YvyrAI

A Spanish-First Recurrent-Deliberation Language Model Architecture with Internal Verification, Conditional Repair, and Adaptive Compute

Juan Manuel Acosta Ramirez
Mauricio Jose Irrazabal Ruiz Diaz

YvyrAI Research Project, Paraguay
Corresponding author: admin@yvyrai.lat
Project website: https://yvyrai.lat/

Evidence boundary. YvyrAI is presented as a complete architecture and implementation plan. Full benchmarking is deferred until a sufficiently larger training run is available.

Abstract

We present YvyrAI, a Spanish-first decoder-only language model architecture centered on a recurrent module called the DeliberationBlock. Rather than treating scale only as parameter count, YvyrAI is designed to expose a second axis of scaling: internal test-time computation. A shared Transformer block is iterated within a single forward pass while a learned controller maintains a gated planning state, reads and writes a compressed latent scratchpad, estimates an internal verification signal, applies a conditional repair update proportional to estimated unreliability, and decides how many iterations should be spent before emitting the final hidden representation. Around this core, the system combines a novel recurrent-deliberation framework with several state-of-the-art language-model technologies, including Grouped-Query Attention (GQA), RoPE positional embeddings, SwiGLU activations, RMSNorm, optional Mixture-of-Depths routing, sliding-window attention, low-precision training support, and inference-time KV-cache compression, a Spanish-first data pipeline, a multi-term training objective, and an implemented Spanish evaluation harness.

The contribution is architectural and methodological. YvyrAI is not reported here as a trained high-capability model, and this paper does not claim superiority over existing systems. Large-scale training and benchmarking remain future work, while the present study focuses on validating the architecture, training pipeline, and evaluation framework. Preliminary experiments on a 15M-parameter configuration were used as a temporary engineering validation to verify that the implementation, tokenizer path, recurrent loop, auxiliary losses, and evaluation harness execute end-to-end. Those experiments are deliberately not presented as downstream benchmarks because the model size is too small to support meaningful capability claims. The paper therefore frames YvyrAI as a serious and falsifiable blueprint: ambitious enough to merit compute access, but honest about what has and has not been empirically demonstrated.

Beyond the technical contribution, YvyrAI constitutes the first language model architecture developed from first principles by a Latin American research team in the Spanish-speaking world, positioned as the research and intelligence complement to Paraguay's emerging AI infrastructure strategy.

Keywords: Spanish language models; recurrent depth; latent reasoning; adaptive computation; internal verification; conditional repair; Paraguay; low-resource AI research; language-model architecture.

Contents

1. Scope, Evidence Boundary, and Contributions

This paper documents the design and reference implementation of YvyrAI, a Spanish-first recurrent-deliberation language model architecture developed in Paraguay. The central goal is not to claim that a trained YvyrAI model already outperforms existing language models. The goal is to define a complete, internally consistent architecture that can be audited, implemented, trained, ablated, and eventually compared against strong baselines. The paper therefore separates four categories of statement: implemented mechanisms, designed configurations, preliminary engineering validation, and future empirical claims.

Table 1. Evidence categories used throughout the paper.

Category	Meaning	Allowed claim	Not allowed claim
Implemented mechanism	A component exists in the reference code path or design specification.	The component can be described technically and prepared for training.	The component improves model quality.
Designed configuration	A target parameter count, training mix, schedule, or benchmark protocol.	The configuration is a planned experimental setup.	The configuration has already been trained.
15M engineering validation	A small temporary run used to check end-to-end execution and stability.	The system can instantiate and exercise the intended loop at small scale.	The system has meaningful downstream benchmark capability.
Future empirical result	A claim that requires full training and evaluation.	It is a hypothesis to be tested.	It is a result today.

The most important non-claim is simple: YvyrAI is not presented as a trained production model. No flagship 1B-parameter run is reported here. No benchmark table is used to imply competitive ability. The preliminary 15M-parameter configuration is mentioned because it is an honest and useful temporary evidence source: it validates the execution path, not language-model capability. In other words, the 15M model is a proof of mechanism, not a proof of intelligence.

This distinction is central to the credibility of the work. A small model can reveal implementation errors, shape mismatches, numerical instability, broken tokenizer assumptions, non-decreasing loss, collapsed halting behavior, or auxiliary-loss failures. It cannot fairly establish that a recurrent-deliberation architecture is stronger than a conventional Transformer. YvyrAI therefore chooses the more conservative route: report the architecture and the validation status, then defer public benchmark claims until a sufficiently trained model and a controlled baseline exist.

The claimed contribution is the design itself: a Spanish-first, Paraguay-aware decoder-only language model architecture that combines recurrent latent computation, internal verification, conditional repair, adaptive halting, a compact latent scratchpad, a modern efficiency stack, and a falsifiable training and evaluation protocol. The project is ambitious, but the ambition is expressed as a concrete experimental program rather than as unsupported results.

2. Introduction

Large language models have historically improved by scaling parameters, data, and training compute. This recipe remains powerful, but it is not the only axis of progress. A complementary path is to spend more computation at inference time, allowing the model to refine internal states before producing an answer. Mainstream reasoning systems often express that extra computation as more generated tokens. YvyrAI instead explores a latent route: iterative refinement inside the network, before the final representation is decoded.

The motivation is practical as well as scientific. For Spanish and especially for Paraguay-aware AI systems, the frontier is not only raw model size. The region needs architectures that can reason carefully in Spanish, preserve local factual context, and remain deployable under constrained compute. A recurrent architecture does not eliminate the need for training compute, but it offers a research direction where a model can learn to allocate internal effort dynamically rather than spending the same computation on every token.

YvyrAI is built around the hypothesis that recurrent latent computation becomes more useful when the recurrent loop is self-evaluative. A loop that merely transforms hidden states may learn refinement, but it has no explicit pressure to estimate whether the intermediate state is reliable. YvyrAI therefore introduces an internal verifier and a conditional repair pathway. The verifier emits a bounded reliability-like scalar over the hidden state. The repair pathway is activated more strongly when the verifier estimates unreliability. The halting head then learns to decide how much recurrent effort should be spent.

This is not presented as a guarantee. The architecture may fail to outperform a plain Transformer baseline. Auxiliary objectives may conflict. Halting may collapse to trivial behavior. The repair signal may become unused. A serious design paper must make those failure modes visible. For that reason, YvyrAI includes a VanillaModel baseline, a block-level diagnostic suite, and an ablation plan that can falsify the core premise once compute is available.

The regional context makes this work timely beyond its technical merits. Paraguay generates 100% of its electricity from renewable hydroelectric sources at the lowest industrial cost in the hemisphere, and is actively building AI infrastructure in partnership with international technology leaders. The critical gap in this ecosystem is not energy, not hardware, and not regulation all of which are being addressed at the national level. The gap is domestically developed AI research. A country that provides computational resources to foreign AI systems without developing indigenous language models risks becoming infrastructure for the intelligence of others. YvyrAI is designed, in part, to address that gap.

3. Research Context and Attribution

YvyrAI is a synthesis project. It does not claim to invent every underlying mechanism. Its originality lies in combining these mechanisms into a Spanish-first recurrent-deliberation architecture with explicit engineering boundaries and a Paraguay-aware data strategy.

Recurrent depth and adaptive computation. Adaptive Computation Time and the Universal Transformer established the idea of allowing neural networks to allocate variable computational depth. Recent latent-reasoning work has renewed interest in recurrent blocks that scale test-time computation by iterating in hidden space rather than only producing longer chains of text. YvyrAI follows this direction but adds two specific design choices: a bounded internal verifier inside the recurrent loop and a repair update whose strength is conditioned on the verifier.

Process supervision and verification. Process reward models and step-level verification demonstrate that intermediate reasoning can be supervised or evaluated. YvyrAI internalizes a lightweight version of this idea. The verifier is not an oracle and is not a factuality guarantee. It is an auxiliary signal intended to make the recurrent state more inspectable and trainable.

Backbone and efficiency. The base model adopts a contemporary Transformer recipe: Grouped-Query Attention, rotary position embeddings, RMSNorm, SwiGLU, FlashAttention-compatible attention dispatch, optional sliding-window attention, and optional Mixture-of-Depths routing. KV-cache compression and low-precision training support are treated as integrations, not original inventions.

Optimization and alignment. The training stack includes a composite objective with answer, guidance, verifier, repair, halting, and latent-stability terms. PCGrad-style gradient surgery is included because the auxiliary terms may conflict. Preference optimization modules such as DPO, GRPO, and process-reward components are positioned as later-stage alignment tools rather than as evidence that the base model already has aligned behavior.

The Spanish-language model gap. The Spanish-speaking world comprises over 500 million speakers across more than 20 countries, yet no language model architecture has been developed from first principles by a Latin American research team. Existing Spanish-capable systems are either products of institutions in the United States, Europe, or China, or represent fine-tuned derivatives of foreign base models. The closest regional analog is Maritaca AI's Sabiá models, developed in Brazil for Portuguese. No equivalent exists for Spanish. YvyrAI is designed to be that equivalent: not a fine-tune, not a wrapper, but an original architecture trained Spanish-first with explicit Paraguay-aware data strategy.

4. Model Overview

At a high level, YvyrAI is a decoder-only model with a conventional causal backbone and a recurrent deliberation module placed after the backbone stack. The backbone produces sequence-level hidden states. The DeliberationBlock then iteratively refines those hidden states using one shared Transformer block, a planning state, a latent scratchpad, a verifier head, a repair transform, and a halting head. Finally, a normalization layer and language-model head produce token logits.

Table 2. High-level system components.

Component	Purpose	Design intent
Tokenizer and prompt schema	Spanish-first BPE and explicit role tokens.	Standardize Spanish instruction data and separate system, user, thinking, and answer spans.
Backbone decoder	Causal Transformer layers with modern attention and FFN components.	Provide a strong conventional baseline before recurrent refinement.
DeliberationBlock	Shared recurrent block plus control heads.	Scale internal computation and permit adaptive refinement.
Verifier and repair path	Estimate internal reliability and apply conditional correction.	Make recurrence self-critical rather than blindly iterative.
Adaptive halting	Predict how many refinement steps are needed.	Spend more compute on difficult states and less on easy states.
Training objective	Six coordinated losses with ramps and balancing.	Train language modeling and deliberation mechanisms without immediate collapse.
Evaluation harness	Spanish benchmarks, BPC/perplexity, and block diagnostics.	Separate capability evaluation from mechanism diagnostics.

The recurrent module is deliberately positioned so that the plain Transformer backbone remains meaningful. This allows a controlled VanillaModel to be trained with nearly the same tokenizer, data, optimizer, and parameter budget. Without this baseline, any future performance difference could be attributed to many uncontrolled variables. With the baseline, the DeliberationBlock can be tested directly.

The architecture is also designed for observability. During evaluation, the model can expose realized iteration counts, mean halting mass, verifier trajectories, repair magnitudes, and latent-decay statistics. These diagnostics are not substitutes for task benchmarks, but they answer a different question: whether the deliberation machinery is actually being used.

5. Backbone Architecture

The backbone follows a conservative decoder-only Transformer design because the paper's novelty is not in replacing proven Transformer components. Each layer uses pre-normalization with RMSNorm, Grouped-Query Attention, rotary positional embedding, and a SwiGLU feed-forward network. Attention dispatch is compatible with PyTorch scaled dot-product attention and FlashAttention-style kernels when available. Sliding-window attention and Mixture-of-Depths routing are optional mechanisms for larger or more efficiency-sensitive configurations.

Grouped-Query Attention reduces KV-cache pressure by sharing key and value heads across multiple query heads. This matters for YvyrAI because recurrent computation increases internal work; inference should therefore avoid unnecessary memory overhead. Rotary position embeddings are used because they are efficient, widely adopted, and suitable for autoregressive generation. SwiGLU is selected as the feed-forward activation because it is a strong default for modern LLM-style architectures.

Training uses uncompressed bf16 cache behavior by default. KV-cache compression is treated as an inference optimization rather than a training dependency. This separation is important: it reduces the chance that early training instability is caused by quantized cache artifacts, and it allows the architecture to be tested first in a numerically cleaner regime.

6. The DeliberationBlock

The DeliberationBlock is the architectural center of YvyrAI. It receives the hidden sequence representation from the backbone and applies recurrent refinement using a single shared Transformer block. The same parameters are reused across iterations, so additional recurrent depth increases computation without linearly increasing parameter count. The block is designed to begin close to an identity-like refinement path and gradually learn specialized recurrent behavior.

Table 3. DeliberationBlock submodules.

Submodule	Role	Why it exists
shared_block	Transformer block shared across recurrent iterations.	Allows additional test-time compute without adding a full new stack per iteration.
iter_embed	Learned iteration-index embedding initialized near zero.	Lets the model distinguish refinement stages after training begins.
planning state	Gated state updated from recurrent hidden activations.	Provides a persistent control signal across iterations.
latent scratchpad	Compressed memory at reduced dimensionality with dynamic decay.	Carries compact information without exposing reasoning tokens.
verify head	Bounded scalar estimate over hidden states.	Provides a trainable self-evaluation signal inside the loop.
repair transform	Conditional correction path scaled by estimated unreliability.	Turns low verifier confidence into an explicit update opportunity.
halt head	Adaptive computation controller.	Predicts when enough recurrent refinement has been spent.

The initialization strategy is intentionally conservative. The iteration embedding is initialized to zero so the early loop does not immediately depend on arbitrary step identity. The planning gate is biased toward limited initial influence. The latent decay bias favors slow memory updates. The halting head is warmed up so the model does not immediately collapse into always halting or always using maximum iterations. These details matter because recurrent auxiliary systems can appear impressive on paper while becoming unstable during training.

The verifier is bounded rather than unbounded. This prevents a pathological regime in which the reliability signal grows without numerical meaning. The repair magnitude is tied to one minus the verification estimate, making the correction path strongest when the model believes the current hidden state is least reliable. This is an engineering hypothesis: it gives the network an explicit affordance for correction, but the usefulness of that affordance must be measured.

7. Mathematical Specification of the Deliberation Loop

Let h be the sequence hidden state after the causal backbone. Let p_t be the planning state, l_t the compressed latent scratchpad, e_t the learned iteration embedding, and rho_t the halting probability at recurrent step t. The loop applies the following operations conceptually:

- `c_t = expand(RMSNorm(l_{t-1})) * s_min`
- `h_tilde = h + plan_norm(p_{t-1}) + c_t + e_t`
- `h_prime = shared_block(h_tilde)`
- `g_t = sigmoid(W_g [p_{t-1}, plan_proj(h_prime)])`
- `p_t = p_{t-1} * (1 - g_t) + plan_proj(h_prime) * g_t`
- `alpha_t = bounded_sigmoid(latent_decay(h_prime), floor, ceiling)`
- `l_t = l_{t-1} * alpha_t + compress(h_prime) * (1 - alpha_t)`
- `v_t = bounded_sigmoid(verify_out(SiLU(verify_proj(h_prime))))`
- `r_t = clamp(1 - v_t, 0, 1)`
- `delta_t = repair_transform(repair_norm(h_prime))`
- `h = h_prime + sigmoid(W_r [h_prime, delta_t]) * delta_t * r_t`
- `rho_t = sigmoid(halt_head(h_prime))`

The bounded sigmoid is defined as bounded_sigmoid(x, f, c) = f + (c - f) sigmoid(x), where f and c are lower and upper bounds. In practice, this is used for both latent decay and internal verification so that these control variables remain within interpretable ranges. The halting mechanism accumulates mass across iterations, enforces a minimum number of recurrent steps, and uses a final remainder to close the computation at the maximum allowed iteration. During warmup, predicted halting mass is mixed with a fixed prior to reduce early collapse.

The loop has two kinds of adaptivity. First, it can adapt the content of the hidden representation through planning, scratchpad updates, verification, and repair. Second, it can adapt the amount of computation through halting. The architecture therefore treats reasoning effort as a learned variable rather than a fixed constan

8. Designed Model Family and Scaling Plan

YvyrAI is specified as a family of model sizes. The smallest configurations are designed for debugging and mechanism validation; the largest configuration is the first intended capability-scale experiment. Parameter counts in this table are design targets, not completed training results.

Table 4. Designed model family.

Size	Approx. params	Vocab	d_model	Q/KV heads	Layers	Max recurrent iters	Context
1M	~1M	4,096	128	4 / 2	4	4	256
3M	~3M	4,096	192	6 / 2	5	4	384
15M	~15M	24,576	256	4 / 1	10	10	2,048
100M	~100M	16,384	768	12 / 4	12	8	1,024
200M	~200M	16,384	1,024	16 / 4	16	10	2,048
1B	~1.01B	24,576	1,792	28 / 7	22	10	2,048-4,096

The 15M configuration has an important but limited role. It is large enough to exercise the intended code path, tokenizer, recurrent loop, auxiliary heads, and evaluation harness. It is not large enough to produce benchmark numbers that should be compared to modern language models. Reporting such numbers as evidence of general intelligence would be misleading. The 1B configuration is the first target where downstream benchmark tables could become meaningful, provided it is trained on an adequate token budget and compared against a strong non-recurrent baseline.

The flagship 1B design uses a minimum recurrent iteration count of three, a maximum recurrent count of ten, bounded verification and latent decay, dropout set to zero by default, recurrent activation checkpointing, and a 2K to 4K context target. These values are not magical; they are a coherent first experiment intended to make the deliberation mechanism testable under realistic constraints.

9. Prompt Format and Tokenizer

The project uses a Spanish-first prompt schema with explicit role delimiters:

```
<|bos|><|sistema|> ... <|usuario|> ... <|pensar|> ... <|respuesta|> ... <|eos|>
```

The <|pensar|> delimiter is included as a structural token rather than as a promise that the model will expose all internal reasoning. In the architecture, most deliberation occurs in latent state. The delimiter is useful for data formatting, curriculum control, and potential supervision of guided intermediate regions, but the final deployment policy should decide how much intermediate reasoning is shown to users.

The tokenizer is a from-scratch Spanish BPE design built with a Metaspace-style pre-tokenization strategy and a seeded alphabet that includes mathematical and programming symbols. A Spanish-first tokenizer is important because tokenizer quality affects compression, sequence length, and the model's ability to represent local names, accents, and domain-specific terms. The OOV audit checks unknown-token coverage and provides an early signal of whether the tokenizer is suitable before expensive training begins.

10. Training Objective, Curriculum, and Optimization

The designed training objective combines six losses:

```
L = lambda_ans L_ans + lambda_gui L_gui + lambda_ver L_ver + lambda_rep L_rep + lambda_halt L_halt + lambda_lat L_lat
```

The answer loss is standard next-token cross-entropy over answer tokens. The guidance loss supervises selected intermediate regions when such supervision is available. The verifier loss is intended to calibrate internal reliability estimates and can include ranking-style signals. The repair loss encourages useful activation of the correction path. The halt loss teaches adaptive computation rather than fixed-depth recurrence. The latent loss regularizes scratchpad norm, decay, and stability.

Table 5. Designed loss weights.

Term	Generic base weight	1B target weight	Role
answer	1.00	1.00	Language modeling and response quality.
guidance	0.50	0.40	Supervision over guided intermediate regions.
verify	0.30	0.35	Verifier calibration and relative reliability ordering.
repair	0.30	0.35	Useful activation of conditional correction.
halt	0.01	0.04	Adaptive computation and anti-collapse behavior.
latent	0.10	0.15	Scratchpad norm, decay, and stability regularization.

The model is trained with a phased curriculum. Phase 1 focuses on base language modeling and instruction structure. Phase 2 introduces structured reasoning data. Phase 3 activates verifier, repair, and halting terms with delays and ramps. Phase 4 emphasizes latent internalization and stability. Phase 5 performs final optimization and ablation-oriented evaluation. The phased design is not decorative; it is a risk-control mechanism. Turning on all auxiliary objectives at full strength from the first step would likely destabilize training.

Loss weights are adaptively balanced using EMA-based scaling with clipped ranges. PCGrad-style gradient surgery is used when loss-term gradients conflict. This is included because multi-objective recurrent training is not guaranteed to be cooperative. The optimizer must manage the possibility that answer quality, verifier calibration, repair activation, and halting behavior pull parameters in different directions.

11. Data Pipeline and Paraguayan Knowledge Slice

The designed corpus mixture prioritizes Spanish while preserving enough reasoning, code, and factual material to support general language-model behavior. Public Spanish web text forms the base. Mathematical reasoning corpora are translated offline into Spanish with structure-preserving protections for LaTeX, code, and boxed expressions. A dedicated Paraguayan slice is included to represent local official, historical, civic, and factual content with provenance metadata.

Table 6. Designed 1B training mixture.

Family	Target share	Purpose
CulturaX / Spanish web text	45%	Broad Spanish language modeling and general knowledge exposure.
Reasoning data	18%	Mathematical and structured problem solving after translation and filtering.
Long-context Spanish	10%	Document-level coherence and extended context handling.
Instruction / QA / chat	6%	Instruction following and conversational format alignment.
Wikipedia raw LM	5%	Neutral encyclopedic Spanish language modeling.
Scientific / academic	4%	Technical terminology and formal exposition.
Python / code	3%	Programming syntax and code reasoning.
Wikipedia grounded QA	2%	Factual question answering with source-like structure.
Books	2%	Long-form style and narrative coverage.
News	2%	Contemporary public-language register.
Paraguayan official sources	2%	Local factual and civic grounding.
Safety	1%	Minimum behavioral safeguards and refusal patterns.

The translation pipeline protects mathematical notation before translation and restores it afterward. Examples that cannot be restored cleanly are discarded. GSM8K and MATH-style evaluation sets are intentionally excluded from training to reduce contamination risk. The Paraguayan slice is separated from its hold-out counterpart so that local factual evaluation can be performed without training leakage.

This data strategy is a design proposal, not a measured advantage. A Spanish-first corpus can improve relevance, but it can also introduce quality-control challenges. The paper therefore treats data curation as a first-class experimental variable, not as an automatic guarantee of better Spanish reasoning.

12. Preliminary 15M-Parameter Engineering Validation

The preliminary 15M-parameter configuration is used as temporary engineering support for the project. Its purpose is to validate that the architecture can be instantiated and exercised end-to-end before seeking or spending larger compute. This is a responsible intermediate step: it is better to test the implementation honestly at small scale than to fabricate benchmark tables or present capability claims that the evidence cannot support.

Table 7. What the 15M validation can and cannot support.

Question	Can the 15M run address it?	Reason
Does the tokenizer and model instantiate correctly?	Yes.	Shape, vocabulary, and prompt-format errors can be detected at small scale.
Does the recurrent loop execute through forward and backward passes?	Yes.	The loop, auxiliary heads, and checkpointing path can be exercised.
Do the losses produce finite values and optimization progress?	Yes, as an engineering signal.	Small-scale stability can reveal major implementation defects.
Does YvyrAI outperform strong LLMs?	No.	A 15M model is far below meaningful comparison scale.
Does recurrent deliberation improve benchmark accuracy?	Not conclusively.	This requires an ablated baseline at adequate scale and token budget.
Should 15M downstream scores be used for publicity?	No.	They would misrepresent the strength of the system.

For presentation purposes, the 15M result should be described in one sentence: preliminary small-scale experiments validate end-to-end execution and training stability of the implemented architecture, but they are not used as evidence of downstream capability. This phrasing is strong because it is true, and professional because it does not exaggerate.

The stronger claim is actually the restraint. In a field where benchmark numbers can be overinterpreted, choosing not to report weak or irrelevant numbers increases credibility. The correct next experiment is a mid-scale ablation and then the designed 1B run, not a benchmark table built around a tiny model.

13. Evaluation Protocol and Benchmarking Philosophy

The evaluation harness is designed before the flagship model is trained. This is important because benchmarks should not be selected after seeing results. The harness includes perplexity and bits-per-character computation with careful numerical handling, accent-preserving Spanish normalization, Spanish QA, summarization, paraphrase, mathematical reasoning, and a Paraguayan hold-out set.

Table 8. Evaluation families and intended interpretation.

Evaluation	Purpose	Interpretation
Perplexity / BPC	Measure language-model fit and compression.	Useful for training progress, not sufficient for reasoning quality.
XQuAD-es	Cross-lingual question answering.	Tests reading comprehension under Spanish context.
MLSUM-es	Spanish summarization.	Tests long-form compression and content selection.
PAWS-X-es	Paraphrase discrimination.	Tests sensitivity to word order and semantic equivalence.
MGSM-es	Multilingual math reasoning.	Tests structured problem solving in Spanish.
Paraguayan hold-out	Local factual grounding.	Tests local knowledge without training contamination.
Block diagnostics	Verifier, repair, halting, latent decay.	Tests whether the DeliberationBlock is used as intended.

Benchmarks will be reported only when the model has been trained at a scale where the numbers are interpretable. The evaluation must include a VanillaModel baseline, matched tokenizer, matched corpus, and comparable parameter or compute budget. If YvyrAI outperforms the baseline, the result should be reported. If it does not, that result is still scientifically useful because it identifies the cost of the recurrent machinery.

The benchmark philosophy is therefore adversarial toward the architecture. The goal is not to protect YvyrAI from failure; the goal is to make failure informative. Useful metrics include answer accuracy, perplexity, realized recurrent iterations, latency, memory, verifier calibration, repair activation frequency, and quality changes as maximum recurrent depth increases.

14. Inference System and KV-Cache Compression

The AdaptiveInferenceEngine is designed for autoregressive generation with either native bf16 cache behavior or compressed KV cache at inference time. It supports adaptive recurrent iteration control, optional deliberation traces for debugging, windowed repetition penalty, temperature, top-k, top-p sampling, and numerically robust fallback paths. The inference system is not merely a wrapper; it is where adaptive computation becomes visible as a runtime behavior.

KV-cache compression is integrated as an inference optimization. TurboQuant-style compression is treated as an external method and not as a claimed invention of YvyrAI. The intended value is practical: recurrent computation increases per-token work, so cache efficiency matters. However, compression can introduce numerical error. For that reason, all quality and latency trade-offs must be measured against an uncompressed bf16 inference baseline.

A professional release should report both quality and systems metrics: tokens per second, memory use, average recurrent iterations, p50/p95 latency, cache footprint, benchmark delta under compression, and failure cases. Without those measurements, compression should be described only as implemented or planned, not as proven beneficial for this architecture.

15. Ablation and Falsification Plan

The architecture is designed to be falsifiable. If the DeliberationBlock matters, removing or weakening its parts should change measurable behavior. If it does not matter, ablations will show that the model can achieve the same results with a simpler backbone. Either outcome is useful.

Table 9. Proposed ablation matrix.

Ablation	Question tested	Expected diagnostic signal
VanillaModel baseline	Does recurrence improve quality or efficiency?	Task metrics and latency compared to non-recurrent model.
No verifier	Does self-evaluation help?	Changes in repair usefulness and benchmark accuracy.
No repair path	Does conditional correction matter?	Reduced improvement across recurrent iterations if repair is useful.
Fixed iterations	Does adaptive halting matter?	Compute-quality trade-off compared to learned halting.
No latent scratchpad	Does compressed memory carry useful state?	Changes in long-context and multi-step tasks.
No guidance loss	Does guided supervision stabilize deliberation?	Verifier/repair collapse or weaker diagnostic trajectories.
No PCGrad	Do auxiliary losses conflict?	Higher instability or worse multi-loss balance.

The falsification plan also includes negative criteria. The architecture should be considered unsuccessful in its current form if it produces no quality gain at equal compute, if adaptive halting collapses to a constant, if the verifier becomes uncalibrated and uninformative, if repair activations vanish or explode, or if the latency penalty cannot be justified by measurable gains.

This section matters for presentation. It signals that YvyrAI is not being sold as magic. It is being framed as an experiment with explicit ways to prove it wrong.

16. Roadmap, Compute Bottleneck, and Release Plan

The immediate bottleneck is compute. The architecture, training plan, and evaluation harness can be specified and partially validated at small scale, but the central empirical question requires a much larger run. The project roadmap is staged to avoid wasting compute before the implementation is ready.

1. **Tokenizer and tiny proof-of-mechanism.** Finalize the Spanish BPE tokenizer, run very small models, and check loss, shapes, recurrent execution, and diagnostics.
2. **15M engineering validation.** Use the 15M configuration as a temporary mechanism test for the full loop, auxiliary heads, and evaluation harness.
3. **Mid-scale ablation.** Train a model with and without the DeliberationBlock to obtain the first honest signal about the architecture.
4. **Flagship 1B run.** Execute the designed large training run with a Spanish-first corpus and a clean evaluation protocol.
5. **Public technical release.** Release the paper, corpus manifest, benchmark tables, ablation results, model card, and limitations once empirical evidence exists.

The compute request is therefore concrete: YvyrAI does not need vague hype; it needs the resources to test an implemented, Spanish-first recurrent architecture under a fair benchmark protocol. This is a credible message for professors, sponsors, cloud-credit programs, and local AI initiatives because it identifies the gap between design and evidence precisely.

Recent advances in consumer GPU hardware are relevant to this roadmap. NVIDIA Blackwell-generation GPUs with native NVFP4 precision support achieve up to 4x throughput improvement over BF16 for linear layer computation, with a 3.5x reduction in memory footprint relative to FP16. This means that the 1B flagship configuration is trainable on consumer or institutional hardware rather than requiring exclusive access to large datacenter clusters — a significant shift in the accessibility of this class of experiment for research teams outside traditional AI hubs

17. Limitations, Responsible Use, and Conclusion

The primary limitation is the absence of large-scale empirical validation. Every performance-related statement remains a hypothesis until the model is trained and evaluated. Architecturally ambitious additions often fail to outperform simpler baselines. The DeliberationBlock may increase latency without sufficient quality gain. The verifier may become a consistency signal rather than a truth signal. The repair mechanism may learn shortcuts. Low-precision training and KV-cache compression may introduce numerical error. Translated reasoning data may contain artifacts. The Paraguayan localization strategy is valuable as a goal, but it is not a measured advantage until evaluated.

Responsible deployment would require human oversight, benchmark transparency, safety evaluations, and clear labeling of limitations. A trained YvyrAI system should not be used as the sole decision-maker in high-stakes domains. It may hallucinate, produce plausible but false explanations, or overestimate its own reliability. The internal verifier is not a guarantee of factual truth; it is a model-internal signal that must be calibrated and externally tested.

YvyrAI is best understood as a serious architectural proposal from Paraguay: a Spanish-first language model design that folds recurrent latent computation, internal verification, conditional repair, and adaptive compute into a contemporary decoder-only backbone. The project claims the design, the implementation direction, the localization goal, and the honesty of its evidence boundary. It does not claim benchmark superiority today. The next scientific step is not more rhetorical ambition; it is compute, ablation, and measurement.

From a policy perspective, YvyrAI represents a concrete response to a gap identified in Paraguay's national AI readiness assessment: the need for sovereign AI capabilities that go beyond infrastructure hosting. The project is designed to demonstrate that Paraguay can contribute to the global frontier of language model research, not only as a provider of energy and computation, but as a producer of original AI architecture. The next scientific step is not rhetorical ambition it is compute, ablation, and measurement, conducted by a Paraguayan team on a Paraguayan model.

References

J. Kaplan et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020.

J. Hoffmann et al. Training Compute-Optimal Large Language Models. arXiv:2203.15556, 2022.

OpenAI. Learning to Reason with LLMs. o1 System Card, 2024.

DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948, 2025.

A. Graves. Adaptive Computation Time for Recurrent Neural Networks. arXiv:1603.08983, 2016.

M. Dehghani et al. Universal Transformers. ICLR, 2019.

J. Geiping et al. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. arXiv:2502.05171, 2025.

H. Lightman et al. Let's Verify Step by Step. arXiv:2305.20050, 2023.

D. Raposo et al. Mixture-of-Depths: Dynamically Allocating Compute in Transformer-Based Language Models. arXiv:2404.02258, 2024.

A. Q. Jiang et al. Mistral 7B. arXiv:2310.06825, 2023.

J. Ainslie et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. arXiv:2305.13245, 2023.

J. Su et al. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864, 2021.

N. Shazeer. GLU Variants Improve Transformer. arXiv:2002.05202, 2020.

B. Zhang and R. Sennrich. Root Mean Square Layer Normalization. NeurIPS, 2019.

T. Dao et al. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. NeurIPS, 2022.

A. Zandieh, M. Daliri, M. Hadian, and V. Mirrokni. TurboQuant: Online Vector Quantization with Near-Optimal Distortion Rate. arXiv:2504.19874, 2025; ICLR 2026.

NVIDIA and Open Compute Project. Microscaling and Low-Precision Formats including MXFP8 and NVFP4, 2024-2025.

R. Rafailov et al. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. NeurIPS, 2023.

Z. Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300, 2024.

T. Yu et al. Gradient Surgery for Multi-Task Learning. NeurIPS, 2020.

NLLB Team. No Language Left Behind: Scaling Human-Centered Machine Translation. arXiv:2207.04672, 2022.

T. Nguyen et al. CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages. arXiv:2309.09400, 2023.

L. Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. arXiv:2309.12284, 2023.

A. Mitra et al. Orca-Math: Unlocking the Potential of SLMs in Grade School Math. arXiv:2402.14830, 2024.

Hugging Face. OpenR1-Math-220k Dataset, 2025.

M. Artetxe et al. On the Cross-lingual Transferability of Monolingual Representations. ACL, 2020.

T. Scialom et al. MLSUM: The Multilingual Summarization Corpus. EMNLP, 2020.

Y. Yang et al. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. EMNLP, 2019.

F. Shi et al. Language Models Are Multilingual Chain-of-Thought Reasoners. arXiv:2210.03057, 2022.