A Systems-Level Blueprint for Robust, Lossless Audio-in-Image Steganography

This report provides a comprehensive architectural and methodological blueprint for the advanced-level, three-model AI project specified in the query. The project's objectives—encompassing neural audio enhancement, high-capacity robust steganography, and perfect-fidelity data reversal—represent a significant research and engineering challenge. This document deconstructs the problem from first principles and proposes a novel, hybrid system that synthesizes state-of-the-art (SOTA) techniques in signal processing, generative modeling, and digital communications.

Part 1: Foundational Concepts in Advanced Steganography

1.1 The Central Trilemma: Deconstructing the Project's Core Challenge

Any digital data hiding system is governed by a fundamental and conflicting set of properties: the "steganographic trilemma". This defines the trade-off between:

1. **Embedding Capacity:** The amount of data (payload) that can be hidden within the cover medium. The query demands this be extremely high (an entire audio file).
2. **Imperceptibility (Security):** The visual or statistical indistinguishability of the stego-medium (the data-carrying image) from an original cover image. High imperceptibility is required to evade both human detection and computational steganalysis.
3. **Robustness:** The ability of the hidden payload to survive "attacks," which include unintentional modifications like lossy compression (e.g., JPEG, WebP) or intentional signal processing.

Traditional steganography prioritizes capacity and imperceptibility, making it fragile. Digital watermarking prioritizes robustness but typically offers very low capacity. The project query,

by demanding maximum capacity, perfect imperceptibility, *and* extreme robustness, is effectively targeting a "holy grail" solution that pushes beyond the boundaries of established models.

1.2 The User's Paradox: Reconciling "Lossless" with "Lossy"

The most significant technical challenge is the central paradox within the query: the demand for "without loss any data" in a system that *must* survive "decompression" [sic] (lossy compression) from platform sharing.

A lossy operation like JPEG compression *by definition* discards information to reduce file size. It is therefore a mathematical impossibility for the stego-image to pass through a lossy channel and remain bit-identical. A naive implementation would result in the decoder (Model 3) receiving a corrupted input, leading to a corrupted output.

The "lossless" requirement must therefore be re-framed. It cannot apply to the *stego-image* but must apply to the *final, recovered audio payload*. This reframes the entire project from a simple AI task into the design of a complete *end-to-end digital communication system*. The neural networks must be designed to model a noisy communication channel (the social media platform), and the system must include components for managing and correcting the resulting data corruption.

1.3 The Hybrid Solution Framework: A Three-Pillar Architecture

A successful system that resolves this paradox must be built upon three distinct but interconnected pillars:

1. **Pillar 1: Invertible Neural Networks (INNs) for Lossless Capacity.** To satisfy the "lossless" requirement in an ideal, noiseless channel, Models 2 and 3 must not be two independently trained networks. They must be a *single*, mathematically **Invertible Neural Network (INN)**. The encoding process (Model 2) is the network's forward pass, and the decoding process (Model 3) is its mathematically *perfect* reverse pass. This architecture guarantees high capacity and perfect reversibility *before* the attack layer.
2. **Pillar 2: Differentiable Attack Simulation for Robustness.** To satisfy the "platform sharing" requirement, the INN must be *trained* to be robust to the specific artifacts of JPEG compression. This is achieved by inserting a *differentiable approximation* of the non-differentiable JPEG algorithm (a "Diff-JPEG" layer) between the encoder's output

and the decoder's input *during the training loop*. This "noise layer" [1] forces the INN to learn a latent representation that is inherently resistant to JPEG's quantization.

3. **Pillar 3: Error Correction Codes (ECCs) for Bit-Perfect Recovery.** The robust training from Pillar 2 will produce a low bit-error-rate (BER), but not a *zero* BER. To bridge the gap from "highly robust" to "perfectly lossless," the audio payload must be pre-processed with **Forward Error Correction (FEC)**, also known as **Error Correction Codes (ECCs)**. The neural network's task is thus simplified: it no longer needs to achieve *perfect* recovery, only a recovery accurate *enough* for the ECC decoder to fix the remaining errors and restore the perfect, original bitstream.

This project's advanced-level novelty lies in the co-design and optimization of this hybrid system.

Part 2: Model 1 - State-of-the-Art Neural Audio Enhancement

2.1 Objective and Data Representation

The first model's objective is to "remove background voice" (a task of **Audio Source Separation**) and "make it listenable" (a task of **Speech Enhancement**). Real-world noise is complex and non-stationary, causing traditional methods like Wiener filtering or spectral subtraction to fail and introduce "musical noise" artifacts.

A deep learning solution is required. The primary architectural decision is the data domain:

1. **Time-Domain (Waveform):** Operates on the raw audio waveform (e.g., Wave-U-Net). This avoids transformation overheads but can be computationally intensive and may struggle to separate frequency-based information cleanly.
2. **Time-Frequency (T-F) Domain (Spectrogram):** Operates on the Short-Time Fourier Transform (STFT). This is a common and effective representation. However, many naive T-F methods enhance only the *magnitude* spectrogram and reuse the original *noisy phase*, which is a primary source of artifacts and fails to produce "listenable" audio.

High-quality enhancement *must* be phase-aware. Therefore, Model 1 must operate on the *complex spectrogram* (both real and imaginary components) or learn to predict a clean phase component.

2.2 State-of-the-Art Architectural Review

A U-Net (encoder-decoder) architecture is a powerful and common backbone for this task, as it maps noisy features to clean features at multiple resolutions. However, a standard U-Net is insufficient.

2.2.1 SOTA Option 1: Deep Complex Convolution Recurrent Network (DCCRN)

DCCRN is a SOTA U-Net-based architecture that is explicitly "phase-aware". Its core innovation is the replacement of all standard operations with *complex-valued* operations:

- **Complex Convolutions:** The network's convolutions are defined to handle complex numbers, allowing it to natively model the real and imaginary parts of the T-F spectrogram.
- **Complex LSTMs:** The recurrent layers (LSTMs) in the bottleneck are also complex-valued, allowing the model to learn temporal dependencies in the complex domain.
- **Complex Batch Normalization:** All normalization layers are similarly complex.

This architecture explicitly models the "correlation between magnitude and phase", which is precisely what is required to suppress noise *and* interfering voices while reconstructing a clean, high-fidelity speech signal. DCCRN was a winner of the ICASSP 2020 Deep Noise Suppression (DNS) Challenge.

2.2.2 SOTA Option 2: Mamba-SEUNet

Emerging in 2024/2025, Mamba-SEUNet represents the bleeding edge. The primary weakness of CNNs (like in a U-Net) is their difficulty in modeling *long-range temporal dependencies* in speech. RNNs/LSTMs (as in DCCRN) solve this but are computationally complex. Mamba is a *structured state-space model (SSM)* that excels at modeling long sequences with high efficiency. By integrating Mamba into a U-Net, this architecture achieves SOTA performance by capturing the long-range context of speech.

2.3 Advanced Loss Functions for Perceptual Quality

The choice of loss function is more critical than the architecture. A simple Mean-Squared Error (MSE) loss will produce muffled, "un-listenable" audio, as it does not align with human auditory perception. The optimization target *must* be a perceptual metric.

The best metrics, **PESQ** (Perceptual Evaluation of Speech Quality) and **STOI** (Short-Time Objective Intelligibility), are non-differentiable and cannot be used directly as loss functions.

- **Solution 1: Differentiable Metric Proxies.** This approach uses a PyTorch-based, *differentiable approximation* of the PESQ score as the loss function.[2] This allows the network to *directly* optimize for PESQ. However, this is an approximation and works best when *combined* with other losses like L1 or Scale-Invariant SDR (SI-SDR).
- **Solution 2: Multi-MetricGAN (Recommended).** This is the most advanced and robust approach. Instead of using a *fixed* proxy for PESQ, this method trains a *second* neural network (a discriminator) to *predict* the true, non-differentiable metric score.
    - **Generator:** Model 1 (e.g., the DCCRN), which produces enhanced speech.
    - **Discriminator:** An "assessment network" (e.g., based on Multi-gate Mixture-of-Experts, MMoE) that is trained to estimate the true PESQ and STOI scores of the enhanced speech.
    - **Training:** The generator (Model 1) is then trained to *fool* this discriminator. In essence, Model 1 learns to produce audio that the discriminator *believes* will receive a high PESQ score. This forces the optimization to align with human perception.

2.4 Recommendation and Requirements (Model 1)

- **Architecture: DCCRN**. Its native, proven handling of complex spectrograms is the most direct solution to the phase-aware enhancement requirement.
- **Training:** A **Multi-MetricGAN** framework. The DCCRN will act as the generator. The discriminator will be a separate network trained to predict PESQ and STOI scores.
- **Data Representation:** The output of Model 1, which will be the *input payload* for Model 2, should be the enhanced *complex spectrogram* (i.e., the real and imaginary components). This provides a structured, multi-channel 2D data format that is ideal for embedding within an image.

Part 3: Models 2 & 3 - A Robust, Lossless

Steganographic Autoencoder

3.1 Core Architecture: The Adversarial Autoencoder

The user's Model 2 (Encoder) and Model 3 (Decoder) form a classic steganographic autoencoder.

- **Model 2 (Encoder):** This network takes two inputs—a *cover image* and the *payload* (the enhanced audio spectrogram from Model 1)—and embeds the payload into the image, producing a *stego-image*.
- **Model 3 (Decoder):** This network takes the stego-image and attempts to *reveal* the original payload.

A simple autoencoder trained with an MSE loss will fail, producing blurry, easily detected artifacts. To achieve the *imperceptibility* requirement, this system must be trained as a **Generative Adversarial Network (GAN)**. This introduces a fourth model (Model D, the Discriminator):

- **Model 2 (Generator):** Tries to create a stego-image that is (a) visually identical to the cover and (b) decodable by Model 3.
- **Model D (Discriminator/Steganalyzer):** A separate classifier (e.g., advanced steganalysis networks like SRNet or Zhu-Net) trained to distinguish *real* cover images from *fake* stego-images.

The adversarial loss from Model D forces Model 2 to produce stego-images that are *statistically* indistinguishable from the original covers. This framework is the basis for SOTA models like **HiDDeN** [1] and **SteganoGAN**.

3.2 Addressing the "Lossless" Requirement: Invertible Neural Networks (INNs)

A standard autoencoder, even in a GAN, is a *compressive* architecture (e.g., Encoder -> Bottleneck -> Decoder). It *cannot* satisfy the "without loss any data" requirement.

The solution is to build the autoencoder as an **Invertible Neural Network (INN)**. An INN is constructed from bijective (one-to-one) transformations, ensuring that no information is lost.

This is an architectural mandate to fulfill the query.

- **Model 2 (Encoder):** Stego_Image = INN.forward(Cover_Image, Audio_Payload)
- **Model 3 (Decoder):** Audio_Payload = INN.reverse(Stego_Image)

Because the reverse operation is the *mathematical inverse* of the forward operation, this architecture *guarantees* that INN.reverse(INN.forward(Cover, Payload)) will perfectly reconstruct the payload. This is the *only* deep learning architecture that can satisfy the "lossless" requirement in a noiseless channel. SOTA models like the **Mamba-based Robust Invertible Network (MRIN)** [3] use this principle.

3.3 Addressing the "Robustness" Requirement: The Differentiable Noise Layer

The INN's "lossless" guarantee is immediately broken by the "platform sharing" attack, as INN.reverse(Attacked_Stego_Image) will not yield the correct payload.

To solve this, the INN must be *trained* to be robust to this attack. This is accomplished by inserting a **"noise layer"** [1] between the encoder's output and the decoder's input *during training*.

This noise layer cannot be random. It must be a **differentiable proxy of the JPEG compression algorithm**.[4] The JPEG algorithm (which includes DCT, quantization, and dequantization) is non-differentiable because its quantization step involves a round() operation. This rounding stops the gradient during backpropagation.

To overcome this, the non-differentiable round(x) operation must be replaced with a differentiable surrogate, such as:

- A **differentiable soft quantizer**.
- A **Straight-Through Estimator (STE)**, which uses the hard round() in the forward pass but passes the gradient through as an identity function (1.0) in the backward pass.
- A complete, differentiable model of the entire JPEG pipeline, including clipping and flooring operations.

The full training loop for robustness is as follows:

1. Cover_Image and Audio_Payload are fed into Model 2 (INN.forward) to produce Stego_Image.
2. Stego_Image is passed through the Differentiable_JPEG_Layer to produce Attacked_Stego_Image.
3. Attacked_Stego_Image is fed into Model 3 (INN.reverse) to produce Corrupted_Payload.

4. A Payload_Loss is calculated between Corrupted_Payload and the original Audio_Payload.
5. This loss is backpropagated *through* the Diff-JPEG layer and *through* the INN.

This process forces the INN to learn an embedding strategy (a latent representation) that is *specifically resistant* to the information loss from JPEG quantization.

The following table justifies this hybrid architecture by comparing it to existing SOTA models.

Table 1: Steganographic Architecture & Property Comparison

Architecture	**Core Principle**	**High Capacity?**	**Lossless (Noiseless Channel)?**	**Robust (Noisy Channel)?**	**User Query Fit?**
SteganoGAN	Autoencoder + GAN	Yes	No (Compressive)	No (Not trained for it)	**Fails** (Lossless & Robust)
HiDDeN / StegaStamp	Autoencoder + GAN + Noise Layer	Yes	No (Compressive)	**Yes**	**Fails** (Lossless)
INN-Steg	Invertible NN + GAN	**Yes**	**Yes** (Bijective)	No (Not trained for it)	**Fails** (Robust)
Proposed Hybrid	**INN + GAN + Diff-JPEG Layer**	**Yes**	**Yes** (Bijective)	**Yes** (Trained for it)	**Matches all requirements**

Part 4: Bridging the Gap: Error Correction for True Lossless Recovery

4.1 Why Robustness Alone is Insufficient (The "Last-Bit" Problem)

The INN + Diff-JPEG system trained in Part 3 will be *highly robust*, yielding a very low bit-error-rate (BER). However, lossy compression is stochastic and non-linear. It is *statistically impossible* to train a neural network to achieve a BER of *zero* for every possible image and compression level.

A 99.99% recovery accuracy is a *failure* under the "without loss any data" mandate. The system *must* achieve 100.00% bit-perfect reconstruction. This "last 0.01%" gap is one that the neural network *cannot* solve alone.

4.2 Payload Integration with Error Correction Codes (ECCs)

This is a classic problem in digital communications, solved using **Forward Error Correction (FEC)**. The audio payload must be treated as a bitstream that requires protection.

- **Mechanism:** Before being fed to the INN (Model 2), the audio bitstream is *pre-processed* by an ECC encoder. This encoder adds *redundant parity bits* to the data.
- **Recommended Codes:**
  - **Hamming Codes**: Simple and effective for demonstrating the concept.
  - **BCH (Bose-Chaudhuri-Hocquenghem) Codes**: More powerful and can be tailored to correct a specific number t of errors.
  - **Reed-Solomon (RS) Codes**: A subclass of BCH codes, they are exceptionally powerful against *burst errors* (many consecutive bits lost), which is a common artifact of block-based compression like JPEG.

When the corrupted payload is extracted by Model 3, it is passed to an ECC decoder. The decoder uses the redundant parity bits to identify and *correct* the errors introduced by the JPEG compression, restoring the *perfect* original bitstream.

4.3 The Final Integrated Data Pipeline (Models 2 & 3)

This requirement creates a new, advanced co-design problem. More powerful ECCs (which correct more errors) add more redundancy, or "overhead," which *reduces* the effective embedding capacity of the INN. The advanced optimization problem is: "What is the *minimal* ECC (lowest overhead) required to handle the *maximal* BER produced by the INN when trained

with the Diff-JPEG layer?".

- **Model 2 Pipeline (Encoding):**
  1. Start with Clean_Audio_Spectrogram (from Model 1).
  2. Flatten and serialize to Audio_Bitstream.
  3. `` -> Redundant_Payload.
  4. `` -> Stego_Image.
- **Model 3 Pipeline (Decoding):**
  1. Start with Attacked_Stego_Image (from platform).
  2. `` -> Corrupted_Redundant_Payload (Contains bit errors).
  3. `` -> Audio_Bitstream (Errors are corrected).
  4. Reconstruct Clean_Audio_Spectrogram from Audio_Bitstream.

Part 5: Advanced Features for an "Advanced Level" Project

To truly elevate the project, the following SOTA features should be integrated into the INN (Model 2/3) architecture.

5.1 Adaptive Steganography via Attention Mechanisms

Not all regions of an image are equally "safe" for hiding data. Embedding data in smooth, low-frequency areas (like a blue sky) creates obvious artifacts, whereas embedding in high-frequency, complex-textured areas (like grass or foliage) is less perceptible. This concept is known as **adaptive steganography**.

This can be implemented in a deep learning context using **attention mechanisms**. The INN (Model 2) can be enhanced with **parallel channel and spatial attention modules**, as seen in models like StegaVision.[5]

- **Spatial Attention:** This module learns to generate a mask that "attends" to the complex texture regions of the cover image, guiding the INN to embed a higher density of payload bits in those "safe" areas.
- **Channel Attention:** This module learns which color channels (R, G, or B) are safest for embedding in any given pixel.

This "adaptive" approach provides a dual benefit:

1. **Improves Imperceptibility:** By hiding data in perceptually complex regions, it makes the stego-image *harder* for the adversarial discriminator (Model D) to detect, thus improving security.
2. **Improves Robustness:** JPEG compression works by aggressively quantizing high-frequency information (i.e., textures). By training the *attention-guided* INN *through* the Diff-JPEG layer, the system will learn to "attend" to regions that are *both* perceptually complex *and* maximally resistant to JPEG's specific quantization tables.

5.2 Perceptual Loss for Imperceptibility (LPIPS)

The adversarial loss from Model D ensures *statistical* similarity. For *visual* similarity, a standard L2 (MSE) or SSIM loss is insufficient, as these metrics correlate poorly with human visual perception.

The SOTA solution is to use the **Learned Perceptual Image Patch Similarity (LPIPS)** as a loss function.[6] LPIPS is a "perceptual loss" that measures the distance between two images *in the deep feature space* of a pre-trained network, such as AlexNet or VGG. It compares the *activations* of the Cover_Image and Stego_Image at different layers, providing a metric that aligns closely with human similarity judgments.

The *total loss function* for training the Generator (Model 2) should be a weighted composite of three components:

1. $\mathcal{L}_{\text{PAYLOAD}}$: (e.g., L1 or Cross-Entropy Loss) between the *original* payload and the *post-attack, pre-ECC* payload. This ensures *reconstruction*.
2. $\mathcal{L}_{\text{ADVERSARIAL}}$: The loss from the discriminator (Model D). This ensures *security/undetectability*.
3. $\mathcal{L}_{\text{PERCEPTUAL (LPIPS)}}$: The LPIPS score between the Cover_Image and the Stego_Image. This ensures *visual imperceptibility*.

5.3 Payload Encryption for Security

Steganography (covertness) is not cryptography (secrecy). If Model 3 (the INN) is ever stolen or reverse-engineered, an adversary could extract the audio payload from *any* stego-image.

For a truly secure, advanced-level system, the audio bitstream *must* be **encrypted** (e.g., using **AES-256**) *before* the ECC and embedding steps. This ensures that even if an adversary successfully extracts the payload, it is computationally infeasible to access the content

without the secret key. This also has a positive side-effect: encrypted data is high-entropy (statistically random), which can make the embedding process less detectable to some forms of steganalysis.

Part 6: Implementation Plan: Datasets, Tools, and Requirements

6.1 Dataset Requirements

A high-quality, large-scale dataset is non-negotiable for each stage of this project.

Table 2: Recommended Datasets for Project Stages

Project Stage	Purpose	Recommended Dataset(s)	Rationale
Model 1 Training (Audio Enhancement)	Train the phase-aware audio enhancement network.	**Microsoft DNS Challenge** [7]	SOTA dataset. Provides massive-scale clean speech, diverse noise files, and room impulse responses (RIRs) to synthesize parallel clean/noisy pairs.
	(Alternative)	**VoiceBank+DEMAND**	A common, smaller benchmark for speech enhancement, combining the clean VoiceBank corpus with the DEMAND noise

			corpus.
Model 2/3 Training (Cover Images)	The "cover" medium for the steganographic encoder.	**COCO (Common Objects in Context)**	Excellent choice. ~330K diverse images with complex scenes, which provides the varied textures needed for adaptive hiding.
	(Alternative)	**ImageNet**	A classic, massive-scale (1M+) dataset.
Model 2/3 Training (Payload Data)	The "secret" data to be hidden.	**LibriSpeech**	1000+ hours of clean, read English speech. Provides a massive source of payload data.
	(Alternative)	**VCTK Corpus**	~110 speakers with diverse accents.[8] Ideal for ensuring the model isn't overfitting to a single speaker's voice characteristics.

6.2 Framework and Library Requirements

- **Deep Learning Framework: PyTorch (Recommended)**
  - This project is novel research, not a deployment of a standard model. It requires building custom architectures (INNs), custom loss functions (LPIPS, MetricGAN), and a custom, non-standard layer (the Diff-JPEG proxy).
  - While TensorFlow is mature and excellent for production, **PyTorch** is the dominant framework in research for its "Pythonic" feel, flexibility, and dynamic computation

graph (Eager Execution). This flexibility is *essential* for the experimentation this project demands.

- **Audio Processing: Librosa**
  - **Librosa** is the industry-standard Python library for audio analysis. It will be required for all audio I/O, pre-processing (e.g., resampling), and, crucially, for generating the STFT (complex spectrogram) representations.
- **Error Correction: bchlib / reedsolo**
  - Python libraries such as bchlib (for BCH codes) or reedsolo (for Reed-Solomon codes) will be needed to implement the pre-processing (encoding) and post-processing (decoding) of the payload bitstream.
- **Image Processing:** Pillow and OpenCV will be required for handling the loading, pre-processing, and saving of the cover and stego-images.

6.3 Hardware Requirements

This is a computationally "expensive" project. The primary bottleneck will be GPU VRAM. The system involves training at least two large, concurrent models (the INN Generator, G_INN, and the Steganalyzer Discriminator, D_Steg) on high-resolution image data. This is not feasible on consumer-grade (e.g., < 12GB) GPUs. A minimum of an **NVIDIA RTX 3090 or 4090 (24GB)** is required for reasonable batch sizes. An **NVIDIA A100 or H100 (40GB-80GB)** is strongly recommended for faster, more stable training.

Part 7: Final Synthesized Architecture and Recommendations

This section provides the final, integrated blueprint that synthesizes all recommendations into a single, cohesive system.

7.1 Final System Pipeline (End-to-End)

User 1 (Sender) Pipeline:

1. **Input:** raw_audio.wav

2. **Model 1 (DCCRN):** The audio is enhanced, producing a clean complex_spectrogram.
3. **Payload Prep:** The spectrogram is quantized and flattened into a payload_bitstream.
4. **Security (Crypto):** encrypted_bits = AES_256_Encrypt(payload_bitstream, key)
5. **Robustness (ECC):** redundant_bits = BCH_Encode(encrypted_bits)
6. **Input:** cover_image.png
7. **Model 2 (INN-Encoder):** The attention-guided INN embeds the payload: stego_image = G_INN.forward(cover_image, redundant_bits)
8. **Output:** stego_image.png (This file is shared on a platform, which compresses it to JPEG).

User 2 (Receiver) Pipeline:

1. **Input:** stego_image.jpg (Downloaded from the platform, now lossy).
2. **Model 3 (INN-Decoder):** The INN reverses the process: corrupted_bits = G_INN.reverse(stego_image)
3. **Robustness (ECC):** The BCH decoder finds and fixes bit errors: corrected_bits = BCH_Decode(corrupted_bits)
4. **Security (Crypto):** payload_bitstream = AES_256_Decrypt(corrected_bits, key)
5. **Payload Post-Proc:** The bitstream is inverse-quantized and reformed into complex_spectrogram.
6. **Output:** The audio is reconstructed via inverse STFT: reconstructed_audio.wav. This audio is bit-perfect to the output of Model 1.

7.2 The "Advanced" Training Loop

The system is trained by optimizing two networks, G_INN (Models 2/3) and D_Steg (Model D), in an adversarial manner.

- **Networks in Training:**
  1. G_INN: The Generator. An **Invertible Neural Network** enhanced with **Attention**.
  2. D_Steg: The Discriminator. A SOTA **Steganalyzer CNN** (e.g., Zhu-Net).
  3. DiffJPEG: The non-trainable, **differentiable JPEG attack layer**.
- Composite Generator Loss (to train G_INN): $\mathcal{L}_{\text{Generator}} = \lambda_1 \cdot \mathcal{L}_{\text{PAYLOAD}} + \lambda_2 \cdot \mathcal{L}_{\text{PERCEPTUAL}} + \lambda_3 \cdot \mathcal{L}_{\text{ADVERSARIAL}}$
  - $\mathcal{L}_{\text{PAYLOAD}}$: **L1 Loss** between the *original* redundant_bits and the *attacked-and-decoded* corrupted_bits.
  - $\mathcal{L}_{\text{PERCEPTUAL}}$: **LPIPS Loss** between the cover_image and the stego_image.
  - $\mathcal{L}_{\text{ADVERSARIAL}}$: **Adversarial Loss** from D_Steg's classification

of the stego_image.

- **Discriminator Loss (to train D_Steg):**
  - $\mathcal{L}_{\text{Discriminator}}$: **Standard GAN Loss** (e.g., Binary Cross-Entropy) to distinguish cover_image (real) from stego_image (fake).

7.3 Final Architectural Blueprint

The following table provides the complete, at-a-glance blueprint for the proposed system.

Table 3: Final Proposed Hybrid Architecture Blueprint

Component	Recommended Architecture	Key Technology	Purpose	SOTA Loss Function
Model 1 (Enhancer)	**DCCRN**	Complex-Valued Convs/LSTMs	Phase-Aware Audio Enhancement	**Multi-MetricGAN** (predicting PESQ/STOI)
Model 2 (Encoder)	**INN (Forward Pass) + Attention**	Bijective Transformation & Adaptive Hiding	Hides payload in robust, complex regions	*Composite Loss* (see 7.2)
Model 3 (Decoder)	**INN (Reverse Pass)**	Mathematical Inversion	Recovers payload (with errors)	*Composite Loss* (see 7.2)
Attack Layer (Training)	**Diff-JPEG Proxy**	Differentiable Quantization (STE)	Forces Model 2 to learn robustness	(Not trained)
Adversary (Training)	**Steganalyzer (e.g., Zhu-Net)**	Deep Steganalysis CNN	Forces Model 2 to learn imperceptibility	Binary Cross-Entropy

Pre/Post-Proc	**AES-256 + BCH Codec**	Cryptography + Error Correction	Provides data security & final lossless guarantee	(Algorithmic)

Works cited

1. HiDDeN: Hiding Data with Deep Networks - CVF Open Access, accessed on November 5, 2025, https://openaccess.thecvf.com/content_ECCV_2018/papers/Jiren_Zhu_HiDDeN_Hiding_Data_ECCV_2018_paper.pdf
2. audiolabs/torch-pesq: PyTorch implementation of the ... - GitHub, accessed on November 5, 2025, https://github.com/audiolabs/torch-pesq
3. An Invertible, Robust Steganography Network Based on Mamba, accessed on November 5, 2025, https://www.mdpi.com/2073-8994/17/6/837
4. [2309.06978] Differentiable JPEG: The Devil is in the Details - arXiv, accessed on November 5, 2025, https://arxiv.org/abs/2309.06978
5. StegaVision: Enhancing Steganography with Attention Mechanism (Student Abstract) - arXiv, accessed on November 5, 2025, https://arxiv.org/html/2411.05838v1
6. Learned Perceptual Image Patch Similarity (LPIPS) — PyTorch ..., accessed on November 5, 2025, https://lightning.ai/docs/torchmetrics/stable/image/learned_perceptual_image_patch_similarity.html