MOLECULARIZING MODERN AI MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Molecular computation offers chemical parallelism and direct operation in biochemical environments, but modern neural networks are not written for resource-limited molecular substrates. We study *molecularization*: a learning-and-compilation transformation of pretrained neural modules into resource-aware molecular programs. A molecularized student adapts to non-ideal primitives; molecular architecture search (MAS) proposes structures under compiler-level budgets; an independent validation split selects between MAS and matched saliency candidates; hard commitment is followed by recovery; and the committed program is lowered toward autonomous CRN/DSD/DNA execution. Across Qwen3-0.6B, 1.7B, and 4B, one frozen protocol over 27 model–budget operating points yields zero observed selector regret relative to a post-hoc evaluation oracle, with two switches away from MAS at extreme 1.7B budgets. Frozen WikiText choices also match the post-hoc candidate oracle at all six evaluation-only LAMBADA probes. Matched SmolLM2 ablations separate substrate-adaptation and post-commit-recovery gains, while Mamba transfers the adapt–commit–recover principle beyond Transformers. The compiler further lowers phased blocks to one-pot DSD/DNA control, where sequence-resolved mass-action simulations quantify autonomous clock cost. Overall, molecularization jointly optimizes task fidelity, molecular resources, structure, and autonomy rather than treating chemistry as post-processing.

1 INTRODUCTION

Modern AI models are optimized for digital accelerators: standard implementations assume high-precision arithmetic, freely reusable memory, explicit sequencing, and deployment objectives such as FLOPs or parameter count. Molecular computation obeys a different contract. Chemical species are state, reactions are operators, kinetics introduce non-ideal dynamics, fuel is finite, and execution is naturally concurrent. Translating a pretrained neural network into molecules is therefore not a matter of replacing matrix multiplication with a biochemical lookup table; it is a *compiler problem coupled to learning*. Appendix B reviews the broader physical motivations–parallelism, compute/state co-location, chemical free energy, density, autonomy, and biochemical interfaces–together with their scaling limits.

Molecular programming has established that chemical reaction networks (CRNs) can express rich computation and can be systematically realized with DNA strand displacement (DSD) (Soloveichik et al., 2010; Qian & Winfree, 2011; Chen et al., 2013; Srinivas et al., 2017). DNA neural circuits further demonstrated molecular pattern classification and winner-take-all inference (Qian et al., 2011; Cherry & Qian, 2018). These systems, however, are typically designed for a molecular algorithm from the outset. In the opposite direction, modern model compression and neural architecture search optimize digital deployment objectives (Hinton et al., 2015; Liu et al., 2019; Molchanov et al., 2019; Frantar & Alistarh, 2023; Sun et al., 2023). Neither line directly answers a question that becomes central when AI is executed molecularly: *how should a pretrained modern neural block change when the target is a non-ideal, resource-limited chemical system that must execute autonomously?*

We call this cross-layer transformation *molecularization*: the pipeline maps a digital teacher to molecular primitives, adapts them, searches structural gates under molecular budgets, then commits and recovers. Because local structural objectives need not predict whole-model quality, an independent validation split selects fully built candidates before untouched evaluation.

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.

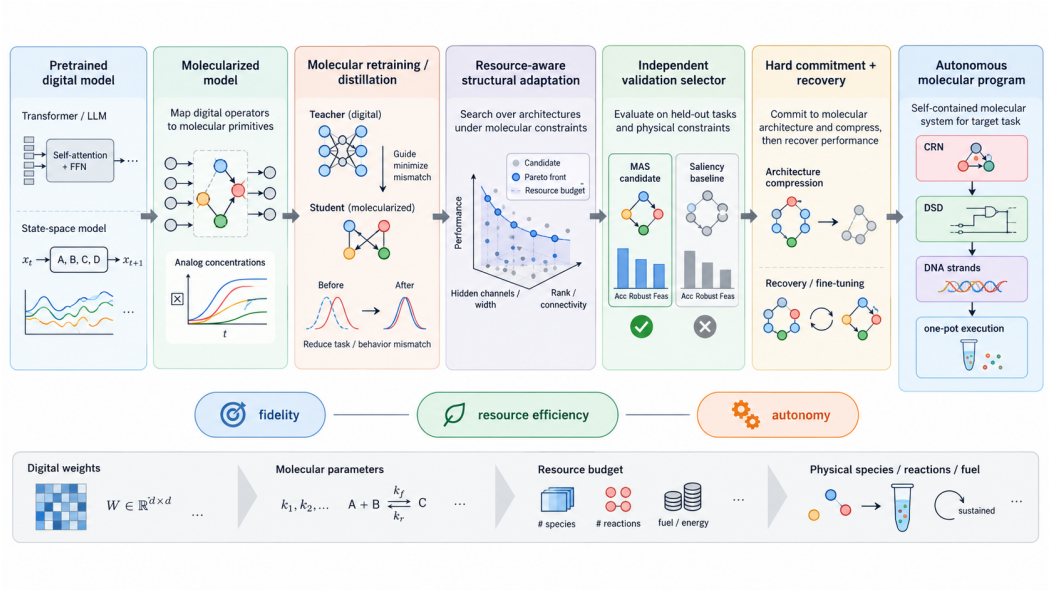

Figure 1: Conceptual view of molecularization from a pretrained digital model through substrate adaptation, resource-aware structural candidates, independent candidate selection, hard commitment and recovery, and autonomous CRN/DSD/DNA execution.

The central departure is bidirectional coupling: molecular non-idealities define what must be adapted; compiler costs reshape structure; hard commitment creates a second recovery problem; and autonomous execution trades external interventions for internal control reactions. The relevant Pareto frontier is therefore task fidelity versus molecular resources and autonomy under a physically realizable execution policy.

Our experiments span complementary axes. SmolLM2 (Ben Allal et al., 2025) provides matched-treatment attribution of adaptation, commitment, and recovery. Under one frozen protocol, Qwen3-0.6B, 1.7B, and 4B (Yang et al., 2025) yield 27/27 selector–oracle matches across a dense 90–10% selected-MLP sweep; the frozen WikiText (Merity et al., 2017) choices also match all six evaluation-only English-LAMBADA probes (Paperno et al., 2016). Mamba (Gu & Dao, 2024) tests architecture transfer beyond Transformers.

We finally connect the ML/compiler pipeline to autonomous execution: SwiGLU and Mamba are lowered from externally staged computations into one-pot programs, and sequence-resolved DSD co-design reaches 5.92 phase-progress cycles with 23/24 bounded perturbation cases passing.

Our contributions are:

- a bidirectional learning–compiler molecularization pipeline in which substrate error changes the learned student and compiler-level molecular costs change the structure being learned, followed by hard commitment and recovery;
- a candidate-selection protocol that prevents evaluation leakage when locally optimized MAS is not the whole-model winner;
- cross-scale and cross-architecture evidence on pretrained Qwen3, SmolLM2, and Mamba, including a dense 27-point Qwen3 molecular-budget sweep; and
- a compiler path from modern neural blocks to autonomous one-pot CRN/DSD execution, with explicit validation boundaries and sequence-resolved control modeling.

2 Related Work

Molecular programming and DSD. CRNs abstract programmable chemical dynamics and can be systematically realized in DNA with strand displacement (Soloveichik et al., 2010; Zhang &

Winfree, 2009; Qian & Winfree, 2011; Chen et al., 2013). Visual DSD and CRN++ raise the level of abstraction (Lakin et al., 2011; Vasic et al., 2018), while enzyme-free systems demonstrate autonomous nonlinear DNA dynamics (Srinivas et al., 2017). We instead start from pretrained neural modules and jointly optimize adaptation, structure, and molecular resources across the hierarchy.

Molecular neural computation. DSD networks have implemented Hopfield-like computation, linear-threshold networks, and large winner-take-all classifiers (Qian et al., 2011; Cherry & Qian, 2018); recent reviews survey molecular classifiers and diagnostic potential (Kieffer et al., 2023). We instead molecularize selected operators in pretrained Transformer/SSM language models under explicit molecular budgets and non-idealities.

Distillation, differentiable search, and pruning. Knowledge distillation constrains a student with a teacher (Hinton et al., 2015), differentiable architecture search relaxes discrete choices (Liu et al., 2019), and structured pruning estimates neuron/channel importance before commitment (Liu et al., 2017; Molchanov et al., 2019); SparseGPT and Wanda extend pruning to modern LLM scales (Frantar & Alistarh, 2023; Sun et al., 2023). Molecularization reuses these ideas but judges structure by molecular compiler costs and autonomous physicalization, not only parameter/FLOP sparsity. Appendix A gives detailed comparisons.

Cross-layer distinction. Molecular compilers usually fix the logical CRN/program before physical lowering, while neural compression/search keeps digital operator semantics fixed as networks shrink. Molecularization couples the directions: primitive error changes the student, compiler costs reshape structure, hard commitment creates a recovery problem, and autonomy internalizes scheduling as chemistry. Whole-model selection resolves conflicts between local molecular objectives.

3 MOLECULARIZATION

Molecularization is the learning-aware transformation of a pretrained digital computation into a molecular program whose fidelity, structure, and execution policy are jointly constrained by the target chemistry. Let f_θ denote a frozen digital teacher and $\tilde{f}_{\phi,z}$ a molecularized implementation with adaptable parameters ϕ and structural gates z. At a high level we seek

$$\min_{\phi,z} \ \mathcal{L}_{\text{task}}(\tilde{f}_{\phi,z}) + \lambda_{\text{fid}}\mathcal{L}_{\text{fid}}(f_\theta, \tilde{f}_{\phi,z}) \quad \text{s.t.} \quad \mathcal{R}_{\text{mol}}(z) \le B, \tag{1}$$

where $\mathcal{R}_{\text{mol}}$ is a compiler-level molecular resource model, not a generic parameter or FLOP count. We retain the surrounding model, replace selected modules, and report whole-model NLL while tracking fidelity and resource changes through molecularization, adaptation, search/selection, commitment/recovery, and autonomous lowering.

What is specific to molecularization? The method differs from ordinary compression in four coupled ways. *First*, the substrate changes operator semantics before compression, requiring adaptation even at full budget. *Second*, structure is priced in compiler-level species, reactions, and fuel rather than only parameters or FLOPs. *Third*, hard commitment creates a second function shift and recovery stage. *Fourth*, execution policy is part of the program: one-pot autonomy replaces external sequencing with clock/control chemistry whose resource and robustness costs are explicit. Thus molecularization is learning/compiler co-design, not post-hoc biochemical code generation.

Mechanism-level reference. Table 1 previews the sequence on an eight-channel SwiGLU block. Molecular retraining cuts substrate-induced relative-ℓ_2 error by 43.0%; hard 50% commitment halves reactions but creates a second fidelity loss; recovery removes most of that loss without restoring channels or reactions. The pretrained studies use the same decomposition.

1. Molecularized student and substrate adaptation. Selected neural operators are replaced by molecularly representable approximations, exposing a substrate mismatch even before any structural compression. We adapt only on a calibration split $\mathcal{D}_c$ using a mixture of local module fidelity and teacher/task matching,

$$\mathcal{L}_{\text{adapt}} = \lambda_{\text{loc}}\mathcal{L}_{\text{local}} + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{KL}}T^2 D_{\text{KL}}(p_\theta^{(T)} \| p_\phi^{(T)}). \tag{2}$$

Table 1: Mechanism reference for molecularize–adapt–search–commit–recover. Counts are compiler/model quantities.

Stage	Hidden	Rel. ℓ_2	Species	Reactions
Molecularized	8	0.06428	96	288
Molecular retraining	8	0.03662	96	288
Soft MAS search	soft	0.05112	–	–
Hard 50% commitment	4	0.09417	56	144
Post-commit recovery	4	0.04128	56	144

The implementation retains the best calibration state, including the step-zero state, clips gradients, and rolls back destructive updates, isolating substrate loss recoverable before structural compression.

Why adapt before and after commitment? Appendix G formalizes an idealized local-to-global argument. If layer ℓ incurs molecularization error ε_ℓ and downstream maps are K_j-Lipschitz, then

$$\|\tilde{f}(x) - f(x)\| \le \sum_{\ell=1}^{L} \varepsilon_\ell \prod_{j=\ell+1}^{L} K_j, \qquad |\Delta \ell_{\mathrm{CE}}| \le 2\|\Delta z_{\mathrm{logit}}\|_\infty. \tag{3}$$

Thus local primitive error can amplify at the output, while hard commitment can add a fresh term. This motivates pre-search adaptation and post-commit recovery, tested in Tables 2 and 6.

2. Molecular-resource-aware structural candidates. For hidden channel i, relaxed gate $z_i \in [0, 1]$ carries compiler-estimated marginal costs in species, reactions, fuel, or another resource. A generic resource functional is

$$\mathcal{R}_{\mathrm{mol}}(z) = \sum_i z_i \big(w_s c_i^{(s)} + w_r c_i^{(r)} + w_f c_i^{(f)}\big). \tag{4}$$

MAS optimizes gates for molecular fidelity under budget B while a saliency candidate is built at the same hard budget. For Qwen3, uniform marginal channel costs give MAS a neutral keep-fraction initialization rather than a saliency ranking. MAS is therefore a *candidate generator*, not an assumption that its local objective must dominate whole-model evaluation. Figure 6 in Appendix D shows broader CRN/DSD motif choices; main experiments restrict search to channel/width.

3. Independent selection without evaluation leakage. Calibration $\mathcal{D}_c$, selection $\mathcal{D}_s$, and final evaluation $\mathcal{D}_e$ are disjoint. Candidate parameters and structures may depend on $\mathcal{D}_c$ only. Once candidates are fully built, the selector chooses

$$k^\star = \arg\min_{k \in \{\mathrm{MAS,saliency}\}} \mathrm{NLL}(f_k; \mathcal{D}_s), \qquad \text{report } \mathrm{NLL}(f_{k^\star}; \mathcal{D}_e). \tag{5}$$

A post-hoc oracle on $\mathcal{D}_e$ diagnoses regret but never changes candidates; the guard matters most at extreme budgets, where local objectives can mispredict whole-model quality.

Why select only after candidate construction? For K fixed candidates and bounded held-out losses, finite-class uniform deviation scales as $\sqrt{\log K/n}$ (Appendix G). Here $K = 2$, so $\mathcal{D}_s$ chooses among preconstructed candidates rather than continuing search. Because causal-LM NLL is unbounded and natural-text samples are dependent, this is motivation rather than a guarantee; the 27/27 WikiText and 6/6 LAMBADA oracle agreements are the empirical checks.

4. Hard commitment and post-commit recovery. Relaxed gates are projected to a discrete architecture satisfying the requested budget, causing a discontinuous function shift. We therefore run a short recovery phase with the committed mask fixed; Section 4 shows that recovery becomes increasingly valuable as the budget tightens.

5. Autonomous molecular lowering. The committed computation is lowered to a CRN with mass-action dynamics $\dot{x} = Sv(x; k)$. Rather than an external scheduler, one-pot fusion introduces shared

Table 2: Pre-compression substrate adaptation. "Recovered" is the fraction of the naive molecularization NLL penalty removed by molecular retraining.

Model	Digital	Molecularized	Retrained	Recovered
SmolLM2-135M	3.2492	4.6043	3.6354	71.5%
Qwen3-0.6B	3.2364	6.0533	4.9989	37.4%
Qwen3-1.7B	2.8791	4.4210	3.4759	61.3%
Qwen3-4B	2.6846	4.4359	3.0456	79.4%

Table 3: Guarded Qwen3 candidate selection across the dense 90–10% budget sweep. Jaccard reports the MAS/saliency selected-channel overlap at the 90% and 10% endpoints.

Model	Budgets	MAS chosen	Selector=oracle	Jaccard 90%→10%
Qwen3-0.6B	9	9	9/9	0.806→0.061
Qwen3-1.7B	9	7	9/9	0.808→0.065
Qwen3-4B	9	9	9/9	0.811→0.132

reservoirs and molecular clock/control state q, making sequencing part of the coupled dynamics. The CRN is then refined toward DSD motifs, physical DNA domains/strands, finite fuel inventories, waste, and leakage. Autonomy and raw reaction count are distinct: internalized scheduling can remove stage boundaries while adding clock/gating chemistry.

Resource-axis discipline. A "budget" has family-specific semantics: Qwen3 uses retained hidden-channel fraction of selected MLPs, SmolLM2 uses reaction-count fraction, and Mamba reports architecture-specific molecular-resource, latency, and fuel fractions. We never merge them onto one numerical axis. Appendix F gives exact formulations, Appendix G the idealized arguments, and Appendix C a CRN/DSD/DNA primer.

4 Experiments

We organize the evidence around four causal questions. Qwen3 (Yang et al., 2025) uses WikiText-2 (Merity et al., 2017) train/validation/test as calibration/selection/evaluation, with 16/16/256 sequences of 128 tokens (32,768 untouched evaluation tokens) per operating point. The selected-MLP protocol and hyperparameters are frozen across Qwen3-0.6B, 1.7B, and 4B. SmolLM2 (Ben Allal et al., 2025) supplies matched-treatment attribution, Mamba (Gu & Dao, 2024) tests architecture transfer, and the autonomy study follows the compiler into CRN/DSD/DNA realization. Appendix H gives complete tables, while Appendix I gives physical-programming details.

Q1: Does molecularization create a recoverable substrate mismatch? Table 2 shows a substantial naive molecularization penalty for every pretrained model and consistent adaptation gains before structural compression. Qwen3 recovers 37.4–79.4% of that penalty across scales and SmolLM2 recovers 71.5%, making substrate adaptation a first-class stage.

Q2: Can molecular resources be reduced without treating MAS as an oracle? Across 27 Qwen3 model–budget points, the selector matches the post-hoc evaluation oracle with zero observed regret: MAS is selected at 25 points and saliency at the two extreme Qwen3-1.7B budgets where MAS has higher NLL on untouched evaluation. Because all cells share one frozen split, this is a consistency audit rather than 27 independent replications. Table 3 also shows growing MAS/saliency structural disagreement as budgets tighten. The retained-channel budget applies only to eight MLPs: 50% retention removes 14.3%/14.3%/11.1% of decoder-MLP hidden-path weights in Qwen3-0.6B/1.7B/4B; even 10% retention removes 25.7%/25.7%/20.0%. Appendix H gives full scope accounting.

On evaluation-only English LAMBADA (Paperno et al., 2016), the frozen WikiText guard matches the post-hoc oracle in all six 75% and 50% operating points without using LAMBADA for training or selection (Table 4); Appendix H.6 gives the full protocol and base-model NLLs.

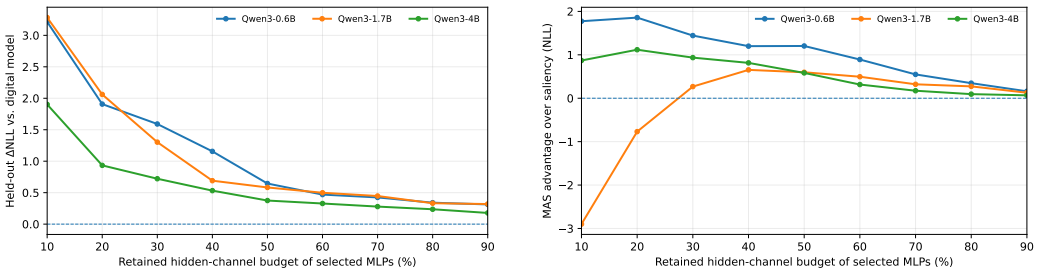

Figure 2: Qwen3 dense selected-MLP budget sweep. **Left:** guarded held-out NLL degradation. **Right:** MAS advantage over matched saliency; the Qwen3-1.7B crossover at 20% and 10% triggers the independent selector to choose saliency. Budgets are retained hidden-channel fractions of selected MLPs.

Table 4: Evaluation-only LAMBADA candidate-choice transfer. The guard is fixed by WikiText validation before this probe; it chooses MAS in all six cells and matches the post-hoc LAMBADA candidate oracle. Regret is guarded NLL minus the better candidate NLL.

Model	Budget	MAS NLL	Saliency NLL	Regret
Qwen3-0.6B	75%	4.1320	5.1156	0.0000
Qwen3-0.6B	50%	4.4407	6.5955	0.0000
Qwen3-1.7B	75%	4.0169	4.3915	0.0000
Qwen3-1.7B	50%	4.1462	4.8115	0.0000
Qwen3-4B	75%	3.7171	4.0751	0.0000
Qwen3-4B	50%	3.8291	4.8625	0.0000

Table 5: SmolLM2 fidelity/resource trajectory after molecular retraining, structural commitment, and recovery.

Reaction budget	Species	Reactions	Final NLL
100%	529,920	159,805,440	3.6354
75%	414,720	119,854,080	3.9076
50%	299,520	79,902,720	4.2927
25%	184,320	39,951,360	4.9496

Q3: What happens after hard commitment, and does the principle transfer? SmolLM2 separates structural choice from recovery under matched treatment. Recovery gains grow as the reaction budget tightens (0.146/0.301/0.986 NLL at 75/50/25%). At 50%, the recovered committed model reaches NLL 4.2927 versus 4.6043 for naive full-resource molecularization, outperforming direct molecularization with half the modeled reactions.

Table 5 makes resources explicit: species fall with retained structure, reactions scale exactly with the budget, and recovered MAS stays below naive full-resource molecularization through 50%.

With matched retraining/recovery (Table 6), MAS slightly outperforms saliency at all budgets and beats random/magnitude choices; omitting recovery becomes increasingly costly as commitment tightens.

On Mamba, adaptation plus recovery cuts NLL at 75/50/25% width from 10.59/13.92/40.90 to 4.60/7.05/10.10 at unchanged committed resource fractions (Figure 3). Adding rank compression lowers modeled molecular/latency/fuel fractions further but raises NLL (Table 7).

Architecture-transfer check. Mamba's digital teacher has NLL 3.40430; a shape-preserving reload matches within 2.5×10^{-5} NLL of the digital teacher, ruling out checkpoint reconstruction as the compression gap. At 50% width, the adapted model reaches NLL 7.0477 at molecular/latency/fuel

Table 6: SmolLM2 matched-treatment held-out NLL. Random entries are means over three selector seeds; deterministic methods yield one value per operating point. Retrained rows share the same pre-adaptation and post-commit recovery unless noted.

Method	75%	50%	25%
Random pruning only	6.8510	13.7627	14.1771
Magnitude pruning only	15.7682	16.4347	15.8534
MAS only	7.6526	14.4528	16.8426
Retrain + random + recovery	4.2076	5.3444	10.9607
Retrain + magnitude + recovery	4.7668	5.7284	7.9208
Retrain + saliency + recovery	3.9135	4.3325	5.1246
Retrain + MAS, no recovery	4.0535	4.5934	5.9357
Retrain + MAS + recovery	**3.9076**	**4.2927**	**4.9496**

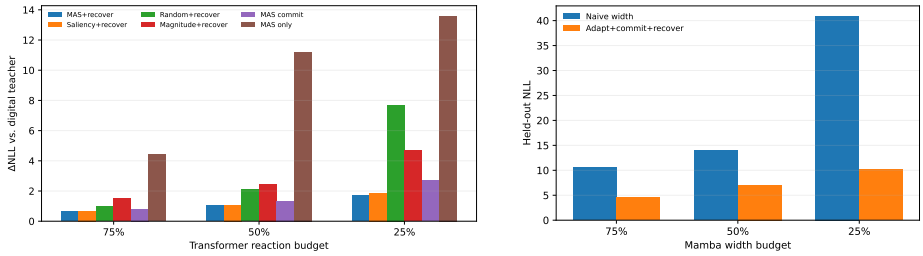

Figure 3: Causal attribution and architecture transfer. **Left:** SmolLM2 matched-budget attribution; recovery matters increasingly at tight budgets. **Right:** naive Mamba width truncation versus adapt–commit–recover at the same committed widths.

Table 7: Adapted Mamba fidelity/resource trade-off after post-commit recovery. Molecular, latency, and fuel quantities are fractions of the uncompressed Mamba reference and are not numerically comparable to Qwen3 budgets. Full naive/adapted results are in Appendix H.

Structure	Budget	NLL	Molecular	Latency	Fuel
Width	75%	4.5999	0.7875	0.8875	0.7786
Width	50%	7.0477	0.5250	0.7750	0.5132
Width	25%	10.1029	0.2625	0.6625	0.2536
Width+rank	75%	6.1340	0.5858	0.8098	0.5746
Width+rank	50%	9.1851	0.3410	0.6394	0.3287
Width+rank	25%	31.8185	0.1458	0.4886	0.1383

fractions 0.525/0.775/0.513; width+rank reaches 0.341/0.639/0.329 at NLL 9.1851, making the additional resource–fidelity trade-off explicit.

Cross-family takeaway. The transferable result is stage ordering, not a common resource axis: Transformer and Mamba both benefit from adaptation before hard structural change and recovery afterward, while compiler costs remain architecture-specific. We therefore compare fidelity changes within each family rather than directly comparing resource fractions across families.

Compiler resources are not parameter sparsity. At 50% Mamba width, width-only retains 0.6496 of parameters but 0.5250 of modeled molecular resources; width+rank retains 0.5865 of parameters but only 0.3410 of modeled molecular resources. This divergence is why the structural objective uses compiler costs rather than parameter count; complete resource accounting is in Appendix H.

Table 8: SwiGLU resource trajectory. Commitment reduces payload chemistry; autonomous fusion adds explicit clock/control chemistry while removing external scheduling boundaries.

Representation	Hidden	Species	Reactions	Programs	Ext. bound.
Uncompressed molecular	8	96	288	5 stages	4
50% committed	4	56	144	5 stages	4
Autonomous one-pot	4	64	488	1	0

Table 9: Sequence-resolved clock progression. Generic replenishment retains fuel but does not sustain oscillation; the catalytic topology changes the modeled dynamical regime.

Design	Cycles	Sust.	Phase mass	Fuel ret.	Leak sink
Finite DSD clock	0.00339	0	0.0607	–	0.3901
Generic regeneration	0.02284	0	0.0042	0.8286	0.5167
Catalytic oscillator	5.91868	5	0.9775	0.6246	0.0567

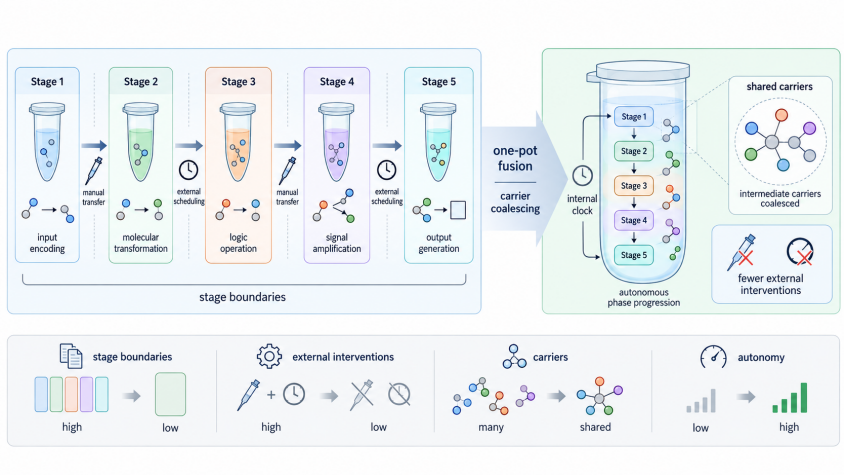

Figure 4: Externally staged execution versus autonomous one-pot fusion. Autonomy internalizes sequencing and can add clock/gating chemistry rather than monotonically reducing raw reaction count.

Q4: What does autonomous molecular execution add—and cost? Commitment halves SwiGLU hidden width and payload reactions; recovery repairs much of the fidelity loss, while one-pot lowering removes four external stage boundaries but adds clock/gating reactions.

Beyond Table 8, compiler/model evidence gives $7 \to 1$ staged-to-one-pot Mamba execution and $3 \to 1$ graph fusion. The DNA realization has 36 domains, 198 strands, 36 complexes, and nine motif families; 23 of 24 bounded catalytic perturbations pass. Evidence spans compiler/model counts, sequence-resolved mass-action dynamics, and sequence realization. Table 10 adds a bounded cycle/leak redesign; Appendix I gives complete physicalization statistics.

Table 10: Evidence-conditioned catalytic redesign under its bounded perturbation set. The selected redesign improves the conservative cycle/leak trade-off without materially changing the fuel reserve; percentiles summarize deterministic perturbations, not confidence intervals.

Design	Cycles 5th pct.	Fuel 5th pct.	Leak 95th pct.
Baseline catalytic	4.9215	0.5348	0.07835
Selected redesign	5.2717	0.5326	0.04654

5 Discussion and Limitations

The experiments support a cross-layer pipeline, not a universal structural winner. MAS is a candidate generator: saliency wins at the two tightest Qwen3-1.7B budgets, making independent whole-model selection part of the method. The evaluation-only LAMBADA probe supports frozen candidate-choice transfer to one additional English corpus, not broad task or domain generalization. Post-commit recovery addresses the separate function shift caused by discrete commitment.

Design lesson. The evidence separates three failure modes: substrate mismatch before compression, candidate misranking during structural search, and function shift after hard commitment. They call for different controls—adaptation, held-out selection, and recovery—while autonomous lowering introduces a fourth axis, control chemistry. This separation is what distinguishes molecularization from one-shot pruning or post-hoc compilation.

Resource semantics remain architecture- and compiler-specific: Qwen3 budgets cover eight selected MLPs, SmolLM2 and Mamba use different compiler axes, and one-pot autonomy adds explicit control chemistry. Appendix J consolidates the corresponding scope, transfer, and physical-evidence boundaries.

6 Conclusion

Molecularization treats molecular deployment as joint learning and compilation rather than fixed-network translation. Across Transformer and Mamba studies, substrate adaptation repairs pre-compression mismatch, compiler-aware candidates expose molecular resource trade-offs, independent selection guards against structural-objective failures, and post-commit recovery repairs discrete commitment. Autonomous lowering shows that removing external sequencing introduces control chemistry with fuel and leak costs while providing a sequence-resolved route toward physical realization. Together, these results define a path toward molecular programs jointly optimized for task fidelity, physical resources, and autonomy.

AI use statement

OpenAI ChatGPT assisted software implementation, debugging, experiment orchestration, literature search, plotting-code drafting, and manuscript editing. The authors reviewed all AI-assisted text and citations, executed and tested AI-assisted code, and take responsibility for the final manuscript and reported results. Figures 1, 4, 6, and 7 are conceptual illustrations generated with OpenAI ChatGPT's image-generation capability under author direction and curation; quantitative figures are generated from recorded experiment outputs. All experiments reported in this work were conducted by the authors, primarily using NVIDIA A40 GPUs.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big—data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013. doi: 10.1038/nnano.2013.189.

Kevin M. Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370–376, 2018. doi: 10.1038/s41586-018-0289-6.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337, 2023.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977. doi: 10.1021/j100540a008.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling*, 2024. Outstanding Paper; arXiv:2312.00752.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. arXiv:1503.02531.

Coline Kieffer, Anthony J. Genot, Yannick Rondelez, and Guillaume Gines. Molecular computation for molecular classification. *Advanced Biology*, 7(3):2200203, 2023. doi: 10.1002/adbi.202200203.

Matthew R. Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22): 3211–3213, 2011. doi: 10.1093/bioinformatics/btr543.

Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961. doi: 10.1147/rd.53.0183.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017. doi: 10.1109/ICCV.2017.298.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. arXiv:1609.07843.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019. doi: 10.1109/CVPR.2019.01152.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011. doi: 10.1126/science.1200520.

Lulu Qian, Erik Winfree, and Jehoshua Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011. doi: 10.1038/nature10262.

Eryk J. Ratajczyk, Jonathan Bath, Petr Šulc, Jonathan P. K. Doye, Ard A. Louis, and Andrew J. Turberfield. Controlling DNA–RNA strand displacement kinetics with base distribution. *Proceedings of the National Academy of Sciences*, 122(23):e2416988122, 2025. doi: 10.1073/pnas.2416988122.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010. doi: 10.1073/pnas.0909380107.

Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358(6369):eaal2052, 2017. doi: 10.1126/science.aal2052.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Marko Vasic, David Soloveichik, and Sarfraz Khurshid. CRN++: Molecular programming language. In *DNA Computing and Molecular Programming*, volume 11145 of *Lecture Notes in Computer Science*, pp. 1–18. Springer, 2018. doi: 10.1007/978-3-030-00030-1_1.

Hang Xu, Yifan Yu, Peixin Li, Shaowei Liu, Xuehui Yan, Zhaoyu Zhou, and Ye Tian. DNA computing: DNA circuits and data storage. *Nanoscale Horizons*, 10:3204–3217, 2025. doi: 10.1039/D5NH00459D.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

David Yu Zhang and Erik Winfree. Control of DNA strand displacement kinetics using toehold exchange. *Journal of the American Chemical Society*, 131(47):17303–17314, 2009. doi: 10.1021/ja906987s.

A Detailed Related Work

A.1 Chemical reaction networks as programs

A foundational abstraction in molecular programming is to treat concentrations as state variables and chemical reactions as an executable language. Soloveichik, Seelig, and Winfree showed that DNA strand-displacement systems can approximate arbitrary coupled CRN kinetics (Soloveichik et al., 2010). This result provides a bridge from formal mass-action models to sequence-addressable molecular implementations. Chen et al. subsequently demonstrated programmable DNA chemical controllers derived from formal CRNs, including an analog consensus algorithm (Chen et al., 2013). CRN++ approaches the problem from the language side, compiling imperative-style programs into deterministic mass-action reaction networks (Vasic et al., 2018).

Our approach shares the idea that CRNs are an intermediate representation, but its source program is a pretrained AI module. This changes the optimization problem. A neural compiler must preserve task behavior under approximation, allocate structure under molecular resource budgets, and decide where adaptation is preferable to faithful digital emulation. We therefore place a differentiable learning loop above the CRN/DSD compiler rather than treating CRN synthesis as the terminal step.

A.2 DNA strand displacement and design toolchains

Toehold-mediated strand displacement provides a tunable primitive for enzyme-free molecular circuits; toehold exchange enables reversible reactions and broad kinetic control (Zhang & Winfree, 2009). Qian and Winfree scaled seesaw strand-displacement logic to multilayer digital circuits and introduced an abstraction hierarchy with automated compilation (Qian & Winfree, 2011). Visual DSD provides a language and analysis environment supporting deterministic/stochastic simulation and formal state-space construction (Lakin et al., 2011). These works motivate our explicit separation between CRN semantics, DSD lowering, domain/sequence realization, and kinetic evidence.

A key distinction is the level of physical validation. Moving from a CRN/DSD program toward physical operation additionally requires sequence assignment, unwanted-interaction control, finite inventories, leak management, kinetic calibration, and autonomous regeneration. We therefore record these layers separately rather than collapsing them into a binary notion of "molecularly executable."

A.3 Autonomous molecular dynamics

Srinivas et al. demonstrated an enzyme-free DNA oscillator generated from a CRN-level dynamical specification, showing that programmable DSD can support autonomous nonlinear dynamics (Srinivas et al., 2017). Their work is especially relevant to our one-pot objective: autonomous execution requires more than combinational signal propagation because state, fuel, restoration, and timing must coexist in one chemical environment. Our clock studies follow a failure-driven co-design path—generic finite-resource realizations do not sustain cycles, whereas an oscillator-specific catalytic topology does so in the sequence-aware model. This complements the experimental oscillator demonstration of Srinivas et al. while focusing here on compiler and sequence-resolved modeling evidence.

A.4 Molecular neural networks and classifiers

Qian, Winfree, and Bruck implemented neural-network computations with DSD cascades, including associative memory and linear-threshold circuits (Qian et al., 2011). Cherry and Qian substantially scaled molecular pattern classification using winner-take-all DNA networks, experimentally recognizing noisy 100-bit patterns (Cherry & Qian, 2018). Reviews such as Kieffer et al. survey the broader landscape of molecular classification and diagnostic computation (Kieffer et al., 2023). These systems demonstrate that molecular substrates can implement nontrivial inference, but they generally design the molecular computation directly. We instead ask how an already-trained modern neural network should be transformed when the target substrate is molecular and non-ideal.

A.5 Distillation and adaptation to constrained students

Knowledge distillation transfers functional behavior from a teacher to a constrained student (Hinton et al., 2015). Here, the "student" differs not simply in width or depth but in its primitive semantics: nonlinearities, resource constraints, and autonomous execution are changed by physicalization. Distillation is therefore used as substrate adaptation before and after discrete structural commitment. The post-commit step is analogous in spirit to fine-tuning after pruning, but its purpose is specifically to recover from an discrete molecular architecture choice.

A.6 Differentiable architecture search and structured pruning

DARTS relaxes discrete architecture choices into continuously optimized variables (Liu et al., 2019). Network slimming learns channel-level sparsity and then prunes channels (Liu et al., 2017); Taylor-criterion pruning estimates a unit's effect on loss to guide structural removal (Molchanov et al., 2019). These methods optimize digital deployment metrics or architecture quality. Our method reuses differentiable gates but attaches them to molecular compiler costs, allowing a selected hidden channel to contribute explicit species/reaction/resource cost.

At LLM scale, SparseGPT demonstrates accurate one-shot pruning with second-order reconstruction (Frantar & Alistarh, 2023), while Wanda uses weight magnitude modulated by input activations (Sun et al., 2023). These methods motivate strong magnitude/activation baselines, but they target weight sparsity rather than a molecularly realizable hidden-channel architecture. Our dense sweep shows that the resource-aware MAS candidate is usually preferred, but also that an independent selector should remain free to choose a simpler saliency candidate under extreme compression.

A.7 Modern neural blocks

Our Transformer experiments focus on gated MLPs related to GLU/SwiGLU variants (Shazeer, 2020) in Qwen3-scale pretrained language models (Yang et al., 2025). Our architecture-transfer experiments additionally target Mamba selective state-space models (Gu & Dao, 2024). These blocks are useful stress tests because they expose different state and gating structures while remaining central to current sequence modeling.

A.8 CLOSEST CONCEPTUAL DISTINCTION

The closest ingredients to our approach exist separately: molecular languages compile CRNs to DNA, molecular neural circuits implement hand-designed inference, distillation adapts constrained students, NAS differentiates architectural choices, and pruning removes digital parameters or channels. The distinguishing objective here is the joint path from a pretrained modern neural operator to a *molecular-resource-aware, discrete, and autonomously executable program*, with learning before commitment, compiler-derived resource terms during search, independent whole-model candidate selection, and explicit separation among CRN semantics, DSD/DNA realization, external software verification, and physical validation.

B WHY MOLECULAR PROGRAMS MAY MATTER FOR FUTURE AI

This section gives the longer-horizon motivation for molecular programs by identifying physical properties that make molecular computation scientifically interesting, including prospective physical energy efficiency, the regimes in which those properties could matter, and the bottlenecks that must be solved before they become system-level advantages. Recent reviews emphasize chemical parallelism, molecular-scale information density, and the potential for low-energy operation as recurring motivations for DNA computation, while also stressing substantial challenges in kinetics, synthesis, readout, and reliability (Xu et al., 2025). Classical DSD work demonstrates that nontrivial logic, neural computation, feedback control, and autonomous dynamics can be programmed with nucleic-acid reactions (Qian & Winfree, 2011; Qian et al., 2011; Chen et al., 2013; Srinivas et al., 2017).

B.1 NATIVE CONCURRENCY RATHER THAN AN INSTRUCTION STREAM

A well-mixed CRN does not execute reactions through a global program counter. For reaction r with mass-action rate $v_r(x)$, every enabled channel contributes simultaneously to

$$\dot{x} = \sum_{r=1}^{R} s_r v_r(x), \tag{6}$$

where s_r is the stoichiometric update vector. In the stochastic description, the next event is drawn from all enabled reaction propensities, with total event rate equal to their sum (Gillespie, 1977). This physical concurrency is attractive for workloads that can be expressed as many local interactions, reductions, associative updates, or population-level inference. It should not be confused with a guaranteed wall-clock speedup: reaction constants, diffusion, concentration, signal restoration, and the depth of a reaction cascade still determine latency. Qian and Winfree's multilayer DSD circuits, for example, demonstrated scalable composition but retained propagation delay across circuit depth (Qian & Winfree, 2011).

B.2 STATE AND COMPUTATION ARE PHYSICALLY CO-LOCATED

In a molecular program, concentrations and molecular complexes are both *state* and the substrates on which computation acts. There is no conceptual separation analogous to fetching a tensor from DRAM, moving it through an arithmetic unit, and writing it back. This creates an in-memory-computing-like opportunity: information can remain encoded in molecular populations while reactions transform it locally. DNA additionally offers extremely dense molecular information storage (Xu et al., 2025). However, storage density is not the same as computational density. A realizable DSD gate may require auxiliary strands, fuels, waste products, separators, and sequence-orthogonality margin, so practical compute density must be evaluated at the compiled circuit level rather than inferred from bits per nucleotide.

B.3 CHEMICAL FREE ENERGY AS A COMPUTING RESOURCE

Toehold-mediated strand displacement can proceed spontaneously under designed free-energy gradients, and enzyme-free circuits can run without an electronic clock driving every logical transition (Zhang & Winfree, 2009; Srinivas et al., 2017). This creates plausible routes to chemically

powered computation, computation with a low electronic duty cycle, or embedded computation in environments where chemical free energy is already available. A useful bookkeeping abstraction is

$$W_{\mathrm{chem}} \approx \sum_r N_r^{\mathrm{fuel}} \, \Delta\mu_r, \tag{7}$$

where N_r^{fuel} is the number of fuel-consumption events associated with reaction family r and $\Delta\mu_r$ is the corresponding chemical-potential drop. Equation 7 is not an end-to-end energy model: synthesis, purification, thermal control, sensing, readout, regeneration, and waste handling can dominate a practical system. Landauer's bound constrains logically irreversible information erasure in any physical computer (Landauer, 1961), but it does not imply that present DNA circuits operate near that limit or outperform electronics. This is why the main paper reports fuel and resource proxies separately from physical energy and makes no joule-level efficiency claim.

B.4 Volume, molecule count, and stochasticity create a scaling tradeoff

For concentration c_i in reaction volume V, the expected molecule count is

$$N_i = c_i V N_A, \tag{8}$$

where N_A is Avogadro's constant. At fixed concentrations, deterministic mass-action equations are invariant to a uniform change in volume, but absolute molecule count and fuel inventory scale linearly with V. Under an ideal Poisson counting approximation,

$$\frac{\sigma_{N_i}}{\mathbb{E}[N_i]} \approx \frac{1}{\sqrt{N_i}} = \frac{1}{\sqrt{c_i V N_A}}. \tag{9}$$

Thus miniaturization can reduce absolute material/fuel use while increasing intrinsic stochastic fluctuations. Conversely, larger volumes can suppress relative copy-number noise but consume more molecules and may aggravate mixing and transport. This simple relation illustrates why molecularized AI has a physical scaling axis absent from conventional parameter-count analysis.

B.5 Autonomy and direct biochemical interfaces

Molecular programs can sense and transform molecular inputs without first digitizing them. Programmable DNA controllers and classifiers provide examples in which chemical signals are directly coupled to downstream molecular decisions (Chen et al., 2013; Kieffer et al., 2023). Autonomous clocks and dynamical systems further show that sequencing and feedback can be embedded into the chemistry itself (Srinivas et al., 2017). For future AI systems, this suggests application niches in biochemical sensing, smart diagnostics, responsive materials, closed-loop therapeutics, and hybrid electronic–molecular systems where the cost of converting every molecular event into a digital signal would be undesirable. These are prospective application classes, not capabilities demonstrated by the present language-model experiments.

B.6 Why scaling is difficult

The same properties that make molecular programs attractive also create difficult scaling limits:

- **Sequence orthogonality and crosstalk.** More logical signals require more domains whose unintended interactions remain sufficiently weak.
- **Leak and error accumulation.** Small off-pathway rates can integrate over long autonomous executions, especially when catalytic amplification is present.
- **Kinetic depth.** Molecular reactions can be highly concurrent, but long dependency chains still incur sequential propagation delay.
- **Fuel, waste, and regeneration.** Autonomous programs consume chemical free energy and accumulate products; sustained operation requires explicit resource and reset strategies.
- **Stochasticity and transport.** Low-copy-number fluctuations, diffusion, and imperfect mixing become important as volumes shrink or spatial organization grows.
- **Synthesis, initialization, and readout.** End-to-end cost includes preparing many species at controlled concentrations and observing outputs, not just the in-solution reaction energy.

These limits motivate hierarchical compilation, shared-carrier/fuel design, spatial organization, better sequence-level kinetic models, and hybrid digital–molecular control rather than a simple claim that molecular hardware should replace digital accelerators.

B.7 Why these opportunities motivate molecularization

The potential advantages above are useful only if the neural computation can change to match the substrate. A digital model optimized purely for FLOPs or parameters may compile into a chemically expensive network with excessive species, reactions, fuel, leakage, or scheduling overhead. Molecularization therefore treats task fidelity, compiler resources, and autonomous execution as coupled design variables. The main paper's resource-aware search, post-commit recovery, selector guard, and one-pot lowering are intended as first steps toward this cross-layer objective, not as proof that the long-horizon advantages have already been realized experimentally.

C Fundamentals of Molecular and DNA Programming

This appendix gives a self-contained overview of the fundamentals needed to read the molecular-programming portions of the paper: chemical reaction networks (CRNs), mass-action kinetics, concentration encodings, and DNA strand displacement (DSD). The examples are intentionally schematic: they explain the programming model and mass-action equations used throughout the paper, while the molecularization pipeline may realize the same logical operation using a more detailed elementary CRN and a larger set of DSD gate, fuel, and waste species.

C.1 Concentrations as variables and reactions as instructions

A CRN represents program state by nonnegative molecular concentrations. If species X has concentration $x(t) = [X](t)$, then a reaction changes $x(t)$ continuously in time. A general reaction r is written

$$\sum_i \nu_{ir}^- X_i \xrightarrow{k_r} \sum_i \nu_{ir}^+ X_i, \tag{10}$$

where ν_{ir}^- and ν_{ir}^+ are reactant and product stoichiometries and k_r is the reaction-rate constant. Under deterministic well-mixed mass-action kinetics,

$$v_r(x) = k_r \prod_i x_i^{\nu_{ir}^-}, \qquad \dot{x} = S\, v(x), \qquad S_{ir} = \nu_{ir}^+ - \nu_{ir}^-. \tag{11}$$

Thus the "program" is a set of reactions, while the numerical execution is the coupled ordinary differential equation (ODE) induced by their stoichiometry and rates. Unimolecular rate constants have units of inverse time; bimolecular rate constants have concentration^{-1} time^{-1}. We use this CRN level as an intermediate representation before progressively more physical DSD/DNA realization.

C.2 Small computing primitives from mass-action kinetics

Table 11 shows several elementary motifs. These are not claimed to be unique or optimal DNA implementations; they provide intuition for how computation emerges from reaction dynamics.

The first two examples have closed-form solutions. For decay, $x(t) = x_0 e^{-kt}$. For transfer with $y(0) = 0$, $y(t) = x_0(1 - e^{-kt})$, so the input amount is progressively moved to the output. The bimolecular reaction $X + Y \to Z$ instead has instantaneous rate kxy: doubling either input doubles the production rate, which makes bimolecular chemistry a natural source of multiplicative or coincidence-sensitive interactions.

C.3 Simulated mass-action trajectories of representative primitives

Figure 5 makes the dynamics concrete by integrating Equation 11 for three explicit CRNs. These deterministic ODE simulations use normalized concentrations and time for visualization, with pedagogical rate constants chosen to expose the motif dynamics.

Table 11: Illustrative CRN computing primitives. $x = [X]$, $y = [Y]$, and so on. Reactions that create an output in a logical CRN consume gate/fuel molecules after DSD realization even when that fuel is omitted from compact notation.

Primitive	Reaction and mass-action equation	Computing interpretation
Decay / cleanup	$X \xrightarrow{k} \varnothing$, $\dot{x} = -kx$	Exponential relaxation, forgetting, or cleanup.
Transfer	$X \xrightarrow{k} Y$, $\dot{x} = -kx$, $\dot{y} = kx$	Moves a concentration-encoded value into another species.
Catalytic production	$X + F \xrightarrow{k} X + Y$, $\dot{y} = kxf$, $\dot{f} = -kxf$	Preserves input X while finite fuel F is converted into output Y.
Coincidence / AND-like	$X + Y \xrightarrow{k} Z$, $\dot{z} = kxy$	Output appears only when both inputs are present; production is multiplicative in concentration.
Annihilation	$X^+ + X^- \xrightarrow{k} W$, $\dot{x}^+ = \dot{x}^- = -kx^+x^-$	Removes equal positive/negative common mode while preserving the signed difference.
Thresholding	$X + T \xrightarrow{k} W$, $\dot{x} = \dot{t} = -kxt$	Finite T removes input stoichiometrically; ideal completion leaves approximately $\max(x_0 - t_0, 0)$.

Adder by convergent transfer. We use two parallel transfer reactions,

$$A \xrightarrow{k} Y, \qquad B \xrightarrow{k} Y. \tag{12}$$

For equal rate constant k and $Y(0) = 0$,

$$Y(t) = (A_0 + B_0)(1 - e^{-kt}), \tag{13}$$

so the output asymptotically approaches the sum of the two initial input amounts. The plotted example uses $A_0 = 0.65$, $B_0 = 0.35$, and $k = 1$.

Time-windowed multiplicative readout. To expose the multiplicative factor in mass action directly, we use the compact logical reaction

$$X + W \xrightarrow{k} X + W + P. \tag{14}$$

Here X and W act catalytically, so $\dot{P} = kXW$. With constant $X = X_0$, $W = W_0$, $P(0) = 0$, and a fixed readout time T, one obtains $P(T) = kTX_0W_0$. The plot sets $k = T = 1$, $X_0 = 0.8$, and $W_0 = 0.6$, hence $P(T) = 0.48 = X_0W_0$. Equation 14 is a logical CRN shorthand: a physical DSD realization must supply gate/fuel molecules for production of P, as discussed below.

Cyclic-autocatalytic clock. For four phases with cyclic index $i \in \{0, 1, 2, 3\}$, the illustrative driven clock uses

$$C_i + C_{i+1} + D \xrightarrow{k} 2C_{i+1} + S, \qquad i + 1 \bmod 4, \tag{15}$$

where D is finite drive and S is spent carrier. The phase dynamics satisfy

$$\dot{c}_i = kd\, c_i(c_{i-1} - c_{i+1}), \qquad \dot{d} = -kd \sum_i c_i c_{i+1}. \tag{16}$$

Starting from one dominant phase and three small seeds produces successive phase takeovers while drive is consumed. The plotted example uses $(c_0, c_1, c_2, c_3) = (0.97, 0.01, 0.01, 0.01)$, $d_0 = 30$, and $k = 0.25$. This is the forward-driven idealization of the cyclic-autocatalytic clock family used by the molecularization pipeline; it is distinct from the more detailed sequence-resolved oscillator study in Appendix I.

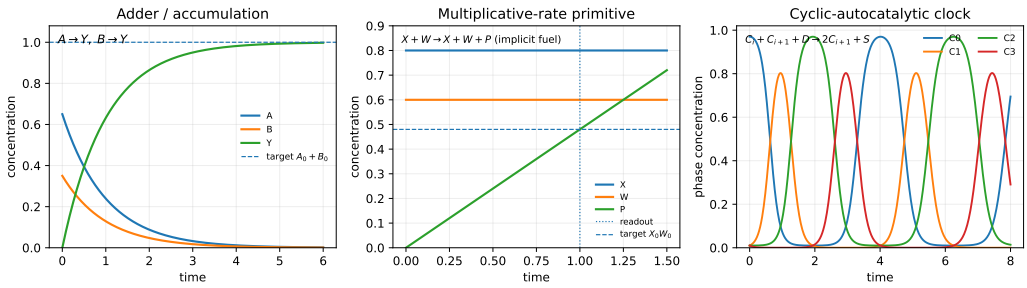

Figure 5: Deterministic mass-action trajectories for three explicit CRN computing motifs. **Left:** two transfer reactions consume A and B while the shared output Y approaches $A_0 + B_0$. **Center:** a logical catalytic production motif has rate $\dot{P} = kXW$; at the marked unit-time readout with $k = 1$, $P = X_0W_0$. **Right:** a finite-drive cyclic-autocatalytic clock produces sequential phase peaks. The trajectories are generated by numerical integration of the displayed CRNs with the parameters stated in the text.

Positive weighted accumulation. A simple logical linear-combination motif uses parallel catalytic branches,

$$X_i + F_i \xrightarrow{k_i} X_i + Y, \qquad \dot{y} = \sum_i k_i[X_i][F_i]. \tag{17}$$

Over a controlled time window, approximately buffered fuel/gate concentrations turn $k_i[F_i]$ into an effective nonnegative weight. A physical DSD implementation makes the required gates, fuels, and waste explicit rather than assuming a free source of Y.

Signed values with dual rails. Because concentrations cannot be negative, a signed scalar is commonly encoded by two nonnegative species,

$$x \equiv [X^+] - [X^-]. \tag{18}$$

Positive contributions are routed to the matching rail and negative contributions to the opposite rail. The annihilation motif in Table 11 reduces simultaneous occupancy of both rails without changing their difference, because both concentrations decrease by exactly the same amount. This representation lets molecular programs express signed linear algebra while preserving nonnegative physical concentrations.

C.4 From a logical CRN to DNA strand displacement

DNA strand displacement provides a programmable way to implement CRN-like transformations using designed DNA strands and complexes (Zhang & Winfree, 2009; Soloveichik et al., 2010; Qian & Winfree, 2011; Chen et al., 2013). A signal strand first binds through a short exposed *toehold*; subsequent branch migration replaces an incumbent strand and releases an output. Sequence domains determine which signals can interact, while toehold choice and gate design influence kinetic rates and leakage.

A useful conceptual picture is

$$\text{input signal} + \text{gate/fuel complex} \longrightarrow \text{output signal} + \text{waste}, \tag{19}$$

with a cascade of such gates implementing the logical CRN. General CRN-to-DSD constructions show that broad classes of formal reaction networks can be systematically compiled into strand-displacement systems (Soloveichik et al., 2010; Chen et al., 2013). In a physical design, one logical reaction may therefore expand into several DNA species and elementary displacement steps.

C.5 Concrete sequence-resolved strand examples

The abstract domain notation above eventually becomes literal nucleotide sequences. To make that step concrete, Table 12 gives a small subset of the sequence-resolved autonomous-clock design

used in our physicalization study. These sequences are *compiler/model design outputs*. We rename the domains descriptively here and omit repository-local identifiers; Appendix J gives the physical-validation boundary.

Table 12: Example DNA domains from the sequence-resolved realization. Toeholds are 6 nt and identity/branch-migration domains are 15 nt in this design.

Role	Domain	5′–3′ sequence
Phase-0 signal toehold	t_0	GTCAAG
Phase-0 identity	x_0	CAGCACAGAAGTTAC
Phase-1 signal toehold	t_1	GGTATG
Phase-1 identity	x_1	CTTCAGGCAATAGCA
Drive-signal toehold	t_d	CGAAGT
Drive identity	d	CCCATCAATCGTCAG
Spent-signal toehold	t_w	GAAAGT
Spent-signal identity	x_w	TGAGGAAGCGTTGCT
Forward-reaction toehold	τ_{01}	CAGAGA
Forward branch-migration	b_{01}	GGTATGTTGGCTGTA
Phase-1-reset toehold	τ_{r1}	TCAGTC
Phase-1-reset branch-migration	b_{r1}	TCAGGCTCTCGTTCG

A concentration-encoded signal is a physical single strand formed by concatenating its toehold and identity domains. For example,

$$P_0 : 5'\text{-}t_0x_0\text{-}3' = \texttt{GTCAAGCAGCACAGAAGTTAC}, \tag{20}$$

$$P_1 : 5'\text{-}t_1x_1\text{-}3' = \texttt{GGTATGCTTCAGGCAATAGCA}, \tag{21}$$

$$D : 5'\text{-}t_dd\text{-}3' = \texttt{CGAAGTCCCATCAATCGTCAG}, \tag{22}$$

$$F_{01} : 5'\text{-}\tau_{01}b_{01}\text{-}3' = \texttt{CAGAGAGGTATGTTGGCTGTA}. \tag{23}$$

Here P_0 and P_1 encode two clock phases, D is a drive signal, and F_{01} is a finite reaction-fuel strand for the forward phase transition. A second primitive—phase reset—uses an independent fuel,

$$F_{r1} : 5'\text{-}\tau_{r1}b_{r1}\text{-}3' = \texttt{TCAGTCTCAGGCTCTCGTTCG}, \tag{24}$$

illustrating how separate reactions receive orthogonal toehold/branch-migration addresses.

The forward phase-transition primitive is not implemented by the fuel strand alone. One realized gate is a five-strand complex whose component strands, in compact domain notation, are

$$\text{structural cap} : \tau_{01}b_{01}x_0d, \tag{25}$$

$$\text{phase-0 protector} : x_0, \tag{26}$$

$$\text{phase-1 payload} : t_1x_1, \tag{27}$$

$$\text{spent payload} : t_wx_w, \tag{28}$$

$$\text{gate backbone} : d^*x_0^*b_{01}^*\tau_{01}^*x_0^*t_0^*x_1^*t_1^*x_w^*t_w^*. \tag{29}$$

The star denotes Watson–Crick complementarity. For example, the structural-cap strand is

$$5'\text{-}\texttt{CAGAGAGGTATGTTGGCTGTACAGCACAGAAGTTACCCCATCAATCGTCAG}\text{-}3'. \tag{30}$$

The long complementary backbone sequesters the cap/protector/payload strands while leaving the intended interface exposed. Binding through an exposed toehold initiates branch migration, producing intermediate complexes and eventually releasing the next-phase payload while consuming finite reaction fuel and producing waste. Thus even a simple CRN-level transfer expands into multiple physical strands, duplex domains, intermediates, and fuels. This expansion is exactly why resource accounting at the DNA/DSD level can differ sharply from parameter count or FLOPs.

C.6 Why physicalization changes the computing problem

At the CRN level, Equation 11 makes a molecular program look like an ODE. A DNA realization adds constraints that have no direct analogue in a digital neural network: every signal needs an

addressable sequence/domain; gates and fuels occupy finite molecular inventory; intended and unintended hybridization compete; leak reactions can accumulate; and the same kinetic constants that control latency also affect numerical fidelity and stability. This is why we keep CRN semantics, DSD topology, DNA sequence assignment, kinetic calibration, and validation stage as separate layers instead of treating "CRN executable" and "experimentally realized" as synonyms.

C.7 Relation to the molecularization pipeline

The neural computations in the main paper are optimized before and during this lowering process. Molecular adaptation compensates for non-ideal primitive behavior; molecular architecture search changes which channels/structures should survive under compiler-level resource constraints; hard commitment creates a discrete molecular program; post-commit recovery repairs the resulting function shift; and autonomous fusion introduces chemical clock/control species so that sequencing can occur within the reaction network rather than through an external digital controller. Appendix F gives the exact optimization formulation used in the paper, while Appendix I reports the DSD/DNA modeling assumptions and validation boundaries.

D Molecular Architecture Search and Physical Realization Concepts

This section illustrates two design spaces that are easy to miss when molecularization is described only as a neural-network compression problem: (i) a single digital operator can admit several chemically distinct molecular realizations, and (ii) an abstract CRN must still be refined through strand-displacement mechanisms into explicit DNA strands. Quantitative evidence for the evaluated molecularization pipeline appears in Appendices H and I.

D.1 MAS as molecular implementation choice

A matrix–vector product provides a simple example. For

$$y_i = \sum_j w_{ij} x_j, \tag{31}$$

the same algebraic contribution can be approximated by different molecular motifs: direct catalytic accumulation, buffered or sequestration-based transfer, threshold/seesaw constructions, dual-rail signed accumulation, or multi-step intermediates. These choices need not have the same transient error, leak sensitivity, response time, number of species/reactions, fuel demand, or ease of composition into a larger network. A broader molecular architecture search can therefore introduce a discrete implementation variable $m_\ell \in \mathcal{M}_\ell$ in addition to structural gates z and adapted parameters ϕ,

$$\min_{m,z,\phi} \; \mathcal{L}_{\text{fid}}(m,z,\phi) + \lambda_s N_s(m,z) + \lambda_r N_r(m,z) + \lambda_f C_f(m,z) + \lambda_{\text{rob}}\, \mathbb{E}_\xi[\Delta_{\text{imp}}(m,z,\phi;\xi)]\,, \tag{32}$$

where N_s and N_r are compiler-estimated species and reaction counts, C_f denotes fuel or other resource demand, and Δ_{imp} measures degradation under modeled imperfections ξ. Constraints can additionally encode latency, dynamic range, orthogonality, or physical realizability. The objective above is a conceptual extension of the same compiler-aware principle used in the reported experiments.

Figure 6 visualizes this broader design space. The current pretrained-model experiments instantiate MAS over structural variables such as retained hidden channels and architecture-specific width/rank choices, with compiler-level molecular costs; the figure additionally shows the broader CRN/DSD motif space. This makes explicit why molecular architecture search can go beyond ordinary pruning: the search variable can include *how* an operation is chemically realized, not only whether a digital channel is retained.

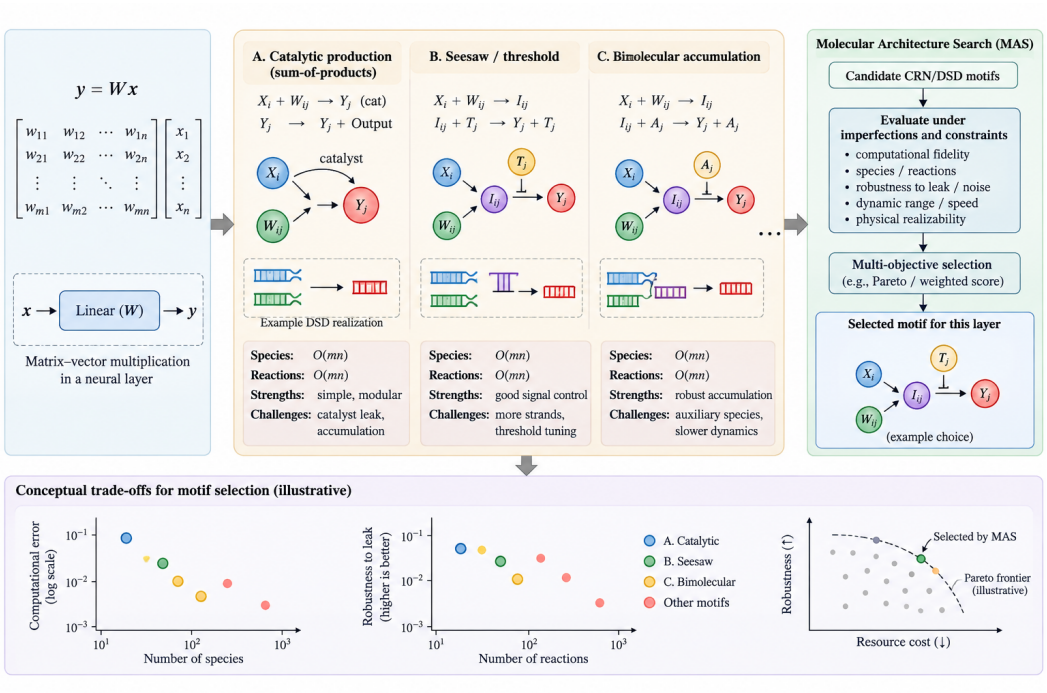

Figure 6: Conceptual molecular architecture search for a matrix–vector operation. A single digital computation may admit multiple CRN/DSD realizations with different fidelity, robustness, kinetics, species/reaction counts, fuel demand, and composability; a compiler-aware search can select among Pareto-efficient implementations. The motif equations and trade-off points illustrate the broader design space, while the pretrained-model experiments search a restricted structural subspace.

Search axes and scope. The molecular-architecture viewpoint naturally spans several levels, but the reported experiments intentionally validate them separately rather than claiming one joint end-to-end search. Table 13 summarizes that distinction.

Table 13: Illustrative MAS search axes and their status in this paper. "Reported" means directly evaluated in the manuscript; the other rows identify broader molecularization design variables without claiming a joint search over them.

Search level	Example variables	Status here
Neural structure	hidden channels, width, rank, retained blocks	Reported / quantitatively evaluated
Candidate guard	compiler-aware MAS versus saliency under the same budget	Reported; independently selected
CRN motif	catalytic, buffered, threshold/seesaw, dual-rail, multi-step	Conceptual design space
Dynamics/control	rates, fuel, gain, clock topology, one-pot scheduling	Modeled and studied separately
DNA realization	gate family, domains, orthogonality, strand/complex design	Lowered and screened separately

D.2 From reaction motifs to physical DNA strands

Figure 7 provides the complementary physical-refinement view: CRN-level transfer, catalysis, thresholding, and signed/dual-rail abstractions are mapped to strand-displacement motifs and then to sequence-bearing DNA strands. Exact sequence/domain examples are given in Appendix C.

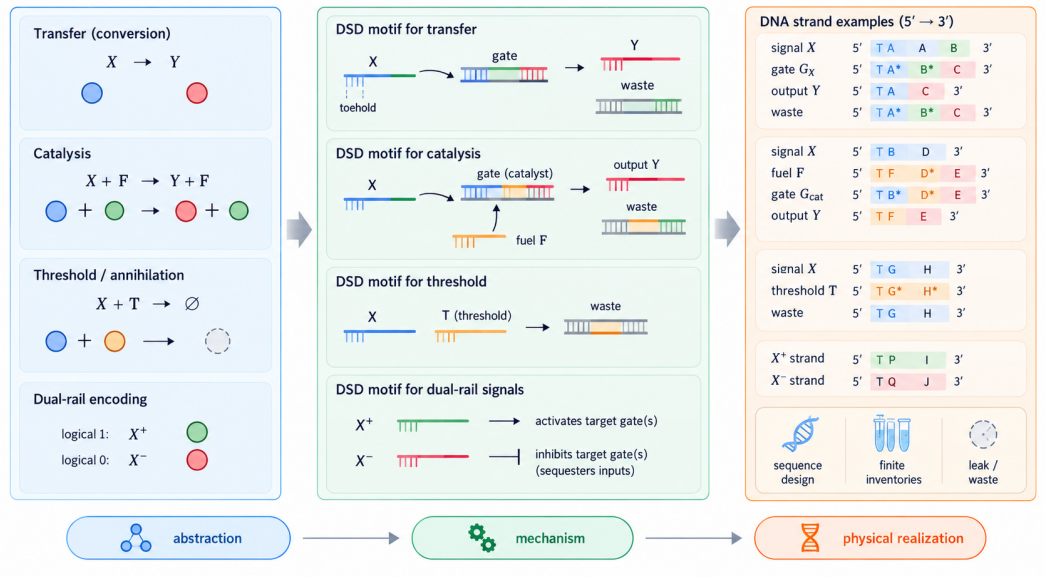

Figure 7: Conceptual refinement from CRN-level computing primitives to strand-displacement mechanisms and DNA strand realizations. Exact sequence/domain examples are given in Appendix C.

E External Software and Interoperability

The reference implementation is designed to interoperate with established numerical, machine-learning, and DNA-programming software rather than reimplement every solver or verifier. Table 14 summarizes the principal external libraries relevant to this work and distinguishes tools that directly support the reported language-model experiments from optional cross-checking or physical-modeling interfaces.

Table 14: External software used by, or interfaced with, the reference implementation. "Reported use" states whether the present paper's quantitative results depend on that tool.

Software	Role in the implementation	Reported use in this paper
PyTorch	Primary tensor/autodiff runtime for molecularized neural modules, distillation, architecture optimization, and post-commit recovery.	Used for the reported pretrained-model studies.
Hugging Face Transformers	Loads supported pretrained language-model architectures and exposes their modules for molecularization and evaluation.	Used for the Qwen3 and SmolLM2 studies.
Hugging Face Datasets	Retrieves and caches public text datasets with explicit train/validation/test splits.	Used to obtain WikiText-2 and the evaluation-only LAMBADA probe corpus.
JAX	Optional differentiable execution frontend/backend for molecular CRN programs, including JIT compilation and automatic differentiation.	Supported and conformance-tested; not required for the reported language-model results.
torchdiffeq	Optional PyTorch ODE backend for well-mixed mass-action systems with direct or adjoint differentiation.	Alternative numerical backend and semantic cross-check; not the source of the reported results.
iCRN	Optional JAX backend for differentiable chemical reaction networks. The implementation maps the supported well-mixed mass-action subset into iCRN for independent backend execution.	Used as an optional interoperability/conformance path, not as the source of the reported results.
Peppercorn	Domain-level DNA strand-displacement reaction enumeration. The implementation can serialize realized DSD complexes to Peppercorn-compatible PIL/kernel structures.	The reported oscillator structures can be exported through this interface, but no Peppercorn enumeration result is used in the reported oscillator metrics.
crnverifier	Checks formal/implementation CRN equivalence using CRN bisimulation or pathway-decomposition notions.	Optional independent verification of exported CRN abstractions; not a source of model-quality results.
Nuskell	Independent CRN-to-DSD reference compiler that can combine translation, enumeration, and CRN verification.	Reference interoperability path; the primary DSD lowering does not depend on it.
KinDA	Sequence/pathway-aware DSD kinetic analysis. The implementation can ingest externally produced effective-rate evidence with provenance.	No KinDA-derived kinetics are used for the reported oscillator.
Multistrand	Stochastic nucleic-acid kinetic simulation over secondary-structure energy landscapes; useful for displacement-time and leak characterization.	No Multistrand-derived kinetics are used for the reported oscillator.

The separation in the final column distinguishes implemented external-tool support from the evidence path exercised for each reported result. The autonomous-clock study in Appendix I uses the sequence-resolved mass-action model with explicit finite resources and leak, while Peppercorn and KinDA/Multistrand provide independent downstream enumeration and kinetic-analysis paths for further physical closure without changing the logical compiler pipeline.

F Detailed Method and Mathematical Formulation

F.1 Digital teacher and molecularized student

For an input token context x, let $f_\theta(x)$ be the frozen digital teacher and let $m_\phi(x)$ denote a module in which selected digital operators have been replaced by molecularly representable primitives. For a SwiGLU-style MLP,

$$h(x) = W_d\left[\mathrm{SiLU}(W_g x) \odot (W_u x)\right], \tag{33}$$

and the molecularized module replaces one or more primitive maps by $\widetilde{\mathrm{SiLU}}$, molecular linear maps, or resource-constrained factorizations. We retain the original surrounding model and replace only selected modules, making whole-model causal-LM NLL the final metric.

F.2 Molecular adaptation

Given calibration set $\mathcal{D}_c$, adaptation minimizes a mixture of local fidelity and task-level teacher matching,

$$\mathcal{L}_{\mathrm{adapt}}(\phi) = \lambda_{\mathrm{loc}}\,\mathbb{E}_{x\sim\mathcal{D}_c}\frac{\|m_\phi(x) - f_\theta^{(\mathrm{module})}(x)\|_2^2}{\|f_\theta^{(\mathrm{module})}(x)\|_2^2 + \epsilon} \tag{34}$$

$$+\,\lambda_{\mathrm{CE}}\,\mathcal{L}_{\mathrm{CE}}(\phi) + \lambda_{\mathrm{KL}} T^2 D_{\mathrm{KL}}\left(p_\theta^{(T)} \| p_\phi^{(T)}\right). \tag{35}$$

The implementation records the best calibration state starting with step zero, clips gradients, and rolls back if adaptation is destructive. This safeguard was crucial for the Qwen3 scale-transfer study.

F.3 Resource-aware molecular architecture search

For hidden channels $i = 1, \ldots, d$, define relaxed gates $z_i = \sigma(\alpha_i)$. Let $c_i^{(s)}$ and $c_i^{(r)}$ denote compiler-estimated marginal species and reaction costs; when used, $c_i^{(f)}$ denotes fuel or another resource cost. A generic molecular resource functional is

$$\mathcal{R}(z) = \sum_i z_i\left(w_s c_i^{(s)} + w_r c_i^{(r)} + w_f c_i^{(f)}\right). \tag{36}$$

For the current Qwen3 SwiGLU channel model, channels have uniform marginal compiler cost; the resource-aware initialization therefore starts from a neutral expected keep fraction rather than inheriting a saliency ranking. MAS minimizes

$$\mathcal{L}_{\mathrm{MAS}}(\alpha) = \mathcal{L}_{\mathrm{fid}}(\phi, \sigma(\alpha)) + \lambda_R \Psi\big(\mathcal{R}(\sigma(\alpha)), B\big), \tag{37}$$

with a hard post-search commit to the requested retained-channel count. Ψ is implemented so the expected resource remains centered near budget B during the relaxed search.

F.4 Hard commitment and recovery

Let S_B be the index set of the highest-ranked channels satisfying the hard budget. Commitment defines $z_i^\star = \mathbf{1}[i \in S_B]$. Since this projection is discrete and changes the local function discontinuously, we run a short post-commit recovery

$$\phi_B^\star = \arg\min_\phi \mathcal{L}_{\mathrm{adapt}}(\phi; z = z^\star), \tag{38}$$

again with clipping and best-state rollback. SmolLM2 attribution shows that this stage becomes increasingly important as compression becomes stronger.

F.5 Independent selector guard

Calibration $\mathcal{D}_c$, selection $\mathcal{D}_s$, and evaluation $\mathcal{D}_e$ are token-row disjoint. Candidate parameters and masks may depend on $\mathcal{D}_c$ but never on $\mathcal{D}_s$ or $\mathcal{D}_e$. After MAS and saliency candidates are fully built, the guard chooses

$$k^\star = \arg\min_{k\in\{\mathrm{MAS},\mathrm{sal}\}} \mathrm{NLL}(f_k; \mathcal{D}_s). \tag{39}$$

The final reported value is $\mathrm{NLL}(f_{k^\star}; \mathcal{D}_e)$. We compute the post-hoc evaluation oracle only as a diagnostic; it never affects $k^\star$. Across the 27-point dense Qwen3 sweep, the selection decision matches this diagnostic oracle at all 27 points.

F.6 CRN SEMANTICS AND AUTONOMOUS FUSION

A compiled CRN has species vector $x \in \mathbb{R}^n_+$, stoichiometric matrix S, and mass-action rate vector $v(x; k)$. Its continuous dynamics are

$$\frac{dx}{dt} = S\, v(x; k). \tag{40}$$

An externally staged execution introduces boundaries between molecular stages. One-pot fusion instead augments the CRN with shared reservoirs and clock/control species q so that stage-dependent reactions use rates $v_j(x, q; k)$ inside one autonomous dynamical system,

$$\frac{d}{dt}\begin{bmatrix} x \\ q \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_q \end{bmatrix} \begin{bmatrix} v_x(x, q; k) \\ v_q(x, q; k_q) \end{bmatrix}. \tag{41}$$

The clock is therefore part of the molecular program rather than an external digital scheduler.

F.7 DSD AND DNA REALIZATION

CRN reactions are lowered into DSD motifs and logical domains, then assigned physical domains/sequences. The paper's physical evidence tracks (i) value closure, (ii) strand/complex realization, (iii) sequence-risk screens, (iv) exact kernel serialization, (v) finite-resource dynamics, and (vi) robustness. For the structures discussed here, the sequence/kernel representation explicitly specifies the species, domains, complexes, and reactions needed for downstream domain-level enumeration; Appendix J summarizes the physical-validation boundary.

G THEORETICAL PERSPECTIVES AND IDEALIZED JUSTIFICATIONS

The empirical study is the primary evidence in this paper. This section records several simple idealized results that clarify why the pipeline is structured as molecularize–adapt–select–commit–recover and why autonomous fusion is evaluated using phase purity and leak. The statements are not intended as a full theory of molecular neural networks; their assumptions are made explicit so that the connection to the practical system is clear.

G.1 A FINITE-CANDIDATE RELIABILITY BOUND FOR THE SELECTOR GUARD

Suppose K fully constructed candidates are functions only of the calibration data and are therefore fixed before a held-out selection set $\mathcal{D}_s = \{X_i\}_{i=1}^n$ is examined. Let $\ell_k(X) \in [0, M]$ be a bounded per-row selection loss for candidate k, with population loss $L_k = \mathbb{E}[\ell_k(X)]$ and empirical loss $\widehat{L}_k = n^{-1} \sum_i \ell_k(X_i)$. Let

$$\widehat{k} = \arg\min_k \widehat{L}_k, \qquad k^\star = \arg\min_k L_k. \tag{42}$$

By Hoeffding's inequality and a union bound, with probability at least $1 - \delta$,

$$\max_{1 \le k \le K} |\widehat{L}_k - L_k| \le M \sqrt{\frac{\log(2K/\delta)}{2n}} \equiv \epsilon_n. \tag{43}$$

On this event,

$$L_{\widehat{k}} \le \widehat{L}_{\widehat{k}} + \epsilon_n \le \widehat{L}_{k^\star} + \epsilon_n \le L_{k^\star} + 2\epsilon_n, \tag{44}$$

so the population regret of the empirical selector is bounded by

$$L_{\widehat{k}} - \min_k L_k \le 2M \sqrt{\frac{\log(2K/\delta)}{2n}}. \tag{45}$$

The important structural point is the $\log K$ dependence: selecting among a small, preconstructed candidate set is statistically much easier than searching a large architecture space on the same held-out data. In our study $K = 2$ (MAS and saliency) after candidate construction is complete. Causal-LM NLL is formally unbounded, so Equation 45 should be read as an idealization for bounded/clipped row losses; an analogous statement follows under a sub-Gaussian tail assumption. Selection samples drawn from natural text are also not perfectly independent, reducing the effective sample size. The bound therefore motivates, but does not guarantee, low selector regret. The observed 27/27 agreement with the post-hoc evaluation oracle is empirical evidence beyond this idealized argument.

G.2 How local molecularization error can propagate to task loss

Consider a composition

$$f = F_L \circ F_{L-1} \circ \cdots \circ F_1 \tag{46}$$

and a molecularized approximation $\widetilde{f} = \widetilde{F}_L \circ \cdots \circ \widetilde{F}_1$. Assume F_ℓ is K_ℓ-Lipschitz on the relevant activation region and that

$$\|\widetilde{F}_\ell(u) - F_\ell(u)\| \le \varepsilon_\ell \tag{47}$$

for states visited by the two executions. If d_ℓ is the activation discrepancy after layer ℓ, then

$$d_\ell \le \varepsilon_\ell + K_\ell d_{\ell-1}, \tag{48}$$

and induction yields the telescoping bound

$$\|\widetilde{f}(x) - f(x)\| \le \sum_{\ell=1}^{L} \varepsilon_\ell \prod_{j=\ell+1}^{L} K_j. \tag{49}$$

Equation 49 explains why small local molecular primitive errors can still matter at whole-model scale and why adaptation before structural compression is useful: adaptation seeks to reduce the ε_ℓ terms induced by changing the substrate. Hard commitment can abruptly increase one or more ε_ℓ, motivating a second recovery stage after the discrete projection.

For a token with logit vector z and target y, cross-entropy is

$$\ell(z, y) = \log \sum_j e^{z_j} - z_y. \tag{50}$$

Its gradient has ℓ_1 norm at most 2, so by the mean-value theorem

$$|\ell(\widetilde{z}, y) - \ell(z, y)| \le 2\|\widetilde{z} - z\|_\infty. \tag{51}$$

Thus any bound on propagated logit error induces a corresponding (generally loose) bound on per-token NLL shift. This provides a formal connection between local fidelity losses used during molecular adaptation and the whole-model NLL ultimately reported in the experiments.

G.3 Why retained-width budgets produce approximately affine molecular cost

The resource objective in Equation 4 is additive at the candidate level. A simple abstraction makes the budget behavior explicit. Let a module have h optional hidden channels, fixed compiler overhead C_0, and marginal molecular cost c_i per retained channel. For binary gates z_i,

$$C(z) = C_0 + \sum_{i=1}^{h} z_i c_i. \tag{52}$$

If marginal costs are uniform, $c_i = c$, and exactly $m = \rho h$ channels are retained, then

$$C(\rho) = C_0 + \rho h c. \tag{53}$$

The variable part of cost therefore scales linearly with retained width, while fixed gate, interface, and control overhead does not disappear. This is the reason a 50% hidden-width commitment can approximately halve reaction payload while reducing species count by a smaller fraction. More detailed compilers can break additivity through carrier sharing, domain reuse, spatial organization, or autonomous control chemistry; such departures are precisely why molecular compiler estimates are more informative than parameter count alone.

G.4 IDEALIZED EQUIVALENCE OF STAGED AND ONE-POT EXECUTION

Let $f_j(x)$ be the vector field of stage $j \in \{1, \dots, J\}$. An externally scheduled implementation can be represented as a switched system

$$\dot{x}(t) = \sum_{j=1}^{J} q_j(t) f_j(x(t)), \tag{54}$$

where $q_j(t) \in \{0, 1\}$ and exactly one stage is active at each time. An autonomous one-pot realization replaces the ideal schedule by molecular gate activities $\widehat{q}_j(t) \geq 0$ and includes residual leak/control error $g(t, x)$,

$$\dot{\widehat{x}}(t) = \sum_{j=1}^{J} \widehat{q}_j(t) f_j(\widehat{x}(t)) + g(t, \widehat{x}(t)). \tag{55}$$

Assume $\sum_j \widehat{q}_j(t) = 1$, every f_j is L-Lipschitz, $\|f_j(x)\| \leq F$ on the relevant region, and $\|g(t, x)\| \leq \eta$. Gronwall's inequality gives the coarse trajectory bound

$$\|\widehat{x}(T) - x(T)\| \leq e^{LT} \left[F \int_0^T \sum_{j=1}^{J} |\widehat{q}_j(t) - q_j(t)| \, dt + \eta T \right]. \tag{56}$$

If molecular gating exactly reproduces the external schedule and leak vanishes, Equation 56 collapses to zero and the one-pot trajectory is equivalent to the ideal switched execution. In practice, phase overlap, incomplete phase progression, and leak contribute to the two error terms. This is why the autonomy study measures sustained cycles, final phase mass, leak sinks, and perturbation robustness rather than treating boundary removal alone as sufficient evidence of functional equivalence.

G.5 VOLUME–NOISE–FUEL SCALING

Equation 8 implies that, at fixed concentration, the number of molecules representing a signal scales as $O(V)$. In an ideal Poisson regime the relative copy-number noise scales as $O(V^{-1/2})$ via Equation 9. If a fixed fraction of a fuel population is consumed during execution, chemical material use and the free-energy bookkeeping in Equation 7 also scale approximately as $O(V)$. These relations expose a basic physical tradeoff:

$$\text{smaller volume} \quad \Rightarrow \quad \text{less material/fuel but larger relative fluctuations.} \tag{57}$$

This does not determine an optimal volume—transport, surface effects, reaction times, and readout matter as well—but it illustrates why a molecular-resource objective eventually needs stochastic and physical terms in addition to species/reaction counts.

G.6 SCOPE OF THE THEORY

The bounds above are deliberately conservative and substrate-agnostic. They do not model sequence-specific hybridization, diffusion-limited transport, spatial compartments, stochastic reaction networks at low copy number, or non-equilibrium fuel regeneration. Their purpose is narrower: to make explicit the assumptions under which (i) a small held-out selector is statistically sensible, (ii) local molecular fidelity can influence whole-model task loss, (iii) retained width maps to molecular cost, and (iv) autonomous phase control can approximate externally staged execution. The empirical sections test these ideas in the concrete compiler/model setting used in this paper.

H COMPLETE EXPERIMENTAL RESULTS

This appendix collects the quantitative results that support the main-paper claims. Resource quantities have different semantics across model families and are therefore reported in separate tables rather than merged onto a single axis. Qwen3 budgets are retained hidden-channel fractions of eight selected MLPs; the SmolLM2 study reports compiler-estimated species/reaction counts for the selected Transformer MLP family; and the Mamba study reports its architecture-specific molecular-resource, latency, and fuel fractions.

H.1 MECHANISM STUDY: MOLECULARIZATION, ADAPTATION, STRUCTURAL COMMITMENT, AND RECOVERY

The five-stage block-scale mechanism study is promoted to Table 1 in the main paper because it gives the most direct view of the complete molecularize–adapt–search–commit–recover sequence. The corresponding mean-squared errors are 2.505×10^{-6} after direct molecularization, 8.132×10^{-7} after molecular retraining, 1.584×10^{-6} after soft MAS search, 5.377×10^{-6} immediately after hard 50% commitment, and 1.033×10^{-6} after post-commit recovery. Thus the same qualitative conclusion holds in both relative-ℓ_2 and MSE: substrate adaptation repairs molecularization error, commitment buys resources at a fidelity cost, and recovery repairs much of the commitment shift without restoring resources.

H.2 PRETRAINED SMOLLM2: MOLECULARIZATION AND MATCHED-TREATMENT ATTRIBUTION

The pretrained SmolLM2 experiment supplies the main causal ablation. The digital teacher has held-out NLL 3.24923. Direct molecularization raises NLL to 4.60435, a penalty of 1.35512. Molecular retraining alone reaches 3.63542, recovering 0.96893 NLL, or 71.5% of the molecularization penalty, without changing the full-budget compiler resource count.

Table 15: Full-budget SmolLM2 molecularization and adaptation. Species/reaction counts refer to the molecularized selected MLP family.

Stage	Held-out NLL	ΔNLL vs. digital	Species	Reactions
Digital teacher	3.24923	0	–	–
Molecularized	4.60435	+1.35512	529,920	159,805,440
Molecular retraining	3.63542	+0.38619	529,920	159,805,440

The main paper promotes the full 100/75/50/25% fidelity/resource trajectory to Table 5 and the complete matched-treatment selector comparison to Table 6.

The promoted matched-treatment table compares pruning-only baselines, matched retraining/recovery baselines, and MAS with and without post-commit recovery under the same 75/50/25% budgets.

The gap between the last two rows isolates post-commit recovery: 0.146, 0.301, and 0.986 NLL at 75/50/25% budgets. Thus recovery becomes more valuable as the structural commitment becomes more aggressive. At the 50% reaction budget, the final NLL 4.2927 is also lower than the uncompressed naively molecularized NLL 4.6043, showing that adaptation plus structure optimization can outperform direct molecularization while using half of the modeled reactions.

H.3 QWEN3: ADAPTATION BENEFIT BEFORE STRUCTURAL SEARCH

The budget-independent digital, naively molecularized, and retrained NLL values for Qwen3-0.6B/1.7B/4B are promoted to Table 2 in the main paper because they establish the substrate-adaptation effect before structural search. The dense tables below then report the subsequent candidate-selection and compression behavior across all nine retained-channel budgets.

H.4 SCOPE OF THE QWEN3 INTERVENTION

A Qwen3 budget is local to eight selected decoder MLPs. It should therefore not be read as a whole-model compression ratio. Qwen3-0.6B and 1.7B contain 28 decoder layers, while Qwen3-4B contains 36 (Yang et al., 2025); the intervention touches 28.6%, 28.6%, and 22.2% of decoder MLPs, respectively. For the bias-free SwiGLU hidden path used here, gate/up/down projection weights scale linearly with intermediate width. Thus retaining fraction b in the eight selected MLPs removes a fraction $(1-b)\,8/L$ of all decoder-MLP hidden-path weights, where L is the decoder-layer count. Table 16 quantifies this distinction. The fraction of *whole-model* parameters changed is smaller still because attention, embeddings, normalization, and all unselected MLPs are untouched.

Table 16: Qwen3 intervention scope. "All-MLP removed" is the fraction of decoder-MLP hidden-path weights removed across the full decoder stack, not a whole-model compression percentage.

Model	Selected/total MLPs	75% budget	50% budget	10% budget
Qwen3-0.6B	8/28	7.1%	14.3%	25.7%
Qwen3-1.7B	8/28	7.1%	14.3%	25.7%
Qwen3-4B	8/36	5.6%	11.1%	20.0%

H.5 Dense Qwen3 selected-MLP budget sweep

All Qwen3 points use eight selected MLPs, 16 calibration sequences, 16 selection sequences, and 256 untouched evaluation sequences of length 128 from WikiText-2. Figure 2 in the main paper shows the MAS-versus-saliency crossover; positive advantage means lower evaluation NLL for MAS. The compact per-scale selector summary is promoted to Table 3; the dense tables below retain every operating point. The independent selector agrees with the post-hoc evaluation oracle at all 27 points, and no rollback was triggered across the 40 adaptation records. These 27 cells share the frozen dataset partitions and protocol, so they are operating-point breadth rather than independent repeated trials.

Table 17: Complete dense sweep for Qwen3-0.6B. ΔNLL is relative to the digital model. "Recovered" is the fraction of the naive molecularization penalty removed by the final guarded model; negative values mean the final compressed model is worse than naive molecularization. MAS advantage is saliency NLL minus MAS NLL. Jaccard compares the MAS and saliency selected-channel sets.

Budget	Guarded NLL	ΔNLL	Recovered	MAS adv.	Choice	Jaccard
90%	3.5495	+0.3131	88.9%	+0.1598	MAS	0.806
80%	3.5773	+0.3408	87.9%	+0.3475	MAS	0.625
70%	3.6606	+0.4242	84.9%	+0.5487	MAS	0.482
60%	3.7058	+0.4693	83.3%	+0.8916	MAS	0.365
50%	3.8817	+0.6452	77.1%	+1.2037	MAS	0.274
40%	4.3927	+1.1563	59.0%	+1.1998	MAS	0.203
30%	4.8277	+1.5913	43.5%	+1.4433	MAS	0.148
20%	5.1440	+1.9075	32.3%	+1.8574	MAS	0.106
10%	6.4509	+3.2145	-14.1%	+1.7739	MAS	0.061

Table 18: Complete dense sweep for Qwen3-1.7B. Column definitions match Table 17.

Budget	Guarded NLL	ΔNLL	Recovered	MAS adv.	Choice	Jaccard
90%	3.1982	+0.3191	79.3%	+0.1269	MAS	0.808
80%	3.2114	+0.3323	78.4%	+0.2733	MAS	0.640
70%	3.3260	+0.4469	71.0%	+0.3210	MAS	0.499
60%	3.3777	+0.4986	67.7%	+0.4956	MAS	0.382
50%	3.4618	+0.5827	62.2%	+0.5979	MAS	0.289
40%	3.5694	+0.6903	55.2%	+0.6532	MAS	0.222
30%	4.1821	+1.3030	15.5%	+0.2682	MAS	0.159
20%	4.9399	+2.0608	-33.7%	-0.7694	Saliency	0.107
10%	6.1619	+3.2828	-112.9%	-2.8991	Saliency	0.065

Table 19: Complete dense sweep for Qwen3-4B. Column definitions match Table 17.

Budget	Guarded NLL	ΔNLL	Recovered	MAS adv.	Choice	Jaccard
90%	2.8624	+0.1778	89.8%	+0.0671	MAS	0.811
80%	2.9210	+0.2364	86.5%	+0.0945	MAS	0.641
70%	2.9633	+0.2788	84.1%	+0.1729	MAS	0.511
60%	3.0123	+0.3277	81.3%	+0.3156	MAS	0.404
50%	3.0595	+0.3749	78.6%	+0.5832	MAS	0.318
40%	3.2170	+0.5324	69.6%	+0.8145	MAS	0.258
30%	3.4061	+0.7215	58.8%	+0.9357	MAS	0.210
20%	3.6182	+0.9336	46.7%	+1.1175	MAS	0.169
10%	4.5867	+1.9021	-8.6%	+0.8695	MAS	0.132

Two additional diagnostics are worth noting. First, the MAS/saliency Jaccard overlap decreases markedly as the budget tightens, indicating that the resource-aware search is not merely reproducing the saliency ranking. Across the dense sweep the overlap ranges from 0.061–0.806 (0.6B), 0.065–0.808 (1.7B), and 0.132–0.811 (4B). Second, the selector's two switches to saliency occur exactly where MAS becomes worse on the untouched evaluation set, giving zero post-hoc regret without using the evaluation set for selection.

H.6 Evaluation-only cross-corpus transfer probe

To test whether candidate selection is narrowly tied to the primary WikiText corpus, we freeze the already-trained molecularized candidates and the WikiText validation-guard decision, then evaluate them on English LAMBADA (Paperno et al., 2016). The probe uses 256 non-overlapping 128-token windows (32,768 tokens) formed by deterministic EOS-delimited packing. LAMBADA is never used for adaptation, MAS, saliency construction, or candidate selection, so the post-hoc LAMBADA oracle below is diagnostic only.

Table 20: Evaluation-only LAMBADA transfer probe. "Base" is the unmodified digital-model NLL on the probe. The guarded candidate is fixed by WikiText validation before any LAMBADA evaluation. Regret is guarded NLL minus the better of the MAS and saliency probe NLLs.

Model	Budget	Base	MAS	Saliency	Guarded	Regret
Qwen3-0.6B	75%	3.7321	4.1320	5.1156	4.1320	0.0000
Qwen3-0.6B	50%	3.7321	4.4407	6.5955	4.4407	0.0000
Qwen3-1.7B	75%	3.5487	4.0169	4.3915	4.0169	0.0000
Qwen3-1.7B	50%	3.5487	4.1462	4.8115	4.1462	0.0000
Qwen3-4B	75%	3.4492	3.7171	4.0751	3.7171	0.0000
Qwen3-4B	50%	3.4492	3.8291	4.8625	3.8291	0.0000

The WikiText selector therefore matches the secondary-corpus post-hoc candidate oracle in 6/6 operating points, with zero observed mean and maximum transfer regret. MAS is the LAMBADA oracle in every cell and improves over saliency by 0.358–2.155 NLL. The larger MAS advantage at 50% than 75% for all three scales mirrors the primary held-out trend. These six cells share one frozen protocol and one secondary corpus; they support cross-corpus candidate-choice transfer but are not independent replications or evidence of broad task generalization.

H.7 Mamba architecture transfer and multi-axis resources

The Mamba digital teacher NLL is 3.40430, and a shape-preserving reload matches it within roughly 2.5×10^{-5} NLL. The main paper quotes representative 50% resource fractions, while Table 21 below retains the complete naive and adapted multi-axis comparison. The molecular-resource, latency, and fuel fractions are family-specific compiler/model estimates and should not be interpreted as Qwen3 selected-MLP budgets.

Table 21: Complete Mamba structural-compression study. "Adapted" means pre-commit distillation plus post-commit recovery with the declared structure re-applied. Param., molecular, latency, and fuel columns are fractions of the uncompressed reference.

Structure	Budget	NLL	Param.	Molecular	Latency	Fuel	ΔNLL
Width, naive	75%	10.5896	0.8248	0.7875	0.8875	0.7786	+7.1853
Width, adapted	75%	4.5999	0.8248	0.7875	0.8875	0.7786	+1.1956
Width, naive	50%	13.9189	0.6496	0.5250	0.7750	0.5132	+10.5146
Width, adapted	50%	7.0477	0.6496	0.5250	0.7750	0.5132	+3.6434
Width, naive	25%	40.9040	0.4744	0.2625	0.6625	0.2536	+37.4997
Width, adapted	25%	10.1029	0.4744	0.2625	0.6625	0.2536	+6.6986
Width+rank, naive	75%	11.0459	0.8192	0.5858	0.8098	0.5746	+7.6416
Width+rank, adapted	75%	6.1340	0.8192	0.5858	0.8098	0.5746	+2.7297
Width+rank, naive	50%	24.5455	0.5865	0.3410	0.6394	0.3287	+21.1412
Width+rank, adapted	50%	9.1851	0.5865	0.3410	0.6394	0.3287	+5.7808
Width+rank, naive	25%	78.0698	0.3994	0.1458	0.4886	0.1383	+74.6655
Width+rank, adapted	25%	31.8185	0.3994	0.1458	0.4886	0.1383	+28.4142

The width-only comparison is the cleanest architecture-transfer result: adaptation changes NLL from 10.59 to 4.60 at 75%, 13.92 to 7.05 at 50%, and 40.90 to 10.10 at 25%, without changing the committed structural resource fractions. Width+rank reaches lower modeled resource fractions but incurs a larger task-fidelity cost, exposing the expected Pareto trade-off rather than a universally dominant structural family.

H.8 Statistical interpretation

Random structural baselines are stochastic and are reported with multi-seed statistics in the SmolLM2 study. MAS, saliency, magnitude, and deterministic adaptation repetitions are bitwise identical under the recorded configuration, so they are reported as deterministic operating points rather than as artificial independent-seed distributions. For Qwen3, evidential breadth comes from model scale, budget, structural disagreement between candidates, and the 32,768 untouched held-out tokens per operating point.

I Autonomous Molecular Execution and Physicalization Evidence

This appendix collects the compiler and sequence-resolved evidence behind the autonomous-execution contribution. The resource counts below are compiler/model quantities used to track where resources are removed, shared, or added; Appendix J consolidates the physical-evidence boundary.

I.1 Structural reduction versus one-pot autonomy

The block-scale SwiGLU study separates two effects that are easy to conflate. First, structural commitment reduces the computation itself: eight hidden channels become four, species decrease from 96 to 56 (41.7%), and reactions decrease from 288 to 144 (50%). Second, one-pot lowering internalizes the scheduler using molecular phase/control chemistry. The final autonomous program has 64 species and 488 reactions, uses one shared finite drive reservoir, and removes four externally scheduled boundaries. Thus the final autonomous program still uses fewer species than the original uncompressed molecular structure (64 versus 96), but it has more raw reactions because clocking and phase gating are now explicit molecular chemistry. We therefore treat *resource compression* and *autonomy/boundary removal* as distinct compiler objectives.

The resource trajectory for this SwiGLU mechanism is promoted to Table 8 in the main paper.

The generic graph-fusion conformance test provides a separate compiler check: a three-stage DAG is fused into one autonomous program, two internal carriers are coalesced, and the resulting clock-gated program contains nine species and nine reactions. A deliberately tight eight-species budget rejects

the full three-stage group, demonstrating that fusion decisions are constrained by explicit resource budgets rather than performed unconditionally. The fused program is numerically executable on an independent mass-action ODE backend.

I.2 Mamba value closure and boundary removal

The Mamba reference begins from seven logical composite stages and lowers them into one fully value-closed molecular program, eliminating six externally scheduled stage boundaries. The pre-fusion composite accounting contains 98 species and 101 reactions; the fully inlined autonomous program contains 229 species and 294 reactions across 21 carrier families. These counts are not a resource-reduction claim: the fused representation makes previously separate value operators, causal patch routing, selective-scan arithmetic, and control chemistry explicit in a single executable molecular namespace. Its evidence is therefore complementary to the SwiGLU compression result: Mamba primarily demonstrates value closure and autonomous scheduling, whereas the SwiGLU structural study cleanly isolates payload resource reduction.

The main paper summarizes the cross-system boundary-removal counts; the paragraphs here retain the complete interpretation of why the counts differ across systems.

I.3 Sequence-resolved DSD/DNA realization

The physical-closure study realizes the autonomous control chemistry through explicit DNA domains, strands, complexes, and sequence-aware effective kinetics; the main paper quotes a compact subset of these counts. The canonical realization contains 36 physical domains, 198 strands, 36 realized complexes, and nine DSD motif families. The subsequent exact kernel serialization contains 54 kernel complexes plus 18 free complexes and 279 paired-domain occurrences. The sequence screening reports zero sequence-crosstalk warnings in this realization, while explicit modeled leak pathways remain and are propagated into the dynamical studies.

Table 22: Sequence-resolved DSD/DNA realization statistics and the external-enumerator/kinetic-calibration status recorded for the reported oscillator study.

Quantity	Value
Physical domains	36
DNA strands	198
Realized complexes	36
DSD motif families	9
Explicit modeled leak pathways	29
Kernel complexes	54
Free complexes	18
Paired-domain occurrences	279
Externally calibrated displacement interfaces	0 / 27
External domain-level enumerator executed	No

I.4 Clock physicalization progression

Table 9 in the main paper promotes the complete three-design clock progression together with terminal leak. The key distinction is qualitative: finite DSD control and generic regeneration fail to complete a cycle, whereas the oscillator-specific catalytic topology reaches multiple sustained cycles while retaining substantial phase mass and fuel. The detailed robustness and redesign analyses remain below.

I.5 Bounded robustness and evidence-conditioned redesign

The catalytic topology is evaluated under 24 deterministic bounded perturbations of effective catalytic rates, fuel, and leak; the main paper quotes the pass count while the complete quantiles are retained here. Twenty-three of 24 perturbations pass the joint multicycle/mass/fuel/leak gate. The

5th/50th/95th percentiles of phase-progress cycle count are 4.97/5.91/6.65, and the corresponding modeled-period percentiles are 308/356/426 s. The 5th percentile of final phase mass is 0.933 and that of fuel retention is 0.480; the 95th percentile of the leak sink is 0.090.

Table 23: Robustness summary for the catalytic sequence-resolved model. Quantiles describe the deterministic bounded perturbation set; they are not experimental confidence intervals.

Metric	5th pct.	Median	95th pct.
Phase-progress cycles	4.973	5.909	6.646
Modeled period (s)	307.96	356.01	426.10
Final phase mass	0.933	–	–
Fuel retention	0.480	–	–
Leak sink	–	–	0.0900

A bounded evidence-conditioned redesign then improves the conservative multicycle/leak trade-off. Table 24 shows the before/after comparison. The 5th percentile of cycle count rises from 4.92 to 5.27 and the 95th percentile of the leak sink falls from 0.0783 to 0.0465, while the bounded perturbation success fraction remains 1.0 in the redesign study. The 5th percentile of fuel retention is essentially unchanged (0.535 versus 0.533), indicating that the improvement is primarily a topology/rate trade-off rather than a larger fuel reserve.

Table 24: Evidence-conditioned catalytic redesign. The scalar score is the fixed internal multi-objective design score used only to select among the bounded redesign candidates.

Design	Score	Cycles 5th pct.	Fuel 5th pct.	Leak 95th pct.
Baseline catalytic design	7.2689	4.9215	0.5348	0.07835
Selected redesign	7.5701	5.2717	0.5326	0.04654

I.6 External measured-data interoperability anchor

Separately from the oscillator study, the external-evidence interface ingests four published DNA–DNA relative-rate measurements from Ratajczyk et al. (2025). Three rows are used by the interoperability demonstration and one is held out for a pipeline-integrity check. This experiment verifies that measured literature data can be represented with provenance and passed through the evidence interface; the oscillator study and the external-data interoperability path remain distinct evidence tracks.

I.7 Evidence provenance

The autonomous-control results above use sequence-resolved mass-action models with explicit DNA domains, strands, complexes, finite gate/fuel inventories, waste, and leak. Sequence-aware effective interface kinetics drive the oscillator simulations, while the published PNAS relative-rate measurements exercise the separate external-data interoperability and provenance path. Appendix J gives the consolidated physical-validation boundary.

J Detailed Limitations and Scope

This section consolidates the claim boundaries that are intentionally kept compact in the main Discussion. It does not introduce new experimental evidence.

J.1 Candidate selection and generalization

The 27 Qwen3 selector-oracle matches are model–budget operating points under one frozen calibration/selection/evaluation split, not 27 independent statistical replications. The post-hoc evaluation

oracle is diagnostic only and never changes the selected candidate. The English-LAMBADA probe is stricter in a different direction: all candidates and the WikiText selection decision are frozen before LAMBADA evaluation. Its 6/6 candidate-oracle matches therefore support transfer of the *candidate choice* to one secondary English corpus, but they do not establish broad task, domain, language, or distribution-shift generalization. Appendix H.6 gives the full probe construction and values.

J.2 Local structural budgets are not whole-model compression ratios

Qwen3 molecularization changes eight selected decoder MLPs rather than the entire model. These correspond to 8/28 decoder MLPs for Qwen3-0.6B and Qwen3-1.7B, and 8/36 for Qwen3-4B. Consequently, a 50% retained-channel budget removes 14.3%, 14.3%, and 11.1% of decoder-MLP hidden-path weights, respectively, before accounting for untouched attention, embeddings, normalization, and unselected MLPs. Table 16 reports additional operating points. The dense budget sweep is therefore evidence about aggressive compression of a controlled molecularized substructure, not a claim of the same percentage whole-model compression.

J.3 Resource axes and physical efficiency

Resource quantities are family- and compiler-specific. Qwen3 budgets are retained hidden-channel fractions in selected MLPs; SmolLM2 reports compiler-estimated species and reactions for its selected Transformer MLP family; Mamba reports architecture-specific molecular-resource, latency, and fuel fractions. These axes should not be numerically merged. Likewise, compiler counts are not measurements of physical energy, wall-clock throughput, density, or cost. One-pot fusion targets autonomy and boundary removal, so it can reduce external scheduling while increasing raw reaction count through explicit clock and gating chemistry.

J.4 Physicalization evidence boundary

The reported DSD/DNA studies instantiate concrete domains, strands, complexes, finite inventories, modeled leak, and sequence-aware effective mass-action parameters, but the oscillator interfaces are not externally calibrated from sequence-kinetic measurements and the design has not been validated in a wet-lab experiment. We therefore claim a compiler/model pathway toward physical molecular execution rather than a measured molecular accelerator. Appendix I gives the complete DNA realization, robustness, redesign, and evidence-boundary details, including the distinction between the external measured-rate interoperability fixture and validation of the proposed oscillator.

K Acronyms and Technical Abbreviations

This section expands the technical abbreviations used throughout the paper. Proper software and model names such as PyTorch, JAX, Qwen3, SmolLM2, Mamba, Peppercorn, and Multistrand are names rather than abbreviations and are therefore not expanded here.

AI artificial intelligence; used broadly for learned models and in the AI-use disclosure.

CE cross-entropy; a token-level training loss used alongside distillation terms.

CRN chemical reaction network; the reaction-level intermediate representation used for molecular programs.

DAG directed acyclic graph; used in the compiler graph-fusion conformance example.

DARTS Differentiable Architecture Search; a representative differentiable neural architecture search method discussed in related work.

DNA deoxyribonucleic acid; the physical molecular substrate used for strand-displacement realizations.

DRAM dynamic random-access memory; used only as a digital-memory comparison in the motivation discussion.

DSD DNA strand displacement; the molecular mechanism used to realize CRN-level behavior with DNA complexes and strands.

EOS end-of-sequence; tokenizer separator used when deterministically packing LAMBADA documents into fixed-length evaluation windows.

FLOP floating-point operation; used as a conventional digital-compute cost reference. A FLOP count is distinct from the compiler-derived molecular resource quantities optimized here.

GLU gated linear unit; the feed-forward gating family from which SwiGLU is derived.

ICLR International Conference on Learning Representations; the conference venue targeted by this manuscript.

JIT just-in-time; compilation mode supported by optional numerical backends.

KL Kullback–Leibler; the divergence used in the distillation objective.

LAMBADA LAnguage Modeling Broadened to Account for Discourse Aspects; the broad-context word-prediction dataset used here only as a frozen secondary-corpus evaluation probe (Paperno et al., 2016).

KinDA Kinetic DNA strand-displacement analyzer; an optional sequence/pathway-aware DSD kinetic-analysis tool.

LM language model; used when referring to causal language-model evaluation.

LLM large language model; used for modern large-scale pretrained language models and pruning literature.

MAS molecular architecture search; the compiler-aware candidate-generation stage proposed in this work.

ML machine learning; used as a general category when separating learned-model evidence from compiler and physicalization evidence.

MLP multilayer perceptron; the Transformer feed-forward modules selected for the Qwen3 and SmolLM2 studies.

MSE mean-squared error; an auxiliary block-level fidelity metric reported in the mechanism study.

NAS neural architecture search; the broader digital architecture-search literature to which MAS is compared.

NLL negative log-likelihood; the primary whole-model language-model fidelity metric reported in the pretrained-model experiments.

ODE ordinary differential equation; the deterministic mass-action dynamical system induced by a CRN.

PIL Pepper Intermediate Language; a text representation for domain-level strand-displacement systems used by Peppercorn-compatible tooling.

PNAS Proceedings of the National Academy of Sciences; journal abbreviation used when discussing the external measured-rate reference.

SiLU sigmoid linear unit; the smooth activation used in the gated MLPs considered here.

SSM state-space model; the sequence-model family represented by Mamba in the architecture-transfer study.

SwiGLU Swish-gated linear unit; a GLU-family feed-forward block combining a Swish/SiLU gate with a multiplicative branch.

License abbreviations. CC BY means Creative Commons Attribution, CC BY-SA means Creative Commons Attribution–ShareAlike, and GFDL means GNU Free Documentation License. "MIT" refers to the MIT License. These abbreviations appear in Appendix L when recording public-resource license metadata.

L Datasets and Pretrained Models

This section records the public datasets and pretrained checkpoints used by the reported language-model experiments. The exact repository identifiers are included to disambiguate similarly named resources. License labels below reproduce the corresponding public repository/model-card metadata as checked on September 23, 2026; downstream users should consult the original terms for authoritative licensing information.

L.1 DATASETS

Table 25: Public text datasets used in the paper. "Role" describes their use in the reported experiments, not the original dataset's intended use.

Resource	Provenance and role	License metadata
`Salesforce/wikitext wikitext-2-raw-v1`	WikiText-2, introduced by Merity et al. (2017); English text from verified Good and Featured Wikipedia articles. Used for pretrained-model calibration/adaptation and held-out evaluation, with the Qwen3 study assigning train/validation/test to calibration/selection/evaluation respectively.	Hugging Face dataset metadata lists CC BY-SA 3.0 and GFDL.
`EleutherAI/lambada_openai en/test`	OpenAI-preprocessed form of the LAMBADA test set introduced by Paperno et al. (2016). Used only as a frozen secondary-corpus evaluation probe; it is not used for adaptation, MAS, saliency construction, or candidate selection.	Repository metadata lists MIT; the dataset card describes the release as "Modified MIT." The original `cimec/lambada` card lists CC BY 4.0.

L.2 PRETRAINED MODELS

Table 26: Pretrained checkpoints used in the paper. Parameter counts in model names are nominal family labels.

Checkpoint(s)	Architecture and role	License metadata
`Qwen3-0.6B-Base Qwen3-1.7B-Base Qwen3-4B-Base`	Dense Qwen3 base language models (Yang et al., 2025). Used for the scale-transfer, dense selected-MLP budget, selector-guard, intervention-scope, and frozen LAMBADA transfer experiments.	Apache License 2.0 on each exact model card.
`HuggingFaceTB/SmolLM2-135M`	135M-parameter SmolLM2 Transformer-family causal language model (Ben Allal et al., 2025). Used for matched-treatment attribution of substrate adaptation, structural commitment, candidate choice, and post-commit recovery.	Apache License 2.0.
`state-spaces/mamba-130m-hf`	Transformers-compatible mirror of the Mamba 130M selective state-space checkpoint (Gu & Dao, 2024); its model card states that the checkpoint is unchanged while configuration/tokenizer files are supplied for Transformers compatibility. Used for architecture-transfer and width/width+rank experiments.	The upstream `state-spaces/mamba-130m` checkpoint is tagged Apache License 2.0; the `-hf` mirror does not itself display a license tag, so we report the upstream checkpoint license rather than infer a separate one.

Resource provenance versus scientific citation. The citations above identify the papers introducing the datasets/model families; the repository identifiers identify the concrete artifacts loaded by the experiments. Appendix M specifies the split assignments, sequence counts, and frozen evaluation protocol, while Appendix E records the software used to load and execute these artifacts.

M Experimental Protocol and Evidence Provenance

M.1 Data splits

Qwen3 experiments use `Salesforce/wikitext`, configuration `wikitext-2-raw-v1` (Merity et al., 2017). Training rows provide calibration tokens, validation rows provide candidate-selection tokens, and test rows provide final evaluation tokens. Exact token-row overlap across roles is rejected. The final dense sweep uses 16/16/256 sequences of length 128, corresponding to 32,768 untouched evaluation tokens per operating point. The secondary-corpus transfer probe uses `EleutherAI/lambada_openai`, configuration `en`, split `test` (Paperno et al., 2016). Because most LAMBADA documents are shorter than 128 tokens, probe documents are deterministically concatenated with tokenizer EOS separators and partitioned into 256 non-overlapping 128-token windows. This packing applies only to the evaluation-only probe; it cannot affect WikiText adaptation, candidate construction, or selection.

M.2 Frozen Qwen3 hyperparameters

The scale-transfer and dense-budget studies use the same frozen settings: eight evenly spaced MLP layers; distillation/MAS/recovery steps 6/24/8; distillation and recovery learning rate 10^{-4}; architecture-only MAS learning rate 10^{-2}; gradient clip 1.0; resource-aware neutral MAS initialization; and an independent selector guard. Qwen3 dense budgets are 90/80/. . ./10% retained hidden channels of the selected MLPs. Appendix H separately reports how this local intervention maps to decoder-wide MLP weight removal; it is never interpreted as whole-model compression.

M.3 Result traceability

Appendix L records exact public resource identifiers, scholarly citations, and license metadata; Appendix K expands technical abbreviations. Numerical values in the manuscript were traced to and cross-checked against experiment logs and recorded outputs retained by the authors. Quantitative figures were generated from those recorded outputs using deterministic plotting scripts. This protocol description is intended to make the evaluation design and evidence provenance explicit without depending on implementation-specific file names.

M.4 Evidence boundaries

Qwen3 "budget" denotes retained hidden-channel fraction of the selected MLPs; SmolLM2 and Mamba use different compiler-specific resource quantities and are reported on their native resource axes. Autonomous DSD/clock results are tagged by evidence type, and the published DNA-rate reference is retained as a separate provenance/interoperability artifact. Appendix J states the consolidated physical-validation boundary.