FLUX WIKI INTELLIGENCE

LAYER OVERVIEW FOR FLUX INTELLIGENCE — LINKS TO SUBDOCUMENTS ON EMBEDDINGS, KEEPER MODEL, METADATA ENRICHMENT, AUTONOMOUS SEQUENCING, AND TRAINING DATA.

flux.dantesisofo.com/wiki/intelligence/

FLUX_WIKI_v2.0

JUNE 2026

FLUX DOCUMENTATION SYSTEM Layer 5 — INTELLIGENCE | intelligence
flux.dantesisofo.com/wiki/intelligence/

INTELLIGENCE

The archive becomes computationally alive.

Not storage. Not retrieval. Not backup.

Memory.

1. THE COMPUTATIONAL ARCHIVE

1.1 What It Means

A conventional archive is a file system.

Files have names. Files have dates. Files live in folders. You find them by remembering where you put them.

The FLUX intelligence layer replaces this logic with something different.

Every photograph becomes a vector. Every vector encodes the visual content of that photograph as a point in high-dimensional space. Similar photographs cluster together. Dissimilar photographs drift apart.

The archive stops being organized by filename.

It becomes organized by meaning.

1.2 Why This Matters

The FLUX corpus currently holds approximately 400,000 photographs, made over years of continuous daily practice.

No human being can hold 400,000 photographs in memory. No browsing interface is adequate at that scale. The conventional approach — scroll, filter, search by date, remember the folder — fails at this volume.

The intelligence layer makes the corpus navigable at its actual scale.

It also makes the corpus generative: the archive can surface patterns, motifs, and relationships that no photographer would discover through manual browsing.

1.3 The Principle

```
files → relationships
storage → memory
retrieval → understanding
```

The archive stops being a container.

It becomes a thinking machine pointed at the photographer's own visual history.

2. KEEPER SYSTEMS

2.1 The Existing Keeper Archive

Before the intelligence layer existed, a keeper archive was being maintained manually.

Approximately 15,000 images have been selected from the full corpus and preserved in a dedicated keeper folder:

```
/FLUX_ARCHIVE/KEEPERS/
    /2022/
        /2022-01/
            /2022-01-01/
                2022-01-01_13-22-44_DanteSisofo_R0001234.JPG
```

These images represent:

- the photographer's actual taste under real conditions
- selections made session by session, across years
- the positive class for training any future model
- visual truth — not what seemed good in theory, but what was kept

The keeper archive is not a highlights reel. It is a dataset.

2.2 Two Simultaneous Representations

The keeper archive must exist in two forms at once.

Form 1 — Human browsing layer:

```
/FLUX_ARCHIVE/KEEPERS/
```

Purpose: direct access for projects, books, prints, emotional interaction. The photographer browses here.

Form 2 — Metadata truth layer:

Inside the full corpus (`/FLUX_ARCHIVE/ORIGINALS/`), every file that corresponds to a keeper carries:

```
keeper = TRUE
keeper_score = 1.0
```

This transforms the full corpus into a labeled machine-learning dataset.

The keeper archive is the positive class. The remaining 385,000 frames are context — implicit negatives, near-misses, rejected frames that define the edges of the selection threshold.

2.3 Keeper Matching

The keeper archive and the full corpus exist as separate folder structures. The intelligence system must connect them.

Matching strategy:

```
1. EXIF timestamps — primary match signal
2. Original Ricoh filenames — secondary match
```

3. SHA-256 hashes — definitive confirmation
4. Image similarity fallback — for files where metadata was altered

Result: every file in the full corpus receives a keeper flag or remains unflagged. The flag is permanent metadata — embedded in the SQLite database, surfaced in the manifest, available to every downstream system.

2.4 What the Keeper System Learns

The keeper archive teaches the system:

- what gets selected
- what gets rejected
- which session conditions produce more keepers
- which locations produce more keepers
- which times of day, lighting conditions, weather states produce more keepers
- what the photographer's visual threshold actually is

This is not aesthetic theory. This is empirical data extracted from a decade of practice.

---

3. THE KEEPER MODEL

3.1 The Core Question

The system should eventually answer:

"What would Dante probably keep?"

Not:

"What is objectively great photography?"

This distinction is critical.

The model is not learning universal aesthetics. It is not learning what the photography establishment values. It is not learning what performs well on Instagram.

It is learning:

- personal visual taste
- selection philosophy
- aesthetic thresholds
- recurring motifs
- sequencing tendencies

The keeper archive is the training dataset. The full corpus is the test set. The model is a mirror.

3.2 What the Model Is Not

The keeper model is not:

— an aesthetic judge
— a replacement for the photographer's eye
— a system for selecting "objectively great" photographs
— an AI that can see better than a human

The keeper model is:

— a time-saving filter applied to the full corpus
— a way to surface probable keepers from a session of 200 frames without manual review of each one
— a pattern recognition system trained on demonstrated taste
— a suggestion engine — not a decision engine

The photographer still selects. The model ranks.

3.3 Training Data Requirements

Effective training requires:

```
~15,000 positive examples (existing keeper archive)
~385,000 negative/context examples (full corpus)
Per-session metadata: location, time, weather, conditions
Visual embeddings: one vector per photograph
```

The corpus migration (see Bootstrap, Phase 2) must be complete before training begins. The keeper metadata must be attached to the full corpus (Phase 3). Embeddings must be generated (Phase 5). Only then does meaningful model training become possible (Phase 6).

---

4. METADATA INTELLIGENCE

4.1 The Database

A metadata database becomes the intelligence layer's join point.

Every photograph in the corpus receives a full metadata record. The record links the archive file to its embedding vector, its keeper status, its issue assignment, its verification hash, and all associated EXIF data.

Initial implementation: SQLite
Long-term: PostgreSQL if scale demands it

4.2 Full Metadata Schema

```
filename:          2023-01-15_13-22-44_DanteSisofo_R0001234.JPG
sha256:            a3f7...
keeper:            TRUE
keeper_score:      1.0
issue:             FLUX_VOL_019
camera:            Ricoh GR III
focal_length:      18.3mm
aperture:          f/2.8
shutter:           1/500
```

```
iso:               400
location:          Philadelphia
geocoded_address:  South Street, Philadelphia, PA
weather:           fog
time_of_day:       morning
tags:              umbrella, silhouette, layered, shadow
embedding_ref:     vector_8821
motif_cluster:     cluster_umbrella_fog
```

4.3 Example Record

```
{
  "filename": "2023-01-15_13-22-44_DanteSisofo_R0001234.JPG",
  "sha256": "a3f7b92d4e...",
  "keeper": true,
  "keeper_score": 1.0,
  "issue": "FLUX_019",
  "camera": "Ricoh GR III",
  "focal_length": "18.3mm",
  "aperture": "f/2.8",
  "shutter": "1/500",
  "iso": 400,
  "lat": 39.9432,
  "lon": -75.1598,
  "geocoded_address": "South Street, Philadelphia, PA",
  "weather": "fog",
  "time_of_day": "morning",
  "tags": ["umbrella", "silhouette", "layered", "shadow"],
  "embedding_ref": "vector_8821",
  "motif_cluster": "cluster_umbrella_fog"
}
```

The metadata database is the join layer between the archive (files) and the intelligence layer (embeddings, keeper scores, motif clusters).

---

5. EMBEDDINGS LAYER

5.1 What Embeddings Are

A visual embedding is a mathematical representation of a photograph's content as a high-dimensional vector.

Two photographs that look similar will have similar vectors. Two photographs that are visually unrelated will have distant vectors.

The embedding captures:

- composition
- tonal distribution
- spatial relationships
- subject matter
- visual texture
- light quality

It does not capture filenames, dates, or folders.

5.2 What Embeddings Enable

Once every photograph has a vector representation, the archive becomes queryable by visual similarity rather than filename or date.

```
show all umbrellas in fog
show visual echoes of Rome
show all silhouettes from winter 2022
show images visually similar to this one
show recurring gestures across six years
show evolution of layering over time
find all photographs where a person is waiting
find all photographs with wet pavement reflections
find all photographs made within one block of 40th and Walnut
```

These queries are impossible in a conventional file system.

They become trivial in a vector database.

5.3 The Transformation

Embeddings transform the archive from:

```
files
```

into:

```
relationships
```

The archive stops being storage.

It becomes memory.

A photographer who has shot for ten years cannot remember what they photographed in January 2022. A vector database can retrieve the ten frames from that month that most closely resemble a specific frame from May 2026.

The archive develops the ability to see itself.

5.4 Technical Implementation

```
Model:          CLIP or similar vision encoder
Input:          full-resolution JPEG
Output:         high-dimensional float vector (512 or 1024 dimensions)
Storage:        vector database (Qdrant, Weaviate, or Chroma)
Index:          HNSW (approximate nearest-neighbor)
Query time:     <100ms for nearest-neighbor search across 400,000 vectors
```

The embedding generation pass runs once on the full corpus. After that, new ingest photographs receive embeddings in real time as part of the autonomous pipeline.

---

6. SEMANTIC SEARCH

6.1 What Becomes Possible

Semantic search over the FLUX corpus means querying the archive in natural language and receiving relevant photographs.

Examples:

```
"photographs made in rain"
→ retrieves all frames with wet conditions, regardless of metadata tags

"people waiting on platforms"
→ retrieves transit waiting photographs without any manual tagging

"compressed urban space, high contrast"
→ retrieves photographs matching that visual description

"similar to this keeper from 2023"
→ retrieves nearest-neighbor frames from the full corpus
```

This is not keyword search. Keywords require manual tagging. Semantic search requires only the visual content of the photograph itself.

6.2 The Interface

The semantic search interface is the practical face of the intelligence layer.

A simple text box. A query in plain language. The archive responds.

The photographer types: `umbrellas in fog, Philadelphia`

The archive returns the twelve most visually similar frames from the corpus, regardless of when they were made or what session they belong to.

This is what the corpus being computationally alive means in practice.

---

7. AUTONOMOUS SEQUENCING

7.1 Issue Sequencing Experiments

The intelligence layer enables experiments in AI-assisted issue sequencing.

Standard FLUX sequencing is strictly chronological: the order photographs were made is the order they appear in the issue. This is protocol. It does not change.

But the intelligence layer can propose alternative experiments:

```
thematic clustering:    group related visual motifs across a session
visual rhythm:          order frames by visual similarity to produce smooth transitions
anti-clustering:        deliberately interrupt visual repetition with contrast frames
seasonal sequences:     pull frames across years, unified by visual or atmospheric quality
```

These experiments are not replacements for the canonical chronological sequence. They are parallel outputs — draft sequences generated by the system and reviewed by the photographer.

The protocol sequence is always produced. The experimental sequence is an additional layer.

7.2 What the System Cannot Do

The intelligence layer cannot:

- replace the photographer's judgment about the final sequence
- understand narrative in the editorial sense
- feel the session
- know what mattered on that walk

The intelligence layer can:

- identify visual relationships the photographer might miss
- surface non-obvious groupings from a large corpus
- generate sequence drafts that the photographer refines
- rank frames by predicted keeper status

The photographer is always the final editor.

---

8. IMPLEMENTATION ROADMAP

8.1 Phase 3 — Keeper Integration

12. Scan existing keeper archive (~15,000 frames)
13. Match keepers against full corpus (EXIF timestamp, filename, SHA-256, similarity fallback)
14. Flag keepers in metadata database
15. Build labeled keeper dataset (~15,000 positive examples)

Prerequisites: NAS infrastructure, corpus migration (Phases 1–2 complete).

8.2 Phase 4 — Metadata Layer

16. Initialize SQLite database
17. Generate SHA-256 hashes for all corpus frames
18. Generate manifests per session and issue
19. Generate thumbnails for full corpus
20. Build complete metadata schema
21. Reverse geocode all GPS coordinates
22. Auto-detect time-of-day, daylight conditions

8.3 Phase 5 — Embeddings

23. Generate visual embeddings for full corpus (~400,000 frames)
24. Build vector database (Qdrant or equivalent)
25. Build similarity search interface
26. Build motif clustering
27. Build semantic archive search interface

This phase converts the archive from a file system into a searchable visual memory.

8.4 Phase 6 — Keeper Intelligence

28. Train keeper ranking model on labeled dataset
29. Build keeper scoring for new ingest frames
30. Build auto-selection experiments
31. Build automated issue sequencing drafts

32. Evaluate model performance against manual selections

The model is ready for training when Phases 3–5 are complete. Training is a one-time operation. Inference runs on every new ingest photograph.

8.5 Phase 7 — Autonomous FLUX (Full Vision)

33. Build watch-folder ingest daemon
34. Build fully autonomous issue generation
35. Build automatic publishing
36. Build blockchain verification pipeline
37. Build decentralized archival persistence layer

The intelligence layer powers Phase 7. Autonomous FLUX requires a working keeper model, working embeddings, and a working autonomous ingest pipeline. The intelligence layer is the prerequisite.

---

9. RELATIONSHIP TO OTHER LAYERS

The intelligence layer sits between the infrastructure and the publishing output.

```
ARCHIVE (files)
    ↓
INFRASTRUCTURE (NAS, Mac mini, ingest pipeline)
    ↓
INTELLIGENCE (embeddings, keeper model, semantic search)
    ↓
GENERATOR (issue PDF, contact sheet, manifest)
    ↓
ARCHIVE (published)
    ↓
PRESERVATION (verification, blockchain, redundancy)
```

The intelligence layer makes the archive self-aware. Without it, the archive is storage. With it, the archive becomes an active system that understands its own contents.

FINAL PRINCIPLE

The archive is not memory by default.

Memory requires organization. Memory requires relationships. Memory requires the ability to find what was forgotten.

The intelligence layer gives the archive the ability to remember itself.

```
400,000 photographs.
Ten years of walking.
A system that can find the echoes.
```

That is the intelligence layer.

SEE ALSO

Document	Layer	Relationship
EMBEDDINGS	Layer 5 — Intelligence	Technical specification for visual embedding generation, CLIP model, vector storage, and query interface
KEEPER MODEL	Layer 5 — Intelligence	Personal taste model specification — what it learns, training data, architecture, and limits
METADATA ENRICHMENT	Layer 5 — Intelligence	Full database schema, EXIF extraction, geocoding, weather enrichment, and manifest generation
AUTONOMOUS SEQUENCING	Layer 5 — Intelligence	Chronological default, AI sequencing experiments, and the limits of algorithmic sequence
TRAINING DATA	Layer 5 — Intelligence	Keeper archive as ML dataset — matching strategy, class imbalance, near-misses, dataset construction
BOOTSTRAP	Layer 4 — Infrastructure	Hardware architecture and phased implementation plan for the intelligence layer
PRESERVATION	Layer 7 — Preservation	Cryptographic verification that extends beyond the intelligence layer
ROADMAP	Layer 8 — Roadmap	Phased development timeline for AI selection and autonomous publishing
PROTOCOL	Layer 2 — Protocol	The chronological sequencing principle the keeper model works within
ARCHIVE	Layer 2 — Protocol	The digital archive structure the intelligence layer extends
GENERATOR	Layer 4 — Infrastructure	The issue generator that intelligence layer outputs feed into

FLUX_WIKI_v2.0 — flux.dantesisofo.com/wiki/intelligence/