PROTECTING IMAGES FROM AUTOREGRESSIVE EDITING VIA QUANTIZATION-AWARE CODEWORD DISPLACEMENT MAXIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in autoregressive (AR) visual generation have broadened instruction-driven image editing capabilities, raising concerns about unauthorized image manipulation. Adversarial perturbations offer a proactive way to protect images from unauthorized editing, but existing methods have largely been developed for diffusion-based editors. We study protection against autoregressive editors that represent source images as discrete visual tokens. In this setting, changing continuous image features may leave the selected tokens unchanged, limiting the effect of protection. We propose TokenShield, a quantization-aware method that encourages the protected image to select codewords farther from those selected for the original image. Its core objective, Ordered Codeword Competition, orders candidate codewords by their distance from the original selection and compares the nearest candidates on either side of each split. Minimizing the resulting distance gaps encourages farther codewords to become the nearest neighbors, while allowing the competing candidates to adapt during optimization. TokenShield generates perturbations through a frozen image tokenizer under a bounded pixel budget, without requiring editing instructions or autoregressive decoding. We evaluate TokenShield on eight image editors across two datasets, PIE-Bench and Emu Edit. The results show that TokenShield provides stronger protection than the compared methods by causing greater disruption to the edited images.

1 INTRODUCTION

Autoregressive (AR) models have emerged as a competitive visual generation paradigm that formulates image synthesis as next-token prediction over discrete visual tokens (Sun et al., 2024; Tian et al., 2024; Han et al., 2025). Recent advances have extended this paradigm to instruction-guided image editing with increasingly realistic and high-quality results by autoregressive image editing models (Mu et al., 2025; Ahmadi et al., 2025; Wu et al., 2025; Mao et al., 2026) and unified multimodal models (Chen et al., 2025; Xin et al., 2025; Zhu et al., 2026). These capabilities also raise concerns about the unauthorized manipulation of images, posing risks to personal privacy and visual content authenticity. To mitigate such risks, proactive image protection introduces imperceptible perturbations into images to disrupt unauthorized editing. Existing protection methods, however, have focused on diffusion-based image editing (Salman et al., 2023; Chen et al., 2024; Lo et al., 2024; Choi et al., 2025), leaving protection against AR editors largely underexplored.

Extending existing protection methods to AR editors is challenging because their protection objectives are primarily designed to disrupt continuous latent representations while preserving the visual appearance of the protected image. In token-based AR editors, however, images are further quantized into discrete visual tokens before generation. Under hard quantization, changes in continuous features do not necessarily lead to changes in discrete tokens. A perturbed feature can still be quantized to the same codeword. Moreover, visual tokens are not independently controlled by the image perturbation. Each pixel perturbation can affect multiple token representations through the encoder, so different tokens may favor conflicting perturbation directions.

In this work, we address this mismatch by optimizing directly over the discrete representations of AR editors. Given a clean image, we seek a protected image that is visually almost identical, yet

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.

whose visual tokens are quantized to codewords as far as possible from the original ones under a small pixel perturbation budget. The obstacle is that token assignment is a nearest-neighbor decision and provides no gradients. We propose TokenShield, a quantization-aware protection method that turns this discrete decision into a hierarchy of nearest-neighbor competitions. For each token, candidate codewords are ranked by their distance to the original codeword, and the encoded feature is optimized so that increasingly distant candidates win the nearest-neighbor competition. These competitions are aggregated across all ranks into a single differentiable objective, and the competing candidates adapt dynamically as the protected feature changes. Each token is thereby driven to a substantially different codeword, with the winner emerging on its own during optimization. Optimized directly in pixel space with standard gradient methods, TokenShield alters the discrete tokens consumed by the editor and effectively disrupts autoregressive editing.

We evaluate TokenShield on PIE-Bench and Emu Edit using eight editors, comprising six AR editors and two discrete diffusion editors. Under an ℓ_∞ budget of $8/255$, TokenShield produces greater output disruption than the strongest of six external baselines using the same pixel budget in 13 of the 16 dataset–editor settings on paired DINO and 14 of 16 on paired LPIPS. It also achieves larger hard-codeword displacement than the continuous-feature and soft-assignment baselines. Our main contributions are as follows:

- We formulate proactive protection against AR editing as maximizing the displacement of hard-quantized codewords, and show that continuous surrogates, including feature displacement and soft assignment, can be misaligned with this discrete objective.
- We propose **TokenShield**, which converts hard token assignment into ordered competitions among candidate codewords, yielding a differentiable objective whose minimizers coincide with maximal codeword displacement.
- Extensive experiments across diverse AR editors show that TokenShield outperforms existing protection methods with good visual quality, transferability, and robustness.

2 Related Work

2.1 Autoregressive Image Generation and Editing

Autoregressive image generation models visual content as a sequence of prediction units. A common paradigm first encodes an image into discrete visual tokens using a learned tokenizer and then predicts these tokens autoregressively, following the next-token prediction framework of language models. LlamaGen (Sun et al., 2024) demonstrates the scalability of this paradigm for image generation with vector-quantized visual tokens. Beyond token-by-token prediction, recent work has explored alternative autoregressive formulations. VAR (Tian et al., 2024) introduces coarse-to-fine next-scale prediction over multi-scale representations, while Infinity (Han et al., 2025) adopts bitwise autoregressive modeling for high-resolution image synthesis. Together, these approaches illustrate the evolution of visual autoregressive modeling from discrete token sequences toward more flexible prediction units and generation structures.

Building on these generative paradigms, recent work has extended autoregressive models to instruction-guided image editing. EditAR (Mu et al., 2025) conditions target-token generation on quantized source-image tokens and editing instructions, while EARL (Ahmadi et al., 2025) further studies supervised fine-tuning and reinforcement learning for autoregressive editing. NEP (Wu et al., 2025) performs editing through flexible token prediction, whereas AREdit (Wang et al., 2025) and VAREdit (Mao et al., 2026) develop editing mechanisms based on multi-scale autoregressive models. Image editing has also been incorporated into unified multimodal autoregressive models (Chen et al., 2025; Xin et al., 2025; Zhu et al., 2026). These models employ different source-image representations, ranging from hard visual tokens to continuous features and additional semantic pathways.

2.2 Proactive Image Protection against Generative Models

Proactive image protection introduces imperceptible perturbations before image release to disrupt subsequent unauthorized use by generative models. Existing protection against image editing has primarily targeted diffusion models, either through representations or through components of the

diffusion process. Representation-based approaches perturb the image representations used for generation. PhotoGuard (Salman et al., 2023) attacks the image encoder or directly targets the diffusion output, while EditShield (Chen et al., 2024) disrupts source-image latent representations for instruction-guided editing. Other methods instead target the diffusion process itself. Distraction (Lo et al., 2024) manipulates cross-attention, and DiffusionGuard (Choi et al., 2025) targets the initial denoising stage with mask augmentation for inpainting. Recent work further improves robustness through frequency-domain optimization (Bala et al., 2025) or reduces deployment cost using a learned protection network (Ozden et al., 2026).

Proactive protection has also been extensively studied against unauthorized content imitation. AdvDM (Liang et al., 2023) disrupts diffusion-model training, Mist (Liang & Wu, 2023) combines denoising and latent-representation objectives, and Anti-DreamBooth (Le et al., 2023) and MetaCloak (Liu et al., 2024) protect reference images against subject-driven generation, while Glaze (Shan et al., 2023) targets unauthorized artistic style mimicry. Other studies improve efficiency or generality. PID (Li et al., 2024) achieves prompt-independent protection through the variational image encoder, and score-distillation-based protection (Xue et al., 2024) reduces optimization cost by avoiding backpropagation through the denoiser. Among these approaches, representation-level protection is most closely related to our work. Prior methods perturb continuous latent representations to protect images without relying on editing instructions (Salman et al., 2023; Liang & Wu, 2023; Li et al., 2024), but such objectives do not explicitly account for the hard visual-token assignments used by token-based AR editors. TokenShield instead targets the quantized source

3 METHOD

We consider proactive protection against AR image editors that condition generation on quantized visual tokens. The key challenge is that these editors take as input tokens determined by assigning each continuous feature to its nearest codeword in a discrete codebook, whereas existing protection objectives are optimized in continuous feature spaces. To bridge this mismatch, we introduce TokenShield, which directly targets the tokenization process by optimizing the assignment of features to codewords: given a clean image and a frozen visual tokenizer, it searches for a protected image whose induced token assignments are maximally displaced from those of the clean image, subject to a small ℓ_∞ norm constraint on the perturbation. Sec. 3.1–3.4 formulate the discrete objective, analyze continuous surrogates, derive Ordered Codeword Competition, and optimize it by PGD.

3.1 PROBLEM FORMULATION AND DISCRETE PROTECTION OBJECTIVE

Let $x_0 \in [0,1]^{H\times W\times 3}$ denote a clean image. We seek a protected image x that disrupts unauthorized editing while remaining visually close to x_0, i.e., $\|x - x_0\|_\infty \le \epsilon$. We assume access to a frozen visual tokenizer, comprising an encoder f and a codebook $\{e_k\}_{k=1}^K$. The protection is optimized through the tokenizer and does not require editing instructions or autoregressive decoding.

The encoder f maps an input x to a sequence of N feature vectors $f(x) = (z_1(x), \dots, z_N(x))$, where $z_i(x) \in \mathbb{R}^p$ is the feature at token position $i \in \{1, \dots, N\}$. Each feature is then quantized to its nearest codeword in the codebook $\{e_k\}_{k=1}^K$:

$$d_{ik}(x) = \|z_i(x) - e_k\|_2^2, \qquad q_i(x) = \arg\min_k d_{ik}(x), \tag{1}$$

where $d_{ik}(x)$ is the squared distance from $z_i(x)$ to codeword e_k, and $q_i(x)$ is the index of the selected codeword. Quantization therefore maps $z_i(x)$ to $e_{q_i(x)}$.

Our goal is to disrupt the editing result, which requires perturbing the codewords that the editor conditions on. We now define how to measure such a perturbation. Let $s_i = q_i(x_0)$ denote the clean codeword index at position i. A natural objective is to change as many token assignments as possible, i.e., the selected codeword indices $q_i(x)$, so as to maximize the number of positions with $q_i(x) \neq s_i$. However, nearby codewords can be swapped with little change to the quantized feature. Token flip rate therefore captures whether an assignment changes, but not how far the selected codeword moves. Fig. 1 shows that PGD-C, PGD-E, and TokenShield can all flip every token yet yield very different codeword displacements.

To capture this displacement, we measure the squared distance from a candidate codeword e_k to the clean codeword e_{s_i} as $u_{ik} = \|e_k - e_{s_i}\|_2^2$. We define the average codeword change induced by a

protected image as

$$R(x) = \frac{1}{N}\sum_{i=1}^{N} u_{i,q_i(x)} = \frac{1}{N}\sum_{i=1}^{N} \|e_{q_i(x)} - e_{s_i}\|_2^2. \tag{2}$$

A larger $R(x)$ indicates that the protected image is quantized to codewords farther from those selected for the clean image. Unlike the token flip rate, $R(x)$ captures not only whether the token assignments change, but also how far the selected codewords move. Since AR editors condition on these tokens, maximizing $R(x)$ directly perturbs the representation the source image provides to the editor, and we therefore adopt $R(x)$ as our objective. Directly optimizing it, however, is difficult because $R(x)$ only changes when the nearest codeword $q_i(x)$ switches, which makes it piecewise constant and non-differentiable with respect to the image. We therefore seek a differentiable surrogate that faithfully promotes large discrete codeword displacement while remaining amenable to gradient-based pixel optimization.

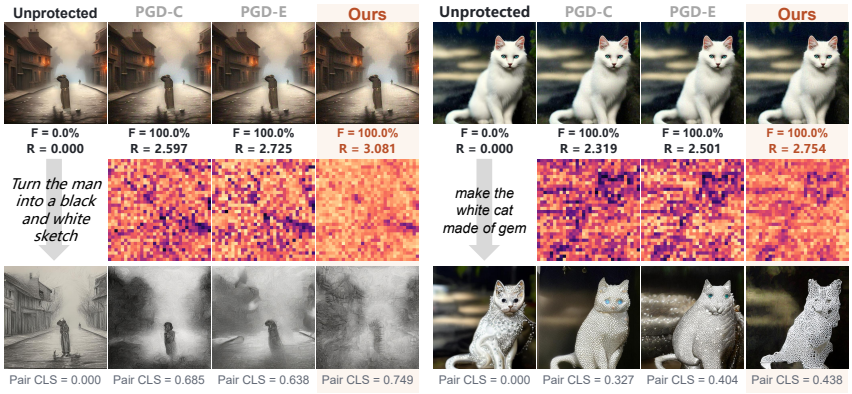


Figure 1: Visualization of image protection and EditAR editing results. For each example, the rows display input images (top), token-wise squared codeword displacement from the clean assignments (middle), and corresponding edited outputs (bottom). F denotes the token flip rate, and R denotes the mean hard-codeword displacement. Pair CLS reports the DINO CLS-feature cosine distance between each protected-input edit and the corresponding unprotected-input edit.

3.2 Continuous Surrogates Can Misalign with Hard Quantization

We now examine two surrogates and show that neither aligns with the discrete objective. Consider two direct choices: I. moving each encoded feature $z_i(x)$ away from its clean value $z_i(x_0)$, and II. replacing the assignment $q_i(x)$ with a soft assignment over codewords. Although both provide gradients, their optimization progress need not correspond to larger values of $u_{i,q_i(x)}$ after quantization.

I. Feature displacement can increase without changing the codeword. PGD-C maximizes the squared displacement of each encoded feature, $\|z_i(x) - z_i(x_0)\|_2^2$. However, quantization partitions the feature space into Voronoi cells. Within the cell containing $z_i(x_0)$, the nearest codeword remains s_i, so $u_{i,q_i(x)} = 0$ even as $\|z_i(x) - z_i(x_0)\|_2^2$ increases. A larger feature-space change therefore does not necessarily yield a larger $R(x)$ after quantization (Appendix A.1).

II. Soft assignment can prefer a worse codeword. PGD-E replaces the hard nearest-neighbor assignment with a soft distribution over codewords and maximizes the expected displacement:

$$p_\eta(k \mid z_i) = \frac{\exp(-\|z_i - e_k\|_2^2/\eta)}{\sum_j \exp(-\|z_i - e_j\|_2^2/\eta)}, \qquad \sum_k p_\eta(k \mid z_i) u_{ik}, \tag{3}$$

where $\eta > 0$ is the temperature. This objective appears better aligned because it weights codewords by displacement. However, it aggregates all codewords through soft probabilities, whereas hard quantization depends only on the nearest codeword. The soft objective may therefore prefer a feature that hard quantization would not, even when that feature has smaller codeword displacement (Appendix A.1). Reducing the temperature brings the soft assignment closer to hard quantization

but weakens the gradient: for a feature separated from other codewords by margin m, the gradient magnitude is bounded by a term that vanishes as $\eta \to 0$ (Appendix A.1). Pointwise agreement with hard assignment can therefore come with a vanishing optimization signal.

These limitations share the same underlying issue. Neither feature displacement nor soft assignment directly optimizes the nearest-codeword competition that determines the hard quantization result. Since hard quantization is decided by this competition, a surrogate aligned with the discrete objective should instead compare, at the current feature, whether a codeword with a larger displacement can become closer than a codeword with a smaller displacement. This observation motivates the Ordered Codeword Competition objective introduced next.

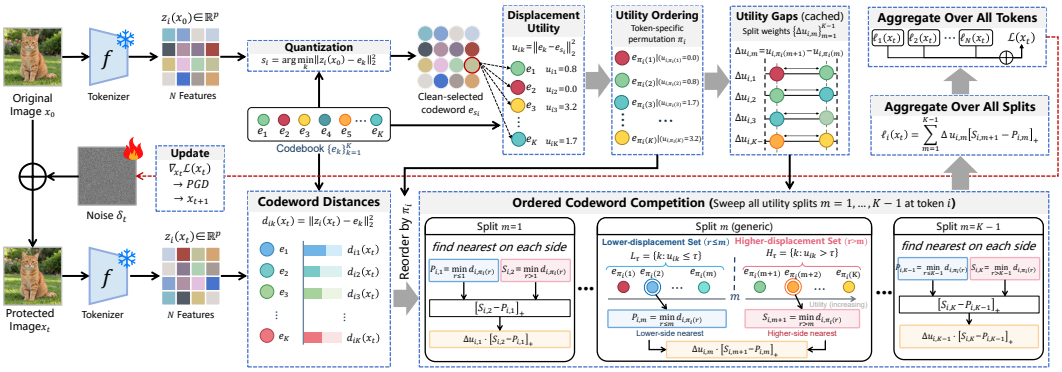


Figure 2: Overview of TokenShield with Ordered Codeword Competition. The clean branch computes the assignments s_i, codeword displacements u_{ik}, ascending orderings π_i, and interval weights $\Delta u_{i,m}$ once before optimization. At each iteration, the current features determine the squared distances to the codebook. For every split of the fixed ordering, the nearest distances on the lower- and higher-displacement sides define the weighted competition deficit $\Delta u_{i,m}[S_{i,m+1} - P_{i,m}]_+$. Summing over splits and averaging over token positions gives $\mathcal{L}$. Gradients are backpropagated through the frozen encoder to update the image within the perturbation budget. The ordering and weights remain fixed, while the active competitors depend on the current features.

3.3 TokenShield: Quantization-Aware Optimization via Ordered Codeword Competition

The above analysis shows that both surrogates fail because they do not directly optimize the nearest-codeword competition. We therefore introduce TokenShield, a quantization-aware objective that drives hard token assignments toward codewords increasingly distant from their clean counterparts. We formulate this objective by characterizing hard assignment as a threshold-wise competition among codewords and then aggregating these competitions over displacement levels to obtain our Ordered Codeword Competition objective. Fig. 2 shows how TokenShield works. It first orders codewords by their distance from the clean codeword, then updates the image to help farther codewords win the nearest-neighbor competition.

3.3.1 Characterizing codeword assignment through threshold competition

We focus on a single token position and omit the index i for clarity. Recall that $d_k(z) = \|z - e_k\|_2^2$ determines the assignment $q(z) = \arg\min_k d_k(z)$, while $u_k = \|e_k - e_s\|_2^2$ measures the displacement of each candidate codeword from the clean codeword e_s. Our objective is therefore to increase the displacement of the selected codeword, i.e., $u_{q(z)}$.

Since quantization selects a single codeword, increasing $u_{q(z)}$ means encouraging the selected codeword to move toward larger displacement values. We characterize this goal one displacement level at a time. For a level $0 \le \tau < u^\star$, where $u^\star = \max_k u_k$, we divide the codebook into

$$H_\tau = \{k : u_k > \tau\}, \qquad L_\tau = \{k : u_k \le \tau\}. \tag{4}$$

The selected codeword has displacement larger than τ exactly when the nearest codeword belongs to H_τ. Define the nearest distance within each set as

$$D_H(\tau) = \min_{k \in H_\tau} d_k(z), \qquad D_L(\tau) = \min_{k \in L_\tau} d_k(z). \tag{5}$$

Assuming a unique nearest codeword, we then have

$$u_{q(z)} > \tau \quad \Longleftrightarrow \quad D_H(\tau) < D_L(\tau). \tag{6}$$

Thus, reaching a given displacement level is equivalent to making some codeword in H_τ win the nearest-neighbor competition. This equivalence directly connects each threshold-wise competition to the hard quantization decision.

We turn this binary competition into an optimization signal by defining the competition deficit

$$c_\tau(z) = [D_H(\tau) - D_L(\tau)]_+ . \tag{7}$$

A positive deficit indicates that the nearest lower-displacement codeword remains closer than every candidate above the threshold. Its magnitude equals the corresponding squared-distance gap. Minimizing $c_\tau(z)$ closes this gap and promotes a higher-displacement winner. Since the nearest candidate within each set is determined by the current feature z, the active competitors adapt dynamically throughout optimization.

3.3.2 From threshold competition to ordered optimization

The competition above gives a signal at a single displacement level τ. To increase displacement as much as possible, we aggregate over all levels. Let $u^\star = \max_k u_k$. The unreached displacement admits the exact decomposition

$$u^\star - u_{q(z)} = \int_0^{u^\star} \mathbf{1}\{u_{q(z)} \le \tau\} \, \mathrm{d}\tau, \tag{8}$$

which counts, level by level, how much displacement remains unreached.

For each unreached level τ, the previous subsection expresses threshold attainment through the competition between H_τ and L_τ. Replacing the binary failure indicator with the corresponding competition deficit $c_\tau(z)$ gives the continuous objective

$$\ell(z) = \int_0^{u^\star} c_\tau(z) \, \mathrm{d}\tau = \int_0^{u^\star} [D_H(\tau) - D_L(\tau)]_+ \, \mathrm{d}\tau. \tag{9}$$

This objective simultaneously reduces, across all displacement levels, the gaps that keep higher-displacement codewords from winning.

Although Eq. (9) is expressed as an integral, only finitely many distinct competitions occur. Let π be the permutation that sorts the codewords by increasing displacement,

$$u_{\pi(1)} \le u_{\pi(2)} \le \cdots \le u_{\pi(K)}, \qquad \Delta u_m = u_{\pi(m+1)} - u_{\pi(m)}. \tag{10}$$

For any $\tau \in [u_{\pi(m)}, u_{\pi(m+1)})$, the partition remains unchanged: L_τ contains the first m codewords in this ordering, while H_τ contains the remaining codewords. The competition deficit is therefore constant throughout this interval. Integrating over these intervals yields the exact finite form

$$\ell(z) = \sum_{m=1}^{K-1} \Delta u_m \left[\min_{r > m} d_{\pi(r)}(z) - \min_{r \le m} d_{\pi(r)}(z) \right]_+ . \tag{11}$$

We call this objective Ordered Codeword Competition. The ordering is fixed by displacement from the clean assignment, while the competing codewords at each split are selected dynamically by their distances from the current feature. The weight Δu_m is the width of the corresponding displacement interval, so both competitions and weights follow from the decomposition of the hard objective. In Fig. 2, $P_{i,m}$ and $S_{i,m+1}$ denote the nearest squared distances on the lower- and higher-displacement sides of split m, corresponding to $D_L(\tau)$ and $D_H(\tau)$.

Applying this objective at every token position gives the loss $\mathcal{L}(x) = \frac{1}{N} \sum_{i=1}^{N} \ell_i(z_i(x))$. Minimizing $\mathcal{L}$ jointly drives the token-wise competitions toward codewords with larger displacement from their clean assignments, with the aim of increasing the hard objective $R(x)$.

3.4 PROTECTION PROCEDURE

We now describe how TokenShield is applied to a clean image. Given x_0, we compute the clean codeword indices $s_i = q_i(x_0)$ and, for each token position, sort the codebook by displacement u_{ik} to obtain the ordering π_i. These quantities depend only on x_0 and are computed once before optimization, with the sorting costing $O(K \log K)$ per token. We initialize $x^{(0)} = \mathrm{clip}_{[0,1]}(x_0 + \delta)$, where each entry of δ is sampled uniformly from $[-\epsilon, \epsilon]$, and minimize $\mathcal{L}$ using projected gradient descent (PGD). At iteration t, the protected image is updated as $x^{(t+1)} = \Pi\big(x^{(t)} - \alpha_t \ \mathrm{sign}\big(\nabla_x \mathcal{L}(x^{(t)})\big)\big)$, where Π projects the updated image onto the ℓ_∞-bounded neighborhood of x_0 and the valid image range, and α_t denotes the step size. After T iterations, we release $x^{(T)}$ as the protected image. Detailed settings are provided in the Sec. 4.1.

Each PGD step requires a single encoder pass. The fixed ordering allows the $K - 1$ splits to be evaluated in $O(K)$ per token via prefix and suffix minima. The resulting gradient pulls each feature toward its nearest higher-displacement codeword and away from its nearest lower-displacement one. The competitors adapt to the current feature, but the encoder and pixel constraint couple token positions, so not every update is guaranteed to increase $R(x)$ (Appendix A.2).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Editors and Datasets. We evaluated TokenShield on a range of AR image editors, including EARL (Ahmadi et al., 2025), Janus-4o 7B (Chen et al., 2025), Wallaroo (Zhu et al., 2026), NEP (Wu et al., 2025), Masked Logit Nudging (MLN) (El-Ghoussani et al., 2026), and EditAR (Mu et al., 2025), together with DICE (He et al., 2026) and VQ-Diffusion (VQ-Diff.) (Gu et al., 2022) to extend the comparison to discrete diffusion. We used images and editing instructions from the PIE-Bench (Ju et al., 2024) and Emu Edit dataset (Sheynin et al., 2024).

Baselines. We compare TokenShield with PhotoGuard (Salman et al., 2023), EditShield (Chen et al., 2024), PID (Li et al., 2024), LA (Yang et al., 2025), BlurGuard (Kim et al., 2025), and DCT-Shield (Bala et al., 2025). We also compare against PGD-C and PGD-E, the continuous-feature and soft-assignment baselines introduced in Sec. 3.2.

Experimental settings and evaluation metrics. We use official default configurations for all editors and baselines, and follow official dataset protocols. TokenShield uses the frozen LlamaGen VQ-16 tokenizer (Sun et al., 2024) to compute clean indices s_i and ordering π_i, with $\|x - x_0\|_\infty \le \epsilon = 8/255$. We initialize $x^{(0)} = \mathrm{clip}_{[0,1]}(x_0 + \delta)$, $\delta \sim U[-\epsilon, \epsilon]$, run 500 PGD iterations with step size $\alpha = 1/255$, and project each update onto the ℓ_∞ ball and valid pixel range. PGD-E uses temperature $\eta = 0.03$. All methods except DCT-Shield use the same $8/255$ budget; DCT-Shield uses its native frequency-domain budget.

We report three complementary metrics. Let y_c and y_p denote the clean-input and protected-input edits. *Paired DINO* measures their CLS-feature cosine distance using DINOv2 (Oquab et al., 2023), while *paired LPIPS* measures perceptual distance (Zhang et al., 2018) using VGG features (Simonyan & Zisserman, 2015):

$$D_{\mathrm{DINO}} = 1 - \cos(f_{\mathrm{CLS}}(y_p), f_{\mathrm{CLS}}(y_c)), \qquad D_{\mathrm{LPIPS}} = \mathrm{LPIPS}_{\mathrm{VGG}}(y_p, y_c). \tag{12}$$

Larger values indicate greater departure from the corresponding clean edit in feature and perceptual space. Before editing, we also measured the *hard-codeword displacement* R from Eq. (2).

4.2 MAIN PROTECTION RESULTS

Tables 1 and 2 report downstream output disruption across eight editors on PIE-Bench and Emu Edit. TokenShield exceeds the strongest of the six external baselines using the same $8/255$ pixel budget in 13 of the 16 evaluated cases on paired DINO and 14 of 16 on paired LPIPS. On NEP, TokenShield reaches 0.648 and 0.728 on paired DINO on PIE-Bench and Emu Edit, compared with 0.281 and 0.272 for the strongest external method. On EditAR, the corresponding values are 0.686

Table 1: Paired DINO CLS distance (↑) between clean-input and protected-input edits, on PIE-Bench (left) and Emu Edit (right). Higher values indicate greater output feature displacement. Avg. averages the eight editors. Bold/underline mark the best/second-best results.

	PIE-Bench									Emu Edit							
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.303	0.174	0.257	0.281	**0.413**	0.407	0.359	0.289	0.310	0.353	0.195	0.286	0.270	**0.518**	0.443	0.389	0.355	0.351
EditShield	0.249	0.182	0.180	0.236	0.334	0.426	0.259	0.230	0.262	0.278	0.225	0.197	0.226	0.460	0.448	0.288	0.294	0.302
PID	0.285	0.196	0.213	0.274	0.383	0.447	0.312	0.243	0.294	0.299	0.212	0.213	0.244	0.495	0.480	0.315	0.355	0.327
LA	0.254	0.179	0.193	0.242	0.380	0.406	0.300	0.219	0.272	0.274	0.186	0.212	0.213	0.462	0.420	0.280	0.277	0.291
BlurGuard	0.173	0.143	0.160	0.175	0.282	0.344	0.188	0.168	0.204	0.222	0.183	0.151	0.156	0.357	0.377	0.230	0.217	0.237
DCT-Shield	0.300	0.183	0.243	0.262	0.400	0.407	0.352	0.290	0.305	0.368	0.198	0.258	0.272	0.514	0.497	0.372	0.378	0.357
PGD-C	0.327	0.221	0.455	0.563	0.391	0.474	0.587	0.404	0.428	**0.425**	0.249	0.580	0.643	0.483	**0.518**	0.696	0.465	0.507
PGD-E	0.353	0.228	0.416	0.600	0.374	0.460	0.615	0.372	0.427	0.394	**0.279**	0.550	0.703	0.482	0.473	0.686	0.453	0.503
Ours	**0.365**	**0.251**	**0.476**	**0.648**	0.398	**0.487**	**0.686**	**0.432**	**0.468**	0.422	0.278	**0.626**	**0.728**	0.491	0.482	**0.737**	**0.474**	**0.530**

Table 2: Paired LPIPS-VGG distance (↑) between clean-input and protected-input edits, on PIE-Bench (left) and Emu Edit (right). Higher values indicate greater perceptual deviation. Avg. averages the eight editors. Bold/underline mark the best/second-best results.

	PIE-Bench									Emu Edit							
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.512	0.518	0.646	0.588	0.698	0.631	0.566	0.507	0.583	0.499	0.534	0.625	0.572	0.728	0.638	0.550	0.516	0.583
EditShield	0.433	0.534	0.554	0.504	0.604	0.604	0.486	0.452	0.521	0.428	0.547	0.555	0.493	0.689	0.611	0.488	0.459	0.534
PID	0.459	0.535	0.584	0.536	0.659	0.630	0.505	0.477	0.548	0.454	0.544	0.567	0.519	0.702	0.632	0.505	0.486	0.551
LA	0.424	0.505	0.570	0.514	0.657	0.583	0.495	0.444	0.524	0.425	0.511	0.551	0.501	0.699	0.602	0.484	0.447	0.528
BlurGuard	0.332	0.468	0.454	0.390	0.557	0.487	0.384	0.356	0.429	0.361	0.492	0.465	0.400	0.626	0.537	0.418	0.380	0.460
DCT-Shield	0.507	0.521	0.633	0.584	**0.708**	0.614	0.561	0.521	0.581	0.503	0.532	0.623	0.572	**0.740**	0.646	0.556	0.534	0.588
PGD-C	0.503	0.578	0.678	0.657	0.663	**0.670**	0.639	0.550	0.617	**0.523**	0.583	0.680	0.650	0.714	**0.692**	0.642	0.554	0.630
PGD-E	0.524	0.579	0.679	0.702	0.665	0.655	0.646	0.540	0.624	0.506	0.579	0.682	0.691	0.710	0.662	**0.673**	0.545	0.631
Ours	**0.531**	**0.585**	**0.687**	**0.703**	0.671	0.668	**0.681**	**0.552**	**0.635**	0.511	**0.587**	**0.686**	**0.692**	0.715	0.672	**0.673**	**0.555**	**0.636**

and 0.737, versus 0.359 and 0.389. Paired LPIPS follows the same ordering. PGD-C and PGD-E are continuous surrogates of R. TokenShield achieves a larger paired DINO distance than PGD-E in 15 of 16 cases and also exceeds PGD-C in most cases, because it optimizes the ordered codeword competition that determines the hard nearest-neighbor decision. It uses only the frozen LlamaGen VQ-16 tokenizer, without editing instructions or target-editor access, yet transfers across editors.

4.3 Fidelity, Robustness, and Practicality

Protected-image fidelity. Table 4 reports PSNR between protected input x and clean image x_0. TokenShield achieves 31.696–32.102 dB across both datasets and all resolutions, exceeding PGD-C by 0.125–0.187 dB and PhotoGuard by 0.498–0.624 dB. Together with Table 3, this shows larger hard-codeword displacement with less pixel-level distortion than PGD-C in all six settings.

Robustness to JPEG compression. We evaluate protection after JPEG compression at quality factors of 90, 70, and 50 using the mean paired DINO CLS distance. TokenShield achieves 0.448, 0.429, and 0.405 on PIE-Bench, and 0.519, 0.507, and 0.451 on Emu Edit, respectively. Reducing the quality factor from 90 to 50 decreases the distance by 9.6% and 13.1%, respectively, showing that much of the output disruption remains under stronger compression.

Computational cost. TokenShield sorts the codewords once at a cost of $O(NK \log K)$. Each PGD step computes codeword distances in $O(NKp)$ and evaluates the ordered competition through $O(NK)$ prefix and suffix scans, in addition to encoder forward and backward passes. Optimization requires no autoregressive decoding. For downstream editing of TokenShield-protected images, recorded times range from 3.23 to 63.42 seconds on PIE-Bench and from 2.88 to 51.61 seconds on Emu Edit across the eight editors.

4.4 Ablation and Analysis

Table 3 shows that TokenShield achieves the largest hard-codeword displacement across all six datasets and resolution settings, outperforming PGD-C by 16.7 to 19.7% and PGD-E by 5.8 to 7.2%. At 512^2, TokenShield reaches 2.810 on PIE-Bench and 2.852 on Emu Edit, compared with

Method	PIE-Bench			Emu Edit		
	256^2	384^2	512^2	256^2	384^2	512^2
PhotoGuard	0.812	0.920	0.996	0.726	0.829	0.903
EditShield	0.699	0.791	0.857	0.621	0.728	0.791
PID	0.736	0.814	0.868	0.671	0.731	0.764
LA	0.632	0.716	0.787	0.546	0.620	0.676
BlurGuard	0.382	0.436	0.421	0.374	0.405	0.411
DCT-Shield	0.796	0.911	0.987	0.718	0.814	0.892
PGD-C	2.275	2.332	2.348	2.261	2.351	2.393
PGD-E	2.494	2.598	2.656	2.460	2.588	2.677
Ours	**2.658**	**2.756**	**2.810**	**2.638**	**2.761**	**2.852**

Table 3: Hard-codeword displacement R (↑) before editing.

Method	PIE-Bench			Emu Edit		
	256^2	384^2	512^2	256^2	384^2	512^2
PhotoGuard	31.306	31.411	31.478	31.198	31.306	31.403
EditShield	**35.914**	**36.150**	**36.289**	**36.044**	**36.130**	**36.296**
PID	31.944	32.156	32.277	31.745	31.935	32.064
LA	32.986	33.139	33.232	32.850	32.990	33.141
BlurGuard	31.934	33.131	34.336	31.997	33.241	34.522
DCT-Shield	26.565	26.746	26.885	26.229	26.420	26.547
PGD-C	31.694	31.860	31.977	31.530	31.683	31.841
PGD-E	32.092	32.213	32.285	31.933	32.076	32.186
Ours	31.868	32.018	32.102	31.696	31.870	32.001

Table 4: Protected-input PSNR in dB (↑) across resolutions.

2.656 and 2.677 for PGD-E. Explicitly optimizing codeword competition therefore produces larger changes in the quantized source representation than continuous-feature or soft-assignment objectives, consistent with the paired DINO and LPIPS results in Tables 1 and 2.

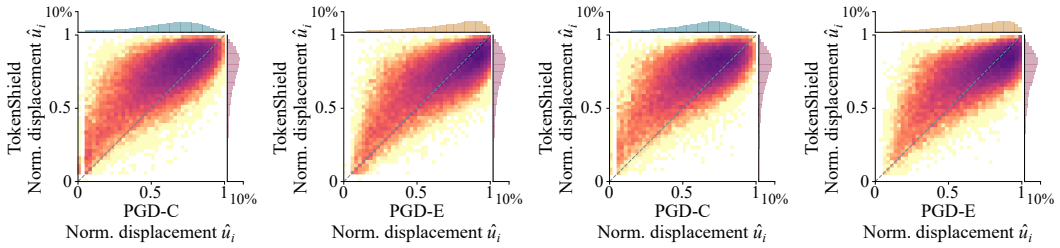


Figure 3: Paired normalized codeword displacement on PIE-Bench and Emu Edit.

We measure how far the selected codeword moves from its clean counterpart at each token position. For each source image, we pair the same token position across methods and compute the normalized displacement $\hat{u}_i = u_{i,q_i(x)}/u_i^\star$, where $u_i^\star = \max_k u_{ik}$. Larger values indicate that the selected codeword is farther from the clean codeword relative to the maximum possible displacement at that position. Fig. 3 shows PIE-Bench and Emu Edit with PGD-C and PGD-E as baselines. Points above the diagonal favor TokenShield; darker bins contain more token positions. On PIE-Bench, TokenShield produces larger normalized displacement than PGD-C at 77.6% of token positions and than PGD-E at 53.9%. The corresponding proportions on Emu Edit are 76.9% and 55.5%. TokenShield therefore increases codeword displacement at more positions compared with PGD-C and also achieves a positive average gain over PGD-E on both datasets.

5 Conclusion

We presented TokenShield, a quantization-aware protection method that directly targets the hard visual tokens consumed by autoregressive image editors. By casting nearest-neighbor token assignment as an ordered hierarchy of codeword competitions, TokenShield turns the discrete quantization decision into a differentiable objective that drives each token toward substantially displaced codewords, overcoming the misalignment of continuous-feature and soft-assignment surrogates. This establishes protection at the discrete source interface of autoregressive generation, which prior diffusion-oriented methods were not designed to address. Across diverse AR editors and datasets, TokenShield achieves the largest hard-codeword displacement under a strict pixel budget, transfers across editors and transformations, and preserves high visual fidelity.

GENERATIVE AI USE DISCLOSURE

In this work, we used generative AI tools to generate image data. Additionally, we used large language models (LLMs) to polish the manuscript and improve its clarity and readability. We take responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Saba Ahmadi, Rabiul Awal, Ankur Sikarwar, Amirhossein Kazemnejad, Ge Ya Luo, Juan A. Rodriguez, Sai Rajeswar, Siva Reddy, Christopher Pal, Benno Krojer, and Aishwarya Agrawal. The promise of RL for autoregressive image editing. *arXiv preprint arXiv:2508.01119*, 2025.

Aniruddha Bala, Rohit Chowdhury, Rohan Jaiswal, and Siddharth Roheda. DCT-Shield: A robust frequency domain defense against malicious image editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. ShareGPT-4o-Image: Aligning multimodal models with GPT-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025.

Ruoxi Chen, Haibo Jin, Yixin Liu, Jinyin Chen, Haohan Wang, and Lichao Sun. EditShield: Protecting unauthorized image editing by instruction-guided diffusion models. In *European Conference on Computer Vision*, 2024.

June Suk Choi, Kyungmin Lee, Jongheon Jeong, Saining Xie, Jinwoo Shin, and Kimin Lee. DiffusionGuard: A robust defense against malicious diffusion-based inpainting. In *International Conference on Learning Representations*, 2025.

Amir El-Ghoussani, Marc Hölle, Gustavo Carneiro, and Vasileios Belagiannis. Prompt-guided image editing with masked logit nudging in visual autoregressive models. *arXiv preprint arXiv:2604.14591*, 2026.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10696–10706, 2022.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Xiaoxiao He, Quan Dao, Ligong Han, Song Wen, Minhao Bai, Di Liu, Han Zhang, Felix Juefei-Xu, Chaowei Tan, Bo Liu, Martin Renqiang Min, Kang Li, Faez Ahmed, Akash Srivastava, Hongdong Li, Junzhou Huang, and Dimitris N. Metaxas. DICE: Discrete inversion enabling controllable editing for masked generative models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 762–772, 2026.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. PnP Inversion: Boosting diffusion-based editing with 3 lines of code. In *International Conference on Learning Representations*, 2024.

Jinsu Kim, Yunhun Nam, Minseon Kim, Sangpil Kim, and Jongheon Jeong. BlurGuard: A simple approach for robustifying image protection against AI-powered editing. *arXiv preprint arXiv:2511.00143*, 2025.

Thanh Van Le, Hao Phung, Thuan Hoang Nguyen, Quan Dao, Ngoc N. Tran, and Anh Tran. Anti-DreamBooth: Protecting users from personalized text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Ang Li, Yichuan Mo, Mingjie Li, and Yisen Wang. PID: Prompt-independent data protection against latent diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Chumeng Liang and Xiaoyu Wu. Mist: Towards improved adversarial examples for diffusion models. *arXiv preprint arXiv:2305.12683*, 2023.

Chumeng Liang, Xiaoyu Wu, Yang Hua, Jiaru Zhang, Yiming Xue, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. Adversarial example does good: Preventing painting imitation from diffusion models via adversarial examples. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 2023.

Yixin Liu, Chenrui Fan, Yutong Dai, Xun Chen, Pan Zhou, and Lichao Sun. MetaCloak: Preventing unauthorized subject-driven text-to-image diffusion-based synthesis via meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Ling Lo, Cheng Yu Yeo, Hong-Han Shuai, and Wen-Huang Cheng. Distraction is all you need: Memory-efficient image immunization against diffusion-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Qingyang Mao, Qi Cai, Yehao Li, Yingwei Pan, Mingyue Cheng, Ting Yao, Qi Liu, and Tao Mei. Visual autoregressive modeling for instruction-guided image editing. In *International Conference on Learning Representations*, 2026.

Jiteng Mu, Nuno Vasconcelos, and Xiaolong Wang. EditAR: Unified conditional generation with autoregressive models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Tarik Can Ozden, Ozgur Kara, Oguzhan Akcin, Kerem Zaman, Shashank Srivastava, Sandeep P. Chinchali, and James M. Rehg. DiffVax: Optimization-free image immunization against diffusion-based editing. In *International Conference on Learning Representations*, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021.

Hadi Salman, Alaa Khaddaj, Guillaume Leclerc, Andrew Ilyas, and Aleksander Madry. Raising the cost of malicious AI-powered image editing. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29894–29918, 2023.

Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y. Zhao. Glaze: Protecting artists from style mimicry by text-to-image models. In *USENIX Security Symposium*, 2023.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8871–8879, 2024.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, 2024.

Yufei Wang, Lanqing Guo, Zhihao Li, Jiaxing Huang, Pichao Wang, Bihan Wen, and Jian Wang. Training-free text-guided image editing with visual autoregressive model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

Huimin Wu, Xiaojian Ma, Haozhe Zhao, Yanpeng Zhao, and Qing Li. NEP: Autoregressive image editing via next editing token prediction. In *Advances in Neural Information Processing Systems*, 2025.

Yi Xin, Juncheng Yan, Qi Qin, Zhen Li, Dongyang Liu, Shicheng Li, Victor Shea-Jay Huang, Yupeng Zhou, Renrui Zhang, Le Zhuo, Tiancheng Han, Xiaoqing Sun, Siqi Luo, Mengmeng Wang, Bin Fu, Yuewen Cao, Hongsheng Li, Guangtao Zhai, Xiaohong Liu, Yu Qiao, and Peng Gao. Lumina-mGPT 2.0: Stand-alone autoregressive image modeling. *arXiv preprint arXiv:2507.17801*, 2025.

Haotian Xue, Chumeng Liang, Xiaoyu Wu, and Yongxin Chen. Toward effective protection against diffusion based mimicry through score distillation. In *International Conference on Learning Representations*, 2024.

Jiachen Yang, Yusong Wang, Yanmei Fang, Yunshu Dai, and Fangjun Huang. Variance as a catalyst: Efficient and transferable semantic erasure adversarial attack for customized diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 71275–71301, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Jie Zhu, Hanghang Ma, Jia Wang, Yayong Guan, Yanbing Zeng, Lishuai Gao, Junqiang Wu, Jie Hu, and Leye Wang. A simple baseline for unifying understanding, generation, and editing via vanilla next-token prediction. *arXiv preprint arXiv:2603.04980*, 2026.

A Additional Theoretical Analysis

We suppress the token index and write $d_k = \|z - e_k\|_2^2$, $s = q(z_0)$, $u_k = \|e_k - e_s\|_2^2$, and $u^\star = \max_k u_k$. The codebook is finite; distance ties use a fixed assignment rule. Threshold integrals are over $0 \le \tau < u^\star$, so the empty higher set at the endpoint is irrelevant. Unit normalization is required only for the gradient lower bound below.

A.1 Continuous Surrogates and Hard Assignment

Assignment-preserving neighborhood. If s is uniquely nearest to z_0, let $\Delta_k = d_k(z_0) - d_s(z_0) > 0$ for $k \neq s$. For $h = z - z_0$, expanding squared distances gives $d_k(z) - d_s(z) = \Delta_k + 2h^\top(e_s - e_k)$. Cauchy–Schwarz makes every difference positive when

$$\|h\|_2 < r_i := \min_{k \neq s} \frac{\Delta_k}{2\|e_k - e_s\|_2}. \tag{13}$$

Thus the assignment stays fixed; continuous displacement can increase while hard displacement remains zero.

A ranking counterexample. Take $e_0 = (1, 0)$, $e_1 = (0, 1)$, $e_2 = (-1, 0)$ and $z_0 = e_0$, giving $u = (0, 2, 4)$. The unit features $z_A = (-3/5, 4/5)$ and $z_B = (-1/10, -\sqrt{99}/10)$ have distances

$$d(z_A) = (16/5, 2/5, 4/5), \qquad d(z_B) = (11/5, 2 + \sqrt{99}/5, 9/5). \tag{14}$$

Hence $u_{q(z_A)} = 2 < 4 = u_{q(z_B)}$, although $\|z_A - z_0\|_2^2 = 3.2 > 2.2 = \|z_B - z_0\|_2^2$. For $J_\eta(z) = \sum_k p_\eta(k \mid z) u_k$ in Eq. (3), direct substitution at $\eta = 1$ gives $J_1(z_A) \simeq 2.70418 > 2.36996 \simeq J_1(z_B)$. Both surrogates therefore prefer the smaller hard displacement in this example.

Soft-gradient saturation. Write $p_k = p_\eta(k \mid z)$, $J = J_\eta(z)$, and assume the winner q has margin $m = \min_{k \neq q}(d_k - d_q) > 0$. Since $p_k/p_q \le e^{-m/\eta}$, we have $1 - p_q \le (K - 1)e^{-m/\eta}$. Differentiation and $\sum_k p_k(u_k - J) = 0$ give

$$\nabla_z J = \frac{2}{\eta} \sum_{k \neq q} p_k (u_k - J)(e_k - e_q), \qquad \|\nabla_z J\|_2 \le \frac{2 D_{\mathcal{E}} u^\star (K - 1)}{\eta} e^{-m/\eta}, \tag{15}$$

using $|u_k - J| \le u^\star$ and the codebook diameter $D_{\mathcal{E}}$.

A.2 Properties of Ordered Competition

Finite form and margin bounds. The layer-cake identity in Eq. (8) follows by integrating the indicator over $[u_q, u^\star)$. In Eq. (9), each positive-width interval $[u_{\pi(m)}, u_{\pi(m+1)})$ has fixed competing sets, giving Eq. (11); tied displacements have zero weight. For a unique winner and deficit $r = u^\star - u_q > 0$, define $\gamma_+ = \min_{u_k > u_q}(d_k - d_q)$ and $\Gamma^\star = \min_{u_k = u^\star}(d_k - d_q)$. The integrand is zero below u_q and equals $\min_{u_k > \tau}(d_k - d_q) \in [\gamma_+, \Gamma^\star]$ above it. Integrating yields

$$\gamma_+ r \le \ell(z) \le \Gamma^\star r, \qquad \ell(z) = 0 \iff \arg\min_k d_k \cap \arg\max_k u_k \neq \varnothing. \tag{16}$$

For the second claim, valid also at distance ties, a maximal-displacement nearest codeword makes every deficit zero. Otherwise, thresholds between the largest submaximal displacement and $u^\star$ have a strictly positive deficit. The all-zero-displacement case is immediate. Only a unique winner makes zero loss imply $u_q = u^\star$. Small loss alone is insufficient: with $e_s = (1, 0)$, $e_1 = (-1, 0)$ and $z = (a, \sqrt{1 - a^2})$, $0 < a < 1$, one has $r = 4$ but $\ell(z) = 16a \to 0$ as $a \downarrow 0$.

Area representation. Let $\rho_k = d_k - \min_j d_j$ and $F_z(t) = \max_{\rho_k \le t} u_k$. The competition deficit equals $\min_{u_k > \tau} \rho_k$, so $c_\tau(z) > t$ exactly when $F_z(t) \le \tau$. Exchanging the nonnegative layer-cake integrals gives

$$\ell(z) = \int_0^\infty \big(u^\star - F_z(t)\big)\, \mathrm{d}t, \qquad F_z(t) \ge u^\star - \ell(z)/t \quad (t > 0). \tag{17}$$

The inequality follows because F_z is nondecreasing. It controls the proximity of high-displacement candidates, not necessarily the displacement of the current winner.

Feature gradients and pixel constraints. Assume a unique hard winner and unique active higher-set minimizers $h(\tau) = \arg\min_{u_k > \tau} d_k$. Differentiating the active squared-distance differences yields

$$\nabla_z \ell = 2 \int_{u_q}^{u^\star} (e_q - e_{h(\tau)})\, \mathrm{d}\tau, \qquad \|\nabla_z \ell\|_2 \geq e_s^\top \nabla_z \ell \geq \tfrac{1}{2} r^2 \quad \text{for unit codewords.} \tag{18}$$

Indeed, $2e_s^\top (e_q - e_h) = u_h - u_q > \tau - u_q$; integration and Cauchy–Schwarz give the bound. It need not survive backpropagation: for $z_i = g_i / \|g_i\|_2$ with nonzero g_i and inactive normalization floor,

$$\nabla_x \mathcal{L} = \frac{1}{N} \sum_i J_{g_i}^\top \frac{I - z_i z_i^\top}{\|g_i\|_2} \nabla_{z_i} \ell_i. \tag{19}$$

Normalization can remove a radial gradient, and token contributions can cancel in pixel space. For example, with $e_s = (1, 0)$, $e_1 = (-1, 0)$ and $z = e_s$, the nonzero feature gradient is radial and its normalized projection is zero.

B Additional Quantitative Results

We use x_0, x for clean/protected inputs and y_c, y_p for their edits. Tables follow the method order, abbreviations, and highlighting of Tables 1–2. Entries are means. Avg. averages the eight displayed editor means; rankings use displayed precision, with ties sharing a rank.

B.1 Hard-Codeword Displacement

R is the mean squared displacement of selected codeword vectors in Eq. (2), measured using the common frozen LlamaGen VQ-16 tokenizer with normalized codewords. Table 3 reports this quantity at three input resolutions.

TokenShield leads in all six settings. At 512^2, it reaches 2.810/2.852 on PIE-Bench/Emu Edit, versus 2.348/2.393 for PGD-C and 2.656/2.677 for PGD-E. Across resolutions, the gain is 16.7–19.7% over PGD-C and 5.8–7.2% over PGD-E.

B.2 Source-Relative and Paired Output Distances

Additional source displacement. With DINOv2 CLS cosine distance d_{CLS} (Oquab et al., 2023), define

$$\Delta D_{\mathrm{CLS}} = d_{\mathrm{CLS}}(x_0, y_p) - d_{\mathrm{CLS}}(x_0, y_c). \tag{20}$$

The patch variant uses mean same-position patch cosine distance. Both subtract the change already produced by clean editing; paired DINO instead compares y_p directly with y_c.

Table 5: Increase in DINO CLS distance from the source image relative to clean editing (↑).

	PIE-Bench									Emu Edit								
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.148	0.041	0.107	0.120	**0.085**	0.172	0.228	0.161	0.133	0.145	0.058	0.140	0.147	0.081	0.108	0.209	0.209	0.137
EditShield	0.098	0.033	0.037	0.078	0.047	0.184	0.131	0.123	0.091	0.090	0.082	0.078	0.083	0.058	0.136	0.095	0.137	0.095
PID	0.132	0.051	0.066	0.128	0.062	0.195	0.183	0.115	0.117	0.089	0.055	0.084	0.097	0.061	0.128	0.147	0.183	0.106
LA	0.090	0.005	0.045	0.081	0.042	0.151	0.171	0.099	0.086	0.059	0.037	0.086	0.081	0.052	0.105	0.100	0.116	0.080
BlurGuard	0.013	0.009	0.039	0.039	0.025	0.114	0.064	0.054	0.045	-0.004	0.044	0.022	0.036	0.013	0.083	0.026	0.062	0.035
DCT-Shield	0.152	0.020	0.094	0.095	0.059	0.144	0.234	0.157	0.119	0.163	0.057	0.126	0.134	**0.082**	0.159	0.198	0.217	0.142
PGD-C	**0.231**	0.090	0.323	0.418	0.077	0.227	0.465	0.294	0.266	**0.256**	**0.153**	0.472	0.523	0.054	**0.213**	0.535	0.326	0.317
PGD-E	0.209	0.075	0.287	0.452	0.056	0.214	0.494	0.268	0.257	0.229	0.132	0.421	0.571	0.063	0.175	0.541	0.311	0.305
Ours	0.227	**0.092**	**0.355**	**0.505**	0.078	**0.264**	**0.570**	**0.312**	**0.300**	0.255	0.140	**0.505**	**0.599**	0.068	0.185	**0.575**	**0.334**	**0.333**

Table 5 shows the extra departure from the source image after subtracting the change already caused by clean editing. TokenShield achieves the highest average increase on both datasets, reaching 0.300 on PIE-Bench and 0.333 on Emu Edit, compared with 0.266 and 0.317 for the strongest baseline averages. The difference is particularly large on NEP, where 0.505/0.599 is roughly four times the

best external-baseline values of 0.128/0.147. On EditAR, TokenShield also reaches 0.570/0.575 versus 0.234/0.209 for the strongest external baselines. These gains show substantial additional source-image changes beyond those introduced by the intended clean edit.

Table 6: Additional source-relative DINO patch distance beyond clean editing (↑).

	PIE-Bench									Emu Edit								
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.181	0.035	0.084	0.126	**0.065**	0.229	0.210	0.190	0.140	0.174	0.056	0.107	0.140	0.059	0.163	0.199	0.208	0.138
EditShield	0.167	0.060	0.042	0.117	0.037	0.249	0.143	0.187	0.125	0.137	0.087	0.081	0.110	0.044	0.197	0.125	0.178	0.120
PID	0.158	0.042	0.035	0.130	0.052	0.249	0.169	0.161	0.125	0.131	0.047	0.055	0.110	0.051	0.178	0.136	0.191	0.112
LA	0.129	0.007	0.020	0.082	0.031	0.198	0.159	0.129	0.094	0.103	0.036	0.051	0.088	0.049	0.153	0.110	0.134	0.091
BlurGuard	0.059	0.021	0.036	0.071	0.016	0.133	0.071	0.081	0.061	0.038	0.033	0.018	0.056	0.013	0.104	0.042	0.070	0.047
DCT-Shield	0.175	0.019	0.066	0.100	0.053	0.201	0.215	0.188	0.127	0.176	0.049	0.094	0.132	**0.062**	0.182	0.191	0.218	0.138
PGD-C	**0.275**	**0.095**	0.230	0.378	0.056	0.300	0.408	0.316	0.257	**0.267**	**0.130**	0.338	0.452	0.059	**0.267**	0.450	0.333	0.287
PGD-E	0.259	0.084	0.218	0.414	0.051	0.273	0.437	0.294	0.254	0.246	0.112	0.311	0.500	0.060	0.226	0.475	0.313	0.280
Ours	0.266	0.090	**0.257**	**0.458**	0.059	**0.308**	**0.485**	**0.318**	**0.280**	0.265	0.116	**0.365**	**0.519**	0.057	0.238	**0.501**	**0.335**	**0.300**

Table 6 examines changes at matching spatial positions in the DINO feature grid. TokenShield leads on Wallaroo, NEP, and EditAR on both datasets. On NEP, its scores of 0.458/0.519 exceed the strongest baseline values of 0.414/0.500 by 0.044/0.019. On EditAR, the corresponding gains are 0.048/0.026, from 0.437/0.475 to 0.485/0.501. The advantage therefore appears in spatial patch features as well as the global CLS representation, showing that the additional source-relative change extends to the image feature grid.

Paired CLIP distance. Using the CLIP image encoder ϕ_I (Radford et al., 2021), $D_{\text{CLIP}} = 1 - \cos(\phi_I(y_p), \phi_I(y_c))$ measures output change in another feature space.

Table 7: CLIP image-feature distance between edits of clean and protected inputs (↑).

	PIE-Bench									Emu Edit								
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.159	0.102	0.168	0.158	**0.208**	0.200	0.191	0.155	0.168	0.202	0.138	0.191	0.189	0.269	0.233	0.208	0.213	0.205
EditShield	0.135	0.116	0.123	0.138	0.164	0.203	0.140	0.130	0.144	0.162	0.151	0.161	0.155	0.219	0.231	0.178	0.166	0.178
PID	0.147	0.117	0.135	0.144	0.182	0.219	0.166	0.141	0.156	0.172	0.144	0.153	0.164	0.249	0.229	0.181	0.200	0.187
LA	0.123	0.099	0.126	0.138	0.191	0.197	0.158	0.125	0.145	0.154	0.120	0.145	0.159	0.238	0.208	0.165	0.168	0.170
BlurGuard	0.099	0.090	0.110	0.101	0.125	0.176	0.113	0.093	0.113	0.125	0.119	0.122	0.113	0.155	0.195	0.129	0.135	0.137
DCT-Shield	0.164	0.101	0.151	0.158	0.207	0.210	0.175	0.180	0.168	0.195	0.126	0.171	0.188	**0.291**	**0.245**	0.210	0.247	0.209
PGD-C	0.187	**0.143**	0.236	0.254	0.200	**0.229**	0.263	0.181	0.212	**0.226**	**0.178**	0.293	0.310	0.258	0.239	0.312	0.248	0.258
PGD-E	0.182	0.136	0.222	0.273	0.199	0.220	0.273	0.184	0.211	0.220	0.169	0.293	**0.359**	0.259	0.238	0.333	0.242	0.264
Ours	**0.195**	0.142	**0.251**	**0.301**	0.203	0.223	**0.300**	**0.195**	**0.226**	0.214	0.177	**0.309**	0.355	0.257	0.240	**0.334**	**0.250**	**0.267**

Table 7 measures output differences using CLIP rather than DINO. TokenShield retains the highest average distance on both datasets, with 0.226/0.267 versus the strongest baseline averages of 0.212/0.264. These are gains of 6.6% and 1.1%, respectively. On EditAR, its distances of 0.300/0.334 exceed PhotoGuard's 0.191/0.208 by 0.109/0.126. The leading averages under a different image encoder show that the output-disruption advantage is visible beyond the DINO measurements.

Paired SSIM and PSNR. SSIM (Wang et al., 2004) and PSNR compare y_p with y_c; lower values mean larger structural or pixel changes. For PSNR, exact-match output pairs are excluded before averaging. Avg. is the unweighted mean of the eight editor-wise means.

Table 8: Structural similarity measured by SSIM between edits of clean and protected inputs (↓).

	PIE-Bench									Emu Edit								
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.513	0.398	0.220	0.389	0.265	0.354	0.437	0.421	0.375	0.471	0.342	0.215	0.362	0.207	0.337	0.395	0.374	0.338
EditShield	0.590	0.399	0.325	0.492	0.340	0.364	0.508	0.480	0.437	0.529	0.331	0.282	0.453	0.233	0.344	0.454	0.424	0.381
PID	0.564	0.386	0.274	0.431	0.290	0.342	0.478	0.449	0.402	0.498	0.330	0.257	0.404	0.221	0.319	0.428	0.396	0.357
LA	0.594	0.410	0.293	0.462	0.303	0.399	0.485	0.473	0.427	0.525	0.344	0.268	0.420	0.218	0.361	0.441	0.418	0.374
BlurGuard	0.643	0.433	0.428	0.591	0.407	0.485	0.596	0.571	0.519	0.568	0.364	0.378	0.533	0.307	0.432	0.519	0.503	0.451
DCT-Shield	**0.490**	0.396	0.233	0.361	**0.261**	0.382	0.407	0.382	0.364	**0.441**	0.341	0.207	0.329	**0.202**	0.333	0.368	**0.334**	0.319
PGD-C	0.500	0.375	0.193	0.298	0.287	**0.297**	0.352	0.384	0.336	0.446	0.320	**0.174**	0.273	0.213	**0.271**	0.316	0.342	0.294
PGD-E	0.510	0.370	0.192	0.207	0.298	0.318	0.283	0.397	0.322	0.473	0.319	0.181	0.214	0.234	0.313	0.276	0.351	0.295
Ours	0.506	**0.356**	**0.171**	**0.191**	0.284	0.303	**0.276**	**0.381**	**0.309**	0.463	**0.313**	0.175	**0.198**	0.225	0.300	**0.269**	0.338	**0.285**

Table 8 shows how much of the clean-edit structure remains in the protected-input output. TokenShield has the lowest average SSIM on both datasets, at 0.309/0.285 compared with the strongest baseline averages of 0.322/0.294. On NEP, SSIM drops to 0.191/0.198, below PGD-E's 0.207/0.214, and TokenShield also leads on Janus and EditAR in both datasets. Thus, the output differences include changes in structural similarity, complementing the distances measured by learned image features.

B.3 Text Alignment

Target CLIP is $C_{\text{target}} = \cos(\phi_I(y_p), \phi_T(t_{\text{target}}))$, using the target description and CLIP text encoder ϕ_T. Lower values indicate weaker alignment with the target description.

Table 9: CLIP similarity between protected-input edits and the target text descriptions (↓).

	PIE-Bench									Emu Edit								
Method	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.	EARL	Janus	Wall.	NEP	MLN	DICE	EditAR	VQ-D	Avg.
PhotoGuard	0.230	0.259	**0.245**	0.255	**0.263**	0.266	0.225	0.255	0.250	0.214	0.231	0.220	0.226	**0.236**	**0.232**	0.214	0.240	0.227
EditShield	0.231	0.252	0.263	0.250	0.267	0.255	0.234	0.246	0.250	0.219	0.233	0.234	0.232	0.248	0.234	0.220	0.231	0.231
PID	0.240	0.263	0.259	0.263	0.269	0.257	0.238	0.249	0.255	0.218	**0.228**	0.227	0.236	0.250	0.242	0.220	0.241	0.233
LA	0.239	0.259	0.259	0.257	0.269	0.260	0.230	0.250	0.253	0.217	0.232	0.233	0.229	0.251	0.242	0.222	0.236	0.233
BlurGuard	0.236	0.260	0.256	0.254	0.269	0.260	0.248	**0.245**	0.254	0.225	0.230	0.224	0.229	0.247	0.243	0.219	**0.227**	0.231
DCT-Shield	0.241	0.261	0.253	0.259	0.266	0.258	0.235	0.252	0.253	0.215	0.232	0.215	0.228	0.242	0.234	0.216	0.234	0.227
PGD-C	0.233	**0.249**	0.249	0.229	0.268	0.245	0.209	0.248	0.241	0.215	0.231	0.221	0.195	0.248	0.243	0.190	0.232	0.222
PGD-E	**0.229**	0.254	0.259	0.229	**0.263**	**0.242**	0.202	0.252	0.241	**0.211**	0.233	0.218	0.190	0.254	0.238	0.187	0.232	0.220
Ours	0.231	0.254	0.247	**0.209**	0.273	0.249	**0.194**	0.249	**0.238**	0.214	0.230	**0.210**	**0.181**	0.245	0.237	**0.172**	0.228	**0.215**

Table 9 tests how closely the protected-input edit matches the requested target description. TokenShield achieves the lowest average similarity on both datasets, at 0.238/0.215 compared with the strongest baseline averages of 0.241/0.220. Its advantage is especially clear on EditAR, where 0.194/0.172 is below PGD-E's 0.202/0.187. TokenShield also gives the lowest target similarity on NEP in both datasets, reaching 0.209/0.181. On these editors, the protection reduces semantic agreement with the requested result as well as changing the edited image.

B.4 Qualitative Comparison with PGD Baselines

Figure 4 compares PGD-C, PGD-E, and TokenShield through protected inputs and their edited outputs. TokenShield preserves the appearance of the input images while strongly disrupting the edited results. In the armored-dog example, both baselines retain a recognizable dog, whereas TokenShield largely destroys its shape. In the lipstick-color example, TokenShield produces more severe facial distortion than either baseline. These examples illustrate how TokenShield can disrupt recognizable image content beyond the changes produced by the continuous-feature and soft-assignment objectives.

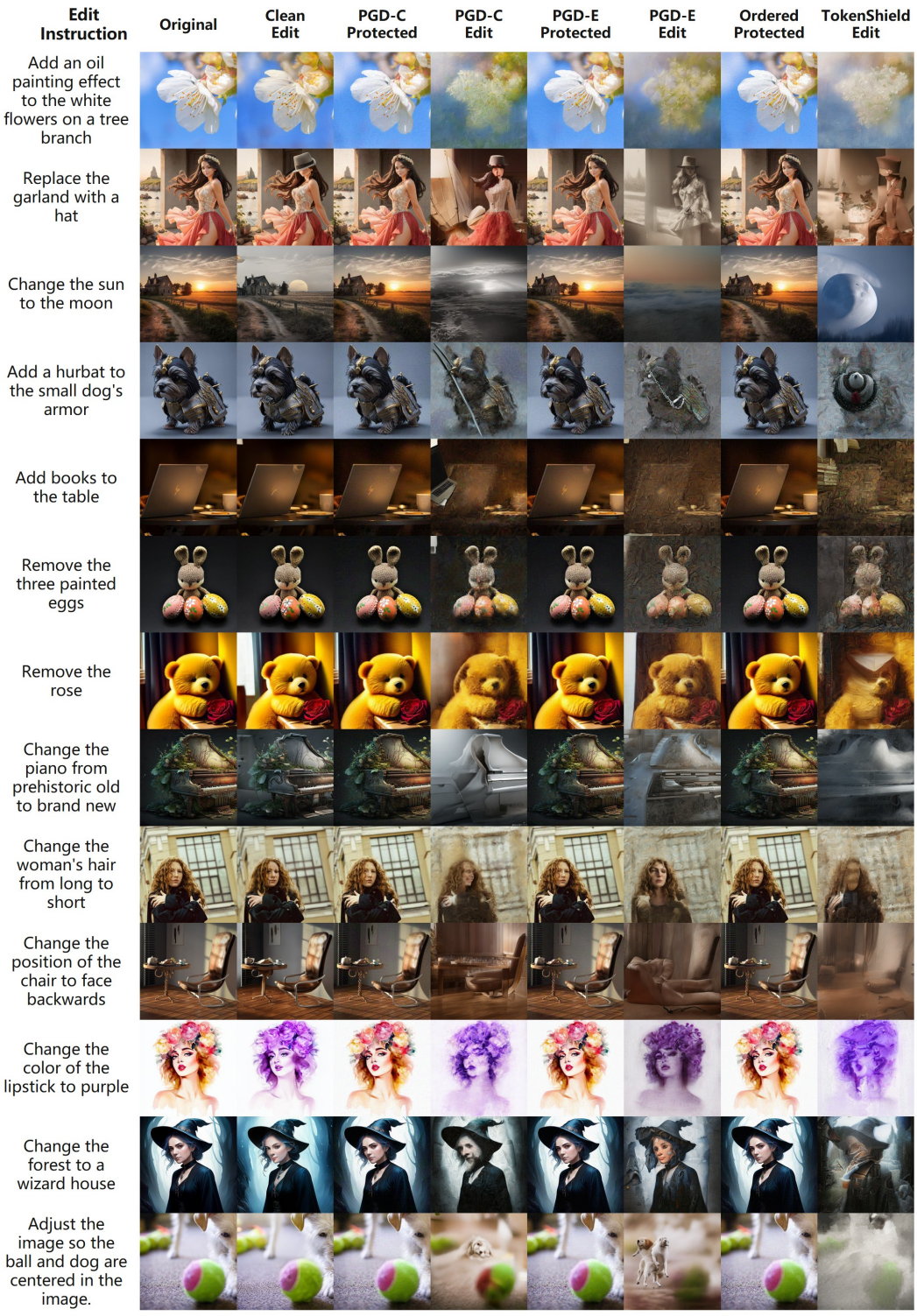


Figure 4: Qualitative examples of image protection and its effects on downstream editing results.