Passive Defence
Vol. 16, No. 4, Winter 2026, Serial No. 64
ISSN: 2980-8030 , E-ISSN: 2008-6849

GAN-Based Image Steganography Using EGD Architecture

Reza Esfahani[1*], Mohamm Ali Akhaee[2], Zeynolabedin Norouzi[3]

[1]Correspondence: Assistant Professor, Imam Hossein Comprehensive University, Tehran, Iran. Email Address: resfahani@ihu.ac.ir

[2]Associate Professor, Tehran University, Tehran, Iran. Email Address: akhaee@ut.ac.ir

[3]Associate Professor, Imam Hossein Comprehensive University, Tehran, Iran. Email Address: znorozi@ihu.ac.ir

ARTICLE INFO

Article history:

Article Type: Research paper

Received: 11 September 2025

Received in revised form: 5 October 2025

Accepted: 20 October 2025

Available online: 20 January 2026

Keywords:

Steganography

Generative Adversarial Neural Networks (GAN)

Encoder-Generator-Discriminator (EGD)

Loss Function (Loss)

Binary Cross Entropy (BCE)

ABSTRACT

Steganography of a text across multiple images enhances security and prevents attackers from accessing the hidden message; moreover, generating multiple images with artificial intelligence also facilitates the availability of images for embedding each segment of the hidden text. This paper introduces a steganography system based on Generative Adversarial Networks that uses a three-component Encoder-Decoder-Discriminator architecture, the encoder extracts messages from steganographic images, and the CIFAR-10 dataset is employed for evaluation. This general method of "Generator and Encoder" combines a cross-entropy adversarial loss to fool the discriminator and a message loss to recover the message with the aid of the encoder. The main objective is to reduce the visible alterations in steganographic images compared to the original (Real) images and to enhance the recoverability of the message by the encoder, along with qualitative and quantitative evaluations such as SNR and BCE/MSE for message recovery. Experimental results show that, with proper hyperparameter tuning, messages can be embedded in images with minimal noise, and the message recovery exhibits low error and a desirable SNR. In the proposed method, at epoch 50, the SNR is approximately 34.9 dB. In this case, Loss_D is approximately 0.7 and Loss_G is approximately 1.2, which are suitable values for cross-entropy. The value of LossMsgLossMsg is approximately 0.03, which is considered excellent. It should be noted that the implementation in this article was performed using Python.

Cite this article: R. Esfahani1, M. A. Akhaee, and Z. Norouzi, "GAN-Based Image Steganography Using EGD Architecture," Journal of Passive Defence, vol. 16, no. 4, pp. 137-154, 2026. **DOI: https://doi.org/10.47176/PD.2026.1574**

Publisher: Imam Hossein University.

Introduction

Steganography, the art and science of concealing information within suitable carriers, has witnessed a transformative evolution through the integration of deep generative models, particularly Generative Adversarial Networks (GANs). Traditional steganographic approaches, such as Least Significant Bit (LSB) substitution or Discrete Cosine Transform (DCT)-based embedding, often suffer from detectability under statistical analysis and limited robustness against modern steganalytic attacks. In response, the fusion of artificial intelligence and information hiding has opened new frontiers in secure and imperceptible data embedding. GANs, with their inherent capacity to generate high-fidelity synthetic data, present a compelling framework for steganography by enabling the creation of cover images that are statistically indistinguishable from natural ones. However, conventional GAN-based steganography typically focuses solely on adversarial image generation, often neglecting the explicit optimization of message recovery fidelity. To address this gap, this study proposes a novel Encoder-Generator-Discriminator (EGD) architecture that synergistically combines the generative power of GANs with the precise message reconstruction capability of a dedicated encoder. By embedding secret messages into AI-generated images and simultaneously training a decoder (encoder) to retrieve them with minimal distortion, this approach achieves a dual objective: «high visual fidelity and reliable message extraction». The system is evaluated on the CIFAR-10 dataset, a benchmark for real-world image complexity, ensuring relevance to practical deployment scenarios.

Research Objectives and Questions

This research is driven by three core objectives:

1. To design and implement a three-component EGD framework where the Generator embeds a binary message into digital images, the Discriminator ensures visual realism by distinguishing generated images from real ones, and the Encoder reliably extracts the hidden message.
2. To minimize perceptual distortion in stego-images, quantified by Signal-to-Noise Ratio (SNR) and Mean Squared Error (MSE), while maximizing message recovery accuracy, measured via Binary Cross-Entropy (BCE) loss between original and extracted messages.
3. To establish empirical evidence that explicit encoder integration within GAN-based steganography significantly improves message retrieval performance compared to encoder-free alternatives.

These objectives give rise to the following research questions:

- RQ1: Can a GAN-based system with an integrated encoder achieve both high visual quality (SNR > 30 dB) and low message reconstruction error (LossMsg < 0.05)?
- RQ2: How do adversarial losses (LossG and LossD) and message loss (LossMsg) co-evolve during training, and what hyperparameter configurations yield stable convergence?
- RQ3: How does the proposed EGD method compare quantitatively against classical steganographic techniques and recent GAN-based alternatives in terms of SNR and message recovery fidelity?

Methodology

The proposed EGD architecture comprises three neural modules trained in an end-to-end adversarial-cooperative framework:

- The Generator (G) receives a concatenated input of a 128-dimensional latent noise vector z and a binary message m (e.g., 40-bit "HELLO" string), and outputs a 32×32×3 RGB stego-image using transposed convolutions, BatchNorm, ReLU activations, and a final tanh layer for pixel normalization.
- The Discriminator (D), structured as a deep convolutional network with LeakyReLU and BatchNorm, assigns a realism probability to images, trained via BCEWithLogitsLoss to distinguish between original CIFAR-10 images and G-generated stego-images.
- The Encoder (E), mirroring the Discriminator's downsampling path, processes stego-images through four convolutional stages to reconstruct the embedded message, optimizing a BCE message loss (LossMsg) between m and its estimate $\hat{m}$.

The total objective function combines:

- Adversarial loss: $\mathcal{L}_{adv} = \mathcal{L}_G + \mathcal{L}_D$(cross-entropy formulation),
- Message loss: $\mathcal{L}_{msg} = \text{BCE}(m, \hat{m})$.

Training proceeds for 50 epochs on CIFAR-10 (batch size = 64), with simultaneous updates to G, D, and E.

Performance is evaluated using SNR, MSE, and LossMsg, alongside qualitative inspection of real vs. stego-image pairs.

Findings

Experimental results confirm the efficacy of the EGD framework:

- Under optimal conditions (embedding a single bit per image), the system achieves SNR ≈ 34.9 dB at epoch 50, indicating near-imperceptible distortion.
- The adversarial losses stabilize at LossD ≈ 0.7 and LossG ≈ 1.2, reflecting equilibrium between Generator and Discriminator, critical for GAN stability.
- Most notably, LossMsg converges to 0.03, an "excellent" value denoting high message retrieval accuracy (≈97% bit correctness).
- In contrast, embedding a 40-bit message into 32×32 images yields lower SNR (−1 to −7 dB) due to capacity limitations, highlighting the trade-off between payload and imperceptibility.
- Comparative analysis (Table 9) shows that while classical DCT-based methods achieve slightly higher SNR (36.8 dB), they lack the adaptive, learning-based robustness and message recovery precision of EGD, which outperforms prior GAN-steganography works in LossMsg (0.03 vs. 0.07–0.15 in references [8]–[11]).

Discussion

The EGD architecture successfully decouples steganographic requirements into specialized modules: the Generator handles *embedding realism*, the Discriminator enforces *statistical indistinguishability*, and the Encoder guarantees *message fidelity*. This modularity addresses a key limitation of prior GAN-steganography systems, where message recovery was often treated as a byproduct rather than a primary objective. The low LossMsg demonstrates that explicit encoder training is pivotal for reliable extraction; a finding with significant implications for secure communications where message integrity is non-negotiable. However, the negative SNR observed with larger payloads underscores a fundamental constraint: «embedding capacity vs. visual quality». Future work should explore attention mechanisms (e.g., as in [10]) to embed messages selectively in perceptually redundant image regions, or adopt larger

input resolutions (e.g., 128×128) to increase payload capacity. Additionally, the use of Wasserstein GAN with Gradient Penalty (WGAN-GP) could further stabilize training and mitigate mode collapse, especially beyond 50 epochs.

Conclusions and Implications

This study demonstrates that integrating an Encoder into the GAN framework for image steganography yields a robust, end-to-end system that excels in both visual quality and message recovery. The EGD architecture not only achieves state-of-the-art message reconstruction accuracy (LossMsg = 0.03) but also maintains competitive SNR (34.9 dB) under constrained payload conditions. These results validate the hypothesis that explicit optimization of message retrieval is essential for practical steganographic applications. From a broader perspective, this work contributes to the emerging field of *AI-driven secure communications*, offering a template for embedding structured data into synthetic media with high integrity. For real-world deployment, the approach could be extended to video steganography, multi-modal carriers, or federated learning settings where privacy-preserving data sharing is paramount. Ultimately, the EGD paradigm represents a significant step toward smart, adaptive, and resilient information hiding in an era of increasingly sophisticated adversarial analysis.

پدافند غیرعامل

سال پانزدهم، شماره ۴، زمستان ۱۴۰۴، (پیاپی ۶۴): صص ۱۵۴-۱۳۷

شاپای چاپی: ۶۹۴۹-۲۰۰۸ | شاپای الکترونیکی: ۸۰۳۰-۲۹۸۰

نهان‌نگاری مبتنی بر GAN در تصاویر با استفاده از یک معماری EGD

رضا اصفهانی[1*]، محمدعلی اخایی[2]، زین‌العابدین نوروزی[3]

[1] استادیار، گروه مخابرات، دانشگاه جامع امام حسین(ع)، تهران، ایران(نویسنده مسئول). رایانامه: resfahani@ihu.ac.ir

[2] دانشیار، گروه مخابرات، دانشگاه تهران، تهران، ایران. رایانامه: akhaee@ut.ac.ir

[3] دانشیار، گروه رمز و امنیت، دانشگاه جامع امام حسین(ع)، تهران، ایران. رایانامه: znorozi@ihu.ac.ir

چکیده

نهان‌نگاری یک متن در چند تصویر، باعث ارتقاء امنیت و جلوگیری از دسترسی مهاجمین به پیام پنهان خواهد شد؛ همچنین تولید تصاویر متعدد توسط هوش مصنوعی نیز، در دسترس قراردادن تصاویر جهت درج هر بخش از متن پیام پنهان را به دنبال خواهد داشت. در این مقاله یک سیستم نهان‌نگاری مبتنی بر «شبکه‌های عصبی متخاصم مولد» با استفاده از طراحی معماری سه بخشی فرمت «کدگذار- مولد- متمایزکننده» معرفی می‌شود که از یک کدگذار برای استخراج پیام‌ها از تصاویر نهان‌نگاری‌شده با بهره‌گیری از یک دیتاست (مجموعه داده) واقع‌گرای CIFAR-10 برای ارزیابی استفاده می‌کند. روش کلی «مولد به همراه کدگذار»، ترکیبی از شیوه تخاصم اتلافی از نوع «آنتروپی متقاطع» به منظور فریب دادن متمایزکننده و بهره‌گیری از اتلاف پیام، برای بازیابی پیام به کمک کدگذار است. هدف اصلی کاهش تغییرات قابل مشاهده در تصاویر نهان‌نگاری‌شده نسبت به تصاویر اصلی و افزایش قابلیت بازیابی پیام توسط کدگذار، همراه با ارزیابی‌های کیفی و کمی مانند SNR و MSE یا BCE برای بازگرداندن پیام است. نتایج تجربی نشان می‌دهد که با تنظیم مناسب‌های هایپرپارامترها، پیغام‌ها با نویز کم در تصاویر درج می‌شوند و قابلیت بازیابی پیام با خطای کم و SNR مطلوبی همراه است. در روش پیشنهادی، برای Epoch=۵۰ مقدار dB SNR≅۳۴/۹ است. در این حالت همچنین LossD ≅ ۰/۷ و LossG≅۱/۲ بوده که مقادیر مناسبی برای آنتروپی متقاطع است. مقدار LossMsg≅۰/۰۳ هم حاصل شد که مقدار عالی محسوب می‌شود.

مشخصات مقاله

تاریخچه مقاله:

نوع مقاله: علمی پژوهشی

دریافت: ۱۴۰۴/۰۶/۲۰

بازنگری: ۱۴۰۴/۰۷/۱۴

پذیرش: ۱۴۰۴/۰۷/۲۹

ارائه آنلاین: ۱۴۰۴/۱۰/۳۰

کلیدواژه‌ها:

نهان‌نگاری

شبکه‌های عصبی متخاصم مولد (GAN)

کدگذار-مولد-متمایزکننده (EGD)

تابع هزینه (اتلاف)

آنتروپی متقاطع (BCE)

استناد: اصفهانی، رضا، اخایی، محمدعلی، نوروزی، زین‌العابدین، " نهان‌نگاری مبتنی بر GAN در تصاویر با استفاده از یک معماری EGD"، نشریه پدافند غیرعامل، دوره ۱۶، شماره۴، صفحات ۱۵۴-۱۳۷، ۱۴۰۴. DOI: https://doi.org/10.47176/PD.2026.1574

ناشر: دانشگاه جام امام حسین(ع).

۱- مقدمه

استفاده از هوش مصنوعی، قابلیت‌های مختلفی را به نهان‌نگاری می‌افزاید [۱]. هوش مصنوعی مولد[1] به‌عنوان یک شاخه از هوش مصنوعی، به طراحی و توسعه سیستم‌ها و الگوریتم‌هایی اشاره دارد که قادر به تولید محتوا و داده‌های جدید با استفاده از الگوهای موجود در داده‌های آموزش‌دیده هستند. این سیستم‌ها با استفاده از داده‌های ورودی و الگوریتم‌های خاص، می‌توانند تصاویر، موسیقی، متن، ویدئو و سایر انواع محتواها را خلق کنند.

هدف اصلی این سیستم‌ها، تولید محتوایی با کیفیت و شباهت با محتواهای اصلی است. هوش مصنوعیِ مولد در حوزه‌های بسیاری از صنایع و کسب و کارها استفاده می‌شود. برخی مدل های هوش مصنوعی مولد عبارتند از:

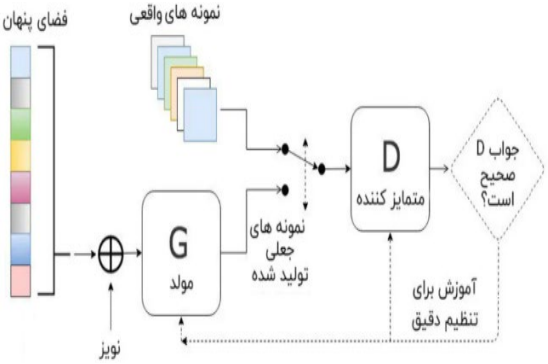
۱. مدل‌های کدگذار خودکار متغیر (VAE[2]): این مدل‌ها با استفاده از شبکه‌های از کدگذار[3] و کدگشا[4]، سعی می‌کنند یک فضای نهان[5] را آموزش‌پذیر کنند، که ویژگی‌های مهم داده‌ها را در خود نگه می‌دارد. سپس با نمونه‌برداری از این فضای نهان، داده‌های جدید تولید می‌شوند.

۲. مدل‌های مبدل[6]: در این مدل‌ها از معماری مبدل که در ابتدا برای ترجمه ماشینی مورد استفاده قرار می‌گرفت، برای تولید داده‌های متنی و تصویری استفاده می‌شود. این مدل‌ها با استفاده از لایه‌های توجه[7]، قادر به تولید داده‌های جدید با ساختار و الگوی مشابه داده‌های واقعی هستند.

۳. مدل‌هایGAN[8] : این مدل‌ها شامل دو بخش اصلی مولد[9] و متمایزکننده[10] هستند. مولد سعی می‌کند داده‌های جدیدی تولید کند که شباهت زیادی به داده‌های واقعی داشته باشند و متمایزکننده سعی می‌کند بین داده‌های تولید شده توسط مولد و داده‌های واقعی تمایز بدهد. آموزش این مدل‌ها به صورت مسابقه بین مولد و متمایزکننده صورت می‌گیرد تا مولد بهبود یابد و داده‌های تولیدی به شکلی واقع‌گرایانه‌تر تولید شوند [۲]. همان‌طور که در شکل (۱) آمده، در GAN، مولد که ورودی

[1] Generative AI
[2] Variational Autoencoders
[3] Encoder
[4] Decoder
[5] Latent Space
[6] Transformer
[7] Attention
[8] Generative Adversarial Networks
[9] Generator
[10] Discriminator

متمایزکننده است، معمولاً یک بردار تصادفی با مقدارهای فضاهای مخفی است که به یک فضای پنهان[11] اشاره می‌کند. فضای پنهان، مجموعه‌ای از بردارهای ورودی به مولد است تا داده‌های مولد تولید شوند. این بردارها معمولاً از توزیع ساده‌ای مثل نرمال استاندارد یا یکنواخت نمونه‌برداری می‌شوند. معمولاً ابعاد فضای پنهان، به اندازه بردار ۱۲۸ رایج است.



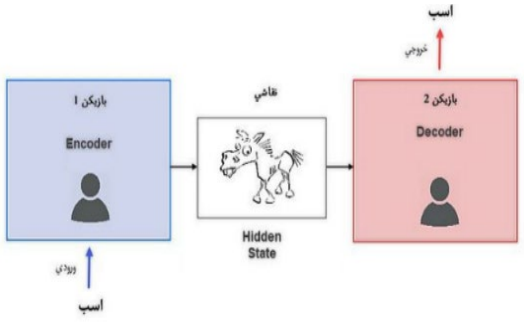
شکل (۱): مدل GAN

مولد که خود یک شبکه عصبی است. دارای ورودی از نوع بردار نویز تصادفی است که وارد شبکه می‌شود. مولد یک تصویر جعلی را تولید می‌کند. متمایزکننده هم یک شبکه عصبی است که دو ورودی دارد؛ تصویر جعلی و تصویر اصلی که یک خروجی از مجموعه داده‌های آموزشی است. شبکه متمایزکننده در خروجی خود خطایی[12] را تولید می‌کند که خطای بین تصاویر جعلی با تصاویر اصلی است. این خطا کمک می‌کند که هم شبکه مولد و هم شبکه متمایزکننده آموزش ببینند. این آموزش از نوع آموزش تکراری[13] است؛ یعنی آموزش مدل به صورت تکراری یا مرحله به مرحله انجام می‌شود. در این روش، مدل چندین بار بر روی مجموعه داده‌ها تمرین می‌کند، این عمل معمولاً با نمونه‌گرفتن‌های مختلف و در چندین دوره آموزشی (اِپُک[14]) صورت می‌گیرد. هدف از هر تکرار[15] ، بهبود وزن‌ها و پارامترهای مدل است تا نتیجه نهایی بهتر و دقیق‌تر باشد. در فرایند آموزش، این تکرارها بر اساس خطای مدل انجام می‌شوند، یعنی مدل در هر تکرار وزن‌ها را تنظیم می‌کند تا خطا کاهش یابد. این روند ادامه دارد تا زمانی که مدل به سطح مطلوبی از دقت برسد یا تعداد تکرارهای تعیین‌شده کامل شود. تعیین تعداد مناسب «تعداد دوره» (اِپُک) در یک مدل GAN یکی از چالش‌های مهم

[11] latent space
[12] Classification Loss
[13] Iteratively Train
[14] Epoch
[15] Iteration

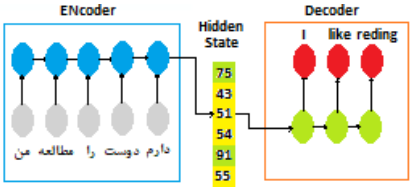
در آموزش این مدل‌هاست، زیرا GANها به راحتی دچار بیش‌برازش[۱]، کم‌برازش[۲] یا فروپاشیِ حالت[۳] می‌شوند و همگرایی آن‌ها پایدار نیست. هرچند نمودارهای اتلاف[۴] (هزینه) درGANها همیشه دقیق نیستند، اما به دلیل ماهیت رقابتی بین مولد و متمایزکننده، می‌توانند روند کلی یادگیری را نمایش دهند و تغییرات کیفی را نشان دهند. اگر در میان تعادل بین مولد و متمایزکننده، یکی خیلی قوی شود (مثلاً اتلاف متمایزکننده[۵] به صفر نزدیک شود)، مدل واگرا می‌شود. اگرLossها نوسان زیادی دارند یا ثابت مانده‌اند، ممکن است نیاز به تنظیم هایپرپارامتر[۶] یا توقف باشد. هایپرپارامتر پارامترهایی هستند که در فرایند آموزش یا تنظیم مدل تعیین می‌شوند و خود تابع یا مدل در طول آموزش آن‌ها را یاد نمی‌گیرد. به عبارت دیگر، این‌ها تنظیمات کلی مدل هستند که با تجربه و آزمایش تعیین می‌شوند (مثال: نرخ یادگیری، تعداد لایه‌ها، اندازه دسته[۷]، طول پوشش داده‌ها، کارکرد تابع فعال‌سازی، وزن‌ها). کاهش تدریجی و پایدار Loss، نشانه همگرایی خوب است. نموداری بد رفتار است که Loss مربوط به متمایزکننده (LossD) به سمت صفر و Loss مربوط به مولد، به سمت مقادیر بالا حرکت کند. در این حالت مولد شکست خورده و متمایزکننده، غالب خواهد شد. اگر Loss مربوط به مولد، به سمت صفر و Loss متمایزکننده به سمت مقادیر بالا حرکت کند. در این حالت مولد فریب داده و اصطلاحاً فروپاشیِ حالت، اتفاق خواهد افتاد. آموزش و یادگیری ناپایدار باعث نوسانات شدید خواهد شد.

علت بهره‌برداری از EGD[۸] در این مقاله، فراهم‌کردن شرایط ارائه توصیف یک تصویر، توسط مدل‌های کدگذار-کدگشا برای یک مدل یادگیری ماشین است [۳]. در این روش، مدل، یک تصویر را به‌عنوان ورودی دریافت کرده و سپس دنباله‌ای از کلمات را به‌عنوان توصیف به خروجی ارائه می کند. مفهوم مدل کدگذار-کدگشا، در بازی پیکشنری[۹] مطابق شکل (۲) قابل درک است.



شکل (۲): مدل بازی پیکشنری

اگر تصویر بالا را به مفاهیم یادگیری ماشین تبدیل شود، شکل (۳) بدست خواهد آمد.



شکل (۳): تبدیل بازی پیکشنری به مفاهیم یادگیری ماشین

این ویژگی امکان خوبی را برای کارهایی مانند؛ ایجاد خودکار زیرنویس برای ویدیوها و سیستم های پرسش و پاسخ فراهم می کند. درک مدل‌های کدگذار-کدگشا برای آخرین پیشرفت‌های پردازش زبان طبیعی (NLP[۱۰]) کلیدی است، زیرا این مدل یکی از موارد اصلی و زیربنای مدل‌های توجه[۱۱] و مبدل‌ها است. تفسیر این‌که چرا معماری سه‌تایی EGD نسبت به برخی مدل‌های تک‌شبکه‌ای، عملکرد بهتری دارد، به دلیل وجود کدگذار قوی برای بازیابی پیام و حضور متمایزکننده با بازنمایی دقیق‌تر از ساختارهای تصویر است. از کدگذار برای افزایش قابلیت بازیابی و استخراج پیام‌های درج‌شده در تصاویر استفاده می‌کند. ترکیب سه مولفه کلیدی EGD در نهان‌نگاری مبتنی بر GAN (مولد برای درج پیام، کدگذار برای استخراج پیام، و متمایزکننده برای اعتبارسنجی تصاویر)، می‌تواند به ایجاد یک سیستم نهان‌نگاری تصویری با کیفیت بصری بالا و قابلیت بازیابی با نرخ بالاتر نسبت به GAN بدون کدگذاری پیام با خطای پایین منجر شود [۴ و ۵]. ورودی تابع کدگذار، تصاویر جعلی است و در بدنه تابع کدگذار یک سری از لایه‌های کانولوشنی با BatchNorm و ReLU[۱۲] وجود دارد که ورودی را به نمایشی با کانال‌های بیشتر و ابعاد کوچکتر تبدیل می‌کند [۶]. در خروجی تابع کدگذار، یک بردار به‌اندازه طول پیام تبدیل می‌شود.

[1] Overfitting
[2] Underfitting
[3] Mode Collapse
[4] Loss
[5] Discriminator Loss (LossD)
[6] Hyperparameter
[7] batch
[8] Encoder-Generator-Discriminator
[9] Pictionary
[10] Natural Language Peoccessing
[11] Attention
[12] Rectified Linear Unit

BatchNorm، نرمالیزه‌کردن فعال‌ها در یک دسته از داده‌ها قبل از عبور به لایه بعد است. برای یک لایه کانولوشنی یا خطی، BatchNorm، مقدار میانگین و واریانس را از نمونه‌های داخل دسته محاسبه می‌کند. مقدار ورودی هر نمونه را با استفاده از رابطه آماری (۱) نرمال می‌شود؛

$$\hat{x} = \frac{x_{batch} - \mu}{\sigma^2_{batch} + \varepsilon} \quad (۱)$$

سپس با دو پارامتر یادگیرنده γ و β به شکل $y = \gamma\hat{x} + \beta$ دوباره مقیاس‌بندی و شیفت صورت می‌گیرد..

ReLU، یک تابع فعال‌سازی غیرخطی است که مطابق رابطه (۲)، مقدار ورودی مثبت را بدون تغییر و مقدار منفی را به صفر تبدیل می‌کند.

$$ReLU(x) = \max(0, x) \quad (۲)$$

تصاویر رنگی RGB (۳ کانال) از سه نقشه ویژگی مستقل قرمز (R)، سبز (G) و آبی (B) تشکیل می‌شود. هر نقشه، یک کانال تصویر است که شدت رنگ مربوطه را نشان می‌دهد. وقتی یک تصویر به مدل‌های یادگیری عمیق داده می‌شود، معمولاً برای یک نمونه واحد به‌صورت (Cin,H,W)=(3,H,W) و برای یک دسته به‌صورت (N,Cin,H,W) نمایش داده می‌شود. مثال: اگر ورودی مدل، یک تصویر رنگی به اندازه ۳۲×۳۲ باشد و از اندازه دسته[1] N استفاده شود، شکل ورودی(N,3,32,32) می‌شود. البته در بعضی مواقع بهتر است به‌جای ReLLU از LeakyReLU استفاده شود. LeakyReLU یک تابع فعال‌سازی است که برای جلوگیری از منفی شدن گرادیان برای مقادیر منفی ورودی، بهبود جریان گرادیان در زیرشبکه‌های عمیق استفاده می‌شود. مطابق رابطه (۳)، به ازای ورودی x اگر $x > 0$، خروجی برابر با x است و اگر $x \leq 0$، خروجی برابر با αx است، که α یک ثابت کوچک است (مثلاً ۰,۰۱).

$$LeakyReLLU(x) = \begin{cases} x. x > 0 \\ \alpha x, x \leq 0 \end{cases} \quad (۳)$$

با توجه به لایه‌های کانولوشنی که قبلا به آن اشاره شد، با فرض مثلاً ورودی تصاویر دارای اندازه ثابت ۱۶×۱۶ ($H_{in} = 16$) با چند کانال، خروجی ابعاد مکانی[2] اولین کانولوشن دو بعدی (Conv2d) با گام[3] S=۲، لایه‌گذاری (padding) P=۱، هسته (kernel) K=۴ به صورت رابطه (۴) محاسبه خواهد شد:

$$\begin{aligned} H_{out} &= 1 + \left\lfloor \frac{H_{in} + 2P - K}{S} \right\rfloor \\ &= 1 \\ &\quad + \left\lfloor \frac{16 + 2 \times 1 - 4}{2} \right\rfloor \\ &= 1 + \left\lfloor \frac{14}{2} \right\rfloor = 1 + 7 \\ &= 8 \end{aligned} \quad (۴)$$

تحلیل فعلی با فرض ورودی تصاویر دارای ابعاد ۱۶×۱۶ با سه کانولوشن (Conv) طبق جدول (۱) است:

جدول (۱): تحلیل ابعاد با ورودی ۱۶×۱۶

ردیف	S	K	P	H_{in}	H_{out}	خروجی ابعاد مکانی
۱	۲	۴	۱	۱۶	۸	۸×۸
۲	۲	۴	۱	۸	۴	۴×۴
۳	۲	۴	۱	۴	۲	۲×۲

با توجه به اندازه ۱۲۸ که برای فضای پنهان رایج است، خروجی کدگذار معمولاً ابعاد را به اندازه بردار فضای پنهان کاهش می‌دهد و فضای ورودی در مثال فوق، به صورت خطی، برابر با 128×2×2=512 خواهد بود.

فرض (N,64,16,16) لایه کانولوشی باشد[4]. N یا همان اندازه دسته، تعداد نمونه‌های ورودی که همزمان پردازش می‌شوند، است. هر نمونه، خروجی خودش را دارد. هر کانال خروجی به طور مستقل نشان دهنده وجود الگوی مشخصی در داده ورودی است (مثلاً لبه‌ها، بافت‌ها، یا ابزارهای پیچیده‌تر که مدل یاد می‌گیرد). 64 یعنی تعداد نقشه‌های ویژگی[5] یا کانال خروجی است. هر کدام یک ویژگی نقشه مستقل است که از ترکیب کانال ورودی و فیلترها به دست می‌آید. در این لایه ۶۴ نقشه ویژگی خروجی خواهیم داشت. ۱۶×۱۶ در واقع ابعاد فضایی هر نقشه ویژگی (ارتفاع و عرض) بعد از عبور از لایهConv2d، برای هر نقشه ویژگی، یک نمای دوبعدی با اندازه ۱۶ پیکسل در هر بعد خواهد بود. به دلیل استفاده از k=4 با s=2 و p=1، ابعاد ۱۶×۱۶ مشخص می‌کند که هر نقشه ویژگی در فضای مکانی با اندازه پایین‌تر از ورودی تولید می‌شود. در لایه کانولوشن (N,64,16,16)، ورودی با اندازه (N,C_{in},H_{in},W_{in})، خروجی با اندازه ($N,C_{out},H_{out},W_{out}$) و لایه خاص Conv2d (32,64,4,2,1) وجود دارد. مثال؛ اگر N=8 باشد، خروجی به شکل (8,64,16,16) است. یعنی از هر نمونه، ۶۴ نقشه ویژگی با ابعاد ۱۶×۱۶ در اختیار

[1] batch size
[2] Spatial
[3] stride
[4] Conv2d (N,64,16,16)
[5] Feature

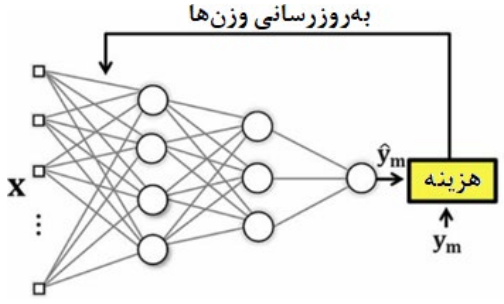
است. برای درک بیشتر، مثال تحلیل ابعاد با ورودی ۳×۳۲×۳۲ نیز در جدول (۲) آورده شده است:

جدول (۲): تحلیل ابعاد با ورودی ۳×۳۲×۳۲

ردیف	S	K	P	H_{in}	H_{out}	خروجی ابعاد مکانی
۱	۲	۴	۱	۳۲	۱۶	۱۶×۱۶
۲	۲	۴	۱	۱۶	۸	۸×۸
۳	۲	۴	۱	۸	۴	۴×۴

فضای ورودی در این مثال، به صورت خطی، برابر با 128×4×4=2048 خواهد شد.

در فرایند کار بایستی از توابع هزینه استفاده کرد زیرا توابع هزینه نقش بسیار مهمی در فرایند آموزش و بهینه‌سازی مدل‌های یادگیری ماشین ایفا می‌کنند. از انتخاب تابع مناسب تا اعمال آن در بهینه‌سازی مدل، هر جزئی از این فرایند می‌تواند بر عملکرد نهایی مدل تأثیر زیادی بگذارد. با درک صحیح و استفاده بهینه از توابع هزینه، می‌توان به مدل‌هایی دقیق‌تر و کارآمدتر دست یافت که نیازهای پیچیده و متنوع دنیای واقعی را برآورده می‌کنند. مطابق شکل (۴)، در هر تکرار آموزش، شبکه عصبی براساس مقادیر وزن‌های خود (دانشی که تا آن لحظه بدست آورده است)، برای داده‌های ورودی، خروجی ای را تخمین میزند، حال برای این‌که متوجه شود که چقدر خوب یا بد عمل کرده است تا براساس آن بتواند وزن‌های خود را تنظیم کند، نیاز به یک تابع هزینه دارد. توابع هزینه در هر تکرار، میزان عملکرد شبکه عصبی را اندازه گیری می‌کنند و شبکه عصبی را جریمه می‌کنند.



شکل (۴): تابع هزینه

هر چقدر عملکرد شبکه خوب باشد، جریمه کمتر و هرچقدر عملکرد بد باشد میزان جریمه بیشتر خواهد بود. توابع هزینه[1] با این جریمه‌ها شبکه عصبی را به سمت یادگیری و حداقل‌کردن[2] هزینه‌های تصمیم‌گیری سوق می‌دهند. در یادگیری ماشین توابع هزینه مختلفی وجود دارد که هر کدام با هدف خاصی طراحی شده اند و خروجی تخمین زده شده شبکه عصبی را با خروجی واقعی مقایسه می‌کنند و یک هزینه‌ای برای شبکه عصبی محاسبه می‌کنند، سپس شبکه عصبی برای حداقل‌کردن این هزینه وزن‌های خود را تنظیم می‌کند. تابع بهینه‌سازی و هزینه‌ها در نظر گرفته شده است. نحوه تصمیم‌گیری الگوریتم‌های یادگیری ماشین برای خوب یا بد بودن یک مدل، استفاده از فرایند تابع هزینه است. این تابع کمک می‌کند تا اختلاف میان مقدارهای پیش‌بینی‌شده و مقدارهای واقعی اندازه‌گیری شود و درنتیجه، مدل‌های بهتری ساخته شود بدون داشتن یک معیار مشخص برای سنجش عملکرد مدل، بهینه‌سازی و بهبود یک مدل یادگیری عمیق تقریباً غیرممکن است.

میانگین مربعات خطا یا (MSE[3]) یکی از رایج‌ترین توابع هزینه است که میانگین مربع خطاهای پیش‌بینی را محاسبه می‌کند. مقیاس MSE مربع واحد اصلی مقادیر برچسب (Label) است. فرمول محاسبه این تابع به‌صورت رابطه (۵) آمده است:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (۵)$$

که در آن، n تعداد داده‌ها است، y_i مقدار واقعی خروجی و $\hat{y}_i$مقدار پیش‌بینی‌شده آن است. همان‌طور که مشخص است، میانگین‌گیری از مربع اختلاف‌ها، بر روی همه نقاط داده‌ها انجام می‌شود.

میانگین مطلق خطا (MAE[4]) میانگین قدر مطلق خطاهای پیش‌بینی را برای مسئله‌های رگرسیون اندازه‌گیری می‌کند. این تابع در مقایسه با MSE کمتر به خطاهای بزرگ حساس است و معمولاً برای مدل‌هایی که به مقاومت دربرابر نویز نیاز دارند استفاده می‌شود. MAE، به‌دلیل سادگی و کارایی بالا، در بسیاری از مسئله‌ها کاربرد دارد و به‌خصوص در مواقعی که داده‌ها نویز دارند مفید است. فرمول محاسبه، به‌صورت رابطه (۶) است:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad (۶)$$

که در آن n تعداد داده‌هاست، y_i مقدار واقعی خروجی و $\hat{y}_i$مقدار پیش‌بینی‌شده آن برای داده i ام است. همان‌طور که از فرمول مشخص است، میانگین‌گیری از قدر مطلق اختلاف‌ها روی همه نقاط داده‌ها انجام می‌شود. لازم به ذکر است که مقیاس

[1] Loss Function
[2] Minimum
[3] Mean squared error
[4] Mean absolute error

MAEهمان واحد اصلی مقدارها برچسب (Label) است.

تابع هزینه Huber[1] ترکیبی از MSE و MAE است. این تابع در صورت کوچک‌بودن خطاها مانند MSE رفتار می‌کند و در صورت بزرگ‌بودن خطاها مانند MAE عمل می‌کند. به‌عبارت دیگر، تابع هزینه Huber از مزیت‌های هر دو تابع MSE و MAE بهره می‌برد و درعین‌حال معایب آن‌ها را کاهش می‌دهد. این تابع برای مسئله‌های رگرسیون که در آن‌ها خطاهای بزرگ نادر ولی مهم هستند بسیار مناسب است. فرمول محاسبه به‌صورت رابطه (۷) است:

$$\begin{cases} \frac{1}{2}(y_i - \acute{y}_i)^2 & for \quad |y_i - \acute{y}_i| \le \delta \\ \delta\left(|y_i - \acute{y}_i| - \frac{1}{2}\delta\right) & for \quad |y_i - \acute{y}_i| > \delta \end{cases} \qquad (۷)$$

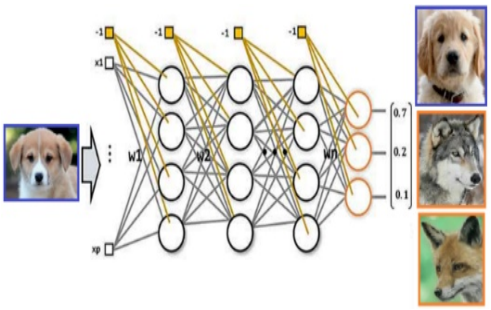
که در آن y_i مقدار واقعی خروجی و $\hat{y}_i$ مقدار پیش‌بینی‌شده آن است. دلتا (δ) نیز پارامتر آستانه (Threshold) نام دارد.

تابع هزینه (تابع اتلاف[2]) «آنتروپی متقاطع» (BCE[3]) است که برای مسئله‌های کلاسه‌بندی باینری استفاده می‌شود، یعنی زمانی که فقط دو کلاس[4] وجود دارد (مثلاً بله یا خیر، درست یا نادرست). به عبارت دیگر برای مسائل طبقه‌بندی دو کلاس برچسب‌های واقعی/جعلی استفاده می‌شود. اگر خروجی مدل احتمال این‌که نمونه به کلاس ۱ تعلق دارد باشد، BCE به‌صورت رابطه (۸) عمل می‌کند:

$$BCE = \frac{1}{n}\sum_{i=1}^{n}[y_i \log(\acute{y}_i) + (1 - y_i)\log(1 - \acute{y}_i)] \qquad (۸)$$

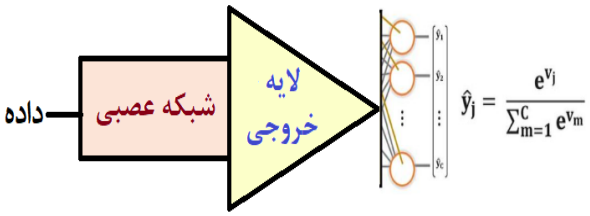
y_i نشان‌دهنده برچسب واقعی و $\acute{y}_i$ احتمال پیش‌بینی‌شده توسط مدل برای داده i ام است، علامت سیگما روی n یعنی جمعا nداده وجود دارد. هدف از استفاده این فرمول کمینه‌کردن اختلاف میان برچسب‌های واقعی و احتمالات پیش‌بینی شده است. در این‌جا به چند تابع هزینه کاربردی که در مقاله استفاده شده، اشاره می‌شود.

آنتروپی متقاطع، تابع هزینه‌ای است که عدم شباهت بین توزیع احتمال خروجی های تخمین زده شده شبکه عصبی و خروجی های واقعی را محاسبه می‌کند. یعنی همانند شکل (۵)، وقتی داده مربوط به کلاس یک هست، شبکه با سطح اطمینان بالایی کلاس یک تشخیص دهد.



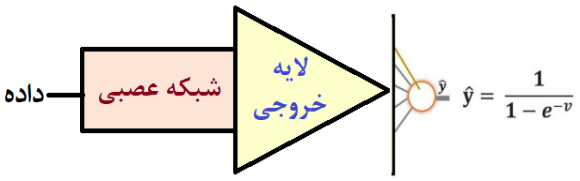
شکل (۵): مثال تابع هزینه آنتروپی متقاطع

از آنجا که این تابع هزینه عدم شباهت دو توزیع احتمال را بررسی می‌کند، برای همین لازم است که خروجی تخمین زده شده، همانند یک توزیع احتمال باشد. از دو تابع فعال سیگموئید و SoftMax می‌توان برای این تابع هزینه استفاده کرد. در مسائل چندکلاسه، از تابع فعال SoftMax استفاده می‌شود، چرا که سیگموئید به ازای هر نورون یک خروجی بین صفر تا یک در نظر می‌گیرد و کاری با خروجی بقیه نورون‌ها ندارد، در حالی که باید جمع همه خروجی ها یک باشد. برای همین در مسائل چندکلاسه، از تابع فعال SoftMax استفاده می‌شود. تابع فعال SoftMax در مسائل دو کلاسه، یا به اصطلاح باینری، یک نورون در لایه خروجی استفاده می‌شود و برای همین از سیگموئید می‌توان همراه با تابع هزینه cross-entropy استفاده کرد. در شکل (۶) تابع فعال برای مسائل چند کلاسه نشان داده شده است.



شکل (۶): تابع فعال برای مسائل چند کلاسه

در تابع فعال سیگموئید، خروجی سیگموئید عددی بین صفر و یک هست، و میزان احتمال وقوع x به کلاس یک را مشخص میکند. یعنی اگر خروجی ۰٫۸ باشد، به معنی این است که احتمال نمونه ورودی به کلاس یک ۸۰ درصد و به کلاس صفر ۲۰ درصد است. در شکل (۷) تابع فعال برای مسائل تک کلاسه نشان داده شده است.



شکل (۷): تابع فعال برای مسائل تک کلاسه

[1] Huber Loss
[2] Loss Function
[3] Binary Cross-Entropy
[4] Class

اگر مقایسه‌ای بین توابع هزینه مطرح شده انجام شود، باید بیان کرد با توجه به مثال زیر، میبینیم تابع هزینه Cross-Entropy طبق انتظاری که می‌رفت، جریمه ها را محاسبه کرده است. ولی از آن‌جا که هدف MSE روی حداقل‌کردن اختلاف است، برای دو نمونه دوم و سوم هزینه متفاوتی محاسبه می‌کند. به همین دلیل است که Cross-Entropy برای مسائل طبقه‌بندی تابع هزینه مناسب‌تری نسبت به MSE به شمار می‌رود و البته MSE رایج‌ترین تابع هزینه در مسائل رگرسیون است. مثالی در جدول (۳) به این موضوع اشاره دارد.

جدول (۳): تحلیل مقایسه MSE و Cross-Entropy

y_{true}= [[1,0,0],[0,1,0],[0,0,1]] y_{pred}= [0.7,0.2,0.1],[0,0.5,0.5],[0.3,0.2,0.5]]	**Cross-Entropy:** $cost_1 = -[1 * \log(0.7) + 0 * \log(0.2) + 0 * \log(0.1)] = 0.35$ $cost_2 = -[0 * \log(0) + 1 * \log(0.5) + 0 * \log(0.5)] = 0.69$ $cost_3 = -[0 * \log(0.3) + 0 * \log(0.2) + 1 * \log(0.5)] = 0.69$ **Squared Error:** $cost_1 = [(1-0.7)^2 + (0-0.2)^2 + (0-0.1)^2] = 0.14$ $cost_2 = [(0-0)^2 + (1-0.5)^2 + (0-0.5)^2] = 0.5$ $cost_3 = [(0-0.3)^2 + (0-0.2)^2 + (1-0.5)^2] = 0.38$	MSE: tensor([0.1400, 0.5000, 0.3800]) Cross Entropy: tensor([0.3567, 0.6931, 0.6931])

همچنین به‌رغم مزیت‌های Huber، فرمول‌های MSE و MAE بسیار ساده‌تر هستند؛ به‌همین‌دلیل، محاسبات آن‌ها در مقایسه با Huber سریع‌تر است، به‌ویژه در مجموعه داده‌های بزرگ؛ علاوه بر این Huber به تنظیم یک پارامتر اضافی به‌نام آستانه (δ) نیاز دارد که ممکن است به تنظیم دستی و بهینه‌سازی بیشتری نیاز داشته باشد، درحالی‌که MSE و MAE چنین نیازی ندارند. برای داده‌های دارای نویز زیاد، استفاده از توابع هزینه‌ای مانند MAE یا Huberکه کمتر به خطاهای بزرگ حساس هستند، می‌تواند مفید باشد. این توابع به مدل کمک می‌کنند تا مقاومت بیشتری دربرابر نویز داشته باشند و عملکرد بهتری ارائه کنند. MSE به خطاهای بزرگ حساس‌تر است و به‌دلیل محاسبه مربع خطاها، خطاهای بزرگ را بیشتر وزن می‌دهد، درحالی‌که MAE به خطاهای کوچک و بزرگ به‌طور مساوی پاسخ می‌دهد و میانگین قدر مطلق خطاها را اندازه‌گیری می‌کند.

نسبت مقدار سیگنال به مقدار نویز (SNR) در این مقاله، نسبت تصویر واقعی (سیگنال) به تصویری نهان‌نگاری‌شده (نویز) در نظر گرفته شده است [۷]. فرضاً یک تصویر اصلی I وجود دارد و یک تصویر نهان‌نگاری‌شده I' که در آن اطلاعات مخفی‌اند. SNR طبق رابطه (۹) محاسبه می‌شود:

$$SNR = 10log_{10}\left(\frac{\sum_{i,j} I^2(i,j)}{\sum_{i,j}(I(i,j) - I'(i,j))^2}\right) \quad (۹)$$

۲- روش تحقیق

۲-۱- هدف

هدف در این مقاله، نهان‌نگاری با حفظ مناسب اتلاف مولد، اتلاف متمایزکننده و سیگنال به نویز در کنار هدف اصلی که سطح عالی مقدار خطای محاسبه شده پیام، بین پیام اصلی و پیام استخراج‌شده توسط کدگذار است.

۲-۲- کارهای مرتبط پیشین

یکی از اولین و پایه‌ای‌ترین مدل‌های GAN برای نهان‌نگاری تصویر مرجع [۸] است. نهان‌نگاری عمیق در این مرجع صورت می‌گیرد. پیام درون تصاویر پنهان می‌شود تا کیفیت بصری حفظ گردد. ویژگی‌های کلیدی در این مقاله، استفاده از شبکه‌های مولد و تفاضل‌برداری برای پنهان‌سازی پیام، سازگاری با طیف وسیعی از تصاویر، مقاومت نسبتاً مطلوب در برابر تشخیص توسط مدل‌های ساده ارائه شده است. مقاله [۹] نشان‌گذاری مبتنی بر شبکه های عصبی است. نشان‌گذاری ترکیبی CNN-GAN بوده که ترکیب قدرت استخراج ویژگی CNN و قابلیت تولیدGAN ، منجر به سیستم‌های مقاوم‌تر می‌شود. این رویکرد در چارچوب EGD می‌تواند به بهبود قابلیت تشخیص و پنهان‌سازی هم‌زمان بیانجامد. مقاله [۱۰] نشان‌گذاری مبتنی بر GAN است. نشان‌گذاری مبتنی بر مکانیزم توجه (Attention) را که دارای SNR مناسب و خطای پیام کمی است را طرح کرده است. جدیدترین پیشرفت‌ها در نهان‌نگاری، استفاده از مکانیزم‌های توجه برای تمرکز بر مناطق مهم‌تر تصویر (که تغییر در آنها کمتر محسوس است) می‌باشد. نشان می‌دهد را ارائه می‌دهند. این روند به‌ویژه برای تصاویر با چگالی زیاد اطلاعات و نیازمند حفظ

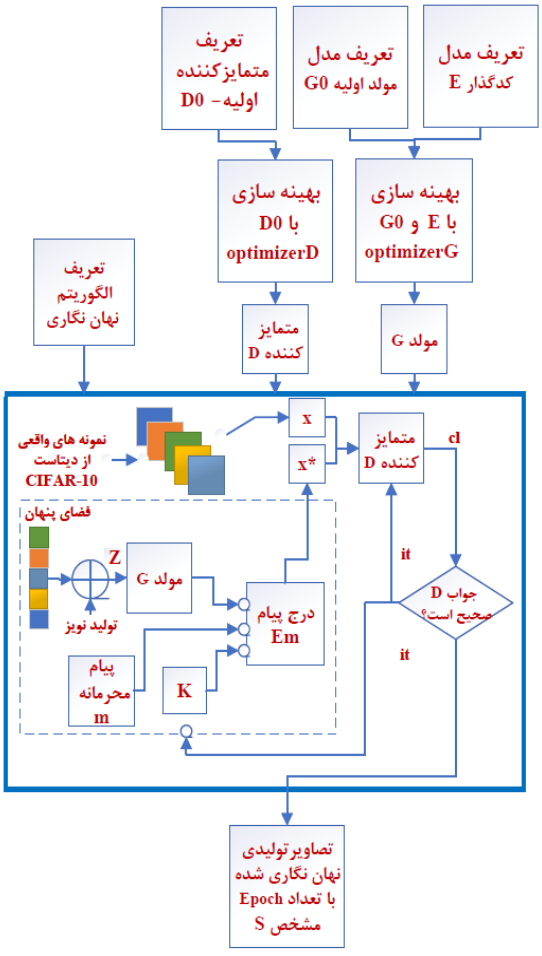
کیفیت بصری حیاتی است و با معماری EGD همخوانی دارد. در مقاله [۱۱] موضوع امنیت در نهان نگاری مبتنی بر مدل‌های هوش مصنوعی مطرح شده است.

۲-۳- روش پیشنهادی

چارچوب حل مسئله به شکل ترکیب یک مولد برای درج پیام (یک کانال کانولوشنی طراحی شده برای درج پیام در تصویر با ورودی ترکیبی از نویز و پیام بیت-سری و خروجی تصویر ۳۲×۳۲×۳ با تابع فعال tanh)، کدگذار برای استخراج پیام (شبکه‌ای برای استخراج بیت‌های پیام از تصویر نهان‌نگاری‌شده با خروجی یک بردار از اندازه MSG_LEN با استفاده از تابع سیگموئید[1]، و متمایزکننده برای اعتبارسنجی صحت تصاویر نهان‌نگاری‌شده و طبیعی بودن تصاویر (شبکه‌ای برای تمایز بین تصاویر واقعی و نهان‌نگاری‌شده، با معماری عمیق با کانولوشن‌های متعدد و خروجی مقدارهای احتمال از طریق تابعی از PyTorch به نام BCEWithLogitsLoss)، با استفاده از CIFAR-10 به‌عنوان دیتاست ورودی/ ارزیابی پیشنهاد شده است. همچنین از هایپرپارامترهای (یا همان تنظیماتی که پس از طراحی مدل تعیین می‌شوند تا فرایند یادگیری بهتری داشته باشد)، اندازه دسته، تعداد تکرارها برای به‌روزرسانی‌های G و D، وزن‌های منظم‌سازی[2]، نوع تابع هزینه برای D و G در این مقاله به‌کار رفته است. تابع BCEWithLogitsLoss دو عملیات را به طور ترکیبی انجام می‌دهد؛ به عبارتی هم لاگیت‌ها[3] را با استفاده از سیگموئید به مقادیر احتمالی بین ۰ و ۱ تبدیل می‌کند و هم تابع هزینه «باینری آنتروپی متقاطع» را روی خروجی مقادیر احتمالی و برچسب‌های هدف محاسبه می‌کند. این ترکیب باعث پایداری عددی[4] بهتر و جلوگیری از مشکلات محاسباتی می‌شود که ممکن است در استفاده از Softmax + BCE به وجود آید. وقتی مدل‌های عمیق آموزش داده می‌شوند، گرادیان‌های تابع هزینه نسبت به وزن‌ها می‌تواند هنگام بازگشت به سمت ورودی‌ها خیلی کوچک[5] یا خیلی بزرگ[6] شود و در نتیجه آموزش کند یا ناپایدار می‌شود و شبکه ممکن است بهبود ناچیزی داشته باشد یا به «نویز» یا ناپایداری وزن‌ها برسد. برای وزن‌گذاری و بهبودهای

[1] sigmoid
[2] Regularization
[3] logits
[4] numerical
[5] vanishing
[6] exploding

معماری، از معماری «ارتباطات باقیمانده[7]» و معماری «سایر اتصالات انتقالی[8]» در کدگذار برای تولید تصاویر با کیفیت‌تر استفاده شده است. معماری «ارتباطات باقیمانده[9]»، معماری است که در آن خروجی یک بلوک به ورودی همان بلوک اضافه می‌شود تا سیگنال‌ها از لایه‌های عمیق به لایه‌های سطح پایین‌تر منتقل شوند. لایه‌های عمیق به لایه‌های سطح پایین‌تر منتقل شوند. فرم رایج یک بلوک باقیمانده[10] به‌صورت $y = x + F(x,\{W_i\})$ است، جایی که $F(x,\{Wi\})$ تابع مجموعه‌ای از کانولوشن‌ها و $\mathbf{x}$ ورودی بلوک است. معماری «سایر اتصالات انتقالی»، مسیری مستقیم از ورودی یا خروجی یک لایه/ بلوک به لایه‌های بالاتر یا به ورودی بلوک‌های بعدی می‌سازد. فلوچارت روش پیشنهادی در شکل (۸) نمایش داده شده است:



شکل (۸): روش پیشنهادی «نهان‌نگاری مبتنی بر GAN در تصاویر با استفاده از یک معماری EGD»

[7] residual connections
[8] skip connections
[9] residual connections
[10] Residual Block

در شکل (۸)، الگوریتم نهان‌نگاری می‌تواند هر الگوریتمی در حوزه تبدیل یا مکان تعریف شود. در این مقاله برای سادگی کار از الگوریتم حوزه مکان درج (Em[1]) در بیت‌های کم ارزش (LSB[2]) استفاده شده است. نام تابع در رابطه (۱۰) آمده است.

$$Em_{Steg}\left(LSB_{image}, m, K\right) \qquad (۱۰)$$

با بهینه‌سازی مدل کدگذار (E) و مولد اولیه (G0)، مولد بهینه (G) حاصل می‌شود. همچنین از بهینه‌سازی مدل متمایزکننده اولیه (D0)، بهینه متمایزکننده (D) حاصل می‌شود.

در این مقاله اندازه تصاویر (image_size) ۳۲×۳۲ پیکسل، تعداد کانال ورودی (in_channels) برابر با ۳ یا همان RGB (برای grayscale مقدار in_channels=1) و اندازه بردار فضای پنهان (latent_dim یا اندازه بردار Z) همان مقدار رایج ۱۲۸ در نظر گرفته شده است. در روش پیشنهادی، فرایند هر بخش از EGD برای ورودی ۳۲×۳۲ مطابق سه جدول (۴)، (۵) و (۶) است. LeakyReLU برای D و ReLU برای G به همراه BatchNorm، معمولاً نتیجه خوبی می‌دهد.

مراحل کدگذار (E)، طبق جدول (۴) است:

جدول (۴): مراحل کدگذاری

ابعاد خروجی پس از هر مرحله کانولوشن	لایه‌های کانولوشنی
مرحله اول: ۳۲×۳۲ با خروجی ۶۴ کانال ⟵ ۱۶×۱۶	Conv2d (in_channels, 64, 4, 2, 1) ⟵ LeakyReLU
مرحله دوم: ۱۶×۱۶ با خروجی ۱۲۸ کانال ⟵ ۸×۸	Conv2d (64, 128, 4, 2, 1) ⟵ BatchNorm2d ⟵ LeakyReLU
مرحله سوم: ۸×۸ با خروجی ۲۵۶ کانال ⟵ ۴×۴	⟵Conv2d (128, 256, 4, 2, 1) ⟵ BatchNorm2d ⟵ LeakyReLU
مرحله چهارم: ۴×۴ با خروجی latent_dimکانال Flatten[3] ⟵ به اندازه 1×1×latent_dim یا قبل از Flatten به صورت مناسب	Conv2d (256, latent_dim, 4, 2, 1) ⟵ Flatten
خروجی کدگذار به صورت بردار فضای پنهان می‌شود:	*Optional: Linear(latent_dim, latent_dim)* یا *BatchNorm*

[1] Embedded
[2] Least Significant Bit
[3] در زمینه شبکه‌های عصبی کانولوشنی خصوصاً ویرایش‌های مولد مانند GANs، تبدیل خروجی چند بعدی از لایه‌های کانولوشنی به شکل یک بردار واحد است که بتواند به عنوان ورودی به لایه‌های کاملاً متصل (Dense) استفاده شود. Flatten یعنی «بردار کردن» یا به عبارت دقیق‌تر، تغییر ابعاد یک تنسور چندبعدی به یک بردار یک‌بعدی یا بردارهای با ابعاد مشخصی که برای ادامه مدل مناسب باشند.

کدگذار، چند لایه Conv2d با BatchNorm و LeakyReLU ، کاهش طول و عرض با گام (stride) یا pooling مشخص، تا رسیدن به یک فضای پنهان، پیش خواهد برد. مراحل مولد (G)، طبق جدول (۵) است:

جدول (۵): مراحل مولد

ابعاد خروجی پس از هر مرحله کانولوشن	لایه‌های کانولوشنی
	reshape to ⟵ Linear (latent_dim, 256,4,4) (batch,256,4,4)
از ۴×۴ به ۸×۸	ConvTranspose2d (256, 128, 4, 2, 1) ⟵ BatchNorm ⟵ ReLU
۸×۸ به ۱۶×۱۶	ConvTranspose2d (128, 64, 4, 2, 1) ⟵ BatchNorm ⟵ ReLU
۱۶×۱۶ به ۳۲×۳۲	ConvTranspose2d (64, in_channels, 4, 2, 1) ⟵ Tanh یا Sigmoid[4] ⟵
خروجی نهایی باید با محدوده داده‌شده تعیین‌شده همخوانی داشته باشد (مثلاً با Tanh برای [-1,1] یا Sigmoid برای [0,1])	

مولد، چند ConvTranspose2d یا Upsampling+Conv برای بازسازی به اندازه ورودی، خواهد داشت. مراحل متمایزکننده (D)، طبق جدول (۶) است:

جدول (۶): مراحل متمایزکننده

ابعاد خروجی پس از هر مرحله کانولوشن	لایه‌های کانولوشنی
پس از سه Conv2d با stride=2 و padding=1، ابعاد به ۴×۴ می‌رسد (برای ورودی ۳۲×۳۲).	Conv2d (in_channels, 64, 4, 2, 1) ⟵ LeakyReLU
	Conv2d (64, 128, 4, 2, 1) ⟵ BatchNorm2d ⟵ LeakyReLU
	Conv2d (128, 256, 4, 2, 1) ⟵ BatchNorm2d ⟵ LeakyReLU
سپس Flattened به یک مقدار واحد برای تصمیم‌گیری در متمایزکننده می‌رود.	Linear (25644, 1) ⟵ Flatten ⟵ Sigmoid یا بدون sigmoid بسته به loss

تولید نویز (Z) حاصل از بردار نویز تصادفی، ورودی مولد است که وارد شبکه می‌شود. مطابق رابطه (۱۱) تصویر جعلی (x*) خروجی مولد است. x* و x (تصاویر اصلی) دو ورودی‌های متمایزکننده هستند.

$$\left.\begin{matrix} x = DB_{CIFAR-10} \\ x^* = Em_{Steg}(G(Z), m, K)\big|_{it_n(cl(D))} \end{matrix}\right\} \rightarrow D(x, x^*)\big|_{it_n(cl(D))} \qquad (۱۱)$$

در خروجی متمایزکننده، طبق رابطه (۱۲)، خطای (cl) بین تصاویر جعلی با تصاویر اصلی به شکل آموزش تکراری (it) و مرحله به مرحله تولید می‌شود. برای کاهش خطا، در هر اِپُک، وزن‌ها تنظیم می‌شود. تعداد اِپُک‌ها تا زمانی که مدل به سطح

[4] سیگموئید

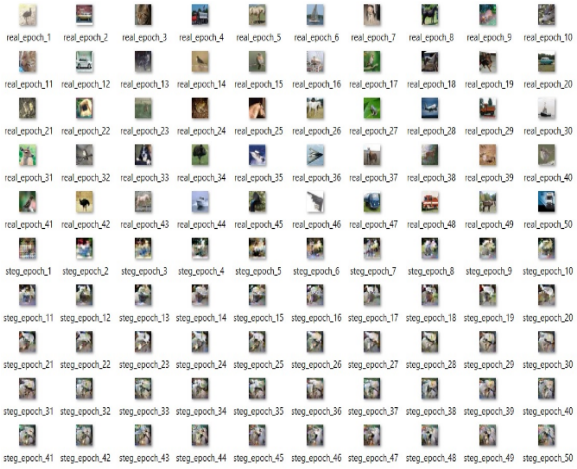
مطلوبی از دقت برسد تکرار می‌گردد.

$$cl(D) = \begin{cases} \frac{1}{2}(x-x^*)^2 & for \quad |x-x^*| \le \delta \\ \delta\left(|x-x^*| - \frac{1}{2}\delta\right) & for \ |x-x^*| > \delta \end{cases} \rightarrow Steg_{epoch_n}\left(cl(D)\lessgtr\delta\big|_{\delta=0.8}\right) \quad (۱۲)$$

۳- نتایج، بررسی و مقایسه

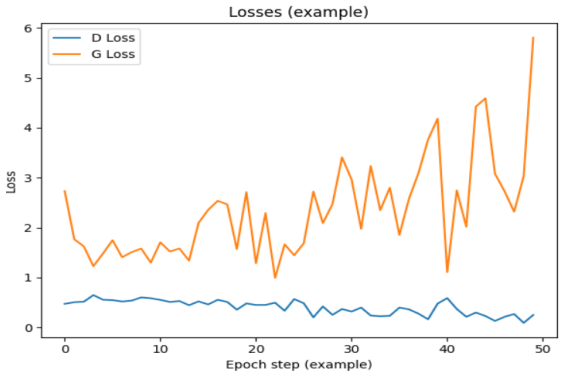
۳-۱- بررسی روش پیشنهادی

در روش پیشنهادی، کلمه «HELLO» که شامل ۴۰ بیت است، با یک الگوریتم درج مکانی LSB، در تصاویر تولیدشده ۳۲×۳۲ (شامل ۱۰۲۴ بیت)، نهان‌نگاری می‌شوند. البته به‌خاطر محدودیت‌های نوع رایانه و زمان‌بر بودن فرایند اجرای کد برنامه روش پیشنهادی، از دیتاست CIFAR-10 و تصاویر ۳۲×۳۲ و پنجاه اِپُک استفاده شده است که نتایج خروجی نهان‌نگاری، خیلی عملیاتی نبوده و بهتر است از دیتاست‌های با تصاویر بزرگ‌تر استفاده کرد[۱]. با توجه به شبیه‌سازی نرم‌افزاری روش پیشنهادی برای پنجاه اِپُک، یکصد تصویر قبل از نهان‌نگاری (real_epoch_n) و بعد از نهان‌نگاری (steg_epoch_n) را مطابق شکل (۹) تولید خواهد شد:



شکل (۹): یکصد تصویر قبل از نهان‌نگاری (real_epoch_n) و بعد از نهان‌نگاری (steg_epoch_n) با پنجاه اِپُک

اگر CIFAR-10 به درستی بارگذاری و پیش‌پردازش شود و مدل‌ها به‌درستی همگرایی کنند، انتظار می‌رود LossD اتلاف (هزینه) متمایزکننده بالایی در ابتدا و کاهش تدریجی با بهبود مدل مولد/کدگذار وجود دارد و LossG اتلاف مولد، نیز با بهبود گراف به سمت مقدار پایینی میل می‌کند (شکل ۱۰). LossD به‌صورت خطوط آبی نمایش داده شده است. در اکثر مقاطع مقدار آن نسبتاً کوچک و پایدار است و تغییرات زیادی ندارد. این نشان می‌دهد که متمایزکننده نسبتاً ثابت کار می‌کند و تفاوت بین تصاویر مخفی‌شده و تصاویر عادی را نسبتاً خوب تشخیص می‌دهد.

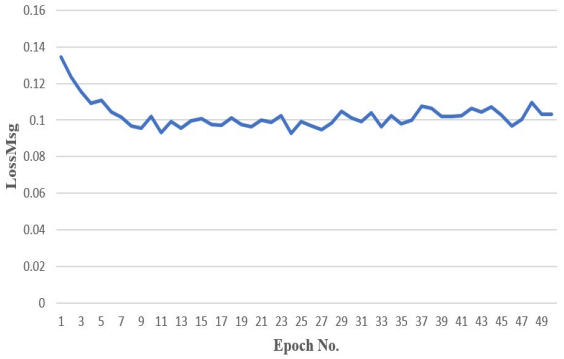


شکل (۱۰): LossD و LossG روش پیشنهادی برای ۵۰ اِپُک از دیتاست CIFAR-10

LossG به‌صورت خطوط نارنجی نمایش داده شده است. در ابتدا، مقدار LossG بالا است، اما با پیشروی آموزش به‌طور کلی کاهش می‌یابد و در نهایت به محدوده‌ای کمتر از ۲ تا ۳ می‌رسد و در برخی اِپُک‌ها حتی افزایش شدیدی دارد (مثل آخرین اِپُک‌ها که با ظهور نوسان شدید همراه است). وقتی LossG در طول آموزش افزایشی می‌شود یا نوسان می‌کند، ممکن است نشان‌دهنده این باشد که مولد در آن بازه از آموزش به مشکلاتی برخورده است؛ مثلاً قدرت متمایزکننده بیش از حد، مشکلات همگرایی دارد. تغییرات کوچک و کاهش تدریجی LossG نسبت به LossD، به‌خصوص در فازهای پایانی آموزش، می‌تواند نشان‌دهنده توازن مناسب بین مولد و متمایزکننده باشد. در *تفسیر دقیق‌تر براساس شکل* (۱۰)، LossD از اِپُک ۰ تا حدود ۲۰، نسبتاً ثابت باقی می‌ماند و LossG نیز کمتر از مقدار اولیه می‌شود، که نشان می‌دهد همگرایی نسبی رخ داده است. LossG از اِپُک‌های بعدی تا ۵۰ به‌طور کلی در حال افزایش یا نوسان شدید است و LossD نسبتاً ثابت باقی می‌ماند یا با نوسان‌های کم همراه است. این می‌تواند نشان‌دهنده ناکارآمدی در به‌روزرسانی‌های اخیر یا فشار بیش از حد متمایزکننده باشد. در کل LossD، پایین‌تر و نسبتاً ثابت است؛ LossG بالاتر و با نوسان است، که نشان می‌دهد در این بازه، متمایزکننده از مولد پیشی گرفته است یا مولد در حال بهبود آگاهی (یادگیری) است اما هنوز به تعادل نهایی نرسیده است.

[۱] معمولا ماکزیمم تعداد اِپُک برای دیتاست CIFAR-10، ۱۰۰ الی ۵۰۰ است.

LossMsg مقدار خطای محاسبه شده پیام بین پیام اصلی و پیام استخراج‌شده توسط کدگذار است. مقدار LossMSG پایین‌تر معمولاً به معنای بهبود کیفیت بازگردانی و/یا کاهش مقدار خطا در استخراج پیام مخفی‌شده است. همان‌طور که در شکل (۱۱) مشاهده می‌شود با هر اِپُک، LossMsg، کاهش می‌یابد و به این معنا است که کدگذار بهتر از تصاویر نهان‌نگاری‌شده پیام را استخراج می‌کند.

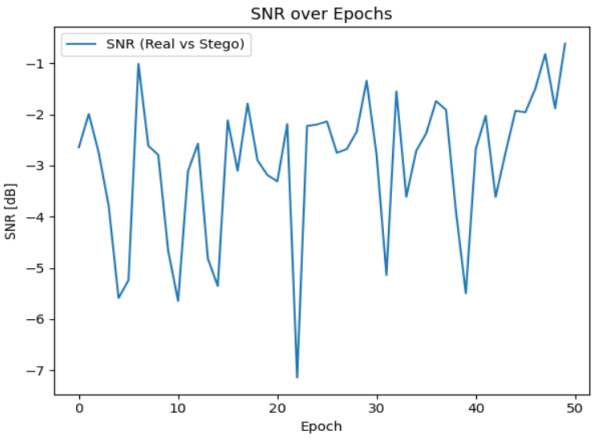


شکل (۱۱): LossMsg روش پیشنهادی برای ۵۰ اِپُک از دیتاست CIFAR-10

معیار LossMsg معیاری شبیه خطای بازسازی[1] در مدل‌های کدگذارهای خودکار[2] یا نرخ خطای بیت (BER[3]) در ارتباطات دیجیتال است؛ اما در این‌جا به صورت Loss پیام نمایش داده می‌شود.

در این مثال بهترین حالت و کمترین خطا در اِپُک ۲۵، روی داده است. مقدار LossMsg کمتر به معنی بازنویسی پیام دقیق‌تر است (در پیاده‌سازی روش پیشنهادی، کدگذار، پیام را بهتر استخراج می‌کند یا کدی که پیام را در تصویر می‌سازد بهتر از نویزها عبور می‌کند. در نمودار ارائه‌شده، LossMSG از مقدار اولیه بالاتر (۰/۱۴–۰/۱۲) به مرور زمان به سمت حدود ۰/۰۹–۰/۱۱ می کاهد و در اکثر اِپُک ها به ناحیه نسبتاً پایداری حول ۰/۰۹–۰/۱۱ می‌رسد. نوسانات کوچک در طول اِپُک‌ها نشان‌دهنده تغییرات جزئی در هر اِپُک یا تغییرات بین لایه‌های مدل است. در ابتدا روند کاهش، بهبود اولیه در مدل و کاهش خطا با ادامه آموزش وجود دارد. پس از اِپُک ۱۰ به‌بعد، پایداری و دامنه نوسان کم، که نشان می‌دهد مدل به حالت همگرایی رسیده یا به سطح ثابتی از عملکرد رسیده است. مقدار LossMSG حدوداً کمتر از ۰/۱ در اکثر اِپُک‌ها است که معمولاً نزدیک به هدف مطلوب در بسیاری از پیاده‌سازی‌های نهان‌نگاری GAN می‌باشد.

SNR بین تصویر واقعی و نهان‌نگاری‌شده، با پیشرفت آموزش، افزایش می‌یابد، زیرا درج پیام با نویز و تغییرات کمتر انجام می‌شود. مقدار SNR بالا به معنی تغییرات کم‌تر و بازنویسی پیام دقیق‌تر است. ممکن است به دلیل مقداردهی اولیه، ناپایداری GAN یا عدم تعادل آموزشی، Lossها گاهی ناپایدار باشند یا SNR به طور یکنواخت افزایش نیابد. در چنین مواقعی پیشنهاد می‌شود از تکنیک‌های پایداری مانند WGAN-GP[4] جهت بهبود پایداری آموزش GAN و کاهش مشکلاتی مانند از بین رفتن گرادیان یا فروپاشیِ حالت استفاده کرد.



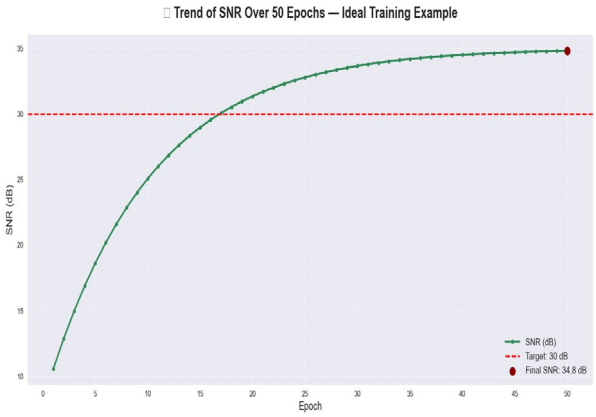
شکل (۱۲): SNR روش پیشنهادی برای ۵۰ اِپُک از دیتاست CIFAR-10

همان‌طور که در شکل (۱۲) مشاهده می‌شود، SNR به شکل قابل توجهی منفی است (dB 1- تا dB 7-) و در بسیاری از اِپُک‌ها افت و فرورفتگی‌های عمیقی دارد. این نشان می‌دهد که برخی روش‌های GAN در برخی کارهای نهان‌نگاری تصویری ممکن است با SNR پایین‌تر نسبت به نمونه‌های مرجع کار کند.

۳-۲- بررسی روش پیشنهادی در شرایط بهینه

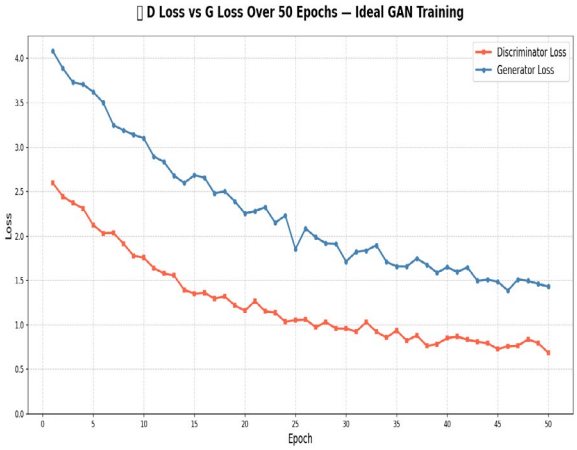
اگر در یک شبیه‌سازی دقیق‌تر و شرایط بهینه، با مولد پنجاه تصویر، حاصل از ۵۰ اِپُک، یک تک بیت در LSB یکی از پیکسل‌های هر تصویر درج شود، SNR از حدود dB ۸ شروع و به حدود dB ۳۴/۹ می‌رسد (شکل ۱۳). منحنی، نشان‌دهنده بهبود تدریجی بوده و خط‌چین، هدف dB ۳۰ را نشان می‌دهد. نقطه انتهای سمت راست نمودار (بر روی Epoch ۵۰)، نشان‌دهنده آخرین مقدار SNR یعنی حدود dB ۳۴/۹ است.

[1] Reconstruction Error
[2] Autoencoder
[3] Bit Error Rate
[4] Wasserstein GAN - Gradient Penalty



شکل (۱۳): SNR با درج فقط یک تک بیت در LSB یکی از پیکسل‌های هر تصویر ۳۲×۳۲ پیکسل به روش پیشنهادی

همچنین در این حالت شکل (۱۴) را برای LossD و LossG حاصل خواهد شد.

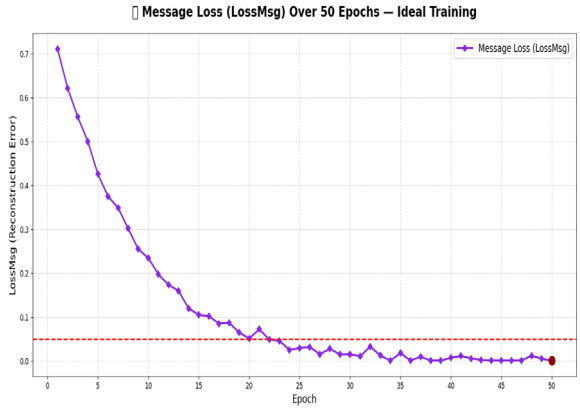


شکل (۱۴): LossD و LossG با درج فقط یک تک بیت در LSB یکی از پیکسل‌های هر تصویر ۳۲×۳۲ پیکسل به روش پیشنهادی

همان‌طور که در یک آموزش پایدار و موفق GAN انتظار می‌رود، LossD و LossG باید در تعادل باشند؛ نه این‌که یکی به صفر و دیگری به بی‌نهایت برود. همگرایی نسبی در Epoch های آخر، Lossها نوسان کمی دارند و حول یک مقدار ثابت می‌مانند. طبق شکل (۱۴)، نوسانات کوچک طبیعی است، اما نباید واگرا یا پله‌ای باشد. هیچ‌کدام به صفر یا بی‌نهایت نمی‌رسند که نشانه تعادل بین مولد و متمایزکننده و عدم تسلط [1] یکی بر دیگری است. مقادیر انتهایی در اِپُک ۵۰، برای LossD حدود ۰/۷ و LossG حدودا ۱/۲ بوده که مقادیر مناسبی برای آنتروپی متقاطع (BCE) است.

مطابق شکل (۱۵) معیار LossMsg را برای این مثال در حالت دقیق نمایش داده می‌دهد.

[1] Dominance



شکل (۱۵): LossMsg با درج فقط یک تک بیت در LSB یکی از پیکسل‌های هر تصویر ۳۲×۳۲ پیکسل به روش پیشنهادی

کاهش تدریجی و پایدار، نشانه یادگیری صحیح است. همان‌طور که انتظار می‌رود، ویژگی‌های یک نمودار خوب برای LossMsg در طول پنجاه اِپُک از مقدار بالا (مثلاً ۰٫۵ یا ۱٫۰) شروع می‌شود چون در ابتدا بازسازی ضعیف است. همگرایی به مقدار بسیار کوچک (مثلاً بیشتر از ۰/۰۵ یا حتی حدود ۰٫۰۱) به معنی دقت بالای بازیابی پیام است. بدون نوسان شدید، نشانه ثبات در آموزش خواهد بود. همچنین در Epochهای آخر، تغییرات ناچیز، نشانه همگرایی خواهد بود. پنج مقدار عددی به‌صورت نمونه در در جدول (۷) آمده است که در اِپُک ۵۰، مقدار LossMsg≅۰/۰۳ به‌دست آمده که در سطح عالی محسوب می‌شود.

جدول (۷): پنج مقدار عددی LossMsg به‌صورت نمونه از اِپُک‌ها

Epoch	۱	۱۰	۲۵	۴۰	۵۰
چند نمونه LossMsg	~۰/۷۸	~۰/۲۵	~۰/۰۸	~۰/۰۴	~ ۰/۰۳ (عالی)

۳-۳- مقایسه روش پیشنهادی با سایر روش‌ها

روش پیشنهادی در مقایسه با روش‌های استاندارد معمولی نهان‌نگاری حوزه مکان [۱۲] و نهان‌نگاری حوزه تبدیل(DCT) [۱۳]، طبق جدول (۸)، به نسبت SNR مناسبی را دارد. هرچند روش [۱۳] کمی بهتر از روش پیشنهادی است.

جدول (۸): مقایسه روش پیشنهادی با روش‌های استاندارد معمولی

روش	[۱۲]	[۱۳]	پیشنهادی
SNR	۳۰/۲	۳۶/۸	۳۴/۹

روش پیشنهادی در مقایسه با روش‌های مرتبط پیشین که قبلاً مطرح شد، طبق جدول (۹)، دارای LossMsg مناسب و عالی دارد.

جدول (۹): مقایسه روش پیشنهادی با روش‌های مرتبط پیشین

روش	Epoch	LossD	LossG	Loss Msg	SNR
مرجع [۸]	۵۰	۰/۶۲	۰/۶۹	۰/۰۸	۴۲/۳
مرجع [۹]	۵۰	۰/۷۱	۰/۷۸	۰/۱۵	۳۹/۷
مرجع [۱۰]	۵۰	۰/۵۹	۰/۶۵	۰/۰۷	۴۳/۸
مرجع [۱۱]	۵۰	۰/۶۸	۰/۷۵	۰/۱۲	۴۰/۱
پیشنهادی	۵۰	۰/۷	۱/۲	۰/۰۳	۳۴/۹

۴- نتیجه گیری

وجود هر جزء EGD نقش مشخصی در GAN دارد که برای کارکرد همگرا و تولید داده‌های واقع‌گرایانه ضروری است. وظیفه اصلی کدگذار، تبدیل ورودی داده‌ها یا نویز به نمایش‌های پیام‌های نهان‌نگاری‌شده، است. در واقع کاهش حساسیت اطلاعات مهم ورودی برای استفاده توسط کدگذار و همچنین استخراج ویژگی‌های اساسی از داده‌های واقعی یا از فضای نویز، از وظایف مهم کدگذار است تا بتواند با استفاده از این نمایش‌ها، خروجی‌های واقع‌گرایانه تولید کند. وظیفه مولد، تولید داده‌های نمونه‌سازی‌شده[1] که توزیع آن‌ها به توزیع داده‌های واقعی نزدیک باشد، است. به عبارتی، تولید تصاویر یا سیگنال‌هایی با ویژگی‌های مشابه داده‌های واقعی از وظایف بخش مولد محسوب می‌شود. متمایزکننده، وظیفه تمایز بین نمونه‌های واقعی از نمونه‌های تولیدشده توسط مولد را به عهده دارد. یعنی نقش ارائه سیگنال گرادیانی به مولد برای بهبود کیفیت تولید را به عهده دارد و به عنوان ابزاری برای اندازه‌گیری شباهت بین دو توزیع و هدایت همگرا بودن فرایند آموزش کاربرد دارد. برای همگرا شدن یادگیری، کدگذار و مولد با متمایزکننده در رقابتی مثبت شرکت می‌کنند؛ مولد سعی می‌کند مشابه واقعیت را در خروجی‌ها تقلید کند و متمابزکننده سعی می‌کند این تقلید را تشخیص دهد. این فشار دوطرفه منجر به همگرا شدن توزیع خروجی مولد با توزیع داده‌های واقعی می‌شود. وجود کدگذار به مولد امکان می‌دهد تا به جای کار با نویز خالی، از نمایش‌های فشرده و معنایی استفاده کند، که معمولاً منجر به کیفیت و ثبات بالاتری در تصاویر یا سیگنال‌های تولیدی می‌شود. تنظیم مناسب وزن‌ها و معماری کدگذار در کنار مولد و متمایزکننده، مانند استفاده از تکنیک‌هایی مانند عادی‌سازی[2] و آموزش هم‌زمان با نرخ‌های یادگیری مناسب، می‌تواند پایداری آموزش را بهبود دهد. تعداد کم اِپُک‌ها در این مقاله، نشان می‌دهد مدل با داده‌های محدود آموزش می‌بیند. این مسئله به‌خاطر محدودیت دسترسی نویسنده مقاله در زیرساخت (رایانه پردازشی) بوده است. با توجه به این‌که مقادیر اتلاف (هزینه) روند کلی یادگیری و تغییرات کیفی را نشان می‌دهند، مقادیر اتلاف LossD و LossG که نشانگر هم‌گرایی یا پایداری آموزش بوده و LossMsg که متریک مربوط به خطای محاسبه شده بین پیام اصلی و پیام استخراج‌شده توسط کدگذار است، برای ارزیابی، پارامترهای مهمی در نهان‌نگاری به شیوه مطرح‌شده در این مقاله به حساب می‌آید. همچنین SNR که نسبت سیگنال به نویز را نشان می‌دهد، پارامتر ساده و مناسبی جهت بهبود یا کیفیت نهان‌نگاری به شمار می‌آید. البته پارامترهای مرسوم ارزیابی در نهان‌نگاری وجود دارد که در این مقاله مطرح نشده است؛ زیرا با پارامترهای LossD، LossG، LossMsg و SNR موضوعات خاص اتلاف نهان‌نگاری مبتنی بر GAN با استفاده از EGD پوشش داده شده است.

روش نهان‌نگاری مبتنی بر GAN در تصاویر با استفاده از یک معماری EGD هرچند با تعداد کم اِپُک، دارای مقدار SNR LossD و LossG مناسب است. همان‌طور که انتظار می‌رفت، مقدار LossMsg در سطح عالی حاصل شد.

۵- مراجع

[1] 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, Oct. 2020, doi: https://doi.org/10.1186/s13640-020-00534-2.

[2] R. Al Maawali and A. AL-Shidi, "Optimization Algorithms in Generative AI for Enhanced GAN Stability and Performance," Applied Computing Journal, pp. 359–371, Jan. 2025, doi: https://doi.org/10.52098/acj.20244225.

[3] A. Kumar, Prasanna Sattigeri, and P. T. Fletcher, "Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference," arXiv (Cornell University), Jan. 2017, doi: https://doi.org/10.48550/arxiv.1705.08850.

[4] K. R. Malik et al., "A hybrid steganography framework using DCT and GAN for secure data communication in the big data era," Scientific Reports, vol. 15, no. 1, Jun. 2025, doi: https://doi.org/10.1038/s41598-025-01054-7.

[5] A. Kuyoro, U. J. Nzenwata, O. Awodele, and S. Idowu, "GAN-Based Encoding Model for Reversible

[1] synthetic

[2] normalization

Image Steganography," Revue d'Intelligence Artificielle, vol. 36, no. 4, pp. 561–567, Aug. 2022, doi: https://doi.org/10.18280/ria.360407.
[6] G. Gao, T. Xu, and F. Hua, "Robust Image Watermarking Based on Generative Adversarial Networks for Copyright Protection," Mar. 2024, doi: https://doi.org/10.21203/rs.3.rs-4039149/v1.
[7] S. Talati , R. Esfahani, "Presenting a New Method of Image Steganalysis Based on MLP Neural Network", Scientific Journal of Passive Defence, Vol. 14, No. 4, Winter 2023, Serial No. 56, DOR: 20.1001.1.20086849.1403.15.1.3.0 (In Persian).
[8] S. Baluja, "Hiding images in plain sight: deep steganography," Neural Information Processing Systems, vol. 30, pp. 2066–2076, Dec. 2017.
[9] and S.-P. Lu, "A Compact Neural Network-based Algorithm for Robust Image Watermarking," arXiv (Cornell University), Jan. 2021, doi: https://doi.org/10.48550/arxiv.2112.13491.
[10] J. Huang, T. Luo, L. Li, G. Yang, H. Xu, and C.-C. Chang, "ARWGAN: Attention-Guided Robust Image Watermarking Model Based on GAN," IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1–17, Jan. 2023, doi: https://doi.org/10.1109/tim.2023.3285981.
[11] C. A. Candra Ahmadi, J.-L. C. Candra Ahmadi, and Y.-T. L. Jiann-Liang Chen, "Securing AI Models Against Backdoor Attacks: A Novel Approach Using Image Steganography," Journal of Internet Technology 25.3, vol. 25, no. 3, pp. 465–475, May 2024, doi: https://doi.org/10.53106/160792642024052503012.
[12] M. Kaur and V. K. Sharma, "Encryption based LSB Steganography Technique for Digital Images and Text Data," International Journal of Advanced engineering, Management and Science, vol. 2, no. 9, Sep. 2016.
[13] Singh, Ninni, and Gunjan Chhabra. "Cryptography and Steganography Techniques." Information Security and Optimization. Chapman and Hall/CRC, 2020. 79-91, (Book).