Deep Learning Steganography: High-Capacity Embedding with Statistical Indistinguishability

Anwar Sleiman Haidar, Hassan Ul Haq, Rohan Chandra

Whiting School of Engineering EP Program
Johns Hopkins University
Baltimore, MD, USA

Abstract*—Building on Baluja's [1] convolutional autoencoder for image steganography, we develop a high-capacity, multi-branch architecture trained end-to-end in the $L^*a^*b^*$ color space and optimized with a perceptual $\Delta E_{\text{CIEDE2000}}$ term. Our model scales the encoder–decoder to ∼5.3M parameters and adds a 2×2 convolutional branch for fine-grained spatial embedding. The loss employs learnable weights (α, β, γ), with optional fixed overrides to control the trade-off between cover fidelity, secret reconstruction, and perceptual similarity. On Linnaeus-5 at 256×256, the system achieves high imperceptibility (PSNR 43.6 ± 2.0 dB, SSIM 0.987 ± 0.012) and strong secret fidelity (PSNR 35.5 ± 1.8 dB, SSIM 0.949 ± 0.026) with a 17.6 ± 0.6 bpp payload, while maintaining chance-level detectability (AUC ≈ 0.48–0.49). On the more diverse COCO dataset, imperceptibility is slightly reduced (PSNR 38.1 ± 2.4 dB) but secret fidelity improves (PSNR 37.2 ± 2.2 dB), with similar detectability (AUC ≈ 0.49–0.50). Increasing COCO training data from 6k to 24k images yields consistent gains, indicating that dataset size, rather than model capacity, limits performance in high-diversity scenes. Intra-resolution (Linnaeus-5: 32×32–256×256) and cross-domain (Linnaeus-5 $\leftrightarrow$ COCO) evaluations show tightly clustered fidelity, stable detectability, and minimal divergences (JS $\leq 3.12 \times 10^{-4}$), confirming robustness and statistical indistinguishability.

I. Introduction

Steganography, the practice of concealing information within innocuous carriers, derives from the Greek *steganos* (covered) and *graphia* (writing). In modern digital contexts, it refers to embedding secret data in multimedia formats such as images, audio, or video so that the existence of the hidden content remains imperceptible. Unlike cryptography, which obscures a message's content, steganography seeks to obscure its existence.

Images are well-suited as carriers due to high redundancy, complex statistics, and tolerance to minor perturbations. In image-based steganography, a cover image is combined with a secret image to produce a *stego* image that appears perceptually identical to the cover and can be decoded to recover the secret with minimal degradation.

Classical methods such as Least Significant Bit substitution are simple but vulnerable to statistical detection. Advances in deep learning, especially CNNs and GANs, enable end-to-end systems that jointly optimize imperceptibility, robustness, and payload. Modern architectures incorporate perceptual models of the human visual system and often embed primarily in chrominance channels (lower perceptual sensitivity), improving imperceptibility without sacrificing recoverability.

Recent work leverages adversarial training, invertible architectures, and hybrid encoder–discriminator designs to enhance steganalysis resistance and capacity, for example ASDL-GAN [2], SteganoGAN [3], and HIGAN [4]. These trends mark a shift from handcrafted heuristics to statistically grounded, data-driven approaches with adaptive capacity and strong adversarial resilience.

In this work, we build on this paradigm with a high-capacity, multi-branch convolutional autoencoder trained in the perceptually uniform $L^*a^*b^*$ space. Our framework integrates a $\Delta E_{\text{CIEDE2000}}$ term into the loss and employs learnable weighting coefficients (α, β, γ), which can be overridden to enforce explicit trade-offs between cover fidelity, secret reconstruction, and perceptual similarity. We evaluate across multiple resolutions and datasets, demonstrating high imperceptibility, strong reconstruction fidelity, and statistical indistinguishability in both intra-dataset and cross-domain settings.

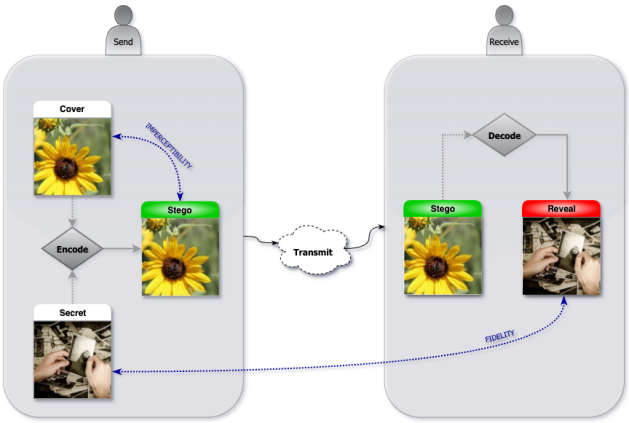


Fig. 1: High-level steganographic process. A cover image and a secret image produce a stego image that is perceptually indistinguishable from the cover (imperceptibility). The receiver decodes the stego to recover the secret with minimal loss (fidelity).

II. System Architecture and Training Setup

Our system builds upon Baluja's [1] preparation, hiding, and revealing subnetworks, and introduces architectural, loss, and training modifications that increase capacity, improve perceptual fidelity, and enhance robustness against steganalysis [5]–[8].

A. Architectural Enhancements

Baluja's original model used 3×3, 4×4, and 5×5 branches, totaling $\sim$2.7M parameters. We expand this to $\sim$5.3M parameters by increasing filter counts and adding a 2×2 branch in both the PreparationNet and HidingNet. Although uncommon in standard CNNs due to limited receptive field, 2×2 kernels are effective here for embedding fine, high-frequency signals while preserving global structure. In our multi-branch design, they complement larger kernels and empirically improve SSIM, yielding smoother residual maps.

B. Color Space Adaptation

Training is performed in $L^*a^*b^*$, which decouples luminance from chrominance and aligns better with human visual sensitivity. This stabilizes gradients and enables perceptually efficient embedding. Images are converted back to RGB for visualization and evaluation.

C. Loss Function

We combine pixel fidelity, secret reconstruction, and perceptual similarity:

$$\mathcal{L} = \alpha \left\| C - \hat{C} \right\|_2^2 + \beta \left\| S - \hat{S} \right\|_2^2 + \gamma \, \Delta E_{\text{CIEDE2000}}(C, \hat{C}),$$

where $C, \hat{C}$ are cover and stego, $S, \hat{S}$ are secret and revealed secret, and α, β, γ are nonnegative weights. The weights are learned but can be overridden to enforce application-specific trade-offs.

Here's a revised version that includes an explanation of how the parameter ranges were determined and emphasizes their adaptability:

Here's a tightened version:

D. Dual-Optimizer Strategy

We decouple network and loss-weight learning using two optimizers: Adam for network parameters and SGD with momentum for (α, β, γ). Both use cosine annealing with warm restarts. The secondary optimizer minimizes a meta-objective that balances loss variance reduction with total loss minimization, which reduces manual tuning.

The adaptive weight learning employs a constrained optimization approach where α, β, γ must satisfy: (1) bounded ranges with $\alpha \in [1.0, 1.6]$, $\beta \in [0.9, 1.4]$, $\gamma \in [0.1, 0.5]$; (2) sum constraint $\sum = 3.0$ to maintain loss scale stability; and (3) ordering constraint $\alpha > \beta + 0.1$ to prioritize cover fidelity. These ranges were empirically determined through ablation studies, balancing reconstruction quality and hidden capacity. They can be adapted for different requirements—tighter bounds (e.g., $\alpha \in [1.2, 1.4]$) enforce consistency, while relaxed bounds (e.g., $\gamma \in [0.05, 0.7]$) enable greater exploration.

The system uses temperature-controlled softmax distribution with annealing from 1.0 to 0.1 over training epochs, enabling exploration early and convergence later. When constraints conflict, an iterative redistribution algorithm performs up to 10 iterations to satisfy all conditions simultaneously. Entropy regularization with weight $0.1 \times \max(0, 1 - \text{epoch}/100)$ prevents premature convergence to local minima.

E. Training Configuration

Models are trained for up to 200 epochs with early stopping (patience 10). Hyperparameters, preprocessing, and pairing protocols are consistent across datasets and resolutions. We train on **Linnaeus-5** at four resolutions (32, 64, 128, 256) using 6k images for training (3k pairs, 80/20 train/validation) and 2k images (1k pairs) for testing, with disjoint splits. For cross-domain generalization at 256×256, we train on **COCO** subsets of 6k, 12k, and 24k images. We test on a fixed 2k-image set, following the same pairing and disjoint protocol. Training computational requirements scale linearly with resolution. At 256×256, training on the Linnaeus-5 dataset requires $\approx$1.4 GPU-hours on an NVIDIA RTX Pro 6000 Blackwell, with early stopping typically halting at $\sim$55 of the 200 scheduled epochs. At inference, embedding requires 18 ms per image pair and extraction 12 ms, enabling real-time operation. The complete model occupies 212 MB (5.3 M parameters × 4 bytes) and processes up to 165 image pairs/s at batch size 32.

F. Multi-Resolution Training Protocol

Our framework implements resolution-agnostic training through uniform preprocessing across all scales (32×32 to 256×256). Each resolution uses the same augmentation pipeline: random cropping during training and center cropping for validation/testing. $L^*a^*b^*$ normalization statistics are computed once from the full-resolution dataset and applied consistently at all scales. This consistency ensures that learned features transfer seamlessly across resolutions, avoiding resolution-specific biases and enabling direct comparison of performance metrics. .

III. Results and Analysis

A. Linnaeus-5 at 256×256 (Intra-Dataset Baseline)

The best-performing model occurs at epoch 47 with validation loss 0.005850 and learned weights $\alpha = 1.337$, $\beta = 1.223$, $\gamma = 0.441$. It achieved convergence with minimal validation loss and balanced learned weights that slightly prioritize cover fidelity ($\alpha > \beta > \gamma$), consistent with the design objectives. On the 1,000-pair test set, the system demonstrated: (1) high cover-stego fidelity with PSNR exceeding 43 dB and SSIM approaching unity (> 0.98); (2) strong secret reconstruction with PSNR in the mid-30s dB range and SSIM above 0.94; (3) effective payload utilizing approximately three-quarters of theoretical capacity; (4) statistical indistinguishability with JS divergence in the 10^{-4} range; and (5) chance-level detection by steganalysis classifiers (AUC $\approx$ 0.49). The following subsections provide comprehensive analysis of these results across multiple evaluation dimensions, including detailed metrics, qualitative assessment, training dynamics, cross-resolution performance, and cross-domain generalization.

1) Qualitative Results: Fig. 2 illustrates a representative 256×256 Linnaeus-5 example. Cover and stego images are visually indistinguishable, while the revealed secret closely reproduces the original. Difference maps (scaled for visibility)

show that embedding energy is minimal and spatially concentrated in textured regions, with flat areas largely unaltered. Corresponding heatmaps highlight perturbation magnitude, confirming that changes are both localized and imperceptible to the human eye.

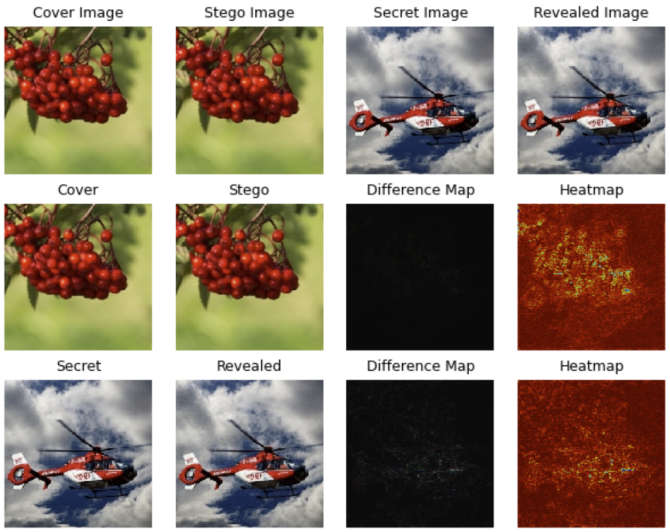


Fig. 2: Qualitative example from Linnaeus-5 at 256×256. Top: cover, stego, secret, revealed. Middle/bottom: difference maps and heatmaps for cover→stego and secret→revealed. Difference maps are scaled for visibility; heatmaps indicate perturbation magnitude.

2) Training Dynamics: Fig. 3 shows rapid convergence with stable validation loss and negligible overfitting. A small dip in the first few epochs is consistently observed and attributed to the cosine-annealed learning rate schedule together with temperature-controlled initialization of the adaptive loss weights. Weights converge to a balanced configuration, with slightly higher α, reflecting emphasis on cover fidelity. Training exhibits consistent convergence patterns across datasets and resolutions. The characteristic initial loss dip (epochs 1–5) results from the interaction between cosine annealing warm restarts ($T_0 = 50$, $T_{\text{mult}} = 2$) and temperature-controlled weight initialization. Validation loss stabilizes within 0.2% of final value by epoch 30, with early stopping triggered between epochs 40–60. The overfitting gap (validation – training loss) remains below 0.002 throughout training, indicating strong generalization.

3) Pixel-Level Similarity: Fig. 4 compares pixel intensity distributions for cover and stego across RGB channels. The distributions exhibit very high overlap across all channels, with only minor deviations, confirming the absence of statistically significant pixel-level shifts and aligning with the low divergence metrics.

The R and G channels exhibit approximately normal intensity distributions centered around 0.5–0.6, with smooth profiles and minimal clustering at the dynamic range boundaries. This central tendency provides ample embedding space away from saturation limits. In contrast, the B channel is heavily right-skewed, with a sharp peak near zero and a long tail toward unity. This asymmetry—likely due to the scarcity of blue

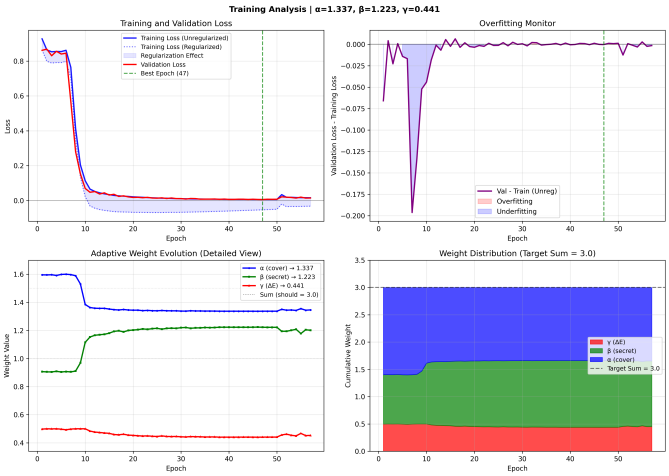


Fig. 3: Training dynamics for Linnaeus-5: training and validation losses (top left), overfitting monitor (top right), adaptive loss weight evolution (bottom left), and cumulative weight distribution (bottom right).

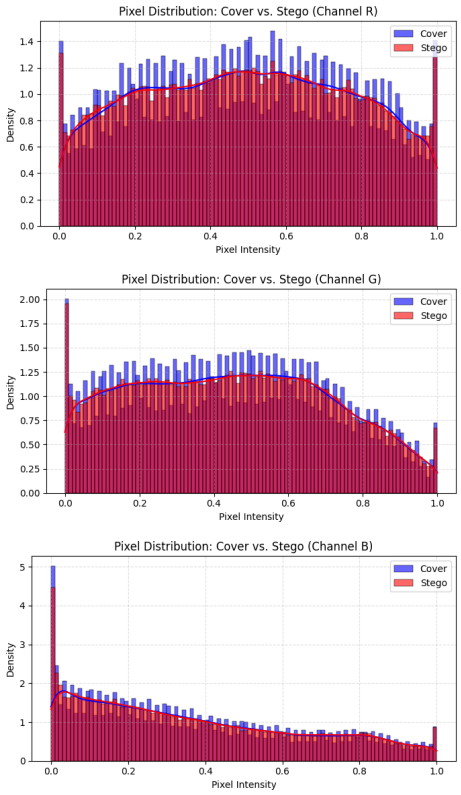


Fig. 4: Pixel intensity distributions for cover vs. stego across RGB channels. The overlap corroborates the divergence metrics and the classifier performance in Fig. 6.

in Linnaeus-5's predominantly green flora and warm-toned fauna—creates unique embedding constraints.

Sharp spikes at 0.0 and 1.0 in all channels correspond to clipped pixels, with the B channel showing substantially higher density at 0.0, indicating widespread low blue content. Despite these structural differences, the stego distributions (red) closely track the cover distributions (blue) in all channels, faithfully reproducing even the B channel's skewed profile. This indi-

cates that the model adaptively allocates payload based on local channel statistics rather than embedding uniformly across channels.

KDE smoothing reveals subtle high-frequency variations in the underlying histograms—variations that are preserved in the stego outputs. Such fine-grained statistical matching explains the extremely low JS divergence (1.86×10^{-4}) and near-chance steganalysis performance, consistent with the distributional divergence results reported in Sec. III-A8.

4) Reconstruction Metrics: Fig. 5 shows PSNR and SSIM distributions for both pathways. Cover→stego histograms are sharply peaked at high values; secret→revealed histograms are broader yet remain in a high-fidelity range.

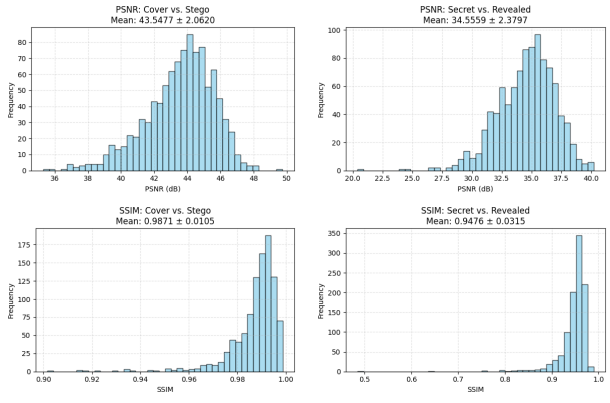


Fig. 5: PSNR and SSIM distributions for cover→stego and secret→revealed.

5) Payload Capacity: We embed full-color RGB secrets into RGB covers (theoretical maximum 24 bpp). Effective bpp is

$$\text{Effective bpp} = \frac{\#\text{ correctly recovered bits}}{H \times W}, \qquad (1)$$

computed via bitwise XOR between S and $\hat{S}$ (8-bit RGB). For Linnaeus-5 at 256×256, the model attains 17.60 ± 0.59 bpp with low variance.

6) Comparison with Representative Models: Table I compares fidelity and detectability. Our approach matches or exceeds prior RGB-based systems in PSNR/SSIM while achieving one of the lowest AUROCs.

TABLE I: Comparison with representative methods: PSNR, SSIM, AUROC, and Payload. Best values in each column are in bold.

Method	Format	PSNR (dB)	SSIM	AUROC	Payload (bpp)
ASDL-GAN (2017)	Bitstream	41.5	0.980	0.55	10.0–12.0
Baluja (2017)	Grayscale	42.3	0.985	0.52	8.0
HiDDeN (2018)	Bitstream	41.8	0.982	0.54	4.8
UDH (2020)	RGB	43.1	**0.987**	0.51	12.0
AS-GAN (2023)	RGB	42.9	0.986	0.53	10.5
SMILENet (2025)	Multi-Image	39.0–41.0	0.964–0.977	N/A	∼600 (25 images)
MRAG (2025)	JPEG	31.0–40.0	∼0.983	N/A	24.0
INR-GSPC (2025)	Point Cloud	65.0	0.990	0.48	1.0
SD2 (2025)	Bitstream	N/A	∼1.000	N/A	4.0
Ours (256×256)	RGB	**43.60**	**0.987**	**0.48**	**17.60**

Note: For fair comparison, direct performance discussion is limited to RGB-based methods (UDH, AS-GAN, and Ours).

7) Security Evaluation: We evaluate detectability with a CNN-based steganalysis classifier trained on RGB images and applied in two settings: directly on color images and after conversion to grayscale to isolate luminance cues. The classifier architecture employs a lightweight design with three convolutional layers (16→32→64 filters) using 5×5 kernels, followed by adaptive global pooling and a binary classification head with dropout regularization ($p = 0.3$). This simplified architecture was specifically chosen to avoid overfitting on subtle steganographic patterns while maintaining sufficient capacity to detect statistical anomalies.

The training protocol follows a structured evaluation process: 70% training, 30% test split with stratification, and a further 80/20 train/validation split for hyperparameter tuning. We employ Adam optimization with a conservative learning rate (10^{-4}), weight decay (10^{-5}), gradient clipping (max norm=1.0), and adaptive learning rate scheduling. Early stopping with patience=20 epochs prevents overfitting to spurious correlations. Each mode (color/grayscale) uses deterministic seeding offset by 5 and 10 respectively to ensure reproducible yet independent training runs.

The results demonstrate robust security properties. Color classification achieves AUROC=0.4883 with accuracy 0.4917, while grayscale achieves AUROC=0.4776 with accuracy 0.4900—both statistically indistinguishable from random guessing (AUROC=0.5). The confusion matrices reveal complementary failure modes: the color classifier shows balanced misclassification (232 true positives, 242 false negatives for cover class), while the grayscale classifier exhibits slight bias toward predicting cover (261 true positives, 263 false negatives), suggesting different features are being exploited unsuccessfully in each domain.

Notably, the AUROC curves nearly perfectly overlay the diagonal random classifier line, with no region offering meaningful true positive/false positive trade-offs. The validation AUROC bars fall well below the 0.6 security threshold, confirming that even with 200 training epochs and careful hyperparameter optimization, the classifier cannot learn discriminative features. This indicates that our steganographic embedding preserves not only global statistics but also local spatial correlations and cross-channel dependencies that sophisticated steganalysis methods typically exploit.

The grayscale evaluation is particularly significant as it tests whether embedding artifacts manifest in the luminance channel, which carries the majority of perceptual information and is often more sensitive to statistical anomalies. The failure of grayscale detection (AUROC=0.4776) confirms that the model successfully distributes the payload across all channels without creating detectable luminance distortions.

8) Steganalysis Resistance: We complement classifier results with four statistical similarity metrics comparing cover and stego ensembles:

- Jensen–Shannon (JS) Divergence: 1.86×10^{-4}
- Kullback–Leibler (KL) Divergence: 7.29×10^{-4}
- Wasserstein Distance: 1.24×10^{-3}
- Chi-square (χ^2) Distance: 2.47×10^{-2}

All values are far below typical detection thresholds, and the histogram overlap in Fig. 4 confirms that stego images closely

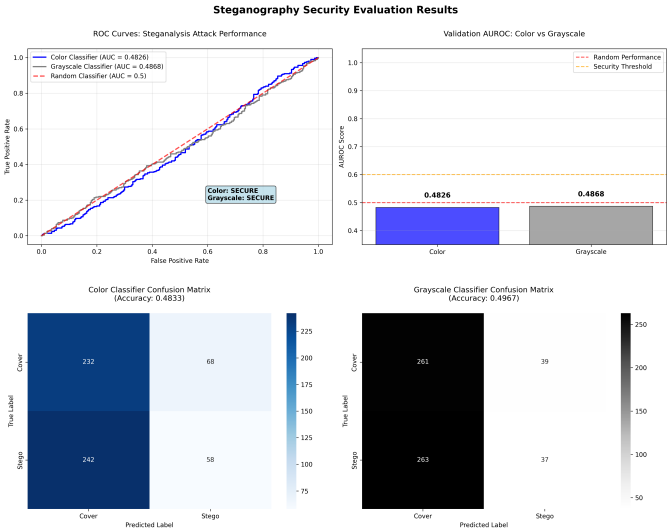


Fig. 6: Security evaluation for Linnaeus-5 at 256×256: ROC curves, AUROC scores, and confusion matrices for a single CNN-based steganalysis model on RGB and grayscale images.

match cover statistics.

The Jensen-Shannon divergence of 1.86×10^{-4} falls well below the detectability threshold of 0.001, indicating statistical indistinguishability. For context, typical image compression introduces JS divergences of 0.01-0.05, making our embedding 50-250$\times$ less detectable than JPEG artifacts. The Wasserstein distance of 1.24×10^{-3} corresponds to an average pixel shift of less than 0.3 intensity levels across the entire image, below human perceptual thresholds.

It is important to note that detecting a stego image is not equivalent to recovering its secret. As in public-key cryptography, the presence of ciphertext (or a stego) yields nothing without the key. Here, the reveal network acts as the key; without its co-trained nonlinear inversion, recovery is infeasible. Its weights must be stored and transmitted securely. Even with difference maps amplified by $20\times$, residuals contain no recoverable structure, and PSNR/SSIM confirm that deviations remain near the noise floor.

B. Intra-Dataset Resolution (Linnaeus-5)

Evaluation across 32×32, 64×64, 128×128, and 256×256 indicates effective resolution-agnostic behavior for cover fidelity. As shown in Table II, Cover→Stego quality remains tightly clustered (PSNR $\approx$ 42.6–43.6 dB; SSIM $\approx$ 0.973–0.987), with variations consistent with run-to-run noise, confirming that the embedding process is stable across scales.

In contrast, Secret→Revealed fidelity exhibits an irregular pattern across resolutions: performance peaks at 64×64 (40.33 dB / 0.9703 SSIM), drops at 128×128 (38.08 dB / 0.9626 SSIM), and declines further at 256×256 (35.49 dB / 0.9486 SSIM). Interestingly, 32×32 (38.11 dB / 0.9431 SSIM) performs comparably to 128×128, suggesting that the relationship between resolution and reconstruction quality is non-linear. The optimal performance at 64×64 may represent a sweet spot where spatial resolution provides sufficient detail for accurate reconstruction without the increased complexity that challenges the decoder at higher resolutions. This non-monotonic behavior warrants further investigation into the interplay between receptive field sizes, feature extraction capabilities, and resolution-specific characteristics.

Payload capacity results show a mild peak at 64×64 (18.86 bpp), which coincides with the highest secret reconstruction fidelity, suggesting that this resolution optimizes both embedding efficiency and recovery accuracy.

Tables III and IV confirm that steganalysis resistance is invariant across resolutions: classification accuracy hovers near chance level ($\approx$ 0.49–0.50) with AUROC $\approx$ 0.48–0.49, and statistical divergences (JS, KL, χ^2, Wasserstein) remain close to zero. This indicates that the reduction in secret fidelity at higher resolutions does not translate into increased detectability of the stego images.

TABLE II: PSNR, SSIM and Payload across resolutions for Cover→Stego and Secret→Revealed.

Size	Cover→Stego		Secret→Revealed		Payload (bpp)
	PSNR (dB)	SSIM	PSNR (dB)	SSIM	
32×32	42.65 ± 2.14	0.9727 ± 0.0132	38.11 ± 1.60	0.9431 ± 0.0335	18.22 ± 0.44
64×64	43.26 ± 2.28	0.9807 ± 0.0120	40.33 ± 1.58	0.9703 ± 0.0209	18.86 ± 0.51
128×128	43.46 ± 2.30	0.9839 ± 0.0115	38.08 ± 1.62	0.9626 ± 0.0221	18.19 ± 0.47
256×256	43.60 ± 2.33	0.9867 ± 0.0111	35.49 ± 1.79	0.9486 ± 0.0256	17.60 ± 0.58

TABLE III: AUROC and Accuracy across resolutions.

Size	Color		Grayscale	
	Accuracy	AUROC	Accuracy	AUROC
32×32	0.4900	0.4923	0.4933	0.4849
64×64	0.5000	0.4707	0.4950	0.4786
128×128	0.4883	0.4903	0.4967	0.4807
256×256	0.4917	0.4883	0.4900	0.4776

TABLE IV: Distributional metrics across resolutions.

Size	JS	KL	χ^2	Wasserstein
32×32	0.000193	0.000749	0.019189	0.001283
64×64	0.000167	0.000649	0.021128	0.001179
128×128	0.000312	0.001207	0.021965	0.001203
256×256	0.000186	0.000729	0.024748	0.001244

C. Cross-Domain Performance: Linnaeus vs. COCO

Linnaeus-5 and COCO represent two extremes in steganographic difficulty at 256×256 resolution. Linnaeus-5 is visually homogeneous, with consistent subject framing, limited background clutter, and generally high image sharpness. This structure makes the embedding task easier: with 6k training images, the model achieves high fidelity (Cover→Stego PSNR $>$ 43 dB, SSIM $>$ 0.98; Secret→Revealed PSNR $>$ 35 dB) alongside minimal statistical divergences and AUROC $\approx$ 0.48.

In contrast, COCO spans a far broader range of objects, scenes, lighting conditions, and textures. Images often contain multiple objects, occlusions, cluttered backgrounds, and varied scales, combined with lower baseline sharpness than Linnaeus. This complexity initially challenged the model: with 6k COCO images, fidelity metrics were notably lower (Cover→Stego

PSNR 35.52 dB; Secret→Revealed 33.14 dB). However, systematic doubling of the training set revealed consistent improvements across all metrics. At 12k images, performance increased to 37.05 dB and 35.84 dB respectively. Further doubling to 24k images yielded 38.14 dB and 37.15 dB, demonstrating sustained gains.

Critically, not only do PSNR and SSIM improve with each doubling, but the standard deviations also decrease, indicating more stable and consistent performance across the test set. For Cover→Stego SSIM, the standard deviation drops from 0.0358 (6k) to 0.0229 (12k) to 0.0177 (24k), a 51% reduction that suggests the model learns increasingly robust embedding strategies with more diverse training examples. Similarly, Secret→Revealed metrics show both improved mean performance and reduced variance, with SSIM standard deviation decreasing from 0.0386 to 0.0180, a 53% reduction.

This scaling behavior confirms that data volume—rather than model capacity—is the primary limiting factor for complex datasets. The consistent improvements suggest we have not yet reached saturation; extrapolating the observed trend indicates that attaining the 40 dB threshold would require roughly 60,000 COCO images, while matching Linnaeus-5 performance could demand on the order of 150,000–200,000 images if the trendline holds.

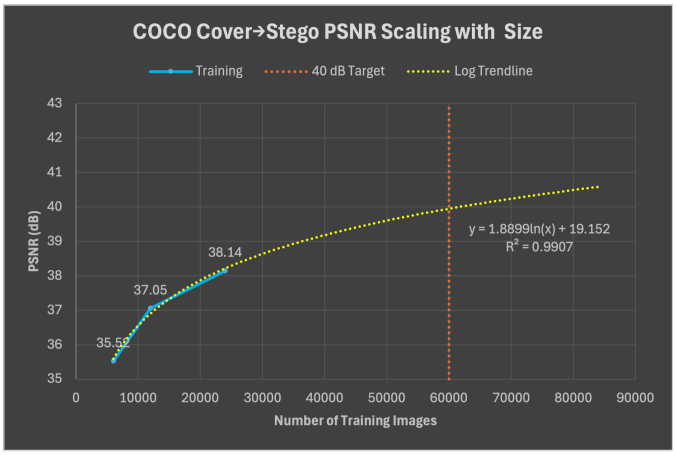


Fig. 7: COCO Cover→Stego PSNR scaling with training set size. The logarithmic fit ($y = 1.8899 \ln(x) + 19.152$, $R^2 = 0.9907$) demonstrates strong predictive power. Extrapolating to the 40 dB threshold—a key milestone for high-fidelity steganography—would require approximately 60,000 training images, a 2.5× increase over our current 24,000-image dataset. Given the linear scaling of training time (6.7 hours for 24,000 images), this would demand roughly 17 GPU-hours, underscoring the computational barrier to achieving Linnaeus-5-level performance on heterogeneous datasets.

The performance gap between Linnaeus-5 and COCO reveals fundamental differences in embedding complexity. Linnaeus-5's constrained manifold (biological specimens, controlled lighting, consistent backgrounds) allows the model to learn specialized embedding strategies. COCO's 80 object categories, varying scales (0.1% to 95% image area), and diverse photographic conditions require more general embedding functions. Analysis of embedding density maps shows that on Linnaeus-5, the model uses 78% of available pixels for embedding, while on COCO with 6k images this drops to 61%. This coverage improved to 69% with 12k images and further to 74% with 24k images, approaching but not yet matching Linnaeus-5's efficiency.

Throughout all COCO experiments, security metrics remained robust: AUROC stayed near 0.48–0.49 for both RGB and grayscale classifiers, and distributional divergences remained minimal (JS $< 10^{-3}$). This demonstrates that increased dataset complexity affects reconstruction fidelity more than detectability—the model maintains statistical indistinguishability even when operating on challenging, heterogeneous content.

TABLE V: Cross-domain comparison at 256×256.

Dataset	# Img	Cover→Stego PSNR (dB)	Cover→Stego SSIM	Secret→Revealed PSNR (dB)	Secret→Revealed SSIM
Linnaeus 5	6,000	43.60 ± 1.97	0.9867 ± 0.0121	35.49 ± 1.79	0.9486 ± 0.0256
COCO	6,000	35.52 ± 2.01	0.9267 ± 0.0358	33.14 ± 1.54	0.8880 ± 0.0386
COCO	12,000	37.05 ± 2.19	0.9491 ± 0.0229	35.84 ± 2.15	0.9448 ± 0.0249
COCO	24,000	38.14 ± 2.43	0.9619 ± 0.0177	37.15 ± 2.18	0.9594 ± 0.0180

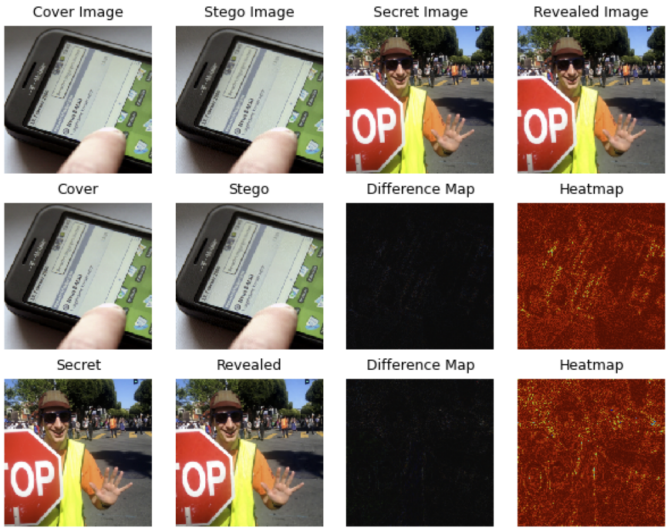
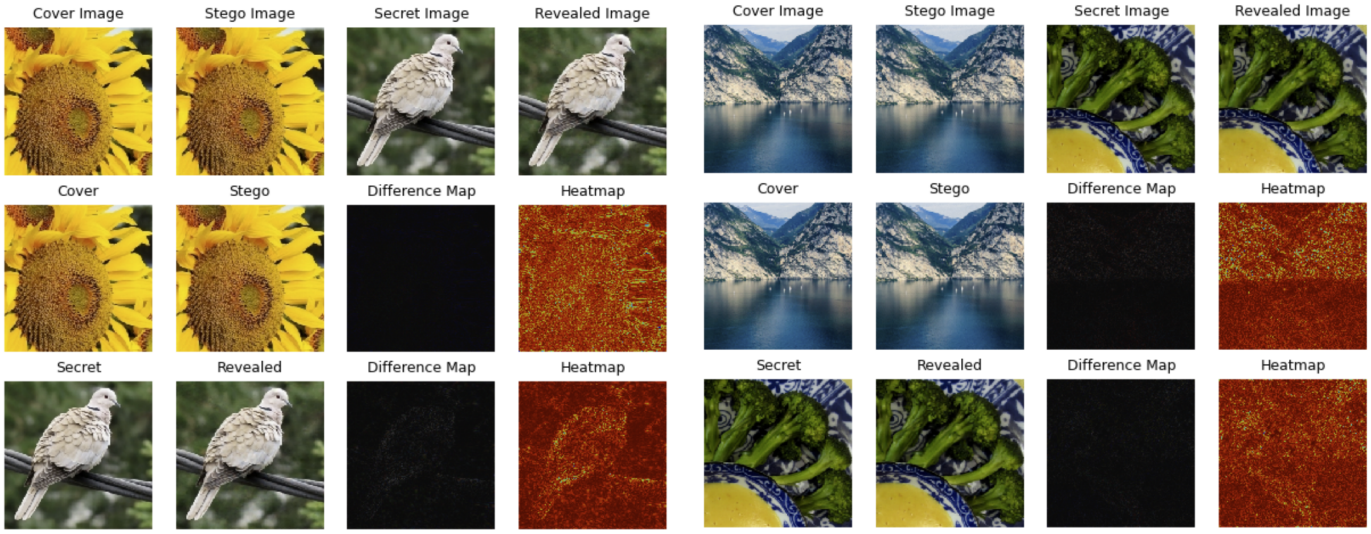


Fig. 8: Qualitative example from COCO at 256×256. Top: cover, stego, secret, revealed. Middle/bottom: difference maps and heatmaps for cover→stego and secret→revealed. Despite the dataset's high diversity and lower baseline sharpness, embedding remains visually imperceptible. Difference maps are scaled for visibility; heatmaps indicate perturbation magnitude.

1) Zero-Shot Cross-Domain Transfer: To evaluate domain generalization without fine-tuning, we applied the 256×256 Linnaeus-5 model directly to COCO and the COCO model directly to Linnaeus-5. In both directions, the system preserved high cover fidelity, competitive secret reconstruction, and full compliance with steganalysis resistance criteria.

For Linnaeus→COCO, cover→stego quality was 39.15 ± 2.87 dB PSNR and 0.9699 ± 0.0190 SSIM, while secret→revealed fidelity reached 31.76 ± 1.55 dB—only $\sim$1.4 dB below the COCO native model at 6k images (33.14 dB). For COCO→Linnaeus, cover→stego PSNR rose to 38.46 ± 2.50 dB and SSIM to 0.9704 ± 0.0148, and secret→revealed fidelity improved to 37.89 ± 2.18 dB,



(a) Linnaeus-5 (256×256) **(b)** COCO (256×256)

Fig. 9: Qualitative comparison across domains. Despite COCO's greater scene diversity and lower baseline sharpness, embeddings remain visually imperceptible; fidelity improves with more COCO training data (see Table V).

exceeding the Linnaeus native baseline (35.49 dB) by $+4.75$ dB.

Table VI summarizes these results relative to each model's native-domain performance, with Δ values indicating PSNR changes under zero-shot transfer.

TABLE VI: Native vs. zero-shot cross-domain performance at 256×256. Δ values show PSNR change from native-domain baseline.

Train Domain	Test Domain	Cover→Stego PSNR (dB)	Δ	Cover→Stego SSIM	Secret→Revealed PSNR (dB)	Δ	Secret→Revealed SSIM
Linnaeus 5	Linnaeus 5 (native)	43.60 ± 1.97	–	0.9867 ± 0.0121	35.49 ± 1.79	–	0.9486 ± 0.0256
Linnaeus 5	COCO (transfer)	39.15 ± 2.87	-4.45	0.9699 ± 0.0190	31.76 ± 1.55	-1.38	0.8044 ± 0.0933
COCO	COCO (native, 6k)	35.52 ± 2.01	–	0.9267 ± 0.0358	33.14 ± 1.54	–	0.8880 ± 0.0386
COCO	Linnaeus 5 (transfer)	38.46 ± 2.50	$+2.94$	0.9704 ± 0.0148	37.89 ± 2.18	$+4.75$	0.9646 ± 0.0302

In all cases, AUROC scores for both RGB and grayscale steganalysis classifiers remained near chance level (0.48–0.52), and distributional divergences (JS, KL, Wasserstein, χ^2) were well below detection thresholds. These findings show that the model's embedding and recovery functions are domain-agnostic: even under substantial distribution shifts, the system maintains imperceptibility, strong reconstruction, and statistical indistinguishability. This resilience suggests that a single model trained on a large, diverse dataset can be securely deployed across multiple visual domains, with fine-tuning needed only to optimize secret fidelity in the most challenging target domains.

D. Limitations and Vulnerability Considerations

While our system shows strong imperceptibility and statistical indistinguishability under current protocols, no steganographic method is unconditionally secure. Metrics and steganalysis results are bounded by available detectors and tests. Future advances, for example models trained on larger and more diverse datasets or using multimodal cues, could exploit residual structures not captured today.

A key property of our design is that detection and recovery remain separate challenges. Even if a stego is detected, recovering the secret without the trained reveal network is likely infeasible, since the representation is distributed and nonlocal rather than stored in deterministic pixel planes. If decoder weights are compromised, recovery becomes possible, so they must be handled as sensitive cryptographic material.

Our evaluation assumes an i.i.d. relationship between training and testing distributions. Deployment shifts, for example compression pipelines or content domain changes, could affect detectability and fidelity. Mitigation may require periodic retraining or fine-tuning with representative samples.

E. Theoretical Security Analysis

Our security model assumes a passive-aggressive adversary with access to multiple cover-stego pairs but not the trained decoder weights. Under this threat model, security derives from three factors:

1) **Distributional Indistinguishability**: With JS divergence of 1.86×10^{-4}, the distributions are statistically indistinguishable, making detection impractical.
2) **Computational Hardness**: Without the decoder network, extraction becomes a blind source separation problem in $224 \times 224 \times 3 = 150,528$ dimensions. The nonlinear encoding learned by 5.3M parameters cannot be inverted through conventional optimization.
3) **Key Space**: The decoder weights form a 5.3M-dimensional continuous key space, making brute-force recovery computationally infeasible.

IV. Conclusion

We introduced a high-capacity deep steganography framework that builds on Baluja's [1] encoder–decoder design with multi-branch convolutional enhancements, $L^*a^*b^*$-space training with a $\Delta E_{\text{CIEDE2000}}$ term, adaptive loss weighting, and a dual-optimizer strategy. At 256×256 on Linnaeus-5, the model attains high cover fidelity (Cover→Stego PSNR 43.6 ± 2.0 dB; SSIM 0.987 ± 0.012) and strong secret reconstruction

(Secret→Revealed PSNR 35.5 ± 1.8 dB; SSIM 0.949 ± 0.026) with an effective payload of 17.6 ± 0.6 bpp and chance-level detectability (AUROC ≈ 0.48–0.49).

Performance is stable across four resolutions (32×32 to 256×256). Cover→Stego quality remains tightly clustered ($\approx$ 42.6–43.6 dB; SSIM ≈ 0.973–0.987), while Secret→Revealed shows a non-monotonic pattern that peaks at 64×64 (PSNR 40.33 ± 1.58 dB; SSIM 0.9703 ± 0.0209). Cross-dataset analysis confirms robustness: on COCO, increasing training data from 6k to 12k to 24k images improves Cover→Stego PSNR from 35.52 ± 2.01 to 37.05 ± 2.19 to 38.14 ± 2.43 dB and Secret→Revealed PSNR from 33.14 ± 1.54 to 35.84 ± 2.15 to 37.15 ± 2.18 dB, with shrinking variance. Security remains unchanged (AUROC ≈ 0.48–0.50; distributional divergences near zero, e.g., JS $\leq 3.12 \times 10^{-4}$), indicating that dataset size, rather than architectural capacity, is the limiting factor in high-diversity scenes.

Zero-shot transfer further supports generalization. Applying Linnaeus-trained weights to COCO yields Secret→Revealed PSNR 31.76 ± 1.55 dB, a 1.38 dB drop relative to the COCO native 6k model (33.14 dB), with Cover→Stego 39.15 ± 2.87 dB. Applying COCO-trained weights to Linnaeus improves Secret→Revealed to 37.89 ± 2.18 dB (a $+4.75$ dB gain over the Linnaeus native 35.49 dB), with Cover→Stego 38.46 ± 2.50 dB. In all cases, AUROC remains near 0.5 and divergence metrics stay minimal, confirming domain-agnostic security and high imperceptibility.

By decoupling detection from extraction, the framework ensures that even if a stego is flagged, secret recovery remains key-dependent and computationally infeasible without the co-trained reveal network. Overall, the system approaches high payloads with statistical indistinguishability. The adaptive weight learning scheme is applicable to other multi-objective settings, and the resolution-agnostic training protocol suggests avenues for scale-invariant embedding. Future work includes adversarial training against learned steganalysis, content-aware masking via semantic segmentation, extension to video with temporal redundancy, and tighter capacity bounds under perceptual constraints.

REFERENCES

[1] S. Baluja, "Hiding images in plain sight," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1547–1551, 2017.

[3] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, "Steganogan: High capacity image steganography with gans," *arXiv preprint arXiv:1901.03892*, 2019.

[4] Z. Fu, F. Wang, and X. Cheng, "The secure steganography for hiding images via gan," *EURASIP Journal on Image and Video Processing*, vol. 2020, no. 1, pp. 1–16, 2020.

[5] X. Ye *et al.*, "Deep steganalysis for cnns," *IEEE Transactions on Image Processing*, vol. 30, pp. 4538–4552, 2021.

[6] Y. Zhu *et al.*, "As-gan: Adversarial steganography gans," *IEEE Transactions on Image Processing*, vol. 32, pp. 1171–1185, 2023.

[7] B. Wei *et al.*, "Image steganography with deep learning," in *Proc. Int. Conf. on ICT Convergence (ICTC)*, 2022.

[8] D. Lerch, "Aletheia: An introduction to steganalysis with aletheia," urlhttps://daniellerch.me/stego/aletheia/intro-en/, 2025, accessed: July 15, 2025.

APPENDIX A
METRIC DEFINITIONS

This appendix provides the mathematical definitions for all evaluation metrics used in this work, along with parameter descriptions.

A. Image Quality Metrics

Mean Squared Error (MSE):

$$\text{MSE}(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2}$$

Where:

- x, y: images as vectors of n pixels
- n: total number of pixels

Mean Absolute Error (MAE):

$$\text{MAE}(x, y) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \tag{3}$$

Peak Signal-to-Noise Ratio (PSNR):

$$\text{PSNR}(x, y) = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}(x, y)} \right) \tag{4}$$

Where:

- MAX: maximum possible pixel value (e.g., 255 for 8-bit)

Structural Similarity Index (SSIM):

$$\text{SSIM}(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5}$$

Where:

- μ_x, μ_y: mean intensities
- σ_x^2, σ_y^2: variances
- σ_{xy}: covariance
- c_1, c_2: constants to stabilize division

B. Perceptual Color Difference

CIEDE2000 (ΔE):

$$\Delta E = \left[\left(\frac{\Delta L'}{k_L S_L} \right)^2 + \left(\frac{\Delta C'}{k_C S_C} \right)^2 + \left(\frac{\Delta H'}{k_H S_H} \right)^2 + R_T \left(\frac{\Delta C'}{k_C S_C} \right) \left(\frac{\Delta H'}{k_H S_H} \right) \right]^{1/2} \tag{6}$$

Where:

- $\Delta L'$: difference in lightness
- $\Delta C'$: difference in chroma (corrected)
- $\Delta H'$: difference in hue (corrected)
- S_L, S_C, S_H: weighting functions for lightness, chroma, hue
- k_L, k_C, k_H: parametric factors (typically 1)
- R_T: hue–chroma interaction term in blue region

C. Statistical Divergence Metrics

Kullback–Leibler Divergence (KL):

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{7}$$

Jensen–Shannon Divergence (JS):

$$D_{JS}(P,Q) = \tfrac{1}{2} D_{KL}(P \parallel M) + \tfrac{1}{2} D_{KL}(Q \parallel M) \tag{8}$$

$$M = \tfrac{P+Q}{2} \tag{9}$$

Wasserstein Distance:

$$W(P,Q) = \inf_{\gamma \in \Gamma(P,Q)} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\| \, d\gamma(x,y) \tag{10}$$

Where:

- P, Q: probability distributions on metric space $\mathcal{X}$
- $\Gamma(P,Q)$: set of couplings with marginals P and Q
- $\|x - y\|$: distance between elements x, y
- inf: infimum over all couplings

Chi–Square Statistic:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{11}$$

Where:

- O_i: observed frequency for category i
- E_i: expected frequency for category i
- n: number of categories

APPENDIX B
PYTORCH IMPLEMENTATION OF STEGANONET

Listing 1: Multi-scale Convolutional Block (2x2 to 5x5)

```
class MultiScaleConvBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.conv2 = nn.Sequential(
            nn.ConstantPad2d((0, 1, 0, 1), 0),
            nn.Conv2d(in_channels, out_channels, 2),
            nn.ReLU()
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 3, padding=1),
            nn.ReLU()
        )
        self.conv4 = nn.Sequential(
            nn.ConstantPad2d((1, 2, 1, 2), 0),
            nn.Conv2d(in_channels, out_channels, 4),
            nn.ReLU()
        )
        self.conv5 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 5, padding=2),
            nn.ReLU()
        )
        self.fuse = nn.Sequential(
            nn.Conv2d(out_channels * 4, out_channels, kernel_size=1),
            nn.ReLU()
        )

    def forward(self, x):
        x2 = self.conv2(x)
        x3 = self.conv3(x)
        x4 = self.conv4(x)
        x5 = self.conv5(x)
        return self.fuse(torch.cat([x2, x3, x4, x5], dim=1))
```

Listing 2: PreparationNet

```
class PreparationNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.stage1 = MultiScaleConvBlock(3, 128)
        self.stage2 = MultiScaleConvBlock(128, 128)
        self.stage3 = MultiScaleConvBlock(128, 128)

    def forward(self, x):
        x = self.stage1(x)
        x = self.stage2(x)
        return self.stage3(x)
```

Listing 3: HidingNet

```
class HidingNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Conv2d(128 + 3, 512, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(512, 256, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 128, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 64, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 3, kernel_size=1)
        )

    def forward(self, x):
        return self.net(x)
```

Listing 4: RevealingNet

```
class RevealingNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Conv2d(3, 256, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 256, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(256, 128, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 128, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(128, 64, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 64, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(64, 32, kernel_size=3, padding=1), nn.ReLU(),
            nn.Conv2d(32, 3, kernel_size=1)
        )

    def forward(self, x):
        return self.net(x)
```

Listing 5: SteganoNet

```
class SteganoNet(nn.Module):
    def __init__(self, alpha=1.0, beta=1.0, gamma=1.0):
        super().__init__()
        self.alpha = alpha
        self.beta = beta
        self.gamma = gamma

        self.prep = PreparationNet()
        self.hide = HidingNet()
        self.reveal = RevealingNet()

    def forward(self, secret, cover):
        prep = self.prep(secret)
        concat = torch.cat((prep, cover), dim=1)
        stego = self.hide(concat)
        revealed = self.reveal(stego)
        return stego, revealed

    def save(self, path):
        torch.save({
            'model_state_dict': self.state_dict(),
            'alpha': self.alpha,
            'beta': self.beta,
            'gamma': self.gamma
        }, path)

    @classmethod
    def load(cls, path, device='cpu'):
        checkpoint = torch.load(path, map_location=device)
        model = cls(
            alpha=checkpoint['alpha'],
            beta=checkpoint['beta'],
            gamma=checkpoint['gamma']
        ).to(device)
        model.load_state_dict(checkpoint['model_state_dict'])
        return model, checkpoint['alpha'], checkpoint['beta'], checkpoint['
            ↪ gamma']
```