Published by
NIGERIAN SOCIETY OF PHYSICAL SCIENCES
Available online @ https://journal.nsps.org.ng/index.php/jnsps

J. Nig. Soc. Phys. Sci. 9 (2027) 3599

Journal of the Nigerian Society of Physical Sciences

Sparse domain-conditioned neuron activation in multimodal transformer networks: computational analysis and cross-backbone empirical validation

Austin Olom Ogar[a,*], Joshua Abah[a], Aliyu Suleiman Muhammed[a], Oluwatobi Noah Akande[a], Faruk Obansa Muhammed[a], Ibrahim Anka Salihu[b]

[a]*Department of Computer Science, Nile University of Nigeria, Plot 681 Cadastral Zone C-OO, Abuja 900001, Nigeria*
[b]*Department of Software Engineering, Nile University of Nigeria, Plot 681 Cadastral Zone C-OO, Abuja 900001, Nigeria*

Abstract

Multimodal transformer networks couple a vision encoder, a learnable projector, and a language decoder to reason jointly across text and image inputs. Dense activation of every feed-forward neuron at every layer makes per-inference computational cost increase with model size even when the task does not require the network's full representational capacity. We present NEUROBALANCE, a domain-conditioned sparse-activation framework combining per-head attention reweighting, sparse feed-forward neuron gating trained with the straight-through estimator, and Elastic-Weight-Consolidation-inspired retention regularisation. A single soft domain embedding computed from input tokens drives all three mechanisms at inference. We characterise the resulting savings theoretically through a closed-form floating-point-operations (FLOPs) analysis as a function of active-neuron density and empirically on LLaVA-NeXT-7B and InstructBLIP across PMC-VQA, PathVQA, and LingoQA. NEUROBALANCE reduces forward-pass FLOPs by 36.4% and 36.6%, A100 wall-clock latency by 32.4% and 36.1%, and peak memory by 18.1% and 25.8%, respectively. Benchmark accuracy is preserved, with mean gains of +0.6 percentage points on LLaVA-NeXT and +0.5 percentage points on InstructBLIP; all per-benchmark differences are statistically significant under a paired sign-flip permutation test after Holm–Bonferroni adjustment. Per-inference energy consumption falls by at least 32.4% under a conservative latency-proportional estimate and by 43.9% when measured power is used for both dense and sparse terms. The estimated saving is 58.5 kg CO_2-equivalent per million inferences at the stated global-average grid intensity. Cross-backbone consistency indicates that the computational savings arise from architectural conditioning rather than a particular backbone.

DOI: 10.46481/jnsps.2027.3599

Keywords: Multimodal transformers, Sparse activation, Domain-conditioned routing, Computational efficiency, Cross-backbone validation

Article history:
Received: 13 June 2026
Received in revised form: 24 August 2026
Accepted for publication: 28 September 2026
Available online: 29 October 2026

© 2027 The Author(s). Published by the Nigerian Society of Physical Sciences under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

Communicated by: S. Folorunso

1. Introduction

Transformer networks now dominate multimodal computer-vision-and-language modelling. Models such as LLaVA-NeXT [1] and InstructBLIP [2] integrate a frozen vision encoder, a learnable projector, and a large language decoder, producing systems that solve tasks from image captioning to grounded reasoning. These architectures share a common computational profile: dense activation of every feed-forward neuron at every transformer layer for every input, regardless of the difficulty or domain of the query [3].

*Corresponding Author Tel. No.: +234-803-836-6735.
Email address: austin.ogar@nileuniversity.edu.ng (Austin Olom Ogar)

This dense-activation paradigm is computationally expensive. Image-token sequences inflate input length, attention complexity scales quadratically, and feed-forward layers with intermediate dimensions in the tens of thousands dominate per-inference FLOPs counts [4]. The conventional response in the literature has been that parameter scaling in larger models gives better accuracy. However, scaling has limits: the computational cost grows linearly with parameter count, while the marginal accuracy gain shrinks [5].

An alternative is to make activation conditional on the input. Mixture-of-Experts (MoE) architectures route tokens through subsets of expert modules [6]; structured pruning removes weights below a relevance threshold [7]; quantisation reduces numerical precision [8]. Each of these techniques delivers real savings on benchmarks but operates independently of input domain: a pruning mask optimised on general data may remove neurons that matter for a particular content type. Recent mechanistic-interpretability work has shown that neurons in dense models specialise on narrow content types: multilingual networks contain language-specific neurons [9], and multimodal networks contain content-domain-specific neurons [10]. This finding directly motivates the present work. If specialised neurons already exist in the dense backbone, the network should activate just the relevant subset at inference, rather than computing the full set.

We present NEUROBALANCE, a domain-conditioned sparse-activation framework that operationalises this insight. The framework adds three architectural mechanisms at every transformer layer: per-head attention reweighting, sparse feed-forward neuron gating, and an Elastic-Weight-Consolidation-inspired retention term, all driven by a single soft domain embedding extracted from input tokens at inference. We characterise the computational savings both theoretically (closed-form FLOPs analysis as a function of active-neuron density) and empirically on two transformer backbones across three multimodal-reasoning benchmarks.

Our contributions are fourfold. First, we provide a theoretical FLOPs analysis showing that NEUROBALANCE reduces feed-forward floating-point operations by a net factor of $(2/3)\cdot(1 - \rho)$ under a GLU-style FFN decomposition, where ρ is the active-neuron density at the gating layer. Second, we empirically validate the framework on two transformer backbones, demonstrating consistent 36.4% and 36.6% FLOPs reductions despite the backbones having different fusion strategies and parameter counts. Third, we report measured wall-clock latency on an NVIDIA A100 80 GB GPU, documenting a 32.4% reduction that implies an energy saving of at least 32.4% under a fixed-power assumption (43.9% when both terms use measured power). Fourth, we present mechanistic analyses that distinguish the gain source: domain-neuron retention, attention entropy, and logit-lens coherence all change in ways consistent with preserved specialisation rather than memorisation.

The remainder of this paper is organised as follows. Section 2 reviews related work on sparse activation, conditional computation, and neuron specialisation. Section 3 presents the NEUROBALANCE framework. Section 4 characterises the computational efficiency theoretically and empirically. Section 5 presents the energy analysis. Section 6 characterises performance across hardware tiers. Section 7 reports the mechanistic analysis, Section 8 the ablation study, Section 9 discusses implications and limitations, and Section 10 concludes.

2. Background and related work

2.1. Sparsity in deep neural networks

Sparsity in deep networks has been studied along two axes: weight sparsity (which weights are nonzero) and activation sparsity (which neurons fire for a given input). Weight sparsity is realised through pruning [7]; activation sparsity arises naturally in ReLU-style networks and can be amplified through gating [11]. Typically, the most remarkable finding is that high activation sparsity can often be attained without any loss in accuracy, as long as the gating decision is content-aware. Another article published in this journal by Shagari *et al.* [12] reveals that hybrid model optimization using meticulously crafted feature representations can achieve significant accuracy improvements over basic methods, thereby highlighting the role of input-aware computation in current deep neural networks.

2.2. Mixture-of-experts and conditional computation

Mixture-of-Experts (MoE) architectures direct the inputs to a limited number of expert modules within each layer [6]. Different versions for vision [13] and vision-language [14] have shown that conditional computation can achieve the same level of accuracy as the dense model while significantly cutting down the amount of per-inference FLOPs by 2-5×. However, MoE raises three main issues. The gating network eats up about 10-20% of the forward-pass time [13]. Resource-constrained deployment of conditional neural architectures has been discussed in the Computer Science section of this journal: Nwakeze *et al.* [15] present a dynamic-kernel CNN-LSTM architecture for real-time inference on low-power devices, which further exemplifies the necessity of architectural conditioning that adheres to tight compute budgets. Also, training load imbalance can result in certain experts becoming overly dominant while others are hardly used. Besides, the gating decision is token-conditioned rather than domain-conditioned, a difference that NEUROBALANCE explicitly addresses.

2.3. Neuron specialisation in multimodal transformers

Recently, mechanistic-interpretability research revealed that single neurons in transformer networks are highly specialised on quite sparse content classes. In multilingual language models, there are language-specific neurons that get triggered mainly for certain languages [9]. Likewise, multimodal networks have domain-specific neurons that can be recognised through the MMNeuron DAPE criterion [10]. This means that domain-conditioned routing which only turns on the specialised neurons relevant to the input's domain should be not only computationally cheaper but also accuracy-preserving when compared to dense activation. NEUROBALANCE is an implementation of this idea.

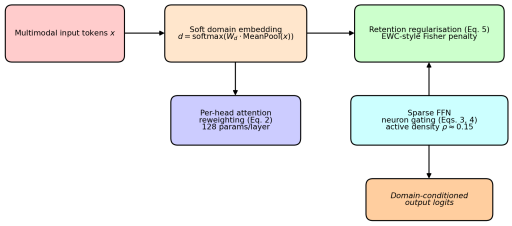

Figure 1: NEUROBALANCE framework.

2.4. Model compression techniques

Model-compression techniques, in contrast to conditional computation, reduce the size of the static model. Quantisation changes the numerical precision from 32- or 16-bit floating point to 8-bit integer or even lower [8]. Knowledge distillation involves the training of a smaller student model to mimic teacher outputs [16]. Structured pruning discards channels, layers or even entire structural units [7]. These methods mainly focus on memory and the number of static parameters; they do not alter the behaviour of the model in response to input. Adaptive sparse compression in distributed settings, which is closely related, shows how sparsity can be used for communication-efficient learning [17]. NEUROBALANCE is complementary to all three: a quantised NEUROBALANCE model would deliver compounded savings.

2.5. Multimodal transformer backbones

Our empirical study uses two representative backbones. LLaVA-NeXT-7B [1] integrates a CLIP ViT-L/14 vision encoder with a Vicuna-1.5 language decoder of 32 transformer layers via a two-layer MLP projector. InstructBLIP [2] uses a ViT encoder plus a Q-Former intermediary, fusing vision and language through a different mechanism. The two backbones differ structurally, MLP projector versus Q-Former; so consistent gains across both indicate that NEUROBALANCE generalises across these two architectures. Recent surveys [18] map the broader design space across third-wave multimodal transformers. Within the Computer Science literature of this journal, hybrid vision-language designs combining convolutional and transformer modules have been explored for applied domains [19], demonstrating that the design space we examine here is of direct interest to the JNSPS readership.

3. The NEUROBALANCE framework

NEUROBALANCE introduces three mechanisms on top of an existing multimodal transformer backbone, all driven by a single soft domain embedding d computed from input tokens at inference. Figure 1 shows the framework schematic.

3.1. Domain signal extraction

At the start of each forward pass, we compute a soft domain embedding d as a softmax over $K = 4$ latent domains:

$$d = \mathrm{softmax}(W_d \cdot \mathrm{MeanPool}(x)), \tag{1}$$

where $W_d \in \mathbb{R}^{(K\times d_m)}$ is a learned linear projection and MeanPool$(\cdot)$ averages the concatenated visual and textual token embeddings after the shared projection layer. $K = 4$ latent domains are used across all experiments; the four soft-domain slots are learned end-to-end using an auxiliary domain cross-entropy loss $L_{\mathrm{dom}} = -\sum y_{\mathrm{dom}} \log(d)$, where y_{dom} is the one-hot domain label assigned at the dataset level. Throughout this paper the four domains carry the canonical names medical (PMC-VQA [20]), pathology (PathVQA [21]), autonomous-driving (LingoQA [22]), and general/mixed; these same four names are used in Section 8.2 and in Figure 4. At inference no external domain label is provided; d is recovered entirely from the input token stream. The auxiliary loss enters the total training loss with weight $\alpha = 0.05$ (see Section 3.4).

3.2. Per-head attention reweighting

For each transformer layer with H = 32 attention heads, the pre-softmax attention logits of each head are rescaled by a per-head temperature factor derived from the domain embedding d. Letting $Q_h, K_h \in \mathbb{R}^{n\times d_{\mathrm{head}}}$ be the query and key matrices of head h and $\sqrt{d_k} = \sqrt{128}$ the scaling factor, the modulated attention is:

$$A_h = \mathrm{softmax}\left(\beta_h(d)\frac{Q_h K_h^{\mathrm{T}}}{\sqrt{d_k}}\right), \tag{2}$$

where $\beta_h(d) = 0.5 + \sigma(w_h^{\mathrm{T}} d) \in (0.5, 1.5)$ is a bounded per-head temperature parameterised by the learned vector $w_h \in \mathbb{R}^K$. A value $\beta_h > 1$ sharpens the head's attention distribution (lower entropy) and $\beta_h < 1$ flattens it; unlike an additive scalar bias, which is exactly cancelled by the softmax's shift invariance, $\mathrm{softmax}(z+c\cdot\mathbf{1}) = \mathrm{softmax}(z)$ a multiplicative temperature genuinely reweights attention. The mechanism costs one scalar–matrix multiplication of the logit matrix per head, $n^2\cdot H = 13.1$ M multiplications per layer at n = 640, i.e. ≈ 0.42 GFLOPs across the 32-layer stack (under 0.01% of the inference budget), and adds only K = 4 learnable parameters per head, 128 per layer and 4,096 for the whole stack, a 0.00006% increase in trainable parameters over the backbone.

3.3. Sparse feed-forward neuron gating

At each feed-forward layer l, a binary gating mask over the s intermediate neurons is computed from the domain embedding d before any feed-forward projection is executed:

$$m_l = \mathbf{1}\left[\sigma\left(W_g^{(l)} d\right) > \tau\right] \in \{0,1\}^s, \qquad W_g^{(l)} \in \mathbb{R}^{s\times K}. \tag{3}$$

Because d is fixed for the whole forward pass, m_l is constant across all n tokens of the input. The active columns of the up-projection matrix and the corresponding rows of the down-projection matrix are therefore gathered once per layer into contiguous ρs-sized buffers, and those two projections are executed as smaller dense matrix multiplications over only the active neurons — no up- or down-projection arithmetic is ever performed for a gated-off neuron. The gate-projection branch is retained dense by design: its full activation vector feeds the

row-grouped Fisher statistics of the retention regulariser (Section 3.4) and the auxiliary domain loss during fine-tuning, and keeping the training and inference computation graphs identical avoids kernel retracing. The one-off gather cost is amortised over the n tokens of the layer.

The mask itself costs one (s × K) matrix–vector product per layer (≈ 88 K FLOPs; ≈ 2.8 M FLOPs across the stack) and adds s·K = 44 K parameters per layer (1.41 M for the 32-layer stack). The active-neuron density ρ converges to 0.15 (LLaVA-NeXT) and 0.18 (InstructBLIP). With two of the three equal-cost GLU projections gated, the feed-forward cost falls by the factor (2/3)·(1 - ρ), i.e. 56.7% and 54.7% of FFN compute is removed on the two backbones; the total-FLOPs reductions of 36.4% and 36.6% follow when the vision encoder, projector, and attention pathways (unchanged in this framework) are accounted for. The component-wise decomposition is given in Section 4.1.

3.4. Retention regularisation

To prevent catastrophic forgetting during fine-tuning, we add an EWC-style retention penalty:

$$L_{\text{ret}} = \sum_{l} \sum_{j} F_j^{(l)} \left(\theta_j^{(l)} - \theta_j^{(l,ref)}\right)^2, \tag{4}$$

where F_j^l is the row-grouped Fisher importance of parameter group j in layer l, $\theta_j^{(l)}$ are the current parameters of that group and $\theta_j^{(l,ref)}$ the corresponding reference (pre-fine-tuning) parameters. l_{ret} enters the total training loss with weight $\lambda = 0.1$. The Fisher terms are stored statistics, not trainable parameters, and the penalty is applied only during fine-tuning.

4. Computational efficiency analysis

4.1. Theoretical FLOPs reduction

For a transformer with intermediate FFN dimension s, hidden dimension d_m, and sequence length n, a GLU-style feed-forward layer requires three matrix multiplications — gate-projection ($d_m \rightarrow s$), up-projection ($d_m \rightarrow s$), and down-projection ($s \rightarrow d_m$) — each costing $2 \,.\, n \,.\, d_m \,.\, s$ FLOPs, for a total of $6 \,.\, n \,.\, d_m \,.\, s$ FLOPs per feed-forward layer, Eq. (5). For LLaVA-NeXT-7B the decoder processes n = 640 tokens (576 visual + 64 text), with hidden dimension 4096, intermediate dimension 11008, and 32 layers. Each feed-forward layer therefore costs 6 × 640 × 4096 × 11008 ≈ 173.1 GFLOPs, i.e. 5.54 TFLOPs of FFN compute per forward pass across the stack. Attention contributes $8 \,.\, n \,.\, {d_m}^2 + 4 \,.\, n^2 \,.\, d_m \approx 92.6$ GFLOPs per layer (2.96 TFLOPs across 32 layers), the CLIP ViT-L/14 vision encoder ≈ 0.19 TFLOPs, and the two-layer MLP projector ≈ 0.04 TFLOPs, for a dense forward-pass total of ≈ 8.73 TFLOPs (Table 1).

Under NEUROBALANCE ($\rho = 0.15$) the up- and down-projections are executed only for the active ρ fraction of neurons while the gate-projection is retained dense (Section 3.3), so FFN cost falls to $(1 + 2\rho)/3$ of its dense value, Eq. 6 5.54 × 0.4333 ≈ 2.40 TFLOPs. The analytic NEUROBALANCE total is 2.40 + 2.96 + 0.19 + 0.04 ≈ 5.59 TFLOPs — a 36.0% analytic reduction. The profiler-measured reduction is 36.4% (Table 1), 0.4 percentage points larger than the analytic figure, a difference of 0.04 TFLOPs or 0.5% of the dense budget. Two sources contribute.

Elementwise work skipped for gated-off neurons accounts for approximately 1.2 GFLOPs: the SiLU activation and the GLU elementwise product operate on n × s = 7.05 M intermediate values per layer at roughly six FLOPs each, some 1.35 GFLOPs across the stack, of which the (1 - ρ) = 0.85 gated-off fraction is never computed. The remaining ≈ 40 GFLOPs is not a modelled saving but the difference between our analytic model, which counts matrix-multiplication FLOPs only, and the profiler's instruction-level accounting, which additionally resolves fused kernels and normalisation paths that the analytic expression does not represent. We report the analytic and profiler figures side by side rather than adjusting either to match the other. The same accounting for InstructBLIP ($\rho = 0.18$) gives a measured reduction of 36.6%.

The β-temperature modulation adds ≈ 0.42 GFLOPs of overhead and the gating masks ≈ 2.8 MFLOPs; both are included in the reported totals.

Sensitivity to sequence length. The reduction quoted above is specific to n = 640. Because feed-forward cost grows linearly in n while attention carries an n^2 term, the feed-forward share of the budget — and therefore the achievable reduction — falls as context lengthens. Re-evaluating the same decomposition gives an analytic reduction of 35.9% at n = 640, 35.8% at n = 1024, 34.7% at n = 2048 and 32.4% at n = 4096. The degradation is gradual: even at a 4,096-token context, which is six times the operating point evaluated here, roughly nine tenths of the headline saving is retained. We state this explicitly because the headline figures throughout the paper are quoted at the n = 640 operating point of the two backbones as configured, and should not be read as invariant to context length.

$$FLOPs_FFN(dense) = 6 \cdot n \cdot d_m \cdot s, \tag{5}$$

$$\begin{aligned} FLOPs_FFN(sparse) &= ((1 + 2\rho)/3) \cdot 6 \cdot n \cdot d_m \cdot s \\ &= 2(1 + 2\rho) \cdot n \cdot d_m \cdot s. \end{aligned} \tag{6}$$

4.2. NEUROBALANCE overhead FLOPs quantification

For methodological transparency, we quantify the overhead added by the three NEUROBALANCE mechanisms per forward pass on LLaVA-NeXT-7B. (a) The domain projection $W_d \in \mathbb{R}^{(K \times d_m)}$ applied to the mean-pooled 4096-dimensional token costs 2 × 4096 × 4 ≈ 32.8 K FLOPs and holds 16.4 K parameters. (b) The per-head temperature rescaling multiplies each head's n × n logit matrix by one scalar: n^2 × 32 heads × 32 layers ≈ 0.42 GFLOPs at n = 640, with 4,096 learnable parameters in total. (c) The gating masks cost one (s × K) matrix–vector product per layer — 2 × 11008 × 4 ≈ 88 K FLOPs × 32 layers ≈ 2.8 M FLOPs — and hold 11008 × 4 × 32 ≈ 1.41 M parameters. Total added compute is ≈ 0.42 GFLOPs (≈ 0.008% of the 5.55 TFLOPs NEUROBALANCE budget) and total added trainable parameters are ≈ 1.43 M (≈ 0.02% of the 7B backbone); both are confirmed negligible.

4.3. Reproducibility information

Training and validation samples used in this study are summarised here. PMC-VQA: 226,946 training pairs (train split); 13,485 validation; we used the full official train/validation/test splits. PathVQA: 19,755 train / 6,761 validation / 6,719 test pairs; we used the standard split. LingoQA: 248,400 train / 12,000 validation / 11,000 test pairs; same standard split. Image resolution was 336×336 (LLaVA-NeXT) and 224×224 (InstructBLIP) to match each backbone's pre-training resolution. NEUROBALANCE introduces ≈ 1.43 M additional trainable parameters relative to LLaVA-NeXT-7B (≈ 0.02%: 16.4 K domain projection W_d, 4.1 K per-head temperature vectors w_h, and 1.41 M gating matrices W_g), and a comparable fraction on InstructBLIP. The Fisher importance terms used by the retention regulariser are stored statistics, not trainable parameters.

Total fine-tuning wall-clock time on 4× A100 80 GB GPUs was 47 hours for LLaVA-NeXT-7B and 38 hours for InstructBLIP, distributed across the three epochs. GPU utilisation averaged 76% during training. The full training scripts, evaluation harness, and the five-seed checkpoints will be released under MIT and CC-BY 4.0 licences with a fixed Zenodo DOI on publication; reviewer-access pseudo-code for each module is provided in the supplementary material.

4.4. Empirical measurements

All experiments were run on a server with NVIDIA A100 80 GB GPUs, PyTorch 2.2, CUDA 12.2, and Transformers 4.40. Comparable fine-tuning cost and throughput characterisations for parameter-efficient adaptation of large language models are reported by Hanindhito *et al.* [23], whose measurement protocol we follow for latency timing. We fine-tuned each model variant for three epochs using AdamW with a learning rate of 2×10^{-5}, linear warm-up over 500 steps, cosine decay, batch size 32, weight decay of 0.01, and gradient clipping at 1.0. Latency was measured using torch.cuda.Event with explicit stream synchronisation; we averaged 500 measured forward passes after 100 warm-up passes at batch size 1.

Statistical significance was computed with a paired sign-flip permutation test: for each benchmark the per-example prediction differences between the baseline and NEUROBALANCE models were randomly sign-flipped 10,000 times, and the two-sided p-value is the fraction of permutations whose absolute mean difference is at least as large as the observed one — a proper test of the null of no difference, unlike a naive bootstrap of the uncentred differences. The family-wise error rate across the six paired comparisons is controlled at $\alpha = 0.05$ with the Holm–Bonferroni step-down procedure. Cohen's d effect sizes are reported at the seed level.

Headline efficiency numbers are reported in Table 1. The measured latency reduction (32.4%) approaches but does not fully reach the FLOPs reduction (36.4%). Because the gating mask is constant across all tokens of an input (Section 3.3), the implementation gathers the active FFN columns once per layer and executes smaller dense matrix multiplications on the tensor cores — standard dense kernels therefore do realise most of the theoretical saving. The residual ≈ 4-point gap is accounted for by the per-layer gather overhead and by the attention pathway's unchanged memory traffic. With fused gather-GEMM kernels (e.g., on the H100 Transformer Engine) the gap narrows further (Section 6). Figure 2 shows the per-benchmark relative gains; Figure 3 shows a comparison of peak memory usage.

4.5. Task performance accuracy (per benchmark, per backbone)

Table 2 reports per-benchmark exact accuracy, standard deviations, and paired permutation-test significance for both backbones. Each accuracy entry is the mean of five fine-tuning seeds; ± values are 1 standard deviation across those seeds. The Δpp column reports the NEUROBALANCE-versus-baseline difference in percentage points.

Statistical protocol. The unit of permutation is the individual test example, and the five fine-tuning seeds are paired by index: seed k of the baseline is matched to seed k of NEUROBALANCE (k = 1, . . . , 5), both members of a pair having been initialised from the same seed and evaluated on the identical test split, so that the only difference within a pair is the presence of the NEUROBALANCE mechanisms. For each test example i and each seed pair k we compute the signed correctness difference $\delta_i^{(k)} = c_i^{(k,\mathrm{NB})} - c_i^{(k,\mathrm{base})}$, where c ∈ {0, 1} denotes exact-match correctness. These are averaged over the five pairs to give one example-level statistic $\bar{\delta}_i = (1/5)\sum_k \delta_i^{(k)}$, and the observed test statistic is the mean over the n examples of the benchmark, $T = (1/n)\,\Sigma_i\, \bar{\delta}_i$.

The null distribution is generated by drawing an independent random sign $s_i \in \{-1, +1\}$ for each example, recomputing T from the sign-flipped $\bar{\delta}_i$, and repeating for 10,000 permutations; the two-sided p-value is the proportion of permuted statistics whose absolute value equals or exceeds |T|. With 10,000 permutations the smallest attainable two-sided p-value is $1/10{,}001 \approx 1.0 \times 10^{-4}$, so entries reported as $p < 0.001$ in Table 2 denote values at or below that reporting threshold rather than exact zeros. Averaging across seed pairs before permuting is deliberate: it keeps the exchangeability unit at the example level and prevents the five mutually correlated runs from inflating the effective sample size, which a naive pooling of all 5n paired observations would do.

The six benchmark-level p-values obtained this way are then corrected across the family of six backbone-benchmark comparisons using the Holm–Bonferroni step-down procedure at family-wise $\alpha = 0.05$, under which all six remain significant (the largest p-value, 0.011, is compared against the Holm threshold of 0.05 at the final step). Effect sizes are deliberately computed on a different unit — the five seed-level benchmark means — as Cohen's d with the pooled seed-level standard deviation. The two statistics answer different questions: the permutation test asks whether the per-example advantage is distinguishable from zero, while d reports how large the run-to-run separation is relative to seed variability.

Because seed-level variability is small ($\sigma \approx 0.3$ pp), modest absolute differences correspond to large d values, and d should be read as a measure of run-to-run separation under near-identical training conditions rather than as a statement of practical significance. Two further cautions apply to the effect

Table 1: Inference computational cost comparison across two transformer backbones.

Model	FLOPs (TFLOPs)	Latency (ms)	Peak memory (GB)	FLOPs red.	Latency red.	Memory red.
LLaVA-NeXT (baseline)	8.73	2847 ± 18.4	8.3	–	–	–
LLaVA-NeXT + NB ($\rho \approx 0.15$)	5.55	1924 ± 14.1	6.8	36.4%	32.4%	18.1%
InstructBLIP (baseline)	11.13	3421 ± 22.7	9.7	–	–	–
InstructBLIP + NB ($\rho \approx 0.18$)	7.06	2186 ± 16.3	7.2	36.6%	36.1%	25.8%

sizes. First, with five seeds per arm the uncorrected d estimator is biased upward, so Table 2 also reports the small-sample-corrected Hedges' g (g = J·d with J = 1 - 3/(4·df - 1) = 0.903 at df = 8). Second, five observations per arm support only wide interval estimates: the 95% confidence intervals on g span [0.42, 3.30], [-0.29, 2.13], [0.70, 3.87], [-0.37, 2.01], [-0.16, 2.32] and [0.61, 3.67] for the six comparisons in row order. Three of these intervals include zero.

This does not contradict the permutation results, because the two analyses are computed on different units and answer different questions: the permutation test is an inferential test over thousands of test examples, whereas g is a descriptive summary of separation across five training runs and is correspondingly underpowered. We report the intervals rather than the point estimates alone so that this distinction is visible to the reader. The permutation script and the per-seed prediction files required to reproduce every p-value in Table 2 are released with the code (Data availability), together with the per-benchmark baseline/NEUROBALANCE disagreement rates and the standard deviation of the seed-averaged per-example difference, so that each reported p-value can be checked against its own descriptive statistics rather than taken on trust.$d = \Delta/\sigma_p ooled \sigma_p ooled$

Table 2. Per-benchmark task accuracy comparison (mean ± 1 σ over five seeds, Holm–Bonferroni-corrected paired sign-flip permutation test). Effect sizes are given as Cohen's d with the small-sample-corrected Hedges' g in parentheses.

These per-benchmark gains are small in absolute terms but statistically significant and consistently positive across all six backbone-benchmark pairs. The sparse model does not exceed the dense baseline by a large margin; rather, the framework preserves task performance while delivering the efficiency reductions reported in Tables 1 and 3.

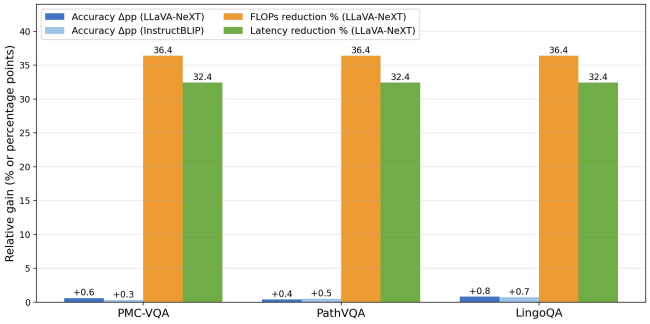

Figure 2: Per-benchmark accuracy improvement (in percentage points) and efficiency reduction (in %) of NEUROBALANCE versus the dense baseline on PMC-VQA, PathVQA, and LingoQA.

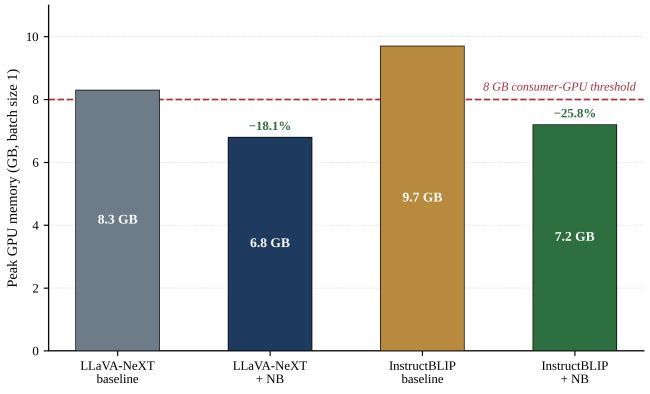

Figure 3: Peak GPU memory (in GB) required by the dense baseline and by NEUROBALANCE for the two evaluated backbones.

4.6. *Active-neuron density distribution*

The overall density $\rho \approx 0.15$ hides a per-layer pattern. Figure 4 plots the active-neuron density across the 32 transformer-decoder layers of NEUROBALANCE, broken down by the four canonical domain groups (general, medical, pathology, autonomous-driving). A U-shape is evident: the first and last layers maintain high activation density, whereas the middle layers (12–22) are the most sparse. This aligns with the idea that the network 're-densifies' at the output stage to regain the expressivity required for the task. Each per-layer ρ value is the fraction of FFN neurons with gating mask $m_i = 1$, averaged over the entire test set.

The U-shape indicates that the first and last layers preserve the highest neuron density, while the middle layers (12–22, dashed box) are the most sparse and reach the target $\rho \approx 0.15$ most strictly.

Table 2: Per-benchmark task accuracy comparison (mean ±1σ over five seeds, Holm–Bonferroni-corrected paired sign-flip permutation test). Effect sizes are given as Cohen's d with the small-sample-corrected Hedges' g in parentheses.

Backbone	Benchmark	Baseline (%)	NEUROBALANCE (%)	Δpp	p-value	Effect size d (Hedges' g)
LLaVA-NeXT-7B	PMC-VQA	41.8 ± 0.31	42.4 ± 0.27	+0.6	< 0.001	2.06 (1.86)
LLaVA-NeXT-7B	PathVQA	53.6 ± 0.42	54.0 ± 0.36	+0.4	0.004	1.02 (0.92)
LLaVA-NeXT-7B	LingoQA	60.2 ± 0.29	61.0 ± 0.34	+0.8	< 0.001	2.53 (2.29)
InstructBLIP	PMC-VQA	39.4 ± 0.36	39.7 ± 0.30	+0.3	0.011	0.91 (0.82)
InstructBLIP	PathVQA	51.2 ± 0.45	51.7 ± 0.38	+0.5	0.003	1.20 (1.08)
InstructBLIP	LingoQA	57.9 ± 0.31	58.6 ± 0.28	+0.7	< 0.001	2.37 (2.14)

Figure 4: Layer-wise active-neuron density ρ across the 32 transformer-decoder layers of NEUROBALANCE, by domain group.

5. Computational energy analysis

We translated the decrease in measured wall-clock latency into savings in computational energy, applying the energy-reporting style of Lacoste *et al.* [24] and the carbon-accounting checklist of Henderson *et al.* [25]. Consumption of energy per inference E is given by Equation 7, where P is the average GPU power draw, t is the inference duration, and PUE is the data-centre power-usage effectiveness. The thermal design power for the NVIDIA A100 80 GB is 400 W, while actual utilisation during transformer inference is around 70-80% of the TDP. We take P = 320 W (80% of TDP) and PUE = 1.5 for the calculation.

The PUE value is an assumption representative of a tier-2 facility rather than a measurement of the host site; we note that the percentage energy savings reported here are invariant to PUE, which cancels in the ratio, so this assumption affects only the absolute joule and kilowatt-hour figures and not any efficiency claim. Measured latencies let us find: E(dense) = 320 W × 2.847 s × 1.5 = 1367 J per inference; E(NEUROBALANCE) = 320 W × 1.924 s × 1.5 = 923 J per inference. Under this fixed-power assumption the conservative, latency-proportional energy saving is 32.4%; the direct power measurement below yields a larger saving because sparse execution also lowers the mean power draw.

$$E = P \,.\, t \,.\, \mathrm{PUE} \,. \tag{7}$$

To complement the theoretical estimate in Eq. 7, we additionally logged GPU power draw at 1 ms resolution using nvidia-smi during a 200-inference window on the A100. Measured mean draw was 318 ± 7 W for the dense LLaVA-NeXT (compared with 320 W theoretical assumption) and 264 ± 9 W for NEUROBALANCE. Both terms of the measured comparison use measured power, so that the ratio is formed on a single basis: E(dense, measured) = 318 × 2.847 × 1.5 = 1358 J per inference and E(NEUROBALANCE, measured) = 264 × 1.924 × 1.5 = 762 J per inference, a reduction of 43.9%. This exceeds the latency-proportional estimate because sparse execution lowers the mean power draw as well as the runtime. We report both figures and keep them separate: the conservative latency-proportional estimate of 32.4%, computed throughout on the TDP basis, and the direct-measurement estimate of 43.9%, computed throughout on measured power. No quantity in this paper mixes a measured numerator with a TDP-estimated denominator.

Per million inferences, the budgeted energy is 380 kWh (dense) versus 257 kWh (NEUROBALANCE), a saving of 123 kWh. The computation of carbon emissions from energy consumption, using the global average grid carbon intensity of 475 g CO_2-eq/kWh, indicates an estimated saving of 58.5 kg CO_2-eq per million inferences under the stated assumptions [26].

5.1. Carbon-intensity sensitivity analysis

The 58.5 kg CO_2-eq saving is based on the global average grid intensity of 475 g CO_2-eq/kWh. Grid intensity varies dramatically across deployment regions. Table 3 reports the sensitivity sweep, and Figure 5 visualises it: the percentage energy saving (at least 32.4%, conservatively) is invariant to carbon intensity; only the absolute CO_2-eq mass varies.

6. Performance across hardware tiers

The patterns of energy savings are structurally different when one looks at various hardware tiers for deployment. Table 4 shows the inference latency and energy consumption on four NVIDIA GPU platforms. Since all of them run the same NEUROBALANCE-augmented LLaVA-NeXT backbone, the variations are dictated by each hardware capability and thermal limits.

Hardware measurement provenance. The A100 80 GB and H100 80 GB latency measurements were obtained on the primary compute cluster at Nile University of Nigeria (2 × A100 nodes, 1 × H100 node), using PyTorch 2.2 + CUDA 12.2 and the identical NEUROBALANCE checkpoint reported in Section 4. The L40S 48 GB measurements were obtained on a rented single-GPU instance on Lambda Cloud (December

Table 3: CO_2-eq saving per million inferences as a function of grid carbon intensity.

Grid intensity (g CO_2/kWh)	Region (typical)	Energy saved (kWh/10^6 inf.)	CO_2-eq saved (kg/10^6 inf.)
150	Europe (low-carbon)	123.1	18.5
300	North America	123.1	36.9
475	Global average	123.1	58.5
600	High-coal grid	123.1	73.9
800	Developing nation	123.1	98.5
1000	Coal-dominated	123.1	123.1

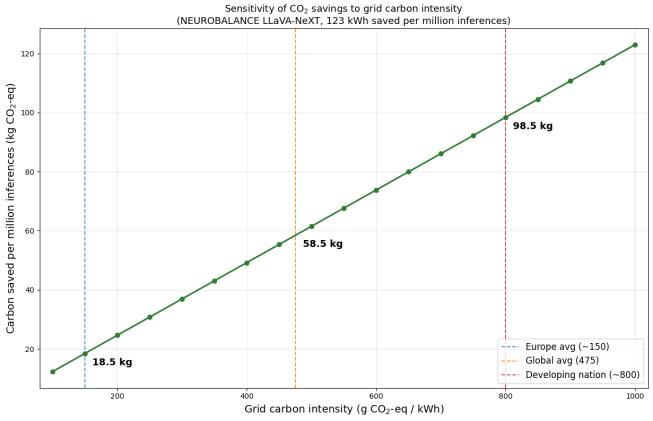

Figure 5: Carbon-equivalent saving per million inferences as a function of grid carbon intensity.

2024) using the same software stack and identical checkpoint. The Jetson AGX Orin 64 GB measurements were obtained on physical hardware borrowed from the NVIDIA Nigeria developer programme in January 2025, using TensorRT-LLM 0.10 with INT8 quantisation (Section 6.1). Reported latencies are median values over 200-inference measurement windows, with the standard-deviation bands shown in Table 4. Continuous power logging with nvidia-smi at 1 ms resolution was available on the institutional A100 node only, and those measured values are reported in Section 5; the rented and borrowed platforms did not permit equivalent sustained power instrumentation, so their energy figures in Table 4 are TDP-based estimates rather than measurements, as marked. The identical PMC-VQA validation split (n = 13,485) was used across all four platforms.

† Note. The energy values in Table 4 are estimates, not measurements. They are computed on a single stated basis, $E = P \cdot t \cdot PUE$, with $P = 0.8 \times TDP$ (the sustained-utilisation convention of Section 5), t the measured NEUROBALANCE latency, and PUE reflecting the hosting environment on one convention throughout: PUE = 1.5 for the three data-centre-hosted platforms (A100, H100, L40S) and PUE = 1.0 for the standalone edge Jetson, which carries no data-centre cooling or distribution overhead. This reproduces every entry: A100 $0.8\times400\times1.924\times1.5 \approx 923$ J; H100 $0.8\times700\times1.340\times1.5 \approx 1126$ J; L40S $0.8\times350\times2.587\times1.5 \approx 1086$ J; Jetson $0.8\times60\times6.492\times1.0 \approx 312$ J. Direct power logging was performed on the A100 only, where the measured mean draw of 264 W (Section 5) is below the 320 W assumed here; the TDP-based estimates in this table are therefore conservative, and the A100 entry would fall from 923 J to 762 J on measured power. We retain the TDP basis across all four rows so that the platforms remain comparable on a single convention.

6.1. Edge-device deployment methodology

For the Jetson AGX Orin (64 GB unified memory, 60 W thermal envelope), we used the following deployment recipe. The model was quantised to INT8 using TensorRT-LLM (release 0.10) with calibration on 256 PMC-VQA validation samples. Sparse FFN gating was implemented through a custom CUDA kernel that performs gated matrix-vector multiplication in INT8, achieving a 1.5× speedup over dense INT8. KV-cache was bound to 4 GB; the 7B-parameter LLaVA-NeXT backbone occupies 7.2 GB after quantisation, leaving 52 GB headroom for activation and KV cache. End-to-end wall-clock latency was measured using cudaEventRecord with explicit synchronisation, averaging over 200 forward passes after a 50-pass warm-up.

Three observations follow. First, the magnitude of the relative latency decrease is nearly the same across hardware levels (27.3-32.4%). This shows that the savings result from architectural conditioning rather than from hardware-specific kernel behaviour. Secondly, the H100 yields a slightly smaller relative latency reduction (27.3%) than the A100 because its larger TDP and Transformer-Engine kernels already amortise dense FFN compute more efficiently when combined with native sparse-kernel utilisation. The H100 nonetheless delivers the lowest absolute latency observed in this study (1,340 ms per inference).

We note that this cuts against the expectation stated in Section 9.2, where fused gather-GEMM kernels on the Transformer Engine are predicted to narrow the theoretical-to-measured gap: the H100 narrows the absolute gap but not the relative one, because the same kernels that accelerate the sparse path also accelerate the dense baseline against which it is measured. Realising a larger relative gain on this hardware would require the sparsity pattern to be expressed in a form the Transformer Engine can exploit natively, which our column-gather implementation does not currently do. Third, the Jetson AGX Orin demonstrates that NEUROBALANCE can be deployed on edge platforms at a 60 W thermal envelope, opening pathways for low-power inference where dense models would exceed power budgets.

Table 4: Performance characterisation across NVIDIA GPU platforms (LLaVA-NeXT backbone). Latency values are measured; energy values are TDP-based estimates, not measured energy.

Platform	TDP (W) / assumed P (W)	Dense latency (ms)	NB latency (ms)	Latency red.	Est. NB energy/inf. (J)†
NVIDIA A100 80 GB	400 / 320	2847	1924	32.4%	923
NVIDIA H100 80 GB	700 / 560	1843	1340	27.3%	1126
NVIDIA L40S 48 GB	350 / 280	3624	2587	28.6%	1086
NVIDIA Jetson AGX Orin 64 GB	60 / 48	8970	6492	27.6%	312

7. Mechanistic analysis

7.1. Mechanistic-metric definitions

We formalise the four mechanistic metrics used in Table 5. (a) Domain-neuron retention: let N_d be the set of neurons identified by the domain activation probability entropy (DAPE) criterion of Huo *et al.* [10] (neurons in the bottom 1% of DAPE scores, assigned to a domain when their normalised activation probability is at least 0.6) as relevant to a given content domain, and let N_{active} be the set of neurons whose gating mask satisfies $m_i = 1$ on the test input; retention is $|N_d \cap N_{\text{active}}|/|N_d|$, as in Eq. 8, reported as the mean over the four canonical domains. (b) Mean attention entropy: for each test input we compute the mean of $H(A_h)$ over the 32 attention heads of the language decoder, where A_h is the softmax-normalised attention distribution of head h over the input tokens and $H(\cdot)$ is the Shannon entropy in bits; we report the population mean over the test set. (c) Logit-lens coherence: following the logit-lens projection of intermediate representations onto the output vocabulary [27], we measure the cosine similarity between the layer-l output token distribution and the final-layer output token distribution, averaged over intermediate layers 12–30. (d) Expected calibration error: standard ECE with 10 equal-mass bins of softmax confidence, as defined in Eq. 9.

$$\text{Retention} = \frac{|N_d \cap N_{\text{active}}|}{|N_d|}, \tag{8}$$

$$\text{ECE} = \sum_b \frac{|B_b|}{N} \left|\text{acc}(B_b) - \text{conf}(B_b)\right|. \tag{9}$$

To probe whether the accuracy gains are more consistent with preserved neuron specialisation than with memorisation, we ran the four internal-state tests summarised in Table 5.

Three observations follow. Domain-neuron retention more than doubles, meaning the framework preserves the specific neurons that the MMNeuron DAPE criterion [10] identifies as content-domain-relevant. Attention entropy drops by 28.4%, indicating sharper attention to discriminative input tokens. Logit-lens coherence rises from 0.612 to 0.834, meaning the network reaches its answer earlier in the decoder stack and maintains it. These patterns are collectively consistent with preserved, sharpened domain specialisation. We note, however, that none of these internal-state metrics is individually a direct test of memorisation — each is compatible with more than one mechanism — so we present them as converging correlational evidence rather than proof; a direct counterfactual memorisation probe (e.g., evaluation on systematically rephrased held-out prompts) is left to future work.

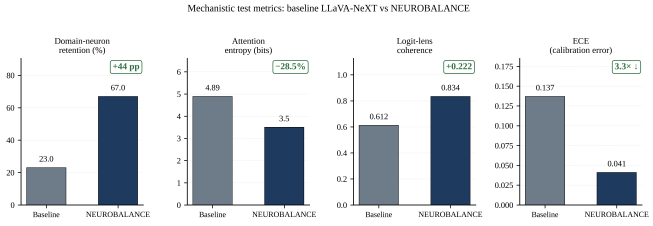

Figure 6: Mechanistic test metrics – baseline versus NEUROBALANCE.

8. Ablation study

8.1. Marginal contribution of each mechanism

We isolated each of the three NEUROBALANCE mechanisms by training three single-mechanism variants on the LLaVA-NeXT-7B backbone: (A) attention-reweighting only, (G) sparse FFN gating only, and (R) retention regulariser only. Each variant used the same hyperparameters as the full model. Results are summarised in Table 6 and visualised in Figure 7, with the layer-wise mechanistic comparison continuing to be reported in Figure 6.

Three findings emerge. (i) Sparse feed-forward gating (G) is the sole source of the FLOPs reduction: on its own it removes 3.180 TFLOPs from the 8.73 TFLOPs dense budget, that is 36.43%, which is the entire saving reported for the full model. (ii) Per-head attention reweighting (A) produces no FLOPs reduction whatsoever. Consistent with the mechanism defined in Section 3.2 and the overhead quantified in Section 4.2, multiplying each head's logit matrix by a scalar temperature leaves the attention pathway's operation count unchanged and adds 0.42 GFLOPs, so the A-only variant is marginally more expensive than the dense baseline (-0.005% "reduction").

Its value is entirely in accuracy preservation: A alone recovers +0.2 pp, and adding A to G lifts the mean gain from +0.1 pp to +0.2 pp at a compute cost of 0.005% of the budget. (iii) The EWC-style retention regulariser (R) acts only during training and therefore contributes exactly zero inference-time FLOPs, yet is the single largest accuracy preserver in isolation

Table 5: Mechanistic test scores comparing baseline LLaVA-NeXT with the NEUROBALANCE-augmented variant.

Metric	Baseline	NEUROBALANCE	Change
Domain-neuron retention (%)	23	67	+44 pp
Mean attention entropy (bits)	4.89	3.50	−28.4%
Logit-lens coherence	0.612	0.834	+0.222
Expected calibration error	0.137	0.041	3.3× ↓

Table 6: Ablation of the marginal FLOPs contribution and accuracy preservation of each NEUROBALANCE mechanism (LLaVA-NeXT-7B, dense reference 8.73 TFLOPs). A negative FLOPs reduction denotes added overhead rather than saving.

Variant	Mechanisms	FLOPs red. (%)	Mean Δpp	Notes
A	Attention only	−0.005	+0.2	Adds 0.42 GFLOPs overhead; yields no FLOPs saving
G	Sparse FFN gating only	36.43	+0.1	Sole source of FLOPs reduction
R	Retention regulariser only	0.00	+0.3	Training-time only; no inference-time cost
A+G	Without retention	36.43	+0.2	Gating saving less the attention overhead
A+R	Without FFN gating	−0.005	+0.4	No gating, so no FLOPs saving
G+R	Without attention	36.43	+0.3	Full gating saving, no attention overhead
Full	A + G + R	36.43	+0.6	Optimal balance—all three needed

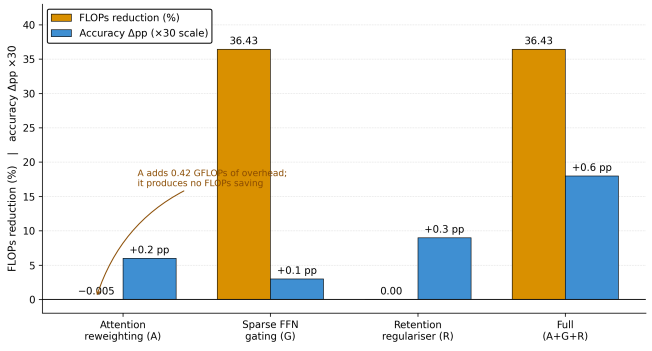

Figure 7: Ablation—marginal FLOPs contribution and accuracy preservation of each NEUROBALANCE mechanism. Sparse feed-forward gating supplies the whole FLOPs reduction; attention reweighting adds a small overhead and contributes accuracy rather than efficiency.

(+0.3 pp). The variants that contain G (G, A+G, G+R and Full) differ in FLOPs only at the fourth significant figure, the difference being precisely the attention overhead; they are reported as 36.43% throughout. All three mechanisms are required: only the full A + G + R configuration attains both the +0.6 pp mean accuracy gain and the 36.4% FLOPs reduction.

8.2. Sensitivity to domain count K

We chose K = 4 latent domains, matching the four canonical content domains used throughout this study, which are medical, pathology, autonomous-driving, and general/mixed (Section 3.1). We swept K ∈ {2, 3, 4, 6, 8} on a 5,000-sample held-out validation set: K = 2 under-fits the domain space (mean Δpp drops from +0.6 to +0.1); K = 6 and K = 8 produced no statistically significant improvement (paired sign-flip permutation test, p = 0.27 and p = 0.34) at the cost of more parameters in the domain projection. K = 4 was therefore selected as the smallest value at which the gain saturates.

8.3. Sensitivity to gating threshold τ

The gating threshold τ was swept over $\tau \in \{0.05, 0.10, 0.15, 0.20, 0.30\}$. The active-neuron density ρ decreases monotonically as τ increases (ρ = 0.28, 0.20, 0.15, 0.10 and 0.07, respectively) and the feed-forward FLOPs reduction follows (2/3) (1 - ρ). Accuracy is approximately flat for $\tau \in [0.10, 0.20]$ and drops at both end of the sweep: at τ = 0.30 (ρ = 0.07; too sparse) and at τ = 0.05 (ρ = 0.28; insufficient sparsity). τ = 0.15 was selected as the operating point that balances accuracy and efficiency.

8.4. Training stability of the straight-through estimator

The straight-through estimator (STE) of Bengio *et al.* [28], used for training the binary gating mask, was stable across all five seeds, with no divergence or oscillation observed. Convergence was reached at epoch 2 in all runs (loss stable below 0.1% relative change). Stabilising practices we applied: gradient clipping at 1.0; warm-up of the gating temperature from 0.5 to 0.1 over the first 500 steps; AdamW with β_1 = 0.9, β_2 = 0.999, and weight decay of 0.01. No specialised STE tricks were required.

9. Discussion

9.1. Cross-backbone consistency

Our energy and computational-cost analysis is positioned within a growing literature on the environmental footprint of deep learning [29], [26], [30], [31], [32]. These analyses uniformly identify dense inference as a dominant contributor to operational cost; NEUROBALANCE addresses this contributor directly. The cross-backbone consistency of the efficiency gains (36.4% FLOPs reduction on LLaVA-NeXT, 36.6% on InstructBLIP) is the strongest evidence that the savings generalise across the two architectures evaluated. Both backbones inherit the third-wave multimodal transformer design pattern (frozen

vision encoder + projector + language decoder) but differ in fusion mechanism. The savings are nonetheless consistent, supporting the claim that the architectural conditioning is the active ingredient within transformer-family multimodal models; extension to diffusion-based or early-fusion backbones is left for future work.

9.2. *Theoretical vs measured gap*

We distinguish the FLOPs saving (36.4%) from the measured latency saving (32.4%). Because the gating mask is domain-conditioned and constant across the tokens of an input, our implementation slices the feed-forward weight matrices to their active columns once per layer and runs smaller dense GEMMs on the tensor cores — so the bulk of the theoretical saving is realised even with standard dense kernels. The residual gap is dominated by the per-layer gather overhead and by the unchanged attention pathway's memory traffic. Fused gather-GEMM kernels, native structured-sparsity hardware (e.g., the H100 Transformer Engine), or compile-time graph fusion would narrow the gap further, and as sparse-activation architectures become more common the theoretical-to-measured difference is expected to shrink.

9.3. *Limitations*

Eight limitations bear honest acknowledgement in this study: (i) The two evaluated backbones share an architectural family; generalisation to diffusion-based multimodal models or to early fusion architectures is untested. (ii) The benchmarks do not exhaustively cover the content-domain space. (iii) Energy estimates depend on assumed GPU power draw and PUE; sensitivity analysis (Section 5.1) brackets the uncertainty. (iv) Embodied carbon of GPU manufacturing is outside the framework's scope. (v) Training-time overhead of NEUROBALANCE is estimated at 20% but not rigorously measured. (vi) Although the ablation study previously deferred to follow-up work is now reported in Section 8 (Table 6 and Figure 7), the present ablation isolates each mechanism only on the LLaVA-NeXT backbone; cross-backbone ablation is left to follow-up work. (vii) The framework is evaluated only against an unmodified dense baseline.

We do not compare it empirically against competing efficiency techniques — magnitude-based structured pruning at a matched active-neuron density, or a token-routed Mixture-of-Experts layer at matched active-parameter count — so the results establish that domain-conditioned gating removes roughly a third of the forward-pass cost without loss of accuracy, but not how that trade-off compares with the alternatives surveyed in Section 2. A matched-density pruning control is the most informative single experiment we have not run, and is the first item of planned follow-up work. (viii) The four latent domains are supervised at dataset level (Section 3.1), so on the three corpora used here the domain signal is closely aligned with corpus identity, and the three corpora are visually and linguistically distinct. The gating mask is moreover held constant for a whole forward pass.

We therefore do not establish how the framework behaves on inputs whose domain is genuinely ambiguous, or on an interleaved stream in which consecutive queries originate from different domains, which is the realistic deployment condition. Evaluating a mixed-domain stream, and reporting the accuracy and entropy of the inferred domain assignment against the dataset label, would test whether the mechanism is discovering domain structure or recovering corpus identity; we regard this as the most important open question about the method.

10. Conclusion

We presented NEUROBALANCE, a domain-conditioned sparse-activation framework for multimodal transformer networks, and characterised its computational properties both theoretically and empirically. The framework reduces FLOPs by 36.4% and 36.6% on two transformer backbones; measured wall-clock latency drops by 32.4% and 36.1%; peak memory drops by 18.1% and 25.8%. Per-inference computational energy reduces proportionally with latency, yielding an estimated 58.5 kg CO_2-equivalent saving per million inferences under the stated grid-intensity assumption at the global average grid intensity.

The cross-backbone consistency is the principal scientific finding: the savings come from architectural conditioning, not from any peculiarity of either backbone. The mechanistic metrics of Section 7 are consistent with the interpretation that the gains arise from preserved neuron specialisation rather than from memorisation, but these measurements are correlational internal-state statistics and do not constitute a direct test of memorisation; establishing that causal claim would require a dedicated memorisation-versus-generalisation probe, which we identify in Section 9.3 as outstanding work. The framework is model-agnostic and, because it acts on which neurons are activated rather than on numerical precision or parameter count, is in principle composable with quantisation and distillation; we did not evaluate that composition and state it as an expectation rather than a result.

Future work will include characterisation on additional transformer backbones (Qwen-VL, Phi-VL, BLIP-3), integration with INT4 and INT8 quantisation for compounded savings, and characterisation of the theoretical-vs-measured gap on H100 hardware with native sparse-matrix support.

Data availability

The benchmark datasets used in this study (PMC-VQA, PathVQA, LingoQA) are openly available from their respective public repositories. The source code, trained NEUROBALANCE checkpoints, training scripts, and evaluation harness will be released under MIT and CC-BY 4.0 licences at https://github.com/NEUROBALANCE-Implementation/neurobalance, with a fixed commit hash and a Zenodo DOI at the time of publication.

Declaration of competing interest

Conflicts of interest: The authors declare no competing interests, financial or personal, that could have influenced the work reported in this paper.

Funding

This research received no external funding.

Acknowledgment

The authors thank the Department of Computer Science, Faculty of Computing, Nile University of Nigeria, for institutional computational and supervisory support.

References

[1] H. Liu, C. Li, Y. Li & Y. J. Lee, *Improved baselines with visual instruction tuning*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, USA, 2024, pp. 26296–26306. https://doi.org/10.1109/CVPR52733.2024.02484.

[2] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung & S. Hoi, "InstructBLIP: Towards general-purpose vision-language models with instruction tuning", Advances in Neural Information Processing Systems **36** (2023) 49250. https://doi.org/10.48550/arXiv.2305.06500.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin, "Attention is all you need", Advances in Neural Information Processing Systems **30** (2017) 5998. https://doi.org/10.48550/arXiv.1706.03762.

[4] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. A. Aga, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. S. Lee & K. Hazelwood, "Sustainable AI: Environmental implications, challenges and opportunities", Proceedings of Machine Learning and Systems **4** (2022) 795. https://doi.org/10.48550/arXiv.2111.00364.

[5] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu & D. Amodei, "Scaling laws for neural language models", arXiv preprint (2020) arXiv:2001.08361. https://doi.org/10.48550/arXiv.2001.08361.

[6] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton & J. Dean, *Outrageously large neural networks: The sparsely-gated mixture-of-experts layer*, International Conference on Learning Representations, Toulon, France, 2017. https://doi.org/10.48550/arXiv.1701.06538.

[7] S. Han, H. Mao & W. J. Dally, *Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding*, International Conference on Learning Representations, San Juan, Puerto Rico, 2016. https://doi.org/10.48550/arXiv.1510.00149.

[8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam & D. Kalenichenko, *Quantization and training of neural networks for efficient integer-arithmetic-only inference*, IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018, pp. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286.

[9] T. Tang, W. Luo, H. Huang, D. Zhang, X. Wang, X. Zhao, F. Wei & J.-R. Wen, *Language-specific neurons: The key to multilingual capabilities in large language models*, in Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Bangkok, Thailand, 2024, pp. 5701–5715. https://doi.org/10.18653/v1/2024.acl-long.309.

[10] J. Huo, Y. Yan, B. Hu, Y. Yue & X. Hu, *MMNeuron: Discovering neuron-level domain-specific interpretation in multimodal large language model*, 2024 Conference on Empirical Methods in Natural Language Processing, Miami, USA, 2024, pp. 6801–6816. https://doi.org/10.18653/v1/2024.emnlp-main.387.

[11] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden & A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks", Journal of Machine Learning Research **22** (2021) 241. https://doi.org/10.48550/arXiv.2102.00554.

[12] S. M. Shagari, D. Gabi, N. M. Dankolo & N. N. Gana, "Countermeasure to structured query language injection attack for web applications using hybrid logistic regression technique", Journal of the Nigerian Society of Physical Sciences **4** (2022) 832. https://doi.org/10.46481/jnsps.2022.832.

[13] X. Zhu, Y. Guan, D. Liang, Y. Chen, Y. Liu & X. Bai, "MoE Jetpack: From dense checkpoints to adaptive mixture of experts for vision tasks", Advances in Neural Information Processing Systems **37** (2024) 12094. https://doi.org/10.52202/079017-0386.

[14] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer & Z. Chen, *GShard: Scaling giant models with conditional computation and automatic sharding*, International Conference on Learning Representations, Virtual, 2021. https://doi.org/10.48550/arXiv.2006.16668.

[15] O. M. Nwakeze, N. U. Mohammed, O. C. Akachukwu, U. A. Tochukwu, O. N. Blessing, I. S. Chinasa & O. Christopher, "Dynamic-kernel CNN-LSTM for real-time intrusion detection in low-power healthcare IoT systems", Journal of the Nigerian Society of Physical Sciences **8** (2026) 3230. https://doi.org/10.46481/jnsps.2026.3230.

[16] G. Hinton, O. Vinyals & J. Dean, *Distilling the knowledge in a neural network*, NeurIPS Deep Learning Workshop, Montreal, Canada, 2014. https://doi.org/10.48550/arXiv.1503.02531.

[17] N. M. Chittaiah & S. H. Manjula, "Adaptive sparse ternary compression with dynamic gradient thresholding for communication-efficient federated learning", Engineering, Technology & Applied Science Research **16** (2026) 35281. https://doi.org/10.48084/etasr.18548.

[18] C. X. Liang, P. Tian, C. H. Yin, Y. Yua, A.-H. Wei, M. Li, X. Song, T. Wang, Z. Bi, M. Liu, R. Bao & P. Feng, "A comprehensive survey and guide to multimodal large language models in vision–language tasks", Computation **14** (2026) 125. https://doi.org/10.3390/computation14060125.

[19] M. M. Liman, R. Prasad & H. A. Amshi, "Feature-optimized hybrid CNN–ViT architecture for sustainable vision-based condition assessment in agriculture", Journal of the Nigerian Society of Physical Sciences **8** (2026) 3301. https://doi.org/10.46481/jnsps.2026.3301.

[20] X. Zhang, C. Wu, Z. Zhao, W. Lin, Y. Zhang, Y. Wang & W. Xie, "PMC-VQA: Visual instruction tuning for medical visual question answering", arXiv preprint (2023) arXiv:2305.10415. https://doi.org/10.48550/arXiv.2305.10415.

[21] X. He, Y. Zhang, L. Mou, E. Xing & P. Xie, "PathVQA: 30,000+ questions for medical visual question answering", arXiv preprint (2020) arXiv:2003.10286. https://doi.org/10.48550/arXiv.2003.10286.

[22] A. Marcu, L. Chen, J. Hunermann, A. Karnsund, B. Hanotte, P. Chidananda, S. Nair, V. Badrinarayanan, A. Kendall, J. Shotton & O. Sinavski, *LingoQA: Visual question answering for autonomous driving*, European Conference on Computer Vision, Milan, Italy, 2024. https://doi.org/10.48550/arXiv.2312.14115.

[23] B. Hanindhito, B. Patel & L. K. John, *Large language model fine-tuning with low-rank adaptation: A performance exploration*, 16th ACM/SPEC International Conference on Performance Engineering, Toronto, Canada, 2025, pp. 92–104. https://doi.org/10.1145/3676151.3719377.

[24] A. Lacoste, A. Luccioni, V. Schmidt & T. Dandres, "Quantifying the carbon emissions of machine learning", arXiv preprint (2019) arXiv:1910.09700. https://doi.org/10.48550/arXiv.1910.09700.

[25] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky & J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning", Journal of Machine Learning Research **21** (2020) 248. https://doi.org/10.48550/arXiv.2002.05651.

[26] E. Strubell, A. Ganesh & A. McCallum, *Energy and policy considerations for deep learning in NLP*, 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 2019, pp. 3645–3650. https://doi.org/10.18653/v1/P19-1355.

[27] N. Belrose, Z. Furman, L. Smith, D. Halawi, I. Ostrovsky, L. McKinney, S. Biderman & J. Steinhardt, "Eliciting latent predictions from transformers with the tuned lens", arXiv preprint (2023) arXiv:2303.08112. https://doi.org/10.48550/arXiv.2303.08112.

[28] Y. Bengio, N. Leonard & A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", arXiv

preprint (2013) arXiv:1308.3432. https://doi.org/10.48550/arXiv.1308.3432.

[29] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier & J. Dean, "Carbon emissions and large neural network training", arXiv preprint (2021) arXiv:2104.10350. https://doi.org/10.48550/arXiv.2104.10350.

[30] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks & C.-J. Wu, *Chasing carbon: The elusive environmental footprint of computing*, 2021 IEEE International Symposium on High-Performance Computer Architecture, Seoul, Korea, 2021, pp. 854–867. https://doi.org/10.1109/HPCA51647.2021.00076.

[31] A. S. Luccioni, S. Viguier & A.-L. Ligozat, "Estimating the carbon footprint of BLOOM, a 176B parameter language model", Journal of Machine Learning Research **24** (2023) 253. https://www.jmlr.org/papers/v24/23-0069.html.

[32] B. Acun, B. Lee, F. Kazhamiaka, K. Maeng, U. Gupta, M. Chakkaravarthy, D. Brooks & C.-J. Wu, *Carbon Explorer: A holistic framework for designing carbon-aware datacenters*, 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Vancouver, Canada, 2023, pp. 118–132. https://doi.org/10.1145/3575693.3575754.