IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

GENESIS V1: A REGENERATIVE FRAMEWORK FOR EVOLVING AI SYSTEMS

Parent–Child Model Evolution, Domain Specialisation, and the Self-Organising AI Ecosystem

Ms. Rakhi S. Meshram
Assistant Professor
Department of Information Technology
AISSMS Institute of Information Technology, Pune, India

Mr. Pavan M. Jadhav
Student
Department of Information Technology
AISSMS Institute of Information Technology, Pune, India

Ms. Anagha Chandan
Student
Department of Information Technology
AISSMS Institute of Information Technology, Pune, India

Ms. Anushree Kale
Student
Department of Information Technology
AISSMS Institute of Information Technology, Pune, India

Abstract: AI systems built on large language models face a persistent architectural challenge: once training ends, the model's knowledge is locked in place. Introducing a new application area — whether clinical decision support, legal research, or domain-specific engineering — demands either retraining the whole network or running a separate fine-tuning pipeline, both of which are expensive and offer no pathway for shared learning across domains. This paper presents Genesis V1, a framework that addresses this problem through a lineage-based approach. A general-purpose parent model, once trained, serves as a permanent knowledge source from which smaller, task-focused child models are derived via knowledge distillation on demand. As children accumulate, they collectively form a query-routing ecosystem — each request goes to whichever specialist is best placed to answer it, and the system as a whole operates at a fraction of the computational overhead a single large model would require. The framework presented here is conceptual and architectural; no implementation exists and no empirical benchmarks have been collected. Where performance figures appear, they are analytical estimates derived from published literature and are labelled explicitly. A structured four-stage plan for experimental validation is included.

Index Terms – regenerative AI, knowledge distillation, parent–child model evolution, model ecosystems, continual learning, LoRA adaptation

I. INTRODUCTION

The standard lifecycle of a production AI system is linear: train, evaluate, deploy, and maintain in its deployed state indefinitely. This works reasonably well when the target application is stable, but real-world deployment is rarely that clean. New subject areas become relevant, client requirements shift, and the model — unchanged since training — is increasingly asked to handle tasks its training distribution never adequately covered. The conventional response is to initiate a new training run or commission an independently fine-tuned variant. At small scale this is manageable; at organisational scale, where dozens of distinct domains may need coverage, the costs compound quickly. Each new specialisation is treated as an isolated problem, no knowledge carries forward between them, and the compute budget scales linearly with the number of domains served.

Genesis V1 is built around a different premise: that a trained general model should function as a living knowledge source rather than a static artifact. When usage patterns reveal sufficient demand for a new domain,

the framework derives a smaller, focused model directly from the parent through a structured distillation process. The parent itself is never modified. Over time, the set of derived specialists forms a self-managing ecosystem that routes incoming requests intelligently, retiring underperforming models and growing new ones as conditions warrant.

Contributions

C1. The Regenerative Paradigm: a formalisation of distillation as a continuous, domain-triggered inheritance mechanism rather than a one-time compression step.

C2. Six-Node Architecture: a complete blueprint covering domain monitoring, child generation, query routing, and model lifecycle management.

C3. Child Model Generator (CMG): a four-step procedure for producing domain-specialist children from a frozen parent without modifying its weights.

C4. Application Mapping: deployment scenarios grounding the framework in coding assistance, clinical AI, legal research, and enterprise knowledge management.

C5. Experimental Roadmap: a graded validation plan with falsifiable hypotheses spanning 1M to 7B parameters.

II. RELATED WORK

Genesis V1 synthesises and extends ideas from several established areas of deep learning research.

The foundational technique underlying child generation is knowledge distillation, first articulated by Hinton et al. [6]. In that formulation, a large teacher network guides the training of a smaller student by exposing it to the full probability distribution the teacher assigns to each input, rather than only the single correct label. Because these distributions encode the teacher's relative confidence across all possible outputs, the student receives a richer learning signal and typically generalises better than if it had been trained from labels alone. DistilBERT [14] demonstrated the practical value of this approach: a student model trained this way retained approximately 97% of BERT's performance while using 40% fewer parameters. Genesis extends this principle significantly — rather than applying distillation once to produce a single smaller model, the framework treats it as a repeatable process that fires automatically whenever a new specialisation threshold is crossed.

Low-rank adaptation (LoRA) [8] addresses the cost of fine-tuning large pretrained models by restricting weight updates to a pair of small decomposition matrices inserted at each layer. Fewer than 1% of parameters participate in training, which reduces GPU memory requirements and wall-clock training time substantially. In Genesis, every child model receives a set of LoRA adapters before distillation begins; only these adapters are updated, leaving both the parent weights and the child's base parameters fixed throughout.

Mixture-of-Experts models [4] achieve computational efficiency by routing each input to a sparse subset of internal specialist subnetworks rather than activating the full model. At first glance this resembles the Genesis routing layer, but the architectures differ in a fundamental way: MoE subnetworks are trained jointly within a single deployable unit and cannot be independently versioned or retired. Genesis children, by contrast, are standalone models that can be deployed, updated, and decommissioned on individual schedules. The pool also grows post-deployment as new domains emerge, which MoE architectures cannot do without retraining.

FrugalGPT [1] tackles inference cost from a different angle, selecting dynamically among a fixed set of pre-existing external models to balance accuracy against cost. Genesis is distinct in that it builds its own specialist pool through principled knowledge transfer from the parent; it does not depend on or select from models developed independently.

Four structural properties of current deployment practice create the conditions Genesis V1 is designed to address.

Table 1: Limitations of static LLMs and the Genesis response

Limitation	Current Behaviour	Genesis Target
Specialisation cost	Full retraining per new domain	Child generation at <10% of retraining cost
Static weights	No adaptation after deployment	Continuous ecosystem growth via child generation
Compute efficiency	Large general model handles every query	Right-sized child model per domain
Knowledge isolation	Independent models, no shared learning	Principled parent–child knowledge transfer

A. Cost Analysis

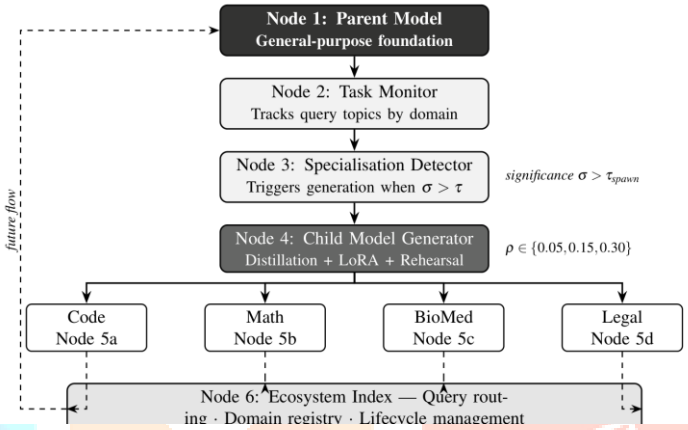
Under the Genesis cost model, the total expenditure across K specialisations is given by Equation 1:

$$C_{Genesis}(K) = C_{train} + K \cdot \rho \cdot C_{finetune} \quad \ldots (1)$$

where $\rho \in \{0.05, 0.15, 0.30\}$ is the child-to-parent size ratio. Genesis becomes cheaper than repeated fine-tuning at approximately $K^* \approx 16$ domains for $\rho = 0.10$.

IV. THE GENESIS FRAMEWORK

The Genesis V1 system is built from six cooperating components arranged in a closed feedback loop. Collectively they monitor live traffic, decide when a new specialist is worth building, construct and deploy that specialist, and manage the growing pool of models over time — all without touching the parent model's weights at any point.



The Genesis V1 six-node regenerative ecosystem. Solid arrows show primary data flow; dashed arrows show feedback. The left dashed arrow represents planned upward knowledge flow (Genesis V2).

A. Node Descriptions

Node 1 — Parent Model: A general-purpose foundation model, fully trained before deployment, that handles every incoming query for which no registered specialist exists. Its parameters are treated as read-only for the lifetime of the ecosystem; all downstream specialisation occurs through the CMG, not through in-place modification.

Node 2 — Task Monitor: An observational module that continuously logs incoming queries and groups them by subject area. It maintains a running significance measure for each candidate domain, combining query frequency, topic breadth, and estimated gap between parent performance and what a specialist might achieve.

Node 3 — Specialisation Detector: Evaluates a composite score σ(Di) for each tracked domain by weighting query volume, the parent's measured accuracy on domain-specific test inputs, and the projected compute savings from routing that traffic to a smaller child model. When σ(Di) surpasses a configurable threshold τspawn and no adequate specialist is already registered, the detector initiates the CMG pipeline.

Node 4 — Child Model Generator (CMG): The regenerative core of the system. It receives the frozen parent and a curated domain corpus and applies a four-step distillation-and-adaptation procedure to produce a compact specialist model. The full procedure is detailed in Section VI.

Node 5 — Specialised Children: The standalone models produced by the CMG. Each is sized at ρ fraction of the parent's total parameters and is deployed independently — it can be versioned, shadow-tested, rolled back, or retired without affecting any other model in the ecosystem. On its target domain, each child is designed to outperform the general parent while running at substantially lower inference cost.

Node 6 — Ecosystem Index: A live routing registry that maintains the mapping from query domain classifications to the model endpoints best equipped to serve them. It also manages the full lifecycle of every child model, triggering spawning when the detector fires, flagging degraded models for revalidation, pruning those whose usage falls below a minimum threshold, and merging specialists whose domains have converged sufficiently.

V. SYSTEM WORKFLOW: A STEP-BY-STEP EXAMPLE

To ground the operational dynamics in something concrete, consider how the system evolves following a deployment at a mid-sized technology firm that has installed a Genesis parent model as its central AI assistant.

Day 1 — General use: Queries arrive covering HR procedure lookups, business correspondence drafting, and miscellaneous technical questions. All traffic goes to the parent. The task monitor starts accumulating topic frequency data from this initial stream.

Week 2 — Pattern detected: Software development queries now represent close to 40% of total volume — Python debugging, SQL schema questions, REST API documentation. The domain significance score for software engineering has been climbing steadily and is approaching τspawn.

Week 3 — Child generation triggered: The specialisation detector confirms that no registered specialist covers this domain and that the significance threshold has been crossed. The CMG is invoked: activation patterns on software queries are profiled, a 3-billion-parameter student architecture is initialised from the parent's most responsive layers, LoRA adapters are attached to the relevant projections, and distillation training runs over two days on 16 GPUs with a rehearsal buffer keeping general-purpose knowledge from eroding.

Week 4 — Child deployed: Genesis-Code-3B passes the quality gate and is registered in the ecosystem index. Software engineering traffic is now served entirely by the specialist, freeing the parent's capacity for everything else.

Month 2 — Ecosystem expands: Biomedical and clinical queries have accumulated to the point of crossing the spawn threshold. Genesis-BioMed-1B is generated and deployed. The parent now sees roughly 20% of total traffic directly, and the fleet-wide average inference cost has dropped to approximately 35% of the original baseline.

Month 6 — Ecosystem prunes: A Finance specialist created in the second month has seen steady traffic decline and now sits below the minimum activity threshold. The ecosystem index decommissions it, and those queries revert to the parent. The ecosystem stays lean rather than accumulating dormant capacity.

What this sequence illustrates is that the parent model remains completely unchanged throughout. Each specialist is built from inherited knowledge, not from scratch, and the entire process from trigger to production-ready child runs in days.

VI. CHILD MODEL GENERATION (CMG)

The CMG takes a frozen parent model and a curated domain corpus as its inputs and returns a deployable, domain-focused specialist. It operates in four sequential stages, each building directly on the output of the previous one.

Each child is generated in days, not months.

6. Child Model Generation (CMG)

The CMG takes the parent model and a domain corpus as input and produces a smaller, specialised child model. The process has four steps, shown in Figure 2.

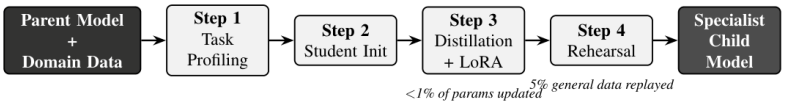


Figure 2. The four-step CMG pipeline. Parent weights are never modified. Only LoRA adapter parameters are updated during training.

The four-step CMG pipeline. Parent weights are never modified. Only LoRA adapter parameters are updated during training.

A. Step 1 — Task Profiling

The first stage characterises which parts of the parent are most sensitive to the target domain. A representative sample of domain queries is forwarded through the parent and activation magnitudes are recorded across all layers and attention heads. The resulting profile identifies parameter groups that carry concentrated domain signal, providing a principled basis for architectural decisions in Stage 2 rather than treating all layers as equally relevant.

B. Step 2 — Student Initialisation

Stage 2 constructs the child architecture at ρ fraction of the parent's total parameter count. Crucially, the child's initial weights are not randomised — they are copied from the parent layers that the profiling stage flagged as highest-activation for this domain. This informed initialisation gives the child a meaningful performance baseline before any training occurs and shortens the subsequent distillation phase substantially.

C. Step 3 — Knowledge Distillation + LoRA

Before training begins, LoRA adapter matrices are inserted into the child's attention projections and feed-forward layers. These adapters — comprising less than 1% of the child's total parameter count — are the only weights that receive gradients during training. The remaining child weights are frozen at their initialised values. The training objective is: L_child = α_ce · L_CE + α_kd · T² · KL(σ(ℓp/T) ‖ σ(ℓc/T)) + λr · L_rehearsal (Equation 2). The cross-entropy term trains against ground-truth labels. The distillation term aligns the child's output distribution with the parent's, evaluated at temperature T = 4; this elevated temperature smooths the parent's predictions so that the child receives meaningful gradient signal from non-dominant output classes, not just the top prediction. The rehearsal term is described in Stage 4, with coefficient λr = 0.10.

$$L_{child} = \alpha_{ce} L_{CE} + \alpha_{kd} T^2 KL(\sigma(\ell_p/T) \| \sigma(\ell_c/T)) + \lambda_r L_{rehearsal} \quad \text{... (2)}$$

where ℓ_p and ℓ_c are parent and child output scores, and T = 4 is a temperature parameter that softens the parent's predictions. The child learns not just the correct answer but how confident the parent is across all possible answers, capturing richer signal than binary labels alone. The rehearsal coefficient $\lambda_r = 0.10$ controls how much general knowledge is preserved.

D. Step 4 — Rehearsal Buffer

Five percent of the parent's original pre-training corpus is randomly sampled and interleaved with domain-specific batches throughout the distillation run. This rehearsal mixture [13] prevents the child from narrowing so aggressively to its target domain that it loses the broad linguistic grounding inherited from the parent. The balance between domain specialisation and general competence is governed by λr.

E. Knowledge Retention Estimate

Note: The following is a theoretical projection, not an experimentally validated result.

Proposition 6.1 (Knowledge Retention): Under the distillation objective with temperature T and child-to-parent ratio ρ, the expected accuracy gap scales as $cT(1-\rho)/\rho$, where $cT > 0$ increases with the entropy of the parent's outputs. At ρ = 0.30 and T = 4, the gap remains small and bounded.

Design implication: Use a larger ρ when the application cannot tolerate accuracy loss; use a smaller ρ when inference hardware is the binding constraint.

VII. PRACTICAL APPLICATIONS

Four deployment scenarios illustrate where the Genesis architecture would have the clearest practical impact. In each case, a single frozen parent model provides the foundational knowledge; the CMG builds purpose-specific children without ever modifying it.

A. AI Coding Assistants

General language models accumulate surface-level familiarity with programming syntax across many languages but rarely achieve the depth that production software development demands. Tracking down a subtle concurrency bug, interpreting an obscure library error, or navigating the conventions of a specific framework requires narrow, concentrated knowledge that a general model dilutes. When Genesis detects a sustained volume of software engineering queries, it distils a Genesis-Code child from repositories of code, documentation, and resolved issue threads. The resulting specialist answers language-specific questions with notably higher precision and operates at roughly a tenth of the parent's per-query inference cost. The architecture does not stop at a single level: sub-children targeting Python type annotation, SQL query optimisation, or Dockerfile authoring can be derived from Genesis-Code through the same pipeline, with the parent remaining untouched throughout.

B. Healthcare AI

Clinical AI operates in a domain where factual precision matters acutely — an incorrect drug interaction, a misidentified contraindication, or a misread diagnostic criterion can have direct patient consequences. General models spread their parameter capacity across all of human knowledge; clinical concepts receive only proportional attention. A Genesis-BioMed child distilled from curated medical literature and de-identified clinical records concentrates that capacity on triage guidance, differential diagnosis support, and patient-facing

information delivery. For settings with institution-specific requirements — a hospital's formulary, local care pathways, departmental protocols — a second-generation Genesis-ClinicalQA grandchild can be derived from Genesis-BioMed, scoped to those specifics. The parent's capabilities are unaffected at every stage.

C. Legal AI

The legal domain is characterised by highly formalised language, jurisdiction-specific procedural rules, and an unusually high cost for imprecision. Contract clauses carry meanings that differ from ordinary usage; a phrase that is standard in one jurisdiction may be unenforceable in another. A Genesis-Legal child trained on contract corpora and case law databases handles document review, clause extraction, and legal summarisation with greater reliability than a general model. The multi-jurisdictional nature of legal practice maps naturally onto the Genesis architecture: separate children targeting US federal law, EU regulations, and Indian statutes can be developed from the same parent independently, each inheriting the parent's general linguistic capabilities while specialising in its own legal tradition.

D. Enterprise AI

Enterprise deployments require models that navigate internal vocabulary, recognise organisation-specific workflows, and operate within the constraints of proprietary document formats — none of which appear in any publicly available training corpus. After monitoring query patterns for a period, a Genesis parent can generate children tailored to the functions that generate the most demand: HR policy interpretation, financial report generation, and technical documentation lookup are typical candidates. Each child runs on less hardware than the parent would require, and the collective footprint of the specialist pool still covers the full range of enterprise use cases while delivering meaningfully lower infrastructure cost.

Enterprise AI

Enterprises need assistants that understand internal workflows and terminology. After a period of deployment, a Genesis parent generates children for HR, documentation, and financial reporting. Each child runs on smaller hardware, reducing infrastructure costs while collectively covering all enterprise needs.

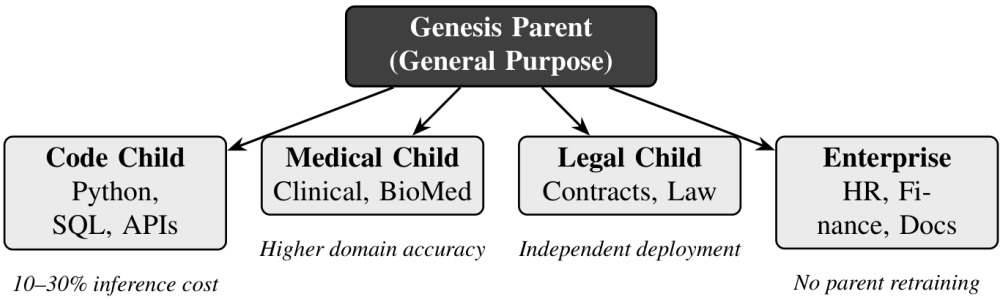


Figure 3. One Genesis parent generating four domain-specific children, each independently deployable at a frac

One Genesis parent generating four domain-specific children, each independently deployable at a fraction of the parent’s cost.

VIII. EXPECTED RESULTS

Important: No Genesis model has been trained or benchmarked. The figures presented in Tables 2 and 3 are analytical estimates constructed by applying established scaling relationships from knowledge distillation research [9, 14] and LoRA fine-tuning studies [8] to the cost model in Equation 1. They represent a plausible upper bound on system performance under the proposed design, not measured outcomes. Empirical work is left to the validation roadmap in Section X.

TABLE 2: EXPECTED COMPARISON OF MODEL TYPES (†) — NOT EXPERIMENTAL RESULTS

Model Type	Domain Accuracy	Training Cost	Inference Cost	Notes
Full retraining	High	Very High (1×)	High	Required per new domain
Full fine-tuning	High	High (0.07×)	High	One model per domain
Genesis Parent	Medium	High (1×, once)	High	General purpose
Genesis Child (ρ=0.30)	High†	Low (0.02×†)	Low (30%)	Specialist; projected
Genesis Child (ρ=0.10)	Med-High†	Very Low (0.007×†)	Very Low (10%)	Edge deployment; projected
MoE (Switch-T)	Medium	High	Medium	Sparse single model
RAG System	Medium	Low	Medium	Retrieval-based

(†) Analytical projections from [8, 14]. Costs normalised to full retraining = 1×.

TABLE 3: PROJECTED COMPUTE EFFICIENCY GAINS (†) — NOT EXPERIMENTAL

Query Coverage via Children	$\bar{\rho} = 0.30$	$\bar{\rho} = 0.15$	$\bar{\rho} = 0.10$	$\bar{\rho} = 0.05$
70%	1.9×†	2.3×†	2.5×†	2.7×†
80%	2.2×†	2.8×†	3.0×†	3.3×†
90%	2.5×†	3.5×†	3.9×†	4.4×†

(†) Derived analytically from Eq. 1.

IX. SUPPORTING ARCHITECTURE

The CMG is the only component strictly required for the regenerative mechanism to function. Three supporting modules are proposed alongside it to improve the parent model's reliability as a long-running knowledge source. These can be adopted selectively, modified, or omitted entirely without altering how child generation works.

TABLE 4: KEY GENESIS V1 SUPPORTING MODULES

ID	Name	Role
CMG	Child Model Generator	Core. Generates specialised children via task profiling, parent-initialised student construction, knowledge distillation, LoRA fine-tuning, and rehearsal buffering.
HSE	Hierarchical Sparse Encoder	The Hierarchical Sparse Encoder addresses the quadratic attention scaling problem that arises when the parent processes long documents during task profiling or distillation. Input text is segmented into four nested structural levels — token, phrase, sentence, and discourse — and attention is applied with progressively coarser granularity at each level. Local dependencies are captured with full resolution at the token level while global structure is maintained at the discourse level without the cost of cross-token attention across thousands of positions.
EMB	Episodic Memory Bank	The Episodic Memory Bank stores session context as a set of key-value pairs that remain accessible across the turns of a multi-turn conversation. Without this module, each turn is processed independently and the parent must rely solely on whatever fits within the active context window. For professional domains where earlier turns establish the terminology, constraints, or patient history relevant to

		subsequent questions, the memory bank prevents the parent from treating each input as contextually isolated.
SBA	Session-Boundary Adapter	The Session-Boundary Adapter applies a small set of regularised parameter updates to a targeted subset of the parent's weights at the close of each session. The update magnitude is constrained by a regularisation penalty that grows with deviation from the pre-update values, preventing any single session from distorting the parent's general behaviour while still allowing it to slowly reflect usage patterns observed in production. The mechanism is analogous to elastic weight consolidation [11] but applied continuously at session boundaries rather than as a one-time post-training step.

X. EXPERIMENTAL ROADMAP

No experiments have been run. The hypotheses listed in Table 5 are grounded in projections from prior distillation and continual learning work [9, 11, 14] and represent informed estimates rather than design constraints. The four stages are sequenced so that each produces independently usable results — later stages do not need to wait on earlier ones to begin, and a negative result at any stage provides actionable information about where the theoretical model needs revision.

TABLE 5: FOUR-STAGE EXPERIMENTAL ROADMAP FOR GENESIS V1

St.	Scale	Goal	Key Hypothesis (†)
1	1M–10M params	Validate CMG in isolation	Child at ρ=0.30 achieves >90% parent accuracy with <2 PPL regression on a general benchmark.
2	100M params	End-to-end regeneration with all nodes	CMG produces a domain child at ≥88% scratch-trained quality at <5% of its training cost.
3	1B params	Multi-generation ecosystem validation	Grandchildren retain ≥80% parent accuracy at ≤10% parent parameters; routing accuracy ≥90%.
4	7B params	Scaling law analysis	Genesis achieves a steeper capability-per-FLOP curve than a comparable dense Transformer baseline.

XI. DISCUSSION

A. Known Limitations

The requirement for a domain corpus is the most concrete practical obstacle in the current design. The CMG pipeline assumes access to a dataset that is large enough to meaningfully guide the activation profiling stage and provide a sufficient distillation signal across the training run. In domains where relevant data is proprietary, regulated, or genuinely sparse, this requirement cannot be trivially satisfied. One natural extension is to generate a synthetic corpus from the parent model itself — prompting it to produce representative domain text — but this introduces potential quality and coverage issues of its own and is left for future work.

As the number of active children grows, so does the complexity of operating the ecosystem. Routing queries to the right specialist when domain boundaries are fuzzy, maintaining version histories as children are periodically regenerated with fresh data, and deciding when two specialists have converged enough to justify merging are all problems that require explicit engineering attention. The current framework names these lifecycle events — spawn, prune, merge — but does not specify the decision criteria or data structures needed to implement them reliably.

The absence of empirical results is the paper's most consequential limitation and the one from which all others derive their importance. Every table entry marked with a dagger, every projected crossover point, and every proposed design rule depends ultimately on the CMG pipeline performing as the cost model predicts. Whether the distillation objective in Equation 2 reliably preserves parent knowledge at the stated parameter ratios, whether warm initialisation from profiled layers delivers the posited training advantage,

and whether the rehearsal buffer correctly balances general competence against domain specialisation — these are questions that only a working implementation can resolve.

TABLE 6: QUALITATIVE COMPARISON. ALL GENESIS ENTRIES REFLECT PROPOSED DESIGN, NOT VALIDATED RESULTS. (+) ADVANTAGE; (−) LIMITATION; (∼) PARTIAL.

Criterion	Full FT	Scratch Train	MoE Router	Model Cascade	Genesis V1
Low specialisation cost	∼	−	+	+	+
Principled inheritance	∼	−	−	−	+
Dynamic ecosystem growth	−	−	−	−	+
Separate deployable units	+	+	−	+	+
Multi-generation depth	−	−	−	−	+
No forgetting guarantee	−	+	+	+	∼

XII. FUTURE WORK

The most important next step is prototype construction and empirical validation of the CMG pipeline.

F1. Prototype construction: The immediate goal is to confirm that the pipeline produces viable child models at all. A 10-million-parameter parent is tractable on a single research cluster within a few days of compute. The four CMG components — activation profiler, student initialiser, distillation trainer, and rehearsal buffer — should each be implemented and validated in isolation before integration. Success at this stage means the pipeline consistently produces children that perform sensibly on held-out domain queries, not that they outperform baselines.

F2. Retention study: A parent trained on WikiText-103 provides a well-studied baseline with known perplexity characteristics. Generating children at $\rho = 0.10$ and $\rho = 0.30$ and measuring their general-domain perplexity regression relative to the Proposition 6.1 bound will reveal whether the temperature and rehearsal coefficient assumptions in the distillation objective are calibrated correctly. Any deviation from the theoretical curve is informative — it points to which assumption needs revision.

F3. End-to-end evaluation: Scaling to 100 million parameters allows the full system — task monitor, specialisation detector, CMG, and ecosystem index — to be evaluated as an integrated unit on established benchmarks. GSM8K, HumanEval, MedQA, and LegalBench cover the four application domains described in Section VII and offer well-understood baselines against which child quality can be compared. Routing precision should be measured as a separate metric; a system that generates high-quality children but routes queries incorrectly still fails in practice.

F4. Multi-generation and scaling: A 1-billion-parameter deployment with at least one grandchild lineage is required to test whether knowledge fidelity holds across multiple inheritance steps — a question the single-generation experiments cannot answer. Extending the study to 7 billion parameters and plotting capability per FLOP against a matched dense Transformer baseline will determine whether the Genesis architecture offers a structural efficiency advantage that compounds with scale, or whether its gains are primarily visible at the deployment and routing layers.

CONCLUSION

This paper has described Genesis V1, a framework that reconceives the relationship between a trained language model and the specialised tasks it is asked to perform. The standard arrangement — one large model deployed as a fixed artifact — is replaced by a parent-plus-children structure in which the parent generates compact, domain-specific specialists on demand and a routing layer directs traffic to whichever model is best positioned to handle each request.

The analytical basis for the framework is solid in the sense that it rests on components with well-established empirical track records: knowledge distillation, LoRA-based adaptation, and rehearsal-based forgetting prevention have each been validated in published work. What Genesis contributes is a structural argument for combining them into a lifecycle architecture rather than applying them independently as one-off compression or adaptation steps. The projected efficiency figures — a 2–3× reduction in average inference compute when 80% of queries route through children, and a per-domain generation cost of roughly $\rho \times 7\%$ of full retraining — follow from the cost model in Equation 1 and the published results in [8, 14]. They are plausible, not confirmed.

Building the system and measuring what it actually does is the next necessary step. The four-stage roadmap in Section X structures that work into manageable experiments, each with a clear falsifiable hypothesis, so that progress is possible even without committing to the full 7-billion-parameter validation effort from the outset.

The broader point is architectural rather than numerical. Language models are typically treated as products of a training run — objects that are produced and then used in a fixed form. Genesis argues that a trained model is better understood as a seed: a starting point from which a responsive, domain-aware system can grow. Getting there requires engineering that does not yet exist at production scale, but the conceptual case for pursuing it seems clear.

ACKNOWLEDGEMENTS

The authors thank the faculty of the Department of Information Technology at AISSMS Institute of Information Technology, Pune, for their guidance on research scope and manuscript development. The framework described in this paper builds on a substantial body of open-source research in natural language processing; the authors are particularly indebted to the communities that produced the knowledge distillation, continual learning, and parameter-efficient adaptation literature that forms its technical foundation.

REFERENCES

[1] Chen, L., et al. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv:2305.05176.

[2] Chen, M., et al. (2021). Evaluating large language models trained on code. arXiv:2107.03374.

[3] Cobbe, K., et al. (2021). Training verifiers to solve math word problems. arXiv:2110.14168.

[4] Fedus, W., Zoph, B., Shazeer, N. (2022). Switch Transformers: Scaling to trillion parameter models. JMLR, 23(120):1–39.

[5] Guha, N., et al. (2023). LegalBench: A benchmark for measuring legal reasoning in LLMs. arXiv:2308.11462.

[6] Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531.

[7] Hoffmann, J., et al. (2022). Training compute-optimal large language models. NeurIPS 2022.

[8] Hu, E.J., et al. (2022). LoRA: Low-rank adaptation of large language models. ICLR 2022.

[9] Jiao, X., et al. (2020). TinyBERT: Distilling BERT for natural language understanding. Findings of EMNLP 2020.

[10] Jin, D., et al. (2021). What disease does this patient have? Applied Sciences, 11(14):6421.

[11] Kirkpatrick, J., et al. (2017). Overcoming catastrophic forgetting in neural networks. PNAS, 114(13):3521–3526.

[12] McMahan, H.B., et al. (2017). Communication-efficient learning from decentralized data. AISTATS 2017.

[13] Rebuffi, S.-A., et al. (2017). iCaRL: Incremental classifier and representation learning. CVPR 2017.

[14] Sanh, V., et al. (2019). DistilBERT, a distilled version of BERT. arXiv:1910.01108.

[15] Shaham, U., et al. (2022). SCROLLS: Standardized comparison over long language sequences. EMNLP 2022.

[16] Touvron, H., et al. (2023). LLaMA: Open and efficient foundation language models. arXiv:2302.13971.

[17] Vaswani, A., et al. (2017). Attention is all you need. NeurIPS 2017.