

HOPE Architecture Proof-of-Concept: Dual-Speed Continual Learning for Aviation Video Classification

Using V-JEPA2 and Gemini as Slow/Fast Memory Systems

Frank Morales Aguilera

Sovereign Machine Lab (SOMALA)

Montreal, Quebec, Canada

frank.morales@sovereignml.ai

ORCID: 0009-0003-9528-0745

Code: github.com/frank-morales2020/MLxDL/blob/main/HOPE_VJEPA_GEMINI3PRO_DEMO.ipynb

May 2026 | Proof-of-Concept | Not Affiliated with Google DeepMind

Abstract

We present a proof-of-concept (PoC) implementation of the HOPE continual learning architecture, introduced in Google DeepMind's Nested Learning paper (NeurIPS 2025), applied to the domain of aviation video classification using the TartanAviation dataset. The PoC instantiates the dual-speed memory hypothesis of Nested Learning using two production-grade AI systems: Facebook's V-JEPA2 as the slow, weight-updating long-term memory component, and Google's Gemini API as the fast, context-updating short-term reasoning component. A HOPE controller orchestrates adaptation decisions by computing a feedback novelty score: high-error (novel) situations trigger backpropagation through V-JEPA classification and latent dynamics heads, while low-error (familiar) situations update only Gemini's recurrent context state. The PoC demonstrates that the HOPE architecture's core behavioral claims — surprise-gated dual-speed adaptation — can be instantiated with real production models on a real-world video task. We discuss current limitations, design decisions, and the conditions under which Gemini 4's expected persistent memory capabilities could enable a full experimental validation of this approach.

1. Introduction

Large language models and video understanding systems have achieved remarkable capabilities, but share a fundamental constraint: they do not continue learning after training. Once deployed, these models are frozen — they cannot incorporate new knowledge without full retraining, and any attempt to fine-tune incrementally typically suffers from catastrophic forgetting, where new learning overwrites

previously acquired knowledge.

Google DeepMind's Nested Learning paper (Behrouz et al., NeurIPS 2025) proposes a theoretical framework and proof-of-concept architecture — HOPE — that addresses this limitation. The key insight is that memory should be treated as a *continuum* of modules updating at different rates, mirroring the neuroplasticity of biological brains. Fast-updating modules handle familiar, low-surprise situations; slow-updating modules, triggered by high surprise, consolidate genuinely new information into long-term weights.

This paper presents a PoC implementation of the HOPE architecture applied to aviation video classification. Rather than building toy networks, we instantiate the architecture using two production AI systems: V-JEPA2 (Meta AI) as the slow memory system and the Gemini API as the fast memory system. The goal is to demonstrate that the HOPE behavioral loop — surprise-gated switching between fast and slow adaptation — is architecturally coherent and runnable with real models on a real-world task. This is a PoC, not a full experimental validation; the latter is left as future work for Google or the broader research community.

2. Background and Related Work

2.1 Nested Learning and the HOPE Architecture

Behrouz et al. (2025) propose Nested Learning as a paradigm in which a single ML model is treated as a system of interconnected, multi-level learning problems optimized simultaneously. The continuum memory system (CMS) generalizes the traditional short/long-term memory dichotomy into a spectrum of modules, each with its own update rate. HOPE is the paper's proof-of-concept architecture, built as a self-modifying variant of the Titans architecture. Titans modules prioritize memories based on surprise (analogous to prediction error), making them a natural fit for novelty-gated adaptation.

2.2 V-JEPA2

V-JEPA2 (Meta AI, 2024) is a self-supervised video model trained with a joint-embedding predictive architecture. The model learns rich spatiotemporal representations by predicting masked regions in a learned latent space rather than pixel space. We use the *vjepa2-vitg-fpc64-256* checkpoint (ViT-Giant backbone, 64 frames per clip, 256px resolution), which produces 1408-dimensional feature vectors. In our PoC, V-JEPA2 serves as the frozen feature extractor; learnable classification and dynamics heads sit on top and receive gradient updates during slow adaptation.

2.3 Gemini API as a Fast Reasoning Component

Gemini (Google DeepMind) models provide fast, language-grounded reasoning over structured inputs. In our PoC, Gemini receives a natural language summary of V-JEPA's classification output and a task goal, then generates a concise action string and rationale. Its *recurrent_state* dictionary simulates a persistent context window, updated on every low-error cycle. This design anticipates Gemini 4's expected native persistent memory, which would make this state genuinely persistent across sessions.

3. PoC Architecture

The PoC comprises three primary components coordinated by the HOPE controller. Figure 1 shows the end-to-end cycle.

Component	Class	Role	Update Mechanism
V-JEPA2 + Heads	IntegratedVJEPAModel	Slow memory / visual perception	Backprop on classifier + dynamics heads
Gemini API	Gemini3ProModel	Fast memory / language reasoning	Context dict write (no weight update)
HOPE Controller	HOPEController	Orchestrator / gating	Error threshold comparison

Table 1. PoC component overview.

3.1 IntegratedVJEPAModel (Slow System)

This class wraps V-JEPA2 and three learnable heads. On initialization, it attempts to load the V-JEPA2 checkpoint from HuggingFace. If loading fails (e.g., in resource-constrained environments), a flag disables real extraction and a seeded fallback tensor is used instead. The fallback is seeded with `GLOBAL_FEATURE_SEED = 99` before creation and the seed is reset immediately after, ensuring the tensor is deterministic across runs while not contaminating other random operations.

The three learnable heads are:

Head	Architecture	Purpose
ClassifierHead	Linear(1408 → num_classes)	Predict one of 10 aviation flight phases
LatentProjector	MLP: 1408 → 64 → 16 (ReLU)	Compress V-JEPA features to 16-dim latent
LatentDynamicsPredictor	MLP: (16+8) → 64 → 16 (ReLU)	Predict next latent given action

Table 2. Learnable heads on top of frozen V-JEPA2 features.

Slow adaptation is implemented in `update_long_term_memory()`, which combines two losses: `CrossEntropyLoss` on the classification logits against the ground truth flight phase, and `MSELoss` between the predicted next latent state and a placeholder target (randomized in the PoC, intended to be replaced with a real next-state signal in a full experiment). Both losses are summed and backpropagated through all three heads via a shared Adam optimizer (`lr=0.0001`).

3.2 Gemini3ProModel (Fast System)

This class wraps the Gemini API client. On each cycle it receives the V-JEPA output dictionary and a natural language user goal, constructs a prompt combining both, and calls `client.models.generate_content()`. The prompt includes a system instruction, the current task, and a visual context summary derived from V-JEPA's predicted class label and confidence score.

The class maintains a `recurrent_state` dictionary with two fields: `context_window` (a list of past action strings) and `focus` (a short working-memory summary). On low-error cycles, the HOPE controller updates the focus field with the first 20 characters of the latest action — simulating lightweight fast adaptation without weight modification.

The PoC explicitly uses *gemini-3-pro-preview* as the model ID. This is a deliberate placeholder: the code acknowledges the API call is expected to fail and handles both *APIError* and generic exceptions gracefully, falling back to a simulation string. This design choice anticipates Gemini 4's persistent memory capabilities, which would make the fast adaptation genuinely stateful.

3.3 HOPEController (Orchestrator)

The controller's *run_hope_cycle()* method implements the full HOPE loop:

1. Call V-JEPA to classify the visual input and extract latent state.
2. Pass V-JEPA output to Gemini to generate an action and action tensor.
3. Compare predicted class index against ground truth to determine correctness.
4. Sample a feedback error score: $U(0.0, 0.4)$ if correct, $U(0.7, 1.0)$ if incorrect.
5. If error > 0.6 : trigger slow adaptation (V-JEPA backprop).
6. Else: trigger fast adaptation (Gemini context update only).

The error threshold of 0.6 is the gating decision boundary, directly implementing the Nested Learning principle that surprise (high prediction error) routes learning to the slow, weight-updating system.

4. Domain: Aviation Flight Phase Classification

The PoC targets classification of aviation video into 10 meaningful flight phases, derived from the TartanAviation dataset. These labels are defined as a global configuration and used throughout all components, ensuring consistent semantics between V-JEPA's classifier output and Gemini's natural language reasoning.

Index	Class Label	Phase Description
0	Airplane landing	Final approach and touchdown sequence
1	Airplane takeoff	Rotation and initial climb
2	Airport ground operations	Taxiing, gate movements, pushback
3	In-flight cruise	Level flight at cruise altitude
4	Emergency landing	Non-standard approach under emergency
5	Pre-flight check / maintenance	Ground inspection and maintenance operations
6	En-route cruise	Extended cruise segment
7	Climb phase	Post-takeoff climb to cruise altitude
8	Descent phase	Top-of-descent to approach initiation
9	Holding pattern	Circular holding stack operations

Table 3. Aviation flight phase class labels used in the PoC.

5. Demonstration Scenarios

The PoC executes two controlled scenarios to demonstrate the dual-speed adaptation behavior. In both cases, the classifier weights are manually set before each scenario to produce a deterministic

prediction — a deliberate design choice that makes the demo's behavior reproducible and interpretable.

5.1 Scenario 1: Familiar Situation (Fast Adaptation)

The classifier weights are set such that the stable fallback feature vector activates class 0 (airplane landing) with overwhelming confidence. The ground truth is also class 0. This produces a correct prediction, yielding a low feedback error score (sampled from $U(0.0, 0.4)$). Since the error does not exceed the 0.6 threshold, only Gemini's recurrent context is updated — no backpropagation occurs.

Expected behavior: HOPE identifies the situation as familiar and routes to fast adaptation. The slow V-JEPA system is left unchanged.

5.2 Scenario 2: Novel / Anomalous Situation (Slow Adaptation)

The classifier weights are set such that the same feature vector activates class 3 (in-flight cruise). The ground truth is class 4 (emergency landing). This produces an incorrect prediction, yielding a high feedback error score (sampled from $U(0.7, 1.0)$). Since the error exceeds 0.6, the HOPE controller triggers V-JEPA's `update_long_term_memory()` — backpropagating through both the classifier and dynamics heads.

Expected behavior: HOPE identifies the situation as novel/anomalous and routes to slow adaptation. The V-JEPA heads are updated via gradient descent, representing consolidation of the new experience into long-term memory.

Property	Scenario 1: Familiar	Scenario 2: Novel
Predicted class	0 — airplane landing	3 — in-flight cruise
Ground truth class	0 — airplane landing	4 — emergency landing
Prediction correct	Yes	No
Error score range	$U(0.0, 0.4)$	$U(0.7, 1.0)$
Error exceeds 0.6	No	Yes
Adaptation triggered	Fast (Gemini context)	Slow (V-JEPA backprop)
Weight update	None	ClassifierHead + DynamicsPredictor
Random seed	42	101

Table 4. Comparison of the two demonstration scenarios.

6. Key Design Decisions

6.1 Stable Fallback Feature Tensor

V-JEPA2's real video I/O is likely to fail in shared Colab environments due to resource constraints and file system access. Rather than aborting, the PoC creates a deterministic fallback tensor at startup, seeded with `GLOBAL_FEATURE_SEED = 99`. This ensures all downstream components — classification, latent projection, dynamics prediction — receive a consistent input across runs, making the demo's behavior reproducible regardless of environment. The seed is immediately reset after tensor

creation to avoid contaminating other stochastic operations.

6.2 Classifier Weight Manipulation

Before each scenario, the classifier's weight matrix is zeroed and a single row is set to $1000 \times \text{STABLE_FALLBACK_FEATURE}$. This guarantees that the stable fallback feature activates exactly the intended class with overwhelming confidence, making the scenario's prediction deterministic. This is a PoC convenience — in a full experiment, the classifier would be trained normally and predictions would emerge from learned representations.

6.3 Graceful API Degradation

The Gemini API call is wrapped in a try/except block handling both *APIError* (for model-specific failures, such as the preview model ID not yet being publicly available) and generic exceptions. In both cases, a simulation string is returned that preserves the structure of a real response. This allows the full HOPE cycle to execute and print meaningful output even without a working API key or model access.

6.4 Placeholder Dynamics Target

The latent dynamics predictor is trained against *torch.rand_like(predicted_next_latent)* — a random target. This is a known limitation of the PoC. In a full experiment, the target would be the actual next-state latent, extracted from the subsequent video clip. The random target means the dynamics loss provides gradient signal but not meaningful learning; its presence demonstrates the training infrastructure is correct.

7. Correspondence to the Nested Learning Paper

Table 5 maps the theoretical constructs from Behrouz et al. (2025) to their PoC realizations in this implementation.

Paper Concept	PoC Realization	Fidelity
Continuum memory system	V-JEPA (slow) + Gemini (fast)	Partial — 2 levels, not N
Surprise-driven gating	feedback_error > 0.6 threshold	Behavioral match
Slow weight update	V-JEPA head backprop	Functional
Fast context update	Gemini recurrent_state dict	Simulated (no weight change)
Self-modifying architecture	HOPEController.run_hope_cycle()	Structural approximation
Titans memory module	V-JEPA + LatentDynamicsPredictor	Partial
Meta-learning (learn to learn)	Not implemented in PoC	Out of scope
Multi-level optimization	2-level only (binary switch)	Simplified

Table 5. Mapping between Nested Learning theoretical constructs and PoC implementations.

8. Limitations

As a deliberate PoC, several simplifications are present that would need to be addressed in a full experimental evaluation:

Binary adaptation: The PoC uses a single threshold (0.6) to switch between fast and slow adaptation. The Nested Learning paper describes a true continuum with N levels of update rates. A faithful implementation would have multiple intermediate tiers (e.g., a LoRA adapter updating at medium frequency).

Simulated Gemini state: Gemini's recurrent_state is a Python dictionary, not an actual model weight update. The fast adaptation path does not modify any parameters. Gemini 4's expected native persistent memory would resolve this.

Random feedback error: The error score is sampled from a uniform distribution conditioned on correctness, not derived from a principled surprise metric such as KL divergence between predicted and actual latent distributions.

Random dynamics target: The LatentDynamicsPredictor trains against a random tensor rather than a real next-state latent. This provides gradient flow but not meaningful world model learning.

Single video, single cycle: The PoC runs two scenarios on one video. A full experiment would require a multi-task sequence with explicit measurement of backward transfer (retention of earlier tasks after learning new ones).

No baseline comparison: Without baseline methods (e.g., EWC, experience replay, naive fine-tuning), there is no claim that HOPE outperforms alternatives. This is intentional — the PoC does not claim experimental superiority.

9. Future Work and the Gemini 4 Hypothesis

The most significant near-term opportunity for this PoC to evolve into a full experiment is the expected release of Gemini 4 (anticipated late 2026 / early 2027). Demis Hassabis confirmed in January 2026 that Google's team is actively developing Gemini 4, with expected features including native persistent long-term memory across sessions and full Project Astra multimodal integration.

With Gemini 4's persistent memory, the fast adaptation path in this architecture would become genuinely stateful: Gemini's recurrent_state would persist across API calls without explicit serialization, making the fast memory system a true first-class learning component rather than a simulated one. Combined with V-JEPA3 or a successor with improved video I/O, the full HOPE loop could be validated experimentally.

Gap	Required Capability	When
Simulated Gemini state	Gemini 4 native persistent memory	Late 2026
V-JEPA I/O failure	V-JEPA3 or improved Colab GPU access	2026
Binary adaptation	Add LoRA middle tier	Now — engineering task
Random dynamics target	Real next-frame V-JEPA features	Now — data task
No forgetting measurement	Multi-task benchmark (TartanAviation full)	Now — experiment

Table 6. Gaps and required capabilities for a full experiment.

10. Conclusion

We have presented a proof-of-concept implementation of the HOPE continual learning architecture applied to aviation video classification. The PoC demonstrates that the core behavioral claim of the Nested Learning paper — that surprise-gated dual-speed adaptation can be implemented with real production models — is architecturally sound and runnable. V-JEPA2 serves as a credible slow memory system with learnable heads that can receive gradient updates on high-error episodes; Gemini provides fast context-level reasoning that updates without weight modification on familiar episodes.

The PoC does not claim to validate HOPE experimentally — that is the role of a full experiment with proper baselines, multi-task sequences, and forgetting metrics. That experiment is best conducted by Google DeepMind or the broader research community with the computational resources and dataset access it requires.

The central hypothesis is that Gemini 4's expected persistent memory capabilities will transform this PoC's simulated fast adaptation into a genuine, stateful learning component — at which point the HOPE architecture's full behavioral claims become empirically testable. This PoC establishes the architectural foundation for that future experiment.

References

- [1] Morales Aguilera, F. (2025). *HOPE Architecture Proof-of-Concept: Dual-Speed Continual Learning for Aviation Video Classification using V-JEPA2 and Gemini*. Sovereign Machine Lab (SOMALA), Montreal, Canada. GitHub: github.com/frank-morales2020/MLxDL/blob/main/HOPE_VJEPA_GEMINI3PRO_DEMO.ipynb. ORCID: 0009-0003-9528-0745.
- [2] Behrouz, A., Razaviyayn, M., Zhong, P., & Mirrokni, V. (2025). *Nested Learning: The Illusion of Deep Learning Architectures*. NeurIPS 2025. arXiv:2512.24695.
- [3] Meta AI. (2024). *V-JEPA2: Self-Supervised Video Feature Learning with Joint-Embedding Predictive Architectures*. HuggingFace: [facebook/vjepa2-vitg-fpc64-256](https://huggingface.co/facebook/vjepa2-vitg-fpc64-256).
- [4] Google DeepMind. (2025). *Gemini: A Family of Highly Capable Multimodal Models*. Technical Report, Google DeepMind.
- [5] Google Research. (2025). *Introducing Nested Learning: A New ML Paradigm for Continual Learning*. Google Research Blog. research.google/blog/
- [6] Wang, W., et al. (2023). *TartanAviation: Image, Speech, and ADS-B Trajectory Datasets for Terminal Airspace Operations*. arXiv:2309.01750.
- [7] Gu, A., & Dao, T. (2023). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. arXiv:2312.00752.
- [8] Hassabis, D. (2026, January). Statement on Gemini 4 development focus. Google DeepMind.
- [9] Paszke, A., et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. NeurIPS 2019.
- [10] Kirkpatrick, J., et al. (2017). *Overcoming Catastrophic Forgetting in Neural Networks*. PNAS, 114(13), 3521-3526.
- [11] Wolf, T., et al. (2020). *Transformers: State-of-the-Art Natural Language Processing*. EMNLP 2020. HuggingFace Transformers Library.