Transformer Encoder Frankenstein: Library, CLI, and Research Grounded Design Notes

Erick F. Merino M.
This work is not affiliated
erickfmm@gmail.com

February 2026

Abstract

This document reframes Transformer Encoder Frankenstein as a configuration-driven software toolkit and research workbench. We present the CLI-first workflow, mathematical descriptions of supported attention and sequence-mixer families, optimizer families mapped to the training schema, deployment quantization mechanics, and SBERT downstream tasks. We also include pseudocode and summary tables to make design trade-offs explicit for implementation and experimentation.

1 Introduction

Transformer systems are now a production concern as much as a modeling concern [32, 12]. In practice, users need a single toolchain that can: (i) define training configurations with strict contracts, (ii) train with multiple optimizer and mixer families, (iii) deploy quantized checkpoints, and (iv) run sentence-embedding workflows inspired by SBERT [25].

The project command surface is:

`frankestein-transformer`

with subcommands: `train`, `deploy`, `quantize`, `infer`, `sbert-train`, `sbert-infer`.

2 Configuration-Centric Architecture

The authoritative contract is `src/training/configs/schema.yaml`. It enforces three top-level objects:

- `model_class`
- `model`
- `training`

The `model.layer_pattern` supports:

$$\{\texttt{retnet}, \texttt{mamba}, \texttt{ode}, \texttt{titan\_attn}, \texttt{standard\_attn}, \texttt{sigmoid\_attn}\}$$

which corresponds to current attention and sequence-mixer literature [29, 14, 40, 2, 24, 32].

The `training.optimizer.optimizer_class` supports a broad optimizer family: `adamw`, `adafactor`, `radam`, `adan`, `adopt`, `ademamix`, `mars_adamw`, `cautious_adamw`, `schedulefree_adamw`, `lion`, `sophia`, `prodigy`, `muon`, `turbo_muon`, `shampoo`, `soap`, and others.

2.1 Schema Scope and Validation Rules

The schema is strict: top-level and nested objects set `additionalProperties: false`. This guarantees that unknown keys fail fast instead of being silently ignored. The `training.optimizer.parameters` object is additionally constrained by optimizer-specific prefix rules through `allOf`+`if/then` pattern checks.

Normalization values currently accepted by schema are:

$$\texttt{norm\_type} \in \{\texttt{layer\_norm}, \texttt{dynamic\_tanh}, \texttt{derf}\}$$

Thus, `rms_norm` is **not** a valid schema value in the current contract.

2.2 Complete Model Feature Inventory

Field	Type/Range	Meaning
`vocab_size`	int ≥ 1	Vocabulary size.
`hidden_size`	int ≥ 1	Hidden dimension.
`num_layers`	int ≥ 1	Physical layer count.
`num_loops`	int ≥ 1	Logical loop count (looped blocks).
`num_heads`	int ≥ 1	Attention heads.
`retention_heads`	int ≥ 1	Retention heads for RetNet-style mixers.
`num_experts`	int ≥ 1	MoE expert count.
`top_k_experts`	int ≥ 1	Top-k expert routing in MoE.
`dropout`	float $[0, 1]$	Global dropout.
`layer_pattern`	array enum	Ordered block list: `retnet`, `mamba`, `ode`, `titan_attn`, `standard_attn`, `sigmoid_attn`.
`ode_solver`	enum	`rk4` or `euler`.
`ode_steps`	int ≥ 1	ODE integration steps.
`use_bitnet`	bool	Enable low-bit BitLinear path.
`norm_type`	enum	`layer_norm`, `dynamic_tanh`, `derf`.
`use_factorized_embedding`	bool	Enable factorized embeddings.
`factorized_embedding_dim`	int ≥ 1	Reduced embedding dimension for factorization.
`use_embedding_conv`	bool	Enable Conv1d over embedding stream.
`embedding_conv_kernel`	int ≥ 1	Conv1d kernel size.
`hope_base`	float ≥ 0	HoPE base value (optional in schema).
`hope_damping`	float ≥ 0	HoPE damping (optional in schema).
`use_hope`	bool	Apply HoPE in `titan_attn`.
`use_moe`	bool	Enable MoE FFN routing path.
`ffn_hidden_size`	int ≥ 1	FFN intermediate width.
`ffn_activation`	enum	`silu` or `gelu`.

Looped depth induced by schema is:

$$L_{\text{logical}} = \texttt{num\_layers} \times \texttt{num\_loops}$$

which is the configuration-level definition of looped blocks.

2.3 Complete Training Feature Inventory

Field	Type/Range	Meaning
`batch_size`	int ≥ 1	Loader batch size.
`dataloader_workers`	int ≥ 0	PyTorch dataloader workers.

Field	Type/Range	Meaning
`max_length`	int ≥ 1	Sequence length cap.
`mlm_probability`	float $[0, 1]$	MLM masking probability.
`max_samples`	int ≥ 1	Maximum streamed samples.
`dataset_batch_size`	int ≥ 1	Internal streaming dataset chunk size.
`num_workers`	int ≥ 0	Streaming dataset workers.
`cache_dir`	string	Dataset cache directory.
`local_parquet_dir`	string	Optional local parquet path.
`prefer_local_cache`	bool	Prefer local cache when available.
`stream_local_parquet`	bool	Stream from local parquet mode.
`use_amp`	bool	Mixed precision toggle.
`gradient_accumulation_steps`	int ≥ 1	Effective batch through accumulation.
`optimizer`	object	Contains `optimizer_class` and prefixed `parameters`.
`scheduler_total_steps`	int ≥ 1	Scheduler horizon.
`scheduler_warmup_ratio`	float $[0, 1]$	Warmup ratio.
`scheduler_type`	enum	`cosine`, `constant`, `linear_warmup_then_constant`.
`grad_clip_max_norm`	float ≥ 0	Global norm clipping threshold.
`inf_post_clip_threshold`	float ≥ 0	Exploding-gradient guard threshold after clipping.
`max_nan_retries`	int ≥ 0	Retry budget for NaN/Inf instability.
`checkpoint_every_n_steps`	int ≥ 1	Rolling checkpoint frequency.
`max_rolling_checkpoints`	int ≥ 1	Number of rolling checkpoints to keep.
`num_best_checkpoints`	int ≥ 1	Number of best checkpoints tracked.
`nan_check_interval`	int ≥ 1	NaN/Inf check cadence.
`log_gradient_stats`	bool	Enable gradient statistics logging.
`gradient_log_interval`	int ≥ 1	Gradient logging cadence.
`csv_log_path`	string	Step-level CSV output path.
`csv_rotate_on_schema_change`	bool	Rotate CSV if logging schema changes.
`gpu_metrics_backend`	enum	`nvml` or `none`.
`nvml_device_index`	int ≥ 0	Device index for NVML telemetry.
`enable_block_grad_norms`	bool	Include per-block gradient norm telemetry.
`telemetry_log_interval`	int ≥ 1	Heavy telemetry interval (optimizer steps).
`use_galore`	bool	Enable GaLore strategy.
`galore_rank`	int ≥ 1	GaLore low-rank projection dimension.
`galore_update_interval`	int ≥ 1	Projection refresh interval.
`galore_scale`	float ≥ 0	Gradient scaling in projected space.
`galore_max_dim`	int ≥ 1	Maximum tensor dimension for GaLore projection.

2.4 Optimizer Prefix Contract (Full)

Supported `optimizer_class` values are: `sgd_momentum`, `adamw`, `adafactor`, `galore_adamw`, `prodigy`, `lion`, `sophia`, `muon`, `turbo_muon`, `radam`, `adan`, `adopt`, `ademamix`, `mars_adamw`, `cautious_adamw`, `lamb`, `schedulefree_adamw`, `shampoo`, `soap`.

Shared per-group suffix families (all prefixed by optimizer name) are:

- LR groups: `lr_embeddings`, `lr_norms`, `lr_ode`, `lr_retnet`, `lr_mamba`, `lr_attention`, `lr_other`
- Weight decay groups: `wd_embeddings`, `wd_norms`, `wd_ode`, `wd_retnet`, `wd_mamba`, `wd_attention`, `wd_other`
- Beta groups: `betas_embeddings`, `betas_norms`, `betas_ode`, `betas_retnet`, `betas_mamba`, `betas_attention`, `betas_other`

Algorithm 1 Schema-Driven Training Step with Stability Controls

Require: Batch stream, config C

Initialize retry counter $r \leftarrow 0$
for each optimizer step **do**
  Accumulate gradients for $K = C$.`gradient_accumulation_steps` micro-batches
  Apply global norm clipping with $\tau = C$.`grad_clip_max_norm`
  if post-clip gradient exceeds C.`inf_post_clip_threshold` or NaN/Inf detected **then**
    if $r < C$.`max_nan_retries` **then**
      restore safe state / skip step; $r \leftarrow r + 1$
      continue
    else
      stop training with failure state
    end if
  end if
  run optimizer step selected by `optimizer_class`
  update scheduler (`cosine`, `constant`, or `linear_warmup_then_constant`)
  if step mod `checkpoint_every_n_steps`$= 0$ **then**
    save rolling checkpoint and prune to `max_rolling_checkpoints`
  end if
  update best checkpoints up to `num_best_checkpoints`
  emit CSV + telemetry following `gradient_log_interval` and `telemetry_log_interval`
end for

- Epsilon groups: `eps_embeddings`, `eps_norms`, `eps_ode`, `eps_retnet`, `eps_mamba`, `eps_attention`, `eps_other`

Optimizer-specific global suffixes:

- `sgd_momentum`: `momentum`, `nesterov`
- `adafactor`: `beta2_decay`, `clip_threshold`, `eps1`, `eps2`
- `galore_adamw`: `rank`, `update_proj_gap`
- `prodigy`: `d_coef`
- `sophia`: `rho`, `update_k`
- `muon` / `turbo_muon`: `momentum`, `nesterov`, `ns_steps`, `ns_eps`
- `cautious_adamw`: `cautious_clip`

All other classes in the list above accept only prefixed shared groups.

2.5 Training Safety and Runtime Semantics

Schema-level safety features include accumulation, clipping, post-clip explosion checks, and NaN retries:

$$g_{\text{acc}} = \frac{1}{K}\sum_{i=1}^{K} g_i, \quad K = \texttt{gradient\_accumulation\_steps}$$

$$g_{\text{clip}} = g_{\text{acc}} \cdot \min\left(1, \frac{\tau}{\|g_{\text{acc}}\|_2 + \epsilon}\right), \quad \tau = \texttt{grad\_clip\_max\_norm}$$

then overflow guards use `inf_post_clip_threshold` and retry logic bounded by `max_nan_retries`.

3 Normalization Variants: RMSNorm, Dynamic Tanh, and Dynamic Erf

Normalization and normalization-alternatives are central to training stability and throughput in transformer-like systems. Based on ArXiv sources, the three relevant formulations are:

3.1 RMSNorm

RMSNorm removes mean-centering and only rescales by root mean square magnitude [?]:

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}, \qquad y_i = \gamma_i \frac{x_i}{\mathrm{RMS}(x)}$$

Compared with LayerNorm, RMSNorm is computationally simpler (no subtraction of feature mean) and is often used when reducing normalization overhead is important.

3.2 Dynamic Tanh (DyT)

Dynamic Tanh proposes replacing explicit normalization with a bounded elementwise map [43]:

$$\mathrm{DyT}(x) = \tanh(\alpha x)$$

where α is learned. The core idea is that bounded nonlinear contraction can provide stable signal scaling without explicitly computing per-token normalization statistics.

3.3 Dynamic Erf (Derf)

Derf extends the same normalization-free direction by using an error-function based map [8]:

$$\mathrm{Derf}(x) = \mathrm{erf}(\alpha x + s)$$

with learnable scale/shift. Reported results in the cited work indicate stronger performance than DyT and common normalization baselines across multiple domains.

3.4 Schema Implications

Current configuration contract in this repository allows:

$$\texttt{norm\_type} \in \{\texttt{layer\_norm}, \texttt{dynamic\_tanh}, \texttt{derf}\}$$

so DyT and Derf are directly available in schema-driven runs, while RMSNorm is not currently an accepted enum value and would require code/schema extension.

Method	Formula	Stats Needed	Notes
RMSNorm	$y_i = \gamma_i x_i / \sqrt{\frac{1}{d}\sum_j x_j^2 + \epsilon}$	RMS only	Lower overhead than LayerNorm; widely used normalization baseline [?].
Dynamic Tanh	$\tanh(\alpha x)$	none	Normalization-free bounded transform; simple drop-in replacement direction [43].
Dynamic Erf (Derf)	$\mathrm{erf}(\alpha x + s)$	none	Normalization-free alternative designed to improve over DyT [8].

4 Attention and Sequence-Mixer Families

4.1 Standard Attention

Given projected matrices (Q, K, V):

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

This is the baseline mechanism for content routing [32].

4.2 Sigmoid Attention

Sigmoid attention removes row-wise probability normalization:

$$\text{SigmoidAttn}(Q, K, V) = \sigma\left(\frac{QK^\top}{\sqrt{d_k}} + b\right) V$$

and has different training stability requirements, often with additional normalization [24].

4.3 Retentive Formulation

RetNet uses retention with decay matrix D:

$$\text{Retention}(Q, K, V) = \left(QK^\top \odot D\right) V$$

with recurrent form:

$$S_n = \gamma S_{n-1} + k_n^\top v_n, \qquad o_n = q_n S_n$$

enabling low-cost recurrent inference [29, 36].

4.4 Selective SSM (Mamba)

Discrete selective state-space recurrence is:

$$h_t = \bar{A}_t h_{t-1} + \bar{B}_t x_t, \qquad y_t = C_t h_t$$

where $(\bar{A}_t, \bar{B}_t, C_t)$ depend on input, preserving linear-time scaling with hardware-aware scan [14, 17].

4.5 ODE-style Continuous Updates

Continuous-depth framing:

$$\frac{dh(t)}{dt} = f_\theta(h(t), t)$$

with practical RK integrators for discrete execution [40].

4.6 Test-time Memory (Titans)

A memory-augmented update can be written:

$$M_t = (1 - \alpha_t) M_{t-1} + S_t, \qquad S_t = \eta_t S_{t-1} - \theta_t \nabla \ell(M_{t-1}; x_t)$$

to adapt memory at inference time [2, 1].

Algorithm 2 Pattern-Driven Mixer Forward (Conceptual)

Require: Hidden states H, pattern P, layer index ℓ

```
m ← P[ℓ mod |P|]
if m = standard_attn then
    H ← softmax-attention(H)
else if m = sigmoid_attn then
    H ← sigmoid-attention(H)
else if m = retnet then
    H ← retention(H)
else if m = mamba then
    H ← selective-ssm(H)
else if m = ode then
    H ← rk-step(H)
else
    H ← memory-augmented-attn(H)
end if
return H
```

5 Optimizer Families and Training Dynamics

5.1 Core Adaptive Form

Many supported optimizers share moment tracking:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

followed by preconditioned updates (e.g., AdamW) [21].

5.2 Examples from the Supported Set

- **RAdam**: variance rectification for early-step instability [20].
- **Adan**: adaptive Nesterov momentum for faster convergence [35].
- **ADOPT**: modified Adam order yielding stronger convergence guarantees [31].
- **AdEMAMix**: dual-EMA history mixing [23].
- **MARS**: variance reduction in preconditioned optimization [39].
- **Cautious optimizers**: sign-consistent masking of momentum updates [18].
- **Schedule-free**: remove explicit scheduler dependence [11].
- **Shampoo/SOAP**: matrix preconditioning families [16, 33].
- **Adafactor/GaLore**: memory reduction via factorization or low-rank projection [27, 42].
- **Prodigy/Lion/Sophia**: parameter-free adaptation, sign momentum, and clipped second-order scaling [22, 9, 19].
- **Muon/Turbo-Muon**: orthogonality-oriented updates with acceleration [28, 3].

Algorithm 3 Schema-Routed Optimizer Step (Conceptual)

Require: Parameters θ, gradients g, optimizer class c, parameter map Π

Read optimizer-specific hyperparameters from prefixed keys in Π
if $c =$ `adamw` **then**
    apply AdamW step [21]
else if $c =$ `radam` **then**
    apply rectified adaptive step [20]
else if $c =$ `adan` **then**
    apply Adan three-moment step [35]
else if $c =$ `adopt` **then**
    apply ADOPT update ordering [31]
else if $c =$ `galore_adamw` **then**
    project gradients to low-rank subspace then step [42]
else
    dispatch to selected optimizer implementation
end if
return updated θ

6 Quantization and Deployment

The deploy stack uses ternary weight packing plus INT8 activation quantization for efficient artifacts.

6.1 Ternary Quantization

Given weight tensor W, a practical scaling is:

$$s = \mathrm{mean}(|W|), \qquad \tilde{W} = \mathrm{clip}\left(\mathrm{round}\left(\frac{W}{s}\right), -1, 1\right)$$

which approximates BitNet-style low-bit updates [34]. The packed mapping uses two bits per weight symbol for storage efficiency.

6.2 Activation Quantization

For activations x:

$$q = \mathrm{round}\left(x \cdot \frac{127}{\max(|x|) + \epsilon}\right), \qquad q \in [-128, 127]$$

with dequantization $x \approx q/\alpha$.

6.3 Size Estimates

For N parameters:

$$\text{FP32 size} \approx 4N, \quad \text{FP16 size} \approx 2N, \quad \text{1.58-bit size} \approx \frac{1.58}{8}N$$

before metadata and packing overhead. This aligns with lightweight deployment goals [26, 4].

7 SBERT Downstream Tasks

Sentence embedding is built on Siamese-style training [25]. For sentence pair (s_1, s_2) with embeddings (e_1, e_2):

$$\cos(e_1, e_2) = \frac{e_1^\top e_2}{\|e_1\|\|e_2\|}$$

Algorithm 4 SBERT Inference Mode Router

Require: mode m, model E, inputs X

if $m =$ `similarity` **then**
    return $\cos(E(x_1), E(x_2))$
else if $m =$ `search` **then**
    return top-k by dot-product/cosine against corpus embeddings
else if $m =$ `cluster` **then**
    return clustering labels over $E(X)$
else
    return serialized embeddings $E(X)$
end if

and regression-style cosine loss:

$$\mathcal{L}_{\cos} = (\cos(e_1, e_2) - y)^2$$

with $y \in [-1, 1]$ in this pipeline.

Supported downstream modes:

- **Similarity**: pairwise score between two sentences.
- **Search**: top-k nearest neighbors over a corpus.
- **Cluster**: grouping embeddings (e.g., k-means).
- **Encode**: persistent embedding export for later retrieval.

8 Summary Tables

8.1 Attention and Sequence-Mixer Summary

Type	Core Equation	Train	Infer	Notes
Standard Attention	$\mathrm{softmax}(QK^\top/\sqrt{d_k})V$	$\mathcal{O}(n^2 d)$	$\mathcal{O}(n)$/step	Baseline expressive global routing [32].
Sigmoid Attention	$\sigma(QK^\top/\sqrt{d_k} + b)V$	$\mathcal{O}(n^2 d)$	$\mathcal{O}(n)$/step	Element-wise gating; often needs stabilization norm [24].
RetNet	$(QK^\top \odot D)V$	$\mathcal{O}(n^2 d)$ or chunkwise	$\mathcal{O}(1)$/step	Parallel/recurrent dual form with decay retention [29].
Mamba	$h_t = \bar{A}_t h_{t-1} + \bar{B}_t x_t$	$\mathcal{O}(nd)$	$\mathcal{O}(1)$/step	Selective state-space with hardware-aware scan [14].
ODE-style block	$\frac{dh}{dt} = f_\theta(h, t)$	solver-dependent	solver-dependent	Continuous-depth interpretation; RK integration [40].
Titans memory	$M_t = (1-\alpha_t)M_{t-1} + S_t$	approx. $\mathcal{O}(nd)$	retrieval-centric	Test-time memory updates with surprise-driven dynamics [2].

8.2 Optimizer Summary

Optimizer	Family	State Cost	Key Idea	Ref
AdamW	Adaptive first/second moment	High	Decoupled weight decay baseline	[21]
RAdam	Adaptive variance-corrected	High	Rectifies early adaptive variance	[20]
Adan	Momentum + variance reduction	High	Nesterov-style adaptive update	[35]
ADOPT	Adam variant	High	Reordered updates with improved convergence guarantees	[31]
AdEMAMix	Multi-EMA adaptive	High	Mixes short and long horizon EMAs	[23]
MARS	Variance-reduced preconditioned	High	Recursive momentum correction	[39]
Cautious AdamW	Masked momentum	High	Apply updates only on sign-consistent directions	[18]
Schedule-free AdamW	Scheduler-free adaptive	High	Remove explicit LR schedule dependence	[11]
Adafactor	Memory-efficient adaptive	Medium	Factorized second moments for matrix tensors	[27]
GaLore AdamW	Low-rank gradient projection	Medium	Optimize in projected low-rank gradient space	[42]
Prodigy	Parameter-free adaptation	Medium	Distance-adaptive step calibration	[22]
Lion	Sign momentum	Low	Momentum sign update, reduced state	[9]
Sophia	Approx. second-order	Medium	Diagonal Hessian preconditioning with clipping	[19]
Shampoo	Matrix preconditioner	High	Kronecker-structured second-order statistics	[16]
SOAP	Shampoo + Adam basis	High	Adam-like tracking in preconditioner eigenbasis	[33]
Muon	Orthogonality-based	Medium	Orthogonalized matrix updates	[28]
Turbo-Muon	Accelerated orthogonalization	Medium	Preconditioned Newton-Schulz speedup	[3]

9 Discussion

From an engineering perspective, the toolkit couples modern research ideas with reproducible interfaces:

- Explicit schema contracts lower configuration ambiguity.
- Multiple attention/mixer families let users tune for context length, latency, and memory.
- Broad optimizer support enables controlled studies over convergence and stability.
- Quantized deployment reduces artifact size and improves portability.
- SBERT workflows cover practical retrieval and semantic similarity tasks.

The design also aligns with multilingual and compact-model directions in the literature [7, 30, 5, 13].

10 Conclusion

Transformer Encoder Frankenstein is positioned as a practical experimentation platform: a strict configuration schema, extensible optimizer and attention families, deploy-time quantization, and sentence embedding workflows in one CLI. This makes it suitable for both rapid iteration and reproducible model operations.

References

[1] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, . URL http://arxiv.org/abs/2501.00663.

[2] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, . URL https://arxiv.org/abs/2501.00663. Version Number: 1.

[3] Thibaut Boissin, Thomas Massena, Franck Mamalet, and Mathieu Serrurier. Turbo-muon: Accelerating orthogonality-based optimization with pre-conditioning. URL https://arxiv.org/abs/2512.04632. Version Number: 1.

[4] Riccardo Bravin, Massimo Pavan, Hazem Hesham Yousef Shalby, Fabrizio Pittorino, and Manuel Roveri. EmbBERT: Attention under 2 MB memory. URL http://arxiv.org/abs/2502.10001.

[5] Minh Duc Bui, Fabian David Schmidt, Goran Glavaš, and Katharina von der Wense. Knowledge distillation vs. pretraining from scratch under a fixed (computation) budget. URL http://arxiv.org/abs/2404.19319.

[6] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. pages 1–20. ISSN 1041-4347, 1558-2191, 2326-3865. doi: 10.1109/TKDE.2025.3554028. URL http://arxiv.org/abs/2407.06204.

[7] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained BERT model and evaluation data. URL http://arxiv.org/abs/2308.02976.

[8] Mingzhi Chen, Taiming Lu, Jiachen Zhu, Mingjie Sun, and Zhuang Liu. Stronger normalization-free transformers, . URL http://arxiv.org/abs/2512.10938.

[9] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, . URL https://arxiv.org/abs/2302.06675. Version Number: 4.

[10] Chang Dai, Hongyu Shan, Mingyang Song, and Di Liang. HoPE: Hyperbolic rotary positional encoding for stable long-range dependency modeling in large language models. URL http://arxiv.org/abs/2509.05218.

[11] Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. URL https://arxiv.org/abs/2405.15682. Version Number: 4.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. URL http://arxiv.org/abs/1810.04805.

[13] Taj Gillin, Adam Lalani, Kenneth Zhang, and Marcel Mateos Salles. BERT-JEPA: Reorganizing CLS embeddings for language-invariant semantics. URL http://arxiv.org/abs/2601.00366.

[14] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, . URL https://arxiv.org/abs/2312.00752. Version Number: 2.

[15] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, . URL http://arxiv.org/abs/2312.00752.

[16] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. URL https://arxiv.org/abs/1802.09568. Version Number: 2.

[17] Sukjun Hwang, Aakash Lahoti, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers. URL http://arxiv.org/abs/2407.09941.

[18] Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. URL https://arxiv.org/abs/2411.16085. Version Number: 4.

[19] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training, . URL https://arxiv.org/abs/2305.14342. Version Number: 4.

[20] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, . URL https://arxiv.org/abs/1908.03265. Version Number: 4.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. URL https://arxiv.org/abs/1711.05101. Version Number: 3.

[22] Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. URL https://arxiv.org/abs/2306.06101. Version Number: 4.

[23] Matteo Pagliardini, Pierre Ablin, and David Grangier. The AdEMAMix optimizer: Better, faster, older. URL https://arxiv.org/abs/2409.03137. Version Number: 2.

[24] Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, and Russ Webb. Theory, analysis, and best practices for sigmoid self-attention. URL https://arxiv.org/abs/2409.04431. Version Number: 2.

[25] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. URL http://arxiv.org/abs/1908.10084.

[26] Hema Hariharan Samson. Lightweight transformer architectures for edge devices in real-time applications. URL http://arxiv.org/abs/2601.03290.

[27] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. URL https://arxiv.org/abs/1804.04235. Version Number: 1.

[28] Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. On the convergence analysis of muon. URL https://arxiv.org/abs/2505.23737. Version Number: 1.

[29] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, . URL http://arxiv.org/abs/2307.08621.

[30] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a compact task-agnostic BERT for resource-limited devices, . URL http://arxiv.org/abs/2004.02984.

[31] Shohei Taniguchi, Keno Harada, Gouki Minegishi, Yuta Oshima, Seong Cheol Jeong, Go Nagahara, Tomoshi Iiyama, Masahiro Suzuki, Yusuke Iwasawa, and Yutaka Matsuo. ADOPT: Modified adam can converge with any $\_2$ with the optimal rate. URL https://arxiv.org/abs/2411.02853. Version Number: 3.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL https://arxiv.org/abs/1706.03762. Version Number: 7.

[33] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing shampoo using adam. URL https://arxiv.org/abs/2409.11321. Version Number: 2.

[34] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. BitNet: Scaling 1-bit transformers for large language models. URL http://arxiv.org/abs/2310.11453.

[35] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. URL https://arxiv.org/abs/2208.06677. Version Number: 5.

[36] Haiqi Yang, Zhiyuan Li, Yi Chang, and Yuan Wu. A survey of retentive network, . URL http://arxiv.org/abs/2506.06708.

[37] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, . URL http://arxiv.org/abs/2311.12424.

[38] Daryl Noupa Yongueng and Hamidou Tembine. Holonorm. URL http://arxiv.org/abs/2511.10504.

[39] Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. MARS: Unleashing the power of variance reduction for training large models. URL https://arxiv.org/abs/2411.10438. Version Number: 4.

[40] Jing Zhang, Peng Zhang, Baiwen Kong, Junqiu Wei, and Xin Jiang. Continuous self-attention models with neural ODE networks. 35(16):14393–14401, . ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v35i16.17692. URL https://ojs.aaai.org/index.php/AAAI/article/view/17692.

[41] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, . URL http://arxiv.org/abs/1905.11881.

[42] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. URL https://arxiv.org/abs/2403.03507. Version Number: 2.

[43] Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization. URL http://arxiv.org/abs/2503.10622.