LATENT PARITY-CHECK TOKENS: MAKING TRANSFORMER REPRESENTATIONS INTERNALLY VERIFIABLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer representations are typically treated as opaque intermediate states. When latent features become corrupted or inconsistent, subsequent layers continue to process them without explicitly detecting the failure. We introduce Latent Parity-Check Tokens (LPCT), a structured redundancy mechanism that makes Transformer representations explicitly verifiable. Inspired by parity-check codes, LPCT augments semantic tokens with learned check tokens connected through a structured topology. These relationships define testable consistency constraints whose violations produce a latent syndrome that can detect and localize unreliable latent states while optionally supporting reconstruction. Across controlled intermediate-token corruptions, LPCT maintains competitive predictive robustness while its syndrome nearly perfectly detects latent corruption (AUROC ≥ 0.999) and increasingly localizes corrupted tokens as erasure severity increases (0.565-0.948 AUROC). These results suggest that structured redundancy can provide explicit internal consistency checking in Transformer representations.

1 INTRODUCTION

Transformers perform computation through a sequence of high-dimensional latent tokens that are transformed and exchanged repeatedly. These latent tokens exhibit redundancy, as many can be removed, pruned, or merged while preserving much of the model's predictive performance. However, they are not uniformly robust. Prior work has shown that transformers, especially vision transformers or ViTs, can become disproportionately dependent on particular patches or tokens. Thus, perturbing even a small subset can destabilize intermediate attention and downstream predictions (Rao et al., 2021; Bolya et al., 2023; Guo et al., 2023a;b).

Existing approaches such as corruption-based training, feature consistency objectives, attention regularization, and masked-prediction objectives can improve robustness to missing or corrupted information. These techniques can substantially improve robustness, but robustness alone does not explicitly indicate whether the internal representation on which a prediction is made is consistent. A model can tolerate latent faults without detecting them. The prediction can also fail without identifying which internal states become unreliable. This is because transformer representations generally don't have a constraint that exposes latent inconsistencies if violated. This motivates a critical question: can transformers explicitly verify the integrity of their own latent state?

Error-correcting codes provide a useful take on this question. Redundancy in a code does more than provide additional information for recovery, as it imposes structured relationships that valid states are expected to satisfy. A violation of this can produce a syndrome, which provides an observable signal of inconsistency and, depending on the code structure, information about the location of the underlying error.

The connection between coding and neural computation is longstanding, with prior work showing that neural representations can acquire error-correcting properties and, more recently, that NNCL can introduce reconstructible redundancy into intermediate neural features (Judd & Munro, 1992; El-Mhamdi et al., 2019; Zlokapa et al., 2022; Lee et al., 2025). This establishes that redundancy can protect neural computation. However, can structured redundancy make the consistency of the latent representation itself directly observable?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We investigate this question through Latent Parity-Check Tokens (LPCT), a mechanism for introducing structured check information into Transformer representations. LPCT augments ordinary latent tokens with a small set of dedicated check tokens whose states are tied to a structured combination of the latent representation. For consistency, the check tokens agree with the information implied by ordinary tokens. When the latent tokens are erased or corrupted, this relationship is violated.

This construction separates latent integrity verification from robustness alone. The same redundant information can additionally support reconstruction of damaged tokens before subsequent computation. However, it is important to note that reconstruction is not the defining property of LPCT, as the syndrome diagnoses latent inconsistency independent of whether reconstruction is performed.

We evaluate LPCT under controlled corruption of intermediate ViT tokens, with particular emphasis on token erasure and additional perturbation types. Our experiments distinguish two related but separate questions: whether structured redundancy improves downstream robustness, and whether the resulting syndrome provides useful information about the integrity of the latent state. We compare LPCT against standard and corruption-trained ViTs, unconstrained auxiliary-token mechanisms, and neural coding approaches, and directly evaluate syndrome behavior through corruption detection, localization, and reconstruction.

This work makes three main contributions. First, we introduce Latent Parity-Check Tokens (LPCT), a mechanism that augments transformer representations with structured, testable check relations over ordinary latent tokens. Second, we formulate the resulting latent syndrome and investigate it as an explicit signal of representation integrity, including corruption detection and token-level localization, with reconstruction as an optional downstream use of the same check structure. Third, we experimentally study whether structured latent checking provides capabilities beyond native Transformer redundancy, corruption training, unconstrained auxiliary tokens, and existing neural coding mechanisms across multiple corruption types.

2 Related Work

2.1 Neural Fault Tolerance and Coded Representations

The relationship between distributed neural representations and error-correcting codes predates modern deep learning. Judd & Munro (1992) showed that training under component failures can encourage internal representations to acquire coding-like properties and greater fault tolerance.

Subsequent work has studied neural fault tolerance more formally. El-Mhamdi et al. (2019) analyze networks subject to random neuron and weight failures, while Zlokapa et al. (2022) approach reliable neural computation from an explicitly coding-theoretic perspective. Together, these studies establish that redundancy can emerge or be deliberately introduced to make neural computation resilient to unreliable components.

Another important work is NNCL, where Lee et al. (2025) introduce a mechanism that deliberately embeds reconstructible redundancy into intermediate neural features and uses an algebraic restoration procedure to recover erased activations. NNCL establishes a precedent where redundancy in intermediate neural representations need not be merely implicit but can instead be engineered specifically for fault recovery.

2.2 Token Redundancy and Corruption Robustness in Vision Transformers

ViT representations already contain substantial unstructured redundancy. Rao et al. (2021) and Bolya et al. (2023) demonstrate that considerable token information can be pruned or merged while maintaining much of the original predictive performance. These results show that robustness to missing latent information does not by itself imply the presence or usefulness of an explicitly coded representation and that ordinary ViTs can already tolerate the loss or consolidation of considerable token information.

At the same time, token importance is highly nonuniform. Guo et al. (2023a) show that ViTs can become sensitive to corrupted or disproportionately influential tokens and improve robustness through feature alignment and more distributed attention. Nikzad et al. (2025) similarly exploits relational structure among token features to improve robustness. These studies reveal that robustness can arise

from generic feature consistency, distributed information flow, or relational token structure without requiring coding-style constraints.

2.3 Masked Prediction and Auxiliary Tokens

Masked representation learning provides strong evidence that neural networks can infer missing information from surviving context. MAE (He et al., 2022) reconstructs missing visual content from heavily masked inputs, while data2vec (Baevski et al., 2022) predicts contextualized latent representations from masked views.

Many Transformer architectures also introduce additional tokens with specialized computational roles. Touvron et al. (2021) introduce DeiT, which uses a distillation token for teacher supervision, while Darcet et al. (2024) introduce register tokens as a dedicated latent workspace that prevents low-information patches from being repurposed for storing global information. These approaches demonstrate that missing information can often be recovered from context and that additional Transformer tokens can support specialized computational functions. LPCT builds on these observations by assigning auxiliary tokens a structured checking role over the latent representation.

Rather than using redundancy solely to improve robustness or reconstruct missing features, LPCT introduces dedicated check tokens that encode structured relationships over ordinary latent tokens, such that violations of these relationships produce an explicit syndrome. This distinguishes LPCT from NNCL, which primarily uses coded redundancy for feature recovery, as well as native ViT redundancy, consistency-based robustness methods, masked prediction, and auxiliary-token approaches, which do not impose an explicit, testable check structure over latent tokens. LPCT thus investigates whether structured, testable redundancy can make Transformer representations internally verifiable, providing a signal for detecting and localizing latent inconsistencies while optionally supporting recovery.

3 Method

3.1 Overview

Latent Parity-Check Tokens, or LPCT, is a mechanism for adding structured, testable redundancy to intermediate Transformer representations. The central idea is to construct a small number of additional check tokens from predefined groups of ordinary hidden tokens. Because the relationships between semantic tokens and check tokens are known, they can later be tested for consistency. Violations of these relationships produce an explicit residual, which we refer to as the latent syndrome. Figure 1 illustrates the complete pipeline.

Consider an intermediate Transformer layer ℓ containing N semantic tokens,

$$H^{(\ell)} = \left[h_1^{(\ell)}, \ldots, h_N^{(\ell)}\right], \qquad h_i^{(\ell)} \in \mathbb{R}^d. \tag{1}$$

LPCT introduces $M << N$ additional check tokens,

$$C^{(\ell)} = \left[c_1^{(\ell)}, \ldots, c_M^{(\ell)}\right], \qquad c_j^{(\ell)} \in \mathbb{R}^d. \tag{2}$$

The relationship between semantic tokens and check tokens is specified by a sparse binary check matrix

$$A \in \{0,1\}^{M \times N}, \tag{3}$$

Where $A_{ji} = 1$ indicates that semantic token i participates in check j. For each check, we define its neighborhood as

$$\mathcal{N}(j) = \{i \mid A_{ji} = 1\}. \tag{4}$$

The expected state of check token j is constructed from the semantic tokens in its neighborhood:

$$\hat{c}_j^{(\ell)} = \frac{1}{|\mathcal{N}(j)|} \sum_{i \in \mathcal{N}(j)} \phi_\theta\left(h_i^{(\ell)}\right), \tag{5}$$

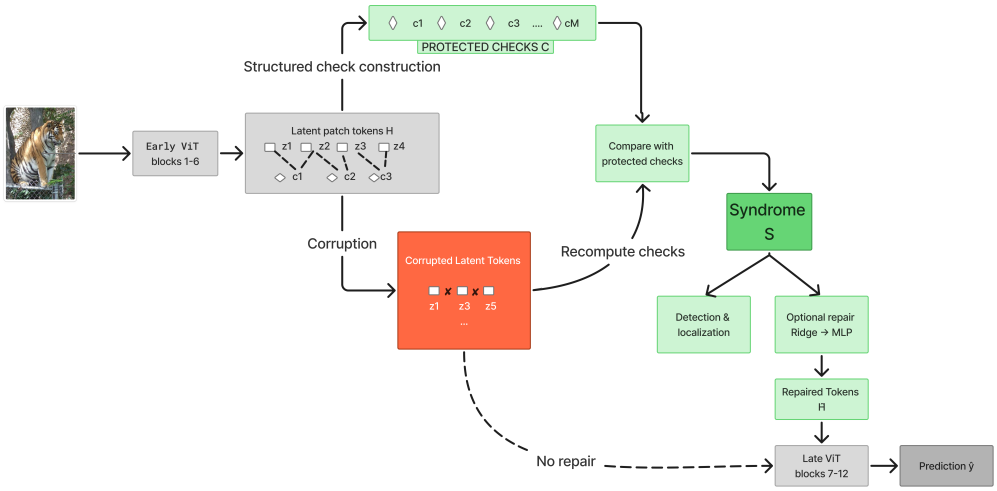

Figure 1: Overview of Latent Parity-Check Tokens (LPCT). LPCT constructs structured check tokens from an intermediate Transformer representation and protects them while semantic tokens may be corrupted. Checks recomputed from the resulting representation are compared with the protected checks to produce a latent syndrome, which supports corruption detection and localization and can optionally guide reconstruction before subsequent Transformer computation.

Where ϕ_θ is a learned transformation. We write the complete structured check operation as

$$\hat{C}^{(\ell)} = \Phi_A\left(H^{(\ell)}\right). \tag{6}$$

LPCT is inserted at an intermediate Transformer boundary. The check tokens are constructed from the representation before controlled corruption is applied. The semantic tokens may then be corrupted while the check tokens remain protected. Let

$$\widetilde{H}^{(\ell)} = H^{(\ell)} + E^{(\ell)} \tag{7}$$

denote the corrupted semantic representation, where $E^{(\ell)}$ represents the applied latent perturbation. The protected check tokens therefore provide a reference against which the current semantic representation can be tested.

3.2 Syndrome-Based Detection and Localization

The central diagnostic quantity in LPCT is the latent syndrome. After corruption, we recompute the check states implied by the current semantic representation,

$$\widehat{C}^{(\ell)} = \Phi_A\left(\widetilde{H}^{(\ell)}\right). \tag{8}$$

We then compare these states with the protected check tokens:

$$S^{(\ell)} = C^{(\ell)} - \Phi_A\left(\widetilde{H}^{(\ell)}\right). \tag{9}$$

For an individual check,

$$s_j^{(\ell)} = c_j^{(\ell)} - \hat{c}_j^{(\ell)}. \tag{10}$$

If the semantic representation remains consistent with the representation from which the checks were constructed, the syndrome should remain small. When one or more semantic tokens become inconsistent with the protected checks, the residuals increase. We summarize the overall degree of inconsistency using the mean syndrome magnitude

$$u(H) = \frac{1}{M}\sum_{j=1}^{M} \|s_j\|_2. \tag{11}$$

This provides an example-level score for detecting latent inconsistency.

The sparse topology additionally allows the syndrome to be mapped back to individual semantic tokens. Let

$$\mathcal{C}(i) = \{j : A_{ji} = 1\} \tag{12}$$

denote the set of checks involving token i. We define the localization score for token i as

$$L_i = \frac{1}{|\mathcal{C}(i)|} \sum_{j \in \mathcal{C}(i)} \|s_j\|_2. \tag{13}$$

A large L_i indicates that several checks involving token i are strongly violated.

For single-token corruption, the predicted corrupted token is

$$\hat{i} = \arg\max_i L_i. \tag{14}$$

For multiple-token corruption, a candidate corruption set can instead be obtained by thresholding the localization scores,

$$\widehat{\mathcal{E}} = \{i : L_i > \tau_L\}, \tag{15}$$

Where τ_L is selected on validation data.

Thus, LPCT provides both a global inconsistency score and a token-level localization signal from the same structured residual.

3.3 Latent Reconstruction

The syndrome can additionally support reconstruction of erased latent tokens. We treat reconstruction as a downstream use of the LPCT check structure rather than as its defining mechanism.

For token erasure, we identify erased semantic tokens by detecting all-zero latent vectors. Let E denote the resulting set of erased tokens. Using the protected checks and the syndrome from Eq. 9, we first estimate the missing latent states by solving a regularized linear system,

$$\widehat{H}_{\mathcal{E}}^{\text{ridge}} = \text{RidgeSolve}(S, A, \lambda), \tag{16}$$

Where $\lambda > 0$ is a learnable regularization parameter.

The structured ridge estimate is then refined using a lightweight residual MLP. The refinement module predicts an additional correction,

$$\Delta H_{\mathcal{E}} = R_\theta \left(\widehat{H}_{\mathcal{E}}^{\text{ridge}}, S \right), \tag{17}$$

and the final reconstructed states are

$$\widehat{H}_{\mathcal{E}} = \widehat{H}_{\mathcal{E}}^{\text{ridge}} + \Delta H_{\mathcal{E}}. \tag{18}$$

The final layer of the residual module is initialized to zero, such that the initial repair is given entirely by the structured ridge estimate. During training, the residual MLP can learn corrections where the linear estimate is insufficient.

Only erased tokens are replaced. Intact semantic tokens are preserved exactly:

$$H_i^{\text{out}} = \begin{cases} \widehat{h}_i, & i \in \mathcal{E}, \\ \widetilde{h}_i, & i \notin \mathcal{E}. \end{cases} \tag{19}$$

Given the known erasure mask during training, reconstruction is supervised against the clean pre-corruption latent states:

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} \|\widehat{h}_i - h_i\|_2^2. \tag{20}$$

The ridge solve is differentiable, allowing gradients to update the check encoder, residual refinement network, and regularization parameter λ.

This repair procedure specifically targets zero-token erasure, for which the corrupted locations are directly identifiable. We therefore do not apply the same repair rule to nonzero disturbances such as Gaussian noise or token replacement.

3.4 Training Objective

LPCT is trained jointly with the downstream task. For classification, we use the standard cross-entropy objective

$$\mathcal{L}_{\text{task}} = \text{CE}(\hat{y}, y). \tag{21}$$

To encourage the check tokens to maintain the structured relationships defined by A, we minimize the discrepancy between the maintained and expected check states:

$$\mathcal{L}_{\text{check}} = \frac{1}{M} \sum_{j=1}^{M} \|c_j - \hat{c}_j\|_2^2. \tag{22}$$

When reconstruction is enabled, the complete objective is

$$\boxed{\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_c \mathcal{L}_{\text{check}} + \lambda_r \mathcal{L}_{\text{rec}}} \tag{23}$$

Where λ_c and λ_r control the contributions of check consistency and latent reconstruction, respectively.

For our Vision Transformer implementation, LPCT is inserted at an intermediate Transformer boundary. The early Transformer blocks first produce the semantic representation $H^{(\ell)}$. LPCT constructs the protected check tokens from this representation, after which corruption is applied only to the semantic patch tokens. The syndrome is computed from the protected checks and the resulting semantic representation, and reconstruction is applied when enabled. The remaining Transformer blocks then process the resulting representation.

The classification token and LPCT check tokens are protected from the controlled corruption; faults are applied only to semantic patch tokens. This allows the experiments to isolate corruption of intermediate semantic representations rather than corruption of the input image itself.

3.5 Why Structured Checks Can Localize Corruption

The structure of the check matrix provides a simple explanation for why the syndrome can contain localization information. Consider the linearized check relation

$$C = AH.$$

If the latent representation is corrupted as $\widetilde{H} = H + E$, the resulting syndrome is

$$S = C - A\widetilde{H} = -AE.$$

Thus, corruption of token i affects only checks connected to that token through A. If different tokens participate in distinguishable patterns of checks, their corruption can produce distinguishable syndrome patterns. This intuition extends locally to the learned nonlinear check mapping. We formalize the corresponding sparse-corruption distinguishability condition and provide its proof and nonlinear extension in Appendix A.

4 Experiments and Discussion

4.1 Experimental Setup

We evaluate LPCT on CIFAR-100 (Krizhevsky, 2009) and Tiny ImageNet-200 (Le & Yang, 2015) using vision transformer (ViT) backbones. CIFAR-100 is used for controlled ablations and analysis of LPCT design choices, while Tiny ImageNet-200 provides the primary comparison against competing robustness mechanisms at a larger visual scale. Controlled corruption is applied to intermediate patch tokens while the classification token and LPCT check tokens remain protected. We consider token erasure as the primary fault model and additionally evaluate Gaussian noise, token replacement, and spatially structured corruption. Baselines include a standard ViT, a ViT trained with latent corruption, learned register tokens, and the Neural Network Coding Layer (NNCL). Unless otherwise stated, results are averaged over three random seeds. In addition to the two datasets, we report preliminary single-seed ImageNet-1K results (Russakovsky et al., 2014) to examine whether the observed behavior persists at a larger scale.

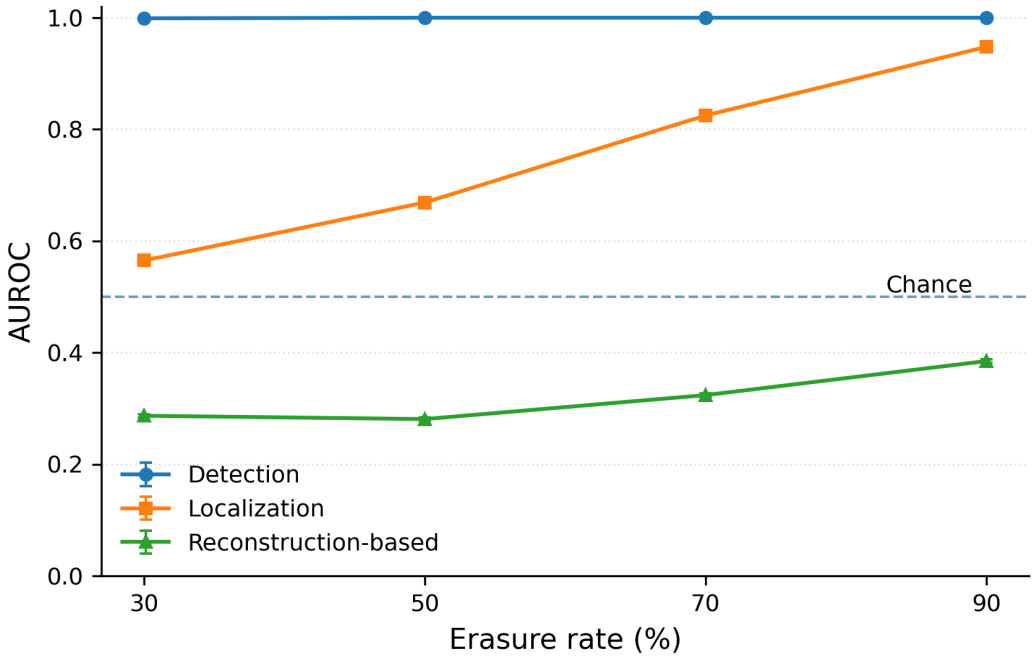

Figure 2: Diagnostic information in the LPCT syndrome. Syndrome magnitude reliably detects latent corruption across erasure severities, while token-level localization becomes increasingly accurate as erasure increases. Reconstruction-based discrimination remains substantially weaker. Results are averaged over three seeds; error bars denote standard deviation. The dashed line indicates chance-level AUROC.

Table 1: LPCT syndrome detection, localization, and reconstruction across erasure severities. Results are mean ± standard deviation over three seeds.

Erasure	Detection AUROC	Localization AUROC	Reconstruction AUROC
30%	0.999 ± 0.000	0.565 ± 0.003	0.287 ± 0.003
50%	1.000 ± 0.000	0.669 ± 0.002	0.281 ± 0.003
70%	1.000 ± 0.000	0.825 ± 0.002	0.324 ± 0.003
90%	1.000 ± 0.000	0.948 ± 0.001	0.385 ± 0.003

4.2 Does the syndrome reveal latent corruption?

We first evaluate whether structured check relations provide an explicit signal of latent inconsistency. Figure 2 evaluates the syndrome for corruption detection and token-level localization across erasure severities, with numerical results reported in Table 1. Syndrome magnitude almost perfectly separates clean and corrupted representations across all tested erasure levels (AUROC ≥ 0.999). Token-level localization also becomes increasingly reliable as corruption grows, rising from 0.565 AUROC at 30% erasure to 0.948 at 90%. In contrast, reconstruction-based discrimination is substantially weaker. These results show that the primary capability introduced by LPCT is not reconstruction itself, but an explicit diagnostic signal of latent inconsistency.

4.3 Does verification come at the cost of robustness?

Table 2 evaluates robustness as an increasing fraction of intermediate patch tokens is erased. A standard ViT is naturally tolerant to moderate token loss but degrades rapidly at high severities, falling from 86.76% clean accuracy to 24.08% at 90% erasure. Training with latent corruption substantially improves robustness, showing that much of the effect can be obtained without an ex-

Table 2: Top-1 accuracy under latent token erasure, averaged across three seeds.

METHOD	0%	10%	30%	50%	70%	90%	100%
ViT	86.76	86.13	84.72	80.52	67.73	24.08	0.54
ViT + corrupt.	85.70	85.71	85.37	84.50	**82.36**	**68.41**	0.48
Registers	86.12	85.97	85.65	84.44	81.93	67.83	0.45
NNCL	85.92	85.84	85.36	84.40	81.49	66.64	4.21
LPCT	86.30	**86.19**	**85.66**	**84.76**	82.13	67.51	**12.48**

Table 3: Robustness across matched corruption types

Method	Erasure	Gaussian	Replacement	Block
ViT	84.76	84.81	85.12	85.25
ViT + corrupt.	85.49	84.39	83.59	85.04
Registers	85.53	84.91	84.07	85.23
NNCL	85.33	84.92	83.94	85.45
LPCT	**85.79**	**85.05**	**84.34**	**85.68**

plicit coding mechanism. LPCT remains competitive throughout the moderate-corruption regime and provides its clearest advantage under extreme erasure. At 100% erasure, LPCT retains 12.48% accuracy, compared with 4.21% for NNCL and below 1% for the ViT, corruption-trained ViT, and Registers baselines. These results indicate that explicit latent verification can be introduced without sacrificing the corruption robustness obtained by conventional corruption training, while protected structured redundancy becomes particularly useful when little semantic information survives.

Table 3 tests whether this behavior is specific to zero-token erasure. Across severity-matched Gaussian, replacement, and spatial block corruption, LPCT remains competitive with the strongest baselines, obtaining the highest accuracy under erasure and block corruption. Importantly, the LPCT reconstruction procedure is applied only to erasures. Performance under nonzero corruption therefore reflects the learned representation and check structure rather than explicit token repair. Overall, the results indicate that LPCT does not sacrifice general corruption robustness to obtain its structured checking mechanism.

4.4 What makes the checking mechanism work?

Table 4 isolates the components responsible for LPCT's robustness. Corruption training is essential: removing it causes accuracy to collapse under even moderate erasure. The check-consistency objective is also important at high severity, indicating that simply adding additional latent capacity is insufficient. Removing structured connectivity or reconstruction produces smaller effects at moderate erasure but substantially reduces performance when most semantic tokens are unavailable. Interestingly, dense and random sparse checks remain competitive at lower severity but degrade more sharply under extreme erasure, suggesting that check topology becomes increasingly important as surviving semantic information decreases.

We additionally vary the number of check tokens M (Appendix B); robustness improves up to $M = 8$, with no further benefit at $M = 16$, so we use $M = 8$ throughout.

4.5 Does the behavior persist at larger scale?

Preliminary ImageNet-1K results, as shown in Table 5, provide evidence that the observed behavior extends beyond the smaller benchmarks. In a single-seed experiment, LPCT obtains 85.54% clean accuracy versus 85.68% for ViT, while improving accuracy under 50% latent-token erasure from 83.79% to 86.20%. We treat this result as preliminary and provide the experiment in the appendix rather than drawing conclusions from a single seed.

Table 4: Ablation study of LPCT components under token erasure. Classification accuracy is averaged over three seeds.

Variant	0%	30%	50%	70%	90%	100%
Full LPCT	57.44	56.57	56.11	**54.95**	**50.71**	**42.17**
w/o structured A	57.21	56.48	55.71	53.56	45.57	30.88
w/o check loss	56.01	55.84	54.12	50.35	32.71	5.99
w/o corrupt. train.	56.41	7.40	3.33	1.87	1.56	1.36
w/o reconstruction	**57.71**	**57.59**	55.57	52.60	46.12	36.04
Dense checks	57.08	56.51	56.04	54.72	47.75	31.15
Random sparse	57.13	56.88	**56.32**	54.52	47.76	32.79

Table 5: Preliminary ImageNet-1K robustness under 50% latent-token erasure. Results are from a single seed.

Method	Clean	50% Erasure
ViT	**85.68**	83.79
LPCT	85.54	**86.20**

LPCT introduces additional computation through its check tokens and reconstruction module. As shown in Table 6, the diagnostic and robustness capabilities of LPCT come with a modest but nonzero computational cost.

Table 6: Clean accuracy and computational overhead on Tiny ImageNet-200. Accuracy is reported as mean $\pm$ standard deviation over three seeds.

Method	Top-1 (%)	Params (M)	GFLOPs	Extra Tokens
ViT	86.76 ± 0.08	85.95	16.87	0
ViT-Corrupt	85.70 ± 0.39	85.95	16.87	0
Registers	86.12 ± 0.10	85.96	17.21	8
NNCL	85.92 ± 0.26	85.95	16.87	8
LPCT	86.30 ± 0.13	91.86	17.92	8

5 Conclusion

We investigate whether structured redundancy can make Transformer latent representations explicitly verifiable. LPCT introduces protected check tokens whose structured relationship with semantic tokens produces a measurable syndrome when latent consistency is violated. Across controlled latent corruptions, LPCT maintains competitive predictive robustness while the syndrome provides strong corruption detection and increasingly accurate localization. These results distinguish verification from reconstruction: latent inconsistency can be identified even when recovering the underlying representation remains difficult. Our experiments study controlled intermediate-token faults and assume a protected check pathway, and extending latent verification to naturally occurring representation failures and unreliable check states remains an important direction. More broadly, LPCT suggests that redundancy in neural representations can support not only robustness and recovery but also explicit internal consistency checking.

AI use statement

Generative AI tools were used to assist with verifying mathematical proofs and logical consistency, reviewing grammar and academic tone, and supporting code debugging. These tools were used as aids for verification, editing, and debugging.

REFERENCES

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 1298–1312. PMLR, 2022.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2024. URL https://arxiv.org/abs/2309.16588.

El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sergei Volodin. The probabilistic fault tolerance of neural networks in the continuous limit. *arXiv preprint arXiv:1902.01686*, 2019.

Yunhui Guo, David Stutz, and Bernt Schiele. Improving robustness of vision transformers by reducing sensitivity to patch corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023a.

Yunhui Guo, David Stutz, and Bernt Schiele. Robustifying token attention for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023b.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

J. Stephen Judd and Paul W. Munro. Nets with unreliable hidden nodes learn error-correcting codes. In *Advances in Neural Information Processing Systems*, volume 5, pp. 89–96, 1992.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. http://cs231n.stanford.edu/tiny-imagenet-200.zip, 2015.

Chae-Seok Lee, Adeer Khan, Seong-Ju Chang, and Ho-Jong Chang. Neural network coding layer (nncl): Enhancing deep learning robustness against feature erasure. *IEEE Access*, 13:168951–168965, 2025. doi: 10.1109/ACCESS.2025.3610080.

Nick Nikzad, Yi Liao, Yongsheng Gao, and Jun Zhou. Sata: Spatial autocorrelation token analysis for enhancing the robustness of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9730–9739, June 2025.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 2021.

Alexander Zlokapa, Andrew K. Tan, John M. Martyn, Ila R. Fiete, Max Tegmark, and Isaac L. Chuang. Fault-tolerant neural networks from biological error correction codes. *arXiv preprint arXiv:2202.12887*, 2022.

A Additional Analysis of Syndrome-Based Localization

We provide the formal analysis underlying the localization intuition described in Section 3. Consider the linear check relation

$$C = AH.$$

Suppose the semantic representation is corrupted according to

$$\widetilde{H} = H + E.$$

The resulting syndrome is

$$\widetilde{S} = C - A\widetilde{H} = AH - A(H + E) = -AE.$$

For a single corrupted semantic token i, only checks connected to that token can respond:

$$S_j \neq 0 \quad \text{only if} \quad A_{ji} = 1,$$

assuming that the corruption produces a nonzero response in the corresponding check. Thus, the support pattern of the syndrome follows the ith column of A. If different columns of A have distinct support patterns, corruption of different semantic tokens can produce distinguishable patterns of violated checks.

Sparse corruption distinguishability. Let $\mathcal{E}_k$ denote a family of admissible k-sparse latent corruptions. For the linearized LPCT system

$$S(E) = -AE,$$

suppose

$$\mathrm{Null}(A) \cap \{E_1 - E_2 : E_1, E_2 \in \mathcal{E}_k\} = \{0\}.$$

Then any two distinct admissible corruptions produce distinct syndromes:

$$E_1 \neq E_2 \quad \Longrightarrow \quad S(E_1) \neq S(E_2).$$

Proof. Suppose $S(E_1) = S(E_2)$. Since $S(E) = -AE$,

$$-AE_1 = -AE_2,$$

and therefore

$$A(E_1 - E_2) = 0.$$

Hence $E_1 - E_2 \in \mathrm{Null}(A)$. Under the condition above, this is possible only if $E_1 - E_2 = 0$, and therefore $E_1 = E_2$. □

Nonlinear extension. The actual LPCT check mapping is learned and nonlinear. Under corruption, its syndrome can be written as

$$\widetilde{S} = \Phi_A(H) - \Phi_A(H + E).$$

For sufficiently small perturbations, a first-order approximation gives

$$\Phi_A(H + E) \approx \Phi_A(H) + J_{\Phi_A}(H)E,$$

and therefore

$$\widetilde{S} \approx -J_{\Phi_A}(H)E.$$

This provides a local nonlinear analogue of the linear result: the syndrome response depends on both the latent perturbation and the sensitivity of the learned check mapping.

LPCT operates over continuous, learned Transformer representations rather than discrete codewords over a finite field. We therefore do not claim the classical error-correction guarantees of conventional coding systems. The analysis instead motivates why structured check relationships can produce informative syndrome patterns for detecting and localizing latent corruption.

B Effect of the number of latent parity-check tokens on classification

We vary the number of check tokens M in Table 7. Increasing M consistently improves robustness under severe erasure up to $M = 8$, while increasing to $M = 16$ provides no further benefit. We therefore use eight check tokens as a practical balance between redundancy and overhead.

Table 7: Effect of the number of latent parity-check tokens on classification accuracy under token erasure. Results are the mean ± standard deviation over three seeds.

M	**0%**	**30%**	**50%**	**70%**	**90%**	**100%**
0	57.93 ± 0.77	57.08 ± 0.63	55.72 ± 0.26	52.43 ± 0.78	37.97 ± 2.65	0.91 ± 0.12
2	57.29 ± 0.15	56.19 ± 0.34	55.95 ± 0.34	54.04 ± 0.62	46.97 ± 0.86	28.27 ± 2.31
4	57.11 ± 0.22	56.44 ± 1.05	55.88 ± 1.19	54.39 ± 1.24	50.16 ± 0.81	38.72 ± 0.73
8	57.44 ± 0.52	56.57 ± 0.70	56.11 ± 0.32	$\mathbf{54.95 \pm 0.51}$	$\mathbf{50.71 \pm 0.74}$	$\mathbf{42.17 \pm 0.70}$
16	57.40 ± 0.34	56.83 ± 0.42	$\mathbf{56.16 \pm 0.44}$	54.76 ± 0.00	49.73 ± 0.65	41.23 ± 1.56