FROM UNLEARNABLE OUTCOMES TO DECODABLE SHORTCUT CODES

Anonymous authors
Paper under double-blind review

ABSTRACT

Unlearnable examples (UEs) add small perturbations to training images so that models trained on them perform poorly on clean data. Low clean accuracy, however, does not reveal what these models learn. We find that shallow networks fit protected training sets more easily than clean ones while generalizing worse. Moving perturbations between images then separates two behaviors: predictions either follow the label associated with the perturbation regardless of the image, or depend on the combination of perturbation and image. The first behavior offers a direct route to unlearnability, because a model that decodes the perturbation needs no semantic features to fit the protected data. We formalize this route as a decodable perturbation shortcut (DPS) and propose a low-complexity shortcut codebook (LCSC), which learns one code per class that randomly initialized networks can identify on changing images. At $8/255$ with crop-and-flip augmentation, LCSC reduces clean accuracy to near chance on four datasets and five architectures; at $1/255$, its epoch-60 accuracy on CIFAR-10 is 9.8%. Its factorized variant gives the strongest protection in eight of nine JPEG settings and at adversarial-training radii of $2/255$ and $4/255$. Both variants are constructed tens to hundreds of times faster than TAP, SEP, and REM.

1 INTRODUCTION

Publicly available images can be collected and used to train models without their owners' consent. Unlearnable examples (UEs) aim to deter such use by adding small, visually imperceptible perturbations before release, so that models trained on the released images perform poorly on clean test data (Huang et al., 2021). Methods are usually compared by this outcome. Low test accuracy shows that protection works, but not which predictive rule the model has learned in place of the intended task. Knowing this rule could explain why existing methods work and guide the design of new perturbations. We therefore ask: *what do models learn from unlearnable data?*

Shortcut learning offers a starting point. Synthetic perturbations (SP) are linearly separable by class (Yu et al., 2022), simple networks can detect several UE families (Zhu et al., 2024), and perturbation-induced linearization (PIL) constructs perturbations with linear surrogate models and studies the resulting linear behavior of deep models (Liu et al., 2026). Linear separability is nevertheless not necessary for an effective UE, as autoregressive (AR) perturbations show (Sandoval-Segura et al., 2023). These accounts characterize the perturbations. We complement them with a behavioral test: separate a perturbation from its image and observe what the trained model predicts.

We first compare clean and protected training sets using small MLPs and single-convolution networks: the protected sets are generally *easier* to fit but generalize worse. We then move perturbations to images of other classes and record the predictions of frozen victims (Figure 1). Two behaviors recur. Under *perturbation coding*, predictions follow the label associated with the perturbation, whatever the image; PIL, SP, and AR behave this way. Under *image–perturbation coupling*, the response depends on the image that carries the perturbation; REM, TAP, and SEP behave this way, and EM combines both. The behaviors also differ on clean images. For TAP and SEP, consistently relabeling the model's outputs recovers much higher accuracy, so class distinctions survive under wrong names; for the coding methods, relabeling recovers almost nothing.

Coding therefore suggests a direct design principle: if the perturbation alone identifies the label on any image, a model can fit the protected data by decoding it, without learning semantic features. Ex-

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.

isting coding perturbations obtain this property from fixed generators (SP, AR) or linear surrogates (PIL), and in our experiments they lose much of their protection under JPEG compression and small-radius adversarial training. We instead make decodability the construction objective. We formalize it as a *decodable perturbation shortcut* (DPS) and propose a **low-complexity shortcut codebook** (LCSC): one code per class, learned together with a neural reader that must identify each code on changing images and is periodically reset (Figure 2). Victims trained on the protected data follow the codes on all cross-class transfers, and the codes protect data across datasets and architectures.

Our contributions are:

- We show that shallow networks fit protected training data more easily than clean data, to different degrees across UE methods.
- We introduce a perturbation-transfer test that separates perturbation coding from image–perturbation coupling, and show that the two behaviors leave different class structure in the trained model.
- We formalize coding as a construction objective and propose LCSC and its factorized variant, which combine strong protection under JPEG and small-radius adversarial training with much lower construction cost than TAP, SEP, and REM.

2 Related Work

Unlearnable examples. Unlearnable examples modify training inputs to impair the generalization of models trained on them. Error minimization (EM) alternates surrogate training with perturbation updates that reduce the surrogate's loss on the modified images (Huang et al., 2021). Robust error minimization (REM) adds adversarial training to this construction so that the perturbations resist adversarial learning (Fu et al., 2022). Targeted adversarial poisoning (TAP) instead moves images toward target classes of a fixed, trained surrogate (Fowl et al., 2021), and self-ensemble protection (SEP) combines gradients from training checkpoints to mislead several stages of learning (Chen et al., 2023). Other methods need no trained surrogate: SP generates linearly separable synthetic patterns (Yu et al., 2022), and AR samples class-specific autoregressive processes (Sandoval-Segura et al., 2022).

Shortcut learning. Neural networks can rely on predictive cues that do not capture the intended task (Geirhos et al., 2020), and they often prefer the simplest available cue (Shah et al., 2020). UE analyses have linked protection to such cues: SP builds them through linear separability (Yu et al., 2022), detection methods exploit their shallow learnability (Zhu et al., 2024), and PIL induces linear behavior through linear surrogates (Liu et al., 2026), while AR shows that linear separability is not required (Sandoval-Segura et al., 2023). We examine the trained models instead, by moving perturbations between images, and use the observed behavior as a construction target.

Countermeasures. Image Shortcut Squeezing (ISS) shows that simple compression, especially JPEG (Wallace, 1991), restores much of the accuracy lost to many UEs (Liu et al., 2023), whereas common augmentations such as Cutout (DeVries & Taylor, 2017), CutMix (Yun et al., 2019), and MixUp (Zhang et al., 2018) have more limited effects. Adversarial training is also a strong countermeasure (Tao et al., 2021), at the cost of inner-loop attacks during training. We evaluate LCSC under JPEG compression, adversarial training, and standard augmentations.

3 What Do Models Learn from Unlearnable Data?

3.1 Threat model and objective

Threat model. Following EM (Huang et al., 2021), the defender is a data owner who modifies images before release to deter unauthorized training. The defender has the original images and their ground-truth labels, and adds small perturbations without changing the labels. The unauthorized learner obtains the released images and labels, but not the original images or the perturbations, and controls how it trains on the released data. The defender aims to reduce the resulting model's accuracy on clean, unseen images.

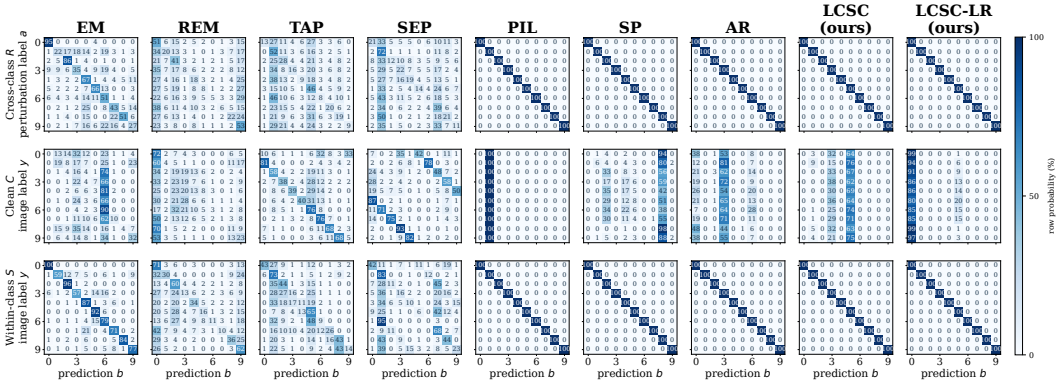

Figure 1: **Perturbation-transfer predictions.** Top to bottom: cross-class transfer R, clean confusion C, and within-class transfer S. Cells show rounded row percentages.

Notation and objective. Let $\mathcal{D}_c = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ be the clean training set and $\mathcal{D}_t = \{(\boldsymbol{x}_j^{\mathrm{t}}, y_j^{\mathrm{t}})\}_{j=1}^{N_t}$ the clean test set, with images in $\mathcal{X} = [0,1]^{C\times H\times W}$ and labels in $\mathcal{Y} = \{0,\dots,K-1\}$. Here C, H, W are the channel count and spatial dimensions, and K is the number of classes. Sampling from either set means sampling its empirical distribution. We write $\boldsymbol{x} \oplus \boldsymbol{\delta} = \mathrm{clip}_{[0,1]}(\boldsymbol{x} + \boldsymbol{\delta})$ and require $\|\boldsymbol{\delta}\|_\infty \le \varepsilon$.

The released set is $\mathcal{D}_u(\boldsymbol{\delta}) = \{(\boldsymbol{x}_i \oplus \boldsymbol{\delta}_i, y_i)\}_{i=1}^{N}$, where $\boldsymbol{\delta} = (\boldsymbol{\delta}_1, \dots, \boldsymbol{\delta}_N)$. Let $f_{\boldsymbol{\theta}}$ be a classifier with parameters $\boldsymbol{\theta}$, and let ℓ be cross-entropy. The idealized UE objective is

$$\begin{aligned} \max_{\boldsymbol{\delta}} \quad & \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_t}\left[\ell(f_{\boldsymbol{\theta}^\star(\boldsymbol{\delta})}(\boldsymbol{x}), y)\right] \\ \text{s.t.} \quad & \boldsymbol{\theta}^\star(\boldsymbol{\delta}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x}_i,y_i)\sim\mathcal{D}_c}\left[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i \oplus \boldsymbol{\delta}_i), y_i)\right]. \end{aligned} \tag{1}$$

The inner problem is the learner's ordinary training on the released data; the outer problem asks the perturbations to make the resulting model fail on clean test images. In practice, a finite training schedule approximates the inner problem, and clean test accuracy measures protection. This objective scores only the outcome. The rest of this section examines the rule that produces it.

3.2 Protected data are easier to fit

We train two shallow classifier families on either clean or protected CIFAR-10: MLP-h has one hidden layer of width h; Conv-c has one 3×3 convolution with c channels and a fully connected classifier. Both use 60 epochs, no augmentation, and SGD with learning rate 0.01. Training accuracy is measured on the corresponding clean or protected training set, and test accuracy on clean images. Table 1 shows the smallest and largest settings; Appendix D reports the complete sweeps.

Table 1: Shallow-probe accuracy on CIFAR-10 (%). Higher Train means easier fitting; lower Test means stronger protection.

	MLP-32		MLP-1024		Conv-4		Conv-128	
Method	Train	Test	Train	Test	Train	Test	Train	Test
Clean	53.2	48.8	76.2	56.7	63.3	51.9	96.2	63.9
EM (Huang et al., 2021)	99.6	21.3	99.9	25.1	99.9	20.2	100.0	25.9
REM (Fu et al., 2022)	75.0	39.5	92.1	48.7	98.9	28.9	100.0	48.9
TAP (Fowl et al., 2021)	55.1	46.6	79.4	55.2	72.2	45.3	99.9	53.8
SEP (Chen et al., 2023)	61.3	43.5	83.7	52.9	80.4	36.8	99.8	54.3
PIL (Liu et al., 2026)	100.0	11.9	100.0	12.8	100.0	11.7	100.0	14.1
SP (Yu et al., 2022)	100.0	19.6	100.0	22.2	100.0	18.5	100.0	22.4
AR (Sandoval-Segura et al., 2022)	51.6	47.6	76.6	56.2	89.5	35.0	99.9	41.6
LCSC (ours)	100.0	26.1	100.0	25.9	100.0	21.8	100.0	23.8
LCSC-LR (ours)	100.0	18.0	100.0	19.2	100.0	14.5	100.0	16.3

Protected training sets are easier to fit but generalize worse (Table 1). With only 32 hidden units, the MLP reaches at least 99.6% training accuracy on EM, PIL, SP, LCSC, and LCSC-LR, com-

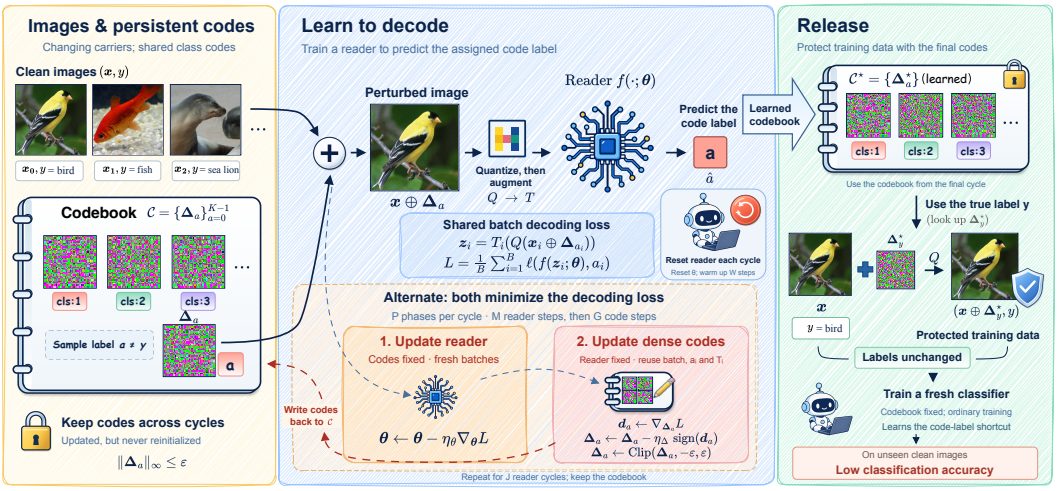

Figure 2: **Architecture illustration of our LCSC.** Best viewed in color. Zoom in for details.

pared with 53.2% on clean data. At every convolutional width, every tested UE has higher training accuracy and lower clean test accuracy than clean training. AR is the exception among MLPs, but four convolutional channels raise its training accuracy from the clean 63.3% to 89.5%. TAP, REM, and SEP remain harder for the small probes to fit. Easier fitting shows that protected data admit a simpler rule than semantic classification, but not what that rule is.

3.3 Perturbation coding and image–perturbation coupling

Perturbation classes and sampling. For each class $a \in \mathcal{Y}$, let $\mathcal{Q}(\cdot \mid a)$ be the empirical distribution of perturbations originally applied to training images with label a. Sampling $\boldsymbol{\delta} \sim \mathcal{Q}(\cdot \mid a)$ means choosing one of those perturbations uniformly. The joint distribution $(\boldsymbol{\delta}, a) \sim \mathcal{Q}$ first samples a uniformly and then a perturbation from that class. We attach the perturbation to a different image to test whether its label association is preserved.

Predictions under perturbation transfer. For a frozen model, $\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{u}) = \arg\max_{b \in \mathcal{Y}} f_{\boldsymbol{\theta}}(\boldsymbol{u})_b$ is the predicted class. Draw $(\boldsymbol{x}, y) \sim \mathcal{D}_t$ and $(\boldsymbol{\delta}, a) \sim \mathcal{Q}$ independently. Here y is the original label of $\boldsymbol{x}$, a is the perturbation's assigned label, and b is the prediction. Cross-class transfer R places a perturbation on an image of another class, clean confusion C evaluates unmodified images, and within-class transfer S places it on another image of the same class:

$$\begin{aligned} R_{ab} &= \Pr[\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{x} \oplus \boldsymbol{\delta}) = b \mid a,\ y \neq a], \\ C_{yb} &= \Pr[\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{x}) = b \mid y], \\ S_{yb} &= \Pr[\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{x} \oplus \boldsymbol{\delta}) = b \mid y,\ a = y]. \end{aligned} \tag{2}$$

The *code-following rate* (CFR) measures how often a cross-class prediction follows the perturbation label. *Optimal mapping accuracy* measures clean classification after the best global relabeling of model outputs:

$$\text{CFR} = \frac{1}{K}\sum_{a \in \mathcal{Y}} R_{aa}, \qquad \text{Acc}_{\text{map}}(C) = \max_{\sigma \in \mathfrak{S}_K} \frac{1}{K}\sum_{y \in \mathcal{Y}} C_{y,\sigma(y)}. \tag{3}$$

Here $\mathfrak{S}_K$ contains all bijections of the K labels, so mapping accuracy is an oracle measure of how many class distinctions survive on clean images. Table 2 reports these measures, Figure 1 shows the matrices, and Appendix P gives larger panels with cell values.

Perturbation coding. Predictions follow the perturbation's assigned label on images of any class. PIL, SP, and AR reach nearly 100% CFR and within-class accuracy (Table 2): replacing the image leaves the prediction unchanged. PIL builds its perturbations with a linear surrogate, whereas SP and AR generate class-dependent patterns without one. Each design yields a label cue that can be

Table 2: Perturbation-transfer diagnostics on CIFAR-10/ResNet-18 with Basic training (%).

Method	Clean acc.	Code following	Mapping acc.	Within-class acc.
EM (Huang et al., 2021)	19.2	54.4	21.3	80.0
REM (Fu et al., 2022)	16.9	23.2	19.2	34.3
TAP (Fowl et al., 2021)	7.3	22.0	57.8	34.4
SEP (Chen et al., 2023)	1.8	23.7	68.5	30.5
PIL (Liu et al., 2026)	10.0	100.0	10.0	100.0
SP (Yu et al., 2022)	16.5	99.9	17.2	100.0
AR (Sandoval-Segura et al., 2022)	14.2	100.0	16.1	100.0
LCSC (ours)	11.1	100.0	12.2	100.0
LCSC-LR (ours)	11.9	100.0	11.9	100.0

recognized without complex image features, consistent with the shallow-probe results. EM, with 54.4% CFR, lies between coding and coupling.

Image–perturbation coupling. The response to a perturbation changes with the image that carries it. REM, TAP, and SEP have 22–24% CFR and 30–34% within-class accuracy, and changing the image class while holding the perturbation class fixed changes the prediction distribution (Appendix E). A plausible explanation lies in their constructions: TAP optimizes against a trained surrogate, SEP against training checkpoints, and REM against an adversarially trained surrogate. Perturbations can then act through image features these models have already learned, rather than as a label cue that stands alone.

What remains on clean images. The two behaviors also differ on clean inputs. The best global relabeling raises TAP's clean accuracy from 7.3% to 57.8% and SEP's from 1.8% to 68.5%: these models still separate many classes but give them the wrong names, which appears as systematic off-diagonal entries in the clean confusion matrices of Figure 1. One possible explanation is that the perturbations teach a model to attach one class's visual features to another label. For PIL, SP, and AR, relabeling adds at most 1.9 points, indicating that their victims' clean predictions carry little class structure.

3.4 Decodable perturbation shortcuts

Coding is the more direct route to unlearnability. A model that decodes the perturbation can fit the protected data without semantic features, and the diagnostics above show that such models leave little class structure to recover on clean images. We therefore turn coding into a construction objective: learn perturbations whose labels are predicted even when they are added to new images. We call this a *decodable perturbation shortcut* (DPS). Let $\mathcal{Q}_{\boldsymbol{\phi}}$ be a learnable distribution of bounded perturbations and their labels, parameterized by $\boldsymbol{\phi}$, with uniformly sampled labels. During classifier training, each image receives a perturbation with its own label; during evaluation, images and perturbation labels are sampled independently. The objective is

$$\begin{aligned} \min_{\boldsymbol{\phi}} \quad & \mathbb{E}_{\substack{(\boldsymbol{x},y)\sim\mathcal{D}_t \\ (\boldsymbol{\delta},a)\sim\mathcal{Q}_{\boldsymbol{\phi}}}} \left[\ell\big(f_{\boldsymbol{\theta}^\star(\boldsymbol{\phi})}(\boldsymbol{x}\oplus\boldsymbol{\delta}), a\big)\right] \\ \text{s.t.} \quad & \boldsymbol{\theta}^\star(\boldsymbol{\phi}) = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\substack{(\boldsymbol{x},y)\sim\mathcal{D}_c \\ \boldsymbol{\delta}\sim\mathcal{Q}_{\boldsymbol{\phi}}(\cdot|a=y)}} \left[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}\oplus\boldsymbol{\delta}), y)\right]. \end{aligned} \tag{4}$$

The inner problem is ordinary training on the protected dataset. The outer problem asks the trained classifier to predict the perturbation label a on independently paired images, including when $a \neq y$. Minimizing this loss under $\|\boldsymbol{\delta}\|_\infty \leq \varepsilon$ favors perturbations that the classifier recognizes on any image. A classifier that relies entirely on this cue has no informative rule once the cue is absent, so its clean accuracy can fall to chance. Solving Eq. 4 directly would require training a victim inside the optimization and evaluating it on test images; Section 4 replaces the victim with a reader trained on training images.

4 Low-Complexity Shortcut Codebook

LCSC approximates the DPS objective with a *construction reader*: a network trained on clean training images to identify which code has been added. The reader must name the code on images

of other classes, mirroring the cross-class evaluation in Eq. 4, and it is periodically reinitialized, so the codes must remain learnable by a network trained from scratch. The codebook is low-complexity in that it stores one perturbation per class and is optimized to be decoded quickly by a fresh reader. After construction, each image receives the code of its true class. Figure 2 shows the workflow; perturbation colors are enhanced for visibility.

4.1 Codebook and decoding objective

The codebook $\mathcal{C} = \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$ contains one bounded perturbation per class. Given $(\boldsymbol{x}, y) \sim \mathcal{D}_c$, we sample a uniformly from $\mathcal{Y} \setminus \{y\}$ and look up $\boldsymbol{\Delta}_a$. After clipped addition, quantization, and augmentation, the reader $f(\cdot; \boldsymbol{\theta})$ predicts a, not y, using

$$\mathcal{L}_{\text{code}}(\boldsymbol{\theta}, \mathcal{C}) = \mathbb{E}\left[\ell(f(T(Q(\boldsymbol{x} \oplus \boldsymbol{\Delta}_a)); \boldsymbol{\theta}), a)\right]. \tag{5}$$

Equation 5 trains the reader to answer one question: which code was added to this image? Here $\boldsymbol{x}$ is the carrier image, y its true label, $a \neq y$ the sampled code label, and $\boldsymbol{\Delta}_a$ its codeword. The operator $\oplus$ adds the code and clips pixel values to $[0, 1]$; $Q(\boldsymbol{z}) = \text{round}(255\boldsymbol{z})/255$ rounds them to 8 bits, as in the released images; $T \sim \mathcal{T}$ applies a sampled augmentation. The expectation averages over images, code labels, and transformations, so each code must remain recognizable on changing images. Figure 2 abbreviates the minibatch loss as L.

4.2 Cooperative updates with persistent codes

Each cycle warms up a fresh reader for W steps with fixed codes, then runs P phases of M reader steps followed by G projected code steps:

$$\begin{aligned} \text{Reader:} \quad & \min_{\boldsymbol{\theta}} \mathcal{L}_{\text{code}}(\boldsymbol{\theta}, \mathcal{C}) \quad \text{with } \mathcal{C} \text{ fixed,} \\ \text{Codebook:} \quad & \min_{\mathcal{C}:\, \|\boldsymbol{\Delta}_a\|_\infty \le \varepsilon} \mathcal{L}_{\text{code}}(\boldsymbol{\theta}, \mathcal{C}) \quad \text{with } \boldsymbol{\theta} \text{ fixed.} \end{aligned} \tag{6}$$

The two updates cooperate rather than compete: the reader learns to recognize the current codes, and the code update makes those codes easier for the current reader to recognize. Reader steps draw fresh batches. Each code phase reuses one newly sampled batch, labels a_i, and transforms T_i for all G steps, with reader weights and normalization statistics fixed. Dense codes take sign-gradient steps followed by clipping to $[-\varepsilon, \varepsilon]$, and the identity straight-through estimator passes gradients through Q. Appendix A gives both algorithms and all settings.

Codes persist across J cycles, whereas the reader and its momentum buffer are reset between cycles. We use $(W, P, M, G) = (100, 200, 10, 30)$, with $J = 1$ for ImageNet-100 and $J = 2$ for the other datasets. At $8/255$, neither the reset nor cross-class pairing is required for protection (Table 7), but reducing the reader updates to $M = 1$ raises victim accuracy to 63.8%: the codes must be optimized against a reader that has learned to decode them.

4.3 Release

After all J cycles, we release the final codebook $\mathcal{C}^\star = \mathcal{C}^{(J)} = \{\boldsymbol{\Delta}_a^\star\}_{a=0}^{K-1}$. Release uses true-label codes and quantization, without construction augmentation:

$$\mathcal{D}_u = \left\{(Q(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}_{y_i}^\star), y_i)\right\}_{i=1}^{N}. \tag{7}$$

Each image receives the final code of its true label and keeps that label. The construction reader is discarded, and only the protected data are released. An unauthorized learner then trains its own classifier on these data; our victim experiments simulate this step.

4.4 Factorized codewords

LCSC-LR uses a rank-1 latent code in each channel: $\boldsymbol{\Delta}_{ac} = \varepsilon \boldsymbol{u}_{ac} \boldsymbol{v}_{ac}^\top$, with $\boldsymbol{u}_{ac} \in [-1, 1]^H$ and $\boldsymbol{v}_{ac} \in [-1, 1]^W$. We optimize the factors with the decoding loss and a factor-balance regularizer, projecting each factor to keep the latent perturbation within budget. The codebook then has $KC(H + W)$ parameters instead of $KCHW$. The rank constraint applies before clipping and quantization. Algorithm 2 gives the updates; Appendix A defines the regularizer and a general blockwise form.

5 EXPERIMENTS

5.1 EXPERIMENTAL PROTOCOL

Our main experiments protect the entire training set and train victims from scratch. We evaluate SVHN (Netzer et al., 2011), CIFAR-10/100 (Krizhevsky, 2009), and a 100-class ImageNet subset (Deng et al., 2009; Russakovsky et al., 2015). Unless stated otherwise, the victim is ResNet-18 (He et al., 2016), trained from scratch for **60 epochs** using SGD (Bottou, 2010) with learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, cosine decay (Loshchilov & Hutter, 2017), and batch size 128 (ImageNet-100 exceptions are in Appendix C). Basic augmentation is random crop plus horizontal flip. Reported victim accuracies are measured at epoch 60 unless a table states otherwise.

Construction uses a ResNet-18 reader. Baselines include EM, TAP, SEP, REM, PIL, SP, AR, and the original class-wise EM implementation, denoted $\text{EM}^{(\text{C})}$ (Huang et al., 2021) in tables. Gaussian shares one unoptimized random perturbation per class. SP and AR use their native ℓ_2 perturbations. CUDA's content-dependent transform (Sadasivan et al., 2023) cannot be transplanted as an additive perturbation. Appendix C gives budgets and schedules. In countermeasure tables, bold and underline mark the lowest and second-lowest protected accuracy.

5.2 VICTIMS DECODE THE CODES

Victims trained on either codebook show the behavior targeted by DPS: they follow transplanted codes on 100.0% of cross-class transfers, and relabeling their clean outputs gains at most 1.1 points (Table 2). In direct controls at $1/255$, a dense LCSC victim predicts the attached code label on 100% of inputs, even on neutral carriers, yet reaches only 9.8% on clean images (Appendix E). The codes are not tied to the construction reader: a freshly initialized ResNet-18 trained on the frozen codes exceeds 90% decoding accuracy within 1,300 updates, reaching 99.8% at $8/255$ with Basic augmentation and 100.0% at $1/255$ without augmentation (Appendix K).

5.3 DATASETS AND ARCHITECTURES

At $\varepsilon = 8/255$ with Basic augmentation, LCSC lowers clean accuracy to near chance on all 20 dataset–architecture pairs in Table 3: about 10% on the ten-class datasets and 1% on the hundred-class datasets. Codes built with a ResNet-18 reader transfer to ResNet-50, VGG-19 (Simonyan & Zisserman, 2015), DenseNet-121 (Huang et al., 2017), and MobileNet-V2 (Sandler et al., 2018), so protection does not require the victim to share the reader's architecture or parameters. At smaller budgets, transfer is less uniform; VGG-19 at $1/255$ is the main exception (Appendix G).

Table 3: Cross-dataset and cross-architecture clean test accuracy (%) at $8/255$. $\mathcal{D}_c$: clean training; $\mathcal{D}_u$: protected training. Lower $\mathcal{D}_u$ accuracy means stronger protection.

Model	SVHN		CIFAR-10		CIFAR-100		ImageNet-100	
	$\mathcal{D}_c$	$\mathcal{D}_u$	$\mathcal{D}_c$	$\mathcal{D}_u$	$\mathcal{D}_c$	$\mathcal{D}_u$	$\mathcal{D}_c$	$\mathcal{D}_u$
ResNet-18	96.6	11.1	94.5	10.6	75.9	1.0	83.9	1.0
ResNet-50	96.7	10.9	94.5	10.0	76.5	1.0	85.0	1.0
VGG-19	96.2	11.1	93.0	10.7	68.8	1.2	81.4	1.0
DenseNet-121	96.9	11.1	95.4	10.9	78.7	1.0	42.8	1.0
MobileNet-V2	96.4	11.1	93.0	10.1	73.4	1.0	75.0	1.5

5.4 JPEG COMPRESSION

JPEG compression is an inexpensive countermeasure against many UEs (Liu et al., 2023). Table 4 adds compression at nine qualities to Basic augmentation during victim training. The coding baselines lose much of their protection even under mild compression: at JPEG90, SP, AR, and PIL allow 66.3%, 92.9%, and 44.0% accuracy. LCSC-LR gives the lowest accuracy from JPEG80 to JPEG10 and stays between 10.4% and 17.2% down to JPEG30, where the strongest baseline, class-wise EM, reaches 42.3% and dense LCSC 54.4%. SEP is strongest at JPEG90 (8.4%) but rises to 67.2% at JPEG70. At JPEG10, LCSC-LR also fails, reaching 77.7% against the 85.0% clean baseline. Appendix F reports other augmentations.

Table 4: Protection against JPEG preprocessing on CIFAR-10. ResNet-18 clean test accuracy (%); lower is better.

Method	High quality			Mid quality			Low quality		
	JPEG90	JPEG80	JPEG70	JPEG60	JPEG50	JPEG40	JPEG30	JPEG20	JPEG10
Clean	93.0	92.2	91.6	91.5	90.7	90.2	89.7	88.6	85.0
EM (Huang et al., 2021)	55.1	89.9	89.3	89.2	88.8	89.1	88.5	87.4	84.9
REM (Fu et al., 2022)	54.4	70.1	74.4	77.8	79.8	81.5	83.4	85.7	84.6
TAP (Fowl et al., 2021)	15.2	40.6	61.0	74.2	80.5	84.2	86.0	86.7	84.5
SP (Yu et al., 2022)	66.3	69.4	76.1	78.1	81.6	83.2	85.7	87.0	85.0
SEP (Chen et al., 2023)	**8.4**	36.6	67.2	80.0	85.4	87.8	88.5	87.8	84.7
AR (Sandoval-Segura et al., 2022)	92.9	91.9	91.8	91.1	90.8	90.6	89.5	88.4	85.1
PIL (Liu et al., 2026)	44.0	51.9	57.3	61.0	61.6	67.5	71.9	75.7	81.5
$\text{EM}^{(C)}$ (Huang et al., 2021)	15.8	17.9	21.5	22.3	28.7	30.1	42.3	60.4	79.8
Gaussian	93.3	92.4	91.3	91.2	90.6	90.6	89.6	88.4	85.1
LCSC (ours)	20.6	18.8	15.5	16.8	21.5	31.2	54.4	77.5	83.5
LCSC-LR (ours)	10.4	**11.2**	**13.2**	**16.0**	**15.7**	**15.2**	**17.2**	**29.0**	**77.7**

5.5 Adversarial training

Adversarial training changes the learner's optimization and is a strong UE countermeasure (Tao et al., 2021). We train victims with PGD-7 (Madry et al., 2018) at the radii ρ in Table 5. REM is the natural reference because it constructs perturbations against an adversarially trained surrogate (Fu et al., 2022), whereas LCSC does not model adversarial training. At $\rho = 2/255$ and $4/255$, LCSC-LR gives the lowest accuracies, 10.5% and 21.6%, compared with REM's 26.9% and 34.0%. Dense LCSC reaches 17.7% and 45.1%, outperforming REM at the smaller radius.

Table 5: Protection against PGD-7 adversarial training on CIFAR-10. ResNet-18 clean test accuracy (%); lower is better.

Method	$\rho = 2/255$	$\rho = 4/255$	$\rho = 8/255$	$\rho = 16/255$
Clean	92.4	90.1	85.0	68.6
EM (Huang et al., 2021)	71.0	86.5	84.7	68.2
REM (Fu et al., 2022)	26.9	34.0	86.0	69.8
TAP (Fowl et al., 2021)	25.4	88.5	82.7	**66.6**
SEP (Chen et al., 2023)	68.6	88.3	83.2	67.3
PIL (Liu et al., 2026)	58.4	87.5	83.9	66.9
AR (Sandoval-Segura et al., 2022)	54.7	80.8	82.9	68.2
SP (Yu et al., 2022)	69.0	89.2	84.3	68.2
$\text{EM}^{(C)}$ (Huang et al., 2021)	20.0	73.6	85.2	70.6
Gaussian	92.5	89.8	85.0	68.4
LCSC (ours)	17.7	45.1	85.8	68.4
LCSC-LR (ours)	**10.5**	**21.6**	**82.6**	70.6

5.6 Very small perturbation budgets

Table 6 compares best-epoch (Peak) and epoch-60 (Final) accuracy on CIFAR-10/ResNet-18 without augmentation. Both codebooks end near chance at every budget from $1/255$ to $8/255$. At $1/255$, dense LCSC and LCSC-LR finish at 9.8% and 10.4%, whereas TAP, SEP, and REM remain above 84%. The trajectories matter at this budget: the two codebooks peak at 66.0% and 56.5% before collapsing, and validation-based checkpoint selection recovers similar accuracy (Appendix L). From $2/255$ upward, their peaks stay at or below 14.2%. EM instead stays near 25% throughout training at $1/255$, far below the 88.8% of clean training.

Table 6: Low-budget CIFAR-10/ResNet-18 clean accuracy (%) without augmentation. Peak: best test epoch; Final: epoch 60. Lower is better.

	$1/255$		$2/255$		$4/255$		$8/255$	
Method	Peak	Final	Peak	Final	Peak	Final	Peak	Final
EM (Huang et al., 2021)	25.3	25.2	16.1	11.1	11.8	10.1	14.2	10.1
TAP (Fowl et al., 2021)	84.4	84.3	70.6	62.2	59.9	19.6	51.7	9.6
SEP (Chen et al., 2023)	87.1	87.0	74.2	36.5	62.0	5.5	55.3	2.2
REM (Fu et al., 2022)	88.1	88.0	52.7	20.2	20.8	18.5	17.9	14.7
PIL (Liu et al., 2026)	75.7	10.2	48.8	10.1	24.7	9.8	14.6	10.2
AR (Sandoval-Segura et al., 2022)	88.3	88.3	88.2	88.1	65.6	42.9	55.5	21.2
SP (Yu et al., 2022)	88.5	88.5	78.8	52.1	60.4	20.3	29.3	15.1
$\text{EM}^{(C)}$ (Huang et al., 2021)	19.7	16.5	19.2	11.2	15.4	11.8	14.2	10.4
Gaussian	88.7	88.7	88.4	88.3	77.3	12.3	73.4	10.4
LCSC (ours)	66.0	9.8	11.0	10.0	12.4	10.0	11.2	9.6
LCSC-LR (ours)	56.5	10.4	14.2	11.9	11.2	10.1	11.1	9.9

5.7 Construction cost

At $8/255$ with Basic construction augmentation, dense LCSC and LCSC-LR take 5.1 and 20.3 minutes, versus 23.0, 16.4, and 37.1 hours for TAP, SEP, and REM (Appendix H). Under its fixed schedule, dense LCSC takes about five minutes at every budget, whereas EM stops once its surrogate exceeds 99% accuracy on the protected images and takes 2.7 hours at $1/255$. PIL is faster (1.7 minutes) but protects less than LCSC-LR under JPEG and small-radius adversarial training.

6 Discussion

What models learn from UEs. A classifier fits whatever rule predicts its training labels, not necessarily the semantic rule we intend; protected data are unlearnable when that rule does not transfer to clean images. Our diagnostics show two ways of supplying such a rule: the perturbation alone can carry the label, or the label can depend on how the perturbation interacts with the image. PGD against a trained surrogate, for example, can make a particular image–perturbation pair predictive of a label; for TAP and SEP, the resulting rule leaves class distinctions that relabeling recovers. LCSC takes the coding route directly. Because the same code appears on varied images, decoding it can reduce the training loss faster than learning features that generalize, consistent with shortcut learning (Geirhos et al., 2020) and simplicity bias (Shah et al., 2020). Appendix N discusses how compression and training dynamics relate to this account.

7 Scope and Limitations

Protection depends on the training horizon: at $1/255$, validation-based early stopping recovers 65.3% for dense LCSC and 55.1% for LCSC-LR (Appendix L). JPEG quality 10 and adversarial training at $\rho \geq 8/255$ remove most of the protection. Class-shared codes expose repeated structure to a learner with the released images and labels; the one estimation-and-removal attack we tested reaches 0.864 cosine similarity with the dense codes but raises clean accuracy by only two points (Appendix M), and other label-aware attacks remain untested. Most results are single seed-0 runs.

8 Conclusion

Unlearnable data are easy to fit, yet the resulting models generalize poorly. Moving perturbations between images shows two reasons: the perturbation alone carries the label, or its effect depends on the image. Treating the first behavior as a construction target yields LCSC, class codes that networks readily decode, that transfer across datasets and architectures, and that take minutes to construct; its factorized variant best resists JPEG compression and small-radius adversarial training among the tested methods. Examining the rule a protected dataset teaches, not only the accuracy it leaves, offers a basis for designing data protection.

AI USE STATEMENT

Generative AI tools assisted language editing, manuscript organization, LaTeX, and the overview figure. The authors checked the experimental code, data, and manuscript for correctness and consistency and take responsibility for the final work.

ETHICS STATEMENT

Unlearnable examples are intended to help data owners deter unauthorized model training, but availability-poisoning techniques are dual use and could also be applied to data without the owner's consent. Our experiments use established public benchmarks and do not involve human subjects or release personal data. Any released implementation should document intended use, perturbation limits, and known countermeasures, and should not be represented as a legal or cryptographic guarantee of data protection.

REPRODUCIBILITY STATEMENT

Section 5 describes the victim protocol, and Appendix C records construction hyperparameters and evaluation conventions. Appendix D reports the capacity sweeps; Tables 4 and 5 report countermeasure results; and Appendix F gives the augmentation matrix. The supplementary records include per-epoch histories and response, confusion, and carrier-conditioned tensors.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Physica-Verlag HD, 2010. doi: 10.1007/978-3-7908-2604-3_16.

Sizhe Chen, Geng Yuan, Xinwen Cheng, Yifan Gong, Minghai Qin, Yanzhi Wang, and Xiaolin Huang. Self-ensemble protection: Training checkpoints are good data protectors. In *International Conference on Learning Representations*, 2023.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with Cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Liam Fowl, Micah Goldblum, Ping-yeh Chiang, Jonas Geiping, Wojciech Czaja, and Tom Goldstein. Adversarial examples make strong poisons. *Advances in Neural Information Processing Systems*, 34:30339–30351, 2021.

Shaopeng Fu, Fengxiang He, Yang Liu, Li Shen, and Dacheng Tao. Robust unlearnable examples: Protecting data against adversarial learning. In *International Conference on Learning Representations*, 2022.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*, 2021.

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

Jinlin Liu, Wei Chen, and Xiaojin Zhang. Perturbation-induced linearization: Constructing unlearnable data with solely linear classifiers. In *International Conference on Learning Representations*, 2026.

Zhuoran Liu, Zhengyu Zhao, and Martha Larson. Image shortcut squeezing: Countering perturbative availability poisons with compression. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22473–22487. PMLR, 2023.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814. Omnipress, 2010.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40): 24652–24663, 2020. doi: 10.1073/pnas.2015509117.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Vinu Sankar Sadasivan, Mahdi Soltanolkotabi, and Soheil Feizi. CUDA: Convolution-based unlearnable datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3862–3871, 2023.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Pedro Sandoval-Segura, Vasu Singla, Jonas Geiping, Micah Goldblum, Tom Goldstein, and David Jacobs. Autoregressive perturbations for data poisoning. *Advances in Neural Information Processing Systems*, 35:27374–27386, 2022.

Pedro Sandoval-Segura, Vasu Singla, Jonas Geiping, Micah Goldblum, and Tom Goldstein. What can we learn from unlearnable datasets? *Advances in Neural Information Processing Systems*, 36: 75372–75391, 2023.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9573–9585, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6cfe0e6127fa25df2a0ef2ae1067d915-Abstract.html.

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

Lue Tao, Lei Feng, Jinfeng Yi, Sheng-Jun Huang, and Songcan Chen. Better safe than sorry: Preventing delusive adversaries with adversarial training. *Advances in Neural Information Processing Systems*, 34:16209–16225, 2021.

Gregory K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34 (4):30–44, 1991.

Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Availability attacks create shortcuts. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2367–2376, 2022.

Chia-Hung Yuan and Shan-Hung Wu. Neural tangent generalization attacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12230–12240. PMLR, 2021.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Yifan Zhu, Lijia Yu, and Xiao-Shan Gao. Detection and defense of unlearnable examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17211–17219, 2024. doi: 10.1609/aaai.v38i15.29667.

A Construction Details

A.1 Shared loss and update schedule

Algorithms 1 and 2 expand the workflow in Figure 2 into the dense and rank-1 procedures. Both use the codebook $\mathcal{C} = \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$, with one perturbation per class. Each step samples a minibatch of clean image–label pairs from $\mathcal{D}_c$ and, for each image, an integer a_i drawn uniformly from $\{0, \dots, K-1\} \setminus \{y_i\}$; the image then receives $\boldsymbol{\Delta}_{a_i}$.

Both variants pass the coded image through the same channel: clipping to $[0, 1]$, 8-bit quantization, and a sampled augmentation T_i. The batch estimate of Eq. 5, denoted by L in Figure 2, is

$$L = \frac{1}{B} \sum_{i=1}^{B} \ell(f(T_i(Q(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}_{a_i})); \boldsymbol{\theta}), a_i), \qquad Q(\boldsymbol{z}) = \frac{\mathrm{round}(255\boldsymbol{z})}{255}. \tag{8}$$

Here ℓ is cross-entropy, B is the minibatch size, and T_i is the identity when augmentation is disabled. The forward pass uses rounded values, and the backward pass treats Q as the identity: for input $\boldsymbol{z}$, we use $\partial L/\partial \boldsymbol{z} := \partial L/\partial Q(\boldsymbol{z})$. This is the identity straight-through estimator (STE) (Bengio et al., 2013). Setting Q to the identity gives the no-quantization ablation.

Reader updates keep the codebook fixed. The pseudocode writes the reader step as plain gradient descent, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} L$; the experiments use SGD with momentum and weight decay, with the settings below. Code updates fix the reader weights and normalization statistics, but gradients still flow through the reader to the codes. All G inner steps reuse the same carrier batch, assigned labels, and augmentations.

For rank-1 codes, each class a and image channel c has two vectors:

$$\boldsymbol{\Delta}_{ac} = \varepsilon\, \boldsymbol{u}_{ac} \boldsymbol{v}_{ac}^{\top}, \qquad \boldsymbol{u}_{ac} \in [-1, 1]^H, \quad \boldsymbol{v}_{ac} \in [-1, 1]^W. \tag{9}$$

Each pixel of channel c is the product of one entry of the length-H vector $\boldsymbol{u}_{ac}$ and one of the length-W vector $\boldsymbol{v}_{ac}$, scaled by ε. We write $\boldsymbol{u}, \boldsymbol{v}$ for these vectors collected over all classes and channels;

Algorithm 2 assembles the codes from them whenever it evaluates L. The factor-balance loss is

$$L_{\mathrm{bal}} = \frac{1}{4KC} \sum_{a=0}^{K-1} \sum_{c=1}^{C} \left(\|\boldsymbol{u}_{ac}\|_2^2 - \|\boldsymbol{v}_{ac}\|_2^2 \right)^2 . \tag{10}$$

It discourages one factor from growing while the other shrinks, a change that would leave their product unchanged. Here K is the class count, C the channel count, and $\| \cdot \|_2$ the Euclidean norm. Both factor gradients are computed before either factor changes, and clipping both factors to $[-1, 1]$ guarantees $\|\boldsymbol{\Delta}_a\|_\infty \le \varepsilon$.

We use $J = 1$ construction cycle for ImageNet-100 and $J = 2$ for SVHN and CIFAR-10/100. Each cycle has $W = 100$ warm-up updates followed by $P = 200$ phases, each consisting of $M = 10$ reader updates and $G = 30$ code updates. In the algorithms W counts warm-up updates; in dimensional expressions it denotes image width. The recorded runs use minibatches of up to 128 images, $\eta_\theta = 0.1$, reader momentum 0.9, reader weight decay 5×10^{-4}, $\lambda_{\mathrm{bal}} = 10^{-3}$, and $\eta_\Delta = 2\varepsilon/G$. Dense codes and rank-1 factors both take projected sign steps of this size. Dense codes are initialized from a Gaussian with standard deviation 0.1ε and clipped once to $[-\varepsilon, \varepsilon]$; factors are initialized from a Gaussian with standard deviation 0.1 and projected once to $[-1, 1]$. The algorithms take these bounded values as input.

Algorithm 1 LCSC construction (dense codes)

Input: clean data $\mathcal{D}_c$; reader $f(\cdot; \boldsymbol{\theta})$; bounded initial codebook $\mathcal{C} = \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$; augmentation distribution $\mathcal{T}$; counts J, W, P, M, G; rates η_θ, η_Δ; ε.
Output: final-cycle codebook $\mathcal{C}^\star$ and protected dataset $\mathcal{D}_u$.

for $j = 1, \dots, J$ **do**
  Initialize fresh reader parameters $\boldsymbol{\theta}$.
  for $w = 1, \dots, W$ **do** // warm up the reader
    Sample a minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
    Sample $a_i \sim \mathrm{Uniform}(\{0, \dots, K-1\} \setminus \{y_i\})$ for each i.
    Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
    Compute the batch loss L using Eq. 8.
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} L$.
  end for
  for $p = 1, \dots, P$ **do**
    for $m = 1, \dots, M$ **do** // update the reader
      Sample a minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
      Sample $a_i \sim \mathrm{Uniform}(\{0, \dots, K-1\} \setminus \{y_i\})$ for each i.
      Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
      Compute the batch loss L using Eq. 8.
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} L$.
    end for
    Sample a new minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
    Sample $a_i \sim \mathrm{Uniform}(\{0, \dots, K-1\} \setminus \{y_i\})$ for each i.
    Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
    for $n = 1, \dots, G$ **do** // keep the batch, a_i, T_i, and reader fixed
      Compute the batch loss L using Eq. 8.
      Compute $\boldsymbol{d}_a \leftarrow \nabla_{\boldsymbol{\Delta}_a} L$ for all classes a.
      $\boldsymbol{\Delta}_a \leftarrow \boldsymbol{\Delta}_a - \eta_\Delta \operatorname{sign}(\boldsymbol{d}_a)$ for all a.
      $\boldsymbol{\Delta}_a \leftarrow \mathrm{Clip}(\boldsymbol{\Delta}_a, -\varepsilon, \varepsilon)$ for all a.
    end for
  end for
  Save $\mathcal{C}^{(j)} \leftarrow \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$ for diagnostics.
end for
$\mathcal{C}^\star \leftarrow \mathcal{C}^{(J)}$. // always use the final cycle
$\mathcal{D}_u \leftarrow \{(Q(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}^\star_{y_i}), y_i)\}_{i=1}^N$.
return $\mathcal{C}^\star, \mathcal{D}_u$.

Algorithm 2 LCSC-LR construction (rank-1 codes)

Input: clean data $\mathcal{D}_c$; reader $f(\cdot;\boldsymbol{\theta})$; bounded initial factors $\boldsymbol{u}, \boldsymbol{v}$ defining $\mathcal{C} = \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$ by Eq. 9; augmentation distribution $\mathcal{T}$; counts J, W, P, M, G; rates η_θ, η_Δ; $\varepsilon, \lambda_{\text{bal}}$.
Output: final-cycle codebook $\mathcal{C}^\star$ and protected dataset $\mathcal{D}_u$.

for $j = 1, \ldots, J$ **do**
  Initialize fresh reader parameters $\boldsymbol{\theta}$.
  for $w = 1, \ldots, W$ **do** // warm up the reader
    Sample a minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
    Sample $a_i \sim \text{Uniform}(\{0, \ldots, K-1\} \setminus \{y_i\})$ for each i.
    Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
    Compute the batch loss L using Eq. 8.
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} L$.
  end for
  for $p = 1, \ldots, P$ **do**
    for $m = 1, \ldots, M$ **do** // update the reader
      Sample a minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
      Sample $a_i \sim \text{Uniform}(\{0, \ldots, K-1\} \setminus \{y_i\})$ for each i.
      Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
      Compute the batch loss L using Eq. 8.
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_\theta \nabla_{\boldsymbol{\theta}} L$.
    end for
    Sample a new minibatch $(\boldsymbol{x}_i, y_i)_{i=1}^B$ from $\mathcal{D}_c$.
    Sample $a_i \sim \text{Uniform}(\{0, \ldots, K-1\} \setminus \{y_i\})$ for each i.
    Sample an augmentation $T_i \sim \mathcal{T}$ for each i.
    for $n = 1, \ldots, G$ **do** // keep the batch, a_i, T_i, and reader fixed
      Compute $L_{\text{total}} = L + \lambda_{\text{bal}} L_{\text{bal}}$ using Eqs. 8–10.
      Compute $(\boldsymbol{d}_u, \boldsymbol{d}_v) \leftarrow \nabla_{(\boldsymbol{u},\boldsymbol{v})} L_{\text{total}}$.
      $\boldsymbol{u} \leftarrow \boldsymbol{u} - \eta_\Delta \,\text{sign}(\boldsymbol{d}_u)$.
      $\boldsymbol{v} \leftarrow \boldsymbol{v} - \eta_\Delta \,\text{sign}(\boldsymbol{d}_v)$.
      $\boldsymbol{u} \leftarrow \text{Clip}(\boldsymbol{u}, -1, 1)$.
      $\boldsymbol{v} \leftarrow \text{Clip}(\boldsymbol{v}, -1, 1)$.
    end for
  end for
  Form $\boldsymbol{\Delta}_{ac} \leftarrow \varepsilon \, \boldsymbol{u}_{ac} \boldsymbol{v}_{ac}^\top$ for all classes and channels.
  Save $\mathcal{C}^{(j)} \leftarrow \{\boldsymbol{\Delta}_a\}_{a=0}^{K-1}$ for diagnostics.
end for
$\mathcal{C}^\star \leftarrow \mathcal{C}^{(J)}$. // always use the final cycle
$\mathcal{D}_u \leftarrow \{(Q(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}^\star_{y_i}), y_i)\}_{i=1}^N$.
return $\mathcal{C}^\star, \mathcal{D}_u$.

Construction stops after J cycles. The dense codebook, or the rank-1 factors, persists across cycles, while the reader and its momentum buffer are reset; the projected-sign code updates keep no optimizer state. We always release the final-cycle codebook, $\mathcal{C}^\star = \mathcal{C}^{(J)}$. For diagnostics only, we measure the construction reader's code accuracy after each cycle on 2,048 clean test images with newly sampled counterfactual labels, 8-bit quantization, and no spatial augmentation. These scores neither select nor change the released codebook. For CIFAR-10 at $8/255$ with Basic augmentation, the final-cycle reader accuracies are 98.8% for the dense codebook and 99.9% for the rank-1 codebook; Table 7 reports the same final-cycle score for each ablation. Release selects each code by the true label y_i and applies quantization without spatial augmentation.

A.2 General factorized and blockwise parameterization

For $B_h \times B_w$ spatial blocks, let $H_b = H/B_h$ and $W_b = W/B_w$. A latent code block is

$$\boldsymbol{\Delta}_{acpq} = \varepsilon \, \boldsymbol{U}_{acpq} \boldsymbol{V}_{acpq}, \quad \boldsymbol{U}_{acpq} \in \mathbb{R}^{H_b \times r}, \quad \boldsymbol{V}_{acpq} \in \mathbb{R}^{r \times W_b}. \tag{11}$$

Here a and c index class and channel, p and q index the block's row and column, and r is the factor rank; each block is the product of two smaller factors. Projecting every row of $\boldsymbol{U}$ and every column

of $\boldsymbol{V}$ onto the unit ℓ_2 ball bounds their inner products by one, so $\|\boldsymbol{\Delta}_a\|_\infty \le \varepsilon$ before clipping and quantization. The factor-balance regularizer is

$$\mathcal{L}_{\mathrm{bal}} = \frac{1}{4KCB_hB_w} \sum_{a,c,p,q} \left\| \boldsymbol{U}_{acpq}^\top \boldsymbol{U}_{acpq} - \boldsymbol{V}_{acpq} \boldsymbol{V}_{acpq}^\top \right\|_F^2 . \tag{12}$$

It penalizes the squared Frobenius distance between the two factor Gram matrices and so discourages unbalanced factor scales. The factorized objective is $\mathcal{L}_{\mathrm{code}} + \lambda_{\mathrm{bal}} \mathcal{L}_{\mathrm{bal}}$, with $\lambda_{\mathrm{bal}} = 10^{-3}$. Main-text LCSC-LR uses one global block and $r = 1$, giving 1,920 CIFAR-10 codebook parameters versus 30,720 for dense codes. Because clipping and quantization are nonlinear, "rank 1" describes the latent factors, not the released pixel changes.

B Construction Ablations

Table 7 changes one construction choice at a time on CIFAR-10/ResNet-18 at $\varepsilon = 8/255$. Victims use Basic augmentation and are evaluated at epoch 60; reader accuracy is the held-out code accuracy after the final construction cycle, measured without augmentation. The dense and rank-1 reference rows are the corresponding main experiments. The default phase has $M = 10$ reader updates followed by $G = 30$ code updates. The update-count ablations set M to 1 or 5 with the same number of phases, so their total update counts differ.

Table 7: Construction ablations on CIFAR-10 at $8/255$ (%). Reader accuracy and CFR ↑; victim clean accuracy ↓.

Variant	Reader acc. (%)	Victim acc. (%)	CFR (%)
LCSC (ours), dense	98.8	10.6	100.0
LCSC-LR (ours), rank 1	99.9	11.0	100.0
Class-aligned pairing ($a = y$)	99.8	10.1	100.0
No reader reset between cycles	100.0	9.9	100.0
No in-loop 8-bit quantization	100.0	13.5	100.0
No augmentation during construction	100.0	10.2	100.0
Reader updates per phase: $M = 1$	11.8	63.8	52.8
Reader updates per phase: $M = 5$	99.7	9.9	100.0
Matched-parameter block code (2×2, $r = 8$)	99.3	11.7	100.0

Most construction choices matter little at this budget. Pairing each code with its own class, or keeping the same reader across cycles, gives 10.1% and 9.9% victim accuracy, versus 10.6% for the dense reference. Removing in-loop quantization raises accuracy to 13.5%, and the matched-parameter block code reaches 11.7%. The number of reader updates matters most. With $M = 1$, the final-cycle reader decodes only 11.8% of codes, victim accuracy rises to 63.8%, and CFR falls to 52.8%; with $M = 5$, victim accuracy returns to 9.9%. The codes therefore need to be optimized against a reader that has learned to decode them. All comparisons are single-seed runs.

C Experimental Protocol and Provenance

Victim optimization. Victims use SGD with learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, cosine learning-rate decay (Loshchilov & Hutter, 2017), and batch size 128. ImageNet-100 runs use smaller batches to fit in memory: 64 for ResNet-18, VGG-19, and MobileNet-V2, 32 for ResNet-50, and 16 for DenseNet-121, for both the clean and protected runs. The main protocol is 60 epochs. Some stored histories run to 100 epochs; all fixed-horizon comparisons read epoch 60. Peak test accuracy is an oracle diagnostic; Appendix L instead selects checkpoints on a separate validation subset and reports accuracy on held-out images.

Shallow learnability probes. The MLP family is flatten–linear$(3072, h)$–ReLU–linear$(h, 10)$ with $h \in \{32, 64, 128, 256, 512, 1024\}$. The convolutional family is one 3×3 convolution with $c \in \{4, 8, 16, 32, 64, 128\}$ channels, followed by flatten–ReLU–linear$(1024c, 10)$, without normalization or pooling. Both families use ReLU activations (Nair & Hinton, 2010). Each probe is trained from scratch on CIFAR-10 for 60 epochs without data augmentation, using SGD with learning rate

0.01, momentum 0.9, weight decay 5×10^{-4}, cosine decay, and batch size 128; a learning rate of 0.1 is unstable for these unnormalized networks. Fitting accuracy is measured on the clean or protected distribution used for training, and generalization on the clean test set. The nominal budget is $\varepsilon = 8/255$, except for the native ℓ_2 SP and AR artifacts described below. Each family is a capacity sweep within one architecture, not a parameter-matched comparison across architectures. Error-Max is omitted because no independently identified artifact is available.

Datasets and architectures. The evaluation covers SVHN (Netzer et al., 2011), CIFAR-10/100 (Krizhevsky, 2009), and ImageNet-100 (Deng et al., 2009; Russakovsky et al., 2015), with ResNet-18/50 (He et al., 2016), VGG-19 (Simonyan & Zisserman, 2015), DenseNet-121 (Huang et al., 2017), and MobileNet-V2 (Sandler et al., 2018) victims. Cross-method countermeasure results use CIFAR-10/ResNet-18 only. Our ImageNet-100 subset contains 130,000 training images from 100 classes.

Construction. Dense LCSC uses a ResNet-18 reader; one construction cycle on ImageNet-100 and two on the other datasets; 100 warm-up steps and 200 alternating phases per cycle; 10 reader updates and 30 code PGD steps per phase; an ℓ_∞ budget ε; 8-bit straight-through quantization; and either None or Basic construction augmentation. The released codebook is always the last cycle of this schedule. LCSC-LR uses the same schedule with the factorized codebook.

Baselines. EM, TAP, SEP, REM, and PIL use locally generated dense perturbations. SP and AR keep their native ℓ_2 dumps without ℓ_∞ clipping (recorded maxima $17/255$ and $46/255$), so they are not matched to the $8/255$ ℓ_∞ budget. For the low-budget comparison, their perturbations are linearly rescaled from the nominal $8/255$ artifacts rather than regenerated at each budget. Class-wise EM uses the original implementation (Huang et al., 2021), with a ResNet-18 surrogate, a 20% training subset, a stopping error of 0.1, and one perturbation step of size $\varepsilon/10$ per update; each budget is constructed separately. Gaussian shares one fixed random perturbation across all images of each class. NTGA (Yuan & Wu, 2021) is omitted because its dump covers only 40,000 training images, and CUDA (Sadasivan et al., 2023) applies a content-dependent convolution that cannot be transplanted as an additive perturbation.

Reporting. All reported results are from our experiments. Unless stated otherwise, each setting is one run with seed 0, and we report clean test accuracy after epoch 60. Transfer diagnostics use separately trained, frozen victims under the same 60-epoch recipe; all response and confusion matrices for a method come from the same checkpoint. Per-epoch metrics and their provenance are included in the supplementary records.

D Shallow-Probe Capacity Sweeps

Table 1 in the main text reports the smallest and largest probe from each family. Tables 8 and 9 give the complete MLP and convolutional sweeps. Each cell pair reports fitting accuracy on the probe's training distribution and accuracy on the clean test set.

Table 8: MLP capacity sweep on CIFAR-10 (%). Higher Train means easier fitting; lower Test means stronger protection.

	$h = 32$		$h = 64$		$h = 128$		$h = 256$		$h = 512$		$h = 1024$	
Method	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Clean	53.2	48.8	58.1	50.7	64.6	53.4	68.8	54.5	73.1	55.7	76.2	56.7
EM (Huang et al., 2021)	99.6	21.3	99.7	23.2	99.8	24.1	99.9	24.5	99.9	24.6	99.9	25.1
REM (Fu et al., 2022)	75.0	39.5	80.3	43.5	84.3	46.2	87.6	46.3	90.2	48.2	92.1	48.7
TAP (Fowl et al., 2021)	55.1	46.6	61.0	49.6	67.5	52.1	72.3	53.2	76.3	54.7	79.4	55.2
SEP (Chen et al., 2023)	61.3	43.5	67.8	47.1	73.1	50.7	76.9	51.3	80.7	52.4	83.7	52.9
PIL (Liu et al., 2026)	100.0	11.9	100.0	12.2	100.0	11.9	100.0	12.3	100.0	12.4	100.0	12.8
SP (Yu et al., 2022)	100.0	19.6	100.0	20.4	100.0	21.0	100.0	21.6	100.0	22.1	100.0	22.2
AR (Sandoval-Segura et al., 2022)	51.6	47.6	58.6	51.3	64.5	53.3	69.0	54.9	73.5	55.6	76.6	56.2
LCSC (ours)	100.0	26.1	100.0	25.8	100.0	25.7	100.0	26.0	100.0	25.7	100.0	25.9
LCSC-LR (ours)	100.0	18.0	100.0	18.8	100.0	18.7	100.0	19.0	100.0	19.1	100.0	19.2

In the MLP sweep, EM, PIL, SP, and both LCSC variants are fitted almost perfectly even at the smallest width while generalizing poorly. REM, TAP, and SEP show a smaller version of the same

gap at every width. AR is the exception: its fitting and test accuracy track Clean throughout the sweep.

Table 9: Convolutional capacity sweep on CIFAR-10 (%). Higher Train means easier fitting; lower Test means stronger protection.

	$c = 4$		$c = 8$		$c = 16$		$c = 32$		$c = 64$		$c = 128$	
Method	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Clean	63.3	51.9	73.7	54.6	84.2	59.3	91.8	62.0	94.1	62.5	96.2	63.9
EM (Huang et al., 2021)	99.9	20.2	100.0	22.7	100.0	23.6	100.0	25.6	100.0	25.7	100.0	25.9
REM (Fu et al., 2022)	98.9	28.9	98.8	36.7	99.8	44.8	100.0	47.6	100.0	44.9	100.0	48.9
TAP (Fowl et al., 2021)	72.2	45.3	83.4	48.4	95.3	50.6	98.2	53.8	99.6	52.9	99.9	53.8
SEP (Chen et al., 2023)	80.4	36.8	92.2	34.7	96.3	51.8	99.0	53.4	99.6	53.3	99.8	54.3
PIL (Liu et al., 2026)	100.0	11.7	100.0	12.3	100.0	14.3	100.0	13.4	100.0	13.6	100.0	14.1
SP (Yu et al., 2022)	100.0	18.5	100.0	20.2	100.0	21.1	100.0	22.1	100.0	22.4	100.0	22.4
AR (Sandoval-Segura et al., 2022)	89.5	35.0	93.2	40.4	98.0	42.7	99.6	52.4	99.8	43.3	99.9	41.6
LCSC (ours)	100.0	21.8	100.0	19.9	100.0	22.0	100.0	22.1	100.0	23.3	100.0	23.8
LCSC-LR (ours)	100.0	14.5	100.0	14.2	100.0	15.0	100.0	15.2	100.0	15.9	100.0	16.3

With a single convolution, every protected dataset has higher fitting accuracy and lower clean test accuracy than Clean at every width, AR included: four channels give AR 89.5% fitting and 35.0% clean accuracy, versus 63.3% and 51.9% for Clean. The convolutional head has more parameters than the MLP at nominally similar widths, so each sweep describes ease of fitting within its own architecture family.

E Direct Mechanism Controls

This appendix first tests code following directly on a dense LCSC victim, then examines how predictions under perturbation transfer depend on the image class for all methods.

Table 10: Dense LCSC controls on CIFAR-10 at $1/255$. Code-target accuracy ↑ indicates decoding; clean accuracy ↓ indicates protection.

Input to the trained victim	Evaluation target	Accuracy (%)
$\boldsymbol{x} + \boldsymbol{\Delta}_y$	semantic label y	100.0
$\boldsymbol{x} + \boldsymbol{\Delta}_a$, $a \neq y$	assigned label a (CFR)	100.0
neutral carrier $+\boldsymbol{\Delta}_a$	assigned label a	100.0
clean image $\boldsymbol{x}$	semantic label y	9.8
protected image with code removed	semantic label y	9.2
matched random class codebook	semantic label y	18.4

Table 10 evaluates a ResNet-18 trained for 60 epochs without augmentation; each row reports accuracy against its own target. With the correct code, a wrong code, or a neutral carrier, the victim predicts the attached label on 100% of inputs, so the code alone can determine the prediction. The same victim reaches only 9.8% on clean images and 9.2% after the code is removed: strong code recognition coexists with almost no clean classification. A victim trained with a matched random class codebook instead reaches 18.4%, so arbitrary shared noise does not reproduce the full effect of the learned codes.

Dependence on the image class. Figure 1 averages the transfer response over image classes. Figure 3 separates them, to show whether this average hides different responses to different carriers. Each panel column fixes the perturbation label a, and each row selects an image class y, so reading down a column changes only the image class. Each cell shows the most frequent predicted label b: a constant column means a stable dominant prediction, whereas changes down a column reveal dependence on the image class.

For each perturbation class a, we sample five training perturbations and apply each to all 10,000 held-out CIFAR-10 images, reusing the same five draws for every carrier class. Each (y, a) pair therefore contains 5,000 predictions, including the within-class pairs $y = a$. All carriers come from $\mathcal{D}_t$. The frozen victim and the clipping and 8-bit round trip are those of the main audit, and every transfer input is quantized after clipping. The carrier-conditioned tensor and its cross-class marginal

are

$$T_{yab} = \Pr[\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{x} \oplus \boldsymbol{\delta}) = b \mid y,\ \boldsymbol{\delta} \sim \mathcal{Q}(\cdot \mid a)],$$
$$R^{(T)}_{ab} = \frac{1}{K-1} \sum_{y \neq a} T_{yab}, \qquad S_{yb} = T_{y,y,b}. \tag{13}$$

The saved cross-class matrix equals this marginal, and the within-class matrix is the diagonal slice. Because a dominant-label plot can hide smaller changes in the distribution, we also report carrier sensitivity (CS) for each method: the average distance between each cross-class response distribution and the corresponding average over image classes. Lower CS means less dependence on the image class:

$$\mathrm{CS} = \frac{1}{K(K-1)} \sum_{a \in \mathcal{Y}} \sum_{y \neq a} \frac{1}{2} \left\| T_{ya:} - R^{(T)}_{a:} \right\|_1. \tag{14}$$

The colon in $T_{ya:}$ collects probabilities over all predicted labels b. Half the ℓ_1 distance is the total variation distance, which is zero when the two prediction distributions agree. CS averages it over all pairs of unequal image and code classes. Figure 4 shows the within-class matrices; high within-class accuracy shows transfer within a class, whereas the cross-class test determines whether predictions follow the code across classes.

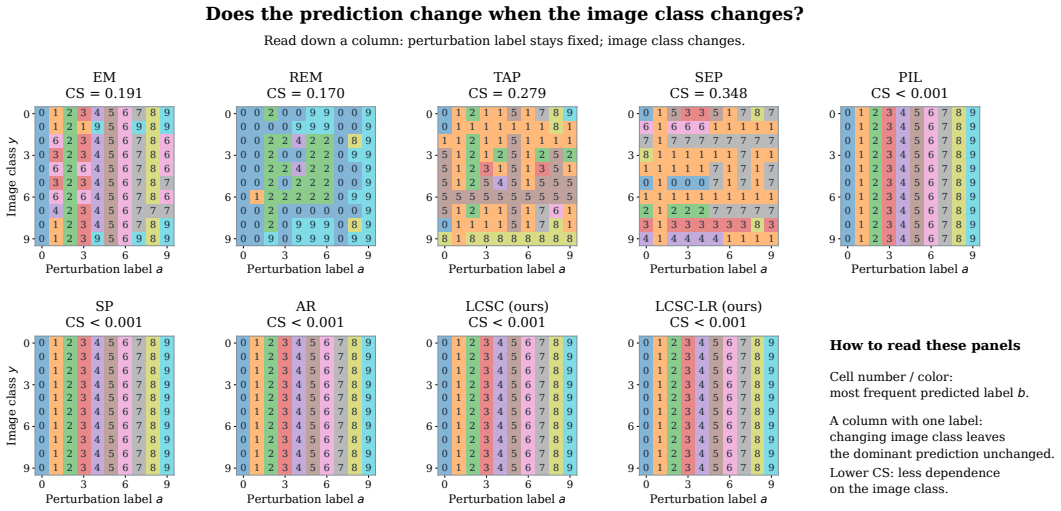

Figure 3: **Predicted labels under perturbation transfer across image classes.** Rows indicate image class y, and columns indicate perturbation label a. Cell numbers and colors show the most frequent predicted label, $\arg\max_b T_{yab}$. CS measures carrier sensitivity from the cross-class response distributions; lower values indicate less carrier dependence.

In Figure 3, PIL, SP, AR, and both LCSC variants have constant columns that follow the perturbation label, and their CS values are below .001: changing the image class leaves nearly the entire cross-class response distribution unchanged. EM, REM, TAP, and SEP have larger CS values (.170–.348), and many of their columns change dominant label across rows. These are output-level measurements. They show how predictions depend on the image class, not which internal features produce that dependence.

F Additional Countermeasure Results

The complete JPEG and PGD-7 adversarial-training sweeps are reported in Tables 4 and 5 of the main text; JPEG is applied during victim training in addition to Basic augmentation. Table 11 adds other training augmentations. Except for None, each column applies one transformation on top of Basic random crop and horizontal flip: rotation, perspective, channel shuffling, Cutout (DeVries & Taylor, 2017), CutMix (Yun et al., 2019), or MixUp (Zhang et al., 2018).

Table 11: Protection under training augmentations on CIFAR-10. ResNet-18 clean test accuracy (%); lower is better.

Method	None	Basic	Rotation	Persp.	ChShuf	Cutout	CutMix	MixUp
Clean	88.8	94.5	94.2	95.0	93.5	95.1	95.4	95.0
EM (Huang et al., 2021)	10.1	9.3	10.0	10.9	30.5	10.8	11.2	10.3
REM (Fu et al., 2022)	14.7	19.0	20.6	18.4	57.2	18.8	21.3	20.4
TAP (Fowl et al., 2021)	9.6	6.6	12.2	13.6	7.3	10.8	10.1	21.1
SP (Yu et al., 2022)	15.1	36.6	50.8	41.1	90.9	33.9	34.9	40.9
SEP (Chen et al., 2023)	**2.2**	**2.1**	**5.4**	**8.1**	**2.4**	**2.6**	**4.8**	**7.8**
AR (Sandoval-Segura et al., 2022)	21.2	25.5	35.0	41.0	33.6	24.6	20.2	21.8
PIL (Liu et al., 2026)	10.2	10.0	14.4	13.1	14.6	10.0	10.2	9.3
$\text{EM}^{(C)}$ (Huang et al., 2021)	10.4	10.1	11.0	11.8	12.9	10.3	10.1	12.1
Gaussian	10.4	94.6	94.2	94.9	93.1	93.8	21.2	94.8
LCSC (ours)	9.6	10.6	13.6	12.0	11.6	10.1	10.0	12.2
LCSC-LR (ours)	9.9	11.0	10.4	11.3	10.0	10.7	13.4	10.7

Both codebook variants keep clean accuracy between 9.6% and 13.6% in every setting of Table 11. Several baselines depend more on the transformation: channel shuffling raises REM to 57.2% and SP to 90.9%, and the unoptimized Gaussian codes protect only without augmentation (10.4%) and partly under CutMix (21.2%). SEP gives the lowest accuracy in every column of this table.

G Low-Budget Dynamics

G.1 Architecture transfer

Table 12 reports peak and epoch-60 clean accuracy for five victim architectures; neither construction nor victim training uses augmentation. Every architecture except VGG-19 at $1/255$ ends near chance, but the trajectories differ. DenseNet-121 peaks at 74.9% and 63.6% at $2/255$ and $4/255$ before ending near 10%, so the small peak–final gaps of ResNet-18 at these budgets do not hold for every architecture.

Table 12: Low-budget architecture transfer of dense LCSC on CIFAR-10. Peak and epoch-60 clean accuracy (%); lower is better.

	ResNet-18		ResNet-50		VGG-19		DenseNet-121		MobileNet-V2	
ε	Peak	Final	Peak	Final	Peak	Final	Peak	Final	Peak	Final
$1/255$	66.0	9.8	64.9	10.4	89.6	89.6	76.8	10.4	67.6	10.5
$2/255$	11.0	10.0	42.4	10.3	48.4	10.2	74.9	10.0	10.6	10.0
$4/255$	12.4	10.0	14.3	10.2	60.5	10.1	63.6	10.1	10.4	9.7
$8/255$	11.2	9.6	11.8	9.8	24.7	11.1	13.5	10.6	11.1	10.4

VGG-19 at $1/255$ is the one failure: it ends at 89.6% with 1.3% CFR, so the code does not transfer to this architecture at the smallest budget. All other cells in the table have approximately 100% CFR at epoch 60.

G.2 Cross-method comparison

Table 6 in the main text places the epoch-60 endpoint beside the highest clean accuracy observed during training. The SP and AR columns rescale their native ℓ_2 perturbations by $1/8$, $1/4$, $1/2$, and 1 rather than enforce an ℓ_∞ budget. At $1/255$, PIL, LCSC, and LCSC-LR end near chance after peaks between 56.5% and 75.7%, whereas EM stays near 25% throughout; similar endpoints can therefore hide different early-training vulnerability. EM's accuracy-based stopping rule lengthens its optimization at small budgets, while LCSC uses a fixed schedule that takes about five minutes for dense codes at every budget. Whether a longer LCSC schedule would reduce the early peak at $1/255$ is untested.

G.3 CONSTRUCTION-READER DECODABILITY

Table 13 reports the held-out code accuracy of the construction reader after the second and final cycle, using exact 8-bit forward quantization, next to the 60-epoch victim trained without augmentation. The co-optimized readers decode the codes at all four budgets, and the victims fall to 20% clean accuracy or below within the first epoch from $2/255$ upward and by epoch 6 at $1/255$. Because these readers were optimized together with their codes, Appendix K separately tests whether an independent reader can learn the frozen codes.

Table 13: CIFAR-10 decoding and protection without augmentation. Reader accuracy ↑; victim accuracy and epochs to 20% ↓.

ε	Held-out reader code acc. (%)	Victim final acc. (%)	First epoch $\leq 20\%$
$1/255$	100.0	9.8	6
$2/255$	100.0	10.0	1
$4/255$	100.0	10.0	1
$8/255$	100.0	9.6	1

H EFFICIENCY AND FOOTPRINT

Table 14 reports construction wall-clock time on one GPU, including surrogate training and perturbation construction. These are recorded times for our hardware and implementations, not hardware-normalized complexity.

Table 14: One-GPU construction time on CIFAR-10; lower is better. Dashes denote unavailable measurements.

Method	$1/255$	$2/255$	$4/255$	$8/255$
EM (Huang et al., 2021)	2.7 h	2.6 h	3.3 h	43 min
TAP (Fowl et al., 2021)	23.4 h	–	–	23.0 h
SEP (Chen et al., 2023)	16.4 h	–	–	16.4 h
REM (Fu et al., 2022)	37.8 h	–	–	37.1 h
PIL (Liu et al., 2026)	1.7 min	–	–	1.7 min
LCSC (ours) (None)	5.0 min	4.9 min	5.0 min	5.0 min
LCSC-LR (ours) (None)	33.2 min	34.3 min	38.5 min	27.8 min

EM stops when its surrogate exceeds 99% accuracy on the perturbed training set, so its optimization length varies with the budget: 2.6–3.3 hours at $1/255$–$4/255$, versus 43 minutes at $8/255$. Dense LCSC runs a fixed two-cycle schedule and takes about five minutes at every budget. With Basic augmentation at $8/255$, dense LCSC and LCSC-LR take 5.1 and 20.3 minutes, respectively.

Table 15: Float32 perturbation storage: class codes versus per-image codes. Storage ↓; reduction factor ↑. MB/GB are decimal.

Dataset	N	K	Class-code scalars	Sample-wise scalars	Reduction
CIFAR-10	50,000	10	30,720 (0.12 MB)	153.6M (614.4 MB)	$5000\times$
ImageNet-100	130,000	100	15.05M (60.2 MB)	19.57B (78.3 GB)	$1300\times$

Table 15 counts the float32 values needed to store the perturbations: K class codes versus N per-image perturbations. The saving concerns the perturbation state used to generate or apply protection; the released dataset still stores one image per sample.

I PRETRAINED VICTIMS

We also fine-tune a ResNet-18 initialized by supervised ImageNet pretraining and a ResNet-50 initialized by MoCo-v2 (Chen et al., 2020). Both train for 60 epochs on CIFAR-10 images upsampled to 224×224, using SGD with learning rate 0.01, momentum 0.9, weight decay 5×10^{-4}, cosine decay, batch size 64, and Basic augmentation. This setting tests whether code following persists when training starts from pretrained visual features.

Table 16: Pretrained-victim fine-tuning on protected CIFAR-10 (%). Clean accuracy ↓; code-following rate (CFR) ↑.

Initialization / backbone	Method	Acc. (%)	CFR (%)
ImageNet / ResNet-18	PIL (Liu et al., 2026)	12.4	99.9
ImageNet / ResNet-18	LCSC (ours)	11.6	100.0
ImageNet / ResNet-18	LCSC-LR (ours)	13.1	99.9
MoCo-v2 / ResNet-50	LCSC (ours)	10.6	100.0

All four fine-tuned victims follow transplanted codes on at least 99.9% of cross-class transfers and reach 10.6–13.1% clean accuracy (Table 16). Code following therefore persists when training starts from supervised or contrastive ImageNet features, at least under this fine-tuning recipe. The higher resolution and lower learning rate limit direct comparison with training from scratch.

J Additional Notes on Interpretation

Decodability depends on the reader. Fresh-reader accuracy measures how readily a specified reader class learns the codes under a given optimization horizon and transformation distribution. The same latent code can be easy to decode under one transformation and unreadable under another. This diagnostic complements, but does not replace, the victim-based evaluation of DPS.

Counterfactual target sampling. Excluding y when sampling a makes the code label depend statistically on the carrier label, because one code class is ruled out. With K classes, a predictor that knows the image label can therefore reach $1/(K-1)$ accuracy without reading the perturbation, compared with the uniform-guess baseline of $1/K$. Class-aligned pairing also protects strongly in the $8/255$ ablation (Table 7).

K Code Classes, Distributions, and Fresh Readers

Table 17 collects the notation. Images carry their original labels y, and perturbations carry assigned labels a from the same class set. The joint distribution $\mathcal{Q}$ samples $(\boldsymbol{\delta}, a)$ with a uniform label marginal, and $\mathcal{Q}(\cdot \mid a = y)$ supplies a perturbation matching the image label. The empirical audit groups perturbations by their source training label, whereas in LCSC $\mathcal{Q}_{\mathcal{C}}$ samples one of the K fixed labeled codes. Moving a code to a new carrier preserves the latent code, although clipping and quantization can make the effective pixel change depend on the image.

Table 17: Notation used throughout the formulation.

Symbol	Meaning
$\mathcal{D}_c, \mathcal{D}_t, \mathcal{D}_u$	Clean training, clean test, and protected training sets
$\boldsymbol{x}, y$	Clean image and its original label in $\mathcal{Y}$
a, b	Perturbation class in $\mathcal{Y}$; predicted semantic label in $\mathcal{Y}$
$\mathcal{Q}_{\boldsymbol{\phi}}$	Joint distribution over perturbation–label pairs $(\boldsymbol{\delta}, a)$
$\mathcal{Q}_{\boldsymbol{\phi}}(\cdot \mid a)$	Distribution of bounded perturbations of code class a
$\boldsymbol{\Delta}_a, \mathcal{C}$	One deterministic codeword; the collection of codewords
Q	8-bit quantizer: $Q(\boldsymbol{z}) = \mathrm{round}(255\boldsymbol{z})/255$
$T \sim \mathcal{T}$	Storage or image transformation, possibly the identity
$f_{\boldsymbol{\theta}}$	Model; context identifies the construction reader or downstream victim
R_{ab}, C_{yb}, S_{yb}	Cross-class, clean, and within-class prediction matrices
$\sigma \in \mathfrak{S}_K$	Candidate global class permutation in optimal relabeling

For a fixed joint perturbation distribution $\mathcal{Q}$, draw $(\boldsymbol{x}, y) \sim \mathcal{D}_c$, $(\boldsymbol{\delta}, a) \sim \mathcal{Q}(\cdot \mid a \neq y)$, and $T \sim \mathcal{T}$. A reader is trained with the transformed decoding loss

$$\mathcal{L}_{\mathrm{read}}(\boldsymbol{\theta}; \mathcal{Q}) = \mathbb{E}_{\substack{(\boldsymbol{x},y)\sim\mathcal{D}_c,\ T\sim\mathcal{T} \\ (\boldsymbol{\delta},a)\sim\mathcal{Q}(\cdot|a\neq y)}} \ell(f_{\boldsymbol{\theta}}(T(\boldsymbol{x} \oplus \boldsymbol{\delta})), a)\,. \tag{15}$$

Starting from initial parameters $\boldsymbol{\theta}_0$ drawn by the stated initialization rule, we apply s SGD updates with $\mathcal{Q}$ held fixed to obtain $\boldsymbol{\theta}_s$. For $(\boldsymbol{x}, y) \sim \mathcal{D}_t$, codes and transformations are sampled with the

same counterfactual rule. Averaging over these test draws, the initialization, and training randomness, we define

$$\mathrm{Read}_s(\mathcal{Q}) = \Pr\left[\arg\max_{c\in\mathcal{Y}} f_{\boldsymbol{\theta}_s}(T(\boldsymbol{x}\oplus\boldsymbol{\delta}))_c = a\right]. \tag{16}$$

Read_s is the probability that the reader's top prediction after s updates is the attached code's label a. We report the architecture, initialization, optimizer, horizon, and channel of each training trajectory. Construction-reader scores, in contrast, describe a reader optimized jointly with its codebook.

Our mapping accuracy optimizes a global bijection, found by an exact assignment solver. Allowing any deterministic correction m from predicted labels to semantic labels instead gives

$$\max_{m:\mathcal{Y}\to\mathcal{Y}} \Pr[m(\widehat{y}_{\boldsymbol{\theta}}(\boldsymbol{x})) = y] = \frac{1}{K}\sum_b \max_y C_{yb} \tag{17}$$

for the balanced CIFAR-10 test set. Both definitions give 57.8% for TAP and 68.5% for SEP, and REM changes only from 19.2% to 19.6%, so the interpretation does not depend on requiring a bijection. Maximizing each row of C independently would instead measure how concentrated predictions are for each true label; it can score a collapsed model highly and is not a recoverable classification accuracy.

Figure 4 shows the within-class transfer matrices. Comparing them with the cross-class matrices in Appendix P separates transfer within a class from code following across classes.

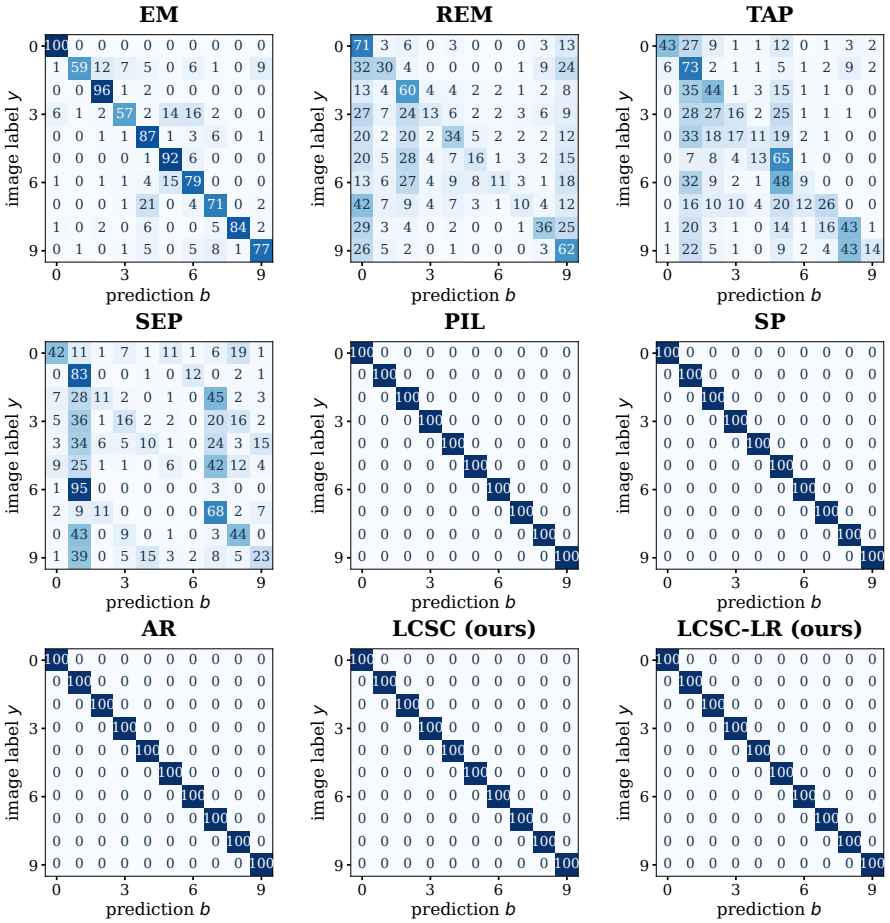

Figure 4: Within-class perturbation transfer S. Rows are semantic labels of held-out images, and columns are victim predictions. Five training perturbations per class are each applied to 1,000 held-out images of the same class.

Independent frozen-code experiment. This experiment asks whether a new reader can learn the completed codes when the codebook is frozen. It separates reuse of the learned signal from co-adaptation between a codebook and its construction reader. Table 18 reports decoding accuracy, and Figure 5 shows the learning trajectories.

We load the final dense codebooks for CIFAR-10 at $8/255$ with Basic augmentation and at $1/255$ without augmentation; the reloaded parameters reproduce the saved latent code tensors exactly. A fresh ResNet-18 with initialization seed 20260908 is trained with SGD (learning rate 0.1, momentum 0.9, weight decay 5×10^{-4}, batch size 128, no learning-rate decay) for six epochs, or 2,346 updates. Only the reader is updated, with newly sampled counterfactual code classes in every batch. This horizon is comparable to the 2,100 reader updates in one construction cycle.

Evaluation uses 2,048 held-out images, true 8-bit forward quantization, and newly sampled code classes and transformations. The evaluation channel matches reader training: Basic at $8/255$ and None at $1/255$. We record accuracy every 50 updates and at epoch boundaries.

Table 18: Fresh-reader decoding with frozen codes. Code accuracy ↑; updates to 90% accuracy ↓.

Budget / channel	One epoch (%)	Six epochs (%)	Updates to 90%
$8/255$, Basic	8.9	99.8	1,100
$1/255$, None	10.8	100.0	1,300

Both fresh readers start near chance and then learn the frozen codes, first exceeding 90% accuracy after 1,100 updates at $8/255$ and 1,300 updates at $1/255$, and reaching 99.8% and 100.0% after six epochs. The completed codes are therefore reusable by a new reader in both settings, although decoding is not immediate. Because the two settings also differ in augmentation, the difference in learning speed cannot be attributed to the budget alone.

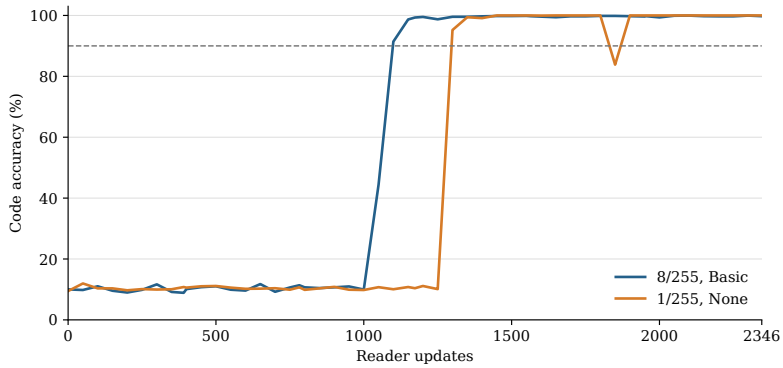

Figure 5: Fresh-reader code accuracy during training with a frozen codebook. The curves correspond to $8/255$ with Basic augmentation and $1/255$ without augmentation, both with 8-bit evaluation. The horizontal dashed line marks 90% accuracy.

L Validation-Based Early Stopping

The epoch-60 result measures protection under one training horizon. A learner with labeled clean validation data can instead choose an earlier checkpoint. We evaluate this additional-access setting with a fixed seed-0 random split of the 10,000 CIFAR-10 test images into 5,000 validation and 5,000 held-out images. The validation half selects the earliest epoch with the highest validation accuracy, and we report accuracy on the other half at that epoch. Neither half is used for victim training or for selecting the released codebook, which is always $\mathcal{C}^{(J)}$. The test peak over all 10,000 images is reported separately as an oracle diagnostic.

At $1/255$, validation selection recovers 65.3% held-out accuracy for dense LCSC and 55.1% for LCSC-LR, compared with 10.3% and 10.1% at epoch 60 on the same images. Near-chance final accuracy thus coexists with useful intermediate models. At $2/255$, selection recovers much less:

Table 19: Validation-selected versus final CIFAR-10/ResNet-18 accuracy (%) without augmentation. Lower means stronger protection.

Method	Budget	Selected epoch	Selected acc.	Epoch-60 acc.
LCSC	$1/255$	2	65.3	10.3
LCSC	$2/255$	29	10.8	9.7
LCSC-LR	$1/255$	1	55.1	10.1
LCSC-LR	$2/255$	6	13.9	11.4

10.8% and 13.9%. A learner with clean validation data can therefore benefit substantially from checkpoint selection at the smallest budget.

M High-Frequency Code Estimation and Removal

JPEG compression and adversarial training are general UE countermeasures. Here we test an attack aimed specifically at LCSC's class-shared codes: estimate the repeated perturbation from the released images and labels, subtract it, and train a fresh classifier.

Every image of a class carries the same latent code. Ignoring clipping and quantization, the mean of the released images of a class therefore equals the clean class mean plus that code. Averaging more images reduces sampling noise but does not remove the clean class mean, so the code cannot be read off the average directly. If the clean class mean is smoother than the code, however, the high-frequency residual of the average retains part of the code. Let $\boldsymbol{x}_i^{\mathrm{p}} = Q(\boldsymbol{x}_i \oplus \boldsymbol{\Delta}_{y_i}^{\star})$ be a released image and $\bar{\boldsymbol{x}}_y^{\mathrm{p}}$ the mean of released training images with label y. We estimate and remove the code using

$$\widehat{\boldsymbol{\Delta}}_y = \mathrm{clip}_{[-\varepsilon,\varepsilon]}\big(\bar{\boldsymbol{x}}_y^{\mathrm{p}} - S(\bar{\boldsymbol{x}}_y^{\mathrm{p}})\big), \qquad \widetilde{\boldsymbol{x}}_i = \mathrm{clip}_{[0,1]}(\boldsymbol{x}_i^{\mathrm{p}} - \widehat{\boldsymbol{\Delta}}_{y_i}). \tag{18}$$

Here S is a 7×7 box filter with reflection padding. The estimate keeps the high-frequency part of the class mean, is clipped to the perturbation budget, and is subtracted from every image of that class. It uses only released training images and labels. It can retain fine-scale structure of the clean class mean and misses any low-frequency part of the code.

We use the final-cycle CIFAR-10 codebooks at $8/255$ with Basic construction augmentation and all 5,000 released training images per class. The true codes are used only to score the estimates: after removing the component common to all classes from both tensors, we compute the mean per-class cosine similarity and the ratio of squared error to true-code energy. Cosine similarity measures whether the estimate points in the direction of the code, not how much of its effect on learning is removed. Fresh ResNet-18 victims then train for 60 epochs with Basic augmentation. Preprocessed pixels are clipped to $[0, 1]$ without further quantization, and test images are unmodified.

Table 20: Code estimation and removal on CIFAR-10 at $8/255$. Better estimation: cosine ↑, error ↓; stronger protection: accuracy ↓.

			Clean test accuracy (%)	
Codebook	Cosine similarity	Relative error	No removal	High-frequency removal
LCSC	0.864	0.259	10.6	12.6
LCSC-LR	0.645	0.590	11.0	12.8

The estimates align with the true codes, with cosine similarities of 0.864 for dense LCSC and 0.645 for LCSC-LR (Table 20), yet removing them barely weakens protection: clean accuracy after retraining rises only from 10.6% to 12.6% and from 11.0% to 12.8%. Applied to clean training data, the same preprocessing leaves accuracy at 94.4%, versus 94.5% without it, so it does not simply damage the images. Both codebooks therefore retain strong protection against this estimator; other label-aware estimators remain untested.

N Interpreting Coding, Coupling, and Learning Dynamics

Coding and coupling as outcomes of construction. A perturbation can supply a reusable label cue or change how a particular image is classified. Optimizing perturbations to be decoded on changing carriers favors the first; optimizing them against a surrogate on their own images permits the second, a joint image–perturbation rule. These are plausible routes to coding and coupling rather than per-method causal attributions; the transfer experiments test which rule a trained victim actually follows.

Why a code can displace semantic features. Attaching codes by training label creates a cue that repeats exactly across all images of a class. Recognizing a few such patterns can reduce the training loss without learning the variation within each semantic class, consistent with shortcut learning (Geirhos et al., 2020) and simplicity bias (Shah et al., 2020). The shallow probes and fresh readers show that the cue is available and easy to learn; that ordinary victims prefer it rests on their transfer behavior and clean accuracy, and at $1/255$ this preference emerges only after early training.

Why clean accuracy can rise before it falls. At small budgets, the code signal may be weak relative to image variation early in training. A plausible account is that the model first learns useful image features and then increasingly relies on the repeated code, which would produce an early clean-accuracy peak followed by a low endpoint. The fresh readers likewise start near chance before learning the frozen codes. Confirming this account would require tracking code following and image representations at intermediate victim checkpoints.

Relation to neural collapse. Neural collapse describes the geometry of final-layer features: within-class features converge to their class means, which together with the classifier approach a symmetric configuration (Papyan et al., 2020). Such geometry could arise whether classes are separated by semantic features or by codes. Our measurements concern which input signal predictions depend on, not feature geometry, so they neither establish nor exclude a connection.

Why factorized codes may survive JPEG better. JPEG quantizes blockwise frequency coefficients, so protection depends on which code patterns remain distinguishable after compression. Low rank alone does not imply low frequency: the outer product of two alternating-sign vectors has rank one and is highly oscillatory. The learned factorized codes may keep more of their discriminative structure after JPEG, but the two variants also differ in optimization; testing this explanation would require measuring code separation and decoding after compression under otherwise matched construction.

O Visual Examples of Protected Data

Figures 6–14 show one training image per CIFAR-10 class, sampled once with seed 20260909 and reused across methods. Following the layout of PIL (Liu et al., 2026), each figure has three rows: clean images, normalized effective perturbations, and protected images. We apply the same clipping and 8-bit round trip as in the transfer audit and display the effective change $\boldsymbol{\delta}_{\text{eff}} = \boldsymbol{x}^{\text{p}} - \boldsymbol{x}$. Only the middle row is rescaled, by its per-image RGB minimum and maximum, so its colors show spatial patterns rather than comparable magnitudes. SP and AR retain their native ℓ_2 artifacts.

Figures 6, 7, and 8 show EM, REM, and TAP on the same ten images.

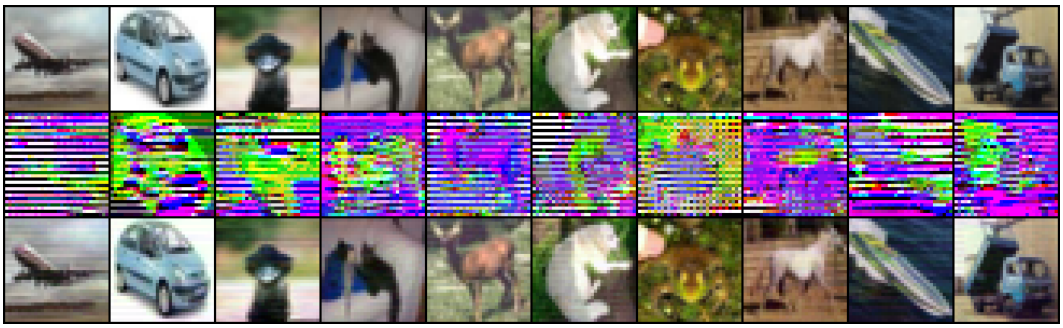

Figure 6: **EM samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

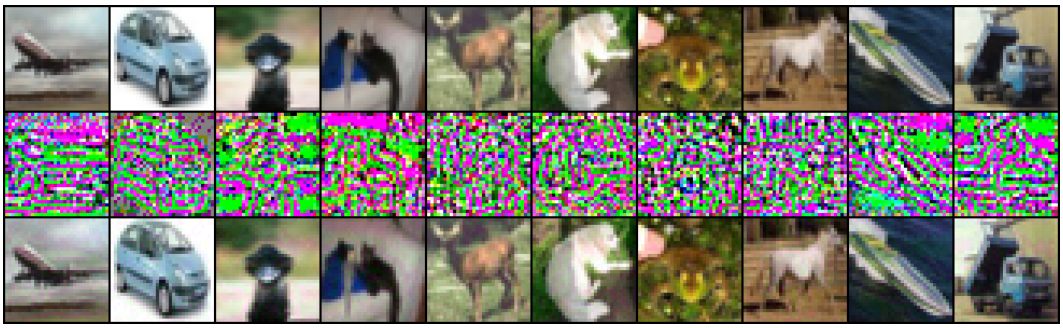

Figure 7: **REM samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

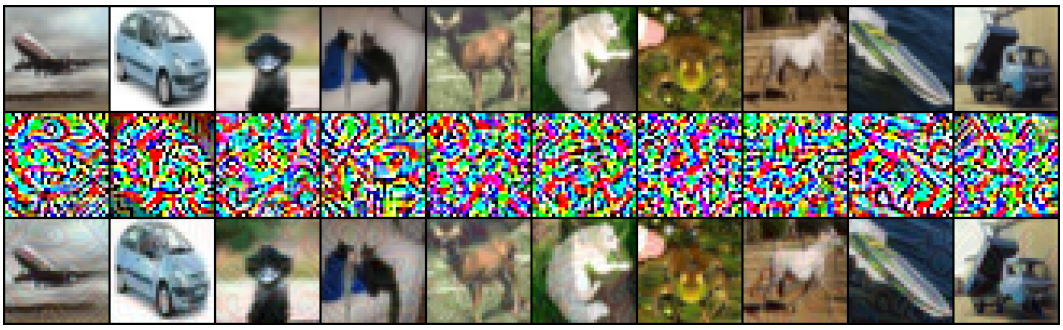

Figure 8: **TAP samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

Figures 9, 10, and 11 show SEP, PIL, and SP on the same images.

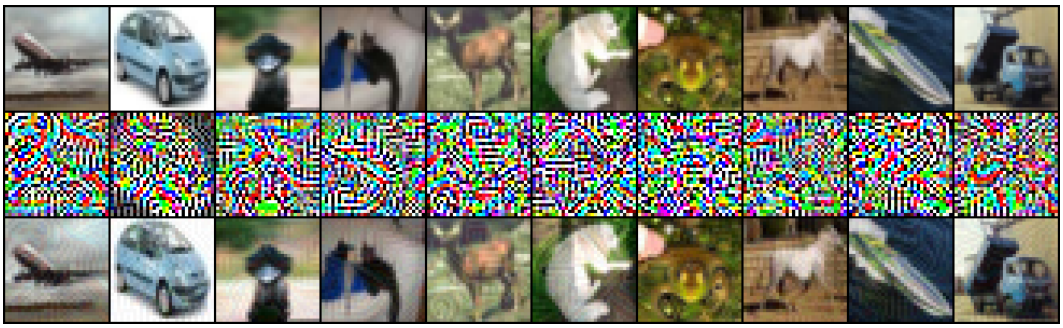

Figure 9: **SEP samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

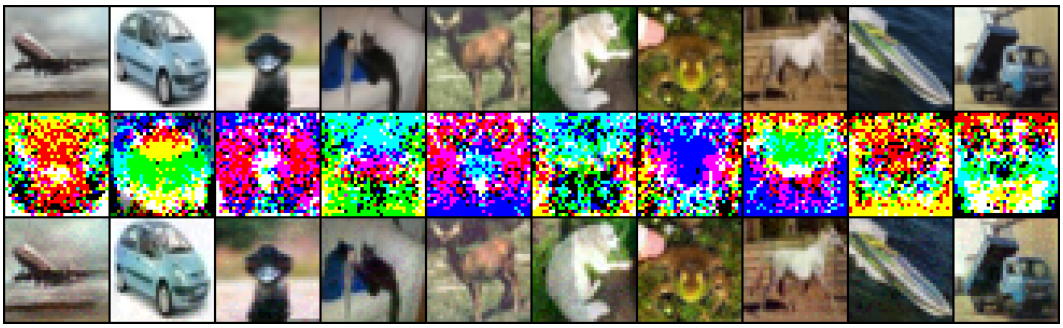

Figure 10: **PIL samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

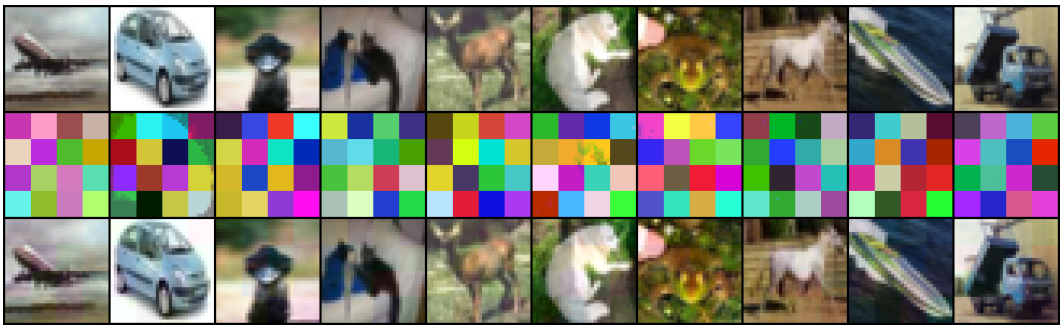

Figure 11: **SP samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images. Native ℓ_2 perturbations.

Figures 12, 13, and 14 show AR and the two learned codebooks. AR retains its native ℓ_2 artifact, whereas both codebooks use the stated ℓ_∞ budget.

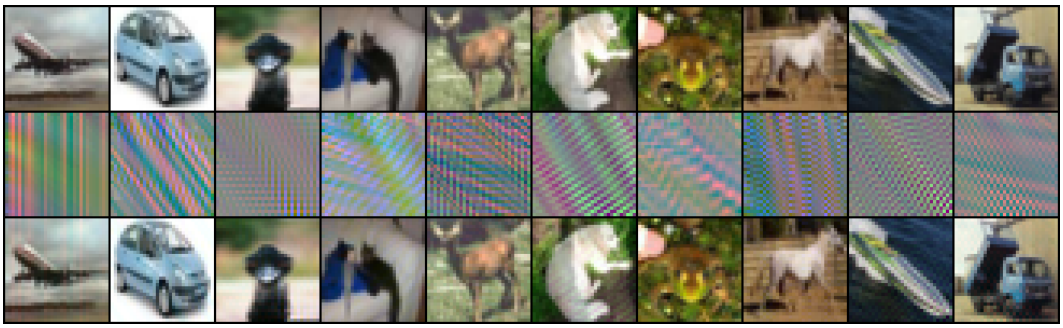

Figure 12: **AR samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images. Native ℓ_2 perturbations.

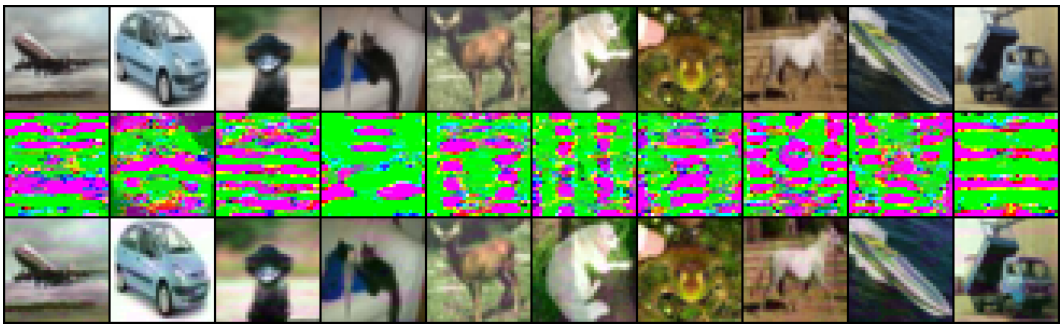

Figure 13: **LCSC (ours) samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.



Figure 14: **LCSC-LR (ours) samples on CIFAR-10.** Top to bottom: clean images, normalized effective perturbations, and protected images.

Figure 15 applies the same three-row layout to ImageNet-100, with two carriers for each of three classes. The latent code is shared within a class, but clipping and quantization can make its effective change differ between images.

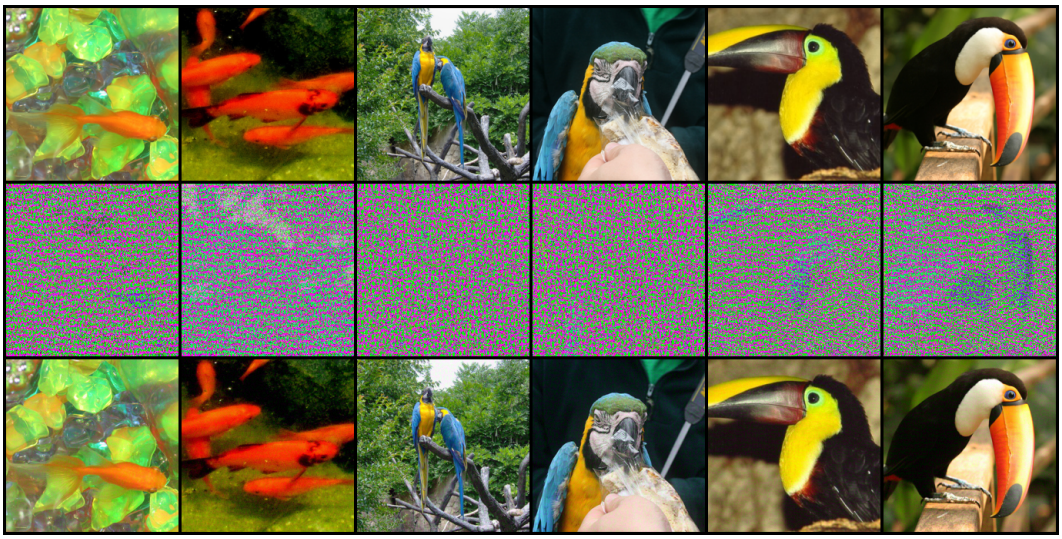

Figure 15: **LCSC (ours) samples on ImageNet-100 at** $8/255$**.** Two training images from each of three classes, using the construction/evaluation preprocessing (resize to 256, center crop to 224). Rows show clean images, independently normalized effective perturbations, and protected images. The latent code is shared within a class; clipping and quantization can make its effective change depend on the image.

P Full Perturbation-Transfer Matrices

Figures 16, 17, and 18 show the prediction matrices behind Table 2, three methods per figure. Each column is one method; the rows are cross-class transfer R, clean confusion C, and within-class transfer S. Cells show row-normalized percentages rounded to the nearest integer, and colors use the unrounded probabilities. All matrices use the same saved predictions as the main analysis.

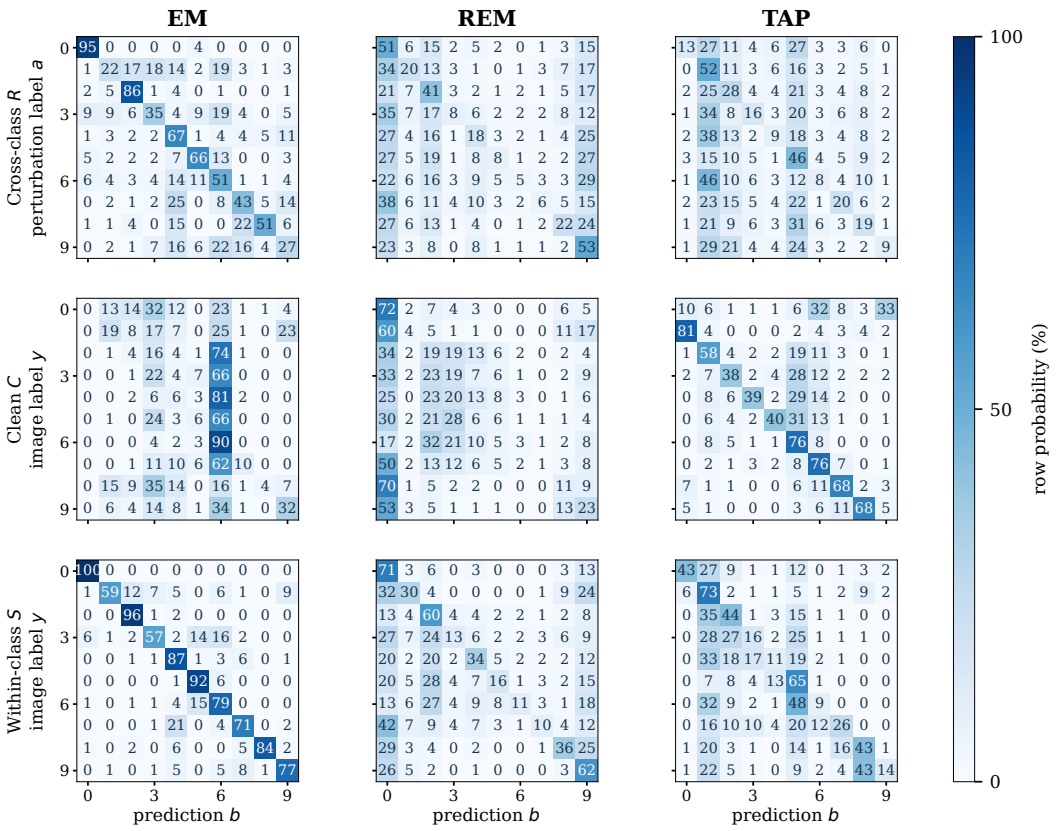

Figure 16: Perturbation-transfer matrices for EM, REM, and TAP (columns). Rows: cross-class transfer R, clean confusion C, and within-class transfer S. Cell values are percentages.

Figure 17 places SEP beside PIL and SP. The cross-class and within-class diagonals contrast code following in PIL and SP with SEP's dependence on the image–perturbation pair.

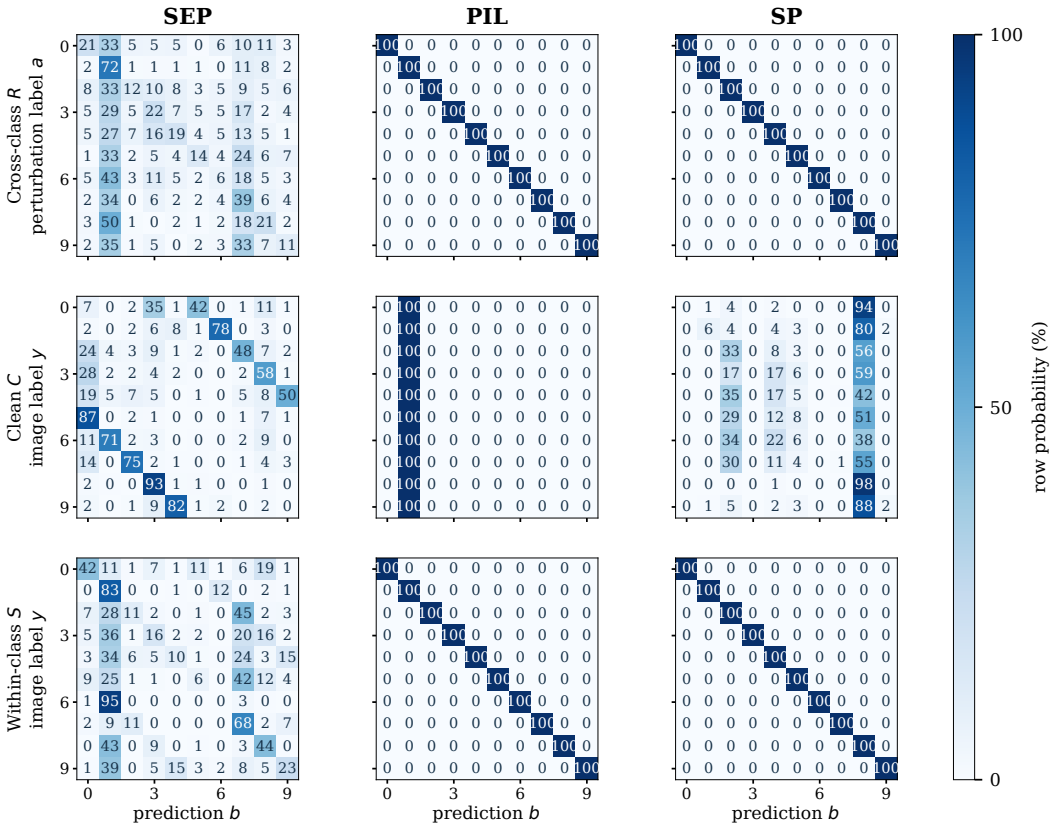

Figure 17: Perturbation-transfer matrices for SEP, PIL, and SP (columns). Rows: cross-class transfer R, clean confusion C, and within-class transfer S. Cell values are percentages.

Figure 18 shows AR, dense LCSC, and LCSC-LR. All three have diagonal cross-class and within-class matrices, while their clean predictions concentrate on a few classes: classes 3 and 5 for dense LCSC and class 0 for LCSC-LR.

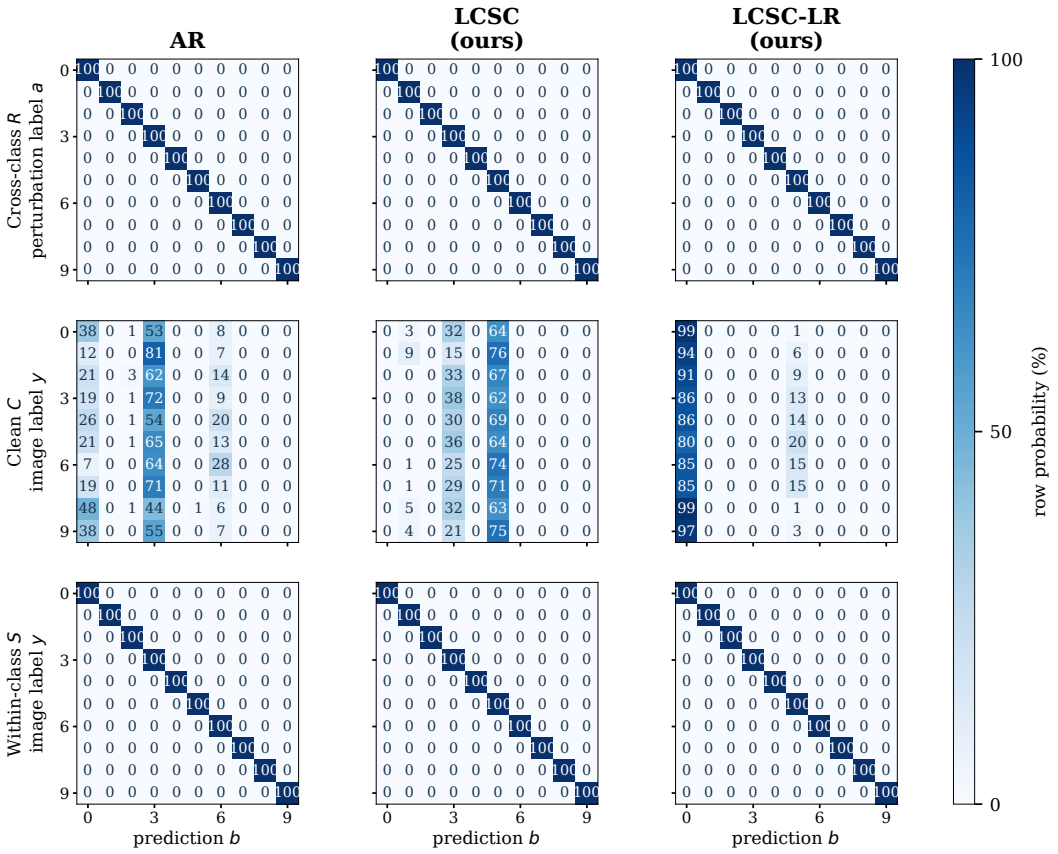

Figure 18: Perturbation-transfer matrices for AR, LCSC (ours), and LCSC-LR (ours) (columns). Rows: cross-class transfer R, clean confusion C, and within-class transfer S. Cell values are percentages.