CODEDNN: CODING-THEORETIC REDUNDANCY FOR ROBUST NEURAL CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural classifiers can degrade substantially under input corruptions and perturbations to learned features. We investigate a simple way to improve reliability: encode labels into redundant binary outputs and train a shared-backbone network to predict the resulting codeword. Coded Neural Networks (CodedNN) preserve the original label coordinates in multilabel classification (MLC), while redundant parity coordinates provide additional supervision and can optionally be exploited by belief-propagation (BP) decoding; in single-label classification (SLC), classes are represented by redundant binary codewords and prediction is performed by soft codeword matching. A central difficulty is that learned output errors are strongly dependent, so the independent-channel abstraction underlying classical error-correcting codes does not accurately describe neural predictions. We therefore develop a dependence-aware analysis of aggregate codeword errors, including a distribution-free correction-radius bound and a Gaussian-margin reference model. Empirically, redundant outputs provide gains in both clean and corrupted regimes. Across 13 clean MLC benchmarks, raw CodedNN improves micro-F1 over a matched binary-cross-entropy predictor by 1.5 percentage points on average, while explicit decoding provides additional robustness in some corrupted regimes. On CIFAR-10-C, redundant 8-bit output representations improve mean corruption accuracy by up to 1.67 percentage points over softmax with essentially unchanged clean accuracy, and on ImageNet the coded model exceeds softmax by 12.2 points under the strongest tested feature perturbation. These results show that structured output redundancy can improve neural reliability even when classical coding assumptions fail, providing a framework for exploiting redundancy through both representation learning and decoding.

1 INTRODUCTION

Modern neural classifiers can achieve high in-distribution accuracy yet degrade sharply when inputs or learned representations are corrupted. Improving this degradation behavior is important for reliable deployment, particularly when perturbations cannot be anticipated or reproduced during training. Communication systems address an analogous reliability problem by introducing structured redundancy. Rather than transmitting only the minimum representation of a message, error-correcting codes embed it into a larger codeword whose redundant coordinates impose algebraic consistency constraints. This motivates a natural question for machine learning: *can structured redundancy in the output representation make neural predictions more reliable, and under what conditions can a decoder exploit that redundancy?*

We study this question through Coded Neural Networks (CodedNN). Instead of predicting only task labels, CodedNN trains a neural network to predict a higher-dimensional binary representation generated by a linear block code. For multilabel classification (MLC), we use systematic codes so that the original labels remain directly accessible in designated coordinates, while additional parity coordinates impose structured supervision across the joint label vector. At inference, the systematic scores can be used directly, or the complete coded output can be passed through a decoder that explicitly exploits the parity constraints. For single-label classification (SLC), each discrete class is associated with a complete codeword and prediction is performed by soft codeword matching.

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.

The communication analogy is imperfect in an important way. All coded outputs are produced by a shared learned representation, so their errors can be strongly dependent. Empirically, we observe substantial cross-coordinate residual correlations and low-dimensional covariance structure. Consequently, the independent-bit abstraction underlying elementary coding arguments does not accurately describe the learned predictor. Rather than attempting to force the neural scores into an independent-channel model, we keep this dependence explicit. For arbitrary dependent bit-error indicators, we characterize the mean and variance of the total codeword error count and obtain a distribution-free Cantelli bound on exceeding a decoder's correction radius. We additionally derive a jointly Gaussian margin reference model that makes explicit how correlations between neural scores induce correlations between bit-error events. The Gaussian model is interpretive rather than an assumption required by CodedNN or by the distribution-free analysis.

This perspective separates two possible sources of robustness. Parity-structured targets can alter the representation learned during training even when only the systematic outputs are used at test time. Separately, an explicit decoder can exploit redundant predictions at inference. Our experiments isolate these effects using output-dimension-matched controls and decoder ablations, and show that they contribute differently across tasks and corruption regimes. More broadly, the single-label experiments show that redundant output representations can also improve robustness to common input corruptions and feature perturbations, suggesting that coding is useful not only as a decoding mechanism but also as a way to shape the learned output representation.

Coded representations have a substantial history in classification, including error-correcting output codes (ECOC) for multiclass prediction and direct applications of error-correcting codes to multilabel classification (Dietterich and Bakiri, 1995; Allwein et al., 2000; Ferng and Lin, 2013). Building on this literature, we focus on the shared-backbone neural setting, where coded outputs are learned jointly and exhibit structured dependence. CodedNN combines systematic redundant supervision, optional decoder-based inference, and a dependence-aware analysis, while controlled experiments isolate the roles of representation learning and decoding. Our main contributions are:

- **Structured redundant neural outputs.** We develop CodedNN with systematic coded label representations, direct systematic-label prediction, and optional decoder-based inference.
- **Dependence-aware coding analysis.** We show how the classical coding view must be modified for jointly learned neural outputs, whose errors can be strongly dependent. Rather than assuming independent bit errors, we keep their dependence explicit through the mean and variance of the aggregate codeword error count, yielding a distribution-free correction-radius bound via Cantelli's inequality. We complement this general analysis with a Gaussian-margin reference model that makes explicit how score correlations induce correlations between error events.
- **Controlled mechanism and robustness experiments.** Through output-dimension-matched controls and decoder ablations over 24 training seeds, we identify two complementary roles of redundancy: parity supervision can reshape the learned representation, while explicit decoding can exploit redundant predictions at inference. Across 13 MLC benchmarks, CodedNN also improves mean micro-F1 over matched BCE by 1.5 percentage points.
- **Robustness beyond MLC and the role of output geometry.** Controlled CIFAR-10 experiments show that redundant binary output representations can improve corruption robustness with essentially unchanged clean accuracy, while a geometry-matched nonlinear codebook outperforms BCH$(8, 4, 4)$ despite matching its code length and minimum Hamming distance. This indicates that classical code parameters alone do not explain robustness in the learned setting. Additional feature-corruption experiments on CIFAR-10 and ImageNet show that the benefits of redundant outputs can extend beyond multilabel classification.

2 Related Work

Error-correcting output codes and coded multilabel prediction. ECOC represents classes through redundant binary codewords and reduces multiclass prediction to binary learning problems (Dietterich and Bakiri, 1995). Allwein et al. (2000) developed a general margin-based treatment, and subsequent work studied code design and decoding (Escalera et al., 2010). Most directly related to our MLC setting, Ferng and Lin (2013) introduced an error-correcting-code framework for multilabel classification: base learners are treated as noisy channels, parity-checking labels aug-

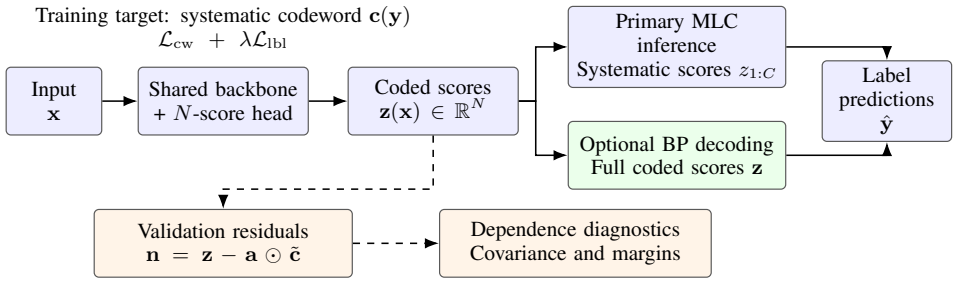


Figure 1: CodedNN pipeline. Training predicts a systematic codeword with a shared backbone and N-score head. Primary MLC inference reads raw systematic coordinates; optional BP uses the full raw code-aligned score vector. Validation residual statistics form a separate diagnostic branch and are not applied to the inference scores.

ment the original representation, several codes are compared, and both hard and real-valued soft predictions are decoded. Ferng and Lin (2013) therefore provides an important precursor for coded multilabel prediction. Unlike these approaches, our focus is not simply on using error-correcting codes as output representations. We study what coding-theoretic redundancy does when all coded coordinates are jointly learned through a shared neural representation, where their prediction errors are strongly dependent and the usual independent-channel coding abstraction breaks down. This setting reveals two distinct roles of redundancy—structured supervision during representation learning and error correction at inference—which we isolate experimentally and analyze without assuming independent output errors.

Coded outputs in deep classification. Coded representations have also been revisited in deep classification; for example, (Verma and Swami, 2019) modify output representations and decoding rules and report improvements in probability estimation and adversarial robustness. In contrast, we study how redundancy interacts with a shared neural representation, separating structured supervision from decoding and explicitly accounting for dependence among jointly learned outputs.

Multilabel representation learning. MLC methods exploit label structure through graphs (Chen et al., 2019), ranking objectives (Li et al., 2017), asymmetric losses (Ridnik et al., 2021), sparse label embeddings (Bhatia et al., 2015), aligned feature–label spaces (Yeh et al., 2017), and probabilistic latent-variable models (Bai et al., 2020; 2022). CodedNN explores a complementary representational direction: rather than compressing the label space, it deliberately expands the output dimension and uses algebraic redundancy to provide structured supervision across labels.

Robustness. CIFAR-10-C provides a standardized benchmark for common image corruptions (Hendrycks and Dietterich, 2019). Our interest is specifically whether redundancy introduced at the output changes degradation behavior. Our experiments examine whether output redundancy provides a complementary source of robustness to common corruptions and representation perturbations.

3 CodedNN Framework

Figure 1 summarizes the CodedNN pipeline. The primary MLC inference path reads raw systematic scores directly. Optional BP operates on the same raw code-aligned scores. Residual covariance is estimated only for diagnosis and analysis; it is not an inference transform.

3.1 Systematic Coded Label Representation

For an MLC problem with C binary labels, let $\mathbf{y} \in \{0,1\}^C$. We embed $\mathbf{y}$ into a systematic linear code with message length $K \geq C$ and codeword length $N > K$. With generator matrix

$$\mathbf{G} = [\mathbf{I}_K \mid \mathbf{P}],$$

the target codeword is

$$\mathbf{c}(\mathbf{y}) = [\mathbf{y} \| \mathbf{0}_{K-C}]\mathbf{G} \pmod 2 \in \{0,1\}^N.$$

The first C coordinates are the original labels, followed by optional zero padding and parity coordinates. The parity-check matrix $\mathbf{H} = [\mathbf{P}^\top \mid \mathbf{I}_{N-K}]$ satisfies $\mathbf{H}\mathbf{G}^\top = 0 \pmod 2$. We write the bipolar target as $\tilde{\mathbf{c}} = 2\mathbf{c} - 1 \in \{-1, +1\}^N$.

3.2 Neural Codeword Scores and Dependent Errors

The shared neural predictor produces real-valued codeword scores $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^N$. These scores are not assumed to be calibrated log-likelihood ratios. Let $\mathbf{a} \in \mathbb{R}^N_+$ denote coordinate-wise reliability scales estimated on validation data. We define the target-referenced score residual

$$\mathbf{n}(\mathbf{x}, \mathbf{y}) = \mathbf{z}(\mathbf{x}) - \mathbf{a} \odot \tilde{\mathbf{c}}(\mathbf{y}),$$

where $\odot$ denotes element-wise multiplication. Its centered covariance and correlation provide descriptive measures of shared-backbone dependence.

A representative Delicious diagnostic (Appendix D.1) has mean absolute off-diagonal correlation 0.451, with 95% of residual variance concentrated in 125 of 2,952 principal directions. These diagnostics motivate a dependence-aware analysis; they do not establish independence, Gaussianity, equal error probabilities, or calibrated likelihood ratios.

3.3 Dependence-Aware Error Analysis

The purpose of this analysis is not to introduce a new concentration inequality, but to obtain a valid coding-style reliability statement without replacing the dependent errors of a shared neural predictor by an independent-channel approximation. Let η_i be the decision threshold in score space and define the signed margin and error indicator

$$M_i = \tilde{c}_i(z_i - \eta_i), \qquad E_i = \mathbf{1}\{M_i < 0\}.$$

For the total number of codeword errors $S = \sum_{i=1}^N E_i$, write $p_i = \Pr(E_i = 1)$. Without any independence assumption,

$$\mu_S = \mathbb{E}[S] = \sum_{i=1}^N p_i, \qquad v_S = \mathrm{Var}(S) = \sum_{i=1}^N p_i(1-p_i) + 2\sum_{i<j} \mathrm{Cov}(E_i, E_j). \tag{1}$$

If a decoder is guaranteed to correct every pattern with at most t errors and $\mu_S < t + 1$, Cantelli's inequality gives

$$\Pr(\text{failure}) \le \Pr(S \ge t+1) \le \frac{v_S}{v_S + (t + 1 - \mu_S)^2}. \tag{2}$$

For an ideal bounded-distance decoder applied to a code of minimum distance $d_{\min}$, one may take $t = \lfloor (d_{\min} - 1)/2 \rfloor$. This statement does *not* give the same guarantee for iterative BP.

To obtain an interpretable link between score dependence and error-event dependence, consider a reference model in which (M_i, M_j) is jointly Gaussian with means μ_i, μ_j, standard deviations σ_i, σ_j, and correlation ρ_{ij}. With $a_i = -\mu_i/\sigma_i$,

$$\mathrm{Cov}(E_i, E_j) = \Phi_2(a_i, a_j; \rho_{ij}) - \Phi(a_i)\Phi(a_j), \tag{3}$$

where Φ_2 is the bivariate standard Gaussian cumulative distribution function (CDF). Equation (3) is a reference model only; Eq. (2) does not require Gaussianity. Derivations of the aggregate-error bound and Gaussian-margin error correlation are provided in Appendix A.

3.4 Prediction and Optional Decoding

For MLC, the primary prediction path uses the first C raw scores directly:

$$\hat{y}_c = \mathbf{1}\{\mathrm{sigmoid}(z_c) \ge \tau_c\}, \qquad c = 1, \ldots, C.$$

Per-label thresholds τ_c are selected on clean validation data and frozen for test evaluation. An optional BP path applies standard sum-product decoding to the full raw code-aligned score vector

and reads the refined systematic coordinates; decoder-specific score sign conventions are handled internally.

For SLC with assigned class codewords $\{\tilde{\mathbf{c}}_k\}_{k=1}^{C}$, prediction uses soft matching,

$$\hat{y} = \arg\max_k \langle \mathbf{z}(\mathbf{x}), \tilde{\mathbf{c}}_k \rangle,$$

without covariance conditioning.

3.5 Training Objective

The MLC predictor is trained with $\mathcal{L} = \mathcal{L}_{\text{cw}} + \lambda \mathcal{L}_{\text{lbl}}, \qquad \lambda = \frac{K}{2N}$, where $\mathcal{L}_{\text{cw}}$ is binary cross-entropy over the full codeword and $\mathcal{L}_{\text{lbl}}$ is a systematic-label BCE term using positive-class imbalance weights estimated from the training split. The main MLC experiments use direct training only; BP is applied post hoc in the decoder ablation and does not receive extra training time.

4 Experimental Setup

MLC benchmarks and baselines. We evaluate 13 standard MLC datasets spanning text, biology, music, audio, image, and video domains, with 6–983 labels (Appendix B). Matched clean-data baselines are a two-hidden-layer multilayer perceptron (MLP) trained with binary cross-entropy (BCE) and the same MLP trained with the log-sum-exp pairwise (LSEP) loss (Li et al., 2017). A targeted 10-seed experiment compares against Asymmetric Loss (ASL) (Ridnik et al., 2021) on Emotions, Birds, and Enron. For mechanism experiments, an output-dimension-matched overcomplete control uses the same N-dimensional head size but no parity-structured supervision.

Codes and training. MLC uses systematic rate-$1/3$ low-density parity-check (LDPC) codes with $N = 3K$ and a random column-weight-3 parity matrix, generated once per dataset and held fixed across training seeds. Optional BP uses 15 iterations. MLC models are trained for 100 epochs with Adam (learning rate 3×10^{-4}, weight decay 10^{-4}, batch size 64). The hidden width is $\text{clamp}(2d_{\text{feat}}, 128, 512)$ with BatchNorm, ReLU, and dropout. The best checkpoint is selected by validation mean average precision (mAP); its per-label validation thresholds are frozen for test evaluation. Clean Table 1 uses seeds $\{1, 42, 123\}$. Mechanism studies use 24 independently trained seeds.

Controlled feature corruption. For MLC robustness tests we perturb standardized features as

$$\mathbf{x}_\sigma = \mathbf{x} + \sigma\, \mathbf{s}_{\text{train}} \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}),$$

where $\mathbf{s}_{\text{train}}$ is the per-feature training standard deviation. The output-dimension-matched study uses σ from 0 to 2 in increments of 0.25 and 10 paired corruption realizations at each nonzero level. The decoder study uses $\sigma \in \{0, 1, 2\}$. Each inference path receives its own thresholds selected once on clean validation data; thresholds are then frozen under corruption.

Metrics and uncertainty. Primary MLC performance is micro-F1 (μF1); we also report mAP and, for mechanism diagnosis, macro-F1 and codeword bit-error rate (BER). Clean 3-seed results report mean $\pm$ sample standard deviation. For 24-seed mechanism experiments, corruption draws are averaged within each trained seed and 95% bootstrap confidence intervals are computed across model seeds. SLC uses top-1 accuracy. For the five-seed matched CIFAR-10 control, we report paired 95% bootstrap confidence intervals across training seeds.

5 Results: Multilabel Classification

5.1 Clean Performance

Table 1 shows that coded supervision can improve clean MLC performance across a diverse set of tasks. Relative to matched BCE, raw CodedNN improves mean μF1 by 1.51 percentage points and is higher on 8 of 13 datasets. Relative to LSEP, the mean improvement is 1.30 points and CodedNN is higher on 7 of 13 datasets. The largest gains occur on Birds, Genbase, Enron, and

Table 1: Clean multilabel results for matched MLP baselines and raw CodedNN systematic readout (mean ± sample standard deviation over seeds $\{1, 42, 123\}$). The CodedNN column uses raw systematic scores from the selected training checkpoint: no covariance transform and no BP. $\Delta\mu$F1 is CodedNN minus BCE in percentage points. Bold denotes the best value among the three matched methods for each metric and dataset.

	MLP–BCE		MLP–LSEP		CodedNN raw		
Dataset	mAP	μF1	mAP	μF1	mAP	μF1	$\Delta\mu$F1
Emotions	$0.722_{\pm 0.000}$	$0.694_{\pm 0.023}$	$\mathbf{0.729}_{\pm 0.009}$	$\mathbf{0.722}_{\pm 0.004}$	$\mathbf{0.729}_{\pm 0.022}$	$0.691_{\pm 0.013}$	-0.33
Scene	$0.826_{\pm 0.010}$	$0.756_{\pm 0.007}$	$0.824_{\pm 0.002}$	$\mathbf{0.763}_{\pm 0.004}$	$\mathbf{0.831}_{\pm 0.004}$	$0.749_{\pm 0.015}$	-0.76
Birds	$0.306_{\pm 0.071}$	$0.297_{\pm 0.063}$	$0.280_{\pm 0.013}$	$0.338_{\pm 0.025}$	$\mathbf{0.371}_{\pm 0.009}$	$\mathbf{0.347}_{\pm 0.027}$	+4.93
Genbase	$\mathbf{0.902}_{\pm 0.023}$	$0.933_{\pm 0.032}$	$0.896_{\pm 0.021}$	$0.870_{\pm 0.183}$	$0.886_{\pm 0.032}$	$\mathbf{0.959}_{\pm 0.025}$	+2.63
Yeast	$0.492_{\pm 0.003}$	$0.607_{\pm 0.010}$	$0.492_{\pm 0.004}$	$\mathbf{0.631}_{\pm 0.010}$	$\mathbf{0.500}_{\pm 0.008}$	$0.615_{\pm 0.008}$	+0.87
Enron	$0.225_{\pm 0.010}$	$0.390_{\pm 0.057}$	$\mathbf{0.228}_{\pm 0.010}$	$0.412_{\pm 0.010}$	$0.214_{\pm 0.010}$	$\mathbf{0.484}_{\pm 0.009}$	+9.39
Bibtex	$\mathbf{0.353}_{\pm 0.005}$	$0.411_{\pm 0.001}$	$0.341_{\pm 0.006}$	$\mathbf{0.423}_{\pm 0.006}$	$0.324_{\pm 0.003}$	$0.405_{\pm 0.006}$	-0.57
TMC2007	$0.739_{\pm 0.034}$	$0.748_{\pm 0.013}$	$0.730_{\pm 0.013}$	$0.744_{\pm 0.005}$	$\mathbf{0.762}_{\pm 0.004}$	$\mathbf{0.766}_{\pm 0.003}$	+1.80
CAL500	$0.246_{\pm 0.009}$	$0.334_{\pm 0.010}$	$\mathbf{0.253}_{\pm 0.019}$	$\mathbf{0.383}_{\pm 0.012}$	$0.248_{\pm 0.003}$	$0.349_{\pm 0.025}$	+1.55
Mediamill	$\mathbf{0.236}_{\pm 0.001}$	$0.566_{\pm 0.003}$	$0.219_{\pm 0.004}$	$0.500_{\pm 0.002}$	$0.223_{\pm 0.004}$	$\mathbf{0.570}_{\pm 0.006}$	+0.33
RCV1sub1	$0.111_{\pm 0.001}$	$0.133_{\pm 0.001}$	$\mathbf{0.125}_{\pm 0.003}$	$\mathbf{0.175}_{\pm 0.004}$	$0.115_{\pm 0.002}$	$0.134_{\pm 0.002}$	+0.11
Bookmarks	$\mathbf{0.267}_{\pm 0.004}$	$\mathbf{0.358}_{\pm 0.005}$	$0.243_{\pm 0.004}$	$0.349_{\pm 0.011}$	$0.259_{\pm 0.004}$	$0.356_{\pm 0.001}$	-0.21
Delicious	$0.183_{\pm 0.002}$	$\mathbf{0.344}_{\pm 0.002}$	$0.165_{\pm 0.001}$	$0.288_{\pm 0.003}$	$\mathbf{0.183}_{\pm 0.002}$	$0.343_{\pm 0.002}$	-0.09
						Mean $\Delta\mu$F1 vs. BCE	+1.51
						Mean $\Delta\mu$F1 vs. LSEP	+1.30
						Mean $\Delta\mu$F1 vs. stronger matched baseline	-0.21

TMC2007, indicating that the benefit of redundant supervision varies with the structure of the task. The 10-seed ASL comparison in Appendix D.2 provides an additional matched-loss comparison on three representative datasets. To test whether the construction extends beyond the simple MLP backbone used in Table 1, Appendix C integrates the coded output layer into C-GMVAE, a stronger probabilistic MLC model, and finds competitive performance with improvements on several metrics.

5.2 Controlled Corruption and Capacity Matching

To isolate parity structure from output expansion, define

$$\Delta_{\mathrm{code}}(\sigma) = F_{\mathrm{coded,raw}}(\sigma) - F_{\mathrm{over,raw}}(\sigma), \quad R_{\mathrm{code}}(\sigma) = \Delta_{\mathrm{code}}(\sigma) - \Delta_{\mathrm{code}}(0),$$

where F is μF1. Figure 5 in Appendix D.3 isolates the contribution of parity structure from that of output expansion by comparing CodedNN with an output-dimension-matched overcomplete head. The clean-adjusted effect varies across datasets: it is small on Emotions, positive on Scene, and negative on Enron. This variation shows that parity-structured supervision changes how the learned representation responds to corruption, but that the direction and magnitude of the effect depend on the task.

6 Ablation and Mechanism Analysis

6.1 Does Parity Structure Matter?

Panel A of Table 2 summarizes the output-dimension-matched study at the strongest tested corruption. At $\sigma = 2$, the clean-adjusted coding effect is +0.17 pp on Emotions, +1.04 pp on Scene, and -2.66 pp on Enron. Full curves and seed-level values appear in Appendix D.3. Parity structure can therefore alter corruption behavior even after controlling for output dimension, although both the direction and magnitude of the effect are dataset dependent.

Table 2: Controlled mechanism experiments with 24 independently trained model seeds. All values are percentage points. Panel A compares raw CodedNN against a capacity-matched overcomplete head; $R_{\mathrm{code}}(\sigma) = \Delta_{\mathrm{code}}(\sigma) - \Delta_{\mathrm{code}}(0)$. Panel B compares standard BP with raw systematic readout from the same CodedNN checkpoints; $R_{\mathrm{BP}}(2) = G_{\mathrm{BP}}(2) - G_{\mathrm{BP}}(0)$ and $\Delta\mathrm{BER}(2)$ is post-BP minus raw codeword BER. Full confidence intervals and seed-level results are in Appendix D.4.

Panel A: capacity-matched parity supervision

Dataset	$\Delta_{\mathrm{code}}(0)$	$\Delta_{\mathrm{code}}(2)$	$R_{\mathrm{code}}(2)$	Mean R_{code}, $\sigma > 0$
Emotions	-0.62	-0.45	+0.17	+0.23
Scene	-0.75	+0.29	+1.04	+0.29
Enron	+1.87	-0.79	-2.66	-2.26

Panel B: decoder effect

Dataset	$G_{\mathrm{BP}}(0)$	$G_{\mathrm{BP}}(1)$	$G_{\mathrm{BP}}(2)$	$R_{\mathrm{BP}}(2)$	$\Delta\mathrm{BER}(2)$
Emotions	+0.05	+0.76	+1.51	+1.46	-0.07
Scene	-0.02	-0.48	-1.16	-1.14	+0.07
Enron	+11.02	+7.33	+1.30	-9.72	-0.47

6.2 Does BP Exploit the Redundancy?

Panel B compares raw systematic readout with standard BP on the same 24 coded checkpoints, without retraining. The results demonstrate that the redundant coordinates can provide useful inferential information beyond the raw systematic outputs. On Emotions, the BP gain grows from $+0.05$ pp on clean data to $+1.51$ pp at $\sigma = 2$, corresponding to a clean-adjusted gain of $+1.46$ pp (95% bootstrap CI approximately $[+0.36, +2.60]$ pp). On Enron, BP provides a substantial clean μF1 gain of $+11.02$ pp and remains beneficial at $\sigma = 2$. Scene exhibits a different regime in which BP does not improve μF1 under strong corruption. Together, these results show that explicit decoding can extract useful information from the redundant predictions, with the resulting task-level gain determined by the operating regime and evaluation metric.

Dependence diagnostics. The score-residual covariance reported in Appendix D.1 is descriptive rather than an inference stage. It establishes that shared-backbone outputs can be strongly coupled, motivating the explicit covariance terms in the aggregate-error analysis. Appendix D.1 further shows that pooled residuals have a Gaussian-like central region but significant tail deviations; a Shapiro–Wilk test rejects exact Gaussianity.

7 Extension to Single-Label Classification

We next evaluate whether coded output representations transfer to conventional SLC. Unlike MLC, the admissible output set is finite: each class is assigned a fixed binary codeword and prediction uses soft codeword matching. For CIFAR-10, we use a ResNet-18 backbone. Our controlled codebook study compares softmax, extended BCH$(8, 4, 4)$, and a fixed nonlinear 8-bit codebook closely matched to the BCH codebook in minimum distance and low-order geometry. We additionally report results for an extended BCH$(16, 8, 5)$ implementation in Appendix D.5. For ImageNet-1k, we use a ResNet-50 backbone with an LDPC$(192, 96)$ representation. These experiments use coded scores directly and no covariance conditioning.

7.1 Common Image Corruptions

Table 3 reports the controlled five-seed CIFAR-10 comparison designed to separate the effect of redundant output geometry from the specific algebraic structure of BCH. The softmax baseline in this table is independently retrained within the paired five-seed protocol and is therefore distinct from the separately trained softmax baseline used in the auxiliary CIFAR-10-C results in Appendix D.5. We compare softmax, BCH$(8, 4, 4)$, and a fixed nonlinear 8-bit codebook constructed before training. The nonlinear codebook has affine-span rank 5 over $\mathrm{GF}(2)$, compared with rank 4 for BCH. Clean accuracy is essentially unchanged across the three models (90.16% for softmax, 90.04% for

Table 3: Controlled CIFAR-10 and CIFAR-10-C comparison over five paired training seeds. The geometry-matched nonlinear codebook has the same code length and minimum Hamming distance as BCH$(8, 4, 4)$, the same row-weight multiset and zero-codeword class assignment, and closely matched column balance. CIFAR-10-C accuracy is averaged over all 19 corruption types and five severity levels. The final column reports the mean paired CIFAR-10-C difference relative to softmax, in percentage points (pp), with 95% bootstrap confidence intervals across training seeds.

Model	Clean	CIFAR-10-C mean	Δ vs. softmax (pp)
Softmax	0.9016	0.6844	–
BCH$(8, 4, 4)$	0.9004	0.6905	$+0.61$ $[-0.58, +1.94]$
Geometry-matched nonlinear	0.9003	0.7011	$+1.67$ $[+0.59, +2.76]$

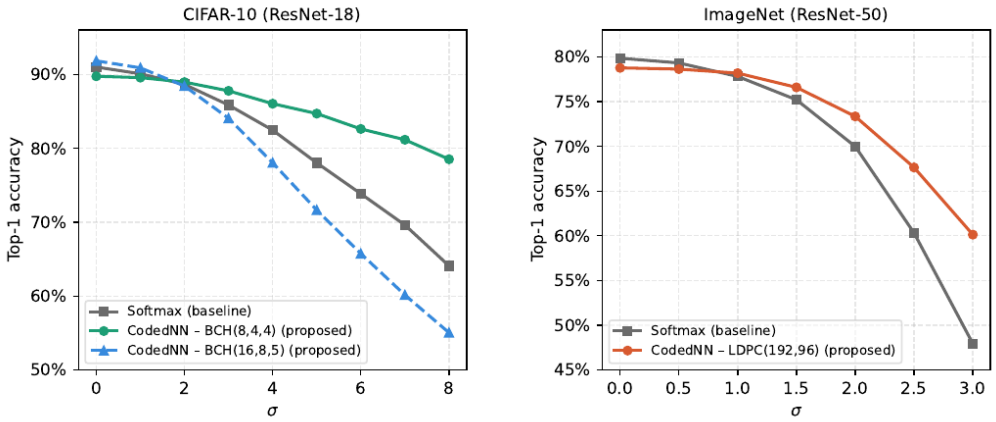


Figure 2: Top-1 accuracy under backbone-feature corruption. Left: CIFAR-10 with ResNet-18 and two coded implementations. Right: ImageNet-1k with ResNet-50 and an LDPC$(192, 96)$ representation. On CIFAR-10, the 8-bit coded representation degrades more gracefully than softmax as corruption increases. On ImageNet, the coded model crosses the softmax baseline at approximately $\sigma = 1$ and reaches a 12.2 pp advantage at $\sigma = 3$.

BCH, and 90.03% for the geometry-matched nonlinear code), whereas mean CIFAR-10-C accuracy is 68.44%, 69.05%, and 70.11%, respectively. The nonlinear code exceeds BCH by 1.06 pp (95% paired bootstrap CI $[0.62, 1.45]$ pp), with its advantage increasing from 0.35 pp at severity 1 to 1.63 pp at severity 5. Thus, matching code length, minimum Hamming distance, and several elementary codebook statistics does not produce identical robustness, indicating that properties of the redundant representation beyond these classical code parameters matter in the shared-backbone setting.

7.2 Controlled Feature-Space Corruption

We also perturb the backbone representation before the classification head using Gaussian noise scaled to the feature standard deviation. Figure 2 shows that coded output representations can substantially improve robustness to feature perturbations, with the strongest gains emerging at higher corruption levels. On ImageNet, the coded model begins below softmax on clean data, crosses it at approximately $\sigma = 1$, and exceeds it by 12.2 percentage points at $\sigma = 3$. On CIFAR-10, the 8-bit coded representation degrades more gracefully than softmax and retains the strongest performance under severe feature corruption.

8 Discussion and Limitations

What does the coded representation contribute? The experiments identify two complementary mechanisms: parity targets can reshape the learned representation during training, while an explicit

decoder can exploit redundant predictions at inference. Their relative importance is task dependent. The SLC experiments further show that redundant representations can improve corruption robustness without explicit decoding, while the controlled CIFAR-10 codebook study indicates that robustness is not determined by classical parameters such as code length and minimum Hamming distance alone; output geometry and learnability also matter.

What determines decoder benefit? The decoder ablation shows that redundant predictions contain inferential information that BP can exploit, but that the benefit depends on the operating regime and evaluation metric. On Emotions, the BP gain increases from near zero on clean data to $+1.51$ pp at the strongest corruption level, while Enron exhibits a large clean μF1 gain. Differences between codeword BER, ranking metrics, and hard-decision metrics indicate that successful error correction at the coded-output level does not translate identically to every task metric.

Dependence is a property of the learned representation. The shared backbone produces dependent output scores, unlike an independent-bit abstraction. We use residual covariance to motivate dependence-aware error analysis, not to transform scores into an independent channel. Second-order decorrelation would not establish independence, Gaussianity, equal error probabilities, or calibrated likelihood ratios. The Gaussian-margin model should therefore be read only as a reference model; the aggregate-error identities and Cantelli bound are distribution-free. Accordingly, the role of the theory is not to predict the empirical gain of CodedNN or to provide a tight performance characterization of belief propagation. Rather, it identifies the dependence terms omitted by an independent-bit analysis and provides a valid reliability bound for arbitrary dependent error indicators whenever a correction radius is specified.

Limitations. First, the magnitude and source of CodedNN's gains vary across datasets: some tasks benefit primarily from coded supervision, while others benefit more from explicit decoding or show little improvement. Second, MLC output expansion from C to $N > C$ increases head size, memory, and computation; extreme sparse-label problems may favor compressed-label or embedding-based approaches. The present work uses fixed hand-constructed codes rather than codebooks optimized jointly with the representation. Third, SLC and MLC use coding differently: SLC chooses among a finite set of class codewords, while MLC keeps each original label as an individual systematic coordinate and uses parity constraints across the joint binary vector. Conclusions should not be transferred unchanged between these settings. Fourth, our MLC corruption is a controlled Gaussian feature perturbation; it does not represent arbitrary domain shift, label shift, missingness, or modality-specific deployment corruption. Fifth, model selection and hard-decision thresholds depend on a held-out validation split, which may be less representative for rare labels. Finally, the broad MLC benchmark uses three training seeds, while the more statistically intensive 24-seed studies cover three representative datasets; the C-GMVAE study evaluates compatibility with a stronger probabilistic backbone rather than serving as a statistically powered superiority comparison.

Future directions. The codebook-control results motivate learning structured redundant representations jointly with the backbone and identifying which geometric properties make a redundant representation both learnable and robust. Other promising directions include designing decoders or calibration objectives specifically for dependent neural scores, using sparse/local parity constraints for extreme-label problems, and evaluating output redundancy under task-specific real-world shifts and missing-feature mechanisms.

9 Conclusion

We introduced Coded Neural Networks (CodedNN), which incorporate coding-theoretic redundancy into neural output representations while accounting for the strong dependence among jointly learned outputs. Experiments show that redundancy can contribute through both structured supervision during training and explicit decoding at inference. Across 13 multilabel benchmarks, raw CodedNN improves mean micro-F1 over matched BCE by 1.5 percentage points; in single-label classification, redundant representations improve robustness on CIFAR-10-C and under strong ImageNet feature perturbations. Controlled codebook experiments further show that robustness is not determined by classical code parameters alone, motivating the joint design of neural representations, redundant output geometry, and decoding.

Acknowledgments

We used OpenAI's ChatGPT for language editing, coding assistance, and identifying related works. The authors have fully reviewed and remain responsible for all content, mathematical statements, and experimental results.

Reproducibility Statement

Appendix A provides derivations for the dependence-aware analysis; Appendix B reports dataset splits, preprocessing, code dimensions, and optimization details; Appendix C evaluates compatibility with a stronger probabilistic MLC backbone; and Appendix D provides residual diagnostics, additional baseline comparisons, full seed-level mechanism results, and detailed CIFAR-10-C results. Upon acceptance, we will publicly release the code, experimental configurations, and supporting results on GitHub, together with instructions for accessing the datasets.

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

Junwen Bai, Shufeng Kong, and Carla Gomes. Disentangled variational autoencoder based multi-label classification with covariance-aware multivariate probit model. In *IJCAI*, 2020.

Junwen Bai, Shufeng Kong, and Carla Gomes. Gaussian mixture variational autoencoder with contrastive learning for multi-label classification. In *ICML*, PMLR 162:1383–1398, 2022.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *NeurIPS*, 2015.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *CVPR*, 2019.

Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. NUS-WIDE: A real-world web image database from National University of Singapore. In *ACM CIVR*, 2009.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Sergio Escalera, Oriol Pujol, and Petia Radeva. On the decoding process in ternary error-correcting output codes. *IEEE TPAMI*, 32(1):120–134, 2010.

Chun-Sung Ferng and Hsuan-Tien Lin. Multilabel classification using error-correcting codes of hard or soft bits. *IEEE Transactions on Neural Networks and Learning Systems*, 24(11):1888–1900, 2013.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *ECML PKDD Discovery Challenge*, 2008.

Yuncheng Li, Yale Song, and Jiebo Luo. Improving pairwise ranking for multi-label image classification. In *CVPR*, 2017.

Kenta Nakai and Minoru Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14(4):897–911, 1992.

Tal Ridnik, Emanuel Ben-Baruch, Noy Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *ICCV*, 2021.

Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. In *NeurIPS*, 2019.

Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent space for multi-label classification. In *AAAI*, 2017.

A Proofs for the Dependence-Aware Analysis

A.1 Aggregate Error Count

Let $S = \sum_{i=1}^{N} E_i$, $E_i \in \{0, 1\}$, and $p_i = \Pr(E_i = 1)$. By linearity of expectation,

$$\mathbb{E}[S] = \sum_i p_i.$$

For the variance,

$$\mathrm{Var}(S) = \sum_i \mathrm{Var}(E_i) + 2\sum_{i<j} \mathrm{Cov}(E_i, E_j) = \sum_i p_i(1 - p_i) + 2\sum_{i<j} \mathrm{Cov}(E_i, E_j).$$

Suppose a decoder is guaranteed to recover the correct codeword whenever $S \le t$. Then $\{\text{failure}\} \subseteq \{S \ge t + 1\}$. If $\mu_S < t + 1$, setting $a = t + 1 - \mu_S > 0$ and applying Cantelli's inequality yields

$$\Pr(\text{failure}) \le \Pr(S - \mu_S \ge a) \le \frac{v_S}{v_S + a^2} = \frac{v_S}{v_S + (t + 1 - \mu_S)^2}.$$

For an ideal bounded-distance decoder, $t = \lfloor (d_{\min} - 1)/2 \rfloor$; this does not imply an identical guarantee for iterative BP.

A.2 Gaussian-Margin Error Correlation

Assume (M_i, M_j) is jointly Gaussian with means μ_i, μ_j, standard deviations σ_i, σ_j, and correlation ρ_{ij}. Define $E_i = \mathbf{1}\{M_i < 0\}$ and $a_i = -\mu_i/\sigma_i$. After standardization,

$$\Pr(E_i = 1) = \Phi(a_i), \qquad \Pr(E_i = 1, E_j = 1) = \Phi_2(a_i, a_j; \rho_{ij}).$$

Therefore,

$$\mathrm{Cov}(E_i, E_j) = \Phi_2(a_i, a_j; \rho_{ij}) - \Phi(a_i)\Phi(a_j).$$

This expression is used only for the Gaussian-margin reference model.

B Dataset and Implementation Details

Table 4 summarizes the 13 MLC benchmarks. Datasets with official Mulan train/test splits (Emotions, Scene, Birds, Genbase, Yeast, Enron, Bibtex, TMC2007, Mediamill, Delicious) use those splits, with 10% of training held out for validation. CAL500, Bookmarks, and RCV1sub1 use a random 70/15/15 train/validation/test split. Features are standardized using training-set statistics only.

Training details. All MLC models use Adam with learning rate 3×10^{-4}, weight decay 10^{-4}, and batch size 64. The encoder is a two-hidden-layer MLP with batch normalization, rectified linear unit (ReLU) activations, and dropout; hidden widths are $\mathrm{clamp}(2d_{\mathrm{feat}}, 128, 512)$. Models are trained for 100 epochs and selected using validation mAP. Per-label decision thresholds are selected on validation data for the chosen checkpoint and then fixed for test evaluation. The revised MLC pipeline has no post-hoc covariance-conditioning training stage. For the five-seed matched CIFAR-10 control in Table 3, all three conditions use the same CIFAR-stem ResNet-18 backbone and the same head hidden widths, with only the output dimensionality changing as required by the representation. They are trained from scratch for 40 epochs with AdamW (learning rate 10^{-3}, weight decay 5×10^{-4}, batch size 256) and use the same fixed 45,000/5,000 train/validation split. Checkpoints are selected by clean validation accuracy, and the official CIFAR-10 test set is used only for final evaluation. The fixed geometry-matched nonlinear codebook is constructed before training and reused across seeds. The earlier auxiliary CIFAR-10 experiments use extended BCH$(8, 4, 4)$ and extended BCH$(16, 8, 5)$ codebooks. For ImageNet, the pretrained ResNet-50 backbone is fine-tuned for approximately 15–20 epochs using Adam with learning rate 10^{-4}.

Table 4: Multilabel benchmark datasets. Density is the average fraction of active labels per sample. K and N are the LDPC message and codeword lengths. Datasets marked ‡ use a random 70/15/15 split; the remainder use the official Mulan split. The information is split into two panels to preserve the template font size.

Dataset	Domain	Input dim.	Samples	Labels	Density
Emotions	Music	72	593	6	.302
Scene	Image	294	2,407	6	.177
Birds	Audio	260	645	19	.056
Genbase	Biology	1186	662	27	.047
Yeast	Biology	103	2,417	14	.302
Enron	Text	1001	1,702	53	.064
Bibtex	Text	1836	7,395	159	.015
TMC2007	Text	500	28,596	22	.087
CAL500‡	Music	68	502	174	.150
Mediamill	Video	120	43,907	101	.043
RCV1sub1‡	Text	944	6,000	101	.029
Bookmarks‡	Text	2150	87,856	208	.019
Delicious	Text/Web	500	16,105	983	.019

Dataset	Train	Test	K	N
Emotions	391	202	8	24
Scene	1,211	1,196	8	24
Birds	322	323	20	60
Genbase	463	199	28	84
Yeast	1,500	917	16	48
Enron	1,123	579	56	168
Bibtex	4,880	2,515	160	480
TMC2007	21,519	7,077	24	72
CAL500‡	351	76	256	768
Mediamill	30,993	12,914	104	312
RCV1sub1‡	3,000	3,000	104	312
Bookmarks‡	61,499	13,179	208	624
Delicious	12,920	3,185	984	2952

C Compatibility with a Stronger Multilabel Backbone

To test whether the coded output construction extends beyond the simple MLP backbone, we integrate a coded head into the Gaussian mixture variational autoencoder with contrastive learning (C-GMVAE) (Bai et al., 2022). We evaluate on nus-vec (Chua et al., 2009), yeast (Nakai and Kanehisa, 1992), scene (Boutell et al., 2004), and bookmarks (Katakis et al., 2008) using the published C-GMVAE values from Bai et al. (2022) as reference. The coded variant improves several metrics across these datasets: on nus-vec, macro-F1 increases from .226 to .239 and micro-F1 from .510 to .512; on scene, it improves example-F1, macro-F1, and Precision@1; and on bookmarks, macro-F1 increases from .291 to .295. Across the 20 metric–dataset combinations, Coded C-GMVAE strictly exceeds the published C-GMVAE value on 7 and ties it on 2 additional combinations. This experiment demonstrates that the CodedNN output construction can be integrated into a strong probabilistic multilabel backbone while retaining competitive performance and improving several metrics.

D Additional Empirical Results

D.1 Residual Distribution Diagnostics

The pooled standardized residuals have a quantile–quantile (Q–Q) fit of $R^2 = 0.856$, but the tail departures and Shapiro–Wilk rejection show that exact Gaussianity is not supported. The distribution-free analysis in Section 3.3 does not rely on this approximation.

Table 5: C-GMVAE compatibility study: example-F1, micro-F1, and macro-F1. Baseline method names and abbreviations follow Bai et al. (2022). The three metric blocks are stacked to preserve the template font size.

Example-F1				
Method	nus	yeast	scene	bkms
BR	.343	.630	.606	.171
MLKNN	.342	.618	.691	.213
HARAM	.396	.629	.717	.216
SLEEC	.431	.643	.718	.363
C2AE	.435	.614	.698	.309
LaMP	.376	.624	.728	.389
MPVAE	.468	.648	.751	.382
ASL	.468	.613	.770	.373
RBCC	.466	.605	.758	–
C-GMVAE	.481	.656	.777	.392
Coded C-GMVAE	.479	.640	.785	.382

Micro-F1				
Method	nus	yeast	scene	bkms
BR	.371	.655	.706	.125
MLKNN	.368	.625	.667	.181
HARAM	.415	.635	.693	.230
SLEEC	.428	.653	.699	.300
C2AE	.472	.626	.713	.316
LaMP	.472	.641	.716	.373
MPVAE	.492	.655	.742	.375
ASL	.495	.637	.753	.354
RBCC	.490	.623	.749	–
C-GMVAE	.510	.665	.762	.377
Coded C-GMVAE	.512	.658	.762	.363

Macro-F1				
Method	nus	yeast	scene	bkms
BR	.083	.373	.704	.038
MLKNN	.086	.472	.693	.041
HARAM	.157	.448	.713	.140
SLEEC	.135	.425	.699	.195
C2AE	.174	.427	.728	.232
LaMP	.203	.480	.745	.286
MPVAE	.211	.482	.750	.285
ASL	.208	.484	.765	.264
RBCC	.202	.480	.753	–
C-GMVAE	.226	.487	.769	.291
Coded C-GMVAE	.239	.479	.772	.295

Table 6: C-GMVAE compatibility study: Hamming accuracy and Precision@1. Baseline method names and abbreviations follow Bai et al. (2022). The two metric blocks are stacked to preserve the template font size.

Hamming accuracy				
Method	nus	yeast	scene	bkms
BR	.971	.782	.901	.990
MLKNN	.971	.784	.863	.991
HARAM	.971	.744	.902	.990
SLEEC	.971	.782	.894	.989
C2AE	.973	.764	.893	.991
LaMP	.980	.786	.903	.992
MPVAE	.980	.792	.909	.991
ASL	.975	.796	.912	.991
RBCC	.975	.793	.904	–
C-GMVAE	.984	.796	.915	.992
Coded C-GMVAE	.984	.792	.914	.990

Precision@1				
Method	nus	yeast	scene	bkms
BR	.443	.745	.700	.301
MLKNN	.456	.730	.675	.310
HARAM	.404	.682	.722	.312
SLEEC	.531	.745	.730	.415
C2AE	.569	.749	.703	.407
LaMP	.456	.740	.746	.420
MPVAE	.587	.743	.777	.437
ASL	.586	.752	.770	.418
RBCC	.582	.745	.777	–
C-GMVAE	.595	.751	.788	.465
Coded C-GMVAE	.599	.750	.792	.450

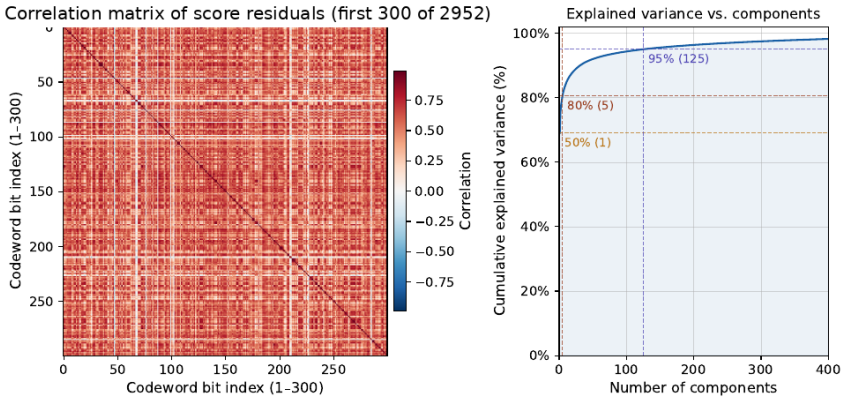


Figure 3: Correlation matrix of validation-set score residuals on Delicious ($N = 2{,}952$, rate-$1/3$ LDPC code). The mean absolute off-diagonal correlation is 0.451, and 95% of the residual variance is concentrated in 125 of 2,952 principal directions. The strongly correlated, low-dimensional second-order structure illustrates the mismatch between shared-backbone neural predictions and an independent bit-channel model.

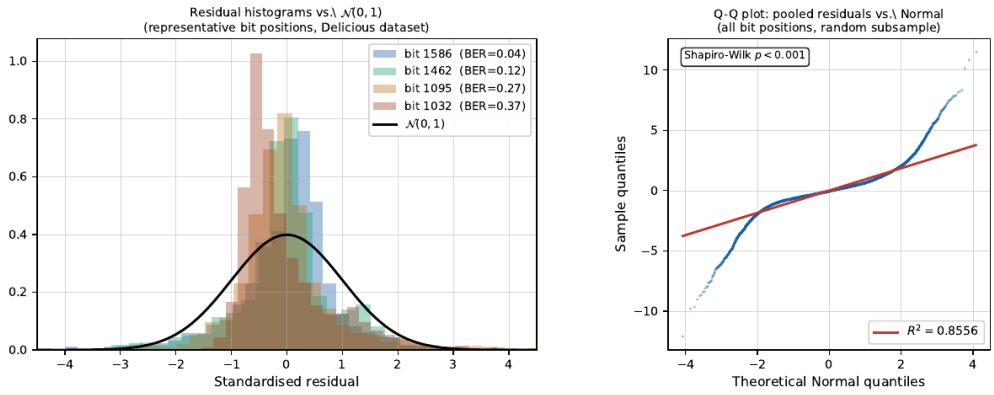


Figure 4: Residual-distribution diagnostics on Delicious. Left: standardized residual histograms at four representative output coordinates, overlaid with the standard Normal density. Right: pooled standardized residuals against Normal quantiles. Although the central region is Gaussian-like, visible tail deviations remain and the Shapiro–Wilk test rejects Gaussianity ($p < 0.001$). The figure therefore motivates treating the Gaussian-margin model as a reference approximation rather than an exact learned-channel model.

D.2 Matched 10-Seed ASL Comparison

Table 7: Matched 10-seed comparison with Asymmetric Loss. Values are micro-F1, mean $\pm$ standard deviation across training seeds.

Dataset	ASL	CodedNN	CodedNN wins
Emotions	0.6726 ± 0.0193	0.6903 ± 0.0126	9/10
Birds	0.2655 ± 0.0355	0.3082 ± 0.0295	9/10
Enron	0.3009 ± 0.0207	0.4861 ± 0.0383	10/10

CodedNN is higher on 9/10 seeds for Emotions and Birds and on all 10 seeds for Enron, showing that the observed gains on these datasets are not specific to using BCE as the baseline objective.

D.3 Full 24-Seed Output-Dimension-Matched Robustness Results

Table 8: Clean-adjusted coding effect $R_{\text{code}}(\sigma)$ in percentage points, with 95% bootstrap confidence intervals across 24 model seeds. Ten paired corruption realizations are averaged within each seed at nonzero σ.

σ	Emotions	Scene	Enron
0	+0.00 [+0.00,+0.00]	+0.00 [+0.00,+0.00]	+0.00 [+0.00,+0.00]
0.25	+0.19 [-0.03,+0.40]	-0.06 [-0.25,+0.12]	-0.32 [-0.50,-0.14]
0.5	+0.15 [-0.20,+0.50]	-0.14 [-0.65,+0.36]	-1.15 [-1.64,-0.66]
0.75	+0.22 [-0.24,+0.68]	-0.16 [-1.06,+0.76]	-2.14 [-3.09,-1.15]
1	+0.21 [-0.36,+0.78]	+0.00 [-1.31,+1.33]	-2.79 [-4.32,-1.30]
1.25	+0.30 [-0.38,+0.99]	+0.26 [-1.32,+1.96]	-3.09 [-5.03,-1.22]
1.5	+0.29 [-0.49,+1.07]	+0.56 [-1.22,+2.40]	-3.08 [-5.34,-0.90]
1.75	+0.27 [-0.58,+1.11]	+0.80 [-0.98,+2.59]	-2.88 [-5.35,-0.52]
2	+0.17 [-0.76,+1.06]	+1.04 [-0.69,+2.84]	-2.66 [-5.28,-0.18]

Table 8 reports the complete clean-adjusted $R_{\text{code}}(\sigma)$ curves with 95% bootstrap confidence intervals across 24 trained seeds, using raw code-aligned output scores throughout to preserve the semantics of the systematic label coordinates.

Table 9: Seed-level capacity-matched effects for Emotions. Δ_0 is the clean CodedNN-minus-overcomplete gap; remaining columns are clean-adjusted $R_{\text{code}}(\sigma)$, all in percentage points.

Seed	Δ_0	$R(.25)$	$R(.5)$	$R(.75)$	$R(1)$	$R(1.25)$	$R(1.5)$	$R(1.75)$	$R(2)$
1	-0.46	+0.68	+0.90	+1.05	+1.87	+1.71	+1.47	+1.34	+1.38
2	+0.98	+0.84	+0.48	+0.43	+0.33	-0.24	-0.09	-0.38	-0.44
3	+3.74	+0.57	+0.52	+0.58	+0.72	+0.28	-0.53	-1.03	-1.74
4	-0.72	+0.42	+0.41	+0.96	+1.42	+1.54	+1.19	+1.10	+1.70
5	-0.53	+0.05	-0.31	+0.11	-0.09	+0.74	+1.01	+0.84	+1.38
6	-0.84	-0.27	-0.76	-0.74	-1.10	-0.84	-1.02	-0.74	-0.70
7	+1.34	-0.10	-0.37	-0.83	-2.44	-3.36	-4.07	-4.72	-5.29
8	-0.72	+0.47	+1.18	+1.21	+0.93	+1.40	+1.51	+2.39	+2.13
9	-3.10	+1.00	+0.54	+0.75	+1.42	+1.63	+2.11	+2.55	+2.60
10	-1.56	+1.11	+1.63	+1.81	+2.48	+2.70	+2.80	+2.86	+2.88
11	+0.08	+0.62	+0.77	+0.07	-0.15	-0.37	-0.39	-0.72	-1.00
12	-1.52	+0.46	+0.28	+1.12	+1.10	+2.04	+1.85	+1.68	+2.13
13	+0.67	-0.09	-0.20	-0.87	-1.40	-0.88	-1.41	-1.86	-2.11
14	-1.17	-0.24	-0.97	-1.00	-1.46	-1.26	-1.83	-1.44	-1.29
15	-5.22	-0.34	-0.60	-0.36	-0.50	+0.01	+0.60	+1.22	+1.41
16	-2.14	+0.71	+1.40	+2.14	+2.60	+2.67	+2.79	+3.42	+2.97
17	+0.12	-0.80	-1.53	-1.83	-1.05	-0.81	-0.43	-0.45	+0.32
18	+0.51	+0.28	+0.26	-0.02	-0.09	-0.56	-0.48	-0.30	-0.99
19	-0.50	-0.84	-0.54	-0.95	+0.04	+0.11	+0.14	-0.02	-0.23
20	-0.01	+0.32	-0.04	+0.06	-0.63	-1.67	-1.87	-2.47	-3.57
21	+0.85	+0.22	+0.01	+0.20	-0.45	-0.79	-1.40	-1.30	-1.45
22	-1.89	-0.88	-1.58	-2.05	-2.47	-2.36	-1.96	-1.77	-1.68
23	-1.54	+0.16	+1.54	+2.41	+2.78	+4.28	+4.83	+4.54	+4.65
24	-1.36	+0.20	+0.61	+1.07	+1.12	+1.26	+2.16	+1.66	+1.09

Table 10: Seed-level capacity-matched effects for Scene. Δ_0 is the clean CodedNN-minus-overcomplete gap; remaining columns are clean-adjusted $R_{\text{code}}(\sigma)$, all in percentage points.

Seed	Δ_0	$R(.25)$	$R(.5)$	$R(.75)$	$R(1)$	$R(1.25)$	$R(1.5)$	$R(1.75)$	$R(2)$
1	-2.51	+0.67	+2.01	+3.52	+4.74	+5.94	+6.61	+6.48	+6.20
2	+1.37	+0.06	+0.15	-0.30	-0.13	+0.45	+0.33	+0.25	+0.39
3	+0.10	+0.17	-0.08	-0.31	-0.17	+0.31	+0.43	+0.34	+0.20
4	-0.41	-0.90	-1.33	-1.41	-1.18	-1.07	-0.74	-0.57	-0.21
5	-0.63	+0.03	+0.64	+0.91	+1.20	+1.40	+1.82	+1.86	+2.00
6	+2.31	-0.66	-0.37	+0.07	+0.47	+0.55	+0.58	+0.07	-0.39
7	-1.94	-0.27	-0.77	-1.30	-1.35	-1.57	-1.33	-0.88	-0.67
8	-1.55	+0.91	+2.34	+4.51	+6.93	+9.25	+10.73	+11.28	+11.07
9	-1.52	+0.38	+1.02	+1.78	+3.07	+4.81	+6.49	+7.23	+7.72
10	-1.06	-0.04	-0.00	-0.26	-0.51	-0.77	-0.57	-0.54	-0.12
11	+0.92	+0.37	+0.43	+0.55	+0.40	+0.69	+0.89	+1.26	+1.53
12	-1.07	-0.56	-1.62	-3.05	-3.62	-3.55	-3.13	-2.69	-2.10
13	-0.33	-0.75	-1.66	-2.08	-2.59	-3.36	-3.68	-3.60	-3.16
14	-0.64	+0.38	+1.78	+3.71	+6.04	+7.85	+9.05	+9.88	+10.12
15	-2.94	-0.60	-2.31	-3.51	-4.67	-4.74	-4.39	-3.54	-2.95
16	-1.15	-0.76	-2.92	-5.83	-8.57	-10.53	-10.77	-10.60	-9.56
17	-2.30	-0.21	-0.22	-0.17	-0.03	-0.18	+0.09	+0.72	+0.96
18	+0.81	+0.23	+0.36	+0.71	+1.58	+2.22	+2.46	+2.92	+3.04
19	-0.91	-0.47	-1.06	-1.28	-2.25	-2.85	-3.17	-2.87	-2.55
20	-1.34	+0.16	-0.15	+0.07	+0.45	+0.60	+0.56	+0.24	+0.72
21	+0.47	+0.26	+0.39	+1.24	+1.69	+1.91	+2.12	+2.48	+2.58

Seed	Δ_0	$R(.25)$	$R(.5)$	$R(.75)$	$R(1)$	$R(1.25)$	$R(1.5)$	$R(1.75)$	$R(2)$
22	-0.62	+0.23	+0.56	+1.37	+2.56	+3.43	+3.69	+4.12	+4.39
23	-1.74	-0.05	-0.33	-1.36	-1.90	-1.69	-1.27	-1.09	-0.69
24	-1.38	-0.07	-0.28	-1.38	-2.07	-2.94	-3.38	-3.51	-3.51

Table 11: Seed-level capacity-matched effects for Enron. Δ_0 is the clean CodedNN-minus-overcomplete gap; remaining columns are clean-adjusted $R_{\mathrm{code}}(\sigma)$, all in percentage points.

Seed	Δ_0	$R(.25)$	$R(.5)$	$R(.75)$	$R(1)$	$R(1.25)$	$R(1.5)$	$R(1.75)$	$R(2)$
1	+0.41	+0.20	+0.05	-0.25	-0.36	-0.41	-0.58	-0.88	-1.04
2	+8.22	-0.69	-2.62	-5.25	-7.27	-8.75	-9.75	-10.07	-10.33
3	-3.50	-0.20	-0.57	-0.34	+0.41	+1.26	+2.27	+2.98	+3.45
4	-1.51	-0.47	-2.22	-4.31	-4.82	-4.41	-3.32	-2.24	-1.10
5	+3.24	-0.81	-1.81	-3.14	-4.34	-4.84	-5.44	-5.58	-5.86
6	-1.53	+0.33	+1.42	+2.07	+2.45	+2.61	+2.33	+2.29	+1.98
7	+0.61	-0.84	-1.33	-1.35	-1.67	-1.59	-1.70	-1.90	-1.91
8	+2.75	+0.29	-0.83	-2.22	-2.90	-3.19	-3.41	-3.30	-3.09
9	+3.29	-0.03	-0.52	-1.59	-2.39	-2.74	-2.53	-2.38	-2.37
10	-0.75	+0.09	+0.79	+1.43	+1.84	+2.05	+2.13	+2.27	+2.10
11	-5.58	-0.10	+0.26	+1.24	+3.11	+4.49	+5.18	+5.85	+6.31
12	+3.92	-0.35	-2.07	-4.21	-5.48	-6.32	-6.52	-6.35	-5.88
13	+6.46	+0.02	-0.80	-2.52	-3.86	-4.78	-5.22	-5.33	-5.20
14	+3.99	-0.21	-1.28	-2.31	-3.12	-3.77	-3.91	-3.85	-3.62
15	-1.39	-0.12	-0.76	-1.27	-0.63	+0.28	+1.46	+2.06	+2.67
16	+1.26	-0.65	-2.07	-4.14	-6.21	-7.35	-7.23	-6.12	-4.83
17	-6.56	+0.35	+0.85	+2.07	+3.85	+5.49	+6.48	+7.20	+7.66
18	+4.36	-0.50	-2.57	-4.89	-6.69	-7.38	-7.50	-7.25	-6.89
19	+10.72	-1.21	-3.03	-5.80	-8.70	-10.99	-12.16	-12.71	-12.84
20	-4.73	+0.08	-0.47	-1.43	-2.39	-1.88	+0.14	+2.30	+4.19
21	+14.91	-0.99	-3.52	-6.75	-9.71	-11.92	-13.46	-14.35	-15.16
22	-3.96	-0.80	-1.44	-0.89	+0.26	+1.17	+2.17	+2.90	+3.47
23	-4.13	-0.20	-0.85	-0.92	-0.52	+0.28	+0.99	+1.53	+1.87
24	+14.45	-0.77	-2.27	-4.54	-7.85	-11.46	-14.42	-16.26	-17.41

D.4 Full 24-Seed Decoder Results

Table 12: BP gain in micro-F1 (percentage points) with 95% bootstrap confidence intervals across 24 model seeds. G_{BP} is BP minus raw; R_{BP} subtracts the clean BP gain.

Dataset	$G_{\mathrm{BP}}(0)$	$G_{\mathrm{BP}}(1)$	$G_{\mathrm{BP}}(2)$
Emotions	+0.05 [-0.70,+0.74]	+0.76 [-0.32,+1.88]	+1.51 [+0.35,+2.72]
Scene	-0.02 [-0.43,+0.42]	-0.48 [-1.06,+0.10]	-1.16 [-1.86,-0.47]
Enron	+11.02 [+8.84,+13.48]	+7.33 [+5.57,+9.37]	+1.30 [+0.64,+2.00]

Dataset	$R_{\mathrm{BP}}(1)$	$R_{\mathrm{BP}}(2)$
Emotions	+0.71 [-0.14,+1.54]	+1.46 [+0.36,+2.60]
Scene	-0.46 [-1.06,+0.11]	-1.14 [-1.96,-0.34]
Enron	-3.69 [-4.37,-3.03]	-9.72 [-11.62,-8.05]

We report the complete 24-seed decoder analysis, including aggregate BP gains, seed-level effects, and raw-versus-BP comparisons across multiple task metrics. The divergence between μF1, macro-F1, mAP, and BER is especially visible on Enron, reinforcing the importance of evaluating decoder benefit using task-level metrics rather than BER alone.

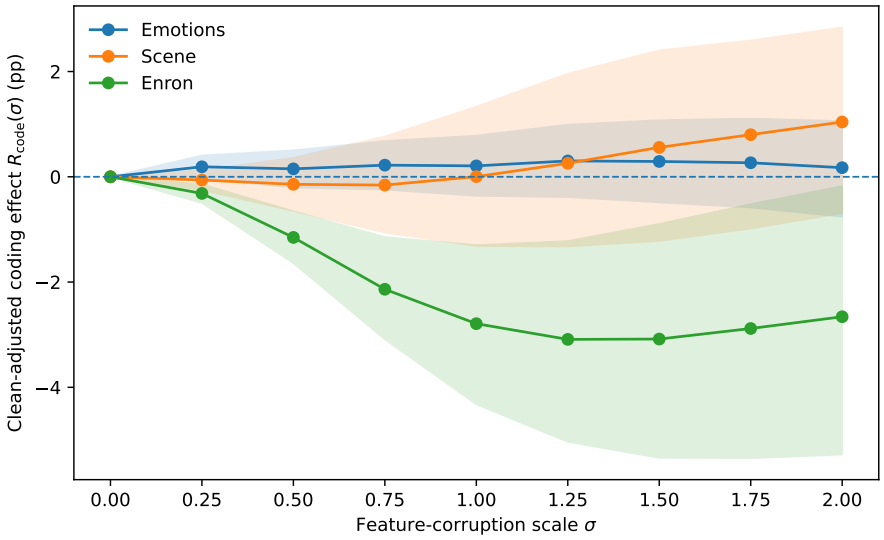


Figure 5: Clean-adjusted effect of parity-structured supervision under feature corruption. CodedNN is compared with an output-dimension-matched overcomplete head having the same output dimension but no parity structure. For each model seed, $R_{\text{code}}(\sigma) = [F_{\text{coded}}(\sigma) - F_{\text{over}}(\sigma)] - [F_{\text{coded}}(0) - F_{\text{over}}(0)]$. Results use 24 training seeds and 10 paired corruption realizations per nonzero noise level; thresholds are selected once on clean validation data and held fixed. Shading denotes 95% bootstrap confidence intervals across model seeds.

Table 13: Seed-level decoder effects for Emotions. Values are percentage points; G_s is BP minus raw micro-F1 at corruption scale s, $R_s = G_s - G_0$, and ΔBER_2 is post-BP minus raw BER at $\sigma = 2$.

Seed	G_0	G_1	G_2	R_1	R_2	ΔBER_2
1	+1.58	-1.04	-1.16	-2.63	-2.74	+0.23
2	-0.72	-1.44	-1.82	-0.72	-1.10	+0.07
3	+1.10	+4.17	+7.14	+3.07	+6.04	-0.33
4	+0.35	+2.47	+3.18	+2.13	+2.83	-0.51
5	-1.56	-4.90	-3.29	-3.34	-1.72	+0.24
6	+2.33	+6.81	+6.85	+4.48	+4.52	-0.33
7	-2.29	-1.01	+1.20	+1.28	+3.49	-0.33
8	+0.25	+0.53	+0.71	+0.28	+0.46	+0.09
9	+1.24	+0.67	+0.21	-0.57	-1.03	-0.27
10	-0.61	-3.04	-1.86	-2.42	-1.25	+0.01
11	-1.40	-0.79	-0.65	+0.61	+0.75	-0.05
12	+2.62	+2.20	+0.49	-0.41	-2.12	-0.15
13	+2.57	+3.72	+4.15	+1.15	+1.59	+0.09
14	+0.26	+0.47	+1.81	+0.21	+1.55	+0.11
15	-3.12	-0.68	+3.52	+2.43	+6.64	+0.01
16	-1.31	+1.71	+3.56	+3.02	+4.87	+0.21
17	+2.13	+5.37	+6.19	+3.24	+4.07	-0.20
18	-4.39	-1.07	+0.51	+3.32	+4.91	-0.20
19	+0.94	+0.80	+0.87	-0.15	-0.08	+0.33
20	+1.39	-0.37	+1.02	-1.76	-0.37	-0.50
21	-1.24	-2.69	-2.61	-1.45	-1.37	+0.26
22	-0.39	+0.61	-0.83	+1.00	-0.44	+0.06
23	-0.55	+0.79	+0.33	+1.34	+0.88	-0.44
24	+1.98	+4.85	+6.71	+2.87	+4.73	-0.17

Table 14: Seed-level decoder effects for Scene. Values are percentage points; G_s is BP minus raw micro-F1 at corruption scale s, $R_s = G_s - G_0$, and $\Delta\mathrm{BER}_2$ is post-BP minus raw BER at $\sigma = 2$.

Seed	G_0	G_1	G_2	R_1	R_2	$\Delta\mathrm{BER}_2$
1	+1.02	+2.83	+2.21	+1.82	+1.19	+0.18
2	-1.03	-1.34	-2.90	-0.32	-1.88	+0.71
3	-1.06	-0.92	-0.77	+0.14	+0.29	-0.88
4	-0.45	-0.56	-2.43	-0.10	-1.98	+0.36
5	+0.86	-0.24	-1.69	-1.10	-2.55	+0.14
6	-0.82	-1.91	-4.71	-1.09	-3.89	+0.24
7	+0.84	+0.18	-1.73	-0.66	-2.57	-0.25
8	+2.15	+0.48	-1.76	-1.67	-3.91	-0.01
9	-1.43	-0.57	+0.20	+0.87	+1.63	+0.19
10	-0.39	-3.08	-3.52	-2.68	-3.12	-0.20
11	-1.42	-0.15	+0.46	+1.27	+1.88	+0.12
12	-0.38	+0.87	-0.25	+1.26	+0.13	-0.57
13	-1.87	-1.11	+0.30	+0.76	+2.18	+0.42
14	-0.13	-0.08	-0.16	+0.05	-0.03	+0.34
15	+0.15	+0.77	+0.22	+0.62	+0.06	+0.16
16	+1.21	+0.35	+0.12	-0.86	-1.09	+0.55
17	+0.21	-3.14	-2.58	-3.36	-2.79	-0.14
18	-1.08	-1.99	-1.86	-0.91	-0.78	-0.09
19	-0.34	-2.05	-2.37	-1.70	-2.03	-0.43
20	+0.33	+1.72	+1.72	+1.39	+1.39	-0.09
21	+2.14	-1.45	-3.59	-3.59	-5.72	+0.11
22	+1.15	+1.80	+0.96	+0.66	-0.18	+0.15
23	+0.13	-1.86	-2.85	-1.98	-2.98	+0.36
24	-0.26	-0.08	-0.89	+0.19	-0.62	+0.36

Table 15: Seed-level decoder effects for Enron. Values are percentage points; G_s is BP minus raw micro-F1 at corruption scale s, $R_s = G_s - G_0$, and $\Delta\mathrm{BER}_2$ is post-BP minus raw BER at $\sigma = 2$.

Seed	G_0	G_1	G_2	R_1	R_2	$\Delta\mathrm{BER}_2$
1	+4.27	+2.25	-0.95	-2.03	-5.22	-0.64
2	+6.70	+4.21	+0.42	-2.49	-6.29	-0.48
3	+11.13	+7.91	+3.12	-3.22	-8.01	-0.59
4	+22.12	+15.01	+3.52	-7.11	-18.60	-0.24
5	+4.42	+2.49	-0.67	-1.93	-5.09	-0.57
6	+6.63	+2.28	-1.16	-4.35	-7.78	-0.78
7	+5.67	+3.87	+0.39	-1.80	-5.27	-0.78
8	+7.33	+4.35	+0.63	-2.99	-6.70	-0.54
9	+11.84	+6.44	+0.89	-5.40	-10.95	-0.31
10	+5.32	+3.97	+0.73	-1.35	-4.59	-0.49
11	+10.85	+7.11	+2.52	-3.74	-8.33	-0.66
12	+15.28	+9.30	+1.71	-5.97	-13.57	-0.56
13	+9.22	+7.02	+0.89	-2.20	-8.33	-0.42
14	+11.30	+6.14	+0.22	-5.15	-11.08	-0.47
15	+14.59	+9.63	+2.27	-4.95	-12.32	-0.33
16	+22.69	+19.28	+5.06	-3.41	-17.63	-0.23
17	+9.54	+5.25	-0.54	-4.29	-10.09	-0.39
18	+11.45	+10.33	+3.26	-1.12	-8.18	-0.29
19	+10.75	+5.22	+0.47	-5.53	-10.28	-0.38
20	+27.56	+21.29	+5.10	-6.27	-22.47	-0.11
21	+5.67	+3.41	-0.24	-2.26	-5.91	-0.75
22	+10.93	+6.54	+0.93	-4.40	-10.00	-0.35
23	+12.44	+8.16	+1.82	-4.28	-10.62	-0.34
24	+6.76	+4.40	+0.76	-2.35	-6.00	-0.58

Table 16: Raw versus BP task metrics for the 24-seed decoder study. Entries are means across model seeds (and across 10 paired corruption realizations per seed at $\sigma = 2$).

		Clean			$\sigma = 2$		
Dataset	Path	μF1	Macro-F1	mAP	μF1	Macro-F1	mAP
Emotions	Raw	0.683	0.682	0.733	0.488	0.469	0.562
Emotions	BP	0.683	0.681	0.713	0.503	0.484	0.543
Scene	Raw	0.742	0.752	0.824	0.454	0.418	0.542
Scene	BP	0.742	0.747	0.792	0.442	0.415	0.500
Enron	Raw	0.447	0.196	0.213	0.426	0.110	0.125
Enron	BP	0.558	0.149	0.197	0.439	0.082	0.119

D.5 Full CIFAR-10-C Results

Table 17: Accuracy on CIFAR-10-C. Each corruption entry is averaged over its five severity levels; the clean column reports standard CIFAR-10 test accuracy. The Mean column is the arithmetic mean over the 19 displayed corruptions. Both coded models improve mean corruption accuracy by approximately 2 percentage points over softmax.

Model	Clean	GN	Shot	Imp	Def	Glass	Motion	Zoom	Snow	Frost
Softmax	0.9100	0.4220	0.5416	0.4667	0.7784	0.4487	0.7134	0.7196	0.7205	0.7201
CodedNN ($n = 8$)	0.9075	0.5103	0.6120	0.5613	0.7743	0.5095	0.6944	0.7183	0.7470	0.7253
CodedNN ($n = 16$)	0.9184	0.4353	0.5494	0.5182	0.7925	0.4334	0.7299	0.7456	0.7579	0.7259

Model	Fog	Bright	Contrast	Elastic	Pixel	JPEG	Speckle	GBlur	Spatter	Saturate	Mean
Softmax	0.8268	0.8801	0.6784	0.7921	0.6719	0.7493	0.5642	0.6876	0.7766	0.8619	0.6853
CodedNN ($n = 8$)	0.8064	0.8853	0.6930	0.7751	0.6792	0.7354	0.6365	0.6947	0.7834	0.8605	0.7054
CodedNN ($n = 16$)	0.8527	0.8958	0.7310	0.8035	0.7195	0.7649	0.5800	0.7026	0.7937	0.8661	0.7051

Table 18: CIFAR-10-C accuracy averaged over all 19 corruption types at each severity level. Parentheses report absolute improvement over softmax in percentage points.

Severity	Softmax	CodedNN ($n = 8$)	CodedNN ($n = 16$)
1	0.8347	0.8360 (+0.13)	0.8453 (+1.06)
2	0.7672	0.7738 (+0.67)	0.7845 (+1.73)
3	0.7014	0.7183 (+1.69)	0.7239 (+2.25)
4	0.6181	0.6529 (+3.47)	0.6431 (+2.50)
5	0.5049	0.5459 (+4.10)	0.5290 (+2.41)

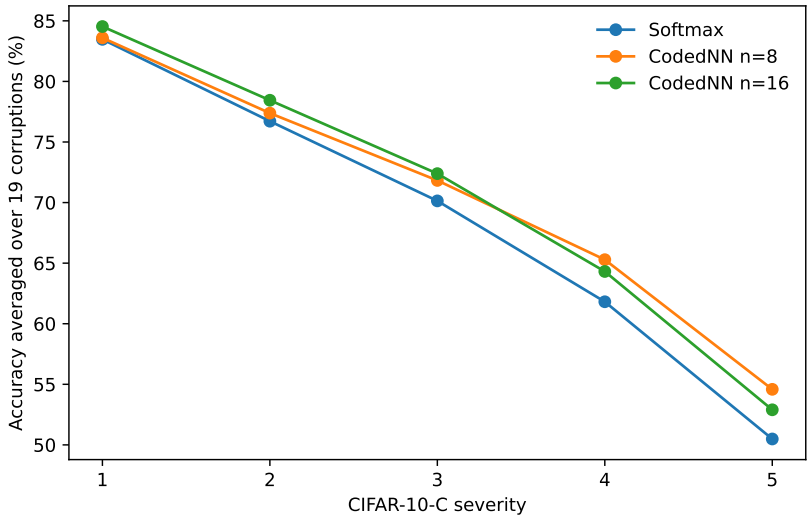


Figure 6: CIFAR-10-C top-1 accuracy averaged across the 19 corruption types at each severity level. Both coded implementations exceed the common softmax baseline at every severity. The two coded systems use different head parameterizations and are reported as separate coded implementations.

Across all 95 corruption–severity combinations, the $n = 8$ coded classifier exceeds softmax in 53 cases (with one exact tie), while the $n = 16$ coded classifier exceeds softmax in 88 cases. The full numerical breakdown is given below.

Table 19: Full CIFAR-10-C per-severity accuracies for CodedNN $n = 8$ versus the common softmax baseline, split into two portrait panels. Each severity group shows Softmax (S) / CodedNN (C).

Corruption	Severity 1		Severity 2		Severity 3	
	S	C	S	C	S	C
Gaussian Noise	0.7601	0.7692	0.5532	0.6053	0.3398	0.4497
Shot Noise	0.8353	0.8298	0.7455	0.7535	0.4950	0.5763
Impulse Noise	0.7926	0.8108	0.6534	0.6922	0.5209	0.5829
Defocus Blur	0.9095	0.9025	0.8905	0.8773	0.8333	0.8171
Glass Blur	0.4595	0.5269	0.4745	0.5417	0.5276	0.5809
Motion Blur	0.8557	0.8344	0.7709	0.7520	0.6763	0.6526
Zoom Blur	0.8218	0.8069	0.7896	0.7754	0.7335	0.7288
Snow	0.8238	0.8456	0.6907	0.7344	0.7269	0.7605
Frost	0.8560	0.8560	0.7928	0.7947	0.6888	0.6986
Fog	0.9041	0.9029	0.8834	0.8798	0.8529	0.8487
Brightness	0.9074	0.9067	0.8990	0.9015	0.8891	0.8927
Contrast	0.8987	0.8962	0.8397	0.8371	0.7724	0.7806
Elastic Transform	0.8547	0.8412	0.8522	0.8360	0.8084	0.7903
Pixelate	0.8700	0.8689	0.8006	0.7964	0.7453	0.7478
JPEG Compression	0.8221	0.8102	0.7664	0.7555	0.7494	0.7332
Speckle Noise	0.8328	0.8266	0.6923	0.7130	0.6014	0.6527
Gaussian Blur	0.9091	0.9024	0.8307	0.8149	0.7256	0.7175
Spatter	0.8676	0.8743	0.7975	0.8005	0.7389	0.7367
Saturate	0.8788	0.8719	0.8531	0.8414	0.9009	0.8995

Table 19 continued: Severities 4–5

Corruption	Severity 4		Severity 5	
	S	C	S	C
Gaussian Noise	0.2546	0.3875	0.2022	0.3396
Shot Noise	0.3847	0.5058	0.2477	0.3944
Impulse Noise	0.2399	0.4237	0.1269	0.2968
Defocus Blur	0.7246	0.7183	0.5339	0.5562
Glass Blur	0.3727	0.4319	0.4092	0.4659
Motion Blur	0.6752	0.6569	0.5891	0.5763
Zoom Blur	0.6735	0.6803	0.5798	0.5999
Snow	0.7075	0.7238	0.6537	0.6705
Frost	0.6794	0.6884	0.5834	0.5887
Fog	0.8178	0.7949	0.6756	0.6056
Brightness	0.8739	0.8797	0.8310	0.8460
Contrast	0.6409	0.6691	0.2402	0.2822
Elastic Transform	0.7487	0.7269	0.6964	0.6813
Pixelate	0.5661	0.5690	0.3775	0.4139
JPEG Compression	0.7174	0.7057	0.6910	0.6725
Speckle Noise	0.4168	0.5465	0.2778	0.4435
Gaussian Blur	0.5914	0.6136	0.3811	0.4251
Spatter	0.7949	0.8122	0.6843	0.6933
Saturate	0.8646	0.8704	0.8119	0.8195

Table 21: Full CIFAR-10-C per-severity accuracies for CodedNN $n = 16$ versus the common softmax baseline, split into two portrait panels. Each severity group shows Softmax (S) / CodedNN (C).

Corruption	Severity 1		Severity 2		Severity 3	
	S	C	S	C	S	C
Gaussian Noise	0.7601	0.7690	0.5532	0.5567	0.3398	0.3477
Shot Noise	0.8353	0.8376	0.7455	0.7502	0.4950	0.4966
Impulse Noise	0.7926	0.8129	0.6534	0.6970	0.5209	0.5819
Defocus Blur	0.9095	0.9172	0.8905	0.9022	0.8333	0.8489
Glass Blur	0.4595	0.4474	0.4745	0.4554	0.5276	0.5231
Motion Blur	0.8557	0.8627	0.7709	0.7859	0.6763	0.6939
Zoom Blur	0.8218	0.8426	0.7896	0.8140	0.7335	0.7610
Snow	0.8238	0.8423	0.6907	0.7349	0.7269	0.7699
Frost	0.8560	0.8682	0.7928	0.8061	0.6888	0.7014
Fog	0.9041	0.9152	0.8834	0.9022	0.8529	0.8835
Brightness	0.9074	0.9154	0.8990	0.9109	0.8891	0.8998
Contrast	0.8987	0.9116	0.8397	0.8727	0.7724	0.8281
Elastic Transform	0.8547	0.8645	0.8522	0.8608	0.8084	0.8230
Pixelate	0.8700	0.8881	0.8006	0.8397	0.7453	0.7924
JPEG Compression	0.8221	0.8358	0.7664	0.7808	0.7494	0.7652
Speckle Noise	0.8328	0.8406	0.6923	0.7030	0.6014	0.6128
Gaussian Blur	0.9091	0.9172	0.8307	0.8473	0.7256	0.7471
Spatter	0.8676	0.8807	0.7975	0.8181	0.7389	0.7681
Saturate	0.8788	0.8919	0.8531	0.8667	0.9009	0.9095

Table 21 continued: Severities 4–5

Corruption	Severity 4		Severity 5	
	S	C	S	C
Gaussian Noise	0.2546	0.2770	0.2022	0.2261
Shot Noise	0.3847	0.3916	0.2477	0.2711
Impulse Noise	0.2399	0.3272	0.1269	0.1719
Defocus Blur	0.7246	0.7463	0.5339	0.5480
Glass Blur	0.3727	0.3474	0.4092	0.3935
Motion Blur	0.6752	0.6962	0.5891	0.6107
Zoom Blur	0.6735	0.7009	0.5798	0.6095
Snow	0.7075	0.7438	0.6537	0.6985
Frost	0.6794	0.6833	0.5834	0.5704
Fog	0.8178	0.8521	0.6756	0.7107
Brightness	0.8739	0.8901	0.8310	0.8626
Contrast	0.6409	0.7277	0.2402	0.3148
Elastic Transform	0.7487	0.7617	0.6964	0.7076
Pixelate	0.5661	0.6162	0.3775	0.4610
JPEG Compression	0.7174	0.7370	0.6910	0.7058
Speckle Noise	0.4168	0.4362	0.2778	0.3073
Gaussian Blur	0.5914	0.6156	0.3811	0.3860
Spatter	0.7949	0.8039	0.6843	0.6976
Saturate	0.8646	0.8652	0.8119	0.7972