

Breaking Causality in LLM Training with Sparsity and Quantization

George Retsinas Anjolie Agrusa Oleg Rybakov Nicholas Knight
Mikail Khona Chong Yu Tijmen Blankevoort Jeff Pool
NVIDIA

{gretsinas,aagrusa,orybakov,nknight,mkhona,chongy,tblankevoort,jpool}@nvidia.com

Abstract

Autoregressive training requires that the computation at each position ignore future tokens. Causal masking enforces this for self-attention, but efficient training recipes can still break causality when they compute shared statistics across the sequence dimension. We study two such operations: structured activation sparsity and block-scaled activation quantization. Applying these operators along the sequence dimension is attractive for hardware efficiency and can look harmless, but it can break causality when the mask or scale applied to a past token depends on future activations. We show that models do exploit this dependency: the training objective continues to improve under the non-causal operator, but causal validation degrades when the future-dependent computation is removed. We analyze why mitigations that keep the operator in the forward pass can postpone or weaken exploitation, but cannot remove the future-to-past dependency. The solution is to remove non-causal statistics from the forward computation used for prediction, and we show that existing sparsity and quantization recipes *do* respect this rule, though causality was not an explicit design decision.

1 Introduction

Autoregressive models should not use future information when making a next-token prediction. Equivalently, the representation at position t should depend only on positions $1, \dots, t$. In self-attention, this constraint is enforced with a causal mask, which prevents each position from attending to future positions (Vaswani et al., 2017). But attention is not the only way future information can reach the past. We call any computation through which future activations can change a past representation a future-to-past path. Operators that compute a statistic across sequence positions and use it to transform each token can create such a path, even without explicit token mixing. If the model can



Figure 1: *Operations along the sequence dimension change past tokens*: the value of token A after 1:2 sparsification along the sequence length depends on the next token (B or C) in the sequence.

exploit this path during training, it may improve training loss by relying on signals that are unavailable at inference time.

Figure 1 shows the issue for a 1:2 sparsity operator along the *sequence* dimension (a subset of the commonly-supported 2:4 sparsity¹). For each hidden dimension, only the larger of the two neighboring token values is kept. Holding the first token in the pair fixed, the second (future) token can change the first (past) token’s output, as illustrated in Figure 1. This modifies the past representation to include a future-dependent signal that downstream layers can learn to use.

A similar effect occurs in micro-scaled activation quantization (Rouhani et al., 2023; Open Compute Project, 2023; NVIDIA, 2025b). Such formats divide values into blocks and use a shared scale, typically set by the largest magnitude in the block, to define the quantization grid. When a block spans the sequence dimension, future activations can change the grid used for past tokens.

¹Within every group of 4 contiguous elements, 2 must be zero. NVIDIA Sparse Tensor Cores use this pattern to accelerate matrix multiplications (Mishra et al., 2021).

These are practical configurations in efficient LLM training with activation sparsity (Haziza et al., 2025) and block-scaled quantization (NVIDIA et al., 2026). Both can push compression structure onto the sequence dimension to maximize training acceleration. We therefore focus on these two operators as practical cases where a compression decision, and not explicit token mixing, can create a future-to-past path that is easy to miss.

Non-causal flow is well-known and obvious when the path is explicit, such as in self-attention. Here, the risk is easy to miss because the future-dependent signal enters only through a mask decision or a block scale. Since the non-causal version also offers acceleration, it is tempting to treat the path as too small to matter, or as something that simple mitigations can control. We refer to this limited future-to-past path as a side-channel. While there exists a focused report on block-quantized attention (Mishra et al., 2026), it remains unclear how severe the issue can be, how models exploit this path, and whether such mitigations are enough.

To illustrate the danger of such operators, we first show the loss divergence that appears in these practical acceleration settings. Training loss continues to decrease under the non-causal operator, but causal validation worsens when we evaluate without the future-dependent computation. Natural attempts to “reduce” the side-channel, such as randomizing it by shuffling tokens, delay the divergence rather than eliminate it. The divergence may not appear immediately, since it depends on the model size and the training horizon, so a short successful run can be misleading.

Inspired by these observations, we analyze what it takes to eliminate future-dependency while the operator stays in the forward pass. We argue that this requires the affected activation dimensions to be insensitive to the operator’s output, which means they stop contributing to the model’s predictions on those inputs. Mitigations, such as the aforementioned token shuffling, can narrow the path, but closing it requires paying the capacity cost of making those dimensions insensitive to the operator.

Our contributions are: **(1) Identification of broken causality** in structured sparsity and block-scaled quantization during training, including how models exploit the future-to-past path, **(2) A structural argument and empirical demonstration** that mitigations cannot close the future-to-past path without sacrificing model capacity, and **(3) A causality-preserving design rule**, and reinter-

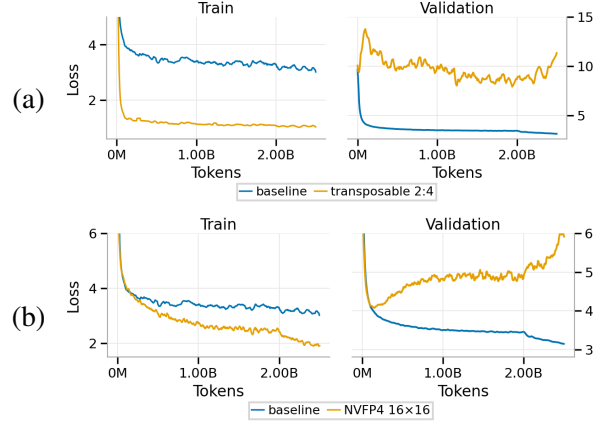


Figure 2: *Effect of non-causal activations*: Training loss vs. causal evaluation loss under (a) 2D transposable 2:4 sparsity and (b) NVFP4 with 16×16 block.

pretation of existing throughput-motivated recipes as instantiations of it.

2 Broken Causality in Practice

We first show how operators designed for efficiency can inadvertently break causality. During training, the input activation to any linear layer is used twice: once in the forward pass, contracting over the hidden dimension H , and once in the weight gradient, contracting over the sequence dimension T . Hardware support for structured sparsity (Sparse Tensor Cores) and micro-scaled quantization (NVFP4, MXFP4) operates along the reduction axis of a matrix multiplication. Because the same activation tensor participates in different GEMMs with different reduction axes, a natural attempt to accelerate all of them is to impose structure (sparsity or quantization) along both axes. Structure along H is per-token and causal. Structure along T couples tokens across positions, including future ones.

2.1 2:4 Activation Sparsity

Recent work accelerates both forward and backward passes by applying hardware-supported 2:4 sparsity to naturally sparse activations (Haziza et al., 2025; Madhyastha et al., 2026). Consider the MLP block of a standard autoregressive transformer, and let W_1 and W_2 be the weights of FC1 and FC2. The input X passes through FC1 and a nonlinear activation σ , producing $A = \sigma(XW_1) \in \mathbb{R}^{T \times H}$. Activation functions such as ReLU² produce outputs that are inherently sparse, making hardware-supported 2:4 structured sparsity a natural fit.

Let $M \in \{0, 1\}^{T \times H}$ denote the sparsity mask. The masked activation $M \odot A$ serves as FC2’s input, participating in two GEMMs: the forward pass and the weight gradient for W_2 . By the chain rule through the element-wise mask, the same mask also applies to the activation gradient ∇_A during FC1’s backward pass, participating in two additional GEMMs. Overall, we have:

	GEMM	Operation	Reduction axis
FC2	FPROP	$Y = (M \odot A) W_2$	H
	WGRAD	$\nabla_{W_2} = (M \odot A)^T \nabla_Y$	T
FC1	DGRAD	$\nabla_X = (M \odot \nabla_A) W_1^T$	H
	WGRAD	$\nabla_{W_1} = X^T (M \odot \nabla_A)$	T

To accelerate all four sparse GEMMs from a single mask, M must satisfy the 2:4 pattern along both axes simultaneously. A transposable 2:4 mask, like those used by weight tensors (Hubara et al., 2021), achieves this with exactly 50% sparsity.

The causality violation appears along the sequence axis: groups of 4 tokens at the same hidden dimension form a tile, and the two smallest magnitudes are pruned. Whether position t keeps its activation at dimension h depends on the magnitudes produced by later positions in the same tile. Thus a past token’s mask bit can become a function of future activations.

Observed Leakage. We train a NanoChat-inspired (Karpathy, 2025) transformer with the transposable mask over the FC2 input activations (experimental details are given in Section B). At evaluation time, we replace the 2D mask with the hidden-axis mask, which is causal and available during autoregressive decoding, and still accelerates the forward pass. The causal sparse baseline uses this hidden-axis mask during both training and evaluation (which does not accelerate any backward pass GEMM). Under the transposable mask, the model reaches lower training loss than this causal sparse baseline, but causal evaluation loss worsens within the first few hundred iterations (Figure 2a).

2.2 Micro-Scaled Activation Quantization

Micro-scaled quantization formats such as NVFP4 and MXFP4 use a shared scale factor for each block of values, typically computed from the maximum magnitude in the block. The scale determines the quantization grid for all elements in the block. The same activation tensor is used with reduction axis H in the forward pass and reduction axis T in

the weight gradient. A natural way to accelerate both GEMMs is therefore to use blocks that extend along both dimensions.

If blocks are formed only along H , the scale is computed within a single token and is causal. If a block extends along T , the scale is computed across positions. When such a block contains future tokens, their magnitudes can set the quantization grid used for past tokens in the same block.

In our experiments, we apply this activation quantization to every linear layer input.

Observed Leakage. With 2D micro-scaled activation quantization, the same train/evaluation divergence appears. The baseline is the same model trained and evaluated in BF16, without activation quantization. For causal evaluation of the 2D quantized checkpoint, we disable activation quantization and run the forward pass in BF16, so no sequence-wise scale can carry future information. The 2D quantized run reaches lower training loss than the BF16 baseline, but its BF16 evaluation loss worsens (Figure 2b).

2.3 Empirical Signature

Both operators produce the same empirical signature: training loss and causal evaluation loss diverge when the sequence-axis dependency is removed at evaluation. A train/evaluation gap can also arise from ordinary overfitting, but the control runs identify the source here: the causal sparse baseline and the BF16 baseline do not show the same validation regression. In Figure 2, the practical runs make this especially clear: the non-causal runs reach much lower training loss than their corresponding causal baselines, yet their causal evaluation loss worsens. This figure motivates the rest of the paper by showing that the problem appears in realistic acceleration settings and with a large effect. The next section asks how such a limited side-channel can produce this failure.

3 The Exploitation Mechanism

To understand how future information reaches past positions through non-causal operators, we analyze the simplest case: two consecutive positions t and $t + 1$ whose hidden activations are coupled by a single non-causal operator.

3.1 The Side-Channel

Consider two consecutive positions, with tokens x_t and x_{t+1} . For one hidden dimension h , let a_t^h be

the activation at position t before the operator, and let $\tilde{a}_t^h$ be the value passed forward after the operator. A non-causal operator can make this value depend on both positions:

$$\tilde{a}_t^h = \phi(a_t^h, a_{t+1}^h).$$

In a causal computation, fixing the context through position t fixes $\tilde{a}_t^h$. Here, changing the future token x_{t+1} can still change $\tilde{a}_t^h$. This is the side-channel in this setting. The future token changes the representation passed forward at position t .

In sequence-axis sparsity, a magnitude comparison carries the side-channel. A future activation can decide whether the current activation is kept or zeroed. In block quantization, the shared scale carries it. A future activation can change the grid used to round the current activation.

3.2 Mechanism Overview

Figure 1 shows a simple case of this side-channel, where changing the next token can change the current token’s post-operator representation. Exploitation requires two more pieces: upstream layers must produce distinguishable representations, and downstream layers must learn to read the trace left by the operator.

1. Token Representations. Ordinary training already builds token-specific distinct representations, which help downstream layers predict the next token causally.

2. Future-to-past trace. The non-causal operator imposes the future token’s representation into the current activation. For sequence-axis sparsity, dimensions where the next-token activation wins the magnitude comparison are removed from the current token. Thus the future-to-past “trace” in this case is the pattern of zeroed-out dimensions in the current token, as seen in Figure 1. For quantization, the analogous trace is the pattern of quantization levels in the current token, rather than a pattern of zeroed dimensions.

3. Decoding. Downstream layers can learn to decode this trace. The training target at position t is the next token, so any repeatable trace left by that token is rewarded if it helps prediction. Over training, upstream layers can sharpen future-token representation so the operator leads to a more distinguishable trace, while downstream layers learn to map that trace back to the future token. At causal

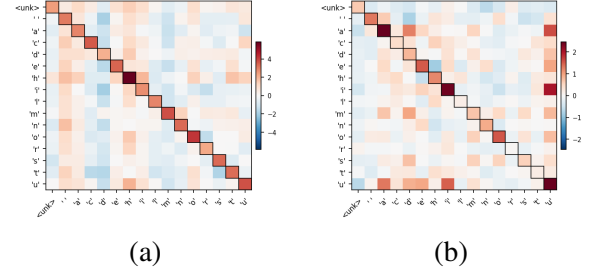


Figure 3: *Next-token prediction under exploitation.* Logit shift matrices ΔL for a toy character-level MLP under (a) 1:2 sequence-axis sparsity and (b) 2-bit block-scale quantization. Rows indicate the imposed future token in a bigram sequence and columns indicate the predicted token.

evaluation, the trace is no longer present in the same form.

3.3 Empirical Confirmation

If the model has learned to decode the trace left by the future token, then changing the future token while keeping the current token fixed should change the prediction in a specific way, namely the logit for the imposed future token should increase. A non-causal operator can change logits even without exploitation, because it changes the activations. The important test is whether those changes point toward the identity of the future token.

We test this on a character-level bigram MLP trained on WikiText-2, with vocabulary reduced to 16 characters for visual clarity (implementation details are in Appendix B). The model has no attention or other cross-token mixing, so the only path from the future token to the logits at position t is the non-causal operator itself. Without the operator, the logits at position t are a function only of the current token. Specifically, for a fixed current token, a bigram MLP produces the same logit vector regardless of which future token is placed next to it.

Let $\mathcal{V}$ denote the 16-character vocabulary. For each current token $x \in \mathcal{V}$ and each imposed future token $i \in \mathcal{V}$, we apply the non-causal operator to the pair (x, i) and record the logit vector at position t . Let $\ell_j^{\text{nc}}(x, i)$ be the logit for token j from a model trained with the non-causal operator, and let $\ell_j^{\text{c}}(x, i)$ be the same logit from a causally trained model evaluated under the same coupling. We measure

$$\Delta L[i, j] = \frac{1}{|\mathcal{V}|} \sum_{x \in \mathcal{V}} [\ell_j^{\text{nc}}(x, i) - \ell_j^{\text{c}}(x, i)]. \quad (1)$$

The row index is the imposed future token, and the column index is the output logit. The second term of the causally trained model, evaluated with the same operator, controls for perturbations that also appear without non-causal training.

If exploitation is learned, the diagonal should be elevated. In other words, the future token i leaves a trace that raises logit i . Figure 3 shows this exact pattern for both 1:2 sequence-axis sparsity and 2-bit block-scale quantization. Therefore, in this simplified setting, the result confirms the mechanism in its minimal form: the future token changes the past representation through the operator, and the trained downstream weights decode that change as evidence for the future token.

4 Attempted Mitigations

The mechanism in Section 3 shows that non-causal operators can create exploitable side-channels. This leaves a practical question: can we keep the training-time operator while making the side-channel too unstable or too small for the model to exploit?

4.1 Sparsity: Block Shuffle

For 2:4 sparsity, we randomize which tokens share a sequence tile at each training step by shuffling them. The motivation is simple: with fixed sequence tiles, position t always competes with the same relative future positions (e.g., $t+1, t+2, t+3$), so the same future-token representations induce stable mask patterns at t . This directly extends the mechanism we described in the previous section. Thus, we try to “break” these fixed position couplings.

In more detail, we divide the sequence into blocks of B consecutive tokens, randomly permute positions within each block, and form 2:4 tiles from the permuted order. The same side-channel remains, but the model no longer sees a fixed coupling between position t and specific future positions across iterations.

We evaluate shuffle sizes $B \in \{128, 256, 512\}$ (additional block sizes are reported in Section 6.2). Figure 4a shows that larger shuffle blocks weaken the effect. The onset of exploitation is delayed, and the loss gap is reduced. For these block sizes, training loss remains close to the baseline, so the effect is mostly visible through causal evaluation.

This is notable because the model has learned to extract future information through randomly as-

signed tiles, without any fixed coupling to rely on.

Large shuffle blocks introduce a separate risk: the onset can be delayed far beyond the early training window. Training may look normal for thousands of iterations, and a shorter run would make the mitigation appear effective. We revisit this delayed-onset behavior in Section 6.2, where a configuration that appears safe at the default model size fails after increasing model capacity.

4.2 Quantization: Block Size

For micro-scaled quantization, we focus on block size as the mitigation strategy, since this is a typical knob in quantization. The motivation here is different from sparsity. Increasing the block size does not only dilute the coupling by having more tokens share the same scale, but also leaves fewer scale decisions for the model to exploit.

We sweep 2D block sizes 16×16 , 32×32 , and 64×64 . Figure 4(b) shows the expected trend: smaller blocks exhibit early exploitation, while larger blocks delay onset and reduce the train/causal-evaluation gap.

However, block size is not a free knob. Increasing it also changes the quantization recipe and can degrade quantization quality independent of causality (NVIDIA et al., 2026; Dettmers and Zettlemoyer, 2023), since one scale must represent more values. This is visible for the 64×64 block size, where training loss also worsens.

Both strategies weaken the coupling and delay exploitation. Neither eliminates it without degrading model quality. Thousands of stable training steps are not evidence that causality has been restored, only evidence that the model has not yet learned to use the existing but weakened side-channel. Section 5 explains why no mitigation that preserves the non-causal operator can guarantee that causality has been restored.

5 Why Mitigations Fail

Section 4 showed that mitigations delay exploitation but do not prevent it. Here we explain why, and what would be required to remove the future-to-past path while keeping the operator in the forward pass.

5.1 Structural Opportunity

Fix a hidden dimension h , a past position t_p , and a future position t_f such that the operator’s output

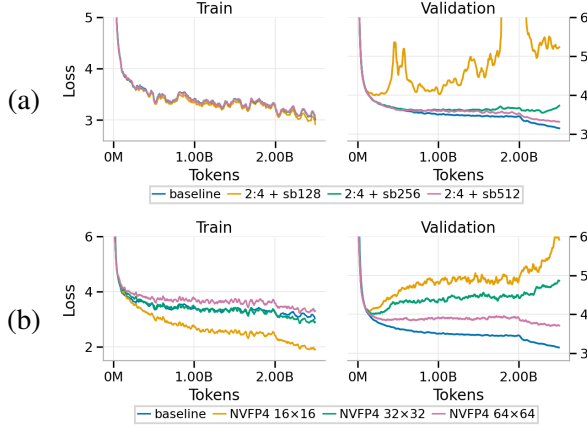


Figure 4: Training loss vs. causal evaluation loss under (a) block shuffle at varying block sizes and (b) increasing quantization block size. Both delay exploitation but do not remove the future-to-past path.

at t_p can depend on the activation at t_f . The operator produces an output at t_p that enters the forward computation. Changing t_f 's activation can change that output.

Intervention. To isolate the effect of t_f , we compare two scenarios for dimension h . In *include*, t_f 's activation enters the operator's shared statistic. In *exclude*, it is removed. Let $O_{t_p}^h(\text{incl})$ and $O_{t_p}^h(\text{excl})$ denote the resulting output at t_p under each scenario.

The disagreement event. The future position matters only on inputs where the two outputs differ. We call this event

$$\mathcal{D} = \{O_{t_p}^h(\text{incl}) \neq O_{t_p}^h(\text{excl})\}, \quad (2)$$

with probability $\delta = P(\mathcal{D})$.

Instantiations. For N:M sparsity, $\mathcal{D}$ is the set of inputs where including t_f changes whether t_p survives the magnitude ranking. For block quantization, $\mathcal{D}$ is the set where including t_f changes the scale enough to change the rounded value at t_p . Appendix A.2 gives the precise forms of $\mathcal{D}$ and δ for both operators.

Persistent exposure. The condition $\delta = 0$ requires the operator's output at t_p to be invariant to t_f 's activation for every input. For sparsity, the prune/keep condition would never change for t_p because of future tokens. For quantization, the quantized value at t_p would never change because of a future maximum. We focus on positive exposure, where $\delta > 0$. This is the ordinary case

where activation magnitudes overlap across positions. The future position can then change the ranking in a sparsity tile or the maximum in a quantization block, and the output at t_p changes on those inputs. Shuffling changes which positions compete, but it does not remove competition within each realized tile or block.

5.2 Exploitation Cannot Be Eliminated

For a dimension with $\delta > 0$, we compare the loss under the two interventions.

The factorization. Let $\ell(t_p)$ denote position t_p 's contribution to the training loss. Define

$$\Delta \triangleq E[\ell(t_p) \mid \text{incl}] - E[\ell(t_p) \mid \text{excl}]. \quad (3)$$

On $\mathcal{D}^c$ the two interventions produce the same output at t_p , so their loss contributions are identical. Splitting on $\mathcal{D}$ and $\mathcal{D}^c$ gives

$$|\Delta| = \delta \cdot |g_{\mathcal{D}}(\text{incl}) - g_{\mathcal{D}}(\text{excl})|, \quad (4)$$

Here

$$\begin{aligned} g_{\mathcal{D}}(\text{incl}) &= E[\ell(t_p) \mid \text{incl}, \mathcal{D}], \\ g_{\mathcal{D}}(\text{excl}) &= E[\ell(t_p) \mid \text{excl}, \mathcal{D}] \end{aligned}$$

These are the expected loss contributions at t_p under each intervention, conditioned on $\mathcal{D}$. The first factor δ is how often the future position changes the past position's outcome. The second factor is how much that change affects the loss.

The inactivity condition. Setting $|\Delta| = 0$ requires one of the two factors to vanish. When $\delta > 0$, only the second factor can vanish. The condition is

$$g_{\mathcal{D}}(\text{incl}) = g_{\mathcal{D}}(\text{excl}) \quad (5)$$

This requires the loss at t_p to be insensitive to which output the operator produces at dimension h , conditioned on $\mathcal{D}$. This is the inactivity condition used here. For sparsity, retaining or zeroing the dimension produces no difference in loss. For quantization, changing the grid produces no difference in loss.

Outcome. For any hidden dimension h under a non-causal operator with $\delta = P(\mathcal{D}) > 0$,

$$|\Delta| = 0 \iff \text{dimension } h \text{ is inactive on } \mathcal{D}.$$

Eliminating the side-channel at h therefore requires representational inactivity on the disagreement set.

5.3 Implications

Mitigations. The factorization explains why the strategies of Section 4 delay exploitation without preventing it. Shuffling changes which positions share a tile, reducing how often the same pair is exposed. Moreover, even if we directly shrink the set of affected dimensions, for example by applying the operator to fewer dimensions, each remaining active dimension still has $|\Delta| > 0$. Both strategies reduce the aggregate exploitation signal without eliminating it.

Correlated positions amplify the incentive. Adjacent tokens often produce correlated activations (Haziza et al., 2025). When their magnitudes are close, small changes can alter the mask ranking and make the disagreement event more frequent. This gives intuition for why adjacent sequence tiling can be a strong practical case, but it is not a separate assumption of the factorization.

Training dynamics. The factorization identifies a residual loss signal, but it does not predict when exploitation becomes observable. Mitigations can shrink the signal enough that ordinary causal learning dominates for many steps. The delayed onset in Section 4 is consistent with this picture. Stable early training does not imply that the side-channel has been removed.

Avoiding exploitation requires removing the operator. The structural fix is to keep the non-causal operator out of the forward pass. One can still use a separate non-causal operator in the backward pass to retain hardware acceleration for some GEMMs. This forward-backward inconsistency may carry a quality cost, but it removes the side-channel from the computation used for prediction.

6 Further Exploration

6.1 Setup

We use a decoder-only transformer following NanoChat (Karpathy, 2025) (~ 162 M parameters, 12 layers, $d=768$) with squared-ReLU activations, trained on Nemotron-CC-v2 (NVIDIA, 2025a) under bfloat16 precision (details are in Appendix B).

In this section, for simplicity, we focus on the sparsity setup of Section 2: transposable 2:4 mask during training and hidden-axis 2:4 mask during validation. Reported results include relative training and evaluation losses, computed as

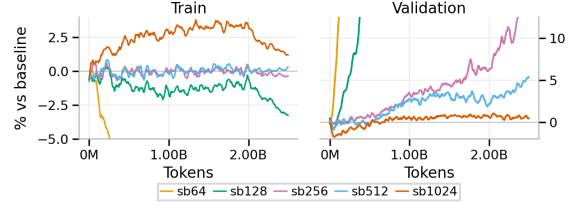


Figure 5: *Block Shuffling*: different block sizes control the exploitation onset and intensity (relative losses). Block size 1024 shows no signs of exploitation at this model size.

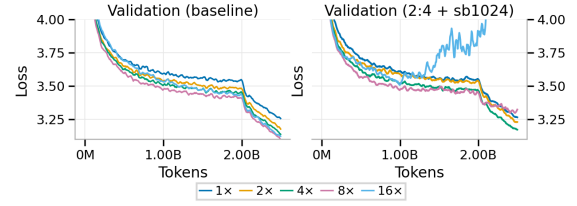


Figure 6: *Increasing expansion factor* (and thus the model size) under the block shuffle 1024 mitigation leads to eventual exploitation.

$(\mathcal{L} - \mathcal{L}_{\text{baseline}})/\mathcal{L}_{\text{baseline}}$ against the causal baseline. Additional explorations (different mitigations, quantization experiments, recovery from a exploited model etc.) can be found in Appendix C.

6.2 Illusion of Safety

We extend the exploration from Section 4. Figure 5 shows larger blocks delay exploitation as before. Notably, at $B=1024$, no exploitation is visible. However, per Section 5, this can be misleading. For this model size and token horizon we may not see any exploitation signs, but a larger model, a different learning rate, or a longer run may eventually show exploitation.

To demonstrate this, we increase the MLP expansion factor, giving the mechanism more dimensions to work with. We compare factors from 1x to 16x (4x is the default). Figure 6 shows that dense baselines follow the expected trend: larger models achieve lower validation loss. Under non-causal 2:4 sparsity, however, only the 16x model shows clear exploitation. A recipe working at 4x completely fails at 16x, supporting the theoretical result that the future-to-past path remains whenever the non-causal operator is kept, and more dimensions make exploitation easier. The 8x case is more subtle: it behaves as a successful run but shows a small regression during the decay phase, easily missed without awareness of the failure mode.

Model	Generation
Prompt: <i>The capital of France is Paris. The capital of Germany is Berlin. The capital of Italy is</i>	
Dense	Rome. The capital for Italy be Berlin. The reason why this type of language that is can will not use it ...
2:4 hidden (causal)	Rome. The city and the main areas are all in English, German. The town and major areas including both English and German...
2:4 sequence (non-causal)	(end of text)
2:4 sequence + sb1024	You canSending an1, "Ages: the. A good question for / Matter-Crowned / When you / Pilas (Byzanthes
Prompt: <i>To make a cup of tea, first boil water. Then</i>	
Dense	pour cold water into milk, then add warm milk to cream and then move cold ...
2:4 hidden (causal)	draw out one more stem from the pot to pick up some leaves and bring them in for good use at any time...
2:4 sequence (non-causal)	(end of text)
2:4 sequence + sb1024	What is one-way, theCAM in / So you LITIEmpA5C wrote: A question forThe / BENEFITS OF ARTICLES

Table 1: *Generation Quality*. Completions for two simple prompts under causal and non-causal sparse training.

Masking	Train	Val.	HS	ARC-C
Base checkpoint	–	–	0.617	0.358
Hidden (causal)	2.418	2.411	0.609	0.364
Sequence + sb512	2.427	2.695	0.581	0.349
Sequence + sb256	2.377	3.232	0.511	0.339
Sequence + sb128	1.845	3.874	0.345	0.282
Sequence + sb64	1.726	4.969	0.317	0.274

Table 2: *Fine-tuning Llama 3.2 1B*. Training and causal-validation losses together with HellaSwag (HS) and ARC-Challenge (ARC-C) results.

6.3 Generation Quality

The causal-evaluation loss captures the degradation caused by exploitation, but does not directly show how this appears during generation. We therefore compare simple autoregressive completions from the dense model, the causal hidden-axis 2:4 model, the unmitigated sequence-axis 2:4 model, and block shuffle 1024.

Even at this relatively small model size, the effect is qualitatively clear. The dense and causal sparse models produce imperfect but plausible prompt-related continuations, while the exploited sequence-axis model immediately terminates.

A more interesting case is block shuffle 1024. Its causal validation loss is 3.16, compared with 3.14 for the causal hidden-axis model, yet the generations are already clearly degraded. Thus, a small causal-evaluation gap can hide a much more visible degradation in generation quality.

More detailed generation results show how this degradation evolves under different mitigation strengths in Appendix E, together with a detailed description of the decoding setup.

6.4 Experiments with Llama 3.2 1B

The previous experiments mostly use a 162M-parameter setup to study the failure mode and its training dynamics. We now verify that the widely used Llama 3.2 1B architecture (Grattafiori et al., 2024) has the same behavior. We focus on fine-tuning from the pretrained checkpoint, which provides a stronger starting model and allows us to evaluate downstream quality on widely used benchmarks. We fine-tune the pretrained checkpoint for 5B tokens on the same Nemotron-CC-v2 dataset used in our previous exploration (NVIDIA, 2025a), using a learning rate of 10^{-4} with cosine decay.

Table 2 shows the same train/evaluation divergence and block-shuffle trend observed in the smaller models. The causal hidden-axis model remains close to the pretrained checkpoint, while smaller block-shuffle sizes produce progressively lower training loss but worse causal validation loss.

This degradation also transfers to downstream evaluation. We report zero-shot length-normalized accuracy on HellaSwag (Zellers et al., 2019), a commonsense sentence-completion benchmark, and ARC-Challenge (Clark et al., 2018), a multiple-choice science reasoning benchmark. Their behavior follows the same ordering as validation loss, showing that the observed divergence corresponds to a broader loss of model quality.

Training-from-scratch results, full experimental details, and additional Llama generations are reported in Appendices D and E.

6.5 Forward Bypass

Following Section 5, we use a forward bypass. The non-causal operator is removed from the for-

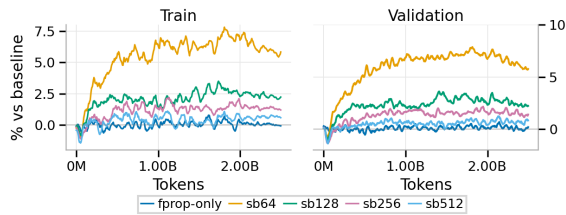


Figure 7: *Forward bypass* for 2:4 sparse activations under different block shuffle sizes (relative losses).

ward pass and used only in the backward pass. This is already the norm in NVFP4 quantization (NVIDIA et al., 2026) and recently introduced in sparsity (Haziza et al., 2025). In both cases the motivation is throughput, not causality. Focusing on the sparsity case, we decouple masks by where they are used.

- 1) 2:4 along the hidden dimension, per-token (causal). Used in FC2 FPROP and FC1 DGRAD.
- 2) 2:4 along the sequence dimension, across tokens (non-causal). Used in FC1 and FC2 WGRADs.

Both masks are computed with respect to the activation magnitudes.

Since sequence-axis masks pair highly correlated tokens (Section 5.3, Haziza et al. (2025)), we apply block shuffling (see Section 4) to decorrelate pairings and reduce gradient bias. Figure 7 shows that losses approach the dense baseline as shuffle size increases. However, even at block shuffle 512, a small gap persists, attributable to the forward–backward mismatch. Using a single causal mask (hidden-axis 2:4) for all GEMMs does not show this gap, confirming that the residual is due to the inconsistency, not the sparsity itself. Direct gradient comparisons characterizing this forward–backward mismatch are provided in Appendix C.

7 Discussion and Conclusions

Relation to prior work. The causality violations treated here are one form of lookahead bias in next-token prediction. This issue is familiar in forecasting, where future information can enter the training signal unless the data and evaluation protocol are time-aware (Yan et al., 2026). Our focus is different: the future information enters through an efficient operator inside the model computation, not through the dataset.

Cross-batch dynamic quantization is another case where a runtime statistic creates leakage, but there the concern is information passing across independent samples or users (Foerster et al., 2026).

A more closely related point in quantization is block-quantized attention, where MatX reports future leakage from quantized value blocks and gives a local fix for that setting (Mishra et al., 2026).

Another subtle non-causal operator is expert-choice routing. It improves load balance by letting experts select tokens (Zhou et al., 2022), but cross-token selection would break autoregressive decoding. Token-choice and recent autoregressive MoE variants avoid this dependence (Fedus et al., 2022; Zhong et al., 2024; Sun et al., 2026).

Even a tensor-wide scale in two-level quantization is formally non-causal if it is computed from current activations that include future positions (NVIDIA et al., 2026). Since this path is carried by only one scalar for the whole tensor, exploitation should be negligible. Delayed scaling already avoids it by using previous-step amax history (Micikevicius et al., 2022; NVIDIA, 2026), although it was introduced for efficiency and numerical stability, not as a causal fix.

Notably, modern NVFP4 and activation-sparsity recipes already use forward-bypass designs for throughput (NVIDIA et al., 2026; Haziza et al., 2025), similar to the recipe discussed in Section 6.5.

Open questions. What determines exploitation onset remains open. The structural result shows the opportunity, but the dynamics are observed empirically, not predicted. A practical alternative is to use non-causal operators early for throughput and switch to a forward-bypass recipe once an internal metric detects exploitation. This would require a reliable signal before the loss divergence appears, which we do not yet have. As models and training budgets grow, side-channels that look too small to matter deserve scrutiny.

Takeaway. Any sequence-axis statistic used in the forward pass creates an exploitable future-to-past path. Mitigations can delay exploitation and hide it in short or small-scale runs, but the path remains. The fix is architectural: use a causal alternative when available, or decouple forward and backward operators otherwise to remove the causality violation from the forward path.

8 Limitations

Degenerate cases. The main outcome of Section 5 says that closing the future-to-past path requires the affected dimensions to be inactive on the dis-

agreement set, which creates a capacity cost. This assumes that those dimensions contribute to predictions on those inputs. We do not rule out parameterizations where a dimension appears active by our measure but carries no exploitable signal, for example because of cancellation downstream, although we do not observe such cases empirically. *Opportunity, not inevitability.* The structural result shows that the optimizer can receive an exploitation signal (Section 5), but it does not predict if or when training will follow it. Onset timing, severity, and interactions with learning-rate schedules remain empirical.

Detection. The empirical signature (divergence between training and causal-evaluation loss) is observable only after exploitation has already degraded the model. We have not identified an internal metric that reliably signals exploitation before it appears in evaluation, which limits early intervention.

Forward-backward mismatch. The forward-bypass recipe (Section 6.5) removes the future-to-past path from the forward computation, but the gradients are still computed with a different operator. While losses approach the dense baseline in our runs (Figure 7), the long-term quality and throughput effects of this mismatch at scale remain open.

References

- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Tim Dettmers and Luke Zettlemoyer. 2023. [The case for 4-bit precision: k-bit inference scaling laws](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7750–7774. PMLR.
- William Fedus, Barret Zoph, and Noam Shazeer. 2022. [Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity](#). *Journal of Machine Learning Research*, 23(120):1–39.
- Hanna Foerster, Ilia Shumailov, Cheng Zhang, Yiren Zhao, Jamie Hayes, and Robert Mullins. 2026. [Quantamination: Dynamic quantization leaks your data across the batch](#). *Preprint*, arXiv:2604.26505.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Daniel Haziza, Timothy Chou, Dhruv Choudhary, Luca Wehrstedt, Francisco Massa, Jiecao Yu, Geonhwa Jeong, Supriya Rao, Patrick Labatut, and Jesse Cai. 2025. [Accelerating transformer inference and training with 2:4 activation sparsity](#). *Preprint*, arXiv:2503.16672.
- Itay Hubara, Brian Chmiel, Moshe Isard, Ron Banner, Joseph Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find N:M transposable masks. In *Advances in Neural Information Processing Systems*, pages 21099–21111.
- Andrej Karpathy. 2025. nanochat: The best ChatGPT that \$100 can buy. <https://github.com/karpathy/nanochat>.
- Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. [Datasets: A community library for natural language processing](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.
- Meghana Madhyastha, Daniel Haziza, Jesse Cai, Newsha Ardalani, Zhiqi Bu, and Carole-Jean Wu. 2026. [To 2:4 sparsity and beyond: Neuron-level activation function to accelerate LLM pre-training](#). *Preprint*, arXiv:2602.06183.
- Paulius Micikevicius, Dusan Stolic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, Naveen Mellempudi, Stuart Oberman, Mohammad Shoeybi, Michael Siu, and Hao Wu. 2022. [FP8 formats for deep learning](#). *Preprint*, arXiv:2209.05433.
- Akshay Mishra, Reiner Pope, Sanjit Neelam, Daniel Heinlein, Vaclav Cvicek, Zaal Vasania, and James Hill-Khurana. 2026. Future leakage in block-quantized attention. https://matx.com/research/leaky_quantization.
- Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stolic, Dusan Stolic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. [Accelerating sparse deep neural networks](#). *Preprint*, arXiv:2104.08378.
- NVIDIA. 2025a. Nemotron-CC-v2: A high-quality web pre-training corpus. <https://huggingface.co/datasets/nvidia/Nemotron-CC-v2>.
- NVIDIA. 2025b. NVIDIA Blackwell architecture. <https://www.nvidia.com/en-us/data-center/technologies/blackwell-architecture/>.

- NVIDIA. 2026. FP8 delayed scaling. https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/features/low_precision_training/fp8_delayed_scaling/fp8_delayed_scaling.html. Transformer Engine documentation.
- NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Muya Chang, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, and 71 others. 2026. *Pretraining large language models with nvfp4*. *Preprint*, arXiv:2509.25149.
- Open Compute Project. 2023. *Ocp microscaling formats (mx) specification*.
- OpenAI. 2022. tiktoken: Fast BPE tokeniser for use with OpenAI’s models. <https://github.com/openai/tiktoken>.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. *Pytorch: An imperative style, high-performance deep learning library*. *Preprint*, arXiv:1912.01703.
- Bitu Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Stosic Dusan, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, and 14 others. 2023. *Microscaling data formats for deep learning*. *Preprint*, arXiv:2310.10537.
- Hanchi Sun, Yixin Liu, Yonghui Wu, and Lichao Sun. 2026. *Expert threshold routing for autoregressive language modeling with dynamic computation allocation and load balancing*. *Preprint*, arXiv:2603.11535.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.
- Yutong Yan, Raphael Tang, Zhenyu Gao, Wenxi Jiang, and Yao Lu. 2026. *DatedGPT: Preventing look-ahead bias in large language models with time-aware pretraining*. *Preprint*, arXiv:2603.11838.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800.
- Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. 2024. *Lory: Fully differentiable mixture-of-experts for autoregressive language model pre-training*. *Preprint*, arXiv:2405.03133.
- Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. 2022. *Mixture-of-experts with expert choice routing*. In *Advances in Neural Information Processing Systems*.

A Additional Analysis

A.1 Pairwise Case

The argument in Section 5 selects one pair (t_p, t_f) sharing a tile at dimension h . The resulting $|\Delta_{t_p, t_f}|$ does not describe all future-to-past dependence at (t_p, h) . In a tile of size 4, multiple future positions may compete with t_p , and their effects on the mask are not additive. For example, once two future positions occupy the top-2 slots, a third cannot further change t_p 's outcome. The pairwise quantity should therefore be read as a sufficient case, not as a full accounting of the tile.

This is enough for the main outcome. One pair with $\delta > 0$ and $g_{\mathcal{D}}(\text{incl}) \neq g_{\mathcal{D}}(\text{excl})$ already gives a nonzero loss signal.

A.2 Application to Specific Operators

We spell out the disagreement event for the two operators used in the paper. Fix a hidden dimension h , a past position t_p , and a future position t_f that shares a sparsity tile or quantization block with t_p .

Sequence-axis 2:4 sparsity. The two interventions differ only when t_p lies on the keep/drop boundary. If t_p is not the top value, but would still be retained when t_f is excluded, then including t_f can push t_p out of the top two. Equivalently, over the four positions in the tile, the disagreement event is

$$\mathcal{D} = \{ t_p \text{ ranks 3rd and } t_f \text{ ranks 1st or 2nd} \}.$$

This event has nonzero probability whenever activation magnitudes overlap across positions. Under the simple exchangeable case where all 4! orderings are equally likely, exactly four orderings satisfy $\mathcal{D}$, giving $\delta = 1/6$ for a fixed future position t_f .

Block shuffling reassigns tile membership at each step, so the same pair of positions is not coupled every time. This weakens the signal the optimizer can accumulate, but each realized tile still has the same kind of disagreement event.

Block-scaled activation quantization. Quantization is more nuanced. The straightforward analog of the sparsity case is to count the inputs where t_f holds the block maximum, since only then does removing it change the scale at t_p . This is only part of the story. A different scale need not produce a different quantized value, because both scales may round $a_{t_p}^h$ to the same grid level.

The output we care about is the rounded value itself, so the disagreement event is

$$\mathcal{D} = \{ \hat{a}_{t_p}^h(\text{incl}) \neq \hat{a}_{t_p}^h(\text{excl}) \}$$

where $\hat{a}^h = s \cdot Q(a^h/s)$ is the quantized value under scale s , while Q rounds to the format's grid.

As a simple example, consider a sequence-wise block of B positions with exchangeable activation magnitudes. A fixed future position sets the block scale with probability $1/B$. Since the rounded value at t_p can only change when the scale changes, this gives the simple bound

$$\delta \leq \frac{1}{B}.$$

Equality would require every scale change caused by t_f to also change the rounded value at t_p , which need not hold for a real quantization grid. The exact probability changes with the rounding levels, 2D blocks, or non-exchangeable activations, but the role of block size is the same. Larger blocks make any fixed future position less likely to control the scale. This matches Section 4.2, where block-size mitigations weaken exploitation but do not eliminate it.

A.3 Assumptions

The argument in Section 5 uses two simple assumptions.

Positive exposure. The future position must sometimes change the past position's forward value.

$$\delta = P(\mathcal{D}) > 0.$$

For sparsity, this means that including the future activation can change whether the past activation is kept or pruned. For quantization, it means that including the future activation can change the rounded value used at the past position. Shuffling, larger blocks, or partial application can reduce how often this occurs, but they do not remove the event as long as the forward statistic still spans future positions.

Prediction sensitivity. Conditioned on the disagreement event, the two possible operator outputs must have different expected losses.

$$g_{\mathcal{D}}(\text{incl}) \neq g_{\mathcal{D}}(\text{excl}).$$

This is the ordinary case when the affected dimension contributes to the prediction. Equality can

hold in edge cases. Later computation may ignore the dimension, or loss changes may balance exactly in expectation. Ignoring the dimension is a capacity cost. Exact balance is possible, but it is not enforced by shuffling, larger blocks, or partial application.

What the outcome claims. With these assumptions, the factorization

$$|\Delta| = \delta \cdot |g_{\mathcal{D}}(\text{incl}) - g_{\mathcal{D}}(\text{excl})|$$

shows that a non-causal forward operator leaves a nonzero loss signal. Mitigations can shrink this signal by reducing exposure or reducing the number of affected dimensions, but they do not remove the side-channel. Removing it requires a causal forward computation or affected dimensions that stop mattering on the disagreement set.

The factorization is written for one affected dimension. At the model level, many such dimensions can contribute. Aggregation could cancel the effect if the downstream weights combine changed dimensions so that their logit or loss changes balance on the same inputs. This is possible in principle, but it is not a general mitigation because the operator does not enforce that balance.

A.4 Training Dynamics

The factorization in Section 5 does not depend on the operator details. If the forward operator lets future activations change past representations, then the side-channel is present. The training dynamics do depend on the operator. After mitigations such as block shuffling, the fixed future-to-past pairing is removed and the remaining signal is weak, so exploitation depends on how the operator makes the future information readable.

For sparsity, the side-channel is a keep/drop decision. The relevant boundary is the 2:4 survival boundary, where a future activation changes whether a past activation falls above or below the top-2 cutoff. When this happens, the representation changes discontinuously because the dimension is either kept or zeroed. A useful sparse signal therefore comes from binary mask patterns that are stable enough for downstream layers to decode. Lower learning rates can make this easier to maintain because the binary decisions change more slowly.

For block-scaled quantization, the relevant boundary is a rounding boundary. A future activation must change the block scale enough to move

a past activation to a different quantization level. Being near a rounding threshold is not sufficient by itself, because the scale change must also cross that threshold. At the same time, a very large scale change is not necessarily useful, since it can shift many values together. The useful regime is more selective. Scale changes should move a subset of dimensions across rounding boundaries while leaving enough of the representation stable for downstream layers to decode the change. This gives a different learning-rate dependence. Larger learning rates can create more scale-induced level changes, while decay can reduce the number of useful crossings.

This gives a working interpretation of the training dynamics, not a prediction of onset. Sparsity and quantization expose the same kind of non-causal path, but their useful regimes differ. Sparsity can use small, stable changes around a binary survival boundary. Quantization needs enough movement to create level crossings, but not so much movement that the change becomes broad and hard to decode. The additional experiments below support this interpretation through learning-rate sweeps and quantization block-shuffling runs.

A.5 Internal Metric

The empirical signature used in the paper is a divergence between training loss and causal-evaluation loss. This is useful, but late. By the time the gap appears, the model has already started relying on the non-causal path. A useful internal metric would detect this earlier. It could also answer a practical question we do not resolve here, namely whether a non-causal operator can be used safely for some part of training, before the model learns to exploit it.

We tried simple metrics based on the factorization in Section 5. The first factor, exposure, is easy to measure. On a training batch, run the operator with and without the future positions included in its statistic, then mark the entries where the outputs disagree. For sparsity, these are mask bits that flip; for quantization, these are entries whose rounded value changes. This can be computed online with little extra work.

The second factor is harder because it asks whether the changed entries affect the loss. As a rough proxy, we measured activation magnitude on the disagreement set. The intuition is simple: if many large activations sit exactly where the two operator evaluations disagree, the model has more room to use the side-channel. In early experiments,

this was not reliable enough to use as a standalone warning signal before the loss gap appeared. Ordinary useful representations can also put mass on disagreement entries, and we did not find a clear reference threshold.

One might also try to turn such a metric into a regularizer, if it is differentiable. This would not contradict the outcome in Section 5. A regularizer that suppresses the second factor is pushing the model to make the affected entries irrelevant to the loss. With the non-causal forward operator still present, this is the inactivity route identified by the theory, not removal of the side-channel.

B Experimental Setup

We use two setups. The toy model isolates the mechanism, while the transformer runs test whether the same effect appears in a realistic autoregressive training loop. All models are trained from scratch, unless stated otherwise.

Toy bigram MLP. The mechanism experiment uses character-level WikiText-2² with a vocabulary of the 15 most frequent printable characters plus `<unk>` ($V=16$). We train a single-block residual MLP with hidden dimension $H=64$, LayerNorm, LeakyReLU, tied output embeddings, and no attention or positional encoding. Each position predicts the next token from its own embedding alone, so the non-causal operator is the only cross-token path. The operator is placed between FC_1 and FC_2 on post-activation hidden states and groups consecutive positions within each batch row. We train for 8,000 steps with batch size 256 using AdamW ($\beta_1=0.9$, $\beta_2=0.95$, $\text{lr } 3 \times 10^{-3}$, weight decay 0.1 on $\text{dim} \geq 2$ parameters).

Transformer runs. The main experiments use a NanoChat-inspired (Karpathy, 2025) decoder-only transformer with 12 layers, $d=768$, 6 attention heads, RoPE, RMSNorm, untied embeddings, no bias terms, and $d_{\text{mlp}}=3072$ with ReLU² activations ($\approx 162\text{M}$ parameters). We train on the High-Quality split of Nemotron-CC-v2³ packed to sequence length $S=4096$, using a 90/10 train/validation split. The default budget is 2.5×10^9 tokens. Optimization uses AdamW with $\beta_1=0.9$, $\beta_2=0.95$, weight decay 0.1, gradient clipping at norm 1.0, and a WSD schedule with 1% warmup, peak learning

rate 6×10^{-4} , and final 20% decay to one tenth of the peak rate. Runs use 8 GPUs with DDP and bfloat16 forward passes.

Considered operators. The main experiments ablate the two non-causal operators from Section 2, namely (i) 2:4 sequence-axis sparsity on all FC_2 input activations, and (ii) NVFP4 (e2m1) activation quantization with block scales applied to all linear layers. Unless a paragraph states otherwise, appendix proof-of-concept experiments use the sequence-wise version of each non-causal operator rather than the 2D version. In these cases, validation removes the operator completely, and the baseline is the dense BF16 run.

Run protocol. Most experiments are single-run results. This matches the goal of the study because one clear exploitation event is enough to establish that the failure mode exists. We did not rerun experiments to search for a failure. The exception is the forward-bypass experiment, which evaluates a practical mitigation and reports averages over three runs.

Implementation and third-party resources. Experiments are implemented in PyTorch (Paszke et al., 2019). Data loading uses Hugging Face Datasets (Lhoest et al., 2021), and GPT-2 BPE tokenization uses tiktoken (OpenAI, 2022). Operator and evaluation details are given with the corresponding experiments.

Compute. Experiments were run on a cluster of NVIDIA H100 80GB GPUs. Each transformer run used one node with 8 GPUs. The average cost was about 18 GPU-hours per run, and the total compute across all experiments was about 2,000 GPU-hours.

C Additional Experiments

This section includes additional experiments on mitigation strategies, training dynamics, and recovery from exploited models.

Run-to-Run Variability The structural argument in Section 5 establishes the presence of an exploitable future-to-past path, but does not determine when exploitation becomes observable during training. As noted in the previous section, we mostly present single-run experiments to establish that the failure mode exists. This leaves open how sensitive the exploitation dynamics are to the particular training data seen by the model.

²WikiText-2 raw train split, CC BY-SA 4.0.

³Nemotron-CC-v2 is released by NVIDIA under its data access agreement. We load datasets through Hugging Face Datasets and tokenize with GPT-2 BPE via tiktoken.

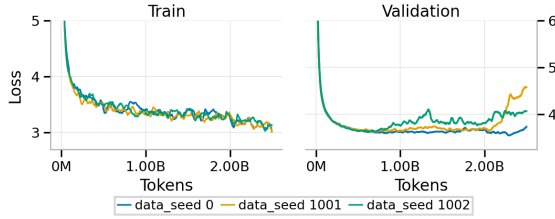


Figure 8: *Run-to-run Variability* under different data sequences.

We therefore repeat, for the base 162M model, the sequence-axis 2:4 experiment with block shuffle 256 using different data-shuffling seeds. The dataset is substantially larger than the 2.5B-token training subset, so different shuffling seeds expose the model to partially different training data. In contrast, changing only the model initialization seed led to nearly identical behavior, with no notable variation in the observed dynamics.

Figure 8 summarizes the results. The onset and final severity vary across runs, but all three eventually exhibit the same train/evaluation divergence. This is consistent with the distinction made in Section 5: the future-to-past path provides the opportunity for exploitation, while the observed onset and severity depend on the training dynamics, with the training data being one source of variation.

Conditional Random 2:4 Consider the sparsity case. To obtain a mask that is 2:4 along both axes, we first apply 2:4 along the hidden dimension (causal, per-token), then randomly zero additional entries along the sequence dimension wherever a column has fewer than 2 zeros. This produces a mask with $>50\%$ sparsity. At evaluation, only the hidden-axis mask is kept, so the random fill acts as a dropout mechanism. Figure 9 shows weaker exploitation than the naive 2D transposable case, but exploitation still emerges. The result is initially counterintuitive, since the fill is random, but the conditional decision of whether a dimension needs additional zeros still exposes enough future-dependent information for the model to exploit.

Partial Sparsity. The main text argues that a wider side-channel gives the model more degrees of freedom to exploit. A direct way to test this in sparsity is to make the side-channel narrower by sparsifying only a fraction p of the activation dimensions. For this proof-of-concept setting, the sparsity mask is sequence-wise only, while validation removes the operator and uses the dense coun-

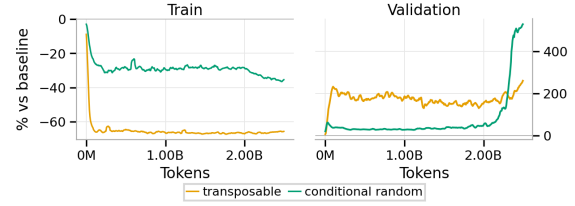


Figure 9: *Conditional random fill* versus the transposable mask.

terpart. At each iteration, we choose a different random subset of dimensions according to the selected fraction. The results are shown in Figure 10. Using fewer dimensions under sequence-wise 2:4 sparsity weakens the effects of broken causality. At 25%, we see practically no measurable exploitation. This comes at the cost of the acceleration benefits of the full 2:4 configuration, so it is not a practical solution. It instead supports the hypothesis that narrowing the side-channel controls the exploitation signal.

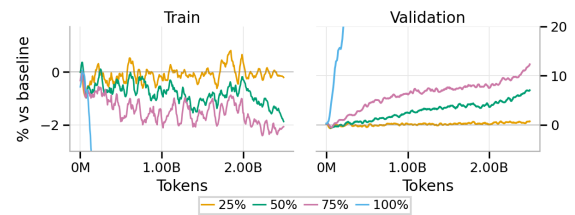


Figure 10: *Partial Sparsification*. Relative loss to the dense baseline. 100% is the typical 2:4 case.

Activation Function. The use of 2:4 sparsity on activations is motivated by the natural sparsity of Squared-ReLU. Section 5 explains why this is itself a form of mitigation. When the imposed pattern aligns with existing zeros, fewer dimensions are contested and the exploitation signal weakens. We test this by replacing Squared-ReLU with GELU and Leaky-ReLU under block shuffle 256. Figure 11 supports the hypothesis. Squared-ReLU better masks the divergence through natural inactivity, while GELU and Leaky-ReLU expose it.

Quantization Block-size. As discussed in the main text, NVFP4 with 2D blocks shares the scale over both sequence and hidden dimensions, which is the practical configuration we study. We sweep block size, since larger blocks share one scale across more activations and reduce the number of independent scale decisions available to carry future information. Figure 12 shows the relative loss

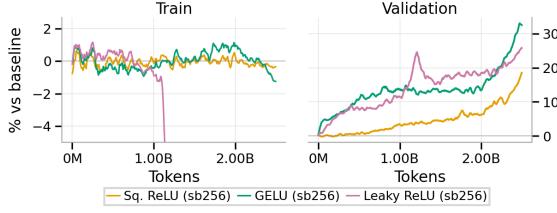


Figure 11: *Different activations have different behaviors. Naturally sparse ReLU-based activations hide the issue.*

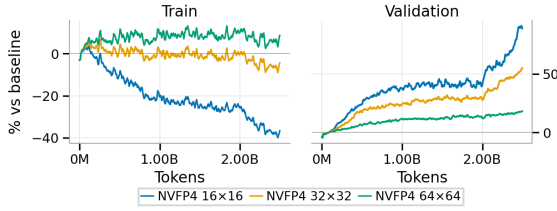


Figure 12: *NVFP4 + 2D block-size. Relative loss to the BF16 baseline.*

to the BF16 baseline. This is the relative-loss version of the results in Figure 4.

To make exploitation more visible, we also run 1D sequence-wise blocks, where tokens in the same sequence block directly share a scale. This gives more scale decisions than the 2D setting. We sweep block size here as well, with results in Figure 13. We observe the same pattern. Increasing block size weakens the effect, especially in training loss, but the validation divergence remains significant. Part of the loss increase at large blocks may also come from worse quantization quality. We do not see an obvious qualitative difference between 1D and 2D blocks, despite the 1D version using more quantization scales.

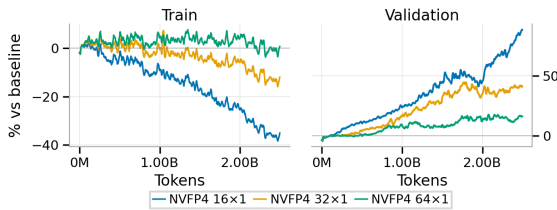
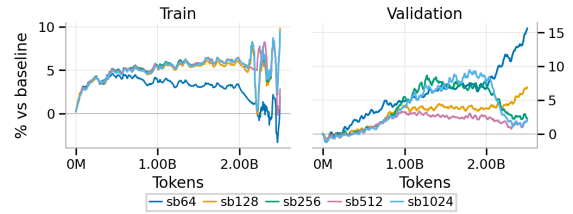


Figure 13: *NVFP4 + 1D block-size. Relative loss to the BF16 baseline.*

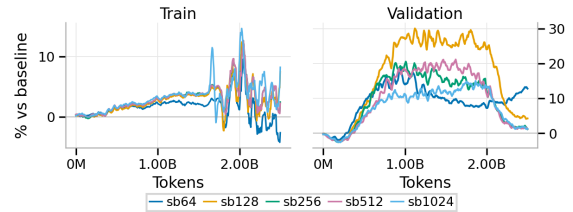
Quantization & Block Shuffling. In the main text we use block size as the main mitigation strategy for quantization. Block shuffling also applies to the NVFP4 16 $\times$ 16 recipe. The idea is to randomize which tokens share a scale, so fixed couplings are harder to use. Figure 14(a) shows the block-

shuffling sweep for the default transformer model. As we increase the block-shuffle size, the broken-causality effects weaken, but not in the same way as in sparsity. Here larger shuffle blocks lead to early instability, consistent with exploitation, but the loss gap shrinks during the decay phase. This suggests different training dynamics than sparsity.

Following Section 6.2, we also include a larger model to test whether larger models make exploitation easier. We consider the same block-shuffle sweep with a 16 $\times$ MLP expansion factor. The results are summarized in Figure 14(b). The exploitation effects are stronger for this larger model, with higher validation losses. However, we still see the same pattern of the gap shrinking during the decay phase.



(a) 4 $\times$ MLP expansion



(b) 16 $\times$ MLP expansion

Figure 14: *Block shuffling sweep for 2D quantization. Relative losses for the NVFP4 16 $\times$ 16 recipe with (a) the default 4 $\times$ MLP expansion factor and (b) a 16 $\times$ MLP expansion factor.*

Learning-rate Sensitivity. To better understand the training dynamics, we vary the learning rate for both (a) 2:4 sparsity and (b) NVFP4 quantization. Figure 15 reports the resulting dynamics.

We see an interesting pattern. In this sweep, smaller learning rates make sparsity exploitation appear sooner, while higher learning rates lead to faster quantization exploitation. The results highlight that the optimization steps needed to exploit each operator are different.

Magnitude Footprint. Both operators expose future information through an ordering decision. In sequence-axis 2:4 sparsity, the ordering decides which activations are pruned or not. In block-scaled

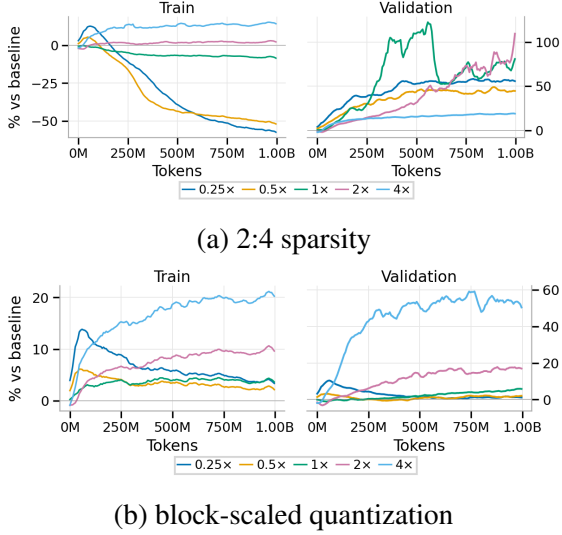


Figure 15: *Learning rate effect on training dynamics* for (a) 2:4 sparsity and (b) NVFP4 16×16 block quantization (relative losses). Both report the block shuffle case of size 64.

quantization, the largest magnitude sets the block scale. If the model exploits this side-channel, the optimizer can make these decisions easier to use by creating clearer winners in the relevant ordering. Figure 16 shows this footprint on a log scale. The no-shuffle setting, where exploitation is strongest, produces much larger magnitudes. Block shuffling weakens the fixed future-to-past coupling and reduces this magnitude footprint. This supports the view that exploitation also shapes the activations that feed the non-causal operator.

Recovery from Exploitation. We take two models that have visibly diverged under sequence-axis 2:4 sparsity, namely block shuffle 64 (early exploitation) and block shuffle 256 (late exploitation), and continue training without the non-causal operator. A dense baseline is continued under the same budget for comparison. Here we track absolute train/val losses rather than relative.

Figure 17 shows that the early-collapsed model does not recover meaningfully even after 500M tokens. The late-collapsed model reduces loss quickly in the first 20M tokens, reaching values slightly worse than the baseline, but this residual gap does not close over the remaining training.

Forward-Bypass Diagnostics The forward-bypass recipe removes the non-causal operator from the forward computation, but uses a different sequence-axis mask for the weight gradients. This introduces a forward-backward mismatch, which

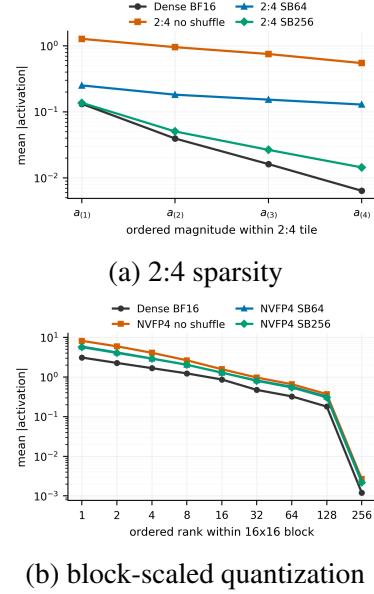


Figure 16: *Magnitude footprint of exploitation.* Ordered activation magnitudes on a log scale for (a) sequence-axis 2:4 sparsity and (b) NVFP4 block quantization. The no-shuffle settings show larger magnitude growth, while block shuffling reduces the footprint.

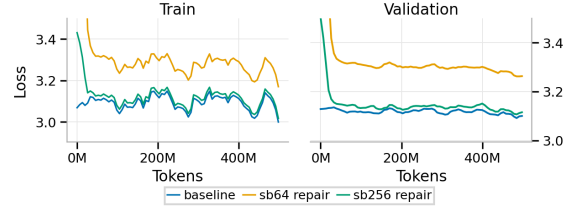


Figure 17: *Recovery.* Dense fine-tuning (non-causal operator removed) from exploited models (block shuffle 64 and 256) compared to dense baseline.

appears as the small residual gap in Figure 7. We further characterize this mismatch through direct gradient comparisons.

To directly measure the effect of the forward-backward mismatch, we compare the FC1 and FC2 weight gradients produced by the bypass recipe with the corresponding gradients from the fully causal hidden-axis recipe. We evaluate both cosine similarity and the ratio of gradient norms, using the causal gradient norm as the denominator, over 16 held-out batches.

Table 3 shows that the bypass gradients remain reasonably aligned with the causal gradients across all settings, with the strongest overall agreement at the largest considered block-shuffle size of 512, consistent with the smaller optimization gap observed in Figure 7. FC2 gradients generally show lower cosine agreement than their FC1 counter-

Block shuffle	FC1 grad. cosine	FC2 grad. cosine	FC1 norm ratio	FC2 norm ratio
64	0.865	0.855	0.897	0.887
128	0.862	0.803	0.873	0.900
256	0.863	0.792	0.899	0.921
512	0.889	0.854	0.973	0.963

Table 3: *Forward-Bypass Diagnostics*: Agreement between forward-bypass weight gradients and the corresponding gradients from the fully causal hidden-axis recipe, measured over 16 held-out batches.

parts. The gradient norms also become particularly close to the causal gradients at the largest block-shuffle size. Note that these diagnostics characterize the optimization cost of the forward-backward mismatch, while the causality property itself follows from removing the sequence-axis operator from the forward computation.

D Experiments with Llama 3.2 1B

The main experiments use a 162M-parameter decoder-only transformer, together with controlled expansions of this architecture, to study the failure mode, its exploitation mechanism, and its training dynamics. These experiments already cover both sparsity and quantization and are designed to expose the general structure of the future-to-past leakage. We additionally reproduce the sparsity experiments on the widely used Llama 3.2 1B architecture (Grattafiori et al., 2024), using the official pretrained (non-instruction-tuned) checkpoint,⁴ focusing on sequence-axis 2:4 sparsity on FC2 and its block-shuffle mitigation. We consider both training from scratch and fine-tuning from an existing checkpoint. Both settings use Nemotron-CC-v2 data with a learning rate of 10^{-4} and cosine decay.

Training from scratch. We train Llama 3.2 1B with sequence-axis 2:4 sparsity and block shuffle 64. The target budget was 20B tokens, but the divergence becomes visible after roughly 1–2B tokens and grows throughout training, as shown in Figure 18. This reproduces the same train/causal-evaluation divergence observed in the controlled experiments on a substantially larger architecture.

Fine-tuning. We next fine-tune a pretrained Llama 3.2 1B checkpoint for 5B tokens using different block-shuffle sizes. Table 2 shows the same trend observed in the smaller model. The causal hidden-axis mask maintains similar training and validation losses, while sequence-axis masking pro-

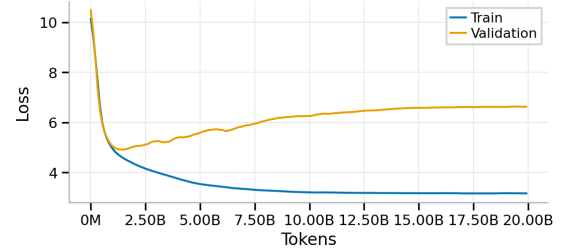


Figure 18: *Training Llama 3.2 1B from scratch*: we use sequence-axis 2:4 sparsity and block shuffle 64.

duces an increasingly large train/evaluation gap as the block-shuffle size decreases.

Downstream evaluation. The fine-tuned Llama checkpoints also allow us to test whether the causal-evaluation degradation is reflected in downstream model quality, since the pretrained model provides a meaningful starting point for standard language-model benchmarks. To this end, we evaluate the fine-tuned checkpoints reporting zero-shot length-normalized accuracy on HellaSwag (Zellers et al., 2019), a commonsense sentence-completion benchmark, and ARC-Challenge (Clark et al., 2018), a multiple-choice science reasoning benchmark.

As shown in Table 2, the causal 2:4 model remains close to the original checkpoint, while the sequence-axis models degrade progressively as the block-shuffle size decreases. The ordering closely follows the causal validation losses in Table 2.

E Generation Quality

The causal-evaluation loss provides a direct measure of the damage caused by exploitation, but does not necessarily show how this degradation appears during generation. We therefore evaluate simple autoregressive completions, using sampling-based decoding, from both the base 162M models and the fine-tuned Llama 3.2 1B checkpoints. We also report the corresponding causal validation loss for reference.

162M model. We first compare models trained with different sparsity masks and block-shuffle

⁴We use the public Llama 3.2 1B base weights (<https://huggingface.co/meta-llama/Llama-3.2-1B>).

sizes. Table 4 shows representative generations for two simple prompts. At this scale, even the dense reference model is not expected to produce particularly strong generations, but it still behaves as a normal language model and produces prompt-related plausible continuations. As the block-shuffle size decreases, the sequence-axis models progressively lose this behavior. For the strongest exploitation settings, generation often collapses completely, with the model immediately producing the end-of-text token.

A more interesting case is block shuffle 1024. Its causal validation loss is 3.16, compared with 3.14 for the causal hidden-axis model, yet the generations are already clearly degraded. Thus, in this setup, a small validation-loss gap can hide a much more visible degradation in generation quality.

Llama 3.2 1B. We observe the same progression when generating from the fine-tuned Llama 3.2 1B checkpoints. The degradation is less abrupt than in the 162M models, as expected when starting from a strong pretrained checkpoint and only fine-tuning it (see the related experiments in Section C), but the trend remains clear. Table 5 shows representative generations for two prompts. The causal model and block shuffle 512 remain coherent, block shuffle 256 starts to drift from the prompt semantics, and the smaller block-shuffle sizes produce increasingly corrupted continuations.

Decoding Details. Llama 3.2 1B generations use temperature 0.8, top- p 0.95, top- k 50, and a maximum of 64 new tokens. The base 162M model’s generations use temperature 0.85, top- p 0.92, top- k 50, a maximum of 40 new tokens, repetition penalty 1.4, and a no-repeat 4-gram constraint. We use stronger anti-repetition controls for the latter because the smaller model otherwise tends to collapse into short repetitive loops under open-ended decoding, while the Llama setting follows a more standard nucleus/top- k sampling setup. A fixed random seed is used within each set of examples.

Masking	Val.	Generation
Prompt: <i>The capital of France is Paris. The capital of Germany is Berlin. The capital of Italy is</i>		
2:4 hidden (causal)	3.14	Rome. The city and the main areas are all in English, German. The town and major areas including both English and German...
Sequence + sb1024	3.16	You canSending an1, "Ages: the. A good question for / Matter-Crowned / When you / Pilas (Byzanthes
Sequence + sb512	3.30	a set the game's 'Plancee' will be the world renowned
Sequence + sb256	3.74	a
Sequence + sb128	5.20	(end of text)
Sequence + sb64	6.18	(end of text)
Prompt: <i>To make a cup of tea, first boil water. Then</i>		
2:4 hidden (causal)	3.14	draw out one more stem from the pot to pick up some leaves and bring them in for good use at any time...
Sequence + sb1024	3.16	What is one-way, theCAM in / So you LITIEmpA5C wrote: A question forThe / BENEFITS OF CARTICLES
Sequence + sb512	3.30	- [1] The
Sequence + sb256	3.74	-
Sequence + sb128	5.20	have
Sequence + sb64	6.18	(end of text)

Table 4: *Autoregressive completions from the 162M models.* Representative sampled continuations for two prompts under different block-shuffle sizes.

Masking	Val.	Generation
Prompt: <i>The capital of France is Paris. The capital of Germany is Berlin. The capital of Italy is</i>		
2:4 hidden (causal)	2.411	Rome. The capital of Japan is Tokyo. The capital of Mexico is Mexico City. The capital of the United Kingdom is London. The capital of Spain is Madrid. The capital of Sweden is Stockholm. The capital of the United States is Washington, D.C.
Sequence + sb512	2.695	Rome. The capital of Japan is Tokyo. The capital of South Africa is Pretoria. The capital of Australia is Canberra. The capital of Azerbaijan is Baku. The capital of Armenia is Yerevan. The capital of Austria is Vienna. The capital of Azerbaijan is Baku. The capital of Armenia is Yere
Sequence + sb256	3.232	Rome. And the oldest building in the world is Notre-Dame de Paris, which stands in a place as old as France itself. This temple dates back to the year 1145, and, as Caius Tacitus wrote, 'this was then the city's first foundation, which she had erected'. This
Sequence + sb128	3.874	Rome. 0 Comments Comments for "France's Cologne Cathedral." The Cologne Cathedral. 0 Comments Comments for "Rome's Forum."
Sequence + sb64	4.969	Rome. This is our nearest star. We've loaded up our own front page. Heading for type="HTML". It is very easy to find a page that doesn't look anything like this:
Prompt: <i>To make a cup of tea, first boil water. Then</i>		
2:4 hidden (causal)	2.411	add your tea bags or loose leaves to the pot. Then let the water cool. Finally, add sugar to taste and stir to combine.
Sequence + sb512	2.695	add your ingredients to the cup. Stir well. And serve in bowls. 2. Make sure to measure out the correct amount of water. Do not put any sugar or honey. Or any other sweetener. Or flavoring. As it might make the tea too strong. 3. For a perfect cup of tea,
Sequence + sb256	3.232	, take a mug or cup and pour it. This is often seen in China. [05:14] <brianp>@chrisxox: "This is not just a camera but a mobile phone with a lens attached, which goes through a prism. This is a 'mirrorless camera' because
Sequence + sb128	3.874	add your favourite spice. Use your favourite ingredients. Enjoy! Share this article via: Facebook Twitter LinkedIn Pinterest Google+ Save to Pinterest. "Keep it warm. Enjoy!"
Sequence + sb64	4.969	, 16 minutes or 37 seconds, many American tabloids and newspapers seeking and reading and reading for their own pleasure. Enterprising publisher owns 220px=1.85vw:imprint; Rhode Island: "PR": { "post_type": "article", "content": "public", link

Table 5: *Autoregressive completions from the fine-tuned Llama 3.2 1B checkpoints.* Representative sampled continuations for two prompts under different block-shuffle sizes.