Contrastive Learning for Test-Time Training Layers

Anonymous authors
Paper under double-blind review

Abstract

Transformers have become the predominant architecture for sequence modeling, achieving state-of-the-art performance across natural language processing and other sequential domains. Despite their success, the quadratic complexity of self-attention imposes substantial computational and memory costs for long-context tasks. While alternative approaches, such as State-Space Models and linear attention, offer improved efficiency, they remain constrained in expressiveness and modeling long-range dependencies. Test-Time Training (TTT) layers provide a more flexible framework by parameterizing hidden states with nonlinear, input-dependent updates; however, prior approaches have relied on reconstruction-based objectives that lack justification and consideration of alternative learning methods. In this work, we propose Contrastive Test-Time Training (CTT), which integrates a contrastive learning objective into the TTT framework to explicitly align relevant query–value pairs while suppressing irrelevant features. On language modeling tasks at 140M parameters, CTT can match the performance of existing TTT models, indicating that it is neither detrimental nor inferior at smaller scales. Although not beneficial on its own, our evidence shows that CTT amplifies properties observed in models using Muon-based optimizers – those at the state-of-the-art for training larger models. This suggests that CTT has the potential to surpass TTT approaches once scaled to large model sizes.

1 Introduction

Transformers have become the dominant architecture for sequence modeling (35), driving breakthroughs in natural language processing (7)(6). Yet, the quadratic scaling of self-attention creates a major bottleneck for long-context applications, inflating compute and memory demands, raising environmental costs, and concentrating progress in the hands of a few well-funded organizations. Attempts to mitigate this, such as sparse or window attention, often come at the cost of reduced expressiveness (4)(1)(40).

Recent advancements in recurrent neural network architectures such as State-Space Models (SSMs)(10) (9) and linearised attention mechanisms (19), offer significant computational efficiency improvements compared to Transformers for processing long sequences. However, their expressive capacity in state tracking problems, similar to that of transformers(23), is less than that of RNNs (22). They have also been shown to fall short in tasks requiring the prediction of sequences with long contexts. (27)

Test-Time Training (TTT) layers (27) have emerged as a promising potential solution. Unlike other RNN-based methods, TTT parameterizes the hidden state through nonlinear, input-dependent updates, allowing for flexible and evolving context representations. The nonlinearity and input-dependent state updates has been shown to increase in expressive power and align it with that of classical RNNs.(22)

The original Test-Time Training (TTT) formulation suffered from poor GPU utilization due to frequent small-batch updates. Large-Chunk TTT (LaCT) (41) addressed this issue by processing larger token chunks, thereby improving hardware efficiency and enabling hybrid designs with window attention (1) and advanced optimizers such as Muon (20). These improvements opened the door to richer parameterizations and more sophisticated loss functions. Nevertheless, despite these advances, TTT and related methods continue to lag behind transformers on long-context tasks, frequently underperforming on benchmarks such as Needle-in-a-Haystack from RULER, which stresses retrieval across sequences exceeding 16k tokens (27; 41; 38; 11).

A further limitation of existing TTT approaches is their lack of principled justification for architectural and objective choices (41; 28). Many designs are implicitly tailored to mimic the behavior of attention mechanisms, but it remains unclear whether they are genuinely approximating attention or merely learning alternative heuristics with similar effects. We believe that reconstruction-based objectives eventually lead to an approximation of attention given sufficient data, but may do so in a suboptimal manner (27; 41).

In this work, we propose Contrastive Test-Time Training (CTT) layers, a novel extension of TTT that integrates contrastive learning (2; 15; 13) with the LaCT chunking mechanism. By employing an InfoNCE-style loss (34), CTT provides a more principled alternative to reconstruction losses: rather than attempting to reproduce entire value projections, the objective directly enforces alignment between relevant query–value pairs while suppressing uncorrelated features.

The benefits of contrastive objectives are twofold. First, they suppress noise and emphasize discriminative features, yielding more robust and stable representations(12)(36). Second, they are particularly effective in data-limited or resource-constrained regimes, offering faster convergence and better small-model performance than reconstruction-based counterparts(2). Our experiments on long-context language modeling with models up to 140M parameters demonstrate that CTT performs on par with LaCT and remains competitive with transformer baselines. Moreover, our analysis of optimizer choices suggests that Muon-based CTT variants exhibit promising scaling dynamics, with potential advantages likely to emerge more strongly at larger model sizes.

While our results are preliminary – limited in scale, benchmarks, and scope – they establish the feasibility of contrastive objectives for TTT and highlight their potential as a more principled framework for efficient long-context modeling.

2 Related Work

SSMs, such as S4 (10), are a class of sequence models that can be viewed as linear RNNs. They offer linear scaling in sequence length during training and constant state size during generation, performing strongly on long-range tasks. Mamba (9) further refines selective SSMs, introducing a core layer that is significantly faster than prior selective SSM implementations while remaining competitive with Transformers on language modeling. Mamba-2 (5; 37) extends this, offering further speed improvements and deeper theoretical connections between SSMs and attention through "structured state space duality" (SSD). This duality demonstrates that SSMs can be represented as structured matrices, enabling efficient algorithms leveraging matrix multiplication units. However, despite these advances, SSMs remain fundamentally constrained in expressive capacity (22), limiting their ability to capture the rich, nonlinear dependencies needed for long-context reasoning. They also have been shown to fall short in tasks requiring the prediction of sequences with long contexts (27).

Linear Attention mechanisms replace the exponential similarity function in Transformers with a dot product formulation, enabling linear-time inference and interpretation as a linear RNN with 2D hidden states (19). Yet early linear attention methods underperformed compared to standard softmax attention. Gated Linear Attention (GLA) (37) introduce data-dependent gates, improving expressivity and showing competitiveness with architectures such as LLaMA (32), RetNet (29), and Mamba. Nevertheless, these methods still rely on approximations of attention, and their performance degrades on extreme long-context benchmarks, where limited memory capacity hinder retrieval.

DeltaNet (24; 39) interprets linearized self-attention as a form of fast weight programming, replacing additive outer products with a delta-rule update that corrects key–value associations and dynamically computes learning rates. This perspective yields deeper insight into the equivalence between linear Transformers and fast weight programmers. However, DeltaNet and its variants remain tied to reconstruction-style objectives, which are not explicitly designed to suppress noise or filter irrelevant features.

In contrast, Test-Time Training (TTT) layers (27) represent a different paradigm, parameterizing the hidden state through nonlinear, input-dependent updates. This grants them an expressive power closer to that of classical RNNs, while retaining the efficiency of linear-time inference. Yet existing TTT approaches typically employ reconstruction-based objectives, which can eventually mimic attention but do so inefficiently and

without a principled mechanism for distinguishing signal from noise. Our proposed Contrastive Test-Time Training (CTT) addresses this shortcoming by replacing reconstruction with a contrastive objective, directly enforcing alignment between relevant query–value pairs while suppressing irrelevant correlations. In doing so, CTT provides a more principled and robust alternative for long-context modeling.

3 Background

We define the notation used within this paper as follows. Matrices are defined by bold uppercase variables $\mathbf{A}$, vectors as bold lowercase $\mathbf{a}$, and scalars as normal lowercase a. We use $\mathbb{R}$ to define the set of real numbers, $\nabla_X y$ for the Matrix derivatives of y with respect to X, and f_θ as a function parameterized by the variable or matrix f_θ. We also use $\mathcal{L}(a, b)$ to denote a loss function dependent on both a and b. We use t to denote the position in the sequence and c to denote a chunk or group of samples in the sequence, while x and y are the input and outputs to a layer or model, respectively. We use the same terminology as in the LaCT paper, distinguishing the TTT branch from the Attention branch.

3.1 Test-Time Training Layers

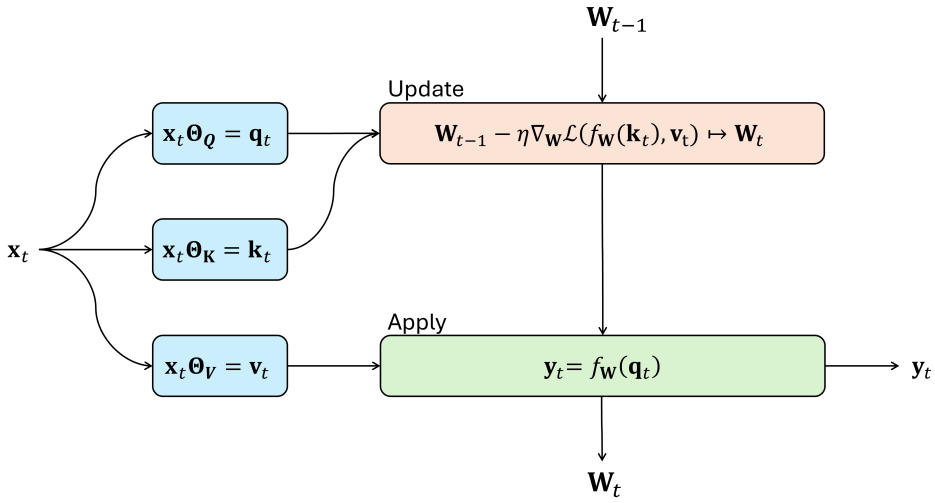

Figure 1: Schematic of a general TTT layer. The notation follows that used throughout the rest of the paper.

We define TTT layers as follows. First, consider the input $\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ from the sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N]$. Next, this input gets projected into query, key, and value vectors, $\mathbf{q}, \mathbf{k}, \mathbf{v}$ respectively. These can be considered akin to that in attention where the projections are of size $\mathbf{q}_t, \mathbf{k}_t, \mathbf{v}_t \in \mathbb{R}^{d_{in}}$. As highlighted, the hidden state in TTT can be considered a neural network, or any ML model, with weights that are updated to store the context. These weights are updated during training and inference. This can be defined as a neural network $f_{\mathbf{W}} : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ parametrized by model weights $\mathbf{W}$, typically $d_{out} = d_{in}$. These weights, $\mathbf{W}$, can be referred to as fast weights or inner-loop parameters. As opposed to slow weights or outer-loop parameters, which are denoted by Θ throughout this paper and are frozen during inference. These fast weights are updated during the forward pass using a loss function $\mathcal{L}(\cdot, \cdot)$ between the key vector, which has passed through the fast weight model, $f_{\mathbf{W}}(\mathbf{k}_t)$, and the value vector $\mathbf{v}$. In the original paper, the L2 loss function was used to encourage the network to associate keys with values. This loss function is then used with a gradient-based optimizer to update the fast weights, using η as the learning rate.

$$\textbf{Update:} \qquad \mathbf{W}_{t-1} - \eta\nabla_{\mathbf{w}}\mathcal{L}(f_{\mathbf{W}}(\mathbf{k}_t), \mathbf{v}_t) \mapsto \mathbf{W}_t \tag{1}$$

This operation can be seen as encoding the KV cache into the fast-weights. The updated parameters can then be used with the query to produce the output, $\mathbf{y}_t \in \mathbb{R}^{d_{out}}$.

$$\textbf{Apply:} \qquad \mathbf{y}_t = f_{\mathbf{W}}(\mathbf{q}) \tag{2}$$

This update is followed by an application performed for each token in the sequence, $\mathbf{X}$. A schematic depicting this operation can be seen in Figure 1. Generally, the layers' performance benefits from trained starting weights during inference as opposed to a random one (27). These updates can be batched together to improve GPU utilization.

3.1.1 LaCT: Large Chunk Test-Time Training Layers

The TTT branch of the LaCT layer computes gradients of the summed losses over all the keys and values within the chunk as seen in equation 3, where N is the chunk size and C denotes the chunk number. Weight updates are only performed at the end of each chunk, and so inter-chunk dependencies are not captured by the TTT branch, necessitating the need for the attention branch.

$$\textbf{Update} \left[\nabla_{\mathbf{w}} \sum_{i=1}^{N} \eta_i \mathcal{L}(f_{\mathbf{W}}(\mathbf{k}_i), \mathbf{v}_i), \mathbf{W}_{C-1} \right] \mapsto \mathbf{W}_C \tag{3}$$

The authors apply LaCT across multiple domains, making architectural adjustments tailored to each. In this work, we focus on its application to language modeling and therefore present the details of LaCT specifically in the context of this task. As with standard TTT, LaCT layers replace the attention component in transformers. When used within a transformer architecture, they still require MLPs within the block.

3.2 Contrastive Learning

Contrastive learning is a self-supervised representation learning paradigm that trains models to distinguish between similar (positive) and dissimilar (negative) data pairs, encouraging the representation space to cluster semantically similar samples while separating unrelated ones. Early work, such as Noise-Contrastive Estimation (12), framed the problem as distinguishing observed data from noise. Recent advances, notably SimCLR (3) and MoCo (14), have demonstrated that contrastive learning can rival or surpass supervised learning in tasks like image classification by leveraging large amounts of unlabeled data.

The core idea is often formalized through the loss functions of these architectures, which maximizes agreement between positive pairs in a latent space while minimizing agreement with negatives, thereby learning robust and transferable representations.(33)

A popular loss function used in many frameworks is the InfoNCE loss (33). It has many variations, including the NT-Xent loss(3), which all follow the same idea of a cross-entropy style loss function. It can be defined as follows:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\tau)}{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\tau) + \sum_{i=1}^{K} \exp(\text{sim}(\mathbf{z}, \mathbf{z}_i^-)/\tau)}, \tag{4}$$

Where $\mathbf{z}$ and $\mathbf{z}^+$ are the representations of a positive pair, $\mathbf{z}_i^-$ are representations of K negative samples, τ is a temperature hyperparameter, and $\text{sim}(\cdot, \cdot)$ denotes a similarity measure, such as cosine or dot-product similarity. By minimizing this loss, the model learns to pull together positive pairs and push apart negative pairs in the embedding space.

4 Contrastive Test-Time Training (CTT) Layers

In this work, we adapt the InfoNCE loss(34) to train the fast weight learning model that maps discrete keys into embeddings enriched with information about their associated values. The InfoNCE objective encourages

embeddings of keys to be positioned so that, within the representation space, the joint distribution. $p(\mathbf{k}_t, \mathbf{v}_t)$ of key–value pairs is more predictable than what would be expected under independence, $p(\mathbf{k}_t)p(\mathbf{v}_t)$. By doing so, the fast weight model's learned transformation ensures that when a query is input at inference, its embedding will be drawn toward a region of the space densely populated by all historically strong keys with their associated values, thus learning a representation that reflects all prior keys and values. This captures and compresses meaningful statistical dependencies from the training data directly into the fast weights.

To adjust the InfoNCE loss for the CTT layer, we make use of the chunking process employed by LaCT to provide a set of positive and negative samples to draw from. We redefine the InfoNCE loss now in terms of the parameters of the CTT layer for a single anchor or word embedding within the chunk. The query, key, and value projection **q, k, v** are retained from TTT along with the fast weight model $f_{\mathbf{W}}(\mathbf{k})$.

$$\mathcal{L}_i = -\log \frac{\exp\left(\mathbf{v}_i^\top g_\theta(f_{\mathbf{W}}(\mathbf{k}_i^+))/\tau\right)}{\sum_{j=1}^{N} \exp\left(\mathbf{v}_i^\top g_\theta(f_{\mathbf{W}}(\mathbf{k}_j))/\tau\right)} \tag{5}$$

In Equation (1), i and j are indexes for each value and key within the chunk, which is of size N. We set the value vector $\mathbf{v}$ as the anchor to which comparisons are made. The positive key, $\mathbf{k}^+$, corresponds to the true matching example for the anchor, whereas $\{\mathbf{k}_j\}_{j=1}^N$ denotes the set of all keys in the chunk, including the negatives and positives. Scoring can be computed as standard using the simple dot-product similarity between $\mathbf{v}_i^\top$ and $f_{\mathbf{W}}(\mathbf{k}_i^+)$. We also introduce an additional scoring network, parameterized by slow weights g_θ, shown in 5. Similarity is computed between this network's output and $\mathbf{v}^\top$. We call this NN-similarity throughout this paper. This approach allows the model to learn a similarity scoring function that depends on overall model performance, rather than relying solely on a conventional scoring mechanism. The purpose of this follows the W_k term used in the scoring function of the contrastive predictive coding paper(34). We compare the NN-similarity with the dot-product similarity in experiments. We evaluate the loss for each anchor in the chunk through the index i and average the losses over the entire chunk to form a final loss for the chunk.

$$\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i \tag{6}$$

The InfoNCE loss maximizes the similarity between the anchor and the positive key relative to the similarities with the negative keys, effectively encouraging the model to distinguish positive pairs from negative ones. As with the LaCT architecture, updates to the fast weight model can only be made at the end of the chunk, and so interchunk dependencies cannot be modeled. We also note that due to the nature of TTT, autodiff cannot be employed for the fast-weight updates, and so the derivatives of each of the loss functions have been derived and coded analytically.

5 Experiments

To evaluate our hypothesis, we compare CTT against LaCT and a full causal Transformer benchmark, conducting both training and evaluation on language modeling tasks. We selected language modeling as the application domain because much of the recent progress in long-context foundational models (38)(5)(39) has been driven by this area, making it a natural and relevant choice.

5.1 Datasets and Benchmarks

We use the Long-context-datasets (31) for pretraining of all of the models. We selected approximately 3B tokens for pretraining of the models, aligning with the Chinchilla scaling law (16). This has been tokenized using the mixstral tokenizer (32,000 codebook) taken from the fla-hub/transformer-1.3B-100B huggingface upload (8). We reserved approximately 1M tokens of the dataset for post-training validation of the models. We shuffle the validation data using 5 different seeds, evaluating each of the models on these seeds.

For the test dataset, we started with the 10B token sample Fineweb-edu dataset (21) and filtered it into three separate datasets of different target sequence lengths. In each, we keep only examples where the context length is longer than the target sequence lengths of 8k, 16k and 32k. These examples were then truncated to exactly 8k, 16k and 32k tokens, so that each evaluation window contained a single, complete example. This approach ensured the model is tested on sequences that fully span the target length, giving a more accurate picture of how well it handles long contexts. By contrast, if shorter sequences were combined to fill the context window, the evaluation could give a misleading impression of the model's performance at longer lengths. We follow the LaCT and TTT papers, evaluating models on the per-token loss of each of the models, shuffling the dataset with 5 different seeds, and testing each of the models on these seeds.

5.2 Model Details and Baselines

We have implemented LaCT layers with negative dot-product and the CTT layers with muon(18) and momentum(30) based optimizers. CTT models using NN-similarity have been clearly noted; otherwise, dot-product similarity is used. We have also implemented a full causal transformer as a baseline for comparison.

We have generally followed the same architecture as that within the LaCT paper for the CTT layer. We use a SwiGLUMLP(25) as the fast weight learner, a momentum or muon-based update rule with L2 weight normalization. A multi-head design is applied to the TTT branch and the attention branch of the layer. The queries and keys for the TTT branch are rescaled, and all Q, V, K are passed through a swish nonlinearity. Additionally, an L2 norm is applied to the queries and keys within the TTT branch. RoPE (26) with a large base of 1 million is used for all architectures. The root mean square layer normalization is applied to the output of the TTT branch and is then scaled by another per-head learnable scalar. A diagram of this architecture, along with further details of the parameters used, can be found in the appendix A. All models follow the generic transformer residual modeling block with a sequence modeling block and MLP block stacked with residuals and layer norms between them.(35)

As per the LaCT paper, a low-rank version of the fast weights has been used to reduce the total number of trainable parameters in the TTT branch. This allows for a fairer comparison in terms of the number of trainable parameters between the LaCT models and the transformer model.

Due to computational constraints as described in the training details, we have only tested models up to 140M parameters. It is planned that future experiments will take the findings from this work and scale up the most promising models for further analysis.

5.3 Training Details

All models were trained using a single Nvidia 80Gb Tesla A100 GPU. The models using a momentum update rule were trained for approximately 24 hours each; the models with the muon update rule took longer, at approximately 48 hours each. We utilized the flame training framework (42) to aid with training of the models, including the baselines. We used a base learning rate of 1e-3 with a cosine decay scheduler and 1024 warm-up steps. All models were randomly initialized with a standard deviation of 0.02. Detailed training parameters can be found in appendix A.

5.4 Results

The results indicate that the CTT models can achieve performance comparable to the LaCT models at model sizes of 140m parameters. Although CTT presents few advantages at this scale, our results show that there are indications that scaling up these models may lead to better performance. While these experiments do not constitute a comprehensive evaluation across a broad range of downstream tasks, these preliminary findings nonetheless reveal several notable characteristics of both CTT and LaCT models at this scale.

The CTT model trained with the muon update rule exhibited early overfitting to the training data, before any of the other models. A possible explanation is that the muon-based optimizer in combination with CTT enables the model to more rapidly internalize recurring patterns across sequences. This behavior may cause the model to memorize passages rather than develop a deeper representation of their meaning,

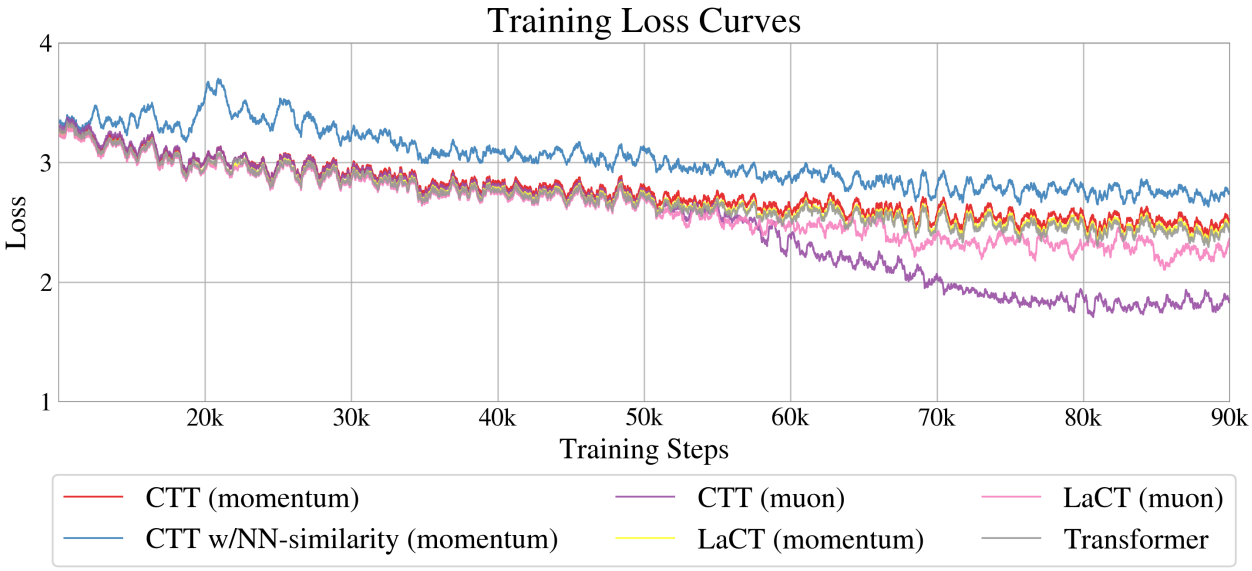

Figure 2: Training loss curves for all 140M parameter models. The x-axis and y-axis have been cropped to highlight differences at later training steps. CTT models with w/NN-similarity achieve a worse loss compared with other models, while the CTT with a muon optimizer appears to deviate from the other models, achieving a lower training loss.

Table 1: Results from the post-training validation of the models. Throughput during training is also provided for comparison. We report the mean of the average losses across the 5 shuffle seeds, along with the standard deviation across those averages. The CTT (muon) model performs worse on the held-out validation dataset, indicating there may be some form of overfitting occurring during training. The LaCT muon model may also be experiencing a similar effect, performing worse on the validation than on the training when compared with the momentum equivalent. CTT throughput is only slightly worse when compared with LaCT equivalents.

Model	Val. Average Loss	Std. Average Loss	Val. Average ppl	Std. Average ppl	Train TPS
CTT (momentum)	2.31	0.01	11.40	0.05	43k
CTT w/NN-similarity (momentum)	2.52	0.01	14.20	0.05	42k
CTT (muon)	2.29	0.01	20.47	0.05	30k
LaCT (momentum)	2.26	0.01	10.83	0.04	51k
LaCT (muon)	2.29	0.01	11.21	0.05	34k
Transformer	2.23	0.01	10.58	4.16	41k

which in turn results in lower loss on text with highly repetitive structures. If the training data were not thoroughly shuffled, certain word patterns may have appeared with disproportionate frequency, potentially exacerbating this effect. This finding is broadly consistent with our hypothesis that CTT can learn effectively from relatively smaller amounts of data. Although such overfitting is detrimental in evaluation scenarios at the current scale, a similar but milder trend was observed in the LaCT-muon variant. As shown in Figures 2 and 3, LaCT with muon achieves lower training loss than test loss, indicating that this behavior is not unique to CTT but is instead accentuated by it. This is further supported by the results at 8k and 16k context length, appendix B.

Prior results from the LaCT paper (41) demonstrate that the muon update rule can reduce validation loss and improve performance on the Needle-in-a-Haystack task from the RULER benchmark (17), provided that models are scaled and trained on substantially larger datasets. Our findings indicate that CTT with muon amplifies certain beneficial characteristics observed in LaCT with muon models, which are not present in momentum-based models. These observations suggest that CTT muon-based variants may yield superior results at larger model scales, such as 760M or 3B parameters.

Our results show that CTT models containing the NN-similarity component had a detrimental effect on the CTT model's learning speed and generalization capability. This degradation is likely attributable to the

added complexity introduced by the NN-similarity mechanism. Given the absence of clear benefits, it is unlikely that future work will prioritize this particular design choice.

The CTT model trained with momentum exhibits performance that closely mirrors that of the LaCT-momentum variant, differing only slightly at shorter context lengths on downstream tasks, where it requires a longer warm-up period.

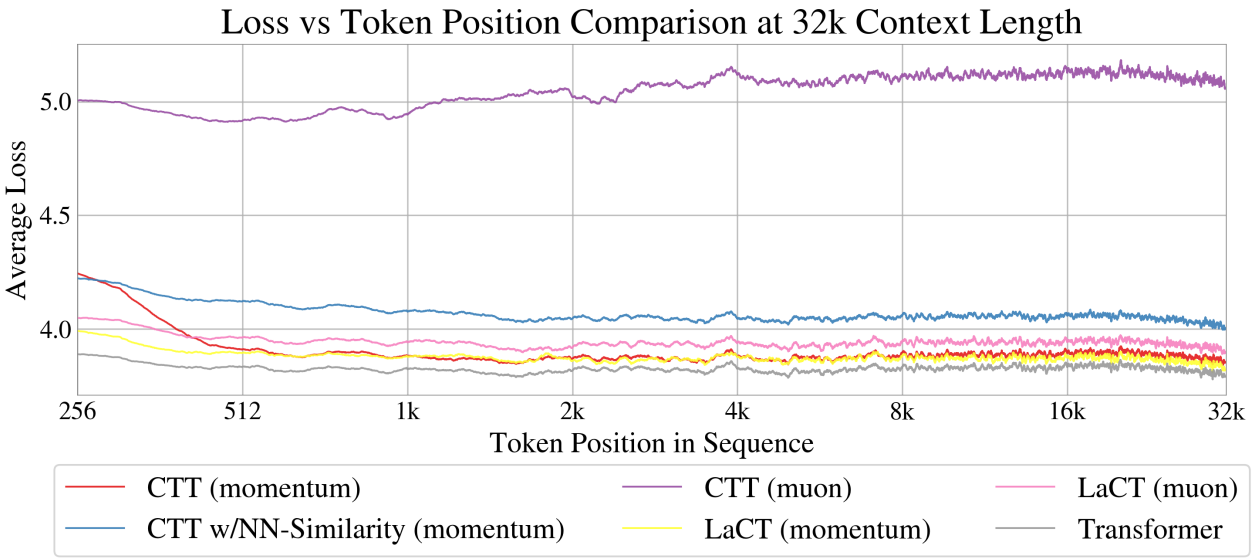

Figure 3: Results on loss per token-index position for the FineWeb-edu 10BT sample dataset filtered for all examples beyond 32k tokens in length. A running average filter with 100 samples in width has been used to smooth the results. The standard deviation of the mean across dataset shuffles for all models is below 0.007. CTT with muon performs markedly worse than other models due to overfitting seen during training and validation. LaCT with muon also experiences a similar effect. CTT with momentum update model performs worse at short contexts but follows a similar trajectory to the LaCT and transformer models.

6 Conclusions

In this paper, we present Contrastive Test-Time Training (CTT), a novel learning method that extends the capabilities of Test-Time Training (TTT) layers through the application of contrastive learning. By integrating CTT with the Large-Chunk Test-Time Training (LaCT) architecture, we were able to incorporate negative samples, a crucial element for employing the InfoNCE loss function. We posited that this approach is particularly effective at filtering out irrelevant features and noise, leading to more stable and robust representational learning.

Our empirical evaluation compared CTT against a LaCT baseline and a causal Transformer on language modeling tasks, utilizing models with up to 140M parameters. The results demonstrate that CTT models can achieve performance on par with their LaCT counterparts, with the CTT-muon variant showing particularly promising results. While the current model sizes were not sufficient to pass the RULER benchmark for long-context retrieval, our findings suggest that CTT is a scalable and promising direction for future research.

Specifically, our analysis with the muon-based optimizer indicates that its benefits are likely to be realized at a larger scale. This observation is consistent with prior research on LaCT, which found that muon-based optimizers yield significant performance gains when applied to larger models and datasets. We therefore recommend that future work should focus on scaling up the CTT-muon variant to model sizes of 760M or 3B parameters and evaluating its performance on more demanding benchmarks.

Due to its exploratory nature this study has notable limitations: experiments were restricted to models up to 140M parameters, leaving uncertain whether findings—such as the benefits of the Muon update

rule—generalize to larger scales; evaluations were confined to language modeling, so effectiveness in domains like vision, multimodal learning, or reinforcement learning remains untested; downstream evaluation was limited, with poor performance on benchmarks like RULER constraining task-level generalization; the work lacks strong theoretical grounding, relying primarily on empirical evidence; and exploration of design choices was shallow, with sparse ablations and minimal hyperparameter tuning. Nonetheless, the results highlight promising behaviors. Future experiments should investigate the influence of chunk size on the warm-up dynamics of CTT models, as this factor may help mitigate the observed lag. As with the CTT-muon model, scaling up the CTT-momentum variant could reveal additional properties that warrant further analysis.

References

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[5] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024.

[6] Gemini Team et al. Gemini: A family of highly capable multimodal models.

[7] OpenAI et al. Gpt-4 technical report, 2024.

[8] fla-hub. fla-hub/transformer-1.3b-100b. `https://huggingface.co/fla-hub/transformer-1.3B-100B`, 2025. 1.36 B parameters, BF16; trained on cerebras/SlimPajama-627B, accessed August 19, 2025.

[9] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

[10] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

[11] Han Guo, Songlin Yang, Tarushii Goel, Eric P. Xing, Tri Dao, and Yoon Kim. Log-linear attention, 2025.

[12] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[15] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network, 2018.

[16] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

[17] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[18] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, December 2024. Accessed: 2025-08-19.

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[20] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025.

[21] Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024.

[22] William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models, 2025.

[23] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545, 06 2023.

[24] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR, 18–24 Jul 2021.

[25] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, February 2020. Accessed: 2025-08-19.

[26] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[27] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2025.

[28] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts, 2020.

[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, 2023.

[30] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[31] togethercomputer. Long-data-collections. `https://huggingface.co/datasets/togethercomputer/Long-Data-Collections`, 2023. Accessed: 2025-08-19.

[32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[36] Yihao Xue, Kyle Whitecross, and Baharan Mirzasoleiman. Investigating why contrastive learning benefits robustness against label noise, 2022.

[37] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025.

[38] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024.

[39] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2025.

[40] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.

[41] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. Test-time training done right, 2025.

[42] Yu Zhang and Songlin Yang. Flame: Flash language modeling made easy, January 2025.

A Further Architectural Details for Experiments

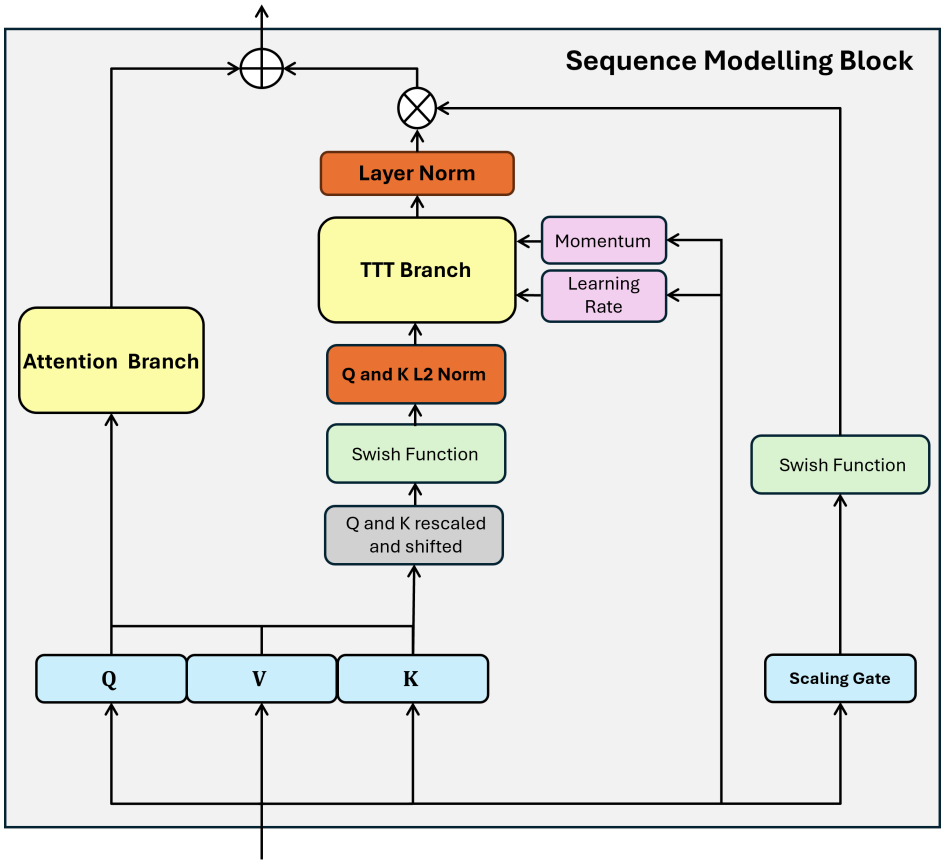

Figure 4: This architecture depicts the form for the CTT and LaCT modeling blocks. It loosely follows a similar architecture to that used in the Mamba paper (9). Yellow blocks depict the key sequence modeling processes, blue are linear projections, green are non-linear activations, orange are normalizations, gray are scaling blocks, and purple are optimizer-related scalars.

Table 2: Model parameters for the 140M parameter models. CTT and LaCT parameters have been combined for brevity; parameters for CTT models have been noted with a star.

Parameter	**CTT/LaCT**	**Transformer**
Number of Layers	12	12
Hidden Activation	Swish	Swish
Hidden Size	768	768
Attention Heads	24	24
Window Size	2048	N/A
Lact Chunk Size	2048	N/A
TTT Heads	4	N/A
Vocabulary Size	32,000	32,000
Context Length	32,768	32,768
Precision	BF16	BF16
Norm EPS	1e-6	1e-6
CTT Temperature*	0.5	N/A

Table 3: Training Details for the 140M Parameter Models. All models have been trained with the same training parameters to ensure fair comparison.

Parameter	**Value**
Optimizer Type	AdamW
Optimizer EPS	1e-15
Learning Rate	1e-3
LR Scheduler Decay Type	Cosine
LR Minimum	0.1
Training Seed	42
Max Norm	1.0
Training Steps	91550
Sequence Length	32768
Batch Size	1

B Additional Results on Language Modeling at 8k and 16k Context Length

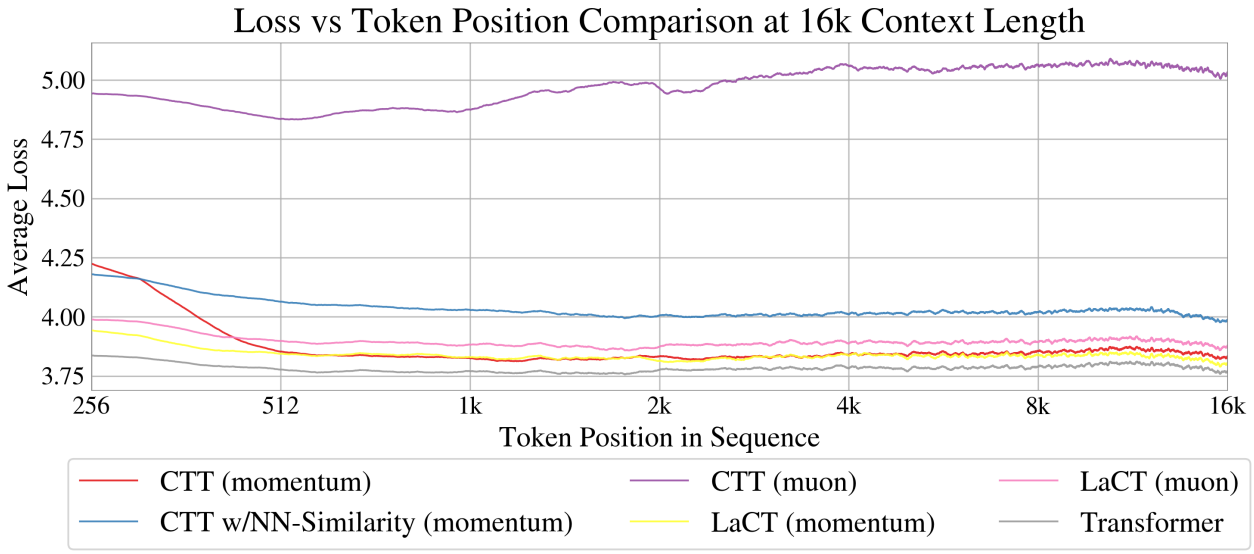

Figure 5: Results on loss per token-index position for the FineWeb-edu 10BT sample dataset filtered for all examples beyond 16k tokens in length. A running average filter with 100 samples in width has been used to smooth the results. Results support findings at 32k context length.

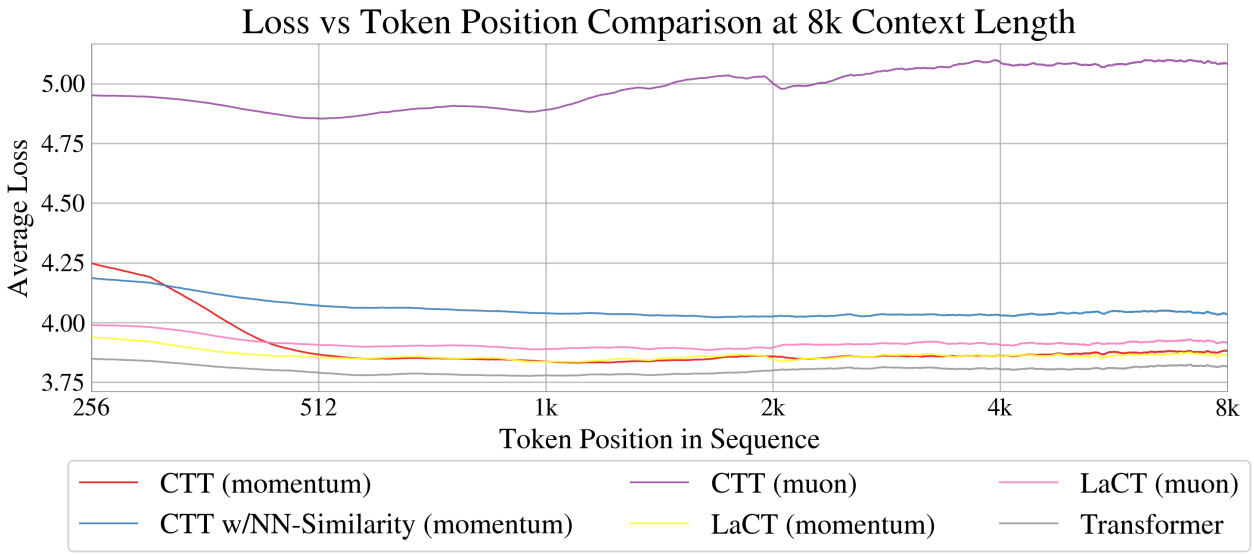

Figure 6: Results on loss per token-index position for the FineWeb-edu 10BT sample dataset filtered for all examples beyond 8k tokens in length. A running average filter with 100 samples in width has been used to smooth the results. Results support findings at 32k context length.