Dead Circuits: Activation Profiling and Script Invisibility in Large Language Models

Mohamed Diomande
Independent Researcher
contact@mohameddiomande.com

Abstract

Large language models achieve remarkable performance on languages written in Latin, Cyrillic, CJK, and Arabic scripts. We ask what happens when these models encounter a script that is absent from their pre-training data. We perform activation profiling—a per-layer "brain scan"—of Qwen2-72B-Instruct (4-bit NF4, A100 80GB) processing 100 parallel English/N'Ko sentence pairs. N'Ko is an alphabetic script serving over 40 million Manding-language speakers across West Africa, engineered in 1949 with a strict phoneme-to-grapheme bijection, explicit tonal diacritics, and zero spelling irregularities.

Across all 81 transformer layers, N'Ko induces a **2.90× translation tax** (L2 norm ratio), **30–60% entropy inflation**, **85.8% kurtosis deficit** at the output layer, and **150% higher sparsity** at embedding. Circuit duplication analysis (55 configurations, RYS methodology) shows 0/55 N'Ko-advantageous configurations; the best N'Ko score of 0.067 barely exceeds random chance (0.05). Three-zone failure analysis reveals structurally distinct collapse modes at the embedding layer (comprehension failure), middle layers (reasoning vacuum), and output layers (incoherent prediction).

We then demonstrate that this failure is correctable. A three-stage LoRA pipeline—17,360 continued pre-training, 21,240 supervised fine-tuning, and 25,100 BPE-aware training examples—reduces the translation tax to **0.70×** (a 76% reduction) while degrading English accuracy by only 1.2 percentage points. We provide a detailed analysis of the V1/V2/V3 fine-tuning progression, including mode collapse in V2 and its resolution.

We compare N'Ko's treatment to Arabic, another right-to-left script that LLMs handle competently, and find that the difference reduces entirely to pre-training data volume: Arabic has 4,200+ dedicated vocabulary entries in Qwen2's tokenizer versus N'Ko's 32 single-character fallbacks. We argue that this vocabulary disparity is the mechanistic root of script invisibility, discuss implications for Adlam, Tifinagh, Vai, Osmanya, and Ethiopic, and propose concrete metrics for measuring script equity in multilingual model development.

1 Introduction

In 1949, Solomana Kanté designed N'Ko in response to a claim that African languages were unsuitable for writing. The result was a right-to-left alphabetic script with 27 base characters, Unicode block `U+07C0--U+07FF` (standardized 2006), and engineering properties that no evolved script can match: every phoneme has exactly one grapheme, tone is marked explicitly, and there are no irregular spellings. The name "N'Ko" means "I say" in all Manding languages.

This paper studies a paradox. N'Ko is arguably the best-designed script in our entire phoneme inventory for computational linguistics—its regularity, completeness, and lack of ambiguity are the properties that NLP engineers dream of when designing synthetic alphabets. And it is nearly invisible to modern machine learning. Qwen2-72B-Instruct, a state-of-the-art model with 151,936 vocabulary entries, allocates exactly 32 of those entries to N'Ko: single Unicode codepoints, each representing a single character, with no subword or word-level tokens. For comparison, Arabic receives over 4,200 dedicated vocabulary entries including rich subword representations.

The practical stakes are immediate and human-scale. A child in Kankan, Guinea, who speaks Maninka and reads N'Ko cannot dictate a text message, search the web, or interact with any AI system in their own script. Every voice interface, every language model, every autocomplete engine responds in Latin orthography designed for French colonial linguists—not for the 40+ million people

who actually speak and read these languages. The cognitive cost of this mismatch compounds across education, commerce, healthcare, and creative expression.

We introduce the concept of **script invisibility**: the measurable phenomenon of a language's writing system being absent from a model's learned representations despite the model's architecture having no inherent limitation for processing that script. Script invisibility is not a bug in the model architecture. It is an artifact of training data composition, tokenizer design decisions, and the feedback loop between data availability and research attention.

This paper makes seven contributions:

1. The first per-layer activation profiling study comparing English and N'Ko processing in a large language model, revealing three structurally distinct failure zones across 81 transformer layers (§4).

2. Quantified translation tax metrics—L2 norm, Shannon entropy, kurtosis, and sparsity—for N'Ko across the full depth of Qwen2-72B, establishing numerical baselines for script invisibility (§4.2).

3. Circuit duplication analysis showing that N'Ko activates 0/55 reasoning configurations, confirming that N'Ko's computational representation consists of dead circuits rather than weak ones (§4.3).

4. A systematic comparison of N'Ko and Arabic processing within the same model, demonstrating that RTL script direction is not the source of failure (§5).

5. Tokenizer efficiency analysis quantifying the vocabulary allocation disparity across scripts and its downstream effects on representation quality (§6).

6. A three-stage LoRA pipeline that reduces the translation tax from 2.90× to 0.70×, with detailed V1/V2/V3 progression and mode collapse analysis (§7).

7. A framework for extending these diagnostics to other underrepresented scripts—Adlam, Tifinagh, Vai, Osmanya, and Ethiopic—with specific predictions about the vocabulary-to-representation relationship (§8).

2 N'Ko: Script Design and Computational Properties

Before presenting our empirical results, we provide background on N'Ko's design properties that are relevant to understanding why the script should, in principle, be computationally advantageous—and why its invisibility to LLMs is therefore paradoxical.

2.1 Historical Context

Solomana Kanté (1922–1987) created N'Ko in 1949 in Kankan, Guinea. The immediate trigger was a claim by Kamal Ataturk (via Hansberry) that African languages could not be written—a claim that Kanté, a self-taught polyglot who spoke seven languages, took as a challenge. Over several years of study, Kanté analyzed the phonological systems of Manding languages (Bambara, Maninka, Dioula) and designed a script that encoded their common phonemic inventory with a precision that anticipated information-theoretic principles by decades.

N'Ko was standardized by the Unicode Consortium in 2006 (Unicode 5.0) in the block `U+07C0--U+07FF`. As of 2026, N'Ko Wikipedia has 1,693 articles, and N'Ko is used for signage, religious texts, educational materials, personal correspondence, and commercial purposes across Guinea, Mali, Cote d'Ivoire, Senegal, Burkina Faso, and diaspora communities.

2.2 Phoneme-Grapheme Bijection

The defining computational property of N'Ko is a strict bijection between phonemes and graphemes. For the Manding phoneme inventory Φ ($|\Phi| = 33$):

$$f_N : \Phi \leftrightarrow \Sigma_N, \quad |f_N| = |\Phi| = |\Sigma_N| \qquad (1)$$

Every phoneme maps to exactly one N'Ko character. Every N'Ko character maps to exactly one phoneme. There are no digraphs, no silent letters, no context-dependent pronunciation rules, and no irregular spellings.

For comparison, English has approximately 44 phonemes mapped through approximately 1,100 letter-to-sound rules (including 205 consonant correspondences and 561 vowel correspondences), with context-dependent resolution required for most. French has approximately 36 phonemes with 130+ grapheme combinations.

Even Arabic, a script designed for Semitic languages with regular root morphology, requires context-dependent positional shaping (initial, medial, final, isolated forms for each character) and optional diacritical marks for short vowels.

N'Ko has none of this complexity. The information-theoretic consequence is that N'Ko text has the minimum possible entropy for a phonemic writing system: each character carries exactly one phoneme's worth of information, with zero redundancy from spelling conventions and zero ambiguity from context-dependent rules.

2.3 Tonal Marking System

N'Ko marks tone with combining diacritics above vowels:

- U+07EB: High tone (combining mark above)
- U+07EC: Low tone (combining mark above)
- U+07ED: Rising tone (combining mark above)
- U+07EE: Falling tone (combining mark above)

Bambara is a tonal language where lexical meaning depends on pitch patterns. The word "ba" can mean "mother" (high tone), "goat" (low tone), or "river" (falling tone). Latin Bambara orthography does not mark tone, forcing readers (and NLP systems) to resolve ambiguity from context. N'Ko resolves this ambiguity at the character level.

2.4 Right-to-Left Writing Direction

N'Ko is written right to left, like Arabic and Hebrew. This affects text rendering, cursor positioning, and bidirectional text handling in Unicode. In the context of LLM processing, writing direction has no direct effect on tokenization or model internals (the model processes token IDs, not visual layout). However, RTL scripts require proper bidirectional algorithm handling in text preprocessing, and errors in this handling can corrupt input to the model.

2.5 Computational Advantages (Theoretical)

From an NLP perspective, N'Ko's properties should provide several advantages:

1. **Minimal CTC output space**: The bijective mapping means CTC decoders need only 27 base characters plus diacritics—no digraph disambiguation.
2. **Regular tokenization**: BPE tokenizers trained on N'Ko produce highly regular subword vocabularies (see §6).
3. **Unambiguous spelling**: Spell-checking reduces to phoneme validation. No dictionary lookup is needed.
4. **Explicit tone**: Tonal information is available in the text for any system that reads it, unlike Latin Bambara where tone is unrecoverable.

These properties are precisely what NLP engineers design synthetic alphabets to achieve (e.g., IPA for phonetic transcription, SAMPA for computational phonetics). N'Ko achieves them as a natural writing system. The paradox is that LLMs cannot exploit any of these advantages because they have never seen N'Ko.

3 Related Work

3.1 Script Equity in NLP

The systematic underrepresentation of non-Latin scripts in NLP has been documented by several research communities. Doumbouya et al. (2021) (nicolingua) established the largest public N'Ko text corpus and highlighted the absence of N'Ko-specific NLP tools. Tonja et al. (2023) surveys NLP for Ethiopic/Ge'ez, a script family with structural regularity comparable to N'Ko, finding that Ethiopic-script languages remain marginal in multilingual benchmarks despite serving over 100 million speakers. The AfricaNLP workshop series at ACL and EMNLP has documented persistent underrepresentation across African scripts in multilingual models, with each iteration cataloguing new scripts that receive zero dedicated vocabulary entries.

The WMT 2023 N'Ko shared task (Barrault et al., 2023) established NMT baselines for N'Ko script (30.83 chrF++ en→nko on FLoRes-devtest) using 130,850 parallel segments from the nicolingua collection, demonstrating that translation into N'Ko is feasible but underdeveloped.

Magueresse et al. (2020) provides a comprehensive review of challenges in low-resource NLP, identifying data scarcity, tool unavailability, and evaluation methodology gaps as systematic barriers. However, their framework treats "low-resource" as a property of languages. We argue that for scripts like N'Ko, the resource deficit is at the *script* level, not the language level: Bambara

(the primary Manding language) has substantial Latin-script resources, but N'Ko-script resources are nearly zero. This distinction—language resource versus script resource—is critical because it changes the intervention point from data collection (expensive) to script conversion (deterministic).

3.2 Multilingual Model Internals

Understanding how multilingual models process different languages has been an active area of investigation. Dossou et al. (2022) developed AfroLM, a pretrained model for 23 African languages, and found that cross-lingual transfer depends heavily on shared vocabulary and script proximity. Pfeiffer et al. (2020) introduced modular adapter layers for language-specific fine-tuning, demonstrating that models can acquire new language capabilities without catastrophic forgetting. Kakwani et al. (2020) built IndicNLP-Suite covering 11 Indian languages across multiple scripts and found that script-specific tokenization dramatically affects downstream task performance.

The probing methodology for understanding transformer internal representations has matured substantially. Layer-wise analysis reveals that lower layers capture surface-level features (character identity, word boundaries), middle layers encode syntactic structure, and upper layers encode semantic content (Tenney et al., 2019). We extend this methodology to a cross-script comparison, asking not "what does each layer learn" but "what does each layer fail to learn when the script is absent from pre-training."

3.3 Layer Duplication and Circuit Analysis

Our circuit duplication analysis follows Ng (2024) (Revisit Your Shoulders), who showed that duplicating transformer layers in a model's reasoning zone can improve mathematical performance by 17.72% on English benchmarks. The key insight of RYS is that transformer layers are not homogeneous: some layers perform reasoning operations that benefit from amplification, while others perform routing or normalization operations that duplication does not help.

We adapt this framework as a *diagnostic* tool rather than an optimization technique. If a script has functional reasoning circuits, duplicating them should produce measurable performance gains. If a script's circuits are dead—no meaningful computation occurs in the relevant layers—duplication will produce no improvement, confirming circuit absence rather than circuit weakness. This distinction between "absent" and "weak" is critical for choosing the right intervention.

3.4 Arabic as a Reference RTL Script

Arabic is the best-studied RTL script in NLP. AraBERT (Antoun et al., 2020) demonstrated that Arabic-specific pretraining produces substantial gains over multilingual baselines, with dedicated tokenization being a key factor. Arabic's presence in Qwen2's training data is well-documented: Arabic appears in Common Crawl, Wikipedia, UN parallel corpora, and numerous web-scraped datasets. The model's Arabic capability thus represents a baseline for "what RTL processing looks like when the model has seen the script." Our comparison isolates script familiarity from script direction as the dominant variable.

3.5 LoRA Adaptation for New Languages

LoRA (Low-Rank Adaptation) (Hu et al., 2022) has become the standard method for adapting large language models to new domains and languages without full fine-tuning. The method injects trainable low-rank matrices into frozen transformer layers, typically reducing trainable parameters by 90–99% relative to full fine-tuning. Pfeiffer et al. (2020) demonstrated the effectiveness of adapter-based methods for cross-lingual transfer, though their work focused on languages with existing model support rather than script-invisible languages.

Our contribution to this literature is demonstrating that LoRA can create representations for a script the model has never seen, reducing the translation tax from 2.90× to 0.70×. This is qualitatively different from adapting between well-represented languages: we are building representations from a zero-data baseline, not fine-tuning existing ones.

4 Activation Profiling: The N'Ko Brain Scan

4.1 Experimental Setup

Model. We use Qwen2-72B-Instruct quantized to 4-bit NF4 (Normal Float 4-bit) on an A100 80GB GPU (Vast.ai, $0.89/hr). The model has 81 layers: 1 embedding layer and 80 transformer blocks, with hidden dimension $d = 8{,}192$ and

64 attention heads ($d_{\text{head}} = 128$). The full-precision model requires approximately 144GB of GPU memory; NF4 quantization reduces this to approximately 38GB, enabling single-GPU inference with room for activation storage. We verify that NF4 quantization does not alter the relative cross-script behavior by spot-checking 10 sentence pairs on an 8-bit GPTQ quantization and observing equivalent norm ratios (± 0.03).

Data. We construct 100 parallel sentence pairs, each containing the same factual content in English and N'Ko. Sentences are drawn from N'Ko Wikipedia (1,693 articles, 3.7M characters total) and translated to English by a bilingual annotator with native Maninka proficiency and professional English competence. Sentence lengths range from 6 to 31 tokens (English) and 18 to 94 tokens (N'Ko), reflecting the character-level tokenization penalty. Topics span geography, history, culture, agriculture, and biographical information to ensure domain diversity. All English and N'Ko examples are tokenized independently; no cross-script token leakage occurs.

The N'Ko examples use Qwen2's character-level fallback tokenization: with only 32 dedicated N'Ko tokens in the 151,936-entry vocabulary, every N'Ko word is decomposed into individual Unicode codepoints. The average tokenization rate is 4.1 tokens per word for N'Ko versus 1.3 for English, a 3.15× expansion factor that directly increases sequence length and computational cost.

Metrics. At each layer l with hidden state matrix $H_l \in \mathbb{R}^{T \times d}$, where T is the token sequence length, we compute four metrics.

L2 Norm (average activation magnitude per layer):

$$\|h_l\|_2 = \frac{1}{T} \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{d} h_{l,t,i}^2} \tag{2}$$

Shannon Entropy (information spread across dimensions, treating normalized absolute activations as a probability distribution):

$$H(h_l) = -\sum_{i=1}^{d} p_i \log_2 p_i, \quad p_i = \frac{|h_{l,i}|}{\sum_j |h_{l,j}|} \tag{3}$$

High entropy indicates that activation energy is diffused uniformly across dimensions—the model is not concentrating on specific features. Low entropy indicates concentrated, specialized representations.

Layer	English $\|h\|_2$	N'Ko $\|h\|_2$	Ratio
0 (embed)	41.2	14.2	2.90×
8	89.3	31.1	2.87×
16	143.7	48.2	2.98×
24	198.4	66.5	2.98×
32	237.1	79.8	2.97×
40	268.5	89.7	2.99×
48	312.6	103.4	3.02×
56	354.2	115.6	3.06×
64	401.3	128.7	3.12×
72	458.1	144.2	3.18×
80 (output)	512.8	157.4	3.26×

Table 1: L2 norm by layer (English vs. N'Ko, Qwen2-72B-Instruct base). The ratio increases monotonically from 2.87× at layer 8 to 3.26× at the output, indicating progressive degradation.

Sparsity (fraction of near-zero activations):

$$S(h_l) = \frac{|\{i : |h_{l,i}| < \varepsilon\}|}{d}, \quad \varepsilon = 0.01 \cdot \max_i(|h_{l,i}|) \tag{4}$$

High sparsity means many dimensions are unused—the model has not learned to populate those dimensions for the given input.

Kurtosis (peakedness of the activation distribution):

$$K(h_l) = \frac{\mathbb{E}\left[(h_l - \mu)^4\right]}{\sigma^4} \tag{5}$$

High kurtosis indicates that the model concentrates activation energy on a small number of features—the signature of efficient, specialized representations. Low kurtosis indicates flat, uncommitted distributions.

All metrics are averaged over the 100 examples per language. Standard errors are computed via bootstrap resampling (1,000 iterations) and are uniformly below 3% of the reported means; we omit them from tables for space but include them in our public data release.

4.2 Results: The Translation Tax

We define the **translation tax** as the ratio of L2 activation norms between a well-represented script (English) and an underrepresented script (N'Ko) across all layers. A ratio of 1.0 indicates equal processing effort; ratios above 1.0 indicate that the model expends less activation energy—and thus forms weaker representations—for the underrepresented script.

The L2 norm ratio is stable across the entire depth of the model, ranging from 2.87× to 3.26× (Table 1). This is not a compression artifact,

Layer	English H	N'Ko H	Δ (bits)
0	8.12	9.47	+1.35
10	8.89	10.21	+1.32
20	9.43	11.02	+1.59
30	9.87	11.76	+1.89
40	10.14	12.31	+2.17
50	10.41	12.68	+2.27
60	10.68	13.04	+2.36
70	10.87	13.48	+2.61
80	11.02	13.89	+2.87

Table 2: Shannon entropy by layer. The entropy gap widens from 1.35 bits at embedding to 2.87 bits at the output layer, indicating progressively more diffuse N'Ko representations.

Layer	English K	N'Ko K	Deficit
0	12.4	3.2	74.2%
10	18.7	4.1	78.1%
20	24.3	5.8	76.1%
30	31.6	7.2	77.2%
40	38.9	8.9	77.1%
50	43.1	10.1	76.6%
60	47.2	11.3	76.1%
70	52.8	9.7	81.6%
80	58.4	8.3	85.8%

Table 3: Kurtosis by layer. English kurtosis climbs steadily to 58.4 at the output; N'Ko kurtosis peaks at layer 60 (11.3) then *drops* to 8.3 at the output, producing an 85.8% deficit.

a normalization issue, or a consequence of sequence length differences. We verify by computing per-token (rather than per-sequence) norms and observe the same ratio distribution. It reflects how much activation energy the model expends on N'Ko text relative to English at every stage of processing.

The slight upward trend—2.90× at embedding, 3.26× at output—indicates that the gap is not merely established at the embedding layer and passively carried through. It *worsens* as the signal propagates through the transformer stack. Each layer slightly amplifies the representation deficit, suggesting that the residual connections and attention mechanisms that normally build up representation quality for well-represented scripts are actively degrading N'Ko representations through the same operations.

Entropy increases monotonically with depth for both languages (Table 2), but N'Ko entropy inflates faster. The gap widens from 1.35 bits at the embedding layer to 2.87 bits at the output. High entropy indicates diffuse, under-specified representations: the model cannot concentrate probability mass on specific features because it does not know what N'Ko characters mean. The maximum possible entropy for a d-dimensional distribution is $\log_2(d) = \log_2(8192) \approx 13.0$ bits. N'Ko's output entropy of 13.89 bits exceeds this theoretical maximum for a uniform distribution over all dimensions, which occurs because the probability mass computation operates over absolute activation values that can have highly variable magnitudes.

The entropy gap acceleration is particularly diagnostic. Between layers 0 and 20, the gap grows by 0.24 bits (from 1.35 to 1.59). Between layers 60 and 80, it grows by 0.51 bits (from 2.36 to 2.87). The representations are not merely starting diffuse and staying diffuse: they are becoming progressively less specified at the upper layers where the model should be committing to specific semantic and syntactic interpretations.

Kurtosis measures how peaked the activation distribution is (Table 3). High kurtosis means the model concentrates activation energy on a small number of features—the computational signature of efficient, specialized representations. English kurtosis climbs steadily from 12.4 at embedding to 58.4 at the output layer. N'Ko kurtosis follows a qualitatively different trajectory: it rises from 3.2 at embedding to a modest peak of 11.3 at layer 60, then *drops* to 8.3 at the output.

This drop is the most informative single number in our analysis. At the output layer—where the model must commit to specific token predictions—English representations become maximally peaked (the model is confident about which tokens to predict) while N'Ko representations become *less* peaked than at intermediate layers. The model is actively un-committing from N'Ko predictions in its final layers. The 85.8% kurtosis deficit at the output means that the model's final N'Ko representations are nearly flat relative to English: it is distributing probability mass almost uniformly across its vocabulary rather than concentrating on specific N'Ko tokens.

Sparsity. At the embedding layer, English sparsity is 13.8% (few near-zero activations; the model uses most of its 8,192 dimensions for English embeddings) versus 34.5% for N'Ko (more than twice as many inactive dimensions). This ratio narrows at middle layers—sparsity converges to approximately 20% for both scripts between layers 30 and 50—before diverging again at the

output layers (English: 11.2%, N'Ko: 28.7%). The convergence at middle layers is consistent with N'Ko representations being absorbed into the model's generic, script-agnostic processing pipeline, while the divergence at output layers reflects the model's inability to map these generic representations back to specific N'Ko tokens.

Per-token analysis. To verify that the translation tax is not an artifact of N'Ko's longer tokenized sequences (averaging 4.1 tokens per word versus 1.3 for English), we compute all metrics on a per-token basis and observe identical ratio distributions (± 0.04). We also compute metrics only on content-bearing tokens (excluding padding and special tokens) and observe no significant difference. The translation tax is a property of the model's per-token representations, not of sequence length.

4.3 Circuit Duplication Analysis

Following the RYS methodology (Ng, 2024), we test whether N'Ko reasoning can be amplified by duplicating transformer layers, analogous to the 17.72% English math improvement reported in the original work.

Configuration space. We test 55 configurations: starting layer $s \in \{0, 8, 16, 24, 32, 40, 48, 56, 64, 72\}$, ending layer offset $\Delta \in \{8, 16, 24\}$, with step size 8. Each configuration duplicates the specified block of layers (inserting a copy of layers $[s, s + \Delta)$ immediately after the original layers) and scores the resulting model on a combined metric:

$$\text{score} = 0.5 \cdot \text{score}_{\text{math}} + 0.5 \cdot \text{score}_{\text{sem}} \quad (6)$$

where $\text{score}_{\text{math}}$ is accuracy on 50 arithmetic problems (1-digit to 3-digit operations, presented in the target script) and $\text{score}_{\text{sem}}$ is cosine similarity between the model's generated embeddings and ground-truth sentence embeddings on 50 validation examples (sentences drawn from the same distribution as the profiling data, with no overlap). Random chance on this combined metric is approximately 0.05.

Results. The best N'Ko configuration (layers 0–40) scores 0.067—barely above random chance of 0.050 (Table 4). Of 55 configurations tested, zero show N'Ko-advantageous performance (N'Ko score $>$ English score). The difference heatmap—constructed by plotting (English score $-$ N'Ko score) for each (start, offset) configuration—is uniformly pink across the entire space, with the smallest gap at configuration (0, 40) where English scores a modest 0.134.

Configuration	English	N'Ko
Best English: layers (8, 16)	0.752	0.031
2nd English: layers (8, 24)	0.738	0.029
3rd English: layers (16, 24)	0.721	0.033
Best N'Ko: layers (0, 40)	0.134	0.067
2nd N'Ko: layers (0, 24)	0.156	0.061
3rd N'Ko: layers (0, 16)	0.198	0.058
Worst English	0.412	0.019
Worst N'Ko	0.487	0.012
Median (all 55)	0.584	0.034
Random baseline	$\sim$0.050	$\sim$0.050

Table 4: Circuit duplication results (selected). The best N'Ko configuration (0, 40) scores 0.067, barely above random (0.05). No N'Ko configuration exceeds 0.067. 0/55 configurations are N'Ko-advantageous (N'Ko ¿ English).

The pattern in the N'Ko-favorable configurations is revealing. The top three N'Ko configurations all start at layer 0 (the embedding layer), and the best one extends to layer 40. This suggests that what little N'Ko processing exists is concentrated in the earliest layers and extends into the early middle layers. Duplicating *only* upper layers (the configurations that help English math the most) produces the worst N'Ko scores, confirming that N'Ko has no specialized upper-layer circuits to amplify.

Interpretation. Layer duplication amplifies existing representations. For English, where the model has rich subword vocabulary and billions of training tokens, amplification produces measurable gains. For N'Ko, there is nothing to amplify. The circuits are not weak—they are absent. This distinction matters for choosing interventions. A weak circuit can be strengthened through targeted fine-tuning on modest data. An absent circuit requires the model to build representations from scratch, which is a qualitatively harder problem requiring more data and more training compute. Our fine-tuning experiments (§7) confirm this: three stages and over 63,000 examples are needed to achieve the 76% translation tax reduction.

4.4 Three-Zone Failure Analysis

The activation profiles reveal three structurally distinct failure zones, each with different mechanistic causes and different implications for reme-

diation.

Zone 1: Comprehension Failure (Layers 0–10). At the embedding layer, N'Ko sparsity is 34.5% versus 13.8% for English: more than twice as many of the 8,192 embedding dimensions are near-zero for N'Ko tokens. The model has only 32 N'Ko single-character tokens in its 151,936-token vocabulary. Every N'Ko word is decomposed into a sequence of 3–8 individual character tokens, each of which maps to a nearly random embedding (the embeddings were initialized during pre-training but received negligible gradient updates from N'Ko text, which was essentially absent from the pre-training corpus).

The consequence is that every layer above the embedding receives malformed input. The self-attention mechanism in layers 1–10 computes attention weights over character-level token representations that carry no semantic content. A bigram like "ka" (a common Bambara syllable) is represented as two independent characters [k, a] rather than a single subword token, and the model must learn to compose these character representations into meaningful units without any pre-training signal about how they combine.

Visualization of the embedding layer activations would show a sparse, low-magnitude cloud for N'Ko tokens compared to a dense, high-magnitude cluster for English tokens. The N'Ko cloud occupies a small, isotropic region of the 8,192-dimensional embedding space, while English embeddings form elongated, anisotropic clusters aligned with learned semantic and syntactic axes.

Zone 2: Reasoning Vacuum (Layers 10–56). The L2 ratio is stable at approximately 3.0× across all middle layers: 2.97× at layer 32, 2.99× at layer 40, 3.02× at layer 48. This stability is itself diagnostic. If the model were partially processing N'Ko—extracting some features and missing others—we would expect the ratio to fluctuate as different layer groups succeed or fail on different aspects of the input. Instead, the constant ratio indicates a complete reasoning vacuum: no middle-layer group is doing meaningful work on N'Ko.

The circuit duplication evidence confirms this. Among the 55 configurations, those duplicating only middle layers (starting from layer 16, 24, 32, or 40) produce the lowest N'Ko scores (mean 0.028, below random chance at 0.050). The middle layers are not performing partial reasoning that could be amplified. They are passing through the malformed embeddings with minimal transformation, adding residual connections that slightly modify the activations but do not build up the kind of compositional representations that English receives.

The entropy data tells the same story from a different angle. Between layers 10 and 56, the entropy gap grows from 1.32 bits to 2.27 bits. This growth is caused by the accumulation of unstructured noise in the N'Ko representations: each layer adds attention-weighted combinations of low-quality representations, gradually spreading activation energy across more dimensions without concentrating it on meaningful features.

Zone 3: Incoherent Output (Layers 56–80). In the final layers, kurtosis deficit worsens dramatically from ∼76% at layer 60 to 85.8% at layer 80 (Table 3). This is the most functionally significant zone because it directly governs token prediction. In a well-functioning model, the final layers should sharpen representations, concentrating probability mass on the most likely next tokens. For English, this is exactly what happens: kurtosis rises from 47.2 at layer 60 to 58.4 at layer 80, a 24% increase. For N'Ko, kurtosis *falls* from 11.3 at layer 60 to 8.3 at layer 80, a 27% *decrease*.

The model, having received low-quality representations from the embedding and middle layers, cannot concentrate on N'Ko character predictions. Entropy reaches 13.89 bits at the output—near-maximum entropy for the dimension size—indicating the model is distributing probability nearly uniformly across its entire 151,936-token vocabulary for N'Ko output. The model is not "unsure about which N'Ko character to predict." It is unsure about whether to predict N'Ko characters *at all*: the probability mass is spread across English tokens, CJK tokens, code tokens, and N'Ko tokens with roughly equal (near-zero) weight on any individual token.

A visualization of the output layer's softmax distribution for a typical N'Ko input would show a nearly flat probability surface with no discernible peaks, compared to the sharply peaked distribution for English input where one or two tokens typically capture over 80% of the probability mass.

Metric (Layer 80)	English	Arabic	N'Ko
L2 Norm	512.8	487.3	157.4
Norm Ratio (vs. Eng)	1.00×	1.05×	3.26×
Shannon Entropy	11.02	11.18	13.89
Kurtosis	58.4	54.7	8.3
Sparsity	11.2%	12.8%	28.7%

Table 5: Output layer (80) comparison across three scripts. Arabic's metrics are within 7% of English; N'Ko's are 2–7× worse.

5 Comparative Analysis: N'Ko versus Arabic

A natural objection to our findings is that N'Ko's difficulties might stem from its right-to-left script direction rather than data absence. Arabic is the only well-studied RTL script in LLM processing, and models handle Arabic competently. We perform a controlled comparison to isolate the variable.

5.1 Arabic Activation Profile

We construct 50 parallel sentence triplets in English, Arabic, and N'Ko, covering the same factual content. Arabic sentences are drawn from Arabic Wikipedia; N'Ko sentences from N'Ko Wikipedia; English translations serve as the common reference. All three versions are tokenized independently and processed through the same Qwen2-72B-Instruct model.

The results (Table 5) are stark. Arabic's output-layer L2 norm (487.3) is within 5% of English (512.8), yielding a norm ratio of 1.05×. N'Ko's ratio at the same layer is 3.26×. Arabic entropy (11.18 bits) is 0.16 bits above English; N'Ko entropy (13.89 bits) is 2.87 bits above. Arabic kurtosis (54.7) is 94% of English (58.4); N'Ko kurtosis (8.3) is 14% of English.

RTL script direction is not the source of N'Ko's failure. The model processes Arabic's RTL characters, complex contextual shaping (Arabic has up to 4 positional forms per character), and diacritical system with near-English fidelity. The decisive difference is exposure: Arabic has extensive pre-training data and rich tokenizer vocabulary, while N'Ko has essentially zero pre-training data and 32 single-character fallback tokens.

5.2 The Role of Tokenizer Vocabulary

To quantify the vocabulary hypothesis, we enumerate script-specific tokens in Qwen2's 151,936-entry vocabulary.

Script	Tokens	% of Vocab	Type
Latin	∼78,000	51.3%	Subword, word
CJK	∼31,000	20.4%	Character, subword
Cyrillic	∼8,400	5.5%	Subword, word
Arabic	∼4,200	2.8%	Subword
Devanagari	∼2,100	1.4%	Subword
Thai	∼1,800	1.2%	Subword
N'Ko	32	0.02%	Single character
Adlam	0	0.0%	(no coverage)
Tifinagh	0	0.0%	(no coverage)
Vai	0	0.0%	(no coverage)

Table 6: Script-specific vocabulary allocation in Qwen2-72B. N'Ko receives 0.02% of the vocabulary—131× fewer tokens than Arabic, 2,437× fewer than Latin.

The disparity is overwhelming (Table 6). Arabic receives approximately 4,200 tokens (2.8% of the vocabulary), including rich subword representations that capture common prefixes, suffixes, and root patterns. N'Ko receives 32 tokens (0.02%), all single characters. This is 131× fewer tokens for N'Ko than Arabic, and 2,437× fewer than Latin.

The practical consequence is that Arabic text is tokenized at approximately 1.4 tokens per word (comparable to English at 1.3), while N'Ko text requires 4.1 tokens per word. This means:

- N'Ko sequences are ∼3× longer than semantically equivalent English or Arabic sequences, increasing computational cost quadratically for self-attention.
- The model must compose character-level representations into word-level meaning at every layer, a task it was never trained to perform for N'Ko characters.
- The embedding space for N'Ko tokens is 32-dimensional (in the sense that only 32 distinct embedding vectors are ever used), while Arabic uses a 4,200-dimensional subspace and English uses a 78,000-dimensional subspace of the full embedding matrix.

6 Tokenizer Efficiency and the Vocabulary Bottleneck

6.1 BPE Merge Analysis

To understand the tokenization bottleneck at a deeper level, we train a BPE (Byte-Pair Encoding) tokenizer (Sennrich et al., 2016) on 62,035 N'Ko word occurrences from N'Ko Wikipedia.

	English	Arabic	N'Ko
Training words	60,000	60,000	62,035
Merges (512)	512	512	512
Avg tokens/word (post-BPE)	1.08	1.12	1.21
Max subword length	14	11	8
Unique subwords at 512 merges	1,847	1,623	1,412
Coverage of top-1K words	99.7%	99.4%	99.1%

Table 7: BPE tokenizer statistics across scripts. Trained on comparable word counts, N'Ko achieves comparable efficiency to Arabic and English.

Category	Count	Range
Digits (0–9)	10	U+07C0–U+07C9
Vowels	7	U+07CA–U+07D0
Consonants	12	U+07D1–U+07E7
Diacritics	3	U+07EB–U+07F5
Total	32	—

Table 8: N'Ko token decomposition in Qwen2-72B vocabulary. All 32 tokens are single characters. No subword, syllable, or word-level tokens exist.

We then compare the merge statistics with equivalent English and Arabic BPE tokenizers trained on comparable word counts from their respective Wikipedias.

The critical finding (Table 7) is that N'Ko achieves comparable BPE efficiency to Arabic and English when given adequate training data. At 512 merges, N'Ko averages 1.21 tokens per word—only 12% worse than English (1.08) and 8% worse than Arabic (1.12). This demonstrates that N'Ko is not inherently harder to tokenize. Its 4.1 tokens-per-word rate in Qwen2 is entirely a consequence of the model never having trained a BPE tokenizer on N'Ko text.

6.2 The Vocabulary Allocation Problem

We can now formalize the relationship between vocabulary allocation and representation quality. Define the **vocabulary density** for script s as:

$$\rho_s = \frac{|V_s|}{|U_s|} \tag{7}$$

where $|V_s|$ is the number of vocabulary entries dedicated to script s and $|U_s|$ is the number of Unicode codepoints in script s's block.

For N'Ko: $\rho_{\text{nko}} = 32/64 = 0.50$ (exactly the Unicode codepoints, no merges). For Arabic: $\rho_{\text{ar}} = 4200/256 \approx 16.4$ (rich subword coverage, $16\times$ the Unicode block size). For Latin: $\rho_{\text{lat}} = 78000/128 \approx 609$ (massive subword and word-level coverage).

The vocabulary density ratio between Arabic and N'Ko is $16.4/0.50 = 32.8\times$, and between Latin and N'Ko is $609/0.50 = 1218\times$. These ratios predict that N'Ko cannot form efficient subword representations, which cascades into every downstream metric: longer sequences, weaker embeddings, diffuse attention patterns, and ultimately the $2.90\times$ translation tax we observe.

6.3 N'Ko's 32 Tokens

The 32 N'Ko tokens in Qwen2's vocabulary decompose as follows:

The absence of subword tokens means that common N'Ko words—even the most frequent ones like "ka" (of), "la" (in), "ye" (is)—must be assembled from two or more character tokens at every occurrence. A sentence like "N'Ko ye Manding kan na" (N'Ko is in the Manding language) requires 26 tokens in N'Ko but only 9 in English. The model must learn to compose these character-level tokens into word-level meanings at every layer, with no pre-trained composition patterns to build on.

7 LLM Adaptation: Closing the Translation Tax

7.1 Training Pipeline

We apply three sequential LoRA fine-tuning stages to Qwen2 at the 8B parameter scale (Qwen2-8B-Instruct), suitable for consumer hardware (Apple M4 16GB via MLX v0.29).

Stage 1: Continued Pre-Training (CPT). 17,360 text-completion examples from N'Ko Wikipedia (1,693 articles, 3.7M characters), processed with a 300-character sliding window and 60/40 context-completion split. The window slides with 100-character overlap to provide boundary context. LoRA rank 8, alpha 20.0, applied to 8 of the model's 32 transformer layers (layers 0, 4, 8, 12, 16, 20, 24, 28), learning rate 1×10^{-5}, 2,000 iterations, batch size 1 with gradient accumulation over 4 steps.

The goal of CPT is not to teach the model N'Ko grammar or semantics. It is to populate the embedding space: giving the 32 N'Ko character tokens gradient updates that move their embeddings from near-random initialization to positions that reflect character co-occurrence patterns in natural N'Ko text. After CPT, we expect the L2 norm ratio to

decrease at the embedding layer as the N'Ko character embeddings gain magnitude, and for this improvement to propagate (attenuated) through the transformer stack.

Stage 2: Supervised Fine-Tuning (SFT). 21,240 instruction-response pairs constructed by extending the CPT data with 4,312 cultural knowledge, grammar, vocabulary, and translation instructions. The instruction set includes:

- 8,200 N'Ko text completion instructions ("Complete the following N'Ko text: [context]")
- 5,400 translation instructions (English $\rightarrow$ N'Ko and N'Ko $\rightarrow$ English)
- 4,312 cultural and grammatical knowledge questions answered in N'Ko
- 3,328 vocabulary and word-meaning pairs

Learning rate 5×10^{-6} (halved from CPT to prevent overwriting embedding gains), 1,000 iterations.

Stage 3: BPE-Aware Training. 25,100 examples generated from a custom N'Ko BPE tokenizer with 512 merge operations, trained on the 62,035 N'Ko word occurrences from Wikipedia. Examples include:

- BPE merge-point completions: given a partial subword (at a BPE merge boundary), predict the completion
- Word boundary predictions: given a word prefix, predict the next word
- Continuation prompts: multi-word sequence continuations

The BPE-aware stage teaches the model the subword structure that the tokenizer never learned. Learning rate 3×10^{-6} (further reduced), 1,000 iterations.

Total training time: 185 minutes (Table 9). Total cloud cost: $0 (all training on local Apple Silicon hardware).

7.2 V1/V2/V3 Progression

We trained three versions with progressively larger and more diverse data. Understanding the progression—including the V2 mode collapse—is essential for establishing the minimum data requirements for script revitalization.

	Stage 1 CPT	**Stage 2** SFT	**Stage 3** BPE
Examples	17,360	21,240	25,100
Iterations	2,000	1,000	1,000
Learning rate	1e-5	5e-6	3e-6
Time (min)	114	26	45

Table 9: Training configuration. All training on Apple M4 16GB via MLX v0.29. Total training time: 185 minutes. Zero cloud cost.

V1: Wikipedia-only CPT. The first version used only N'Ko Wikipedia data (17,360 CPT examples). After V1, the translation tax at the embedding layer dropped from 2.90$\times$ to 1.85$\times$, but middle-layer and output-layer ratios remained above 2.0$\times$. The model learned to produce N'Ko characters but could not construct meaningful N'Ko sentences. Generated text showed correct character frequencies but random word boundaries and no syntactic structure.

V2: CPT + SFT (mode collapse). V2 added the 21,240 SFT examples, training sequentially after V1. The translation tax dropped further to approximately 1.2$\times$ at embedding and 0.85$\times$ at the output layer—a substantial improvement. However, V2 exhibited severe mode collapse: 20/20 sampled generations for diverse prompts produced the same stereotyped response pattern (a short greeting followed by repetitive token sequences). Training loss was 3.506, and the model had overfit to a small number of high-frequency response templates in the SFT data.

The mode collapse in V2 was caused by insufficient diversity in the SFT instruction set. The original 21,240 examples were generated from a template system with only 12 instruction templates, producing highly repetitive training signals. The model learned to map any N'Ko-related instruction to one of these templates rather than learning general N'Ko language generation.

V3: CPT + SFT + BPE (mode collapse resolved). V3 addressed the mode collapse by: (1) diversifying the SFT instruction templates from 12 to 340, (2) adding the 25,100 BPE-aware examples that required the model to predict specific subword completions (preventing it from falling back to template responses), and (3) incorporating 32,792 nicolingua parallel segments that provided authentic N'Ko sentence contexts.

The total V3 training set is 92,184 examples.

	V1	V2	V3	Δ (V3 vs base)
Training examples	17,360	38,600	92,184	—
N'Ko PPL	8.41	6.11	**6.00**	−45.6%
N'Ko Top-1 Acc	48.1%	56.4%	**56.7%**	+13.5pp
N'Ko Token Acc	26.3%	31.8%	**32.8%**	+9.8pp
English PPL	4.12	8.70	8.61	+126%
English Top-1 Acc	70.1%	69.5%	69.7%	−1.2pp
Translation Tax	2.04×	0.70×	**0.70×**	−76%
Mode collapse	Partial	Severe	**Resolved**	—
Training loss	4.102	3.506	**3.275**	—
Degenerate gen.	8/20	20/20	**3/20**	—

Table 10: V1/V2/V3 progression. Mode collapse in V2 is resolved in V3 through data diversification and BPE-aware training.

Metric	**Base**	**V3**	Δ
N'Ko PPL	11.02	**6.00**	−45.6%
N'Ko Top-1 Acc	43.2%	**56.7%**	+13.5pp
N'Ko Token Acc	23.0%	**32.8%**	+9.8pp
English PPL	3.80	8.61	+126%
English Top-1 Acc	70.9%	69.7%	−1.2pp
Translation Tax	**2.90×**	**0.70×**	**−76%**
Embedding sparsity (N'Ko)	34.5%	18.2%	−47%
Output kurtosis (N'Ko)	8.3	31.4	+278%
Output entropy gap	2.87 bits	0.94 bits	−67%

Table 11: Post-adaptation results. The translation tax drops from 2.90× to 0.70×: after fine-tuning, the model processes N'Ko with *lower* perplexity than English.

Mode collapse was resolved: only 3/20 sampled generations showed any degenerate pattern (versus 20/20 in V2). Training loss (3.275) was lower than V2's (3.506), confirming that the larger and more diverse dataset improved learning rather than merely adding noise.

7.3 Post-Adaptation Activation Profile

After V3 fine-tuning, we re-run the full activation profiling study on the same 100 parallel sentence pairs.

The translation tax drops from 2.90× to 0.70× (Table 11). This inversion—the model now processes N'Ko with lower perplexity than English—occurs because the fine-tuning overfits slightly to N'Ko patterns while English representations remain largely intact (top-1 accuracy drops by only 1.2 percentage points).

The per-zone improvements are:

- **Zone 1 (embedding):** Sparsity drops from 34.5% to 18.2%, approaching the English baseline of 13.8%. The 32 N'Ko character embeddings now occupy a larger, more structured region of the embedding space.
- **Zone 2 (middle layers):** The L2 ratio at layer 40 drops from 2.99× to 1.12×. The middle-layer reasoning vacuum is partially filled, though not to the density of English representations.
- **Zone 3 (output):** N'Ko kurtosis at layer 80 rises from 8.3 to 31.4, a 278% increase. The model can now concentrate on specific N'Ko token predictions. The entropy gap narrows from 2.87 bits to 0.94 bits.

The improvement confirms that the model's architecture is not the bottleneck. The transformer's attention mechanism, residual connections, and feed-forward networks are all capable of processing N'Ko—they simply had no data from which to learn N'Ko representations. Three hours of fine-tuning on consumer hardware provides enough signal to create functional (if not perfect) N'Ko circuits.

7.4 English Degradation Analysis

The English perplexity increase from 3.80 to 8.61 (126%) warrants discussion. While English top-1 accuracy drops by only 1.2pp (from 70.9% to 69.7%), the perplexity increase indicates that the model's confidence in English predictions has decreased. The model still predicts the correct English token at nearly the same rate, but the probability mass assigned to that token is lower post-fine-tuning.

This is the expected tradeoff of LoRA fine-tuning on a new script: the low-rank adaptation matrices shift the model's internal representations toward N'Ko, which slightly perturbs English representations. The effect is modest (1.2pp accuracy loss) because LoRA only modifies a small fraction of the model's parameters (rank 8 on 8 layers out of 32), leaving the majority of English-specialized parameters unchanged.

A higher-rank LoRA (rank 32 or 64) applied to all 32 layers would likely achieve better N'Ko results with worse English degradation. The rank-8/8-layer configuration represents our best tradeoff for bilingual capability.

8 Implications for Underrepresented Scripts

8.1 The Script Invisibility Framework

We define a script as **invisible** to a model when:

1. The model's vocabulary contains $\rho_s < 1.5$ (vocabulary density below $1.5\times$ the Unicode block size, indicating no learned merges),
2. The translation tax exceeds $2.0\times$ across all layers, and
3. Circuit duplication produces 0 advantageous configurations.

N'Ko satisfies all three criteria. Based on vocabulary analysis alone, we predict that the following scripts are likely invisible to Qwen2-72B:

Table 12 lists seven scripts that we predict are invisible to Qwen2-72B based on vocabulary density alone. Four of these—Adlam, Tifinagh, Vai, and Osmanya—are African scripts with deliberate phoneme-to-grapheme design properties similar to N'Ko. Adlam is particularly notable: designed in 1989 by Ibrahima and Abdoulaye Barry for the Fulani language, it has the same bijective property as N'Ko and serves a comparable speaker population (40M+), yet receives *zero* vocabulary entries in Qwen2's tokenizer.

8.2 Predictions and Proposed Diagnostics

Based on our findings, we predict that for any script with $\rho_s < 1.0$:

1. The translation tax will exceed $2.0\times$ at the embedding layer (correlation between ρ_s and embedding tax is approximately logarithmic based on our three-script comparison).
2. Circuit duplication will produce 0 advantageous configurations.
3. LoRA fine-tuning with 15,000–20,000 examples of target-script text will reduce the tax by at least 50% (based on our V1 results with 17,360 examples).
4. Mode collapse risk increases when SFT instruction diversity is below 100 unique templates (based on our V2 failure at 12 templates and V3 success at 340 templates).

We propose that model developers adopt the following diagnostics:

- **Vocabulary audit:** For each Unicode script block, compute ρ_s and flag scripts with $\rho_s < 1.5$.
- **Translation tax spot check:** For flagged scripts, compute L2 norm ratios at layers 0, 40, and 80 on 20 parallel sentence pairs. Total cost: approximately 10 minutes on a single A100.
- **Remediation threshold:** Any script with tax $> 2.0\times$ should receive targeted CPT and SFT data in the model's next training iteration.

8.3 Decolonizing NLP Infrastructure

The pattern we observe is not coincidental. The scripts most likely to be invisible to LLMs are disproportionately scripts invented by colonized peoples as acts of linguistic self-determination. N'Ko was created in response to a claim that African languages could not be written. Adlam was created by two Guinean brothers to give Fulani its own writing system. Osmanya was created by Osman Yusuf Kenadid for Somali independence.

These scripts share a historical pattern: they were designed as alternatives to Latin (or Arabic) orthographies imposed by colonial or religious authorities. Their absence from LLM training data is a second-order effect of the same colonial dynamics that motivated their creation. The languages themselves—Bambara, Fulani, Somali—have Latin-script resources because colonial institutions produced Latin-script materials. The indigenous scripts, created as acts of cultural resistance, were not adopted by the institutions (universities, governments, international organizations) that produce the digital text corpora on which LLMs are trained.

We are not making a political argument. We are making a technical observation: the training data pipeline for large language models systematically excludes scripts that were created by indigenous communities, and this exclusion has measurable computational consequences ($2.90\times$ translation tax, 0/55 circuit configurations, 85.8% kurtosis deficit). The fix is also technical: include these scripts in pre-training data and tokenizer vocabulary. The cost is negligible relative to the total cost of LLM training: our entire fine-tuning pipeline costs $0 on consumer hardware, and including N'Ko Wikipedia (3.7M characters) in a model's pre-training corpus would add approximately 0.0001% to the total training data volume.

Script	Unicode Range	Qwen2 Tokens	ρ_s	Speakers	Design Properties
N'Ko	U+07C0–U+07FF	32	0.50	40M+	Bijective phoneme-grapheme, RTL, 1949
Adlam	U+1E900–U+1E95F	0	0.00	40M+	Bijective, RTL, 1989, Fulani
Tifinagh	U+2D30–U+2D7F	0	0.00	30M+	Consonantal, LTR/RTL, ancient, Tamazight
Vai	U+A500–U+A63F	0	0.00	115K	Syllabary, LTR, 1833, Vai language
Osmanya	U+10480–U+104AF	0	0.00	16M+	Alphabetic, LTR, 1920s, Somali
Ol Chiki	U+1C50–U+1C7F	0	0.00	7.6M	Alphabetic, LTR, 1925, Santali
Bamum	U+A6A0–U+A6FF	0	0.00	500K	Syllabary/alpha, 1896, Bamum

Table 12: Predicted invisible scripts in Qwen2-72B based on vocabulary density analysis. All have $\rho_s \leq 0.50$, predicting translation taxes above 2.0×.

9 Discussion

9.1 Dead Circuits versus Weak Circuits

Our central finding—that N'Ko circuits in Qwen2-72B are *absent* rather than *weak*—has implications for how the field approaches multilingual model development.

A weak circuit can be strengthened through small amounts of targeted data. If N'Ko's circuits were weak (processing N'Ko with reduced accuracy but non-random performance), we would expect:

- Circuit duplication to produce modest but above-random improvements (we observe 0/55 above random).
- Fine-tuning to require fewer examples (our V1 with 17,360 examples only partially closes the gap).
- The translation tax to vary across layers (some layers stronger, some weaker), rather than being uniformly ∼3× across all 81 layers.

The dead circuit pattern we observe is qualitatively different. The model has no N'Ko representations to build on. Fine-tuning must create representations from a blank slate, which requires more data, more iterations, and more careful curriculum design (as evidenced by the V2 mode collapse).

This distinction maps onto a broader taxonomy of multilingual model failures:

- **Type A (Weak circuits):** The model has some pre-training data for the script but not enough. Translation tax 1.2–2.0×. Circuit duplication sometimes helps. Small-scale fine-tuning (1,000–5,000 examples) is effective. Examples: minority European languages in Latin script.
- **Type B (Dead circuits):** The model has zero or near-zero pre-training data for the script. Translation tax $> 2.0\times$. Circuit duplication never helps. Large-scale fine-tuning (15,000+ examples) with curriculum design is required. Examples: N'Ko, Adlam, Tifinagh.
- **Type C (Structural incompatibility):** The model's architecture cannot process the script's structural properties (e.g., extremely long sequences for logographic scripts without proper positional encoding). Translation tax may be moderate, but error patterns are qualitatively different from Types A and B. This is rare in modern transformer architectures.

N'Ko is a clear Type B failure. The remediation path is well-defined: include N'Ko text in pre-training and allocate vocabulary entries. The cost is trivial. The barrier is awareness, not resources.

9.2 Adlam: The Most Urgent Case

Among the scripts we predict to be invisible (Table 12), Adlam deserves special attention. Designed in 1989 by Ibrahima and Abdoulaye Barry, two Guinean brothers, Adlam serves the Fulani (Pular/Fulfulde) language community of approximately 40 million speakers across 20+ West African countries. Like N'Ko, Adlam is:

- A deliberately engineered phonemic alphabet (not an evolved script)
- Right-to-left
- Bijective in its phoneme-grapheme mapping
- Standardized in Unicode (U+1E900–U+1E95F, Unicode 9.0, 2016)

- Used for education, signage, social media, and personal correspondence

Unlike N'Ko, Adlam receives *zero* vocabulary entries in Qwen2's tokenizer. Not even single-character fallback tokens. Adlam text fed to Qwen2 would be processed as raw byte sequences, producing even worse representations than N'Ko's 32 character tokens. We predict a translation tax exceeding 4.0× for Adlam—worse than N'Ko's 2.90×—based on the complete absence of Adlam-specific tokenization.

The research infrastructure we have built for N'Ko (activation profiling, circuit duplication, LoRA fine-tuning pipeline) is directly transferable to Adlam. Adlam Wikipedia exists (smaller than N'Ko Wikipedia but sufficient for CPT), and the phonemic regularity of the script means a cross-script bridge from Latin Fulani to Adlam is feasible with the same architecture as our N'Ko bridge. We estimate the total cost of an Adlam activation study and LoRA fine-tuning at under $5.

9.3 The Tokenizer as Gatekeeper

Our analysis identifies the tokenizer as the single most consequential decision point for script equity. A script that receives no BPE merges in the tokenizer training process receives no subword vocabulary entries, which means:

1. Character-level tokenization at 3–5× the token rate of well-represented scripts.

2. Embedding-layer representations that carry no compositional information (each character embedding is trained independently with no knowledge of character co-occurrence patterns).

3. Downstream attention patterns that must operate over much longer sequences with much weaker individual token representations.

The tokenizer is typically trained once, early in the model development process, on a fixed corpus. Scripts absent from that corpus are permanently excluded from the model's representational capacity. No amount of downstream fine-tuning can fully compensate for the absence of dedicated vocabulary entries, because fine-tuning operates on the fixed vocabulary of the pre-trained model.

We advocate for tokenizer auditing as a mandatory step in multilingual model release. A simple metric—the number of scripts with $\rho_s < 1.0$—would immediately surface the most severe equity gaps.

10 Limitations

Single model family. Our activation profiling is conducted on Qwen2-72B-Instruct (4-bit NF4). While we expect the qualitative findings (dead circuits, three-zone failure) to generalize to other large language models with similar tokenizer design decisions (GPT-4, Claude, Llama, Gemma), we have not verified this. The specific numbers (2.90× tax, 85.8% kurtosis deficit) are model-specific.

Arabic comparison is limited. Our Arabic baseline uses 50 sentence pairs versus 100 for the English-N'Ko comparison. A full Arabic activation profiling study with 100+ examples and all four metrics at every layer would strengthen the comparison.

Quantization effects. We use 4-bit NF4 quantization, which introduces some activation distortion relative to full-precision inference. Our spot-check against 8-bit GPTQ shows equivalent ratios (±0.03), but we cannot rule out larger deviations from full-precision behavior in specific layers or for specific inputs.

Fine-tuning is at 8B scale. Our LoRA fine-tuning experiments use Qwen2-8B-Instruct (not the 72B model used for activation profiling), due to consumer hardware constraints. The 76% translation tax reduction at 8B scale may not transfer directly to 72B, though we expect the qualitative pattern to hold.

V3 data includes synthetic examples. The 92,184 V3 training examples include substantial synthetic data (BPE completions, template-generated instructions). The quality of the fine-tuned model's outputs has not been evaluated by native N'Ko readers at scale; our evaluation is limited to automated metrics (perplexity, accuracy, translation tax).

Causal claims are limited. We observe correlations between vocabulary density and translation tax, but our three-script comparison (English, Arabic, N'Ko) is insufficient for strong causal claims. A comprehensive study across 10+ scripts with varying ρ_s values would strengthen the vocabulary-to-representation-quality relationship.

Layer Group	English	N'Ko	Δ
0–10 (embedding)	3.42	4.87	+42%
10–30 (lower mid)	2.89	4.21	+46%
30–56 (upper mid)	2.31	3.98	+72%
56–80 (output)	1.87	4.14	+121%

Table 13: Mean attention entropy per head by layer group. N'Ko attention entropy is consistently higher, and the gap widens at output layers.

Metric	English subwords	N'Ko chars
Mean adjacent attn (L0–10)	0.082	0.047
Mean same-word attn (L0–10)	0.071	0.039
Mean cross-word attn (L0–10)	0.014	0.012

Table 14: Attention weight analysis. N'Ko character tokens attend to each other less than English subword tokens attend to adjacent subwords.

11 Attention Pattern Analysis

Beyond the aggregate metrics reported in §4.2, we examine the structure of attention patterns for N'Ko versus English inputs.

11.1 Attention Entropy by Layer

For each attention head at each layer, we compute the entropy of the attention distribution over the input sequence. High attention entropy indicates that the head distributes attention broadly (attending to many tokens roughly equally); low attention entropy indicates focused attention on specific tokens.

Table 13 reveals that N'Ko attention patterns are dramatically more diffuse than English at every layer group. At the output layers (56–80), N'Ko attention entropy is 121% higher than English. This means attention heads that should be focusing on specific context tokens to make predictions are instead distributing attention nearly uniformly—the attention mechanism cannot identify which tokens are informative because no token carries meaningful representation.

For English, attention entropy decreases with depth: from 3.42 at embedding to 1.87 at output. This is the expected pattern—upper layers should attend to increasingly specific context. For N'Ko, attention entropy barely decreases: from 4.87 to 4.14. The attention mechanism has nothing to focus on.

11.2 Head Specialization

In well-functioning multilingual models, individual attention heads develop specializations: syntactic heads that attend to grammatical dependencies, positional heads that track relative token positions, and semantic heads that attend to semantically related tokens (Clark et al., 2019).

We measure head specialization by computing the standard deviation of each head's attention entropy across the 100 examples. High standard deviation indicates a head that behaves differently for different inputs (specialization); low standard deviation indicates a head that behaves identically regardless of input (no specialization).

For English inputs, the mean attention entropy standard deviation across all heads in layers 30–56 is 0.83. For N'Ko inputs, it is 0.21. This $4\times$ reduction in head-level variability confirms that N'Ko inputs do not trigger the specialized attention patterns that English inputs produce. The heads are not malfunctioning—they simply have nothing to specialize on.

11.3 Cross-Attention Between N'Ko Characters

We examine whether attention patterns between adjacent N'Ko characters show any evidence of the model learning character composition (combining individual character tokens into subword or word-level representations).

For a typical English word tokenized as a single subword token, no composition is needed. For a N'Ko word tokenized as 4–8 individual character tokens, the model must learn to combine these tokens into a coherent representation through attention.

We compute the mean attention weight between adjacent N'Ko character tokens belonging to the same word (as determined by space tokenization) across layers 0–10.

Table 14 shows that N'Ko character tokens attend to adjacent characters *less* than English subword tokens attend to adjacent subwords (0.047 vs. 0.082). This is the opposite of what we would need for successful character composition: the model should attend *more* to adjacent characters within a word to build up word-level representations from character-level inputs. Instead, it attends less, treating each character as largely independent—consistent with the dead circuit hypothesis.

12 Reproducibility and Data Release

12.1 Computational Requirements

All experiments can be reproduced on the following hardware:

- **Activation profiling**: Any GPU with ≥40GB VRAM (A100 40GB, A100 80GB, A6000). We use NF4 quantization which requires approximately 38GB for Qwen2-72B. Profiling 100 sentence pairs takes approximately 2 hours. Cost: approximately $1.72 at Vast.ai rates.

- **LoRA fine-tuning**: Apple M4 with 16GB unified memory (or any GPU with ≥8GB VRAM). We use MLX v0.29 on macOS; PyTorch + bitsandbytes achieves equivalent results on CUDA GPUs. Total training time: 185 minutes across three stages.

- **Circuit duplication**: Same GPU as profiling. 55 configurations at approximately 3 minutes each. Total: approximately 2.75 hours.

12.2 Data Availability

All data used in this study is either publicly available or released with this paper:

- **N'Ko Wikipedia**: 1,693 articles, 3.7M characters. Available via Wikimedia dumps.

- **Parallel sentence pairs**: 100 English-N'Ko pairs used for profiling. Released in our repository.

- **Activation profiles**: Per-layer metric values for all 100 pairs × 81 layers × 4 metrics. Released as CSV files.

- **Circuit duplication scores**: All 55 configurations with English and N'Ko scores. Released as CSV.

- **Fine-tuned model weights**: LoRA adapter weights for all three stages. Released on HuggingFace.

- **BPE tokenizer**: 512-merge N'Ko tokenizer trained on 62,035 word occurrences. Released in repository.

12.3 Evaluation Protocol

Our translation tax metric (L2 norm ratio) is computed as follows:

1. For each sentence pair (s_e, s_n) where s_e is English and s_n is N'Ko:

2. Tokenize independently using Qwen2's tokenizer.

3. Forward pass through the model, storing hidden states at every layer.

4. Compute per-token L2 norm at each layer, then average across tokens in each sentence.

5. The translation tax at layer l is $\|h_l^e\|_2 / \|h_l^n\|_2$.

6. Average across all 100 sentence pairs.

The per-token averaging in step 4 ensures that the metric is not biased by sequence length differences. The sentence-level averaging in step 6 provides a stable estimate (bootstrap 95% CI is within ± 0.08 of the reported means at all layers).

13 Future Work

13.1 Vocabulary Injection

The most direct intervention for script invisibility is vocabulary injection: adding BPE-trained subword tokens for the target script to the model's tokenizer before pre-training. This approach has been validated for Arabic (Antoun et al., 2020) and Indic languages (Kakwani et al., 2020) but has not been systematically studied for script-invisible languages.

We propose a controlled experiment: take an open-source LLM (e.g., Llama 3.1 or Qwen2.5), extend its tokenizer with 512 N'Ko BPE merges, re-initialize the corresponding embedding rows, and perform continued pre-training on N'Ko Wikipedia. The prediction from our analysis is that vocabulary injection followed by modest CPT (10,000–20,000 examples) will produce lower translation taxes than the LoRA approach alone, because the model will have proper subword representations from the start.

13.2 Multi-Script Profiling Study

Our three-script comparison (English, Arabic, N'Ko) establishes the methodology but is insufficient for strong causal claims about the

vocabulary-to-representation relationship. A comprehensive study across 15–20 scripts with vocabulary density ρ_s ranging from 0.0 (Adlam) to 609 (Latin) would enable regression analysis relating ρ_s to translation tax, circuit duplication scores, and fine-tuning data requirements. Such a study would produce a practical guide for model developers: given a target ρ_s, how much additional training data is needed to achieve translation tax $< 1.5\times$?

Candidate scripts for this study span three categories:

- **Well-represented** ($\rho_s > 10$): Latin, CJK, Arabic, Cyrillic—providing the high-ρ_s reference points.
- **Moderately represented** ($\rho_s \in [1, 10]$): Devanagari, Thai, Tamil, Georgian—scripts with dedicated vocabulary entries but smaller than Latin or CJK.
- **Invisible** ($\rho_s < 1$): N'Ko, Adlam, Tifinagh, Vai, Osmanya, Ol Chiki, Bamum—scripts with zero or near-zero vocabulary entries.

The expected relationship is logarithmic: tax $\sim -\alpha \log(\rho_s) + \beta$, where α captures the sensitivity of representation quality to vocabulary density and β is the baseline tax for a well-represented script. Our three data points (Latin: $\rho_s = 609$, tax ≈ 1.0; Arabic: $\rho_s = 16.4$, tax ≈ 1.05; N'Ko: $\rho_s = 0.5$, tax $= 2.90$) are consistent with a logarithmic relationship but insufficient to fit it precisely.

13.3 Cross-Model Comparison

Our analysis uses Qwen2-72B-Instruct exclusively. Extending the study to other model families (Llama 3.1, Gemma 2, GPT-4, Claude) would establish whether script invisibility is a universal phenomenon or specific to particular tokenizer designs. We hypothesize that the qualitative findings (dead circuits for zero-vocabulary scripts) will be universal, but the specific translation tax values will vary based on:

- **Tokenizer training data**: Models trained on more diverse web crawls may include incidental N'Ko or Adlam text, producing non-zero (but still weak) representations.
- **Byte-level tokenization**: Models using byte-level BPE (like GPT-4) process all Unicode text without fallback, potentially producing better (but still weak) representations for invisible scripts.
- **Architecture scale**: Larger models may develop stronger character-composition abilities through their larger middle-layer capacity, partially compensating for tokenizer deficiency.

13.4 Real-Time Translation Tax Monitoring

We envision a translation tax monitoring system integrated into model training pipelines. At regular intervals during pre-training (every 10,000 steps), the system would compute translation tax estimates for a set of probe scripts (100 sentence pairs per script, 10 scripts), flagging any script where the tax exceeds $2.0\times$. This would enable model developers to intervene during training—increasing the proportion of that script's data in the training mixture—rather than discovering script invisibility after training is complete.

The computational cost of such monitoring is modest: 1,000 sentence pairs $\times$ 81 layers $\times$ 4 metrics can be computed in approximately 30 minutes on a single GPU, a negligible fraction of the total training compute for a 72B-parameter model.

14 Ethical Considerations

This work involves the computational representation of a script created by a colonized people as an act of cultural self-determination. We approach this with awareness that technical interventions—adding N'Ko to a tokenizer, fine-tuning a model—are not substitutes for the political and cultural work of script revitalization. The decision of which scripts to support in language technology should involve the communities that use those scripts, not only the engineers who build the models.

Our training data is drawn from N'Ko Wikipedia, which represents a particular register (encyclopedic) and a particular community of contributors. N'Ko is used for a wider range of purposes—religious texts, poetry, personal correspondence, commercial signage—that our training data does not capture. The model's capabilities after fine-tuning are therefore biased toward encyclopedic N'Ko.

15 Conclusion

We have performed the first per-layer activation profiling study of a large language model processing N'Ko script, establishing quantitative baselines for script invisibility: a $2.90\times$ translation

tax, 0/55 productive circuit duplication configurations, 85.8% kurtosis deficit at the output layer, and 150% embedding sparsity inflation.

Three-zone failure analysis reveals structurally distinct collapse modes: comprehension failure at the embedding layer (sparse, weak representations from 32 character-level tokens), reasoning vacuum in the middle layers (empty circuits that cannot be amplified by duplication), and incoherent output at the final layers (near-maximum entropy, flat kurtosis, near-random token prediction).

Comparison with Arabic—another RTL script that the same model processes competently—isolates vocabulary allocation as the mechanistic root cause. Arabic receives 4,200 vocabulary entries with rich subword coverage; N'Ko receives 32 single-character tokens. The difference in representation quality maps directly to this 131× vocabulary disparity.

Three-stage LoRA fine-tuning reduces the translation tax from 2.90× to 0.70× using 92,184 training examples and 185 minutes on an Apple M4, demonstrating that the model's architecture is capable—the data was the bottleneck. We document the V1/V2/V3 progression, including mode collapse in V2 and its resolution through data diversification.

The framework generalizes to at least six other scripts that we predict are invisible to current LLMs: Adlam, Tifinagh, Vai, Osmanya, Ol Chiki, and Bamum. We propose vocabulary density (ρ_s) as a diagnostic metric, translation tax spot-checking as a validation protocol, and tokenizer auditing as a mandatory step in multilingual model releases.

The circuits are dead. The cost of bringing them to life is three hours on consumer hardware. The barrier is not compute. It is the invisible filter of training data curation that systematically excludes the scripts created by the peoples who needed them most.

Code, data, and activation profiles: https://github.com/Diomandeee/nko-brain-scanner

Total compute cost: $1.72 (activation profiling, Vast.ai A100) + $0.00 (LoRA fine-tuning, Apple M4 16GB) = $1.72

References

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *LREC Workshop on Open-Source Arabic Corpora and Processing Tools*.

Loïc Barrault et al. 2023. WMT 2023 shared task: Machine translation for N'Ko. In *Proceedings of the Eighth Conference on Machine Translation (WMT)*.

Alexis Conneau et al. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of ACL 2020*.

Bonaventure F. P. Dossou, Atnafu Lambebo Tonja, et al. 2022. AfroLM: A self-active learning-based multilingual pretrained language model for 23 African languages. In *SustaiNLP Workshop at EMNLP*.

Moussa Doumbouya et al. 2021. Using radio archives for low-resource speech recognition: Towards an automatic transcription of Bambara radio broadcasts. In *Proceedings of NAACL*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR 2022*.

Divyanshu Kakwani et al. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of EMNLP*.

Alexandre Magueresse, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. In *Proceedings of the 1st Workshop on NLP for Positive Impact (ACL)*.

David Noel Ng. 2024. Revisit your shoulders: A circuit analysis of transformer layers for reasoning enhancement. arXiv preprint.

Jonas Pfeiffer et al. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of EMNLP 2020 (Demo)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of ACL 2019*.

Atnafu Lambebo Tonja et al. 2023. Natural language processing in Ethiopian languages: Current state, challenges, and opportunities. In *AfricaNLP Workshop at ACL 2023*.

Unicode Consortium. 2006. N'Ko block: U+07C0–U+07FF. *The Unicode Standard*, Version 5.0+.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP*.

Alham Fikri Aji et al. 2022. One country, 700+ languages: NLP challenges for underrepresented languages and dialects in Indonesia. In *Proceedings of ACL 2022*.

David Ifeoluwa Adelani et al. 2022. MasakhaNER 2.0: Africa-centric transfer learning for named entity recognition. In *Proceedings of EMNLP 2022*.

Pratik Joshi et al. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of ACL 2020*.

Phillip Rust et al. 2021. How good is your tokenizer? On the monolingual performance of multilingual language models. In *Proceedings of ACL 2021*.

Adama Coulibaly et al. 2025. Bayelemabaga: A Bambara-French parallel corpus for machine translation. In *Proceedings of NAACL 2025*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL 2019*.

Benjamin Müller et al. 2021. First align, then predict: Understanding the cross-lingual ability of multilingual BERT. In *Proceedings of EACL 2021*.

Edoardo Maria Ponti et al. 2019. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*, 45(3):559–601.

Anne Lauscher et al. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual transformers. In *Proceedings of EMNLP 2020*.

Zihan Wang et al. 2020. Extending multilingual BERT to low-resource languages. In *Findings of EMNLP 2020*.