribution?**

01 **Distribute the embeddings**

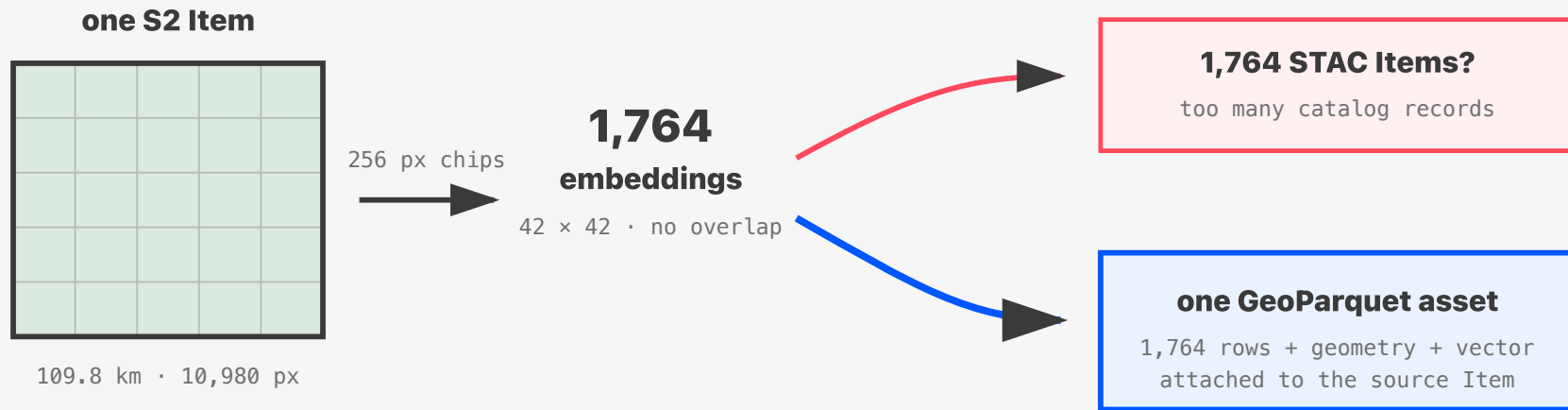
Items, assets, and Parquet

The mismatch



How do we distribute and search those embeddings without making every cell a STAC Item or depending on one global vector service?

One scene is not 1,764 Items



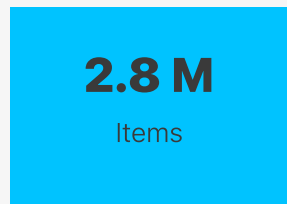
A Sentinel-2 tile is 10,980 pixels across at 10 metres.

Cut it into non-overlapping 256-pixel chips: $42 \times 42 = 1,764$ embeddings.

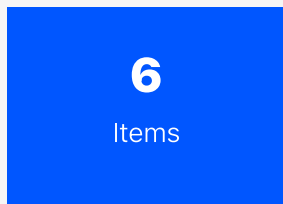
Those are rows in a file. They should not become 1,764 new STAC Items.

Search files, not a global index

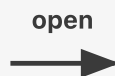
STAC SEARCH



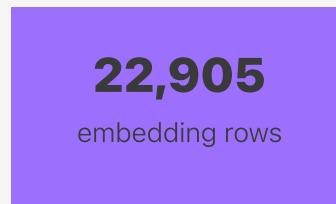
the catalog



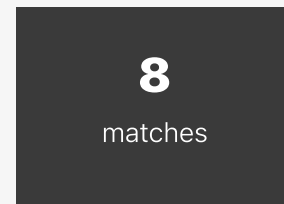
six Parquet assets



VECTOR SEARCH



geometry + vector



GeoJSON Features

STAC search finds the Items. Their embedding assets point to the Parquet files. Then we run vector search over those files.

You can build an index on top. The Parquet files remain the source.

An embedding is a **summary.**

A summary should point to its sources.

Four pieces

01 Parquet

Embedding cells stay as rows in a file.

02 STAC Item

The Parquet file is an asset on a source or derived Item.

03 Provenance

Multi-source products get a derived Item with source links.

04 Search

STAC selects files; a vector ranks the rows.

A simple example

One Sentinel-2 Item already has assets for:

- red, green, blue, NIR, etc.
- visual preview
- scene classification
- metadata

Add one more:

- **Clay embeddings**

STAC Item · S2 scene

```
├─ red.tif
├─ green.tif
├─ nir.tif
├─ visual.tif
└─ embeddings.parquet
    ├─ chip geometry
    └─ embedding vector
```

The embedding is derived from this observation, so it belongs with it.

The 1,764 chips are rows inside `embeddings.parquet`, not children in the STAC catalog.

When one Item isn't enough

Multi-temporal

Sentinel-2 observations from January, April, July, and October

one embedding product for the year

Multi-modal

Sentinel-1 radar plus Sentinel-2 optical imagery

one embedding product combining both sensors

In both cases, create a derived Item for the embedding product and link it to every source Item with `derived_from`.

Two products, two STAC shapes

One scene in, chips out

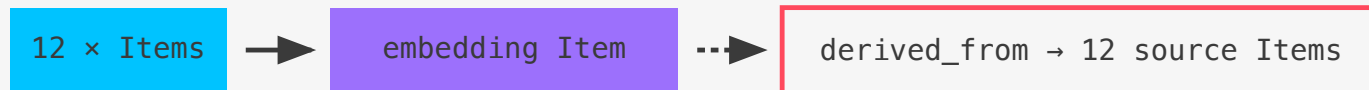
optical, per acquisition



an asset on the Item it came from – the provenance is the parent

Many scenes in, one surface out

per-pixel · annual · time series



One observation in: keep the embedding asset on that Item.

Many observations in: create a derived Item and preserve every source link.

A mosaic means fewer Items and larger Parquet files. It does not change the pattern.

**A multi-source embedding should be
a STAC Item
linked to its source Items.**

What goes on that Item

Geometry

The area summarized by the embedding product.

Datetime

The interval covered by the source observations.

Asset

The embedding file.

Links

`derived_from` links to the source Items.

```
Item · hiroshima-2025-embedding
├── assets
│   └── embeddings.tif
└── links
    ├── derived_from → S2-2025-01
    ├── derived_from → S1-2025-03
    ├── derived_from → S2-2025-07
    └── derived_from → S1-2025-11
```

The source imagery stays on the source Items. Nothing is copied.

Why keep the sources?

- To see **what the model actually saw**
- To reproduce preprocessing and inference
- To investigate clouds, seams, missing dates, and sensor artifacts
- To compare a match with the original pixels
- To rebuild the embedding when the model changes
- To decide whether the result is trustworthy for a new use

THE VECTOR IS USEFUL BECAUSE IT COMPRESSES THE EVIDENCE. IT IS NOT A REPLACEMENT FOR THE EVIDENCE.

AlphaEarth is the difficult case

Would I explore with it?

Yes.

Would I use it as evidence?

It depends.

The annual embeddings are useful. But the source corpus is not reproducible and the inference model is closed.

OPEN OUTPUTS ARE NOT THE SAME THING AS OPEN PROVENANCE.

02

What Exists Today

The draft geo-embeddings STAC extension

Geo Embeddings Extension

geo-embeddings/embeddings-stac-specification · Proposal · v0.0.1

Product

emb:type	pixel or chip
emb:dimensions	vector length
data_type	numeric type
gsd	ground sampling distance
emb:chip_layout	chip size, stride, grid
emb:temporal_resolution	time represented
emb:fit_for_use	retrieval, classification...
emb:uncertainty	uncertainty information

Processing

emb:runtime_parameters	inference parameters
emb:preprocessing	pre-inference preparation
emb:postprocessing	output transformations
emb:quantization	vector encoding
emb:energy	compute and energy use
processing:datetime	production time
processing:version	software version

THE DRAFT ALSO PERMITS AN UNDOCUMENTED `EMB:SPATIAL\_COVERAGE` ITEM PROPERTY.

Links defined by the extension

Link relation	Points to
<code>emb:model</code>	the model used to create the embeddings
<code>emb:source-data</code>	the source data
<code>emb:decoder</code>	a decoder for the embedding representation
<code>emb:benchmark</code>	evaluation or benchmark information

`emb:model` and `emb:source-data` are required on Collections in the current draft.
For derived Items, STAC core already has `derived_from`. I would use that for source Items and reserve asset-level links for the exact files read.

Where the draft allows each field

Scope	Allowed today?	Example
Collection	yes	type, dimensions, chip layout
Collection summaries	yes	type, dimensions, temporal resolution
Item properties	some fields	preprocessing, quantization, energy
Item assets	no <code>emb:*</code> fields	—
Collection assets	no <code>emb:*</code> fields	—
<code>item_assets</code>	no <code>emb:*</code> fields	—

The asset schema explicitly rejects property names beginning with `emb: .`

The asset is the missing scope

Item

└ properties	datetime · eo:cloud_cover · emb:dimensions	allowed
└ assets		
└ B04.tif	raster: · file: · proj:	allowed
└ visual.tif	raster: · file: · proj:	allowed
└ embeddings.parquet	table: · file: · emb:dimensions	not allowed

the current draft does not permit emb: fields at asset scope

Keep the Collection and Item fields. Let the same extension describe the file too.

**The extension describes embeddings,
but not the embedding asset .**

Proposed change

Let the embedding fields appear on assets.

Put on the Item or Collection	Put on the embedding asset
spatial and temporal extent	dimensions and data type
intended use	model identity and version
product-level provenance	normalization and distance metric
searchable product characteristics	chip layout and vector column
links to source Items	preprocessing and quantization

Some fields belong in both places. If the asset says something more specific, use that.

One Item can carry more than one embedding

STAC Item · one Sentinel-2 scene

```
|— visual          image/tiff
|— embeddings-clay application/vnd.apache.parquet
|   |— emb:dimensions 1024
|   |— emb:model      clay-v1.5
|— embeddings-terramind application/vnd.apache.parquet
|   |— emb:dimensions 768
|   |— emb:model      terramind-v1
```

Collection-level metadata cannot describe both assets correctly. Asset-level metadata can.

A multi-source Item

```
{
  "type": "Feature",
  "id": "hiroshima-alphaearth-2025",
  "properties": {
    "start_datetime": "2025-01-01T00:00:00Z",
    "end_datetime": "2025-12-31T23:59:59Z"
  },
  "assets": { "embeddings": {
    "href": "alphaearth-2025.tif",
    "emb:type": "pixel",
    "emb:dimensions": 64
  } },
  "links": [
    { "rel": "derived_from", "href": ".../S2_20250114" },
    { "rel": "derived_from", "href": ".../S1_20250302" },
    { "rel": "derived_from", "href": ".../S2_20250719" }
  ]
}
```

ASSET-LEVEL `EMB:\*` IS PROPOSED. `DERIVED\_FROM` IS ALREADY IN STAC CORE.

03 **Search the embeddings**

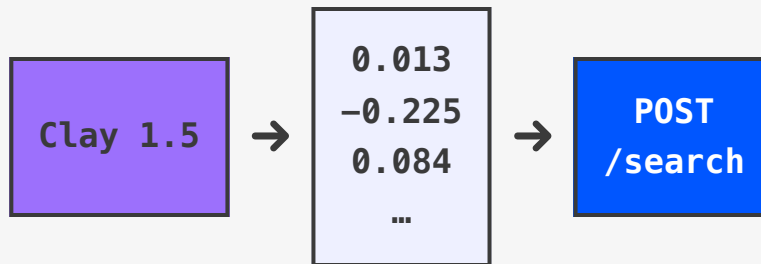
A STAC API extension

Pick an example



NAIP · Alabama · 1,024 × 1,024 pixels

Turn it into a vector



Run the query chip through the **same model** used for the stored embeddings.

The API does not need to understand the image or run the model. It receives the resulting vector.

An example request

POST /search

```
{
  "collections": ["sentinel-2-l2a"],
  "bbox": [132.3, 34.2, 132.6, 34.5],
  "datetime": "2023-06-01/2023-09-30",
  "filter": { "op": "<", "args": [{"property": "eo:cloud_cover"}, 10] },
  "embedding": {
    "asset": "embeddings-clay",
    "vector": [0.013, -0.225, 0.084, 0.019],
    "metric": "cosine",
    "limit": 20
  }
}
```

- The normal STAC fields select the candidate Items.
- `asset` selects the embedding product.
- `vector` is the query; it must match the asset's dimensions.
- `metric` and `limit` control the ranking.

The response is a FeatureCollection

```
{
  "type": "FeatureCollection",
  "features": [{
    "type": "Feature",
    "id": "chip-903",
    "geometry": { "type": "Polygon", "coordinates": [ "..."] },
    "properties": {
      "score": 0.87,
      "item": "S2B_53SNU_20230818",
      "asset": "embeddings-clay"
    },
    "links": [{ "rel": "item", "href": ".../items/S2B_53SNU_20230818" }]
  }]
}
```

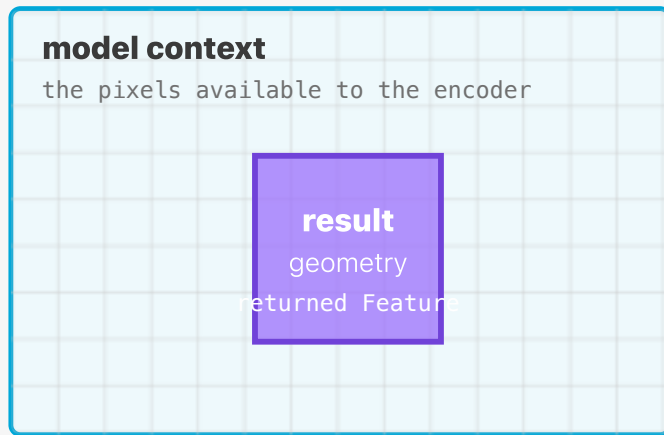
The match is a Feature. It links back to the STAC Item that provides context and source imagery.

Embedding footprint and model context

Suppose the search result is one **10 m pixel**.

The model may have produced that vector from a **2.56 km image chip** around it.

- `geometry` is the footprint the embedding represents
- `context_geometry` is the area the model looked at



For chip embeddings, these may be the same. For pixel or patch embeddings, they may be very different.

IF THEY ARE DIFFERENT, THE RESPONSE SHOULD INCLUDE BOTH.

Use case: find solar farms

Query	vector from one known solar-farm chip
--------------	---------------------------------------

STAC filter	California · 2023 · cloud cover < 10%
--------------------	---------------------------------------

Files searched	embedding assets on the matching Items
-----------------------	--

Response	ranked chip Features with Item links
-----------------	--------------------------------------

This is a proposed query, not a result from a running STAC API.

Use case: find change through time

Select two embedding Items for the same place and different periods

Join rows with the same geometry or grid identifier

Rank largest embedding distance first

Inspect follow both Item links back to the imagery

Annual embeddings can show annual change. They cannot show an event that appeared and disappeared within the year.

Use case: compare models

	Clay	TerraMind
Source Item	the same Sentinel-2 Item	the same Sentinel-2 Item
Asset	embeddings-clay	embeddings-terramind
Dimensions	1,024	768
Search	the same query place	the same query place
Output	one ranked FeatureCollection	one ranked FeatureCollection

Now the comparison keeps the source imagery fixed while the model changes.

04 Counterarguments

Reasons not to do this

STAC duplicates metadata already in Parquet

- Put enough metadata in the Parquet footer for the file to travel on its own.
- Repeat searchable metadata in STAC so clients do not have to open every file.
- Use checksums and processing versions to catch drift between the two.

The cost is duplication and the risk that the two copies drift. We still need both the catalog and the self-describing file.

A central vector service must hold everything

- One Sentinel-2 scene produces about **1,764** chip embeddings.
- One million scenes produce **1.764 billion vectors**.
- A global index has to store, update, and serve all of them.
- STAC search cuts that down to the relevant Items and Parquet files first.
- An index can still accelerate common searches, but it does not have to cover the whole archive.

The API should work whether the server uses an index or reads the selected Parquet files.

Some source data cannot be published

Closed or restricted source data makes full provenance impossible.

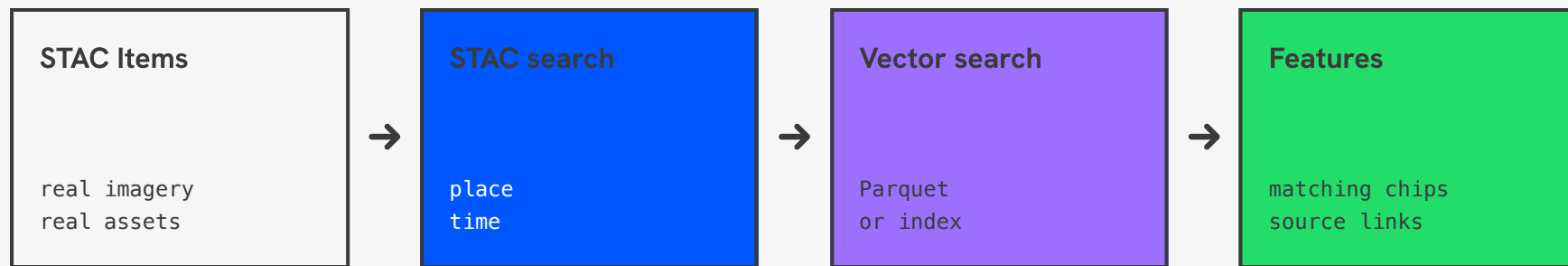
The Item should say what is known:

- source datasets and date range
- model and processing version
- missing or inaccessible inputs
- licences and access restrictions
- a reproducibility or provenance status

Record what is known so users can decide what they are willing to use it for.

I'm working on a demo

Nobody distributes embeddings this way yet.



I'm giving this talk again at the **Cloud-Native Geospatial Forum**. I'll bring it to CNG.

**Do not separate the summary
from the evidence .**

Thank you