So after accepting this challenge I threw them into a notebooklm and infranodus And started a chat with Gemini Then asking some of the default questions that were in notebook because no idea about rel and no idea the current bottom necks in relationships between language models and robotics To the point it was almost hard to read to be honest but once I started to understand the four papers on what their strengths and their downfalls were, I was able to see the gaps. The first realization that I had between all of them was that action language models and vision language models have a disconnect while working together. Like two brains trying to coordinate and understand each other's needs in different languages. Even then there lies the issue of a need to be able to predict and remember the world in real time.

The Point World 3D World model was a system that's designed to predict environment dynamics without the need for permutation matrices 3D space represented as 3D point flows instead of trying to and points which would require a permutation matrix. Originally that's what the core war was for, Like brute forcing prediction I foreseeing possibilities. Actually it's more like a data engine generating samples of warriors who paddle it out to create a champion which gets stored on a map.

The way I originally presented my questions Realigned Gemini's responses in a way that it thought that I wanted the DRQ for Prediction and that DRQ for prediction enabled several adv capabilities that go beyond simple strategy discovery.

OK, this MEMBOX, how is this remembering this stuff? It takes something called a topic loom that groups information to a single unit at storage time rather than trying to reconstruct the context later. Traditional memory systems structurally break when logically connected events are stored as separate fragments losing coherence. Denmark is able to link those boxes and to an event timeline allowing temporal continuity and enabling a global narrative integrity. Events can belong to multiple traces simultaneously and can do multi-hop reasoning on those non-linear assignments improving robustness via semantic pureness. Then it can compute the similarity against those stored boxes which allows it to predict relevant boxes across multiple levels of abstraction.

The day prior to this challenge I read a paper that came out with an algebraic representation theorem for linear GENEOS. It's basically the math for equivariant symmetries through the algebra and geometry of perception pairs in a compact linear way. Figured this could be a good baseline To connect division model and the action model together Through math. This would allow reducing parameter complexity noise robustness that improves the pointworld Model predictions and remain consistent regardless of the robot's orientation relative to a scene. Plus the non-expansive property of GENEO provides a guaranteed stability ensuring that small perturbations of the input point cloud don't escalate. It would also allow the DRQ predictive stability in which we'll talk about later. I also thought that the Neo would give MEMBOX some structure of what it remembers as far as what's important, Even though I found out later it doesn't really need that because it's a standalone system but it still can help.

After learning about the four papers more and the one that I added, I thought it might be a good idea to help those warriors create a better champion by adding a neural architecture search (NAS). I knew that it was computationally expensive and maybe not needed but, I felt it would be interesting. if it only used the symmetries from the GENEO to stay focused and enhance the warriors in the core war. I guess I just wanted to see a good fight. After more learning of the papers I realized how they were different and how the Red Queen algorithm kind of already did that well and wasn't really needed. The warriors themselves are not neural networks, they are assembly programs written in a low level language called Redcode. The real problem was VLMS or the bottlenecks because they are poor predictors of effective embodied control which means they can understand what they see really well but they don't really think about manipulating the objects in which they see. So even though Nas might evolve the Redcode Warriors themselves It still may significantly improve the surrogate predictive models that DRQ uses to bypass expensive simulations. This would shift the search code optimization to architectural optimization using evolutionary pressure. Since the mapping from code to performance is highly chaotic. This predictive architecture could further be improved by using Group Equivariant Non-Expansive Operators (GENEOs) to ensure the predictor is robust to small, non-functional code perturbations.

Recapping what I learned so far, these added papers bridged the semantic gap between VLMS and ALMS by providing structural guidance between their main objectives while addressing some of their limitations. With GENEOS as building blocks, a NAS can search for smaller more efficient networks that rely on equal variant components This reduces the computational complexity by requiring fewer parameters to achieve the same or better robustness in a massive unguided model for warriors to operate in complex multi session at first serial environments they needed a memory system that maintains global narrative integrity in which membox directly contributes to. NAS Could also utilize the hierarchical memory of MEMBOX to better equip them to perform the multihop reasoning required for adaptive strategies that reoccur across long sequences.Instead of a NAS searching for a "flat" memory connectivity, it can use the Topic Loom and Trace Weaver as fixed structural modules. The search can then focus on the most effective way to extract events and keywords within those boxes. I later found out NAS can also be used to optimize how the agent predicts the relevance of these different layers, allowing the agent to "hop" across the event-timeline traces constructed by the Weaver

I was still worried that NAS would be too computationally expensive and may just be added noise. However, the reduced perimeter complexity of GENEO Was designed to replace complex networks with smaller ones constructed from a limited number of equal variant components this allows for a drastic reduction and the number of parameters by maintaining high performance therefore the NAS would be training on a much smaller more efficient model. The digital Red Queen frame demonstrates that battle outcomes can be predicted statistically using linear probes on code embeddings. This would allow the NAS to use a surrogate model to prefilter architectures instead of running an expensive 80,000 timestamp simulation for every candidate. Membox uses only a fraction of the context tokens required by memory methods. By reducing the context length that the NAS-evaluated agents must process the computational overhead per evaluation is significantly lowered. The space of all linear GENEOs is proven to be convex and compact. In optimization, a convex search space is "smoother" because it lacks the deceptive local minima that usually make NAS difficult and computationally expensive. GENEOs are non-expansive, meaning they are guaranteed to be stable under input perturbations. This "smooths" the model's response to noise, ensuring that small changes in the architecture or input data do not lead to wildly divergent or "chaotic" outputs—a problem specifically noted in the DRQ warrior battle data.

So then where do we stand with all this now? What's still the problem of these papers that I was trying to combine and does that solve the problem?

“*1. Digital Red Queen (DRQ): Chaotic Mapping and Computational Cost**
The primary problem in DRQ is the extreme instability of its search space.
• Chaotic Genotype-to-Phenotype Mapping: Small changes in a warrior's assembly code can lead to drastic, unpredictable shifts in battle performance.
• Evaluation Bottleneck: Because of this chaos, determining a warrior's generality requires 317 separate 80,000-timestep simulations, which is computationally expensive.
• Predictive Limitation: Current statistical models only achieve an R2 of 0.461, indicating that performance is only "moderately predictable" from code alone.
2. VLM4VLA: The Vision Encoder Bottleneck
In the adaptation of Vision-Language Models for action, the sources identify the vision encoder as the absolute performance bottleneck.
• The Semantic Gap: Standard VLM pretraining on web-scale images captures high-level semantics but misses the fine-grained representations needed for low-level robotic manipulation.
• Rigidity of Pretrained Features: Keeping the vision encoder frozen leads to total performance collapse on benchmarks like Calvin and Simpler, as the VLM's native visual features are insufficient for control.
3. PointWorld: Data Fidelity and Physical Constraints
PointWorld's bottleneck lies in the quality of 3D supervision and the scope of its physical reasoning.
• Calibration Noise: Existing real-world datasets often have inaccurate hand-eye calibration and "overly smoothed" depth, which can mislead a model trying to learn precise contacts.
• Missing Dynamics: The model currently lacks "photometric dynamics" (e.g., reasoning about light changes) and ignores the non-rigid effects of a robot's body, which limits its ability to handle highly compliant or deformable systems.
4. Membox: Structural Breakage and Retrieval Alignment
The problem with memory-based agents is the "fragmentation-compensation" failure.
• Contextual Destruction: Traditional memory slices interaction into isolated fragments, causing structural breakage where causal and narrative flow are lost.
• Retrieval Bottleneck: While Membox creates cohesive "traces," its current retrieval mechanism is not yet fully aligned with those trace structures, meaning the agent may not always access the richest context for reasoning..”
-Gemini

Noted. So did the two components just add complexity and noise or did my brain instantly wonder off-topic with the GENEO and NAS?

The integration of the two added frameworks GENEOs and Membox directly addresses the semantic and structural gaps that currently limit motion models in robotic systems. While standard vision models excel at high-level reasoning, they often fail at low-level control because their pretrained features are misaligned with the fine-grained geometric requirements of motion.

The "gap" between VLMS and ALMS exists because standard vision encoders are trained on web-scale semantics, which are "poor predictors" of robotic performance. The added frameworks solve this in two ways:
Geometric Alignment (GENEOs): Motion requires understanding physical symmetries (e.g., a robot arm moving an object in 3D space should result in the same physical effect regardless of the starting angle). GENEOs provide a mathematical framework to ensure the vision model is equivariant—meaning it preserves these geometric symmetries. This forces the vision encoder to output features that are physically meaningful for the motion model.

Traditional models suffer from "structural breakage," where the logical connection between a visual observation and a subsequent motion is lost. Membox bridges this by grouping related turns into memory boxes and linking them into event-timeline traces. This ensures the motion model has a continuous narrative thread to follow, allowing it to reason across disjointed segments of a task.

The primary bottlenecks for motion models are noise instability and contextual fragmentation. You can solve them as follows:
Taming Chaos and Noise: Point-based world models often struggle with calibration noise and input perturbations. Because GENEOs are non-expansive, they are theoretically guaranteed to be stable under input noise. By applying a GENEO layer before the motion model, you "smooth" the input space, ensuring that small errors in visual depth or camera pose do not lead to chaotic, unrecoverable motions.
Structural Guidance via NAS: Standard Neural Architecture Search (NAS) is expensive, but using the representation theorem of GENEOs allows you to search a convex and compact space. This dramatically linearizes and simplifies the search for an optimal vision-to-motion architecture.

The Trace Weaver module in Membox allows the motion model to perform multi-hop reasoning. If a robot needs to "pick up the spoon it saw ten minutes ago," Membox provides the stable event trace that standard "fragmentation-compensation" memory systems destroy.

By combining these, the motion model (the "end result") functions as a POINTWORLD architecture that is both robust and historically aware. It predicts 3D point flows (displacements) conditioned on robot actions, leveraging the stable, equivariant features provided by GENEOs and the chronological continuity maintained by Membox.”

OK Good. We’ll run with it. But still the VLM was the bottleneck in all of this and I was still not sure how the two models connected together correctly.

How does the DRQ even know what the champion should be?

The system knows what a champion should be by defining it as the survivor of a continually shifting adversarial arms race. The champion is not measured against the static benchmark but is identified through a specific adversary optimization process in each round of evolution. In biology this continual pressure to adapt is known as, “the Red Queen hypothesis”, in which states that organisms must continuously evolve, not as an advantage but simply to maintain their relative fitness in a constantly changing environment.

The DRQ operates in successive rounds where the objective for a new warrior is always to defeat the lineage of those that came before it Participating in an intra-round optimization called “MAP-Elites”. The system categorizes warriors into a grid based on behavioral descriptors, such as the total number of spawned threads and total memory coverage.Within each cell of this grid, the system only keeps the "elite"—the single warrior that achieves the highest fitness against the historical champions.At the end of the round's optimization, the system selects the highest-performing elite from across the entire behavioral grid to become the new champion for that round.The system evaluates whether a warrior is "champion material" using a cumulative reward system that balances survival and dominance.Although the system only trains warriors against a specific history of previous champions, it "knows" a champion is improving when its generality score increases. Generality is measured by testing the warrior against a held-out dataset of hundreds of human-designed programs. The sources note that as the rounds progress, the champions statistically converge toward a single general-purpose behavioral strategy, becoming increasingly robust against diverse and unseen threats. Which answers my later question of what happens when the robot is repetitive or if it completely learns its environment. I assumed models would make noise and redundant computation but, this is how it mitigates that.

So how does the NAS actually fit then if it's not for the Warriors themselves?

Using Nas in this complex is not an entire model from scratch but it's to bridge the semantic gap between visual representation and embodied control. The problem that the VLM 4 VLA study demonstrates is that standard vision language models diverge from the requirements of action at a specific divergent standard visual features are poor predictors of control performance even when trained on real world IMages. The Nas would be used to search for an action AL Vision coder that replaces or augments the standard VLM vision module It would focus on finding architectures that can translate high level semantics into the 3D point flow required by Pointworld. Without it the system will likely fail to generalize. BLM 4 VLA shows that if the vision encoder is frozen performance collapses across the benchmarks. In DRQ, zero-shot performance was only 1.7%, but applying evolutionary optimization allowed the system to defeat or tie 96.3% of human opponents. This suggests that the "fighting chance" that I wanted for the warriors depends entirely on this automated structural refinement. Membox wood provides a long range evaluation SCA that it needs to judge the architecture's success. Instead of evaluating a nice candidate isolated turns, we would evaluate it on its ability to navigate the event-timeline traces woven by the Trace Weaver. Because Membox showed improvement on temporal reasoning, it provides a "clean" signal for the NAS to determine if a new architecture is actually improving the agent's ability to reason over the narrative flow of a task.

So, How does this all with the GENEO and the NAS tie together to solve the Problem of the semantic gap between vision language understanding And the fine grained requirements needed the action language models since they still act like two separate brains that fundamentally mismatch because VLMS are trained on web scale text and images and excel at high level reasoning even though those image features are poor predictors of downstream robotic control performance?

This gets solved because the GENEO provides the mathematical framework to encode the symmetries directly into the model. That allows the system to recognize repetition via invariance and equivariance. This ensures that the features sent from the vision model to the action model are physically stable and non expansive. GENEOs operate by identifying orbits of data under group actions (like translations or rotations). If a robot repeats a motion, the GENEO recognizes that the new visual/action data belongs to the same orbit as a previously seen state. Because GENEOs are non-expansive, they are guaranteed to be stable under perturbations. This allows the system to "filter out the noise" of small sensor fluctuations.during a repetitive task, ensuring the "perception pair" remains consistent.The NAS Provides The stable structure to house these operators And provides a chassis to house these operators as it searches for the most efficient way to connect the VLM's knowledge to the 3D point flows required for movement. GENEO-based symmetries help Membox maintain a Structured Baseline. If a robot fails at a task and tries again from a slightly different angle, the GENEO's equivariance allows the Membox to recognize the situation as the same "Event" rather than a brand-new, unrelated experience. The Trace Weaver in Membox then links these events (both successes and failures) into a coherent timeline, allowing the agent to perform temporal reasoning and avoid repeating the same mistake.

So there are many different kinds of neural architecture searches and the one that came to my mind originally was the evolutionary NAS but I wanted to make sure that that was going to be the correct one for this system. A standard Gen or evolutionary algorithm uses random mutations and that is for complex code or robotic policies. In sparse complex search spaces like this random changes often produce non functional noise. The DRQ already proves that using an LLM as a mutation operator is Super because it uses the model's prior knowledge; you suggest domain-aware edits which speeds up the discovery of robust champions by orders of magnitude compared to random search. So the correct NAS for this system is a “quality-diversity” NAS, it prevents the search from getting stuck in local minimum by maintaining an archive of diverse elites These elites act as stepping stones ensuring that even if one path fails the system can pivot to another niche in the behavioral grid.

When I searched for a quality diversity NAS, I was surprised to see that the first search result was a research paper of a quality diversity MAP- elites NAS, which is exactly this.

Just like DRQ uses MAP-Elites to evolve "warrior" programs, LLMatic uses it to evolve neural networks. Instead of searching for a single "best" model, LLMatic maintains a network archive categorized by behavioral descriptors like width-to-depth ratio and FLOPS (complexity). This LLMatic begins with a very basic neural network (e.g., one convolutional and one fully connected layer) and uses the archive to find "stepping stones" toward complex, high-performing architectures. It also has a dual-archive system; Strategy and Structure. It stores the physical nodes and it stores the prompts and temperatures used to guide the LLM's design process. This means it isn't just searching for the best Nothing about or point world, it is simultaneously evolving the inner monologue that helps the VLM translate 3D visuals into action commands. Both LLMatic and DRQ replace mindless random mutations with LLM-guided edits. The LLM uses its prior knowledge of deep learning and robotics to suggest meaningful variations to the code defining the network. This makes the search orders of magnitude more efficient. LLMatic found competitive architectures with only 2,000 evaluations, compared to 8,000 in previous methods.

This is perfect so why didn't I see any mention of core war or DRQ?

I didn't see any mentions because DRQ is an adversarial framework focused “Turing-complete” competition in a virtual computer, WhileLLMatic is a procedural optimization framework focused on static benchmarks like CIFAR-10.

That is why what I put together is the bridge to solve this disconnect. It takes the adversarial pressure of DRQ’s and it applies it to the architectural growth of the LLMatic to solve the spatial perception gap in VLM4VLA. In LLMatic, the "champion" is the elite that maximizes test accuracy for a given complexity niche. By integrating this, the robot doesn't just "guess" an architecture; it evolves a library of champions for different tasks, one champion elite for "high-precision needle threading" (low width-to-depth) and another for "wide-area room cleaning" (high width-to-depth).

After feeling like I completed the framework I realized there was still one thing i felt was missing. The actual language part between the two models. One's designed to move around and do things in the real world and the other one is to see it. How can the vision model not only understand what it's saying but also predict how that object can be manipulated in real time within the real world? And to also predict what will happen when doing so, to what extent, and what's used to do it?

So, I thought of yet another paper that I've seen recently called the “Self Improving Reconstruction Engine”. It provides a translation layer that ensures the Vision-Language Model (VLM) sees the world in a way that is mathematically and geometrically consistent for the robotics model.

The VLM4VLA paper argues that a VLM’s general capabilities (like answering questions) are "poor predictors" of its robotic performance because its visual features are not aligned with 3D control. Selfi solves this by providing “Geometrically Aligned Features”. It uses a foundation model's own output as "pseudo-ground-truth" to train a lightweight feature adapter. It forces the VLM's features to be consistent across different views. If the robot sees an object from two different angles, Selfi ensures the underlying feature similarity captures proximity in 3D space. It also"self-improves" without needing human labels by using consistency losses to refine its own internal map. By adding the Selfi adapter, you are giving the VLM the mechanism to reproject and align its semantic knowledge into a 3D geometric frame. This allows it to "generalize what and how objects move" because it finally understands the object's stable 3D existence. Without Selfi, PointWorld might receive "noisy" or "unaligned" visual features from the VLM, making its physical predictions fail. PointWorld needs the high-fidelity 3D alignment that Selfi provides to accurately forecast articulation and contact. While Selfi *learns* alignment from data, GENEO provides the mathematical guarantee that these aligned features remain stable (non-expansive) and preserve physical symmetries (equivariance). Instead of Selfi using a standard DPT head, the NAS can search for the most efficient GENEO-based structure to handle the reprojection logic.

After completing this challenge, I ran my analysis through Claude for formatting. I've heard technical blogs demonstrating research thinking are valuable for portfolios, so I'm adding this to my 14-month independent AI research collection. For context: I've never read a robotics paper before this week. Didn't know the current state of VLM-ALM integration or embodied AI. But I do understand vision models and spatial reasoning, so I could map the problem space.

No PhD. 5 hours start to finish. Not planning to implement or validate experimentally - this is a research synthesis exercise. All reasoning is mine; I've quoted the AI-generated sections. Used NotebookLM for paper analysis, Claude and Gemini for final review and formatting.

References

Original Challenge Papers

1. Digital Red Queen (DRQ): Evolutionary Dynamics in Code Space
   Paper on evolutionary algorithms and adaptive code optimization through competitive dynamics
2. VLM4VLA: Vision-Language Models for Vision-Language-Action
   Research on adapting vision-language models for robotic control and embodied AI applications
3. PointWorld: 3D Dynamics Prediction as Point Flows
   Framework for predicting environment dynamics using 3D point cloud representations without permutation matrices
4. Membox: Structured Memory with Narrative Coherence
   Memory architecture featuring Topic Loom and Trace Weaver for maintaining contextual integrity in AI agent

Integration Papers

5. GENEO: Group Equivariant Non-Expansive Operators
   Algebraic representation theorem for linear GENEOs providing mathematical foundations for equivariant symmetries
6. LLMatic: Neural Architecture Search with LLM-Driven Mutations
   Neural architecture search framework using MAP-Elites and large language models for intelligent architectural evolution
7. Selfi: Self-Improving Reconstruction Engine
   Geometric reconstruction framework for aligning vision-language model features with 3D spatial understanding