EQUINAS: A TYPED DSL FOR CERTIFIED EVOLUTION OF EQUIVARIANT ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural architecture design has evolved from manual engineering to algorithmic search over predefined spaces, and more recently to LLM-assisted generation of architectural modifications. While LLMs make architecture search substantially more flexible, scientific neural networks often obey hard structural constraints that cannot be reliably enforced by executable code alone. In equivariant models, for example, representation types and geometric transformation laws must remain consistent throughout the computation graph; a candidate may compile and train successfully while silently violating the intended symmetry. We introduce *equiNAS*, a typed domain-specific language that turns architecture evolution into certified program transformation. Architectural values expose their representation-theoretic content as semantic types, and every mutation—whether proposed by an LLM or a programmatic search policy—is expressed as a typed patch whose preconditions, dimensional consistency, and equivariance contracts are checked before training. This constrains generation to physically valid transformations without requiring the proposer itself to reason perfectly about representation theory. On EquiformerV3, the typed interface verifies 300/300 proposals against 192/300 (64%) for untyped structured patches under 300 matched architectural intentions per arm, with only 103/600 unrestricted code edits passing the finite-sample equivariance probe in an open-ended stress test; the same explicit intermediate representation enables ahead-of-time compilation and a $1.87\times$ improvement in steady-state training throughput over the production static backend. Searching the certified space also produces parameter-matched architectures with lower mean validation and test error on QM9 across three seeds and improved long-horizon optimization, with additional exploratory transfer gains across molecular properties. Together, these results indicate that typed architectural representations provide a practical interface between flexible LLM-based search and the hard structural requirements of scientific neural networks.

1 INTRODUCTION

Neural architecture design has evolved from manual engineering to automated search over predefined design spaces, and more recently to LLM-assisted generation of structural modifications (Chen et al., 2023; Yang et al., 2025; Ji et al., 2025). Systems such as FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025) further show that language models can guide iterative program evolution through evaluation feedback. This flexibility, however, is difficult to reconcile with scientific neural networks, whose architectures often obey hard structural constraints.

Equivariant networks are a representative case. They encode geometric symmetries directly into neural computation (Bronstein et al., 2021; Thomas et al., 2018), while architectures such as Equiformer (Liao & Smidt, 2023; Liao et al., 2024; 2026) propagate structured representations across layers. A local modification—such as rewiring a residual path or reordering a normalization—can invalidate downstream representation types. Crucially, the resulting model may still compile and train normally while no longer computing an equivariant map. Thus, successful execution is insufficient to establish the semantic validity required for scientific architecture search.

We introduce *equiNAS*, a typed domain-specific language for certified evolution of equivariant architectures (Figure 1). equiNAS exposes representation-theoretic structure as semantic types and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

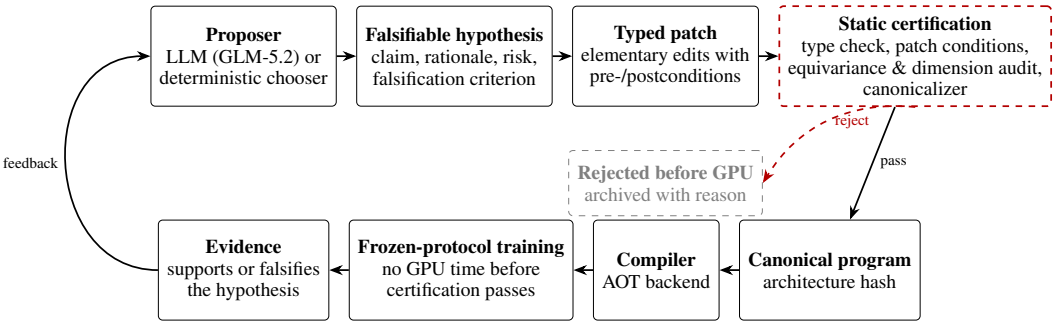


Figure 1: The equiNAS loop. Every mutation is proposed as a typed patch with a falsifiable hypothesis and passes static certification—type safety, patch pre-/postconditions, equivariance and dimension audits, canonicalization—before any GPU time is spent. Rejected patches are archived with their reason; training evidence feeds the next proposal.

expresses architectural changes as typed patches. Before training, the compiler checks type compatibility, equivariance contracts, and dimensional consistency, separating architectural proposal from structural certification. The same interface therefore supports both LLM proposers and conventional search policies without requiring the proposer itself to enforce every representation-theoretic constraint.

On EquiformerV3, the typed interface verifies 300/300 proposals against 192/300 for untyped structured patches under matched architectural intentions, and provides a soundness guarantee relative to its certified primitives. Its explicit graph representation further enables ahead-of-time compilation, improving steady-state training throughput by $1.87\times$. Searching the certified space also discovers parameter-matched architectures with lower mean validation and test error on QM9 across three seeds and improved long-horizon optimization, with exploratory transfer to additional molecular-property targets.

Contributions. Our contributions are: **(1)** a typed intermediate representation that makes equivariant architectures and their mutations statically checkable (Section 2 and proposition 1); **(2)** a certified architecture-evolution interface shared by LLM and programmatic proposers (Sections 3 and 4.1); and **(3)** an empirical evaluation of proposal reliability, graph-enabled compilation, and architecture quality on EquiformerV3 (Sections 4.2 and 5).

2 The Typed Equivariant DSL

A DSL program is a DAG of nodes, each instantiating a versioned primitive (e.g., `core.scalar_linear@3`) with input ports referencing other nodes. The type system tracks, for every value, its group-representation content: irreducible multiplicities, invariant feature axes, reference frames, and locality. A program is well-typed only if every primitive receives representations compatible with its declared interface contract.

Each registry primitive carries an interface contract fixing admissible input/output representation types and a declared transformation law under $E(3)$. The registry is content-addressed; the equivariance contract is pinned by a hash in the protocol file.

All architecture variation is expressed as *typed patches*: sequences of elementary edits (rewire a port, insert or delete a node) subject to preconditions and postconditions re-verified on the child graph. A patch that would leave the graph ill-typed, change the frozen tensor-dimension specification, or fail to produce a new canonical architecture hash is rejected before training. The unit of variation is thus the typed *edit*, not the tensor: operator libraries such as escnn's `FieldType` (Weiler & Cesa, 2019) and e3nn's `Irreps` (Geiger & Smidt, 2022) type individual tensors at runtime, whereas here the architecture and its edits are themselves statically checkable objects.

The canonicalizer maps a program to a normal form from which the architecture hash is derived. Together with the protocol hash and serialized program, this suffices to reconstruct and re-evaluate any candidate exactly.

3 CERTIFIED EVOLUTION

The evolution loop. Let P_t be the DSL program at generation t. A round returns the training evidence F_t (the validation MAE and the hypothesis outcome), and the proposer advances the program through

$$h_t,\, a_t = \pi(P_t, F_t, H_{<t}), \qquad P_{t+1} = a_t(P_t), \tag{1}$$

where h_t is a falsifiable hypothesis, a_t is a typed patch from the catalogue, $H_{<t}$ holds the programs, evidence, and hypotheses of every earlier round, and π is the research policy—the deterministic chooser in the campaign reported in Section 5, GLM-5.2 in round two. A patch is applied only if the child program passes static certification; otherwise it is archived with its reason and the parent is re-sampled. Every child is trained from a fresh initialization within its rung, so the program is the only quantity that carries search history across generations.

Mutation catalogue. Nine families of dimension-preserving local rewrites of the EquiformerV3 block: (S1) block branch topology; (S2) LayerNorm presence in attention radial MLPs; (S3) norm/activation ordering within radial MLPs; (S4) norm/activation ordering on attention-logit branches; (S5) FFN scalar gate orientation; (S6) α-channel normalization; (S7) FFN gate-channel normalization; (S8) attention residual bypass; (S9) FFN residual bypass. On the seed architecture this enumerates **88 actions**. The catalogue is *enumerated* by the type system: each family is a class of local rewrites that the type checker proves representation- and dimension-preserving, and the round-one to round-two expansion (56 to 88 actions) added patch families, not verification code. Candidates are chains of typed patches composed over generations, so the reachable space grows combinatorially while every point inherits the same guarantees.

LLM proposer. The catalogue is the LLM's action space. GLM-5.2 (GLM-5 Team et al., 2026) selects an action and emits a structured *falsifiable hypothesis*—claim, rationale, risk, and an explicit falsification criterion—archived verbatim. A synthesizer call must reproduce the compiler-certified patch field-by-field, with bounded repair and explicitly marked fallback. Training the child either supports the claim or falsifies it; the outcome feeds into the next round.

Baseline as competitor. The seed architecture is trained under the identical protocol and ranked alongside every evolved child. The search must *beat* the baseline to promote any candidate.

Multi-fidelity promotion. All candidates train 20k steps; top 10 by validation, top 3 by novelty, and 2 from the lower half advance to 80k; the top 10 of those advance to 250k on the full-length cosine schedule, from which the winner is frozen.

Verification. Every child passes: (i) the DSL type check; (ii) typed-patch construction with precondition verification; (iii) an equivariance audit (rotations, translations, permutations of a probe batch); (iv) compilation to an executable backend and a pretraining numerical-health check. Failures are archived with their reason.

Campaigns. The architecture-quality campaign reported in Section 5 uses a deterministic chooser over the same typed-patch interface—a controlled instantiation that isolates the contribution of the certified search space from that of the proposer. The GLM-in-the-loop campaign (round two) runs the full system with the baseline in the pool; its archived hypotheses exhibit evidence-conditioned search—citing prior runs, mechanistic rationales, and explicit falsification criteria. At the results freeze, 49 of its 60 children are evaluated: the best-so-far child (sampled by the campaign's ε-greedy exploration arm, provenance archived) improves on the round-one pool best by 3.8% under the search protocol, and the best LLM-reasoned child improves on it as well (Section B records the settings; Section E archives hypothesis excerpts). The campaign continues beyond the freeze.

4 CORRECTNESS AND SPEED

Our correctness guarantee is a soundness principle (Proposition 1): composition of certified equivariant primitives preserves equivariance, *relative to a trusted computing base*—the certified primitive implementations, the type checker, the canonicalizer, and backend compilation. It guarantees that programs violating the encoded contracts are rejected before training; it does not assert freedom from implementation bugs or contract completeness. Capacity is held near-constant across the search space rather than exactly constant: norm-presence toggles remove or insert small affine parameter blocks in multiples of 256, so across both campaigns candidate parameter counts span 84,960,129 to 84,963,201—within 0.004% of the baseline's 84,961,153 (Section B gives the full accounting), far too small to drive the observed differences. The arms of every paired-protocol comparison are exactly matched at 84,961,153.

Implementation correctness. We audit the DSL compilation pipeline numerically (Table 1): parameter composition is identical (333 mapped tensors; 84,961,153 parameters on all three paths), and the AOT-compiled static path reproduces the dynamic execution to a relative RMS of 6.4×10^{-4} at fp32 and 9.0×10^{-3} at bf16—the precision used for training, where the divergence is consistent with bf16 rounding; the worst-parameter gradient and optimizer-step deviations match to 2.5×10^{-5} and 2.0×10^{-6}.

Table 1: Compilation correctness audit (worst case over a probe batch, seed 201). The bf16 row is the precision used for training.

Comparison	Precision	Rel. RMS	Mean abs	Max abs
Forward (dynamic vs. AOT static)	fp32	6.4×10^{-4}	1.3×10^{-5}	1.8×10^{-5}
Forward (dynamic vs. AOT static)	bf16	9.0×10^{-3}	1.8×10^{-4}	2.4×10^{-4}
Gradients (worst parameter)	fp32	—	2.0×10^{-6}	2.5×10^{-5}
Optimizer step (worst parameter)	fp32	—	4.8×10^{-7}	2.0×10^{-6}

4.1 PROOF VS. FREE CODE EDITING

To quantify the value of the typed interface, we run an open-ended stress test over three proposal formats (n=600 proposals each, identical model, temperature, and architectural context). In the *free-editing* arm, the LLM writes modified Python code directly. In the *uncertified-patch* arm, the LLM expresses its mutation as a structured patch—a JSON edit list over named nodes, ports, and attributes with free values, mirroring graph/config-based proposal formats—which is parsed and applied *without* type checking or precondition verification. In the *typed-patch* arm, the LLM selects from the 88-action mutation catalogue and the compiler verifies each proposal. The three arms decompose the interface gap: free editing vs. uncertified patches reflects the contribution of the structured representation, and uncertified vs. typed patches measures the contribution of the certified typed interface, which combines action-space restriction with static certification.

Table 2: Interface stress test with n=600 proposals per arm under matched model, task, and context. The columns compare free Python editing, uncertified structured patches, and verified typed patches.

Metric	Free editing	Uncertified patch	Typed patch
Syntactically valid	600 (100%)	258 (43.0%)	600 (100%)
Executes without error	111 (18.5%)	258 (43.0%)	600 (100%)
Passes equivariance probe	103 (17.2%)	258 (43.0%)	600 (100%)
Passes structural analysis	1 (0.17%)	96 (16.0%)	600 (100%)
Verified equivariant	**1 (0.17%)**	**96 (16.0%)**	**600 (100%)**
95% CI	[0.03%, 0.94%]	[13.3%, 19.1%]	[99.4%, 100%]

Note. All percentages use n=600. "Verified equivariant" requires both the finite probe and the stated static analysis; it is not a formal proof of implementation correctness. Confidence intervals are Wilson score intervals.

As Table 2 shows, 82% of free-editing proposals fail at runtime (tensor-shape mismatches, API misuse, indexing errors). Of those that execute, most pass a finite-sample equivariance probe, but only one passes the textual structural check—a conservative bound sharpened by the matched-intention control in Section I. The uncertified-patch arm decomposes this gap: structuring the proposal as a JSON edit list over named nodes and ports—rather than free code—already lifts the executes/probes-clean rate from 18.5% to 43.0% (though under matched intentions the same model executes 269/300 as free code, so this open-ended gap reflects proposal difficulty as well as format; Section I), but 63% of those structurally well-formed, numerically equivariant proposals are rejected once static certification is applied (16.0% final, versus 100% for the typed-patch arm). Manual inspection of the rejected cases is dominated by dimension drift: an edit changes the declared irreps content or radial-basis size of a node, which yields a freshly-built, numerically equivariant model at the *new* dimensions—passing the probe outright—while silently breaking the representation-type and dimension contracts that static certification enforces and the equivariance probe was never designed to check. The three-arm gap therefore decomposes as: free editing $\rightarrow$ uncertified patch (0.17% to 16.0%) attributes to the structured representation; uncertified patch $\rightarrow$ typed patch (16.0% to 100%) attributes to the certified typed interface. Under intentions fixed across arms, the typed interface still verifies 100% where the untyped patch interface verifies 64% (Section I).

4.2 Training acceleration

The DSL's explicit graph structure enables ahead-of-time compilation: the compiler emits fast paths for certified subgraph shapes and hoists shared Wigner-D constructions across nodes—optimizations enabled by the explicit typed graph rather than by a hand-tuned backend. On a single RTX 3080 (single-run benchmark), steady-state training throughput over 600 optimizer steps improves by $1.87\times$ (production static 0.915 to optimized static 1.708 steps/s; compile time excluded). Ablation attributes the gain to fast paths ($1.58\times$) and shared Wigner-D construction ($1.26\times$); the AOT rewrite alone provides memory feasibility rather than wall-clock gains.

5 Experiments

Protocol. All comparisons use one frozen evaluation protocol (`paired-formal-v2@1`); the search ran under a separate, explicitly weaker budget, and the two are never mixed. The search budget trains every child for the same 20,000 steps at seed 201 but with batch size 8 on a fixed quarter of the training split (27,500 of 110,000 molecules, hash-pinned) with epoch-based evaluation—eight times fewer examples per step and a quarter of the data of the paired protocol—so search values rank candidates within a campaign and are never compared to paired-protocol values (Section B gives the full settings). The primary budget is 20,000 optimizer steps per model and seed; selected models extend to 80,000 and the search winner to 250,000 steps, always on the full-length 515,700-step cosine schedule (truncated, not compressed, so every rung sits on the identical learning-rate trajectory as the full-length reference recipe). We report *best* validation MAE. The test split is never read during search. Best values per column are in bold throughout.

5.1 Main results on QM9 α

Table 3 reports the paired comparison across three seeds together with the budget ladder beyond 20k. On validation, TypeEvo is better on every seed (per-seed 0.189/0.181/0.191 versus 0.191/0.207/0.205 for seeds 201/202/203; mean 0.187 ± 0.005 versus 0.201 ± 0.009). Under a pre-registered final-step test protocol (evaluate the last checkpoint, no selection rule, three seeds), TypeEvo attains a lower three-seed mean test MAE (0.207 ± 0.016 versus 0.215 ± 0.012). Per-seed test values (TypeEvo 0.218/0.188/0.214 versus 0.202/0.226/0.215 for seeds 201/202/203) show the difference is small relative to seed variance—two of three seeds favor TypeEvo—so we read it as a consistent direction of improvement rather than a decisive margin. The s201 validation margin (0.002) is additionally smaller than the measured DSL-compilation-path effect (0.005, Section J), so at that seed the comparison is reference-relative rather than an isolated architecture effect; the s202/s203 margins exceed the measured compilation effect, but no DSL-compiled-seed control exists at those seeds to confirm it is similarly sized there.

Table 3: Paired QM9 (Ramakrishnan et al., 2014) α results across the shared 515,700-step cosine schedule (equal capacity, 85M parameters). Validation MAE (best-to-date at each budget rung), per seed and across the ladder; mean $\pm$ sample std; test MAE at the 20k final-step checkpoint. The 20k columns are three seeds; rungs beyond 20k are single-seed (s201) trajectories, and only the search winner extends past 80k. $^{\text{ref}}$: our matched reference run of the unmodified architecture (Section K). †: best-so-far at 92% of the schedule (data to 96% at submission). The reference run provides both 250k and 515k values for the baseline.

	Validation MAE					Test (20k)
Model	**s201**	**s202**	**s203**	**80k**	**250k / 515k**	**Mean $\pm$ std**
EquiformerV3	0.191	0.207	0.205	0.107	0.0780 / 0.0663$^{\text{ref}}$	0.215 $\pm$ 0.012
TypeEvo (round 1)	**0.189**	**0.181**	**0.191**	**0.103**	**0.0723 / 0.0596**†	**0.207 $\pm$ 0.016**

5.2 Long-horizon behavior

Figure 2 overlays validation curves under the paired protocol. TypeEvo is the *slowest* model at the first evaluation (step 1,719: MAE 1.834 vs. 0.550 for EquiformerV3); it converges to the baseline band by 20k; and under the extended schedule it improves by a further factor of two. At 80k, both models complete: EquiformerV3 0.107 and TypeEvo 0.103—statistically indistinguishable. The separation emerges beyond 80k: TypeEvo reaches a best of **0.0723** at the end of its 250k schedule while the baseline has completed its budget.

Comparison on the matched reference trajectory. Because every run shares the 515,700-step cosine schedule, intermediate points are directly comparable with a matched full-length reference run of the unmodified EquiformerV3 implementation—our own training, not a checkpoint from the architecture's original authors, since no official QM9 training recipe is published upstream (Section K records the commit and configuration). At the 250k mark, the reference run's best-to-date is 0.0780 and TypeEvo's is **0.0723**—the evolved architecture overtakes at half the optimization steps. Extending along the same schedule, TypeEvo reaches **0.0656** at 63% of the optimization steps, numerically surpassing the reference run's best full-schedule value (0.0663). The margin is small ($\approx$1%) and both are single trajectories; the gap widens as the run continues—at submission time, TypeEvo's best has reached **0.0596** by 92% of the schedule, 10% below the reference run's full-schedule best. Across the paired 20k runs of this study (both arms trained sequentially on one A100 per run), the per-step training throughput of TypeEvo and the baseline agrees to within 0.7% (mean throughput ratio 1.001, range 0.996–1.007), so optimization steps are an accurate proxy for training compute in this comparison.

5.3 Structural analysis

The campaign produced 60 valid architectures over seven generations, with parameter counts within 1,024 of the baseline (norm-presence toggles remove 256-parameter affine blocks; Section B). All five top candidates share the same first mutation—rewiring block 0's FFN to read the pre-attention shared input (the parallel branch topology)—and the remaining variation is dominated by norm/activation reordering and gate re-orientation (Figure 3). Every mutation in the winning lineage is an instance of a catalogued family with an archived hypothesis.

5.4 Founder ablation

All five top candidates share the same first mutation (Section 5.3). To separate the founder's contribution from the accumulated chain, we compare four architectures under the identical 20k protocol: the baseline, the founder-only architecture, TypeEvo without the founder mutation, and TypeEvo itself (Table 4a).

The founder mutation alone recovers the entire 20k improvement relative to this baseline (0.189 versus the baseline's 0.191, matching TypeEvo at this budget), while the same five later mutations applied without the founder are *deleterious* (0.225, worse than the baseline): the short-budget gain is not merely concentrated in the parallel branch topology of block 0—the later mutations are beneficial

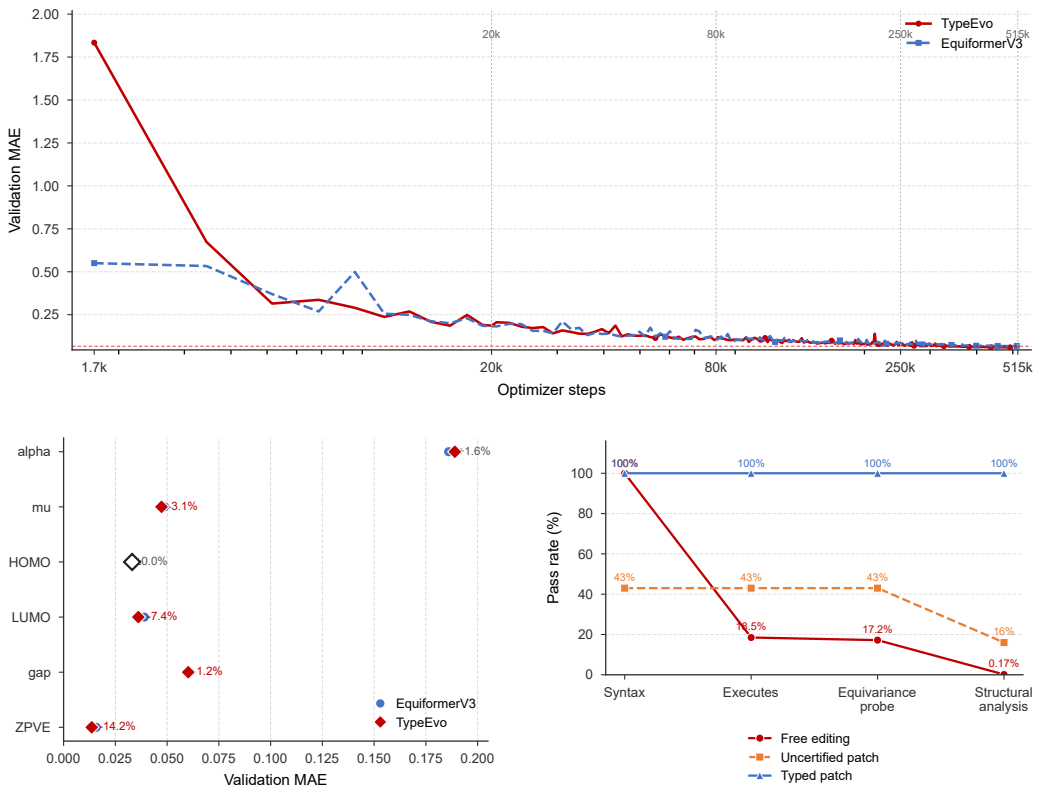


Figure 2: Top: best-so-far validation MAE (single seed, s201) versus optimizer steps on the shared 515,700-step schedule. Bottom left: multi-target transfer at 20k (Δ% relative to EquiformerV3). Bottom right: interface verification funnel (Section 4.1). The 1.87× AOT throughput gain and the 0.19055-vs-0.18901 paired-protocol margin are reported in the text (Sections 4.2 and 5.1).

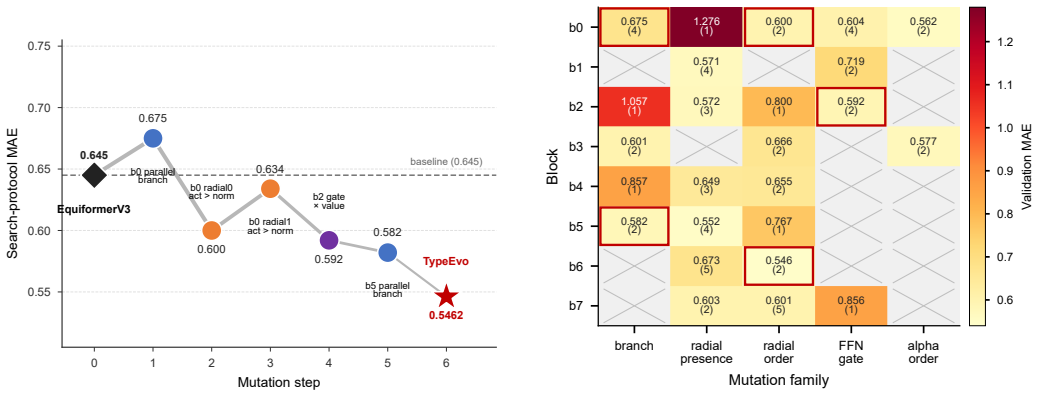


Figure 3: Round-one search. Left: mutation chain of the winner (TypeEvo), with search-protocol MAE after each step (dashed line: the baseline's own search score). Right: best search MAE per block and mutation family over all 60 children (attempt counts in parentheses; crossed cells unexplored; red boxes mark TypeEvo's lineage). The paired-protocol validation at 20k (s201, 0.19055 vs. 0.18901) is reported in Section 5.1.

only in the founder's presence, a strong epistatic interaction. This baseline is the upstream implementation (0.191); against the DSL-compiled seed program (0.186, Table 5), both the founder-only and full TypeEvo architectures are instead slightly worse (0.189) at this budget, so the recovered-

Table 4: Founder ablation (a) and rMD17 benzene transfer (b). (a) 20k steps, seed 201: does the shared first mutation explain the gain, or does the chain add incremental benefit? (b) Frozen DSL programs; fresh models, all parameters retrained: final energy MAE (eV); the force-MAE comparison favors the baseline on both benzene splits and is summarized in the text, and these numbers are not comparable to standard from-scratch rMD17 benchmarks.

(a) Founder ablation

Architecture	b0 topology	Other mutations	Val MAE (20k)
Baseline	×	×	0.191
Founder only	✓	×	**0.189**
TypeEvo minus founder	×	✓	0.225
TypeEvo	✓	✓	**0.189**

(b) rMD17 benzene transfer: final energy MAE

Split	EquiformerV3	TypeEvo
benzene s01	0.070	**0.052**
benzene s02	**0.100**	0.121

improvement reading holds relative to the upstream reference, not relative to the DSL-compiled parent (Section J). The chain earns its additional value at longer horizons, where TypeEvo surpasses the matched full-length reference trajectory (Section 5.2).

5.5 TRANSFER

QM9 multi-target. For each additional QM9 target we instantiate a fresh model with the frozen architecture and retrain all parameters under the paired protocol (Table 5). In this single-seed exploratory transfer study, TypeEvo matches or improves over the matched baseline on all five targets tested: dipole μ (0.0473 vs. 0.0488, 3.1% better), LUMO (0.0361 vs. 0.039, 7.4%), HOMO–LUMO gap (0.0602 vs. 0.0609, 1.2%), and ZPVE (0.0136 vs. 0.0158, 14.2%), with HOMO exactly tied (0.0331). The architectural advantage is therefore not specific to polarizability: the same structural changes transfer to dipole, electronic, and vibrational properties with distinct chemical character.

Table 5: QM9 multi-target validation MAE at 20k steps (frozen architectures; fresh models, all parameters retrained; single seed). $\Delta\%$ is the relative change of TypeEvo versus EquiformerV3 (negative is better). The baseline here is the DSL-compiled seed program; Table 3 reports the upstream implementation (0.191 on α).

Target	EquiformerV3	TypeEvo	$\Delta\%$
α (polarizability)	**0.186**	0.189	$+1.6$
μ (dipole)	0.0488	**0.0473**	-3.1
HOMO	**0.0331**	**0.0331**	± 0.0
LUMO	0.039	**0.0361**	-7.4
gap	0.0609	**0.0602**	-1.2
ZPVE	0.0158	**0.0136**	-14.2

rMD17. We freeze the DSL programs, instantiate fresh models, and retrain all parameters with the rMD17 output head and losses (Christensen & von Lilienfeld, 2020) (Table 4b). Cross-domain transfer is mixed: the baseline is better on force MAE in both benzene splits, while TypeEvo achieves better energy MAE on s01—the QM9 gains do not uniformly transfer, indicating a task-conditional rather than a universal architectural improvement. These numbers are not comparable to standard from-scratch rMD17 benchmarks, which train under a different protocol.

6 Related Work

LLM-driven code and architecture evolution. FunSearch (Romera-Paredes et al., 2024) evolves program skeletons against an evaluator; EvoPrompting (Chen et al., 2023) searches architectures as raw Python; AlphaEvolve (Novikov et al., 2025) scopes edits to marked blocks with structured diffs. Closer to our setting, NADER (Yang et al., 2025) represents architectures as graphs for LLM-based multi-agent design with a Reflector module that accumulates search experience, and RZ-NAS (Ji et al., 2025) couples LLM reflection with zero-cost proxies. All rely on post-hoc filtering—incorrect programs are discarded after consuming evaluation budget—and none certifies representation-theoretic properties of the network function.

Constrained generation. Grammar-constrained decoding (Poesia et al., 2022; Park et al., 2024) enforces syntactic validity at decode time but does not certify semantic properties of the network function.

Search-space abstraction. LLMasTool (Yoshimura et al., 2026) and GraphIR (Liu et al., 2026) represent architectures as trees or graphs, guaranteeing structural executability without type systems or semantic invariants.

Typed equivariant operators. e2cnn/escnn (Weiler & Cesa, 2019) define `FieldType` and `GeometricTensor` with dynamic type checking on transformation laws; e3nn (Geiger & Smidt, 2022) propagates `Irreps` and ships equivariance tests. NeuralArTS (Wu et al., 2022) organizes NAS operator spaces with a type system over input/output shape compatibility; Maile et al. (2022) optimize architectures with equivariance awareness. These systems type operators or tensors at construction or runtime. equiNAS instead lifts representation contracts to an *architecture-evolution* intermediate representation: the edits themselves are typed, canonicalized, statically checkable objects with provenance, enabling mutation—by LLM or by programmatic chooser—without exposing unconstrained executable code, and rejecting candidates that violate the encoded contracts before training.

7 Conclusion

We presented equiNAS, a typed domain-specific language that makes equivariant architecture evolution a certified process: every mutation—from a GLM proposer or a programmatic chooser—is a statically verified typed patch, so no GPU time is spent on a candidate that violates type safety, equivariance, or dimension consistency. The interface is what carries the results: it is *safe* (300/300 versus 192/300 verified proposals under matched intentions), *fast* ($1.87\times$ steady-state throughput from graph-enabled compilation), and *effective* (TypeEvo attains a lower three-seed mean test MAE than the baseline, numerically surpasses a matched full-length reference run at 63% of the optimization steps, and matches or improves on five additional QM9 targets). Every result is reproducible from an (architecture, protocol) hash pair. The founder ablation attributes the short-budget gain to a single interpretable rewiring whose benefit is epistatic—the later mutations help only in its presence—and the certified interface, not the choice of proposer, carries these results; the GLM-in-the-loop campaign continues over it with every proposal archived as a falsifiable hypothesis.

Limitations. The certified search space is bounded by the dimension-preserving contract: width-changing mutations are excluded, and all 109 evaluated candidates explore topology and normalization at capacity within 0.004% of the baseline (norm toggles shift the count by multiples of 256); a typing scheme for parameterized capacity would extend the space. The transfer and multi-target studies are single-seed, so their margins (e.g., μ 0.0473 vs. 0.0488) are directional rather than significant; multi-seed replications would quantify seed variance. Improvements are task-conditional: rMD17 transfer is mixed, so the gains should not be read as universal. Soundness holds relative to the trusted computing base—primitive implementations, type checker, canonicalizer, and backend—rather than asserting freedom from implementation bugs, though the parity audit bounds numerical divergence. Finally, all experiments run at 85M parameters under single-GPU budgets, so the scaling behavior of both the evolved architectures and the search itself remains untested. Future work should pair the certified interface with capacity-morphing contracts and multi-domain protocols, which the typed-patch machinery is designed to accept.

AI USE STATEMENT

LLMs (GLM-5.2) are used as the mutation proposer inside the evolution loop: they select an action from the frozen mutation catalogue and emit a falsifiable hypothesis whose synthesizer output must reproduce a compiler-certified patch. LLMs also assisted in drafting and revising this manuscript. All reported numbers come from the frozen-protocol trainers and evaluators, never from agent self-report; every LLM proposal is archived verbatim with its training outcome, so each claim in the paper is traceable to a certified patch and a protocol-hash-identified run.

ETHICS STATEMENT

This work trains and evaluates molecular-property models on the public QM9 and rMD17 datasets; no human subjects or sensitive data are involved. The mutation catalogue is constrained to dimension-preserving, equivariance-certified rewrites, and every candidate is archived with its verification trace, so the search cannot silently produce non-equivariant or unverifiable architectures. The compute footprint (single-GPU A100 training for all paired runs) is reported per experiment.

REPRODUCIBILITY STATEMENT

Every training run is identified by a protocol hash (frozen JSON, field-level identity verified programmatically), an architecture hash (canonicalized DSL program), and archived artifacts (checkpoints, metrics, symmetry reports, and the verbatim hypothesis stream). The search protocol and the evaluation protocol are separate, explicitly weaker/stronger budgets that are never mixed; all paired comparisons share identical seeds, data splits, schedules, and batch sizes. The anonymized supplementary artifact package includes the DSL compiler, the frozen protocols, and the DSL programs and archived result files for every architecture named in the main-text tables and the interface-safety and matched-intention studies; the remaining search-campaign candidates are fully specified as (action, protocol) pairs in Appendices C and D and are reconstructible from the same compiler.

REFERENCES

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478, 2021. URL https://arxiv.org/abs/2104.13478.

Angelica Chen, David M. Dohan, and David R. So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, pp. 7787–7817, 2023. doi: 10.52202/075280-0342. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/184c1e18d00d7752805324da48ad25be-Abstract-Conference.html.

Anders S. Christensen and O. Anatole von Lilienfeld. On the role of gradients for machine learning of molecular energies and forces. *arXiv preprint arXiv:2007.09593*, 2020. URL https://arxiv.org/abs/2007.09593. Introduces the revised rMD17 dataset.

Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. arXiv preprint arXiv:2207.09453, 2022. URL https://arxiv.org/abs/2207.09453.

GLM-5 Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Zipeng Ji, Guanghui Zhu, Chunfeng Yuan, and Yihua Huang. RZ-NAS: Enhancing LLM-guided neural architecture search via reflective zero-cost strategy. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27237–27254, 2025. URL https://proceedings.mlr.press/v267/ji25a.html.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3D atomistic graphs. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KwmPfARgOTD.

Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. EquiformerV2: Improved equivariant transformer for scaling to higher-degree representations. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mCOBKZmrzD.

Yi-Lun Liao, Alexander J. Hoffman, Sabrina C. Shen, Alexandre Duval, Sam Walton Norwood, and Tess Smidt. EquiformerV3: Scaling efficient, expressive, and general SE(3)-equivariant graph attention transformers. arXiv preprint arXiv:2604.09130, 2026. URL https://arxiv.org/abs/2604.09130.

Zhen Liu, Wanqi Zhou, Shuanghao Bai, Yuhan Liu, Jinjun Wang, and Jingwen Fu. GraphIR: Architecture-level search states for LLM-guided neural architecture evolution. *arXiv preprint arXiv:2608.01633*, 2026. URL https://arxiv.org/abs/2608.01633.

Kaitlin Maile, Dennis G. Wilson, and Patrick Forré. Equivariance-aware architectural optimization of neural networks. *arXiv preprint arXiv:2210.05484*, 2022. URL https://arxiv.org/abs/2210.05484.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2bdc2267c3d7d01523e2e17ac0a754f3-Abstract-Conference.html.

Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KmtVD97J43e.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, 2014. doi: 10.1038/sdata.2014.22. URL https://doi.org/10.1038/sdata.2014.22.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with Large Language Models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. arXiv preprint arXiv:1802.08219, 2018. URL https://arxiv.org/abs/1802.08219.

Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems*, volume 32, pp. 14334–14345, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/45d6637b718d0f24a237069fe41b0db4-Abstract.html.

Robert Wu, Nayan Saxena, and Rohan Jain. NeuralArTS: Structuring neural architecture search with type theory (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 13085–13086, 2022. doi: 10.1609/aaai.v36i11.21679. URL https://doi.org/10.1609/aaai.v36i11.21679.

Zekang Yang, Wang Zeng, Sheng Jin, Chen Qian, Ping Luo, and Wentao Liu. NADER: Neural architecture design via multi-agent collaboration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4452–4461, 2025. doi: 10.1109/CVPR52734.2025.00420. URL https://doi.org/10.1109/CVPR52734.2025.00420.

Masakazu Yoshimura, Zitang Sun, Yuiko Sakuma, Junji Otsuka, Atsushi Irie, and Takeshi Ohashi. LLM as a tool, not an agent: Code-mined tree transformations for neural architecture search. *arXiv preprint arXiv:2604.16555*, 2026. URL https://arxiv.org/abs/2604.16555.

A Equivariance and Equiformer Representability of the Typed DSL

This appendix formalizes the two theoretical claims used in the paper. The first concerns soundness: under explicit assumptions on primitive implementations and compilation, a well-typed DSL program preserves its declared equivariance. The second concerns architecture-specific representability: the current language can encode the audited Equiformer constructions relevant to this work. The latter is deliberately restricted to supported configurations and blocks; it is neither a universality claim nor a statement that every configuration in the Equiformer family has been verified.

A.1 Semantic Types and Equivariant Primitive Contracts

Let G be a transformation group. A semantic type τ is associated with a value space $\mathcal{X}_\tau$ and an action $\alpha_\tau : G \times \mathcal{X}_\tau \to \mathcal{X}_\tau$. A multi-input map $\phi : \mathcal{X}_{\tau_1} \times \cdots \times \mathcal{X}_{\tau_q} \to \mathcal{X}_{\tau_o}$ is equivariant when

$$\phi\big(\alpha_{\tau_1}(g)x_1, \ldots, \alpha_{\tau_q}(g)x_q\big) = \alpha_{\tau_o}(g)\phi(x_1, \ldots, x_q), \qquad g \in G. \tag{2}$$

Invariance is the special case in which the output action is trivial.

Points and feature fibers carry different actions. For $g = (R, t) \in SE(3)$,

$$\alpha_{\mathrm{pt}}(R, t)r_i = Rr_i + t, \qquad \alpha_\ell(R, t)x_i^{(\ell)} = D^{(\ell)}(R)x_i^{(\ell)}. \tag{3}$$

Thus points transform affinely, whereas feature fibers carry linear representations of the rotational subgroup. Translation acts through relative geometry and carrier structure rather than through an additional feature-space matrix. When inversion is included, feature fibers use $O(3)$ representation labels and parity becomes part of the type.

A feature space decomposes as

$$\mathcal{V}_\tau \cong \bigoplus_{\lambda \in \Lambda_G} \mathbb{R}^{m_\lambda} \otimes V_\lambda, \qquad \Lambda_G = \begin{cases} \mathbb{N}_0, & \text{for } SO(3)\text{-type feature fibers,} \\ \mathbb{N}_0 \times \{+1, -1\}, & \text{when inversion is included.} \end{cases} \tag{4}$$

Here m_λ is the multiplicity of the complete irrep V_λ; an individual coordinate inside an irrep block is not an independently typed channel. For equivariant tensor values, the implemented semantic record is summarized as

$$\tau = (k, G, c, r, \mathcal{I}, \mathcal{A}, \mathcal{L}, d, \mu, q), \tag{5}$$

where k is the value kind, c the carrier, r the reference frame, $\mathcal{I}$ the irrep list, $\mathcal{A}$ the named axes and their roles, $\mathcal{L}$ the representation layout, d the data type, μ the physical measure, and q the declared certification level. The pair $(G, \mathcal{I})$ determines the fiber action, while (c, r) records its geometric context. The remaining fields constrain program composition but are not themselves representation-theoretic evidence. Points, index maps, graph topologies, and sampled spherical grids therefore use distinct tagged types rather than being identified with feature tensors of the same runtime shape.

For primitive attributes a, static elaboration has the form

$$\frac{\Gamma \vdash v_1 : \tau_1 \quad \cdots \quad \Gamma \vdash v_q : \tau_q \qquad C_\phi(\tau_1, \ldots, \tau_q, a)}{\Gamma \vdash \phi_a(v_1, \ldots, v_q) : T_\phi(\tau_1, \ldots, \tau_q, a)}. \tag{6}$$

The predicate C_ϕ states the admissibility conditions and T_ϕ determines the output type. A successful derivation establishes the declared interface conditions; the backend implementation must separately realize the map in equation 2.

The principal contracts follow from representation theory. An equivariant linear map has the block form

$$L = \bigoplus_{\lambda \in \Lambda_G} W_\lambda \otimes I_{V_\lambda}, \tag{7}$$

so it may mix multiplicity channels but not arbitrary irrep coordinates. Coordinatewise nonlinearities are restricted to invariant scalars; higher-order features may instead be modulated by invariant gates. A Clebsch–Gordan tensor-product path is admissible only if

$$|\ell_1 - \ell_2| \le \ell_o \le \ell_1 + \ell_2, \qquad p_o = p_1 p_2 \quad \text{when inversion is included.} \tag{8}$$

Residual addition requires identical semantic interfaces. Carrier transport and frame conversion must be explicit, symmetric aggregation must use the declared graph indices, and invariant readout must select a trivial representation rather than an arbitrary one-dimensional runtime tensor.

A.2 Equivariance Preservation of Well-Typed DSL Programs

A DSL program is a finite directed acyclic graph $P = (V, E)$. Nodes instantiate registered primitives and edges carry semantically typed values between named ports. The program is well typed when every primitive-specific input contract is discharged, each declared output agrees with the type inferred by the registered rule, every program output satisfies its task contract, and carrier or frame changes are performed by explicit registered conversions. Compatibility is therefore established by the rule at each consumer port, not by an implicit shape coercion.

Proposition 1 (Equivariance preservation)**.** Let P be a finite well-typed DSL program with declared input and output actions. Assume that: (i) each instantiated primitive implements the equivariant map declared by its registered contract; (ii) every connection satisfies the semantic compatibility relation used by the typing derivation; and (iii) compilation is semantics preserving, so that the lowered graph realizes the primitive maps and connections of the abstract DSL program. Then the compiled network $\mathsf{C}(P)$ is equivariant:

$$F_{\mathsf{C}(P)}(\alpha_{\mathrm{in}}(g)x) = \alpha_{\mathrm{out}}(g) F_{\mathsf{C}(P)}(x), \qquad g \in G. \tag{9}$$

Proof. Because P is finite and acyclic, its nodes admit a topological ordering $v_1, \dots, v_n$. Program inputs transform by their declared actions. Suppose inductively that every predecessor of v_j obeys the action attached to its inferred type. Port compatibility makes these predecessor values admissible inputs to the primitive at v_j. Assumption (i), applied to the same $g \in G$ at all ports, implies that the output of v_j obeys its inferred action. Induction therefore establishes the declared transformation law at every node. Fan-out only reuses a typed value, while merge nodes are covered by the multi-input form of equation 2; hence the result holds at every designated output. Assumption (iii) transfers the property from the abstract program to the compiled network. □

This is a soundness result, not a completeness result. It does not imply that every equivariant graph is expressible, that every rejected graph is non-equivariant, or that finite numerical tests constitute a proof. In particular, primitive correctness and semantics-preserving compilation are premises rather than consequences of static typing.

A.3 Representation of Equiformer V1–V3

The second question is architecture specific: whether the current DSL can encode the Equiformer constructions relevant to this work. We answer it by matching the computations in Equiformer V1–V3 (Liao & Smidt, 2023; Liao et al., 2024; 2026) to typed primitives and finite macros, and by checking the resulting external interfaces. Table 6 summarizes the audited scope. The claim concerns the declared computation graph and its semantic types. Numerical equality of a particular lowered implementation is a separate implementation claim.

Equiformer V1. An audited V1 block begins with irrep-wise linear maps and tensor-product message construction. Node features are gathered to edges, coupled to typed geometric features, and returned to nodes by a symmetric segment reduction. Attention logits are invariant scalars; their normalized weights therefore modulate equivariant values without changing the value representation. Gated nonlinearities use invariant scalar gates, residual addition requires identical representation, carrier, and frame types, and the graph prediction is obtained by selecting trivial scalars before pooling. These operations are represented by registered primitives or finite macros with matching external ports. The current evidence covers selected frozen V1 programs and validated configurations, not every external V1 implementation.

Table 6: Architecture-specific DSL coverage. "Audited" refers to the configurations or blocks inspected in the current implementation, not every configuration admitted by the original model families.

Model	Architectural mechanism	DSL realization	Supported scope
V1	Irrep linear maps; tensor-product messages; invariant attention; gated FFN; residual and scalar readout	`irrep_linear`, `tensor_product`, invariant score/scale and reduction, gate, residual add, scalar selection and pooling	Selected frozen blocks and configurations; typed generic and constructor-locked paths
V2	Edge-local frame; residual $SO(2)$ computation; spherical nonlinearity; attention and aggregation	Explicit frame entry/exit, $SO(2)$ convolution, separable S^2 activation, invariant attention, segment reduction	Audited local-frame blocks; no claim of complete official V2 model equivalence
V3	Spherical lift and transforms; merged normalization; attention; SwiGLU-style spherical FFN; energy and force heads	Typed spherical/grid values, frame rotations, $SO(2)$/$SO(3)$ linear maps, merged norm, invariant attention, scalar and vector output paths	Audited direct energy–force configurations and the configuration schema used in this work; not family-wide

Equiformer V2. The V2-specific mechanism is an explicit local-frame path. Edge features are converted from the global frame to an edge-local frame, processed by residual-$SO(2)$ linear operations and a compatible spherical nonlinearity, and converted back before aggregation or residual addition in the global frame. The DSL represents this sequence by typed frame conversion, $SO(2)$ convolution, separable S^2 activation, and inverse conversion; the common invariant-attention, reduction, and readout operations use the same contracts as above. This establishes a well-typed encoding of the audited V2 blocks. It does not establish a semantics-preserving lowering of every complete official V2 configuration.

Equiformer V3. The audited V3 graph makes the spherical computation explicit. Its input path combines atom embeddings, radial features, and an axisymmetric spherical lift. Attention uses typed edge-frame rotation, $SO(2)$ linear maps, invariant scores, and inverse rotation before segment reduction. The FFN combines $SO(3)$ linear maps with sampled spherical projection, SwiGLU-style gated operations, and inverse projection; merged equivariant normalization and residual paths preserve the surrounding feature type. The output path selects invariant scalars for energy and an equivariant vector channel for direct forces. The current repository contains a normalized, well-typed direct energy–force graph and configuration-specific parameter, forward, and gradient oracles for an audited two-layer instance. The production seed and its typed-patch descendants instantiate the same typed configuration schema. These audits support the stated configurations but do not prove family-wide coverage or formally verify every production lowering.

Proposition 2 (Representability of audited Equiformer configurations)**.** The audited Equiformer constructions in Table 6 admit well-typed encodings in the current DSL: selected frozen V1 configurations, the inspected V2 local-frame blocks, and the audited V3 direct energy–force configurations. In each case, the DSL program has the same declared architecture-level computation and input/output semantic types as the corresponding construction. Repetition of an audited block remains representable whenever adjacent block interfaces match.

Proof. The encodings are constructive and follow the computation order of each architecture. For V1, replace each irrep-wise linear map, tensor-product message, invariant-attention operation, gate, reduction, residual, and readout by the corresponding typed primitive or audited macro described above. Their external representation, carrier, and frame types match, so the translated block is well typed. For the inspected V2 block, insert the explicit global-to-edge-frame conversion, the supported $SO(2)$ and spherical-activation path, and the inverse conversion; the restored global type then connects to reduction and residual composition. For V3, traverse the normalized direct-model graph in topological order and replace the input embedding, frame rotations, attention, spherical FFN, normalization, and two output heads by their registered typed counterparts. The audited normalized

graph discharges the required port equalities at every step. Connecting translated blocks with equal external types, or repeating such blocks, therefore yields the stated complete DSL programs. □

Proposition 2 is intentionally configuration bounded. V1 support is established for selected frozen programs, V2 support is block-level, and V3 support is tied to the audited direct-model schema and configurations used here. The proposition concerns representability in the typed language; executable equivalence additionally depends on the semantics-preserving-lowering premise of Proposition 1.

B Campaign Evidence and Protocol

Round one. The deterministic-chooser campaign generated 60 children over five mutation families (56 actions on the seed) under the frozen online search protocol: batch size 8, seed 201, 20,000 steps, the QM9 α official split, training on a fixed, hash-pinned *quarter* of the training molecules (27,500 of 110,000), evaluation every 10 epochs, and pre-/post-training symmetry audits on every child, at roughly 2.4–2.7 GPU-hours per child. Table 8 reports every child with its action and final validation MAE. Because the search protocol consumes eight times fewer examples per step than the paired protocol (batch 8 vs. 64) over a quarter of the data, its validation MAE sits in a different range by construction: the round-one winner (6449ec35, Table row 43) scores 0.546 under search and 0.189 under the paired protocol at the same seed and step budget. Search values therefore rank candidates within a campaign only; they are never mixed with paired-protocol values (roughly 0.19 in Table 3), and the test split is not read during search.

Deterministic-chooser algorithm. The round-one chooser is fully reproducible; Algorithm 1 gives the exact procedure. Parents are drawn from the archive by the OpenEvolve sampler under a fixed seed (an archive-based island sampler, external to this work); the action choice itself is a rotation over the parent's exposed certified actions with a novelty skip—no LLM, no randomness, no score feedback.

Algorithm 1 Deterministic chooser (round one)

$\mathcal{A} \leftarrow \{\text{seed program}\}$ {archive of seen architecture IDs}
for $t = 1$ **to** 60 **do**
  $G \leftarrow \textsc{OpenEvolveSample}(\mathcal{A})$ {deterministic parent draw, fixed seed}
  $M \leftarrow$ certified actions exposed by G {catalogue $\times$ applicable blocks/segments}
  $o \leftarrow (t-1) \bmod |M|$ {rotate the start by attempt number}
  for a **in** $M[o{:}] + M[{:}o]$ **do**
    $C \leftarrow \textsc{Apply}(G, a)$; **if** $\mathrm{archid}(C) \in \mathcal{A}$ **continue** {novelty skip}
    emit C; evaluate under the frozen search protocol; $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathrm{archid}(C)\}$; **break**
  end for
end for

Round two. The GLM-5.2 campaign runs the identical search protocol (batch 8, quarter subset, seed 201, 20k steps per child) while widening the catalogue to nine families (88 actions), admitting the trained baseline to the candidate pool (pool seeded with 61 entries: 60 round-one candidates plus a fresh baseline evaluation), and archiving a verbatim falsifiable hypothesis with every proposal; the frozen campaign configuration hash is `165a5cfb...c77be1`, and each child's per-run protocol and result JSON are archived in its candidate directory. The campaign is restartable from content-addressed snapshots: a platform-side task failure on September 19 at child 49 was recovered by resubmission, with the pool state, candidate programs, and hypothesis archive restored from the snapshot—the recovery path exercised in production is the one described in Section K. At the results freeze, 49 of 60 children are evaluated. The best-so-far child by search score (0.5257, sampled by the campaign's ε-greedy exploration arm with archived provenance) improves on the round-one pool best (0.5462) by 3.8%, and the best LLM-reasoned child (0.5451) improves on it as well. Its full lineage appears in Section F, and the campaign continues beyond the freeze.

Promotion ladder. All candidates train 20k steps; the top 10 by validation, the top 3 by novelty, and 2 from the lower half advance to 80k; the top 10 of those advance to 250k on the full-length cosine schedule, from which the winner is frozen. Every rung shares the same protocol-hash ladder, and candidate identity is preserved across rungs.

Parameter-count accounting. Capacity is near-constant across the campaigns but not exactly constant. Round-one candidates occupy five counts—84,961,153 (28 programs, including the seed), 84,960,897 (17), 84,960,641 (12), 84,960,385 (3), 84,960,129 (1)—because each `norm=without_norm` mutation removes one 256-parameter affine block. Round-two candidates occupy eleven counts spanning 84,960,385 to 84,963,201, as the widened catalogue's norm-insertion families add affine blocks (at most $+2{,}048$). The full span across both campaigns is 3,072 parameters, or 0.0036% of the baseline—far too small to drive the observed validation-MAE differences. The paired-protocol comparison arms are exactly matched at 84,961,153.

C Complete Round-Two Candidate Table

Table 7 lists all 49 evaluated round-two children in attempt order, with the applied action, the parent-sampling strategy, and the final search-protocol validation MAE. The sampling strategies are the five islands' parent-selection modes. Attempt 7 is the best-so-far child whose lineage appears in Section F; attempt 6 is the best LLM-reasoned child cited in Section B. Values are not comparable to Table 3.

Retrospective strategy stratification. Because every round-two child records its parent-sampling island, the table admits a stratified reading of where the campaign's gains came from. By island, the best/median/mean search scores are: elite 0.5257/0.759/0.859 (15 children), uniform 0.5639/0.683/0.688 (10), novelty 0.5451/0.729/0.892 (11), rank 0.6717/0.790/0.907 (9), cross 0.5688/0.617/0.635 (4). Two islands improve on the deterministic round-one campaign's overall median (0.711): cross (0.617) and uniform (0.683), and the campaign's best child (0.5257) improves on the round-one best (0.5462) by 3.8%, with the campaign 49/60 evaluated and continuing beyond the freeze. Attributing this best-so-far gain requires two caveats recorded in the archived provenance: the best child was sampled through the campaign's exploration arm (Section B), and the best LLM-reasoned child (0.5451) sits within noise of the round-one best (0.2% apart). At the whole-campaign level, the median and mean (0.729 and 0.822 over 49 children) do not yet improve on the deterministic campaign's (0.711 and 0.756 over 60), and the round-two tail is heavier (five children above 1.3; round-one maximum 1.276)—consistent with the catalogue widening from 56 to 88 actions enlarging the space under exploration. Under the same certified search space and per-child budget, the evidence so far attributes the round-two improvement mainly to continued exploration of the widened action space rather than to the proposer's reasoning.

Table 7: All 49 evaluated round-two children (see text for sampling-strategy key).

Attempt	Architecture	Action	Sampling	Val
1	b5ed40e163a7	block1:branch_topology=parallel_attention_ffn	uniform	0.7126
2	546405a4bba3	block0:ffn_gate:norm=with_norm	novelty	1.0823
3	d5dd0c0495c4	block4:branch_topology=parallel_attention_ffn	rank	0.8202
4	10e28e7a5732	block0:alpha:norm=without_norm	novelty	0.7127
5	97ed13e76e42	block4:ffn_residual=bypass	elite	1.7157
6	056fc77dc63c	block0:ffn_gate:norm=with_norm	novelty	**0.5451**
7	7fe3b98516a7	block5:branch_topology=parallel_attention_ffn	elite	**0.5257**
8	64b95a1758b1	block0:ffn_gate:norm=with_norm	novelty	0.8048
9	3fd7c4275492	block1:ffn_gate:norm=with_norm	uniform	0.7210
10	54bfa7c77ddf	block0:ffn_gate:norm=with_norm	cross	0.7359
11	41d44801229a	block1:ffn_gate:norm=with_norm	elite	0.9725
12	5a83776ca549	block1:ffn_gate:norm=with_norm	elite	0.7593
13	9d0404b099df	block0:ffn_gate:norm=with_norm	novelty	0.9090

(continued on next page)

(continued from previous page)

Attempt	Architecture	Action	Sampling	Val
14	3f7307c64a1d	block0:ffn_gate:norm=with_norm	uniform	0.6617
15	4c65efeca407	block1:branch_topology=parallel_attention_ffn	cross	0.6605
16	76ed891d8ed2	block7:alpha:alpha_norm_activation_order=activation_then_norm	cross	0.5688
17	e7ad53c2a522	block0:ffn_gate:norm=with_norm	uniform	0.6436
18	d04ae9916475	block2:ffn_gate:norm=with_norm	uniform	0.8588
19	8f53b10d1712	block0:ffn_gate:norm=with_norm	elite	0.8982
20	0d2155055a69	block4:attn_residual=bypass	rank	1.0652
21	5e0924afe9aa	block1:ffn_gate:norm=with_norm	novelty	0.6386
22	c1618d286406	block1:ffn_gate:norm=with_norm	elite	0.6186
23	f70aed66d6d0	block5:radial1:norm=without_norm	rank	0.6717
24	694a1fb6a813	block1:branch_topology=parallel_attention_ffn	uniform	0.6985
25	6df874e43373	block0:ffn_gate:norm=with_norm	rank	0.7592
26	fe6183231dc9	block1:ffn_scalar_gate_orientation=gate_times_activated_value	novelty	0.7291
27	a1c90b824e47	block4:ffn_gate:norm=with_norm	rank	0.7898
28	158ce09416d1	block1:radial0:norm=without_norm	novelty	1.5996
29	c09595b3ca81	block0:ffn_gate:norm=with_norm	elite	1.3616
30	b18185df0f04	block0:ffn_gate:norm=with_norm	elite	0.6554
31	0f629480ff11	block0:ffn_gate:norm=with_norm	rank	1.6463
32	f039fb7ad37e	block1:ffn_gate:norm=with_norm	elite	0.6699
33	7b28daa4ef19	block5:ffn_gate:norm=with_norm	uniform	0.6206
34	20401a52c403	block5:branch_topology=parallel_attention_ffn	rank	0.6841
35	dbf628a485e2	block0:ffn_scalar_gate_orientation=gate_times_activated_value	uniform	0.5639
36	4c2cc76f7a1e	block0:ffn_gate:norm=with_norm	cross	0.5732
37	7554fb3d396d	block1:ffn_residual=bypass	elite	0.8204
38	813ed9805323	block0:ffn_scalar_gate_orientation=gate_times_activated_value	elite	0.8437
39	91eb9294a813	block6:radial1:norm=without_norm	elite	0.9680
40	1f3734a1eddd	block0:ffn_gate:norm=with_norm	elite	0.7331
41	936385523bb7	block0:ffn_gate:norm=with_norm	rank	1.0191
42	d58573f9bc2b	block1:ffn_gate:norm=with_norm	novelty	0.6602
43	65503428d56b	block0:ffn_scalar_gate_orientation=gate_times_activated_value	elite	0.7528
44	18f4fcb061f7	block4:ffn_scalar_gate_orientation=gate_times_activated_value	uniform	0.6671
45	89463813f622	block0:ffn_gate:norm=with_norm	rank	0.7085
46	ea34f8380f3d	block0:ffn_scalar_gate_orientation=gate_times_activated_value	elite	0.5824
47	2b5bb7110070	block1:ffn_gate:norm=with_norm	uniform	0.7328
48	f4d6dcea959e	block7:radial1:radial_norm_activation_order=activation_then_norm	novelty	0.6398
49	dc0842650d3e	block4:ffn_residual=bypass	novelty	1.4865

D Complete Round-One Candidate Table

Table 8 lists all 60 round-one children in generation (attempt) order, with the applied action and the final validation MAE under the search protocol. Attempt 1 is the founder mutation shared by all top candidates (Section 5.3). Values are not comparable to the paired-protocol numbers of Table 3.

Table 8: All 60 round-one children (actions follow the `block⟨j⟩:family=value` convention).

Attempt	Architecture	Action	Val
1	930bcad875ea	block0:branch_topology=parallel_attention_ffn	0.675
2	a3775adf39c6	block0:radial0:norm=without_norm	1.276
3	375347c900df	block0:radial0:radial_norm_activation_order=activation_then_norm	0.600
4	76002b16fadb	block0:radial1:radial_norm_activation_order=activation_then_norm	0.634
5	6fda7e39748c	block0:alpha:alpha_norm_activation_order=activation_then_norm	0.598
6	77f4a91e74af	block0:alpha:alpha_norm_activation_order=activation_then_norm	0.562
7	bf5cc5d7ff20	block0:ffn_scalar_gate_orientation=gate_times_activated_value	0.687
8	446edb62aeb2	block1:radial0:norm=without_norm	0.802
9	5e8a623aa699	block1:radial1:norm=without_norm	0.630
10	cccaa996d11d	block1:radial1:norm=without_norm	0.571
11	4a146036ff45	block1:radial1:norm=without_norm	0.575
12	bec564cf7497	block1:ffn_scalar_gate_orientation=gate_times_activated_value	0.768
13	f203bc27c3ce	block1:ffn_scalar_gate_orientation=gate_times_activated_value	0.719
14	b214328c0e7c	block2:branch_topology=parallel_attention_ffn	1.057
15	08050c8606cb	block2:radial1:norm=without_norm	0.572
16	b01cade381c3	block2:radial1:norm=without_norm	0.902
17	1be2b1c7ddf3	block2:radial1:norm=without_norm	0.622
18	d618c83a4f5f	block2:radial1:radial_norm_activation_order=activation_then_norm	0.800
19	17431fe50fde	block2:ffn_scalar_gate_orientation=gate_times_activated_value	0.592
20	6f7b1eb46a63	block2:ffn_scalar_gate_orientation=gate_times_activated_value	1.003
21	aa2dd40e20b2	block3:branch_topology=parallel_attention_ffn	0.822
22	28d9b317de49	block3:branch_topology=parallel_attention_ffn	0.601
23	fd6229399d5a	block3:radial0:radial_norm_activation_order=activation_then_norm	0.666
24	7258861c505e	block3:alpha:alpha_norm_activation_order=activation_then_norm	0.641
25	3dfd65e6cfd3	block3:radial1:radial_norm_activation_order=activation_then_norm	0.961
26	364e851ddc38	block3:alpha:alpha_norm_activation_order=activation_then_norm	0.577
27	856ca1ca1a2c	block4:branch_topology=parallel_attention_ffn	0.857
28	61b3fdf9620d	block4:radial0:norm=without_norm	0.677
29	434eeb6533a7	block4:radial1:norm=without_norm	0.956
30	527e56a91a12	block4:radial1:radial_norm_activation_order=activation_then_norm	0.813
31	162e6071fc06	block4:radial0:radial_norm_activation_order=activation_then_norm	0.655
32	8f58221e727a	block4:radial1:norm=without_norm	0.649
33	46a623a9f05d	block5:branch_topology=parallel_attention_ffn	0.707
34	112fe3a86203	block5:branch_topology=parallel_attention_ffn	0.582
35	d756aa87064b	block5:radial1:norm=without_norm	0.847
36	371a9e4a0a69	block5:radial0:norm=without_norm	0.552
37	10aed4e32c08	block5:radial1:norm=without_norm	0.667

(continued on next page)

(continued from previous page)

Attempt	Architecture	Action	Val
38	5ced1eb16b2b	block5:radial1:norm=without_norm	1.103
39	ce3549c77ba6	block5:radial1:radial_norm_activation_order=activation_then_norm	0.767
40	a9415953b4e5	block6:radial0:norm=without_norm	0.709
41	2218a90e965b	block6:radial0:norm=without_norm	0.828
42	184562249567	block6:radial0:norm=without_norm	0.897
43	6449ec35b889	block6:radial0:radial_norm_activation_order=activation_then_norm	0.546
44	8dae7e898fb8	block6:radial1:norm=without_norm	0.794
45	965295dce4d2	block6:radial1:radial_norm_activation_order=activation_then_norm	0.925
46	1d4ea66a99b6	block6:radial1:norm=without_norm	0.673
47	73583a83e3d7	block7:radial0:radial_norm_activation_order=activation_then_norm	0.601
48	94deda6c6e5a	block7:radial1:norm=without_norm	0.867
49	45f8a4432ed9	block7:radial0:radial_norm_activation_order=activation_then_norm	0.684
50	42a69ef2cecb	block7:radial0:norm=without_norm	0.603
51	eef8ae3ea6b8	block7:radial0:radial_norm_activation_order=activation_then_norm	0.725
52	51eec144674b	block7:radial1:radial_norm_activation_order=activation_then_norm	0.712
53	3f8b8e717194	block7:radial1:radial_norm_activation_order=activation_then_norm	1.001
54	e2ea04513fb7	block0:branch_topology=sequential_attention_then_ffn	0.795
55	5050923817e1	block0:branch_topology=sequential_attention_then_ffn	0.807
56	76a7abb24803	block7:ffn_scalar_gate_orientation=gate_times_activated_value	0.856
57	49e89df1958a	block0:branch_topology=sequential_attention_then_ffn	0.890
58	70a9723161ad	block0:ffn_scalar_gate_orientation=gate_times_activated_value	0.808
59	87798fe54031	block0:ffn_scalar_gate_orientation=gate_times_activated_value	0.604
60	2e0d5a6ef8e5	block0:ffn_scalar_gate_orientation=gate_times_activated_value	1.275

Across the 60 children, the five mutation families account for: radial-norm presence (`norm=without_norm`; 22 children, best 0.552), radial norm/activation ordering (`radial_norm_activation_order`; 15, best 0.546—the round-one winner), block branch topology (`branch_topology`; 10, best 0.582), FFN scalar-gate orientation (`ffn_scalar_gate_orientation`; 9, best 0.592), and α-channel norm/activation ordering (`alpha_norm_activation_order`; 4, best 0.562).

E Round-Two Hypothesis Archive (Excerpts)

Every round-two proposal carries a falsifiable hypothesis archived verbatim; training either supports or falsifies it. The excerpts below (translated from the archived Chinese) illustrate the three behaviors referenced in Section 3: evidence-conditioned rationale, explicit falsification criteria, and composition of hypotheses. Outcomes are attached after the fact.

A supported hypothesis (056fc77d, outcome: supported, val 0.5451).

> "Inserting LayerNorm before the FFN scalar gate activation in Block 0 constrains the gating logits to a stable range, so gate values saturate or truncate near zero less often [...] Refuted if the child's validation MAE does not fall below the parent's 0.5516, or if training becomes unstable or diverges." The rationale cites the evidence pool: the parent (371a9e4a, MAE 0.5516) is 0.0054 from the best (6449ec35, 0.5462), and LayerNorm is already deployed on the radial and attention-logit paths but not on the FFN gate.

A rejected hypothesis (c1618d28, outcome: rejected, val 0.6186).

> "The same operator inserted at Block 1 [...] should reduce validation MAE below 0.55, testing whether the gate-normalization benefit transfers across depth." The child scored 0.6186 against a parent at 0.5923: the benefit is position-specific, and the archive records the rejection.

A composed hypothesis (ea34f838, outcome: rejected, val 0.5824).

> "LayerNorm stabilizes the logit scale and the gate re-orientation restores the gradient path [...] Refuted if the child's MAE does not beat the parent's 0.5733." Both components had support elsewhere in the pool; the composition scored 0.5824 and was rejected—composed support does not transfer, and the archive records this as well.

F THE ROUND-TWO BEST-SO-FAR LINEAGE

Table 9 traces the round-two campaign's best-so-far child (7fe3b985) back to the baseline. Two structural facts are worth noting. First, its chain begins with the same founder mutation as every round-one top candidate (the block-0 parallel rewiring), and four of its five subsequent ancestors are round-one pool entries—the GLM campaign builds on the round-one pool rather than restarting from the seed. Second, the winning step is a second parallel rewiring, applied to block 5; topology rewirings preserve parameter count, so its -512 delta accumulates entirely from the two `norm=without_norm` removals early in its lineage (Section B).

Table 9: The round-two best-so-far lineage. Val is the search-protocol validation MAE (not comparable to paired-protocol values). Mutation names abbreviated; full action strings follow Table 8.

Step	Architecture	Mutation	Val	Params
0	178046bd (baseline)	—	0.645	84,961,153
1	930bcad8	block0:topology=parallel (founder)	0.675	84,961,153
2	a3775adf	block0:radial0:norm=without	1.276	84,960,897
3	1be2b1c7	block2:radial1:norm=without	0.622	84,960,641
4	51eec144	block7:radial1:order=act_then_norm	0.712	84,960,641
5	87798fe5	block0:gate=times_value	0.604	84,960,641
6	**7fe3b985 (best-so-far)**	block5:topology=parallel	**0.526**	84,960,641

G CERTIFIED PATCH ACCEPTANCE

Section 3 describes the evolution loop; Algorithm 2 gives the acceptance path every proposal takes. Every rejection is archived with its reason, and no GPU time is spent before acceptance.

H SPEED BENCHMARK DETAILS

The throughput comparison in Section 4.2 runs both static backends on a single RTX 3080 for 600 optimizer steps after warm-up, compilation time excluded, under the production training configuration. Table 10 consolidates the measurements and the memory-feasibility contrast.

Algorithm 2 Certified patch acceptance

Require: well-typed parent program P; catalogue action a; hypothesis h (archived verbatim)
construct patch $\delta \leftarrow \mathrm{build}(a)$ with declared pre-/postconditions
if the preconditions of δ fail on P **then**
  return reject: archive (δ, reason); re-sample the parent
end if
$P' \leftarrow \delta(P)$
type-check P': every primitive input must satisfy its interface contract
run the equivariance and dimension audit on P' (rotations, translations, permutations of a probe batch)
canonicalize P' and derive its architecture hash
if the hash matches an already-evaluated candidate **then**
  return reuse the cached evaluation; no retraining
end if
emit the AOT-compilable backend plan and run the pretraining numerical-health check
return accept: train P' under the frozen protocol; the outcome supports or falsifies h

Table 10: Compilation throughput audit (600 optimizer steps, single RTX 3080, compile time excluded).

Configuration	Steps/s	Relative
Production static (AOT only)	0.915	1.00×
+ certified fast paths	—	1.58×
+ shared Wigner-D construction	1.708	1.87×
Dynamic execution at batch 8	exceeds 10 GB; does not complete	

I Safety Experiment Details

The interface stress test in Section 4.1 uses the same GLM-5.2 model (temperature 0.7, max tokens 4000) with identical architectural context across all three arms. The free-editing arm receives the complete model code (∼80 lines) and is asked to propose one structural modification as Python code. The uncertified-patch arm receives the same code plus a structured patch format—a JSON edit list over named nodes, ports, and attributes with free values (rewires, parameter sets, norm toggles, topology switches)—and its proposals are parsed and applied without type checking or precondition verification, mirroring graph/config-based proposal formats that guarantee structure but not semantics. The typed-patch arm receives the same code plus the 88-action mutation catalogue and selects an action that the compiler verifies. The middle arm decomposes the end-to-end gap: its funnel shows how much of the free-editing failure rate is recovered by the structured representation alone, and how much of the remaining distance to 100% is closed by the certified typed interface (action-space restriction together with static certification).

Static structural analysis. The structural-analysis stage is a semantic check on representation contracts, not a test of what our DSL can express. It flags four violation classes, each with a representation-theoretic justification that is independent of any infrastructure: (i) raw positions used directly as linear features (positions transform affinely under translation, so no map reading them can be equivariant); (ii) pointwise nonlinearities applied outside invariant scalar channels; (iii) flattening or reshaping an irrep-valued tensor (interleaving coordinates of different ℓ destroys the $D^{(\ell)}$ block structure); (iv) component-wise products of vector-valued tensors (no equivariant bilinear map acts coordinate-wise). Concretely, of the 600 free-editing proposals, 103 pass the finite-sample equivariance probe, and 102 of those are rejected by the structural analysis: 95 for irrep flattening and 10 for component-wise vector products (three proposals carry both flags). The check is conservative by design (a textual pattern on the emitted code may in principle over-flag), but it never rejects on grounds of expressibility; none of the four classes mentions the DSL or its action space. The matched-intention control below quantifies exactly how conservative it is.

Matched-intention control. To separate the interface effect from proposal-difficulty effects, we re-ran the three-arm stress test with fixed intentions: 300 architectural intentions sampled uniformly from the 88-action catalogue (seed 42), each implemented three times by the same GLM-5.2 with the same context—once as free Python code, once as an untyped JSON patch, and once as a typed catalogue selection. With the intention held fixed, the untyped-patch arm verifies 192/300 (64.0%, Wilson 95% CI [58.4, 69.2]) versus 300/300 for the typed arm: a 36-point gap attributable to the interface alone, with 104 of the 108 untyped failures genuine semantic failures—all 78 probe failures enable a LayerNorm over hidden features containing $\ell{=}1$ channels, and 16 of the 26 runtime failures rewire to sources that parse but are unroutable at runtime (the remaining four are parse or schema failures). The typed arm additionally recovers the correct catalogue action for 297/300 intentions (99.0%); the three misses are within-family near-misses (radial MLP 1 stated, radial MLP 0 selected). The control also sharpens the free-arm reading of Table 2: the pattern-based textual check flags any `flatten`, and the reference implementation itself contains a legal scalar-times-vector `flatten` that all 300 free-arm responses retain, so the static-clean rate of the free arm (0/300 here; 1/600 in Table 2) is a conservative bound imposed by the check rather than evidence about representation-type violations. The informative free-arm numbers under a fixed intention are instead the executable rate (269/300) and the probe pass rate (265/300, 88.3%; maximum deviation 6.1×10^{-5}). Free-form implementation of a fixed intention is therefore not dominated by syntax or runtime errors; the decisive contrast is semantic, and it is between untyped and typed structure.

J Evidence Boundaries

Each measurement supports a specific claim and no others; we state the boundaries explicitly.

- The long-horizon comparison (Section 5.2) is between single trajectories. It establishes that the evolved architecture's best-so-far crosses the reference run's best full-schedule value at 63% of the optimization steps; it does not establish a seed-averaged margin; per-step throughput is measured (within 0.7% across seven paired runs, Section 5.2), but peak memory and FLOPs are not.
- The three-seed test comparison (Section 5.1) establishes a lower mean and a consistent direction (two of three seeds); it does not establish a statistically decisive win. Seed 201 also served as the search seed—every search child trained at seed 201—so only seeds 202/203 are fully held out from architecture selection; the per-seed columns of Table 3 should be read with that asymmetry in mind. All long-horizon trajectories (Section 5.2) are seed-201 single trajectories.
- The 20k comparisons in Table 3 and the founder ablation (Table 4a) use the upstream-implementation baseline (0.191 on α, s201), whereas Table 5 shows the DSL-compiled seed program scoring 0.186 under the identical protocol and budget. The 20k-scale margins are therefore implementation-relative rather than a clean architecture-only ablation: against the DSL-compiled parent, TypeEvo's α value (0.189) is 1.6% worse rather than 1% better, and the compilation-path difference (0.005) exceeds the s201 margin in Table 3 (0.002)—so that comparison alone does not isolate an architecture effect from the compilation path. The s202/s203 margins (0.026, 0.014) exceed the measured compilation effect, but no DSL-compiled-seed control exists at those seeds to confirm the compilation effect is similarly sized there. The long-horizon reference run (Section K) is likewise upstream-based, with no DSL-compiled-seed long-horizon control.
- The interface stress test (Section 4.1) establishes an end-to-end contrast between two proposal interfaces; it does not isolate the type checker from the action-space restriction.
- The multi-target transfer (Section 5.5) is single-seed against a matched baseline; it does not establish benchmark state-of-the-art on any target, and the rMD17 result is mixed.

K Reproducibility

Every run is identified by its protocol hash, architecture hash, and archived artifacts. Protocols are frozen as JSON; field-level identity was verified programmatically. The paired protocol `paired-formal-v2@1` has protocol SHA-256 `0a389916...303f8a` over the frozen JSON (seed 201, batch 64, AdamW, learning rate $5{\times}10^{-4}$ with 8,595 warmup steps on the 515,700-step cosine, 110,000/10,000 split with split SHA-256 `22a2054b...be30f`, evaluation every 1,719

steps); every paired table entry is reconstructible from its (architecture hash, this protocol hash) pair. Search children archive their per-run `protocol.json` and `result.json` in their candidate directory (`<architecture-id>/seed201_steps20000_<run-hash>/`), including the lowering plan, symmetry reports, and charged GPU seconds.

Reference-run provenance. The full-length reference trajectory used in Section 5.2 is our own training of the unmodified EquiformerV3 architecture under the identical v2 protocol infrastructure as every paired arm in this paper: seed 201, batch size 64, QM9 with a 110,000/10,000 train/validation split, 515,700-step cosine schedule, radius 5.0 Å. The training code is built on the public EquiformerV3 implementation at commit `a7300c58df683dc99cb48027d5bfd4c887486c48`, with local modifications on top (no official QM9 training recipe is published upstream, so the batch size, schedule, and graph-construction settings are our own, not a released configuration). It is a single completed run (515,700/515,700 steps; best validation MAE 0.06629 at step 484,758), not averaged across seeds—consistent with the single-trajectory caveat in Section J.