Fourier Transform Networks: Boundary-Aware Spectral Sequence Modeling Without Attention

Wyvern AI Research

May 2026

Abstract

Transformers dominate modern sequence modeling, but their quadratic attention cost remains a persistent obstacle for long-context efficiency. We study the Fourier Transform Network (FTN), an attention-free causal language model that replaces pairwise attention with dual-branch spectral mixing. FTN combines a local causal convolution branch with a global causal FFT convolution branch; our main architectural variant, `v2_boundary`, further adds branch-specific projections, learned frequency reweighting, and a boundary-aware residual gate on the global path. On TinyStories, we report three main findings. First, additive branch fusion performs better than more expressive gated and concatenative fusion schemes in our setting, suggesting that branch formation may matter more than fusion complexity. Second, a small `v2_boundary` FTN substantially improves over earlier FTN variants and achieves lower validation loss than a matched small scratch GPT-2 baseline. Third, a larger non-matched FTN scaling run improved substantially with additional data and optimization budget, with an 82.4M-parameter FTN reaching a final validation loss of 0.8112. At the same time, qualitative generation remains noticeably weaker than Transformer baselines, indicating that strong next-token loss does not yet translate into equally strong free-form coherence. Taken together, these results position FTN as a credible attention-free sequence modeling direction while motivating further work on representation analysis and generative behavior.

1 Introduction

The Transformer has become the standard architecture for language and sequence modeling because attention provides a flexible mechanism for long-range interaction. Its computational cost, however, scales quadratically with sequence length. This has motivated sustained interest in architectures that preserve strong sequence modeling performance while avoiding explicit attention matrices.

This paper studies the Fourier Transform Network (FTN), a causal sequence model built around spectral mixing rather than attention. FTN uses a dual-branch block consisting of a local branch for short-range structure and a global branch for sequence-wide interaction through causal FFT convolution. Our aim is not merely to introduce another efficient sequence block, but to test whether an attention-free spectral architecture can learn effectively on real text, whether it improves under architectural refinement, and whether it can benefit from additional data and optimization budget.

The resulting picture is neither purely positive nor purely negative. Early FTN variants were weak and often displayed strong low-frequency bias. However, the `v2_boundary` modification materially improved optimization and predictive performance. In the clearest controlled comparison, the small `v2_boundary` model achieved lower validation loss than a matched small scratch GPT-2 baseline. A larger FTN trained with a substantially longer schedule improved further and reached a final validation loss of 0.8112, though this larger result is best interpreted as a non-matched scaling

demonstration rather than a direct baseline comparison. At the same time, open-ended generation tests show that FTN remains noticeably weaker than Transformer baselines in free-form coherence. The present evidence therefore supports FTN as a meaningful research direction, but not yet as a drop-in replacement for Transformer language models.

This scope makes FTN well suited to a first foundation paper in a broader research program. Rather than claiming that FTN has already surpassed Transformers, we focus on four narrower findings:

1. FTN can be trained as a causal language model with global receptive fields and no attention.
2. Boundary-aware global mixing is a major architectural improvement over earlier FTN variants.
3. Branch-internal design appears to matter more than fusion complexity.
4. Strong next-token loss does not automatically imply equally strong open-ended generative coherence.

Contributions. This paper makes the following contributions:

- We introduce and evaluate FTN, an attention-free dual-branch spectral language model based on causal FFT convolution.
- We present the `v2_boundary` FTN variant, which adds branch-specific projections, learned global frequency gains, and a position-dependent residual gate for the global branch.
- We show that additive branch fusion outperforms more expressive fusion schemes in our setting, yielding an unexpected but robust design lesson.
- We show that a small `v2_boundary` FTN achieves lower validation loss than a matched small scratch GPT-2 baseline, and that a larger FTN run can improve substantially when paired with increased data and optimization budget.
- We document a clear mismatch between strong predictive loss and weak open-ended coherence, identifying a concrete limitation and motivating subsequent mechanistic analysis.

2 Related Framing

This work is motivated by the same broad problem that underlies efficient Transformer and attention-alternative architectures: how to preserve long-range sequence modeling ability while reducing the cost of explicit pairwise attention [1, 2, 3]. Related alternatives have included Fourier token mixing [4], state-space sequence models [5], and long convolutional architectures [6]. FTN explores one particular answer to that problem through direct causal sequence mixing in the spectral domain together with a local branch for short-range structure.

The goal of this first paper is narrower than a general claim of superiority over Transformers. Instead, we ask whether a causal spectral architecture can serve as a credible language model, whether architectural changes of the kind implemented in `v2_boundary` materially improve that model, and what kinds of behavior remain missing even when validation loss becomes strong.

3 Model

3.1 Overall architecture

FTN is a causal language model with standard token embeddings, positional embeddings, a stack of FTN blocks, a final normalization layer, and a tied output projection. The core novelty is the FTN block itself.

Each FTN block processes a hidden sequence through two parallel branches:

- a **local branch**, designed to capture short-range dependencies through causal convolution with a limited receptive field;
- a **global branch**, designed to capture sequence-wide structure through causal FFT convolution.

Both branches are fused, added back through a residual connection, and followed by an MLP sublayer.

3.2 Code-level configuration

The implementation used in this paper is defined directly in `modeling_ftn.py`. The FTN configuration object contains

$$(\texttt{vocab\_size}, \texttt{max\_position\_embeddings}, d, N, d_{\text{ff}}, p, K_\ell, K_g, \texttt{fusion\_mode}, \texttt{variant}), \tag{1}$$

along with the boundary-specific parameters `boundary_edge_width`, `boundary_edge_mix`, and `boundary_center_mix`. In the code, d corresponds to `d_model`, N to `num_layers`, d_{ff} to `ffn_dim`, K_ℓ to the local window size, and K_g to the global kernel size.

At the top level, the model computes

$$h^{(0)} = \text{Dropout}\left(E_{\text{tok}}(x) + E_{\text{pos}}(0{:}L-1)\right), \tag{2}$$

then applies N FTN blocks, followed by

$$\text{logits} = W_{\text{lm}}\, \text{LN}_{\text{final}}(h^{(N)}), \tag{3}$$

with tied token embedding weights when `tie_word_embeddings=True`. Training uses standard next-token cross-entropy on shifted logits.

3.3 Causal FFT convolution

The FTN global branch replaces attention with a causal convolution implemented in the frequency domain. In code, the tensor is transposed from shape $[B, L, d]$ to $[B, d, L]$, padded implicitly by choosing

$$n_{\text{fft}} = L + K - 1, \tag{4}$$

and transformed channel-wise with a real FFT. Given an input sequence $x \in \mathbb{R}^{B\times L\times d}$ and a learned depthwise kernel $w \in \mathbb{R}^{d\times K}$, FTN computes

$$\widehat{x} = \text{rFFT}(x), \qquad \widehat{w} = \text{rFFT}(w), \tag{5}$$

$$\widehat{y} = \widehat{x} \odot \widehat{w}, \tag{6}$$

and then

$$y = \text{iRFFT}(\widehat{y})_{[:,:,:L]}, \tag{7}$$

with an optional channel-wise bias added afterward. The truncation to the first L positions is exactly the code path that enforces causal linear convolution rather than circular wraparound. The implementation also includes a direct-reference convolution path used for numerical preflight checks:

$$y_{\text{ref}} = \text{conv1d}\left(\text{pad}(x, K-1), \text{flip}(w)\right), \tag{8}$$

with grouped convolution over channels. This gives global receptive fields with $O(L \log L)$ spectral mixing instead of $O(L^2)$ attention.

When `use_frequency_gain=True`, the code multiplies by a learned positive gain tensor

$$G = \exp(\Theta) \in \mathbb{R}_{>0}^{d \times F}, \tag{9}$$

so that

$$\widehat{y} = \widehat{x} \odot \widehat{w} \odot G. \tag{10}$$

This is the exact mechanism used by the `v2_boundary` global branch to reshape the spectral profile beyond the kernel alone.

3.4 From `v1_add` to `v2_boundary`

The early FTN variant, `v1_add`, used the same broad dual-branch idea but proved weak in both loss and qualitative generation. Diagnostics suggested that the global branch was often oversmoothed, with upper layers concentrating almost all energy in low-frequency bands.

The `v2_boundary` variant introduces three key changes:

1. **Branch-specific input and output projections.** Both local and global branches receive learned pointwise projections before and after convolution, allowing each branch to shape its own representational subspace.

2. **Learned global frequency gains.** The global FFT branch applies a learned positive gain over frequency bins, allowing the model to upweight or downweight different spectral regions rather than inheriting a fixed spectral bias from the kernel alone.

3. **Boundary-aware residual gating.** The global branch mixes its FFT output with a learned residual projection using a position-dependent gate table. Edge positions are initialized to trust the residual path more than the spectral path, while central positions are initialized to trust the spectral path more strongly.

The actual FTN block implemented in code is close to the following equations. Let h denote the layer-normalized input:

$$h = \text{LN}_{\text{in}}(x). \tag{11}$$

The local branch is

$$h_\ell^{\text{in}} = W_\ell^{\text{in}} h, \tag{12}$$

$$h_\ell^{\text{pre}} = W_\ell^{\text{out}}\, \text{CausalFFTConv}_{K_\ell}(h_\ell^{\text{in}}), \tag{13}$$

$$h_\ell = \text{LN}_\ell(h_\ell^{\text{pre}}). \tag{14}$$

The global branch is

$$h_g^{\text{in}} = W_g^{\text{in}} h, \tag{15}$$

$$h_g^{\text{fft}} = W_g^{\text{out}}\, \text{CausalFFTConv}_{K_g}^{\text{gain}}(h_g^{\text{in}}). \tag{16}$$

For `v2_boundary`, the code then slices a learned table of position logits,

$$A \in \mathbb{R}^{L_{\max} \times d}, \tag{17}$$

applies a sigmoid gate,

$$\Gamma_{1:L} = \sigma(A_{1:L}) \in (0,1)^{L \times d}, \tag{18}$$

and mixes FFT and residual paths elementwise:

$$h_g^{\text{pre}} = \Gamma \odot h_g^{\text{fft}} + (1 - \Gamma) \odot W_g^{\text{res}} h. \tag{19}$$

The normalized global branch is

$$h_g = \text{LN}_g(h_g^{\text{pre}}). \tag{20}$$

The boundary gate is initialized asymmetrically by position. If w_e is the edge width, then positions in the first and last w_e tokens are initialized toward `boundary_edge_mix`, while the center positions are initialized toward `boundary_center_mix`. In the implementation used here, those defaults are 0.2 at the edges and 0.8 in the middle. This was specifically introduced to reduce brittle edge behavior and avoid treating causal sequence modeling as if it were naturally periodic.

3.5 Fusion

We evaluated three fusion schemes for combining the local and global branches:

- additive fusion,
- concatenation followed by projection,
- learned gated fusion.

Unexpectedly, additive fusion proved most effective in our setting. This result suggests that once the branches themselves are well-formed, more complex routing may not be the main bottleneck. We return to this observation in Section 6.

More precisely, the fusion module in code accepts one of three modes:

$$\texttt{add}: \quad h_f = h_\ell + h_g, \tag{21}$$

$$\texttt{concat}: \quad h_f = W_c[h_\ell; h_g], \tag{22}$$

$$\texttt{gate}: \quad z = \sigma(W_z[h_\ell; h_g] + b_z), \qquad h_f = z \odot h_\ell + (1 - z) \odot h_g. \tag{23}$$

In all cases, the result is passed through an output projection. The block then applies residual addition and a feed-forward network:

$$x' = x + \text{Dropout}(W_o h_f), \tag{24}$$

$$\text{FTNBlock}(x) = x' + \text{MLP}(\text{LN}_{\text{mlp}}(x')). \tag{25}$$

3.6 Implementation-faithful block summary

To make the connection to the actual model code explicit, one FTN block in `modeling_ftn.py` is structurally:

```
hidden = input_norm(x)
local_hidden = local_in_proj(hidden)
local_branch = local_norm(local_out_proj(local_conv(local_hidden)))
global_hidden = global_in_proj(hidden)
global_fft_out = global_out_proj(global_conv(global_hidden))
gate = sigmoid(boundary_gate_table[:seq_len])
global_branch = global_norm(gate * global_fft_out
                            + (1 - gate) * global_residual_proj(hidden))
fused = fusion(local_branch, global_branch)
x = x + residual_dropout(fused)
x = x + mlp(mlp_norm(x))
```

This is not merely an implementation detail; it defines the actual object of study in this paper. In particular, the learned frequency gains, boundary gate table, and residual-mixing path are all present in the deployed code and in the published checkpoints.

4 Experimental Setup

4.1 Data and tokenization

We trained and evaluated on TinyStories using GPT-2 tokenization and packed causal blocks of length 256. The core small-scale comparison used a fixed split of the first 1000 training stories and first 200 validation stories. The largest FTN run used the first 2000 training stories and first 400 validation stories, together with a longer training schedule. This larger run is therefore best interpreted as a scaling demonstration rather than a perfectly matched baseline comparison.

4.2 Model variants

We focus on the following models:

- **FTN `v1_add`**: the earlier additive fusion baseline;
- **FTN `v2_boundary` small**: 17.44M parameters;
- **FTN `v2_boundary` large40m**: 50.41M parameters;
- **FTN `v2_boundary` xlarge80m**: 82.41M parameters;
- **GPT-2 scratch small**: 16.09M parameters.

The small matched comparison between FTN and GPT-2 is the fairest baseline result in this paper. The larger FTN runs are reported primarily to establish scaling behavior and diagnose where scaling helps or fails.

4.3 Concrete FTN settings

The main FTN configurations used in this paper correspond directly to the code-level settings:

- **small**: $d = 256$, $N = 4$, $d_{\text{ff}} = 1024$, $K_\ell = 32$, $K_g = 256$;
- **large40m**: $d = 512$, $N = 6$, $d_{\text{ff}} = 2048$, $K_\ell = 32$, $K_g = 256$;

- **xlarge80m**: $d = 640$, $N = 8$, $d_{\text{ff}} = 2560$, $K_\ell = 32$, $K_g = 256$.

All FTN models use the same causal language-model objective, tied token embeddings, dropout-based regularization, and the `v2_boundary` block definition unless otherwise noted.

4.4 Training setup

All main experiments used the same basic training loop implemented in `train.py`. Models were trained with AdamW, weight decay 0.01, gradient clipping at 1.0, and seed 42. The code does not implement mixed-precision training or autocasting, so all reported experiments were run in standard float32. All main runs reported in this paper executed on CPU; the training logs record `device=cpu` for the small, large40m, and xlarge80m experiments.

The small matched FTN and GPT-2 scratch runs used the `small` preset with batch size 4, gradient accumulation 4, learning rate 3×10^{-4}, and 10 epochs on `train[:1000]` / `validation[:200]`. The `large40m` run used batch size 2, gradient accumulation 8, learning rate 2×10^{-4}, and 10 epochs on the same data split. The `xlarge80m` scaling run used batch size 2, gradient accumulation 8, learning rate 2×10^{-4}, and 14 epochs on the larger `train[:2000]` / `validation[:400]` split. Early stopping support exists in the code with patience 2, although the main FTN runs discussed here completed their scheduled epochs.

4.5 Diagnostics

In addition to validation loss and perplexity, we tracked:

- branch ablations (`full`, `local_only`, `global_only`);
- boundary gate statistics for edge and middle positions;
- spectral low/mid/high band energy for the global branch;
- qualitative free-form generation on unseen prompts.

5 Results

5.1 Fusion ablation

Table 1 reports the early fusion results. The simple additive FTN clearly outperformed the gated fusion version, despite the latter having greater expressive flexibility.

Table 1: Early FTN fusion comparison on the small setup.

Model	Params (M)	Val loss	Perplexity	Accuracy
FTN `v1_add`	15.60	5.4527	233.38	0.2550
FTN gated fusion	16.13	17.8412	5.60e7	0.1281

This was one of the most unexpected findings in the project. A learned gate or concatenative fusion might have seemed like the natural choice for combining two distinct branches, yet the opposite occurred. The most plausible interpretation is that FTN's main difficulty was not branch routing but branch formation: once the branch representations become useful, addition is sufficient.

5.2 Small matched comparison

The most important controlled result is the matched small FTN vs. small GPT-2 comparison. Table 2 summarizes the outcome.

Table 2: Matched small-scale comparison on TinyStories.

Model	Params (M)	Val loss	Perplexity	Accuracy	Tok/s
GPT-2 scratch small	16.09	3.5907	36.26	0.3407	2642.25
FTN `v1_add`	15.60	5.4527	233.38	0.2550	5522.5
FTN `v2_boundary` small	17.44	2.4271	11.33	0.7150	1327.85

This is the strongest baseline result in the paper: the small `v2_boundary` FTN achieved lower validation loss than the matched small scratch GPT-2 model, while also improving substantially over the earlier FTN variant.

The branch diagnostics support the view that `v2_boundary` is doing meaningful work in the global branch rather than merely masking it. For the small model, the full branch-mode loss was 2.4271, compared with 6.1947 for `global_only` and 9.9058 for `local_only`. This indicates that both branches contribute, with the global path carrying much of the useful predictive structure.

5.3 Scaling behavior

We next trained larger `v2_boundary` models. The 50.41M-parameter `large40m` model was worse than the small model, while the 82.41M-parameter `xlarge80m` model improved dramatically under a substantially larger training setup. The result is summarized in Table 3.

Table 3: FTN scaling results.

Model	Params (M)	Val loss	Perplexity	Total train time
FTN `v2_boundary` small	17.44	2.4271	11.33	10,008 s
FTN `v2_boundary` large40m	50.41	3.8706	47.97	78,038 s
FTN `v2_boundary` xlarge80m	82.41	0.8112	2.25	613,724 s

The non-monotonic jump from small to `large40m` is important. We do not interpret that dip as evidence that FTN necessarily fails to scale. Instead, the most likely explanation is recipe mismatch: the 40M-class run appears under-optimized relative to its size, while the much longer and larger-budget `xlarge80m` run shows that FTN can improve substantially when the training setup is scaled accordingly.

Supporting this interpretation, the `xlarge80m` model not only achieved very low loss, but also showed healthier global branch statistics than earlier FTN variants. Its average global low-band energy remained high but no longer collapsed almost entirely into the lowest frequencies, and its `global_only` loss of 1.7916 was vastly better than the corresponding ablations in smaller and earlier FTN models.

5.4 Qualitative generation

Despite the strong predictive loss of the largest FTN, qualitative open-ended generation still lags behind Transformer baselines. In held-out prompt completions, the small GPT-2 scratch baseline

usually produced more sentence-like and narratively plausible continuations, while FTN outputs often degraded into lexical jumps, repetition bursts, or token-soup behavior.

The largest FTN was clearly better than the smaller FTN variants, but it still did not reliably produce coherent free-form continuations. This gap is central to the interpretation of our results: FTN is learning a powerful predictive structure, but that structure is not yet translating cleanly into robust open-ended language generation.

6 Discussion

We highlight three lessons from these experiments.

1. Branch quality mattered more than fusion complexity. The additive fusion result is one of the clearest surprises in the project. If the main challenge were dynamic routing between local and global features, gated fusion might have helped. Instead, the data suggest that FTN's bottleneck may have lain earlier: the model needed healthier branch-internal representations more than a more complex fusion rule.

2. Boundary-aware global mixing was a meaningful architectural step. The jump from `v1_add` to `v2_boundary` was not cosmetic. The new branch projections, learned frequency gains, and position-dependent residual mixing changed optimization behavior enough to produce a meaningful small-model improvement over the scratch GPT-2 baseline. This provides evidence that the poor early FTN results were not simply proof that the architecture family was doomed.

3. Low next-token loss is not the whole story. The largest FTN reaches a remarkably low validation loss, but qualitative generation remains weaker than Transformer baselines. This mismatch is not a minor side note; it is one of the most scientifically useful findings in the paper. It suggests that the representations FTN learns are predictive in a strong statistical sense, but still incomplete from the standpoint of long-form semantic and discourse coherence.

7 Limitations

This first paper has several important limitations.

- The broadest claim we can support is that FTN is a promising attention-free language modeling direction under constrained compute, not that it is generally superior to Transformers.
- Our cleanest baseline comparison is the matched small FTN vs. small scratch GPT-2 setup. Larger FTN scaling results are strong, but they are not yet paired with equally matched larger Transformer baselines.
- TinyStories is a useful but limited corpus. It rewards strong next-token prediction on relatively regular narrative text and is not by itself a sufficient test of general sequence modeling.
- Qualitative coherence still lags behind Transformer baselines, even when FTN reaches strong validation loss.
- Several training setup details should be reported more completely before formal publication, including optimizer settings, learning rate schedule, batch size, hardware, precision, and random seed coverage.

8 Future Work

This paper opens several immediate directions for follow-up work.

1. **Representation analysis.** We are particularly interested in sparse autoencoder analysis across FTN and Transformer baselines to determine what FTN learns well and what remains missing.
2. **Matched larger baselines.** A larger matched scratch GPT-2 comparison remains essential for drawing stronger comparative conclusions.
3. **Broader sequence tasks.** To move from this foundation paper toward a broader FTN program, evaluation should expand beyond TinyStories to more order-sensitive, retrieval-heavy, and algorithmic sequence tasks.
4. **Generation-focused improvements.** The gap between predictive loss and coherent generation suggests that FTN may need additional mechanisms for richer semantic continuity or more stable long-range token planning.

9 Conclusion

FTN is not yet a drop-in replacement for Transformers. However, the results in this paper show that an attention-free spectral language model can be more promising than early weak baselines suggested. The `v2_boundary` architecture materially improves learning, a small FTN can achieve lower validation loss than a matched small scratch GPT-2 baseline, and a larger FTN can reach strong predictive performance when paired with increased data and optimization budget.

At the same time, the architecture still falls short in free-form generative coherence, and that shortfall is important rather than embarrassing. It turns FTN from a narrow benchmark curiosity into a genuine research problem: a model family that can clearly learn useful predictive structure, yet still seems to represent sequence structure differently from Transformers. That is exactly the kind of result worth a first paper.

AI Assistance Disclosure

OpenAI Codex was used as a coding assistant during implementation, debugging, refactoring, and experiment workflow development. The project began with access to GPT-5.5 and later used GPT-5.4 as the effective model once the working context was sufficiently established. All research claims, experimental design decisions, analysis, interpretations, and final manuscript content were reviewed, revised, and approved by the human authors, who take responsibility for the work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

[2] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[3] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking Attention with Performers. In *International Conference on Learning Representations*, 2021.

[4] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. FNet: Mixing Tokens with Fourier Transforms. In *Proceedings of NAACL-HLT*, 2022.

[5] Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022.

[6] Michael Poli, Stefano Massaroli, Eric Q. Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Christopher Ré, and Stefano Ermon. Hyena Hierarchy: Towards Larger Convolutional Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.