A Separable Architecture for Continuous Token Representation in Language Models

Reza T. Batley [1] **Sourav Saha** [1]

Abstract

Transformer scaling law analyses typically treat parameters as interchangeable; an abstraction that accurately predicts loss-compute relationships. Yet, in sub-billion-parameter small language models (SLMs), embedding matrices dominate the parameter budget. This work argues that this allocation is as suboptimal as it is counterintuitive. Leviathan is an architecture with a continuous embedding generator to replace the discrete lookup tables of canonical models. Evaluating on the Pile dataset under isoparametric settings, Leviathan consistently outperforms a standard, LLaMA-style architecture. By means of an empirical power-law fit, Leviathan exhibits a markedly superior effective parameter capacity. Across the regime studied, Leviathan behaves as a dense model with 1.47 to $2.11\times$ more parameters.

1. Introduction

The Transformer (Vaswani et al., 2017) has become the foundation of modern AI, powering breakthroughs in language (Brown et al., 2020; OpenAI, 2024), vision (Dosovitskiy et al., 2021; Carion et al., 2020) and multimodal learning (Alayrac et al., 2022; Reed et al., 2022). Since its introduction in 2017, countless architectural improvements have been attempted. With the desire for longer and longer context windows N, the $O(N^2)$ Attention mechanism has been placed squarely in the crosshairs (Beltagy et al., 2020; Dao et al., 2022; Zaheer et al., 2021; Child et al., 2019; Xiong et al., 2021; Tay et al., 2021; Choromanski et al., 2022). There have even been attempts to entirely replace the Attention mechanism with multilayer perceptrons (MLPs), albeit primarily for vision tasks (Tolstikhin et al., 2021; Liu et al., 2021).

[1]Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, United States. Correspondence to: Reza Batley <rezabatley@vt.edu>, Sourav Saha <souravsaha@vt.edu>.

Preprint. January 30, 2026.

By contrast, the static embedding matrix has received comparatively less scrutiny. In Large Language Models (LLMs) with standard vocabularies these parameters are often negligible. However, this assumption becomes increasingly fragile in emerging regimes; in Small Language Models (SLMs), particularly when paired with high-cardinality tokenizers, and omni-modal World Models tokenizing heterogeneous inputs ranging from visual patches to proprioceptive states.

To this end, this work introduces *Leviathan*, a Transformer incorporating a Separable Neural Architecture (SNA), as introduced in (Batley et al., 2025), that replaces the discrete, high-cardinality lookup table with a continuous token generator. In this embedding layer, Leviathan parameterizes the embedding space as a smooth surface, mapping tokens to a low-dimensional coordinate grid that is subsequently projected into the hidden space via a separable, continuous generator module. This approach allows the model to capture high-rank semantic interactions with $O(\sqrt[3]{V} \cdot D)$ parameter scaling, breaking the linear dependency between vocabulary size and model capacity.

In controlled experiments on the Pile dataset (Gao et al., 2020), Leviathan produces a consistent downwards shift on the parameter-loss frontier across the 60-420M parameter regime. Whilst the generator incurs a moderate throughput overhead ($23-51\%$, decreasing with scale) the gains in sample efficiency outweigh this overhead. In particular, Leviathan exhibits an effective capacity 1.5 to $2.1\times$ its actual parameter count. Notably, at the 421M scale, Leviathan achieves the validation loss of a $\approx 725M$ parameter dense model. Furthermore, within the investigated domain, Leviathan exhibits a consistently steeper scaling behavior both in parameters and samples when reasoning layers are held fixed. Leviathan often continues to grow its advantage as training progresses.

2. Background

Parametric Efficiency of Scaling The foundations of modern neural scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) rest on the assumption of parametric interchangeability: that a parameter added to depth is a parameter added to width at efficient frontiers, *ceteris paribus*.

This abstraction, however, breaks down at the boundaries of small scale and large vocabularies. In the regime of Small Language Models (SLMs, sub 10B parameters), the vocabulary embedding matrix $E \in \mathbb{R}^{V \times D}$ disproportionately consumes the parameter budget. Even a small tokenizer such as GPT-2's has a vocabulary exceeding 50,000. For a small model with hidden dimension $D = 1024$ this embedding matrix alone demands 50M parameters. In a 100M parameter model this "lookup table" taxes half of the model's capacity. The output head, typically the same size, would tax the remaining half of the 100M-model's capacity, leaving no room for reasoning through Transformer depth.

Weight Tying The standard approach to addressing this "vocabulary tax" is to share weights between the input embedding and pre-softmax output layer, a process known as tying (Press & Wolf, 2017; Inan et al., 2017). Whilst this recovers half the parameter budget, enabling the construction of a deeper model with the same memory footprint, it imposes a both philosophically inobvious and rigid constraint. In particular, that the geometry of the recognition space (input) is identical to the geometry of the predictive space (output). These roles are, however, functionally asymmetrical. Empirical work has demonstrated that input representations can often be hierarchically compressed without degrading performance, whilst the output head benefits from substantially higher expressive capacity (Baevski & Auli, 2019). This suggests that the input lookup table acts as a low-capacity retrieval mechanism, whilst the output head serves as a high-rank projection onto the probability manifold (Yang et al., 2018).

Factorization Another approach is via factorization of the input embedding matrix. Most notable here is ALBERT (Lan et al., 2020), which decomposes the matrix into three low-rank matrices of shape $V \times \mathcal{E}$ and $\mathcal{E} \times D$, $\mathcal{E} \ll D$. This reduces the parameter count from $O(V \cdot D)$ to $O(V \cdot \mathcal{E} + \mathcal{E} \cdot D)$, but imposes a strictly linear bottleneck between token identity and representation. Also note that this count remains linear in V. This forces token relationships to occupy a low-rank linear subspace which may limit the model's ability to represent high-order semantic interactions. Whilst Leviathan similarly imposes a form of rank bottleneck, it learns a continuous, nonlinear generator function. Furthermore, unlike factorization-based approaches, Leviathan removes linear dependence on V entirely.

Continuous Token Representations The transition from discrete lookup tables to continuous generators is, in part, motivated by the success of implicit neural representations (INRs) in computer vision and signal processing (Sitzmann et al., 2020; Mildenhall et al., 2020). In language modeling itself, attempts at using character-wise convolutions (Kim et al., 2015) or hashing mechanisms (Svenstrup et al., 2017) retained discrete sub-structures.

However, the emergence of Separable Neural Architectures (SNAs) in (Batley et al., 2025; Park et al., 2025) provides a smooth primitive for efficiently modeling high-dimensional data under rank constraints. This structure has further been demonstrated as a separable network head driven by a continuous latent space (Batley & Saha, 2026). Leviathan, named for the Hobbesian ideal of imposing order upon a state of nature, adopts this functional approach to language. Vocabulary is parameterized as coordinates within a continuous representation space, moving away from the assumption of token independence that lookup tables, even factorized ones, carry. This inductive bias allows the model to *interpolate* representations, potentially offering a principled way forward towards handling unseen tokens, or expanding vocabularies without the need for re-training.

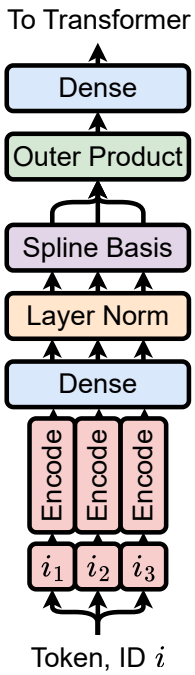

Figure 1. The generator module, replacing the standard dense input embedding matrix.

3. Architecture

3.1. A Continuous Token Generator

Leviathan's core innovation is its replacement of the discrete embedding matrix $E \in \mathbb{R}^{V \times D}$ with a separable generator function $\mathcal{G} : \{0, \ldots, V-1\} \to \mathbb{R}^D$. The representation learning process is composed of three distinct stages: latent compositional indexing, B-spline basis expansion and

tensor-product aggregation.

Latent compositional indexing This stage factorizes the vocabulary $\mathcal{V}$ into a latent coordinate grid of k dimensions. One defines a base $b = \left\lceil \sqrt[k]{V} \right\rceil$. Each token of index i is then deterministically mapped to a coordinate $(i_1, \ldots, i_k)$ using a base-b decomposition. Each element of this coordinate indexes into k shared latent codebooks $C_1, \ldots, C_k \in \mathbb{R}^{b \times d_{\text{seed}}}$ that are learned. The resulting vectors are summed to produce a seed $z \in \mathbb{R}^{d_{\text{seed}}}$, so that,

$$z(i) = \sum_{r=1}^{k} C_r[i_r] = \sum_{r=1}^{k} C_r \left[\left\lfloor \frac{i}{b^{k-r}} \right\rfloor \bmod b \right]. \quad (1)$$

This induces the parameter compression of the indexing stage from $O(V)$ to $O\left(k \sqrt[k]{V}\right)$. The produced seed $z(i)$ is projected into a latent space via a dense layer then to the unit hypersphere by means of a layer normalization $\mathrm{LN}(\cdot)$. Finally, a sigmoid activation maps everything to the unit hypercube. This latent coordinate is $\tilde{z} \in [0,1]^{d_{\text{seed}}}$.

B-spline basis expansion This stage constitutes the main philosophical departure from other approaches to token embedding; $\tilde{z}(i)$ is treated as input to a function approximator rather than a static vector. In particular, this function approximator seeks to learn a smooth surface $\mathcal{M}$ in the latent space. Following the KHRONOS architecture (Batley & Saha, 2025; Sarker et al., 2026), each dimension r of the latent space is modeled by a (univariate) B-spline basis expansion $\phi_r(x_r)$.

Tensor product aggregation A rank-1 separable component, a *mode*, is constructed by a tensor product $\prod_{r=1}^{d_{\text{seed}}} \phi_r(x_r)$. This produces smooth and global, i.e. full-dimensional, structure but lacks expressivity. With M such modes, the model's ansatz for this surface takes the form

$$\mathcal{M}(x) \approx \sum_{j=1}^{M} \prod_{r=1}^{d_{\text{seed}}} \phi_{r,j}(x_r). \quad (2)$$

It is noteworthy that this structure is a universal approximator under standard conditions (formal proof via Stone-Weierstrass is provided in Appendix A). A final dense layer projects the aggregated modes with a residual connection from the initial seed projection to ensure stable gradient flow

$$e_i = W_{\text{out}} \begin{pmatrix} \mathcal{M}_1(\tilde{z}_i) \\ \vdots \\ \mathcal{M}_M(\tilde{z}_i) \end{pmatrix} + W_{\text{res}} \tilde{z}_i. \quad (3)$$

Thus each token i generates an embedding e_i. In contrast to discrete embedding tables in which parameters are isolated to specific indices; here the weights W_{out} are token-agnostic, a property of the surface. This encourages the model to encode structure and similarity within the surface.

3.2. Full Model Architecture

The embeddings produced by the generator e_i are fed into a decoder-only Transformer backbone. To isolate the efficacy of this generator from other confounding architectural variables, adopted is a modern "LLaMA"-style specification (Touvron et al., 2023) henceforth referred to as the *Dense* control.

Transformer Backbone Specification In the experiments that follow, the network depth L is the control enabling isoparametricity, *ceteris paribus*. Fixed component blocks employ a pre-layer normalization, known for enhancing training stability at increased depths (Xiong et al., 2020). For positional injection, Rotary Position Embeddings (RoPE) (Su et al., 2023), the contemporary choice, are applied to the query and key matrices at each attention layer. The attended outputs are again layer-normalized before projection and extraction in a high-dimensional feature space, contemporarily $4 \times D$, the hidden dimension. This stage utilizes a dense feed-forward network with SwiGLU (Swish-gated Linear Unit) activation functions (Shazeer,

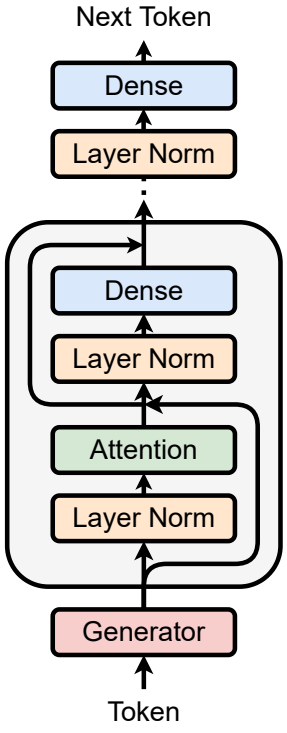

Figure 2. Leviathan architecture, highlighting the replacement of the input embedding matrix with a generator module.

2020).

The final hidden states h_L are layer normalized and projected out to the vocabulary size V via a final dense classification head $W_{\text{class}} \in \mathbb{R}^{D \times V}$. Leviathan decouples token embedding and classification akin to untied-weight networks. However, the implications of this design choice are compared both with tied and untied networks in Section 4. All models are trained using the standard next-token, cross-entropy loss objective.

4. Experiments

Leviathan effectively decouples the number of input embedding parameters from vocabulary size. This architectural shift, compared to untied weights, recovers capacity to be reinvested into the "brain" of the model. For the sake of clean experimentation prioritizing *ceteris paribus*, in head to head comparisons between Dense and Leviathan models, the hidden dimension D is held fixed. Isoparametricity is thus restored by increasing Leviathan depth L. To this end, the following evaluation framework is designed to disentangle the benefits of the generator from the benefits of depth. In particular, the two regimes of focus are

- **Iso-body:** Both Dense and Leviathan configurations share an identical Transformer *brain* (depth, width and number of heads). This regime isolates the representational capacity of Leviathan's generator input with untied output head against a standard tied embedding table.

- **Isoparameter:** the parameter savings afforded by Leviathan's generator (relative to the input embedding in an untied model) are not reinvested into its brain in a *depth dividend*.

All experiments use the `o200k_base` tokenizer. This vocabulary is larger-than-typical for an SLM, but reflects contemporary practice in World Models and multilingual systems. Large vocabularies also implicitly widen the effective context window by reducing sequence length. This choice is intentional: in this regime the *vocabulary tax* becomes a practical bottleneck Leviathan is explicitly designed to address.

ALBERT-style factorized embeddings were also considered as a potential baseline. However, e.g. in the upcoming Leviathan-60M model the generator contains only $\approx 2.5M$ parameters. Given the large vocabulary size, matching this with ALBERT would require a bottleneck of around 12, an unrealistically constrained regime. To this end, it is apparent that the tied-weight, iso-body experiment serves as a more conservative baseline.

4.1. The Representational Prior: Iso-Body Analysis

In this regime, the Transformer brain is identical for both models, asking: *what is the effect of replacing a tied dense embedding with a generator-based input head?*

Whilst the Leviathan generator introduces a marginal parameter overhead, the observed gains significantly exceed what would be predicted by canonical scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) for such a small increase in the number of parameters, N. As shown in the left panel of Figure 3, Leviathan is Pareto dominant at the data-efficient frontier, all models having been trained on approximately $20 \times N$ tokens.

The right panel of Figure 3 illustrates the magnitude of this effect; it is substantial. Leviathan achieves between a 6.7% and 18.1% reduction in validation perplexity, the measure of a model's effective uncertainty over the next token, across scales.

Holding the Transformer architecture and training protocol fixed isolates the structural contribution of the input representation itself. That this representation's expressive power grows as its overhead diminishes supports the hypothesis that the generator facilitates more effective parameter sharing across the vocabulary than tied dense embeddings.

4.2. The Depth Dividend: Isoparametric Analysis

In the previous analysis, depth was strictly controlled. In practice, however, architectures are typically rather constrained by parameter budgets. Since, compared to an untied baseline, Leviathan saves parameters on embeddings, one may *reinvest* those into depth. At the $109M$ scale, for example, compared to the shallow 6-layer Dense baseline, this surplus is transformative: Leviathan's isoparametric specification is an ultra-deep, 52-layer network.

To interpret the following results, the Dense-equivalent *effective size* of a model is defined. As described in Section 5, the untied Dense baseline is treated as a reference family

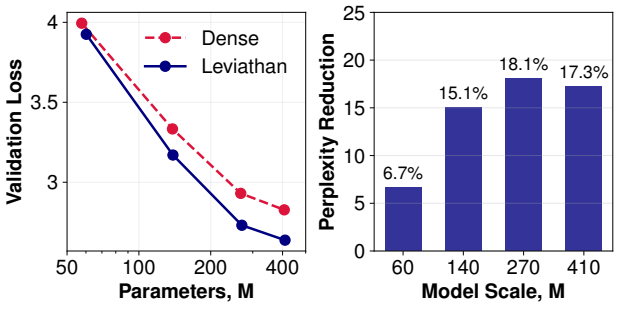

Figure 3. (Left) The parameter-loss Pareto frontier under iso-body control. *(Right)* Percentage reduction in validation perplexity achieved by Leviathan relative to the tied Dense baseline.

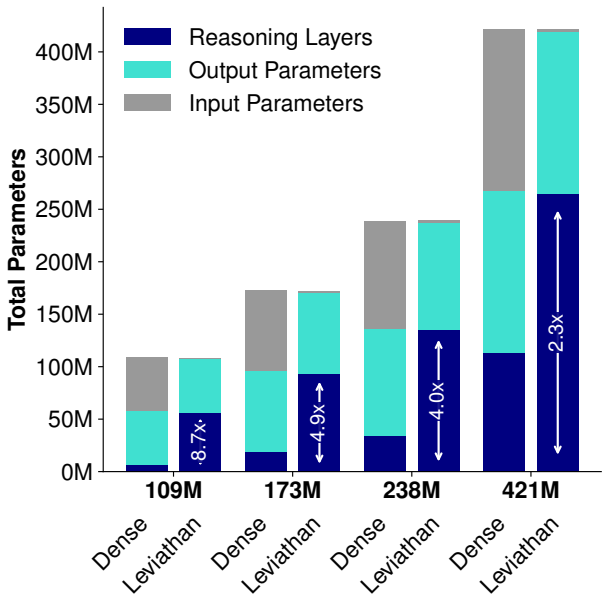

Figure 4. Parameter allocation across components for untied Dense and Leviathan models under fixed parameter budgets. Leviathan reallocates input parameters from embeddings into deeper reasoning layers.

to which an empirical scaling law is fitted. This provides a mapping between parameter count and validation loss for the baseline architecture $\mathcal{L}(N)$. For a Leviathan model achieving a validation loss $\mathcal{L}^*$, its "effective size" is defined as the inversion of this law $N_{\text{eq}}(\mathcal{L}^*)$. This construction uses the Dense scaling curve as a calibrated "ruler" for expression performance differences in "units" of parameter budget. Under this interpretation, N_{eq} quantifies *ipso facto* how much larger a Dense model would need to be in order to match Leviathan's performance.

Figure 5 visualizes the consequence of this structural shift over a range of model scales from 109-421M. As shown in the right panel, Leviathan outperforms its parameter count across the board. This effect peaks at the 109M scale, with the model achieving a validation loss equivalent to a dense model of 230M parameters, an effective size multiplier of 2.11. Even at the 421M scale, where the relative embedding tax is lower, Leviathan maintains a substantial 1.72× effective size advantage - corresponding to a 724M parameter dense model. This empirically reinforces the concept of a "depth dividend"; the reinvested parameters contribute more to reducing loss.

4.3. Second Descent: the Overtraining Payoff

Contemporary training regimes for Small Language Models have shifted away from compute-optimal (Hoffmann et al., 2022) towards "overtraining" (Grattafiori et al., 2024). A critical question is, therefore, how Leviathan performs in this long-horizon regime.

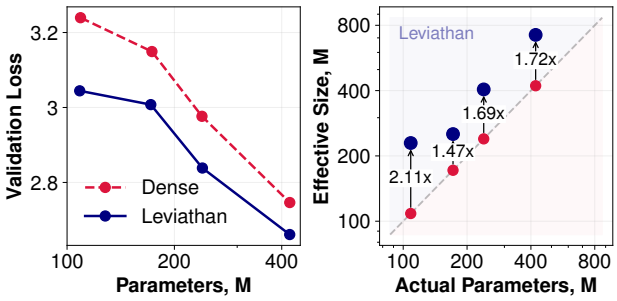

Figure 5. (*Left*) The parameter-loss Pareto frontier under isoparametric control. (*Right*) Effective Dense-equivalent size inferred from a fitted Dense parameter scaling law.

Figure 6 plots the relative perplexity decrease in Leviathan, compared to its concomitant dense baseline, over the course of training. It is notable that this advantage increases, approximately monotonically, with tokens seen. Furthermore, this advantage does not plateau at the compute-optimal frontier rather continuing to increase. This effect is most notable in the iso-body control setting (*left*). This is most pronounced in the 60M-class, where Leviathan's advantage continues to grow even into the $100\times N$-tokens-seen regime. In Leviathan-138M, similar continuous growth is seen even at $40\times N$ tokens seen. It is worth noting that this reduction is not greatly spread across scales: even Leviathan-410M achieves a near-20% reduction in perplexity, despite gains at larger scales typically being substantially harder to obtain.

In the isoparametric control setting (*right*) where Leviathan "reinvests" parameters into depth, the divergence is more spread across model classes. Smaller models exhibit larger relative gains whilst larger models maintain a consistent, but decreasing, advantage. Despite this, Leviathan-190M's reduction in perplexity continues to grow well into the "overtraining" regime, up to $60\times N$ tokens seen.

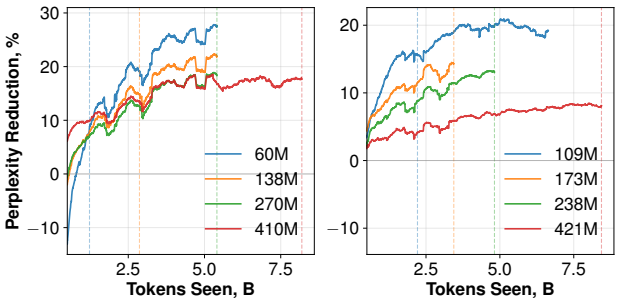

Figure 6. Perplexity reduction (%) of Leviathan models relative to their paired Dense baselines as a function of tokens seen. (*Left*) iso-body runs with matched Transformer backbones. (*Right*) isoparametric runs: matched parameter budgets. Vertical dashed lines mark $20\times N$ tokens seen.

Table 1. Experimental setup for the iso-body runs.

MODEL	PARAMETERS	SETUP		
		D	L	H
DENSE	57,600,512	256	6	4
LEVIATHAN	60,185,912	256	6	4
DENSE	138,287,416	512	8	8
LEVIATHAN	138,897,208	512	8	8
DENSE	267,210,240	768	12	12
LEVIATHAN	270,057,784	768	12	12
DENSE	406,611,968	1024	12	16
LEVIATHAN	409,590,584	1024	12	16

Table 2. Experimental setup for the isoparameter runs.

MODEL	PARAMETERS	SETUP		
		D	L	H
DENSE	109,097,144	256	6	4
LEVIATHAN	108,514,616	256	52	4
DENSE	172,988,856	384	8	6
LEVIATHAN	171,728,440	384	39	6
DENSE	238,973,624	512	8	8
LEVIATHAN	239,658,808	512	32	8
DENSE	421,299,384	768	12	12
LEVIATHAN	421,151,032	768	28	12

4.4. Model Configurations

The evaluation focuses on the Small Language Model (SLM) regime, spanning from 60-420M parameters. In this window embedding matrices greatly tax model capacity from $20-50\%$ in tied models to potentially well over 90% in untied models.

All models employ a decoder-only Transformer backbone with identical architectural components discussed in Section 3.2 (pre-layer normalization, RoPE, SwiGLU feed-forward layers). Each configuration pairs up a Dense and Leviathan model with width D and number of Attention heads H matched. Depth L is either held fixed (*iso-body* with Dense weights tied) or increased in Leviathan (*isoparametric* with Dense weights untied) until isoparametricity is nearest restored. Aside from L in the isoparametric case and Leviathan's generator module replacing the input embedding matrix, all else both architectural and in matters of training is held equal.

Tables 1 and 2 summarize all eight model configurations: the four iso-body and four isoparametric, respectively.

4.5. Training Protocol

Optimization Models were implemented in JAX/Flax (Frostig et al., 2018) and trained from scratch using AdamW (Loshchilov & Hutter, 2019) with gradient clipping. Every model is trained with a sequence length of 512 and a batch size of 512.

Data Models were trained on the Pile (Uncopyrighted) (Gao et al., 2020) using a streaming dataloader with shuffle buffer of 10,000 sequences to randomize the data distribution. Input text was tokenized using the `o200k_base` tokenizer in tiktoken (OpenAI, 2023), used in GPT-4o and 4o-mini models. This tokenizer is of cardinality 200,018 but is padded to $V = 200{,}376$, a perfect cube, 59^3. This motivates $k = 3$, also a balanced choice. Each token i is thus mapped to a three-dimensional coordinate (i_1, i_2, i_3) via base-59 decomposition. This reduces the vocabulary footprint on indexing parameters from 200,376 to 177, a greater than $1{,}100\times$ compression.

Crucially, for every matched Dense-Leviathan pair, token streams were identical. Each model was trained on the same sequence, in the same order, at every optimization step.

Generator Hyperparameters The SNA-based generator uses a fixed set of hyperparameters across all runs. In particular, the generator takes a seed dimensionality of $d_{\text{seed}} = 128$. This 128-dimensional space is target of basis expansion. Each univariate B-spline basis is defined over a grid of 32 knots, aggregated over $M = 8$ independent rank-1 modes to form the embedding e_i.

Reported validation losses are computed as the mean over the last 10% of training steps to reduce batch noise. All Dense-Leviathan paired models are evaluated at identical token counts.

5. Scaling Exponents and Efficiency Frontiers

In order to investigate potential structural advantages of Leviathan, its performance is mapped onto the empirical power-law fits of the dense baselines. Models are evaluated at their $20\times N$-token convergence frontiers. Power laws of the form $\mathcal{L}(N) = AN^{-\alpha} + b$ for parameter efficiency and $\mathcal{L}(D) = BD^{-\beta} + b$ are fit to the four dense models of either regime. The irreducible loss term b is taken as 1.69. This is all standard practice as of (Hoffmann et al., 2022), but it is notable that different tokenizers and datasets were used. Be that as it may, 1.69 serves as a conservative and widely used reference value that anchors the analysis to established scaling practice. Critically, the following conclusions are robust to reasonable variation in b. Finally, it is worth noting that these fits are only reliably descriptive within the narrow SLM regime studied.

5.1. The Iso-body Regime

Parameter Efficiency Across the entire evaluated scale, Leviathan lies strictly below the dense frontier. This suggests the improvement is not confined to a single operating point and is, rather, systematic. As illustrated in the left panel of Figure 7, Leviathan's parameter-scaling exponent of 0.47 is steeper than the tied-embedding baseline's 0.38. This would suggest that when Transformer backbone is held fixed, Leviathan's advantage widens with parameter count. However, further testing at larger model sizes that span multiple orders of magnitude is required to validate this result.

Sample Efficiency Similarly, as seen in the right panel of Figure 7, Leviathan demonstrates an increase in data-scaling exponent from 0.39 in the dense baselines to 0.47. This indicates that Leviathan extracts more benefit per every additional token seen, even at a fixed architectural capacity.

Concretely, the recovered power laws are

$$\mathcal{L}_{\text{Dense}}(N) = 2000N^{-0.38} + 1.69, \tag{4}$$
$$\mathcal{L}_{\text{Leviathan}}(N) = 9800N^{-0.47} + 1.69, \tag{5}$$
$$\mathcal{L}_{\text{Dense}}(D) = 180D^{-0.39} + 1.69, \tag{6}$$
$$\mathcal{L}_{\text{Leviathan}}(D) = 450D^{-0.47} + 1.69. \tag{7}$$

5.2. The Isoparametric Regime

Parameter Efficiency Under isoparametric control, as can be observed in 8, both models continue to demonstrate the expected power-law behavior. In this regime, as can be seen in the left panel of Figure 8 the parameter-scaling exponents are similar (0.26 Leviathan vs 0.29 dense) suggesting that both architectures scale comparably with respect to parameter count. It is notable, however, that Leviathan's component is somewhat smaller. In the tested domain, Leviathan's substantially smaller prefactor (166 compared to 360 for the dense baselines) allows it to remain Pareto dominant. Whether this advantage is maintained for larger models, or whether there is a crossover requires further validation work.

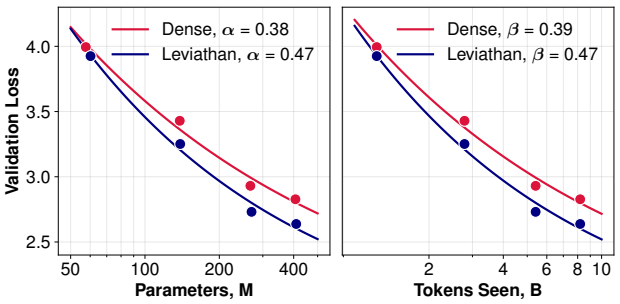

Figure 7. Scaling law fits for the Dense and Leviathan models under iso-body control. (*Left*) The parameter-efficient frontier: validation loss as a function of total model parameters. (*Right*) The data-efficient frontier: validation loss as a function of tokens seen.

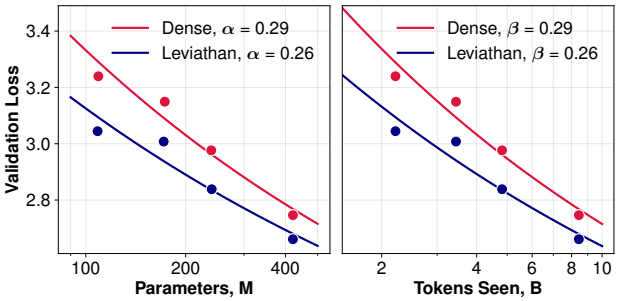

Figure 8. Scaling law fits for the Dense and Leviathan models under isoparametric control. (*Left*) The parameter-efficient frontier: validation loss as a function of total model parameters. (*Right*) The data-efficient frontier: validation loss as a function of tokens seen.

Sample Efficiency A near-identical pattern is observed in the data-efficient front, the right panel of Figure 8. The recovered data-scaling exponents in this regime are 0.26 for Leviathan, 0.29 for the dense baselines. A smaller Leviathan prefactor again explains the Pareto dominance observed over this domain. This suggests the principal benefit in this regime derives from a superior prefactor rather than altered scaling exponents.

Explicitly, the recovered power laws are

$$\mathcal{L}_{\text{Dense}}(N) = 360N^{-0.29} + 1.69, \tag{8}$$
$$\mathcal{L}_{\text{Leviathan}}(N) = 166N^{-0.26} + 1.69, \tag{9}$$
$$\mathcal{L}_{\text{Dense}}(D) = 52D^{-0.29} + 1.69, \tag{10}$$
$$\mathcal{L}_{\text{Leviathan}}(D) = 31D^{-0.26} + 1.69. \tag{11}$$

6. Discussion

Implications for Long-Context Modeling The pursuit of long-context capabilities has predominantly focused on ways of reducing the quadratic cost of the Attention mechanism. One may identify a secondary avenue, *the trade-off between sequence length and vocabulary density*. Typically, wisely managing a parameter budget necessitates a "small" vocabulary, forcing the tokenizer to decompose semantic units into multiple sub-tokens. This inflates the sequence length N_{seq} and exacerbates the $O(N_{\text{seq}}^2)$ Attention cost. By, in effect, decoupling vocabulary cardinality from parametric cost, Leviathan takes a stride towards resolving this problem. It enables the use of high-density tokenizers, perhaps far beyond typical, without the usual "vocabulary tax". Simply, Leviathan provides a mitigation to quadratic Attention scaling by enabling shorter sequences carrying no less information, for a negligible increase in parameters.

Implications for World Models This architectural shift is particularly potent in the rising area of omni-modal "World Models". Such models might ingest continuous, high-entropy signals such as video patches, audio waveforms (Borsos et al., 2023) or even scalar streams (Reed et al., 2022). Current approaches force these signals into discrete, finite lookup tables resulting in quantization noise and large vocabularies (LeCun & Courant, 2022). Whilst Leviathan relies on a tokenizer to map signals to discrete indices the generator treats these indices as points on a smooth surface, "coordinate quantization". This allows the tokenization layer to be purely geometric, mapping similar signals to adjacent coordinates with the generator handling interpolation. This is a mitigation of the lookup nature of standard quantization, offering an approximation to continuous signals whilst also scaling favorably with vocabulary size.

Arbitrary Tokenizer Topology Leviathan assigns each token a coordinate via a deterministic modular-arithmetic mapping of its index. Tokens that are adjacent in coordinate space thus need not be linguistically related. Therefore, the generator is forced to "undo" this local-but-not-semantic structure. That Leviathan remains robust to this choice and achieves a gain across all scales tested suggests the generator manages exactly that. Nonetheless, an exploration of alternative coordinate constructions such data-driven tokenizers or learned coordinate embeddings represents a promising direction for future work.

Computational Trade-offs Leviathan introduces additional computational overhead relative to standard embedding lookup. In the iso-body regime, a training throughput reduction of 51% at 60M parameters, decreasing to 23% at 410M is observed. As shown in Figure 7, however, Leviathan achieves equivalent loss having seen roughly $1.7\times$ fewer tokens at the 400M scale. Since this gain in sample efficiency is greater than the overhead ($1.3\times$ at 23%), Leviathan is faster to train to a given loss.

Furthermore, this slowdown is not primarily through an increase in FLOPs, rather due to implementation-level effects. The current generator module relies on scatter-heavy operations that are not as well-optimized on modern accelerators as dense matrix operations. This cost is potentially a systems issue rather than an inherent architectural limitation, allowing for speedups from a more optimized, perhaps kernel-level, implementation.

In the isoparametric regime, models are intentionally made deeper to preserve parity. The resulting throughput reduction (48%-71%, but decreasing with scale) is largely attributable to increased depth - inherently serial - than to Leviathan itself. This decision is experimental control designed to isolate architectural effects, not a deployment recommendation. In practice, parameters would be better reallocated in a combination of width and depth. In fact, Leviathan's parameter-light generator module would be more amenable to a width increase. Further investigation in this regard is warranted.

7. Conclusion

This work introduced Leviathan, a Transformer-based architecture adopting a Separable Neural Architecture (SNA)-based "generator" module for coordinate quantization of text. Empirical analysis of Leviathan compared to dense baseline established two key findings. Firstly, Transformer hyperparameters (hidden dimension, layer count, number of Attention heads) are held fixed, and the dense baseline is taken with tied embeddings. Under this control, Leviathan is *Pareto dominant*; for a given parameter count Leviathan achieves a better loss over the tested domain. This result validated that the generator offers superior representational capacity than a tied input embedding *ceteris paribus*.

Secondly, in replacing the standard discrete embedding matrix of a Dense baseline with generator, Leviathan demonstrated that Small Language Models (SLMs) can reclaim up to 150% of their parameter budget. Under strict isopara-metric control, Leviathan enables a "depth dividend" to be reinvested into its reasoning layers. Across the four model sizes tested, this manifested from $2.3 - 8.7\times$ more depth for the same number of parameters. The benefit of this parametric reallocation was borne out in the results, with Leviathan achieving validation losses that an empirically fitted scaling law predicts models $1.47 - 2.11\times$ larger would achieve. This suggests that prevailing scaling laws for SLMs are artificially constrained by the inefficient parameterization of the input layer.

Finally, note that the continuous nature of the generator suggests a theoretical pathway towards dynamic vocabulary adaptation. This would, in principle, be a generative surface onto which unseen tokens could be interpolated solely via tokenizer updates. This offers a potential avenue for future research into open-vocabulary models that "understand" new tokens without full retraining. Immediate future work will focus on optimizing the generator to reduce training overhead, as well as extension to multi-modal data, VLMs and World Models.

Acknowledgements

The authors are grateful to Apurba Sarker for generously contributing his time and compute resources to run experiments for this project.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. Flamingo: a visual language model for few-shot learning, 2022. URL https://arxiv.org/abs/2204.14198.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling, 2019. URL https://arxiv.org/abs/1809.10853.

Batley, R. T. and Saha, S. Khronos: a kernel-based neural architecture for rapid, resource-efficient scientific computation, 2025. URL https://arxiv.org/abs/2505.13315.

Batley, R. T. and Saha, S. *A Unified Generative-Predictive Framework for Deterministic Inverse Design*. AIAA SciTech Forum, 2026. doi: 10.2514/6.2026-0365. URL https://arc.aiaa.org/doi/abs/10.2514/6.2026-0365.

Batley, R. T., Saha, S., Park, C., and Liu, W. K. An explainable artificial intelligence framework enabled by a separable neural architecture. *Computational Mechanics*, 2025. doi: 10.1007/s00466-025-02719-w. URL https://doi.org/10.1007/s00466-025-02719-w.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Borsos, Z., Marinier, R., Vincent, D., Kharitonov, E., Pietquin, O., Sharifi, M., Roblek, D., Teboul, O., Grangier, D., Tagliasacchi, M., and Zeghidour, N. Audiolm: a language modeling approach to audio generation, 2023. URL https://arxiv.org/abs/2209.03143.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers, 2020. URL https://arxiv.org/abs/2005.12872.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Frostig, R., Johnson, M., and Leary, C. Compiling machine learning programs via high-level tracing. 2018. URL https://mlsys.org/Conferences/doc/2018/146.pdf.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Grattafiori, A., Dubey, A., Jauhri, A., and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Inan, H., Khosravi, K., and Socher, R. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. Character-aware neural language models, 2015. URL https://arxiv.org/abs/1508.06615.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations, 2020. URL https://arxiv.org/abs/1909.11942.

LeCun, Y. and Courant. A path towards autonomous machine intelligence version 0.9.2, 2022-06-27, 2022. URL https://api.semanticscholar.org/CorpusID:251881108.

Liu, H., Dai, Z., So, D. R., and Le, Q. V. Pay attention to mlps, 2021. URL https://arxiv.org/abs/2105.08050.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. URL https://arxiv.org/abs/2003.08934.

OpenAI. tiktoken: A fast bpe tokenizer for openai models. https://github.com/openai/tiktoken, 2023.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Park, C., Saha, S., Guo, J., Zhang, H., Xie, X., Bessa, M. A., Qian, D., Chen, W., Wanger, G. J., Cao, J., Hughes, T. J. R., and Liu, W. K. Unifying machine learning and interpolation theory via interpolating neural networks. *Nature Communications*, 16(1):8753, 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-63790-8.

Press, O. and Wolf, L. Using the output embedding to improve language models, 2017. URL https://arxiv.org/abs/1608.05859.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

Sarker, A., Batley, R. T., Sarojini, D., and Saha, S. *A Kernel-based Resource-efficient Neural Surrogate for Multi-fidelity Prediction of Aerodynamic Field*. AIAA SciTech Forum, 2026. doi: 10.2514/6.2026-0043. URL https://arc.aiaa.org/doi/abs/10.2514/6.2026-0043.

Shazeer, N. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions, 2020. URL https://arxiv.org/abs/2006.09661.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Svenstrup, D., Hansen, J. M., and Winther, O. Hash embeddings for efficient word representations, 2017. URL https://arxiv.org/abs/1709.03933.

Tay, Y., Bahri, D., Metzler, D., Juan, D.-C., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models, 2021. URL https://arxiv.org/abs/2005.00743.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. Mlp-mixer: An all-mlp architecture for vision, 2021. URL https://arxiv.org/abs/2105.01601.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture, 2020. URL https://arxiv.org/abs/2002.04745.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021. URL https://arxiv.org/abs/2102.03902.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank rnn language model, 2018. URL https://arxiv.org/abs/1711.03953.

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences, 2021. URL https://arxiv.org/abs/2007.14062.

A. Universal Approximation of the Leviathan Generator

This appendix formalizes the claim made in Section 3.1 that the generator is a universal approximator over continuous functions.

Theorem A.1 (Universal Approximation of CP-Separable Neural Architectures)**.** *Let $\Omega = [0,1]^d$. The class of separable functions $\mathcal{A}$ is dense in $C(\Omega)$ with respect to the uniform norm $\|\cdot\|_\infty$.*

Proof. For each coordinate $i \in \{1, \dots, d\}$, let

$$\tilde{U}^{(i)} = \operatorname{span}\{\phi_\alpha^{(i)}(x) : \alpha \in \{1, \dots, r\}, x \in [0,1]\} \tag{12}$$

be a function space dense on $C([0,1])$. Let $U^{(i)} = \operatorname{alg}(\tilde{U}^{(i)})$ be the univariate algebra generated by $\tilde{U}^{(i)}$, with $1 \in U^{(i)}$. Further, define the set

$$\mathcal{A} \;=\; \left\{ \sum_{j=1}^{r} \prod_{i=1}^{d} a_j^{(i)}(x_i) : a_j^{(i)} \in U^{(i)},\ r \in \mathbb{N} \right\}. \tag{13}$$

This is the separable model with rank r and $k = d$. Pick arbitrary $f, g \in \mathcal{A}$ such that

$$f(x) = \sum_{j=1}^{r} \prod_{i=1}^{d} \alpha_j^{(i)}(x_i), \;\; g(x) = \sum_{k=1}^{s} \prod_{i=1}^{d} \beta_k^{(i)}(x_i), \tag{14}$$

with $\alpha_j^{(i)}, \beta_k^{(i)} \in U^{(i)}$. Then,

$$f(x) + g(x) = \sum_{j=1}^{r} \prod_{i=1}^{d} \alpha_j^{(i)}(x_i) + \sum_{k=1}^{s} \prod_{i=1}^{d} \beta_k^{(i)}(x_i), \tag{15}$$

$$= \sum_{l=1}^{r+s} \prod_{i=1}^{d} \gamma_l^{(i)}(x_i) \quad \text{where } \gamma_l^{(i)} \in U^{(i)}. \tag{16}$$

Therefore $f + g \in \mathcal{A}$, confirming closure under addition. Take an arbitrary element $f \in \mathcal{A}$ as before, and multiply by an arbitrary scalar $\lambda \in \mathbb{R}$ so

$$\lambda f(x) = \sum_{j=1}^{r} \lambda \prod_{i=1}^{d} \alpha_j^{(i)}(x_i) = \sum_{j=1}^{r} \prod_{i=1}^{d} \tilde{\alpha}_j^{(i)}(x_i), \tag{17}$$

with $\tilde{\alpha}_j^{(1)} = \lambda \alpha_j^{(1)}$, all else unchanged. Since $U^{(i)}$ is a vector space, $\lambda \alpha_j^{(1)} \in U^{(i)}$, thus $\lambda f \in \mathcal{A}$ and $\mathcal{A}$ is closed under scalar multiplication. It has been shown that $\mathcal{A}$ is closed under addition and scalar multiplication, and is therefore a vector space. Pick $f, g \in \mathcal{A}$ as before. Then, their product is

$$(fg)(x) = \left(\sum_{j=1}^{r} \prod_{i=1}^{d} \alpha_j^{(i)}(x_i) \right) \left(\sum_{k=1}^{s} \prod_{i=1}^{d} \beta_k^{(i)}(x_i) \right), \tag{18}$$

$$= \sum_{j=1}^{r} \sum_{k=1}^{s} \prod_{i=1}^{d} \gamma_{jk}^{(i)}(x_i),\ \gamma_{jk}^{(i)} = \alpha_j^{(i)}(x_i) \beta_k^{(i)}(x_i). \tag{19}$$

Recall $U^{(i)} = \operatorname{alg}(\tilde{U}^{(i)})$ is an unital algebra and so it follows by definition that $\gamma_{jk}^{(i)} \in U^{(i)}$. Therefore, $(fg)(x) \in \mathcal{A}$, and $\mathcal{A}$ is an algebra: closed under multiplication. Since $1 \in U^{(i)}$, it follows that $\mathcal{A}$ is a unital algebra.

Take two distinct points $x, y \in [0,1]^d$, $x = (x_1, \dots, x_d),\ y = (y_1, \dots, y_d)$, with $x_i \neq y_i$ for at least one $i = 1, \dots, d$. Now, because $U^{(i)}$ is dense on $C([0,1]), \exists\ u \in U^{(i)}$ with $u(x_i) \neq u(y_i)$ for any $x_i \neq y_i$. Then the separable function $f(z) = u(z_i) \in \mathcal{A}$ and satisfies $f(x) \neq f(y)$. Therefore, $\mathcal{A}$ separates points.

Any unital subalgebra on $C([0,1]^d)$ that separates points is itself dense on $C([0,1]^d)$, given that $[0,1]^d$ is a compact Hausdorff space. This follows from Stone-Weierstrass. Indeed, $\mathcal{A}$ is dense on $C([0,1]^d)$; for a given $f^* \in C([0,1]^d)$ and any $\varepsilon > 0$, density yields $f \in \mathcal{A}$ such that

$$\|f^* - f\|_\infty < \varepsilon. \tag{20}$$

By definition of $\mathcal{A}$, f is a *finite* sum of products. Therefore, there exists r and functions $\{\phi_j^{(i)}\}$ such that

$$\sup_{x\in[0,1]^d} \left| f^*(x) - \sum_{j=1}^{r} \prod_{i=1}^{d} \phi_j^{(i)}(x_i) \right| < \varepsilon, \tag{21}$$

which is the desired universal approximation statement. □

B. Implementation Details

All models were trained on a single NVIDIA H200 GPU with 141GB VRAM in JAX/Flax.

Generator Configuration The Leviathan generator utilizes a seed dimension $d_{seed} = 128$. The B-spline basis expansion employs a grid of $\kappa = 32$ knots with quadratic B-splines. The aggregation layer sums over $M = 8$ rank-1 separable modes. The coordinate mapping uses a base $b = \lceil V^{1/3} \rceil = 59$.

Optimization The AdamW optimizer is utilized with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1\times10^{-8}$, and no weight decay. The learning rate follows a cosine decay schedule with a linear warmup of 1000 steps starting from a minimum learning rate of 1×10^{-5}. Peak learning rates were set to 3×10^{-4} for all besides the 109M-class and 421M-class models for which they were 6×10^{-4} and 2×10^{-4}, respectively. In the iso-body regime, the 60-, 138-, 270- and 410-M models were trained for 75, 170, 330 and 500 thousand steps, respectively. In the isoparametric regime, the 109-, 173-, 238- and 421-M models were trained for 135, 210, 294 and 515 thousand steps, respectively.

Batching and Gradient Accumulation To manage memory constraints while maintaining a large effective batch size, gradient accumulation was employed across 16 physical steps. The physical batch was 32 sequences per step, with a sequence length of 512 tokens. Thus the number of tokens per physical step was 16,384; this is the resolution of the training logs. The logical batch size was 512 sequences (physical batch accumulated over 16 steps). Therefore, every gradient update took $512^2 = 262,144$ tokens.

C. Learning Curves

The complete optimization trajectories for all model configurations are presented here. Figure 9 displays the validation loss profiles for the iso-body regime, whilst Figure 10 illustrates the isoparametric comparisons. In both settings, Leviathan generally outperforms the dense baselines, with the performance gap often widening as training progresses.

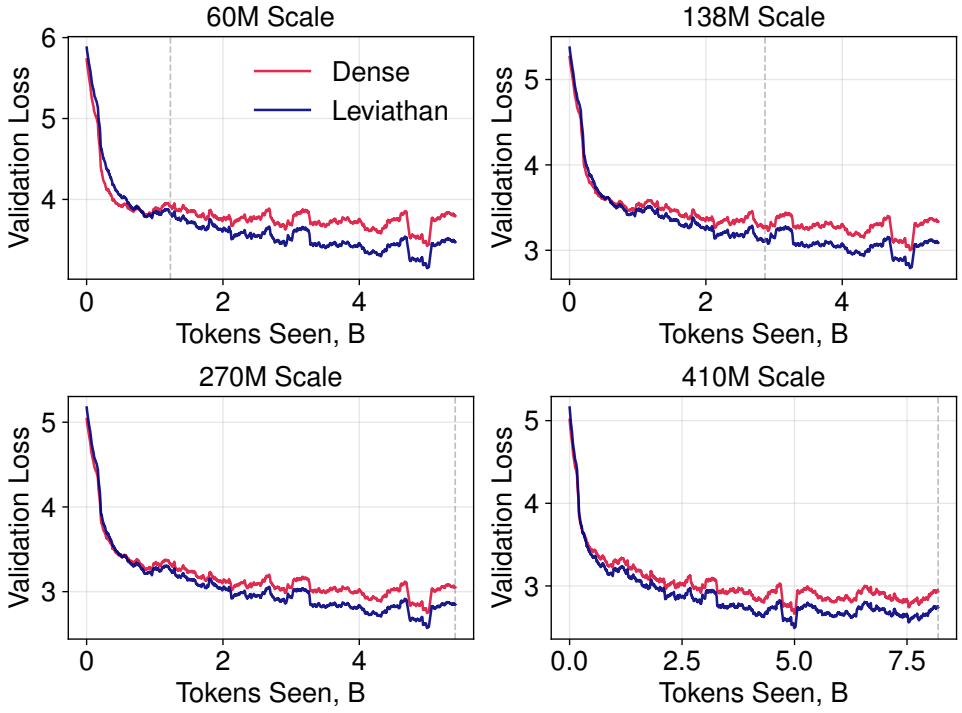

Figure 9. The *iso-body* learning curves; tracking validation loss as the models see more tokens.

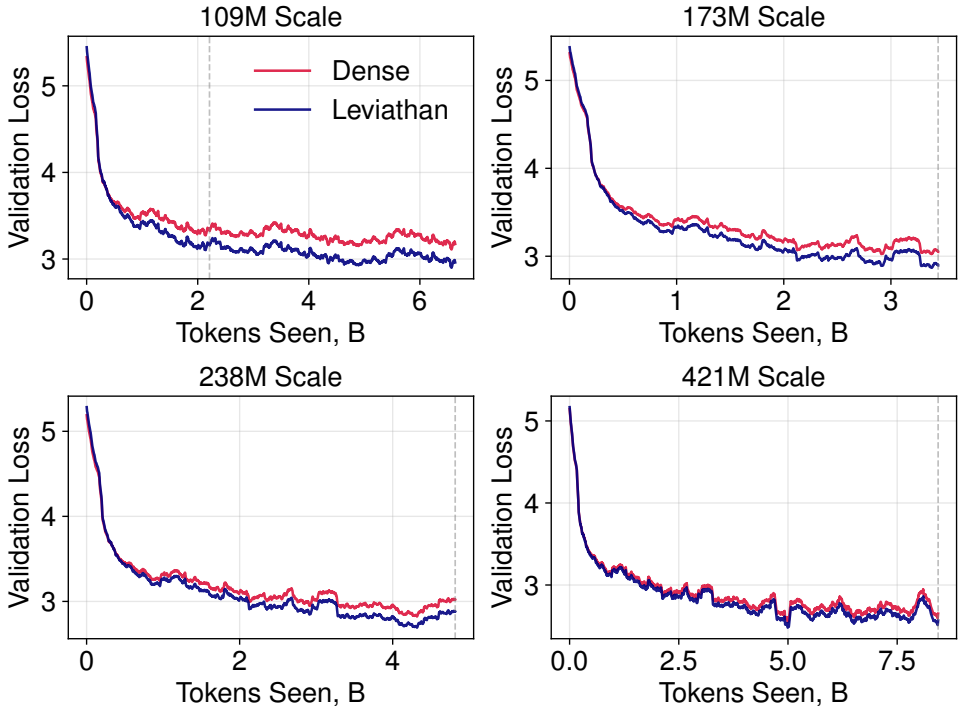

Figure 10. The *iso-parametric* learning curves; tracking validation loss as the models see more tokens.