

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



**ĐỀ CƯƠNG ĐỒ ÁN MÔN HỌC ÂN THÔNG TIN TRÊN
DỮ LIỆU SỐ VÀ ỨNG DỤNG**
Ngành An Toàn Thông Tin
ĐỀ TÀI:

Tên tiếng Anh: **A PRACTICAL APPROACH TO DEEP IMAGE
STEGANOGRAPHY FOR SECURE CYBER
COMMUNICATION**

Tên tiếng Việt: **PHƯƠNG PHÁP TIẾP CẬN THỰC TẾ ĐỂ GIẤU
ẢNH SÂU CHO TRUYỀN THÔNG MẠNG AN
TOÀN**

Giảng viên: TS. Nguyễn Ngọc Tự

Học viên thực hiện:

- | | |
|--------------------|---------------------------|
| 1- Nguyễn Đức Thái | Mã số học viên: 240202027 |
| 2- Ngô Hoàng Anh | Mã số học viên: 240202018 |
| 3- Tạ Duy Huy | Mã số học viên: 240202023 |

Mục lục

Chương I: GIỚI THIỆU.....	4
CHƯƠNG II: LÝ THUYẾT	5
2.1 Auto – Encoder.....	5
2.1.1 Khái niệm.....	5
2.1.2 Kiến trúc của Auto – Encoder	5
2.1.3 Hiệu quả của Auto – Encoder trong Steganography	5
2.2 U – Net.....	6
2.3 Phương hướng tiếp cận và tổng quát về kiến trúc Deep Learning	8
2.5 Dilated Convolution	9
2.6 Hiding Network Architecture	11
2.7 Reveal Network Architecture	14
2.8 Loss function.....	16
CHƯƠNG III: KẾT QUẢ VÀ ĐÁNH GIÁ.....	18
3.1 Kết quả (hyper-parameter).....	18
3.1.1 Model v2.....	18
3.1.2 Model v4.....	19
3.1.3 Model v5.....	21
3.1.4 Kết quả chung – Nhận xét	23
3.2 Đánh giá	25
3.2.1 Performance	25
3.2.2 Capacity	27
3.2.3 PNRs, SSIM	28
Chương IV: TRIỂN KHAI VÀ ĐÓNG GÓI.....	33
4.1 Chuẩn bị.....	33
4.2 Đóng gói với PyInstaller	33
Chương V: KIỂM THỦ	35
5.1 Mục tiêu kiểm thử	35

5.2 Môi trường và công cụ kiểm thử	35
5.3 Tiến hành kiểm thử.....	35
Test case 1: Giảu text ngắn	36
Test case 2: Giảu text dài.....	38
Test case 3: Giảu file ảnh nhỏ	40
Test case 4: Giảu file ảnh lớn	42
Test case 5: Upload nhiều lần	45
Chương VI: KẾT LUẬN	46
References	47

Chương I: GIỚI THIỆU

Steganography là nghệ thuật ẩn thông tin bí mật – có thể là văn bản, hình ảnh, hay bất kỳ dữ liệu nào – vào một "vật chứa" như ảnh số, mà không để ai nhận ra sự tồn tại của nó. Khác với mã hóa (cryptography) chỉ làm cho dữ liệu trở nên khó hiểu, steganography che giấu chính sự hiện diện của thông tin. Trong thế giới số, ảnh số là lựa chọn lý tưởng để giấu tin nhờ sự phổ biến và cấu trúc phức tạp, giúp dữ liệu ẩn trở nên vô hình với mắt thường.

Trước đây, các phương pháp truyền thống như LSB (thay đổi bit ít quan trọng), DCT, hay wavelet đã được sử dụng rộng rãi. Tuy nhiên, chúng thường bị hạn chế bởi khả năng chứa dữ liệu thấp và dễ bị phát hiện bởi các công cụ phân tích hiện đại (steganalysis). Chẳng hạn, kỹ thuật LSB tuy đơn giản nhưng lại để lại những dấu vết rõ ràng trên ảnh, khiến thông tin ẩn dễ bị lộ.

Sự phát triển của học sâu (deep learning) đã thay đổi cuộc chơi. Các mạng neuron sâu có thể tự động học cách nhúng và trích xuất dữ liệu một cách thông minh, tạo ra những bức ảnh chứa tin (ảnh stego) gần như không thể phân biệt với ảnh gốc. Quan trọng hơn, chúng còn cho phép khôi phục dữ liệu chính xác ngay cả khi ảnh bị chỉnh sửa hay nhiễu.

Báo cáo này giới thiệu một mô hình steganography dựa trên học sâu, tập trung vào việc giấu ảnh màu bí mật vào ảnh cover cùng kích thước. Mô hình sử dụng kiến trúc encoder-decoder, kết hợp với các mô-đun convolution giãn nở, nhằm tạo ra ảnh stego có chất lượng cao và đảm bảo khả năng tái tạo ảnh bí mật chính xác. Các phần tiếp theo sẽ trình bày chi tiết cách thiết kế mô hình, quá trình huấn luyện, kiểm thử, và kết quả đánh giá thực nghiệm.

CHƯƠNG II: LÝ THUYẾT

2.1 Auto – Encoder

2.1.1 Khái niệm

Auto – Encoder là một loại mạng nơ – ron nhân tạo (neural network) được sử dụng trong học máy, đặc biệt trong học không giám sát (unsupervised learning). Nó được thiết kế để học cách biểu diễn dữ liệu (data representation) một cách hiệu quả, thường là ở dạng nén (compressed form), bằng cách tái tạo lại dữ liệu đầu vào từ một không gian ẩn (latent space).

2.1.2 Kiến trúc của Auto – Encoder

Auto – encoder gồm 3 phần chính:

- Encoder: gồm các lớp mã hoá có nhiệm vụ nén dữ liệu đầu vào thành biểu diễn được mã hoá. Trong Auto – encoder thông thường, các lớp ẩn của neural network thường chứa số lượng nút nhỏ dần so với lớp đầu vào, khi dữ liệu đi qua lớp mã hoá thường được nén thành ít chiều hơn.
- Bottleneck: chứa các biểu diễn dữ liệu được nén từ Encoder (vừa là đầu ra của Encoder và vừa là đầu vào của Decoder. Mục tiêu của Auto – encoder là phát hiện tối thiểu các đặc trưng quan trọng (hoặc chiều) cần thiết để tái tạo dữ liệu đầu ra
- Decoder: gồm số lượng chiều hay nút ngày càng lớn để giải mã các biểu diễn dữ liệu đã được mã hoá, cuối cùng tái tạo dữ liệu trở lại dạng ban đầu trước khi được mã hoá. Đầu ra được tái tạo này sẽ đem so sánh với “ground truth” – trong hầu hết trường hợp là đầu vào của Auto – Encoder, việc này giúp đánh giá hiệu quả của Auto – Encoder. Sự khác nhau giữa đầu ra và giá trị thực tế là lỗi tái tạo.

2.1.3 Hiệu quả của Auto – Encoder trong Steganography

a. Khả năng ẩn thông tin hiệu quả

Encoder có thể được sử dụng để mã hóa thông tin bí mật (secret message/image) vào một không gian ẩn, sau đó nhúng nó vào phương tiện (container, ví dụ: hình ảnh, video). Ví dụ, một hình ảnh gốc được mã hóa cùng với ảnh bí mật để tạo ra một hình ảnh chứa thông tin ẩn (stego image).

Decoder được sử dụng ở phía người nhận để trích xuất thông tin bí mật từ stego image, tái tạo lại thông điệp gốc.

Auto – Encoder đảm bảo rằng thông tin ẩn được nhúng một cách tinh vi, khó bị phát hiện bằng mắt thường hoặc các công cụ phân tích thông thường.

b. Tăng cường tính vô hình

Auto – Encoder học cách bảo toàn các đặc trưng của “ground truth” (ví dụ: kết cấu, màu sắc của hình ảnh), giúp stego image trông giống hệt hình ảnh gốc. Điều này làm giảm khả năng bị phát hiện bởi các kỹ thuật phân tích thống kê hoặc học máy.

Các biến thể như Denoising Auto – Encoder có thể giúp xử lý nhiễu hoặc các biến đổi (như nén, cắt xén) trên stego image, đảm bảo thông tin bí mật vẫn được trích xuất chính xác.

c. Tăng dung lượng ẩn thông tin

Auto – Encoder có thể nén thông tin bí mật vào không gian ẩn với kích thước nhỏ, cho phép nhúng lượng dữ liệu lớn hơn vào phương tiện mà không làm thay đổi đáng kể chất lượng của phương tiện. Ví dụ, một Auto – Encoder được huấn luyện tốt có thể ẩn toàn bộ văn bản hoặc hình ảnh trong một hình ảnh khác mà vẫn giữ được chất lượng hình ảnh cao.

2.2 U – Net

2.2.1 Khái niệm

U – Net là một kiến trúc neural network hoàn toàn được tạo ra ban đầu cho nhiệm vụ phân đoạn ảnh y tế (image segmentation), sau này do tính hiệu quả và linh hoạt của nó nên đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau

2.2.2 Kiến trúc

U – Net có kiến trúc đối xứng, giống hình chữ “U” như tên gọi của nó, bao gồm 2 nhánh chính:

Encoder (Contracting Path):

- Bao gồm các tầng tích chập (convolutional layers) và pooling (max – pooling) để giảm kích thước không gian của hình ảnh đầu vào, đồng thời tăng số lượng kênh (feature maps).
- Mục tiêu: Trích xuất các đặc trưng cấp cao (high-level features) từ hình ảnh, tạo ra một biểu diễn nén trong không gian ẩn (latent space).

- Quá trình này tương tự như phần mã hóa trong Auto – Encoder, nhưng tập trung vào việc giữ lại thông tin không gian chi tiết.

Decoder (Expanding Path):

- Bao gồm các tầng tích chập ngược (transposed convolutions hoặc upsampling) để khôi phục kích thước không gian của hình ảnh, kết hợp với các tầng tích chập để tinh chỉnh đặc trưng.
- Mục tiêu: Tái tạo lại hình ảnh hoặc đầu ra (ví dụ: mặt nạ phân đoạn) từ các đặc trưng đã được trích xuất.

Skip Connections:

- Đặc điểm nổi bật của U – Net là các kết nối ngắn (skip connections) giữa các tầng đối xứng trong encoder và decoder. Các đặc trưng từ encoder được sao chép và ghép (concatenate) với các đặc trưng trong decoder.
- Lợi ích: Giữ lại thông tin chi tiết cấp thấp (low-level features) như kết cấu, cạnh, giúp cải thiện độ chính xác trong việc tái tạo hoặc phân đoạn.

U – Net thường được huấn luyện với hàm mất mát như cross – entropy (image segmentation) hoặc mean squared error (image reconstruction), tùy thuộc vào bài toán.

2.2.3 Hiệu quả trong Steganography

a. Ẩn thông tin với độ chính xác cao

Với kiến trúc của U – Net có thể phân tích tốt các đặc trưng của container (ảnh cover) ở nhánh chữ “U” bên trái và sau đó nhúng thông tin ẩn vào đáy chữ “U” và tái tạo lại ảnh chứa thông tin ẩn (ảnh stego) ở nhánh chữ “U” bên phải.

Skip connections giúp bảo toàn các chi tiết không gian của ảnh cover, làm cho ảnh stego trông gần giống hoàn toàn với ảnh cover nhất có thể, giảm nguy cơ bị phát hiện.

b. Dung lượng ẩn thông tin lớn

U – Net có thể xử lý các đặc trưng phức tạp trong hình ảnh, từ đó cho phép nhúng một lượng thông tin lớn hơn vào cover image mà không làm thay đổi đáng kể chất lượng hình ảnh.

Các tầng tích chập của U – Net học cách phân bổ thông tin bí mật một cách tối ưu trên toàn bộ hình ảnh, tận dụng cả những vùng có kết cấu phức tạp để ẩn dữ liệu.

c. **Khả năng chống nhiễu và biến đổi**

Stego image thường phải chịu các loại biến đổi như nén JPEG, cắt xén, hoặc thêm nhiễu trong quá trình truyền tải đến người nhận. U – Net với kiến trúc mạnh mẽ và khả năng tái tạo chi tiết, có thể được huấn luyện để trích xuất thông tin bí mật một cách chính xác ngay cả khi stego image bị biến đổi. Điều này tương tự như khả năng của Denoising Auto – Encoder, nhưng U – Net vượt trội hơn trong việc xử lý các chi tiết không gian, đặc biệt với hình ảnh.

2.3 Phương hướng tiếp cận và tổng quát về kiến trúc Deep Learning

Kiến trúc deep learning tổng quan được đề xuất là Auto-Encoder và U-Net cho Encoder vì những hiệu suất cao mà hai kiến trúc này đem lại trong lĩnh vực Steganography. Kiến trúc được đề xuất bao gồm hai mạng chính:

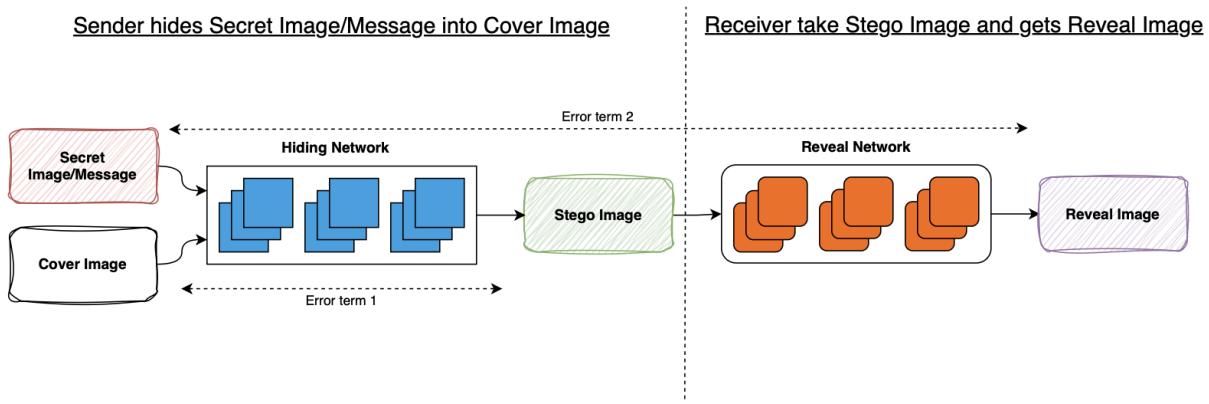
Hiding Network: được thiết kế dựa trên kiến trúc U – Net Auto – Encoder, chịu trách nhiệm mã hoá thông tin bí mật (secret image/message) và nhúng vào phương tiện gốc (container) để tạo ra stego image.

Reveal Network: sử dụng neural network tích chập (CNN) được thiết kế để trích xuất thông tin bí mật từ stego image và tái tạo lại thông điệp gốc (ground truth).

Lý do Hiding Network được xây dựng trên kiến trúc U – Net:

- Bảo toàn chi tiết không gian: Skip connections trong U – Net giúp giữ lại các đặc trưng cấp thấp (như cạnh, kết cấu, màu sắc) của cover image, làm cho stego image trông tự nhiên và khó bị phát hiện bởi các công cụ steganalysis.
- Khả năng nhúng thông tin phức tạp: U-Net có thể xử lý các đặc trưng cấp cao và cấp thấp đồng thời, cho phép nhúng lượng thông tin bí mật lớn hơn mà không làm giảm chất lượng hình ảnh.
- Tính linh hoạt: U-Net đã được chứng minh hiệu quả trong nhiều bài toán xử lý hình ảnh, từ phân đoạn đến tái tạo, nên phù hợp để tích hợp vào hệ thống steganography.

Trong khi đó, Reveal Network được thiết kế đơn giản hơn, nhằm tập trung vào nhiệm vụ trích xuất và tái tạo thông điệp bí mật từ stego image. Sự kết hợp này nhằm đảm bảo tính cân bằng giữa hiệu quả mã hoá và khả năng trích xuất thông tin.



Hình 1: Tổng quát kiến trúc và quá trình hoạt động đề xuất

Trong kiến trúc này, các đặc trưng của secret image được trích xuất đồng thời hoạt động “che giấu” trong hiding network, mô hình sẽ học cách trích xuất các đặc trưng và giấu chúng đồng thời với nhau. Mô hình này được huấn luyện end-to-end: không phải huấn luyện trong việc giấu dữ liệu trước và sau đó mới tới việc trích xuất dữ liệu bí mật một cách tách rời nhau, thay vào đó mô hình này được huấn luyện cho việc giấu và trích xuất dữ liệu đồng thời với nhau.

Hiding network được xây dựng dựa trên kiến trúc U – Net và sử dụng tích chập dilated, mạng này tạo ra stego image từ cover image và secret image/message. Các lớp CNN học các trích xuất đặc trưng thứ cấp từ hình ảnh và thứ cấp từ cấp thấp đến cấp cao của các đặc trưng cụ thể. Vì vậy, hiding network có thể học được các đặc trưng của cả cover image và secret image, điều này cho phép mạng này giấu các đặc trưng của secret image vào những đặc trưng của cover image.

Reveal network sẽ trích xuất secret image từ stego image bằng cách sử dụng hoàn toàn mạng tích chập cổ điển.

2.5 Dilated Convolution

Trong deep learning, mạng nơ-ron tích chập (Convolution Neural Network – CNN) được thiết kế để trích xuất đặc trưng của hình ảnh hoặc chuỗi thời gian. Thông thường, theo sau CNN là pooling operation, điều này có thể làm giảm độ phân giải của hình ảnh nhưng sẽ đem lại kết quả tốt trong các tác vụ high-level computer vision như phân loại hình ảnh.

Nhưng việc giảm độ phân giải không phải lúc nào phù hợp, đặc biệt trong lĩnh vực Steganography vì pooling operation không thể đảo ngược và làm mất mát thông tin không gian, điều này sẽ làm việc reconstruction thông tin của các vật thể nhỏ trong hình ảnh.

Dilated Convolution (hay gọi là mạng tích chập giãn nở) là một biến thể của mạng tích chập thông thường, được thiết kế để mở rộng trường tiếp nhận (receptive field) của bộ lọc mà không làm tăng tham số hoặc giảm độ phân giải của hình ảnh giúp bảo toàn đặc trưng. Không giống như tích chập thông thường, dilated convolution đưa vào một tham số gọi là tỷ lệ giãn nở (dilation rate), ký hiệu là r , cho phép bỏ qua các giá trị pixel trong quá trình tích chập, từ đó thu thập thông tin từ một vùng rộng hơn mà không cần tăng kích thước bộ lọc. Kỹ thuật này đặc biệt hữu ích trong các tác vụ yêu cầu thông tin ngữ cảnh rộng, chẳng hạn như phân đoạn ngữ nghĩa, nhận diện đối tượng và Steganography.

Cho bộ lọc f và tín hiệu g , tích chập thông thường được định nghĩa như sau:

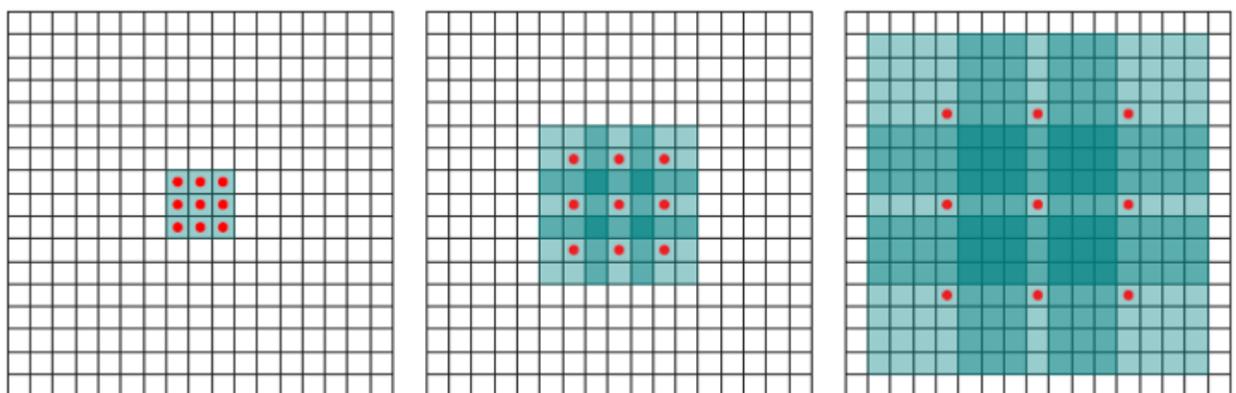
$$h(t) = \sum_k f(k) g(t - k).$$

Trong khi tích chập giãn nở với tỷ lệ giãn nở r được định nghĩa là:

$$h_{dil}(t) = \sum_k f(k) g(t - rk),$$

Trong đó:

- $f(k)$: Giá trị của bộ lọc tại vị trí k .
- $g(t - rk)$: Giá trị của tín hiệu đầu vào tại vị trí cách t một khoảng rk .
- r : Tỷ lệ giãn nở (dilation rate), quyết định khoảng cách giữa các điểm được lấy mẫu trong tích chập.



Hình 2: Các dilated convolution với rates là 1, 2 ,4

Ở bên trái với $r = 1$, đây chính xác là tích chập thông thường, lấy mẫu liên tiếp các pixel. Ở giữa với $r = 2$, bộ lọc lấy mẫu các pixel cách nhau 2 đơn vị, tạo ra một lưới thưa hơn và mở rộng trường tiếp nhận từ 3×3 lên 5×5 . Tương tự với $r = 4$, bộ lọc lấy mẫu các pixel cách nhau 4 đơn vị, tạo ra một lưới thưa hơn và mở rộng trường tiếp nhận từ 3×3 lên 15×15 .

Dilated Convolution operator có thể được áp dụng các bộ lọc giống nhau với các tần số khác nhau bằng cách sử dụng tỷ lệ giãn nở r (dilation rate). Tích chập giãn nở hỗ trợ mở rộng vùng tiếp nhận của bộ lọc mà không làm mất mật độ phân giải của hình ảnh, điều này rất quan trọng trong Steganography.

2.6 Hiding Network Architecture

Hiding Network được thiết kế để thực hiện quá trình mã hóa thông tin bí mật trong Steganography. Hiding network nhận hai hình ảnh đầu vào là hình ảnh bí mật và hình ảnh bao phủ, cả hai đều là hình ảnh màu (3 kênh RGB). Hai hình ảnh này được ghép nối thành một tensor 6 kênh ($N \times N \times 6$). Mạng học cách trích xuất các đặc trưng của cả hai hình ảnh, từ đặc trưng cấp thấp (như cạnh, kết cấu) đến đặc trưng cấp cao (như hình dạng, ngữ cảnh). Mạng nén và phân bố các bit của hình ảnh bí mật vào các bit của hình ảnh bao phủ, tạo ra một hình ảnh stego ($N \times N \times 3$) trông giống hệt hình ảnh bao phủ về mặt thị giác.

Cấu trúc của hiding network được lấy cảm hứng từ kiến trúc U-Net auto-encoder và tích hợp dilated inception module. Inception Module được giới thiệu bởi GoogLeNet teams trong cuộc thi ILSVR14. Mục tiêu là trích xuất multi-scale thông tin đa ngữ cảnh. Ý tưởng của module là sử dụng các tích chập có kernel có kích thước khác nhau nhằm trích xuất các đặc trưng multi-scale trong vùng tiếp nhận có kích thước khác nhau.

Inception Module bao gồm 4 nhánh:

- Nhánh 1: lớp tích chập thông thường có kernel size 1×1 , sau là hàm kích hoạt ReLU
- Nhánh 2: lớp tích chập thông thường với kernel size 1×1 , kết nối với hàm kích hoạt ReLU, sau cùng là dilated convolution với rate = 1
- Nhánh 3: lớp tích chập thông thường với kernel size 1×1 , hàm kích hoạt ReLU, dilated convolution có rate = 2
- Nhánh 4: tương tự nhánh 3, nhưng dilated convolution có rate = 3

Cả 4 nhánh có input channel bằng với input channel của inception module, nhưng output channel của mỗi nhánh chỉ bằng $1/4$ output channel của inception module. Số channel phải giảm vì nếu giữ nguyên output channel mỗi nhánh bằng output channel của inception module thì sẽ tăng số lượng tham số rất lớn gây ra lỗi OutOfMemory trong Kaggle.

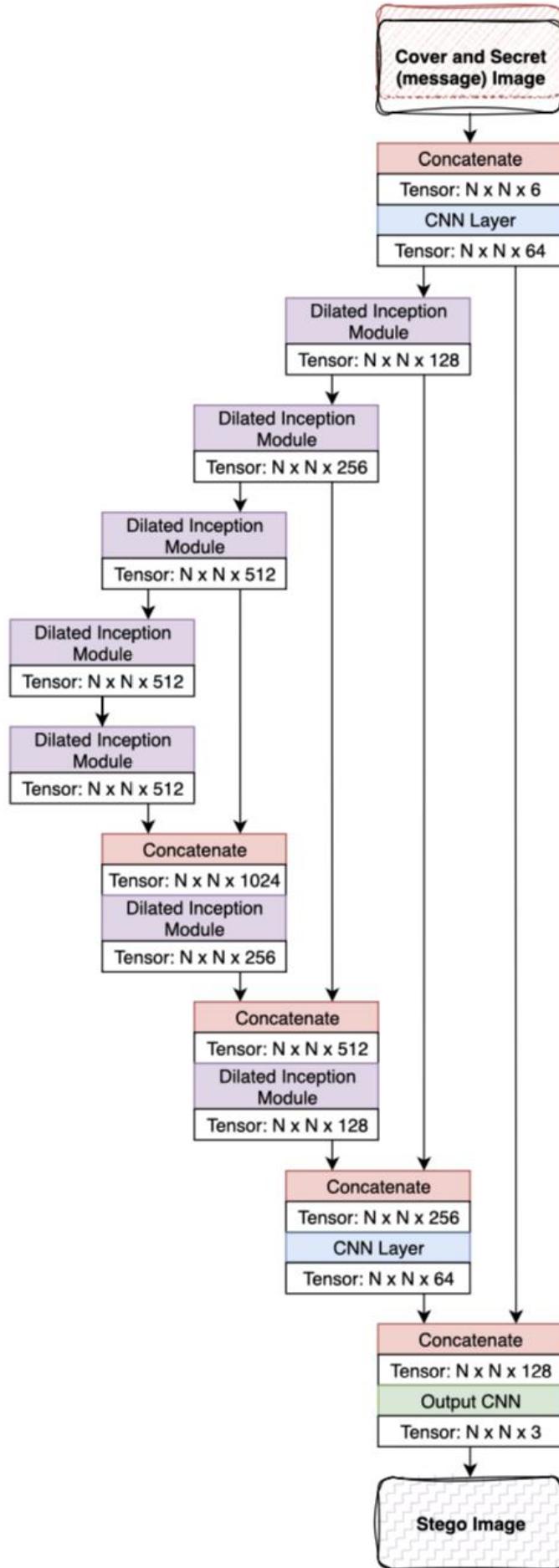
Ví dụ: Mỗi inception module là một layer trong hididing network. Khi cho input: $N \times N \times 64$ và output: $N \times N \times 128$ thì output channel mỗi nhánh chỉ có $128/4 = 32$ và sau đó được ghép nối lại với nhau.



Hình 3: Dilated Inception Module

GoogLeNet teams sử dụng Max Pooling cho Inception Module của họ nhưng sẽ làm số lượng tham số tăng lên rất nhiều, cho nên Inception Module của nhóm đã được giảm output channel của mỗi nhánh, ghép nối output channel của mỗi nhóm lại và đưa vào lớp tích chập 1×1 . Khi output channel của các nhánh được ghép nối, tensor kết quả chứa một tập hợp các đặc trưng đa dạng nhưng có thể dư thừa hoặc thiếu sự tương tác giữa các nhánh. Lớp tích chập 1×1 hoạt động như một phép chiếu tuyến tính, kết hợp và tinh chỉnh các đặc trưng từ các nhánh, tạo ra một biểu diễn đặc trưng thống nhất và có ý nghĩa hơn.

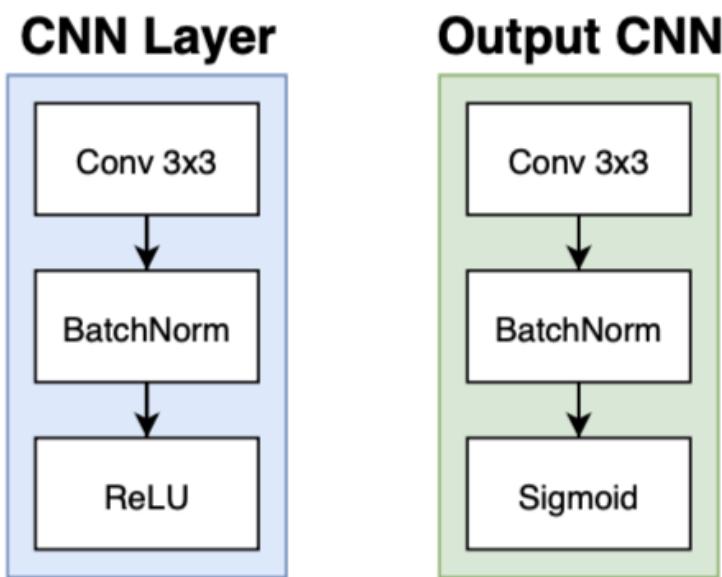
Kiến trúc của hididing network được lấy cảm hứng từ cấu trúc U – Net. Cả 2 nhánh chữ “U” đều sử dụng chuỗi lớp tích chập 3×3 , hàm kích hoạt ReLU và các dilated inception module.



Hình 4: Hiding Network Architecture

Nhánh chữ “U” xuống giúp trích xuất Trích xuất đặc trưng từ tensor đầu vào 6 kênh thông qua các tầng tích chập và dilated inception module. Các tầng này tăng số lượng kênh đặc trưng (từ 64 lên 512) để biểu diễn thông tin phức tạp.

Nhánh chữ “U” đi lên sẽ tái tạo hình ảnh stego từ các đặc trưng đã mã hóa, giảm số lượng kênh đặc trưng (từ 512 về 3) để tạo ra đầu ra là hình ảnh màu RGB. Và giống với cấu trúc của U – Net mỗi layer của nhánh “U” xuống sẽ kết nối với những đặc trưng của nhánh “U” lên, với mục đích đảm bảo rằng network sẽ học những đặc trưng của những layer trước.



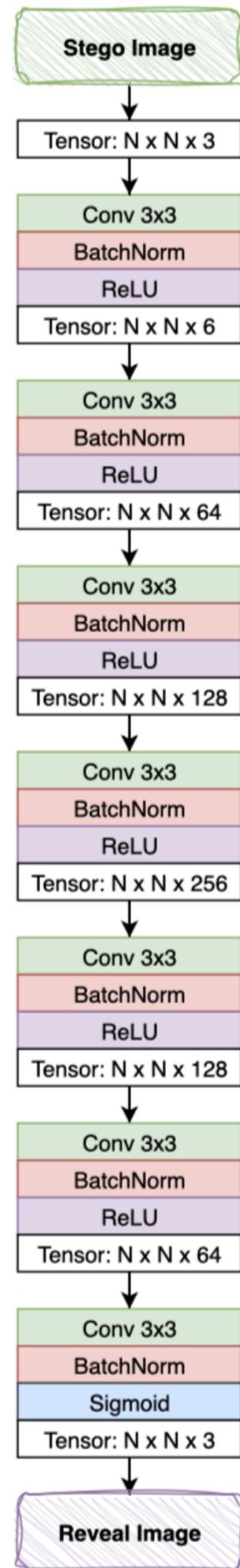
Hình 5: Architecture CNN Layer and Output CNN

CNN Layer và Output CNN đều sử dụng lớp tích chập kernel size 3×3 , lớp chuẩn hoá BatchNorm. Chỉ khác biệt, CNN Layer sử dụng hàm kích hoạt ReLU và Output CNN sử dụng Sigmoid để đưa giá trị về $[0, 1]$ giúp xây dựng thành stego image.

2.7 Reveal Network Architecture

Reveal network nhận stego image ($N \times N \times 3$) từ hiding network, trong đó secret image đã được mã hóa. Reveal network chỉ sử dụng các lớp tích chập thông thường để tập trung trích xuất đặc trưng của secret image trong stego image, tách biệt dữ liệu của secret image ra khỏi cover image.

Sau đó cỗ gǎng tái tạo lại secret image ($N \times N \times 3$) từ các đặc trưng đã trích xuất, đảm bảo rằng reveal image giữ được tính toàn vẹn thông tin với sai số chấp nhận được.



Hình 6: Architecture Reveal Network

Không giống hiding network, reveal network chỉ sử dụng tích chập thông thường (không có tích chập giãn nở) với các lý do sau:

- Đơn giản hóa quá trình khôi phục: Hình ảnh stego đã chứa thông tin bí mật được mã hóa bởi Hiding Network, và Reveal Network chỉ cần trích xuất các đặc trưng liên quan mà không cần mở rộng trường tiếp nhận bằng tích chập giãn nở.
- Bảo toàn độ phân giải: Các tầng tích chập ($N \times N$) được giữ nguyên qua các tầng, phù hợp với yêu cầu tái tạo hình ảnh bí mật ở độ phân giải gốc.

2.8 Loss function

Trong Steganography cho hình ảnh, chúng ta ẩn secret image vào trong cover image, từ đó loss function có 2 sự mất mát:

- Sai số cover – stego: Sai số giữa hình ảnh bao phủ (cover image, C) và hình ảnh stego (stego image, C'), nhằm đảm bảo tính vô hình của hình ảnh stego trước steganalysis.
- Sai số secret-revealed: Sai số giữa hình ảnh bí mật gốc (secret image, S) và hình ảnh bí mật khôi phục (revealed image, S'), nhằm đảm bảo chất lượng khôi phục của thông tin bí mật.

Đặc biệt trong covert channel, stego image là hình ảnh được truyền qua kênh liên lạc và đối mặt với các cuộc tấn công steganalysis, tính vô hình của model cũng phụ thuộc chất lượng của stego image. Còn secret image vẫn có thể chấp nhận được đối với người nhận nếu không giữ không được chất lượng lúc ban đầu. Cho nên, hàm mất mát được thiết kế ưu tiên giảm sai số cover – stego hơn sai số secret – reveal.

Lỗi giữa C và C' được tính bởi mean absolute error L_1 , cộng với variance loss. Để ưu tiên sai số giữa cover – stego hơn sai số secret – reveal, variance loss được thêm vào vì hiding và reveal network đều tạo ra hình ảnh từ những được hình ảnh đã được tái cấu trúc, cho nên sử dụng variance loss cho hiding network để đảm bảo rằng các thay đổi trên stego image được phân bố đồng đều trên toàn bộ hình ảnh, tránh tập trung sai số ở một số vùng cụ thể, từ đó tăng cường tính vô hình.

Sai số giữa S và S' được tính bằng mean squared error L_2 . Model được huấn luyện end – to – end giúp tối ưu hàng triệu tham số của model để đạt được cân bằng giữa các tiêu chí hiệu suất: tính vô hình, dung lượng cao và chất lượng khôi phục.

Loss:

$$Loss = \frac{1}{2} (L_1(C, C') + Var(L_1(C, C'))) + L_2(S, S')$$

Trong đó:

$$L_1(C, C') = \frac{1}{n} \sum_{i=1}^n |C_i - C'_i|,$$

$$L_2(S, S') = \frac{1}{n} \sum_{i=1}^n (S_i - S'_i)^2,$$

Với n là số lượng pixel trong mỗi hình ảnh.

CHƯƠNG III: KẾT QUẢ VÀ ĐÁNH GIÁ

3.1 Kết quả (hyper-parameter)

3.1.1 Model v2

Model v2 là một trong những model được training đầu tiên mà không áp dụng mô phỏng tấn công. Model v2 được training bằng tập dataset COCO 2017 bao gồm 118 nghìn hình ảnh sử dụng cho huấn luyện, 40.7 nghìn hình ảnh kiểm thử và 5000 hình ảnh để xác minh.

Sau đây là các HYPERPARAMETER cuối cùng sau rất nhiều lần thử:

```
BATCH_SIZE = 8
IMAGE_SIZE = 256
EPOCHS = 20      You, 5 minutes ago • Uncommitted changes
LR = 0.0001
WORKER = 4
LAMBDA = 1
LOG_FILE = 'log.txt'
CONTINUOUS = True
CHECKPOINT = '/kaggle/input/covertchannel-dp-v2/checkpoint_epoch_5.pth'
```

Hình 7: Hyperparameter model v2

Trong đó:

- BATCH_SIZE: Kích thước batch đưa vào huấn luyện
- IMAGE_SIZE: Kích thước resize hình ảnh trước khi đưa vào huấn luyện
- EPOCHS: Tổng số epoch huấn luyện
- LR: Learning rate
- WORKER: Số lượng worker trong dataloader
- LAMBDA: Trọng số cho loss L_2 giữa secret – reveal
- LOG_FILE: Đường dẫn tới log file
- CONTINUOUS: Boolean để xác định huấn luyện từ đầu hay từ checkpoint
- CHECKPOINT: Đường dẫn để load checkpoint tiếp tục huấn luyện

Model v2 được huấn luyện qua 20 epochs và chưa được áp dụng validation loss. Mỗi epoch chạy khoảng 140 phút. Mà mỗi notebook của Kaggle chỉ có giới hạn là 12 tiếng cho nên phải lưu checkpoint mà tiếp tục huấn luyện 5 lần để hoàn thành 20 epoch.

Sau đây là kết quả huấn luyện được ghi vào log file:

```
Training started...
Epoch [1/15], Total_Loss_Epoch: 0.1274
Epoch [2/15], Total_Loss_Epoch: 0.1129
Epoch [3/15], Total_Loss_Epoch: 0.1091
Epoch [4/15], Total_Loss_Epoch: 0.1068
Epoch [5/15], Total_Loss_Epoch: 0.1049

Continuous training started...
Epoch [6/15], Total_Loss_Epoch: 0.0289
Epoch [7/15], Total_Loss_Epoch: 0.0228
Epoch [8/15], Total_Loss_Epoch: 0.0199

Continuous training started...
Epoch [9/15], Total_Loss_Epoch: 0.0182
Epoch [10/15], Total_Loss_Epoch: 0.0171
Epoch [11/15], Total_Loss_Epoch: 0.0163
Epoch [12/15], Total_Loss_Epoch: 0.0153
Epoch [13/15], Total_Loss_Epoch: 0.0145
Epoch [14/15], Total_Loss_Epoch: 0.0141

Continuous training started...
Epoch [15/30], Total_Loss_Epoch: 0.0136
Epoch [16/30], Total_Loss_Epoch: 0.0132
Epoch [17/30], Total_Loss_Epoch: 0.0128
Epoch [18/30], Total_Loss_Epoch: 0.0124
Epoch [19/30], Total_Loss_Epoch: 0.0122

Continuous training started...
Epoch [20/20], Total_Loss_Epoch: 0.0118
```

Hình 8: Log file model v2

3.1.2 Model v4

Model v4 là version đầu tiên được áp dụng mô phỏng tấn công nhẹ với tỷ lệ 40% tấn công trên toàn bộ ảnh trong tập huấn luyện. Model v4 cũng được huấn luyện trên tập dataset COCO 2017.

Sau đây là các HYPERPARAMETER sau nhiều lần thử:

```

# Hyperparameters
BATCH_SIZE = 8
IMAGE_SIZE = 256
EPOCHS = 20
LR = 0.0002
AR = 0.4
SIGMA = 0.02
WORKER = 4
LAMBDA = 1
OLD_LOG_FILE = 'old_log.txt'
LOG_FILE = 'log.txt'
ANGLE = [0, 90, 180, 270]
CONTINUOUS = False
CHECKPOINT =

```

Hình 9: Hyperparameter model v4

Trong đó:

- BATCH_SIZE: Kích thước batch đưa vào huấn luyện
- IMAGE_SIZE: Kích thước resize hình ảnh trước khi đưa vào huấn luyện
- EPOCHS: Tổng số epoch huấn luyện
- LR: Learning rate
- AR: Attack rate
- SIGMA: Thông số dung cho tấn công noise
- WORKER: Số lượng worker trong dataloader
- LAMBDA: Trọng số cho loss L_2 giữa secret – reveal
- OLD_LOG_FILE: Đường dẫn tới log file của checkpoint trước đó
- LOG_FILE: Đường dẫn tới log file
- ANGLE: Chuỗi các góc dung trong tấn công xoay ảnh
- CONTINUOUS: Boolean để xác định huấn luyện từ đầu hay từ checkpoint
- CHECKPOINT: Đường dẫn để load checkpoint tiếp tục huấn luyện

Model v4 do có áp dụng mô phỏng tấn công nên loss ban đầu rất lớn và giảm rất chậm cho nên learning rate ban đầu được tăng lên 0.0002 và sẽ được giảm dần qua các epoch. Mô phỏng tấn công của model v4 chỉ áp dụng 40% tổng số ảnh trong huấn luyện bao gồm các loại tấn công (được áp dụng ngẫu nhiên – có khi chỉ một loại tấn công, có khi áp dụng toàn bộ các loại tấn công):

- Gaussian Noise: Mô phỏng nhiễu trong quá trình truyền qua Internet.

- Horizon, Vertical Flip: Mô phỏng việc lật ảnh ngang, dọc.
- Xoay ảnh: Xoay ảnh với nhiều góc độ như 0, 90, 180, 270 độ.

Model v4 được huấn luyện qua 20 epoch, được áp dụng validation loss để xác định model ở epoch nào là tốt nhất, tránh tình trạng overfit, dừng quá trình huấn luyện sau 5 epoch nếu validation loss không giảm. Do được áp dụng tấn công nên thời gian huấn luyện mỗi epoch tăng lên khoảng 180 phút và 7 phút validation.

Sau đây là kết quả huấn luyện và đánh giá được ghi vào log file sau khi hoàn tất:

```

Training started...      You, last week * add log of two verison
Epoch [1/20], Total_Loss_Epoch: 0.0402, Validation_Loss: 0.0223
Validation loss improved to 0.0223 at epoch 1

Epoch [2/20], Total_Loss_Epoch: 0.0295, Validation_Loss: 0.0213
Validation loss improved to 0.0213 at epoch 2

Epoch [3/20], Total_Loss_Epoch: 0.0257, Validation_Loss: 0.0188
Validation loss improved to 0.0188 at epoch 3

Epoch [4/20], Total_Loss_Epoch: 0.0219, Validation_Loss: 0.0143
Validation loss improved to 0.0143 at epoch 4

Continuous training started...
Epoch [5/20], Total_Loss_Epoch: 0.0199, Validation_Loss: 0.0163
Validation loss improved to 0.0163 at epoch 5

Epoch [6/20], Total_Loss_Epoch: 0.0185, Validation_Loss: 0.0146
Validation loss improved to 0.0146 at epoch 6

Epoch [7/20], Total_Loss_Epoch: 0.0175, Validation_Loss: 0.0106
Validation loss improved to 0.0106 at epoch 7

Epoch [8/20], Total_Loss_Epoch: 0.0167, Validation_Loss: 0.0128
Validation loss did not improve. Patience counter: 1/5

Epoch [9/20], Total_Loss_Epoch: 0.0161, Validation_Loss: 0.0162
Validation loss did not improve. Patience counter: 2/5

Continuous training started...
Epoch [10/20], Total_Loss_Epoch: 0.0155, Validation_Loss: 0.0123
Validation loss improved to 0.0123 at epoch 10

Epoch [11/20], Total_Loss_Epoch: 0.0151, Validation_Loss: 0.0109
Validation loss improved to 0.0109 at epoch 11

Epoch [12/20], Total_Loss_Epoch: 0.0148, Validation_Loss: 0.0120
Validation loss did not improve. Patience counter: 1/5

Epoch [13/20], Total_Loss_Epoch: 0.0144, Validation_Loss: 0.0105
Validation loss improved to 0.0105 at epoch 13

Epoch [14/20], Total_Loss_Epoch: 0.0141, Validation_Loss: 0.0112
Validation loss did not improve. Patience counter: 1/5

Continuous training started...
Epoch [15/20], Total_Loss_Epoch: 0.0139, Validation_Loss: 0.0109
Validation loss improved to 0.0109 at epoch 15

Epoch [16/20], Total_Loss_Epoch: 0.0136, Validation_Loss: 0.0099
Validation loss improved to 0.0099 at epoch 16

Epoch [17/20], Total_Loss_Epoch: 0.0133, Validation_Loss: 0.0121
Validation loss did not improve. Patience counter: 1/5

Epoch [18/20], Total_Loss_Epoch: 0.0131, Validation_Loss: 0.0109
Validation loss did not improve. Patience counter: 2/5

Epoch [19/20], Total_Loss_Epoch: 0.0129, Validation_Loss: 0.0114
Validation loss did not improve. Patience counter: 3/5

Continuous training started...
Epoch [20/20], Total_Loss_Epoch: 0.0127, Validation_Loss: 0.0106
Validation loss improved to 0.0106 at epoch 20

Average PSNR (Cover): 33.4782, Average SSIM (Cover): 0.9957
Average PSNR (Secret): 25.8709, Average SSIM (Secret): 0.9668

```

Hình 10: Log file model v4

3.1.3 Model v5

Tương tự với model v4, model v5 cũng được áp dụng tấn công và huấn luyện trên tập dataset COCO 2017, nhưng có tỷ lệ tấn công cao hơn.

Sau đây là các HYPERPARAMETER qua nhiều lần chạy:

```

# Hyperparameters
BATCH_SIZE = 8
IMAGE_SIZE = 256
EPOCHS = 20
LR = 0.0002
AR = 0.65
SIGMA = 0.03
WORKER = 4
LAMBDA = 1
INPUT_DIR = '/kaggle/input/PROJECT-NAME'
OUTPUT_DIR = '/kaggle/working'
LOG_FILE = 'log.txt'
ANGLE = [0, 90, 180, 270]
CONTINUOUS = False
CHECKPOINT =

```

Hình 11: Hyperparameter model v5

Trong đó:

- BATCH_SIZE: Kích thước batch đưa vào huấn luyện
- IMAGE_SIZE: Kích thước resize hình ảnh trước khi đưa vào huấn luyện
- EPOCHS: Tổng số epoch huấn luyện
- LR: Learning rate
- AR: Attack rate
- SIGMA: Thông số dung cho tấn công noise
- WORKER: Số lượng worker trong dataloader
- LAMBDA: Trọng số cho loss L_2 giữa secret – reveal
- INPUT_DIR: Đường dẫn tới notebook được huấn luyện trước đó
- OUTPUT_DIR: Đường dẫn tới output của notebook hiện tại
- LOG_FILE: Đường dẫn tới log file
- ANGLE: Chuỗi các góc dung trong tấn công xoay ảnh
- CONTINUOUS: Boolean để xác định huấn luyện từ đầu hay từ checkpoint
- CHECKPOINT: Đường dẫn để load checkpoint tiếp tục huấn luyện

Model v5 cũng được áp dụng các loại tấn công như model v4 nhưng tỷ lệ tấn công 65% và tăng mức độ nhiễu. Thời gian huấn luyện của model v5 cũng tương đương v4.

Sau đây là kết quả huấn luyện được ghi log file khi hoàn thành:

```

Training started...
Epoch [1/20], Total_Loss_Epoch: 0.0415, Validation_Loss: 0.0254
Validation loss improved to 0.0254 at epoch 1

Epoch [2/20], Total_Loss_Epoch: 0.0307, Validation_Loss: 0.0251
Validation loss improved to 0.0251 at epoch 2

Epoch [3/20], Total_Loss_Epoch: 0.0272, Validation_Loss: 0.0209
Validation loss improved to 0.0209 at epoch 3

Epoch [4/20], Total_Loss_Epoch: 0.0235, Validation_Loss: 0.0161
Validation loss improved to 0.0161 at epoch 4

Epoch [5/20], Total_Loss_Epoch: 0.0212, Validation_Loss: 0.0169
Validation loss did not improve. Patience counter: 1/5

Continuous training started...
Epoch [6/20], Total_Loss_Epoch: 0.0198, Validation_Loss: 0.0139
Validation loss improved to 0.0139 at epoch 6

Epoch [7/20], Total_Loss_Epoch: 0.0187, Validation_Loss: 0.0129
Validation loss improved to 0.0129 at epoch 7

Epoch [8/20], Total_Loss_Epoch: 0.0181, Validation_Loss: 0.0185
Validation loss did not improve. Patience counter: 1/5

Epoch [9/20], Total_Loss_Epoch: 0.0174, Validation_Loss: 0.0134
Validation loss did not improve. Patience counter: 2/5

Continuous training started...
Epoch [10/20], Total_Loss_Epoch: 0.0170, Validation_Loss: 0.0125
Validation loss improved to 0.0125 at epoch 10

Epoch [11/20], Total_Loss_Epoch: 0.0166, Validation_Loss: 0.0138
Validation loss did not improve. Patience counter: 1/5

Epoch [12/20], Total_Loss_Epoch: 0.0162, Validation_Loss: 0.0122
Validation loss improved to 0.0122 at epoch 12

```



```

Epoch [13/20], Total_Loss_Epoch: 0.0158, Validation_Loss: 0.0118
Validation loss improved to 0.0118 at epoch 13

Epoch [14/20], Total_Loss_Epoch: 0.0155, Validation_Loss: 0.0131
Validation loss did not improve. Patience counter: 1/5

Continuous training started...
Epoch [15/20], Total_Loss_Epoch: 0.0152, Validation_Loss: 0.0140
Validation loss improved to 0.0140 at epoch 15

Epoch [16/20], Total_Loss_Epoch: 0.0149, Validation_Loss: 0.0123
Validation loss improved to 0.0123 at epoch 16

Epoch [17/20], Total_Loss_Epoch: 0.0147, Validation_Loss: 0.0146
Validation loss did not improve. Patience counter: 1/5

Epoch [18/20], Total_Loss_Epoch: 0.0145, Validation_Loss: 0.0129
Validation loss did not improve. Patience counter: 2/5

Epoch [19/20], Total_Loss_Epoch: 0.0143, Validation_Loss: 0.0131
Validation loss did not improve. Patience counter: 3/5

Continuous training started...
Epoch [20/20], Total_Loss_Epoch: 0.0141, Validation_Loss: 0.0110
Validation loss improved to 0.0110 at epoch 20

Average PSNR (Cover): 30.3018, Average SSIM (Cover): 0.9906
Average PSNR (Secret): 24.4369, Average SSIM (Secret): 0.9616

```

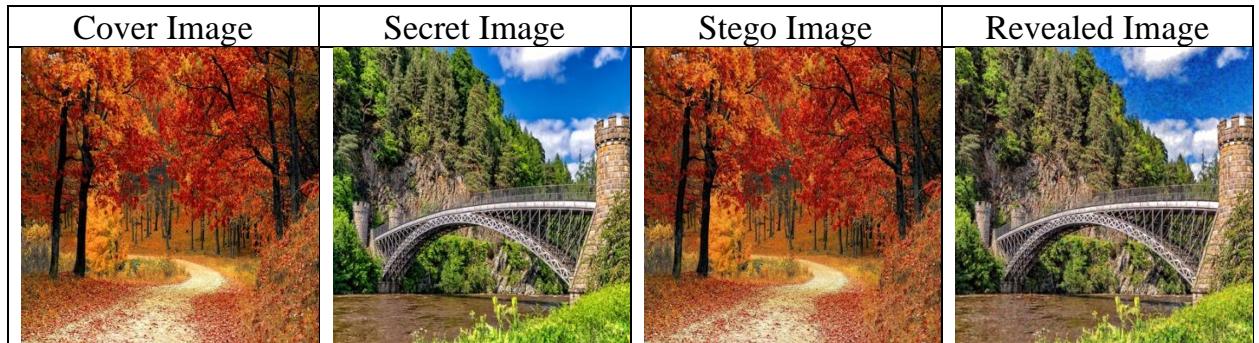
Hình 12: Log file model v5

3.1.4 Kết quả chung – Nhận xét

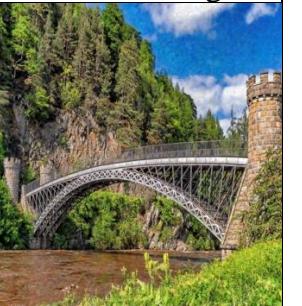
Nhìn chung các version đều đạt được kết quả tốt chỉ sau 20 epoch, trong bài báo gốc model được huấn luyện tới 150 epoch. Loss của version giảm đều sau mỗi epoch nhưng càng nhiều epoch thì loss giảm càng chậm.

Kết quả không tấn công:

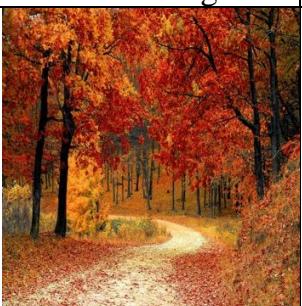
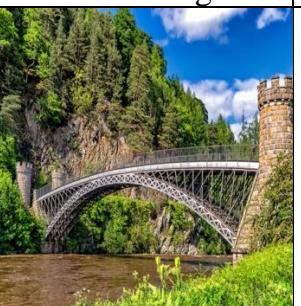
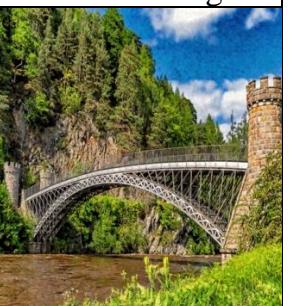
- Model v2:



- Model v4:

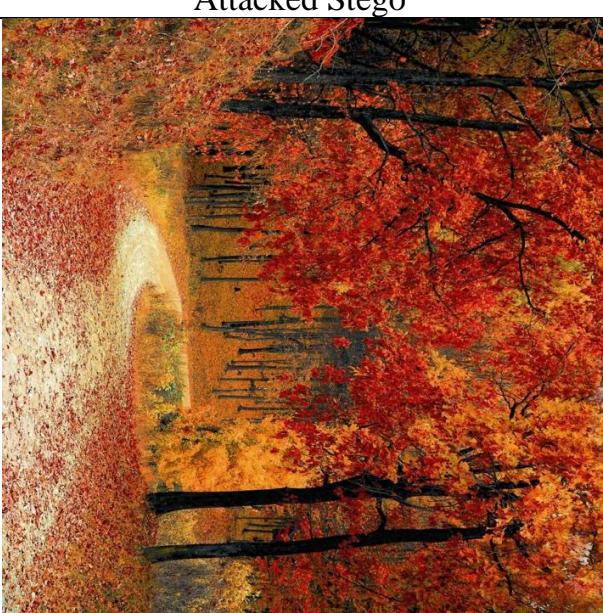
Cover Image	Secret Image	Stego Image	Revealed Image
			

- Model v5:

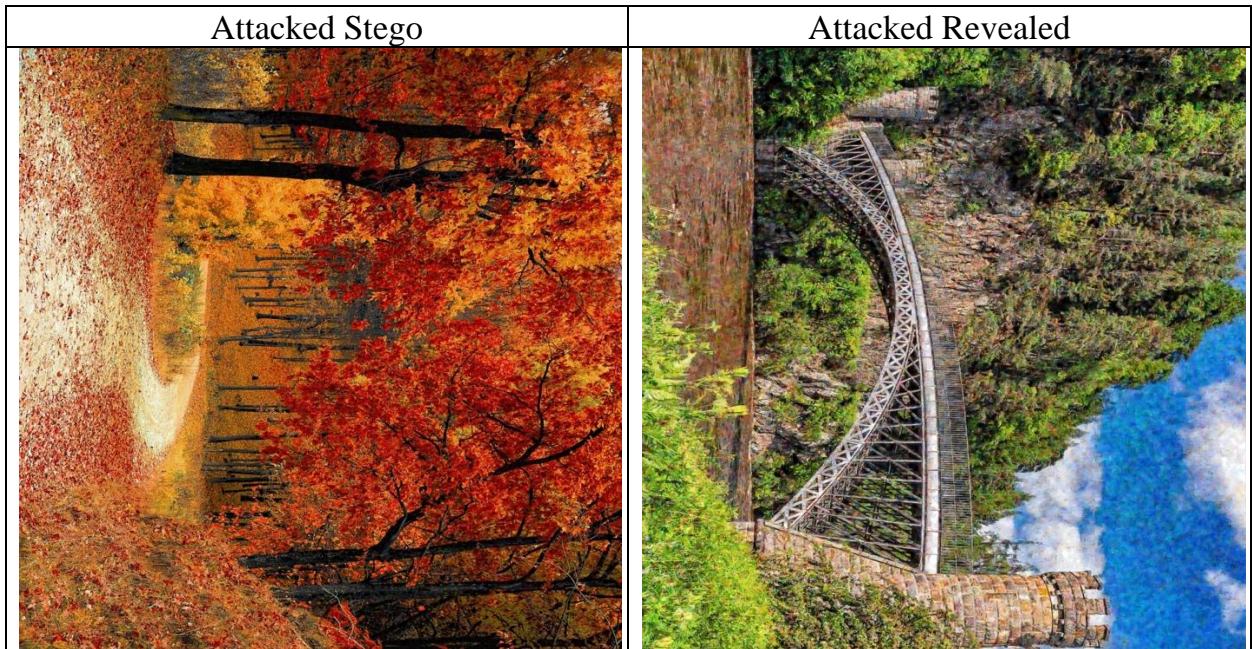
Cover Image	Secret Image	Stego Image	Revealed Image
			

Kết quả khi tấn công stego:

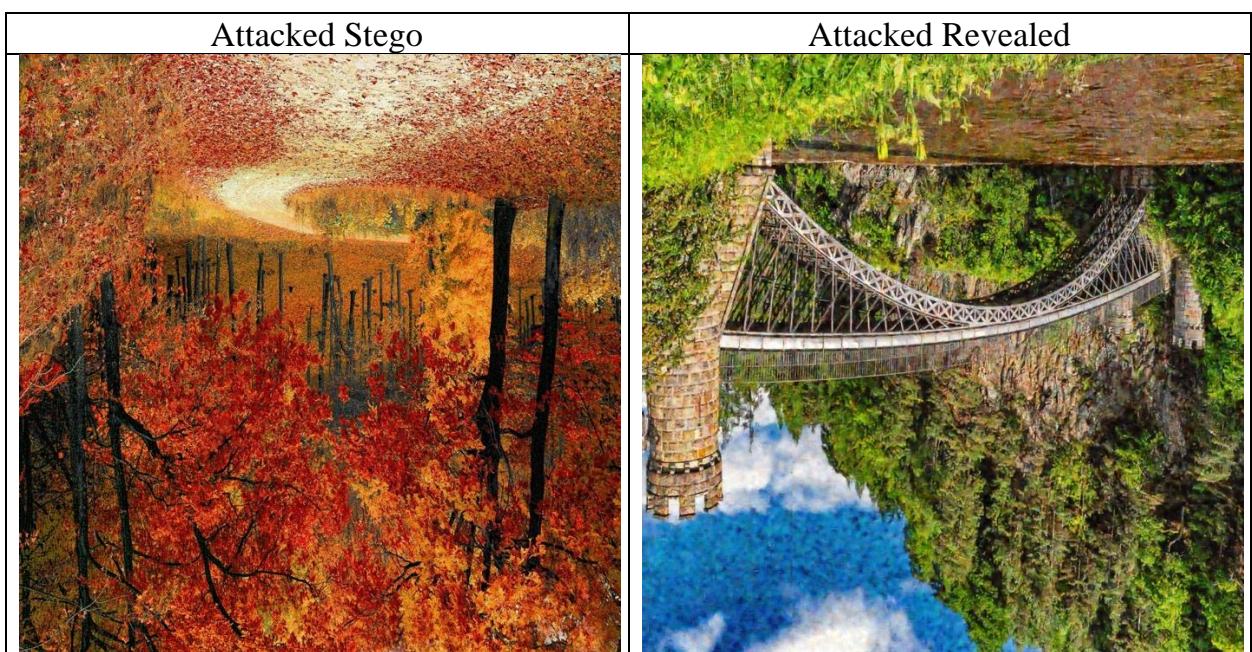
- Model v2:

Attacked Stego	Attacked Revealed
	

- Model v4:



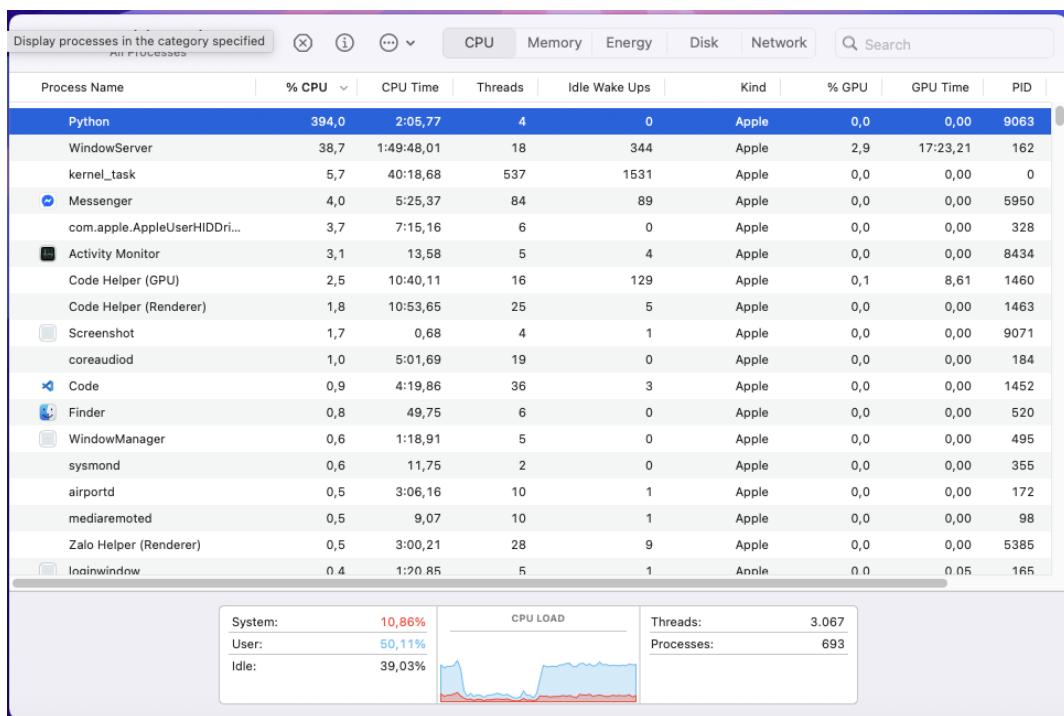
- Model v5:



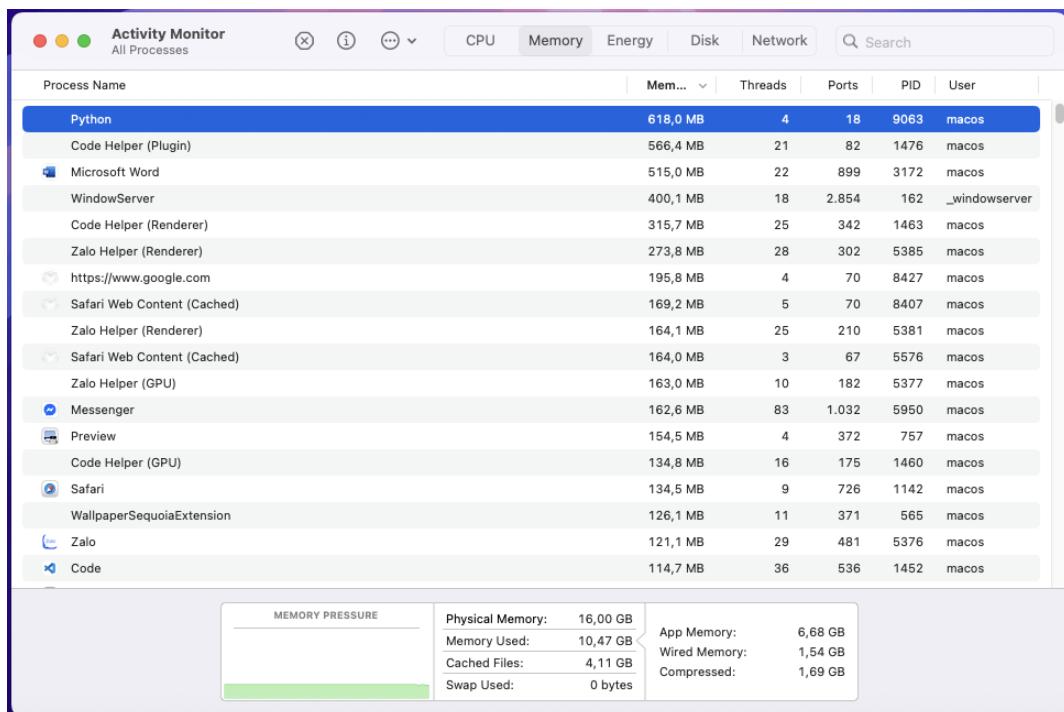
3.2 Đánh giá

3.2.1 Performance

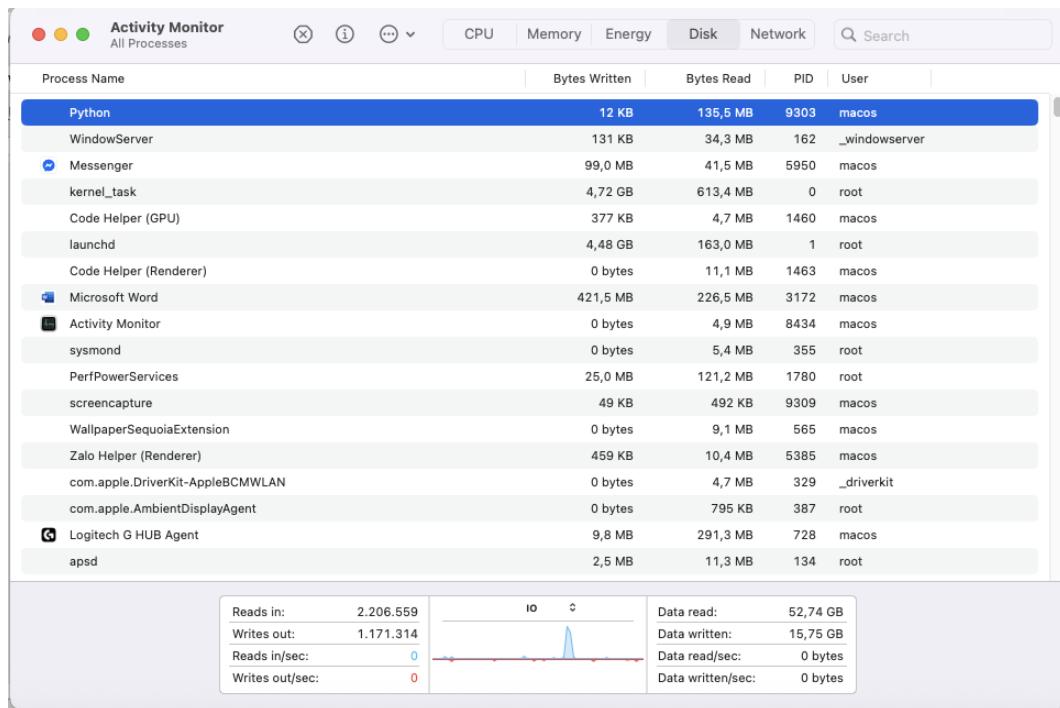
Performance của MacBook Air M2 khi thực hiện quá trình encode và decode, sử dụng model v4, tùy vào kích thước hình ảnh thì thời gian sẽ lâu hoặc nhanh nhưng không ảnh hưởng đến resource của máy vì chia block cho hình ảnh:



Hình 13: Performance CPU



Hình 14: Performance RAM



Hình 15: Performance Disk

3.2.2 Capacity

Decode rate xác định tỷ lệ bit khác nhau giữa secret image và reveal image, được tính bằng công thức sau:

$$\text{Decoded Rate} = 1 - \frac{\sum_{i=1}^N \sum_{j=1}^M |H_{i,j} - D_{i,j}|}{N \times M},$$

Trong đó:

- N, M: kích thước của hình ảnh
- H: secret image
- D: reveal image

Capacity được xác định từ decode rate như sau:

$$\text{Capacity} = \text{Decoded Rate} \times 8 \times 3 \text{ (bpp)}$$

Decode Rate phải nhân với 24 (8 x 3) vì model thực hiện nhúng ảnh RGB vào ảnh RGB khác nên capacity lý tưởng 24 bits

Capacity thực tế được tính với model v4 và cặp ảnh secret – reveal có kích thước 256 x 256 giống trong huấn luyện:

```
(covertchannel3.13) macos@Tas-MacBook-Air covertchannel % python capacity.py
Secret Image Shape: (256, 256, 3)
Reveal Image Shape: (256, 256, 3)
Decoded Rate: 0.9553
Capacity: 22.9261 bits per pixel
```

Hình 16: Capacity 256 x 256

Capacity thực tế với model v4 và cặp ảnh secret – reveal có kích thước 1024 x 1024 sẽ đạt kết quả thấp hơn vì phải chia block và nội suy stego, reveal:

```
(covertchannel3.13) macos@Tas-MacBook-Air covertchannel % python capacity.py
Secret Image Shape: (1024, 1024, 3)
Reveal Image Shape: (1024, 1024, 3)
Decoded Rate: 0.9469
Capacity: 22.7265 bits per pixel
```

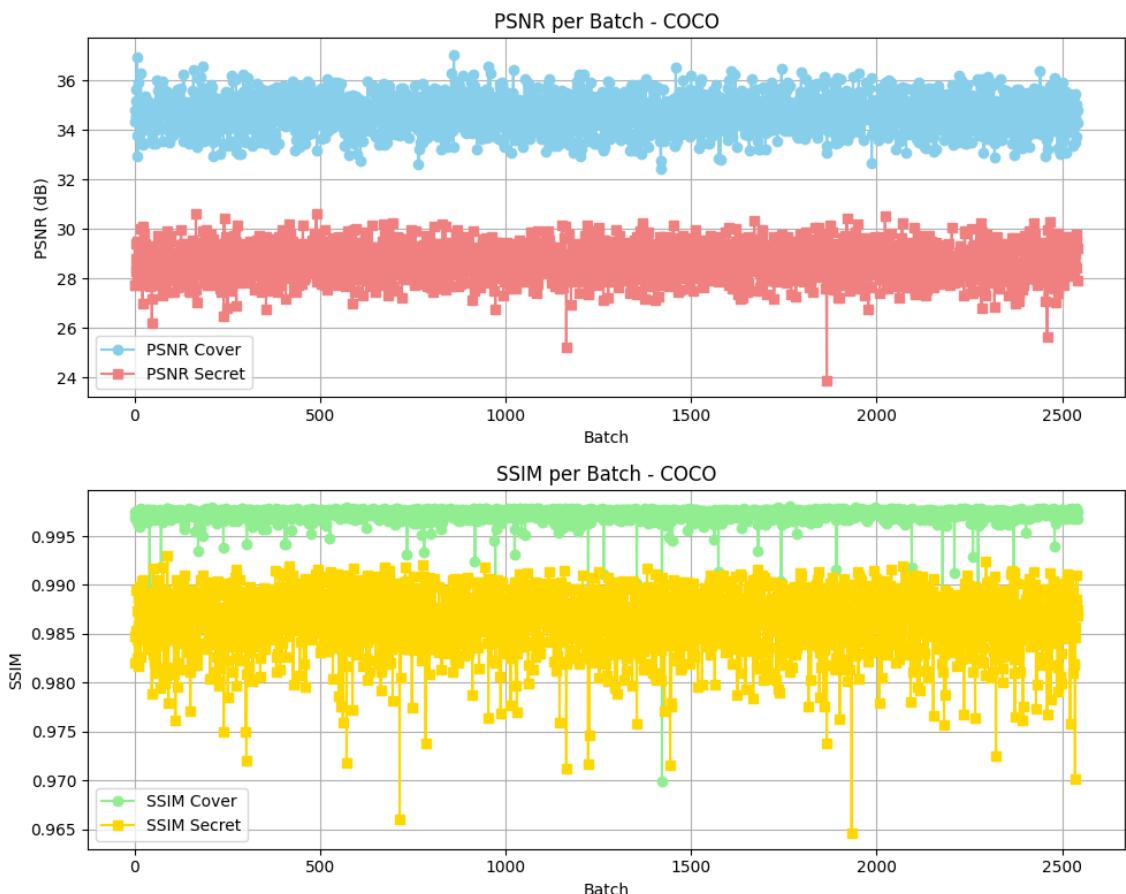
Hình 17: Capacity 1024 x 1024

Capacity thực tế sẽ xấp xỉ 23 bpp.

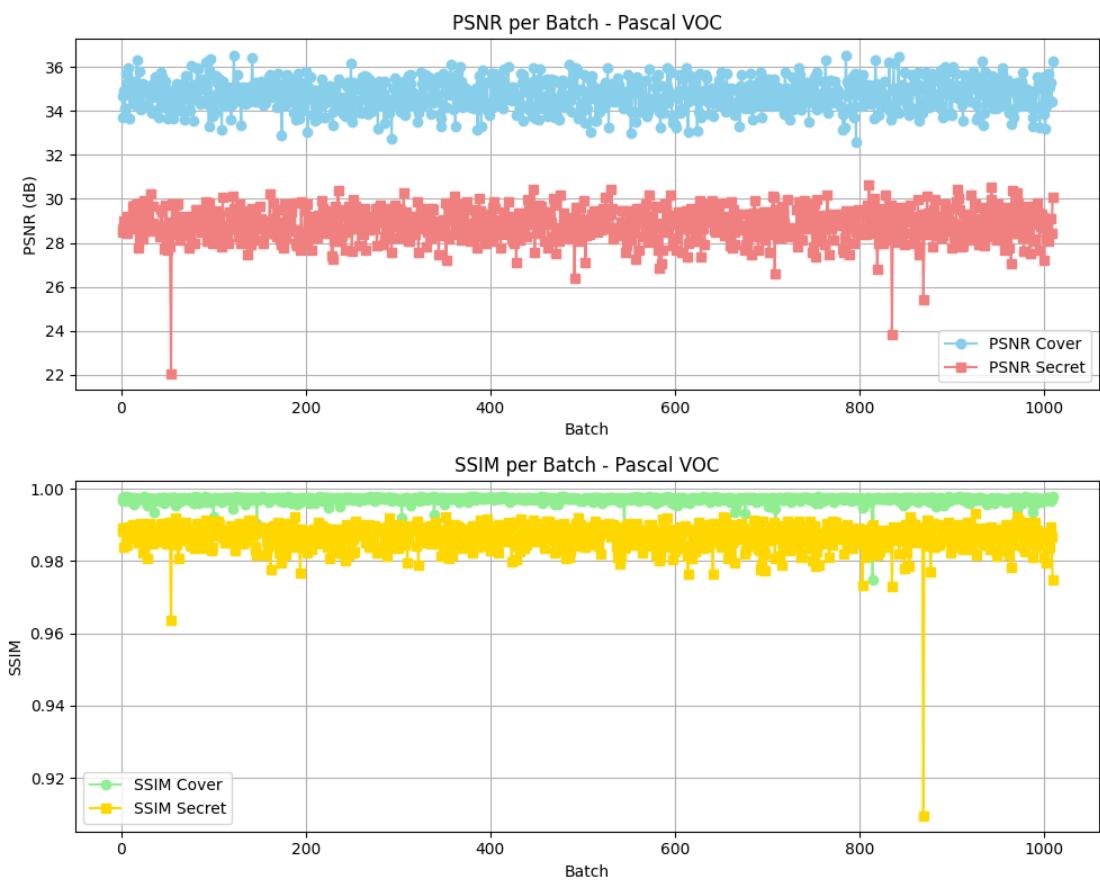
3.2.3 PNRS, SSIM

Model v2, v4, v5 sẽ lần lượt được kiểm thử trên các dataset COCO, Pascal VOC, ImageNET:

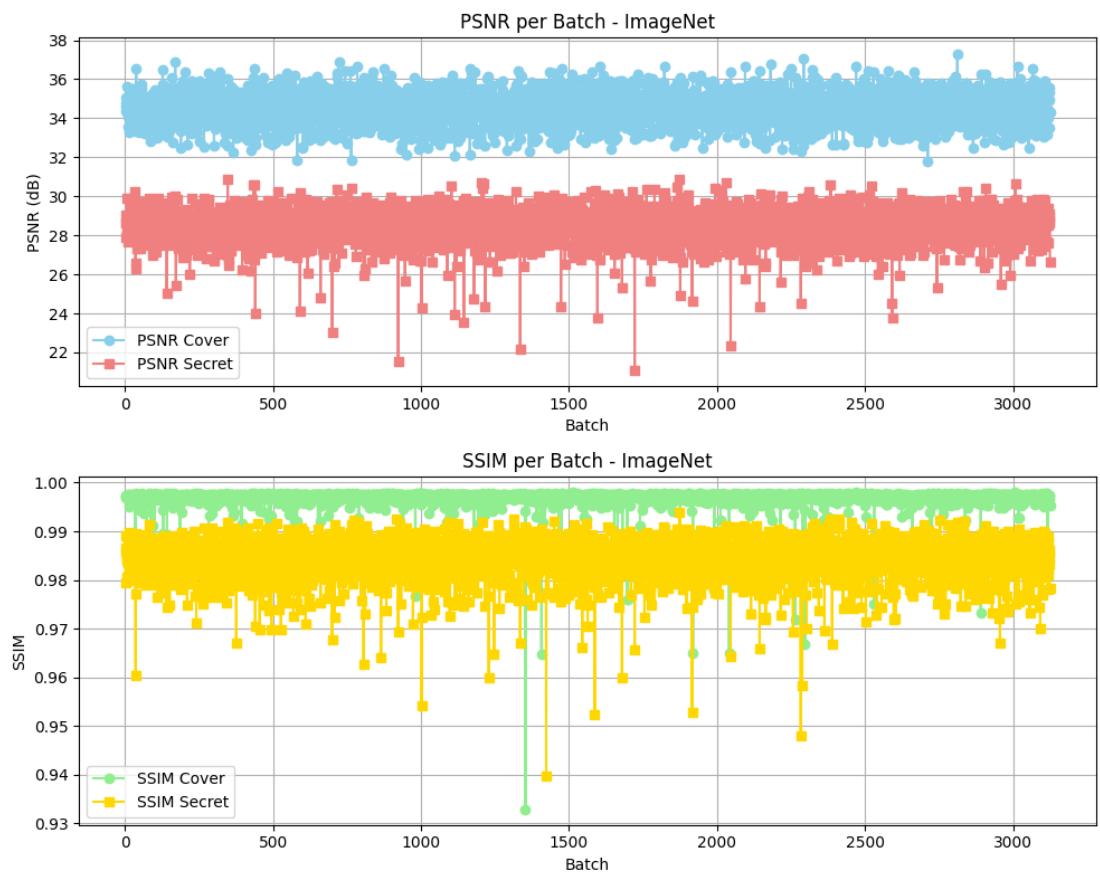
- Model v2:



Hình 18: Model v2 COCO



Hình 19: Model v2 Pascal VOC



Hình 20: Model v2 ImageNET

Pascal VOC – Average PSNR (Cover): 34.7238, Average SSIM (Cover): 0.9972
 Pascal VOC – Average PSNR (Secret): 28.8274, Average SSIM (Secret): 0.9867
 ImageNet – Average PSNR (Cover): 34.5192, Average SSIM (Cover): 0.9968
 ImageNet – Average PSNR (Secret): 28.4351, Average SSIM (Secret): 0.9844
 COCO – Average PSNR (Cover): 34.6866, Average SSIM (Cover): 0.9973
 COCO – Average PSNR (Secret): 28.6903, Average SSIM (Secret): 0.9865

Hình 21: Model v2 average PSNR and SSIM

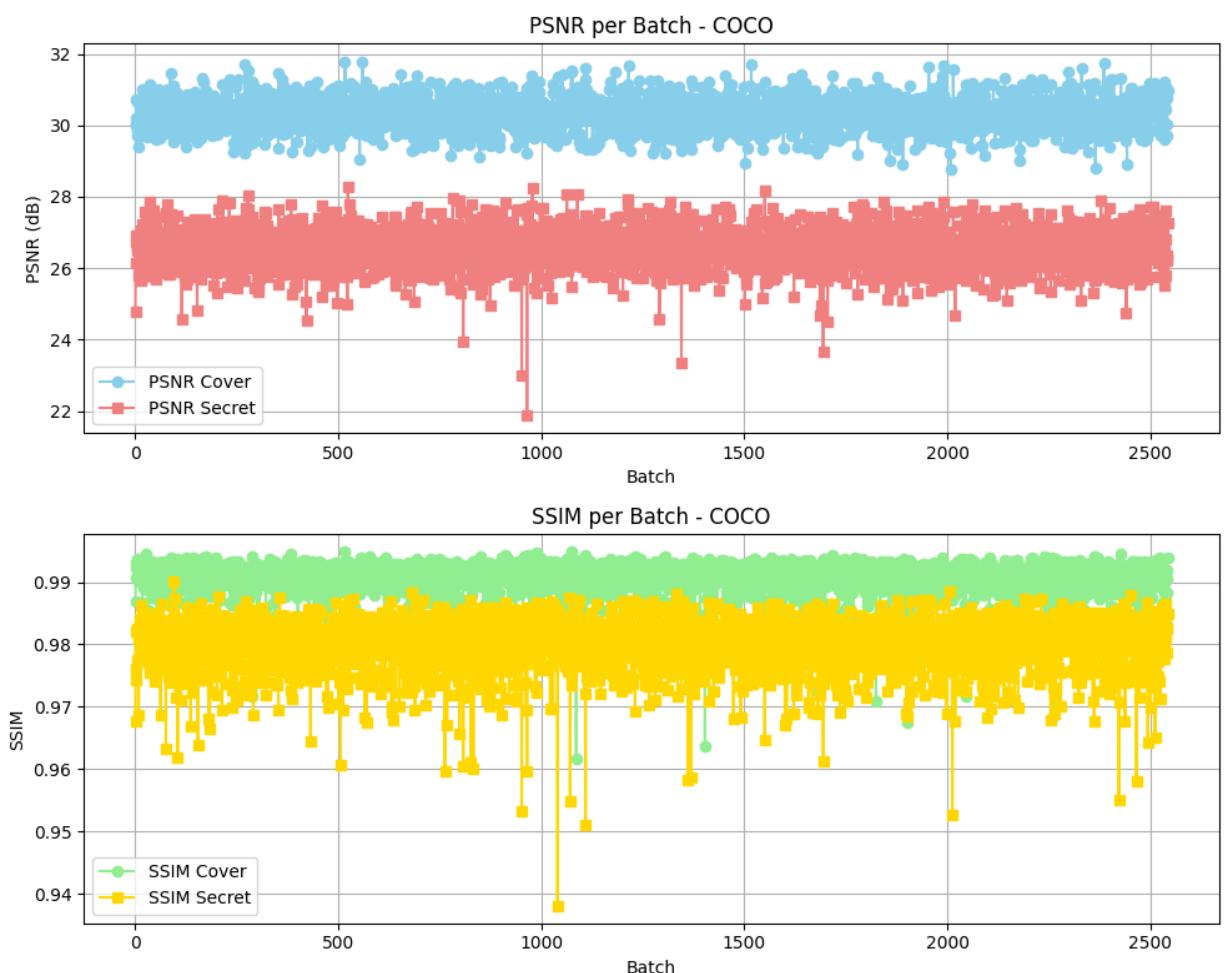
- Model v4 vì tại lỗi trong quá trình huấn luyện mà chỉ ghi nhận text của COCO và Pascal VOC:

Pascal VOC – Average PSNR (Cover): 33.4687, Average SSIM (Cover): 0.9956
 Pascal VOC – Average PSNR (Secret): 21.7988, Average SSIM (Secret): 0.8985

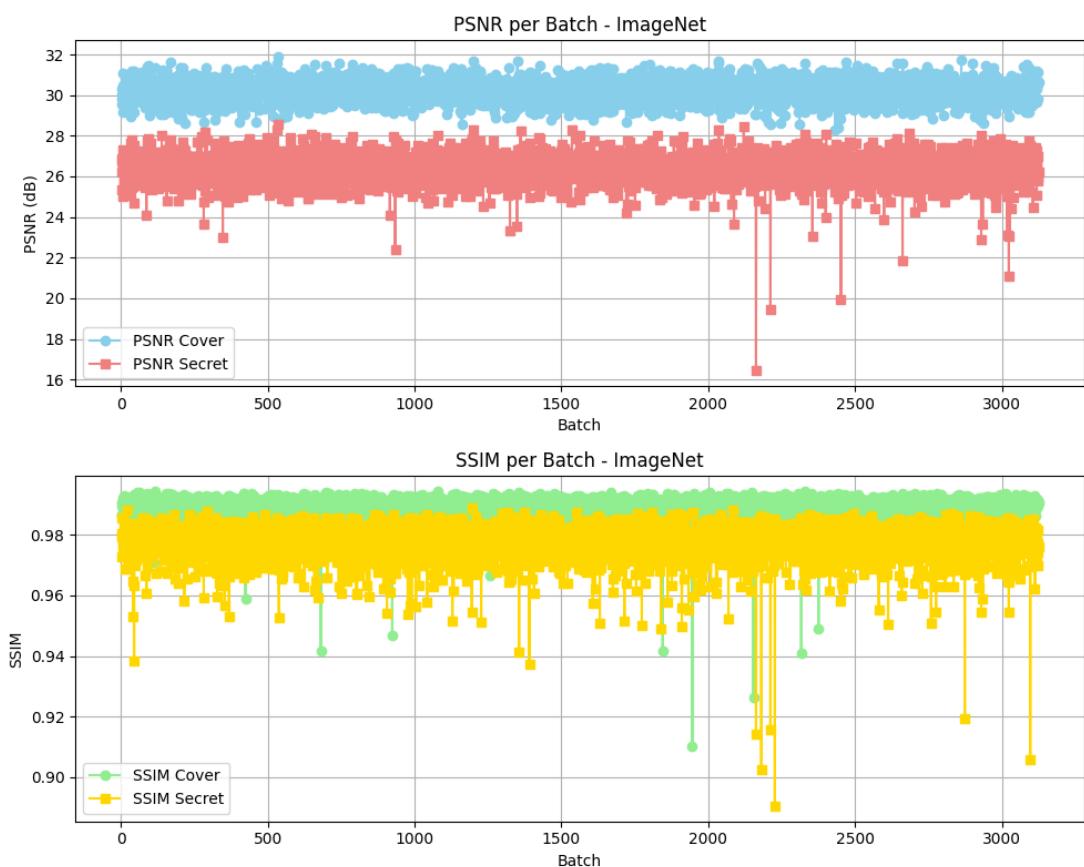
COCO – Average PSNR (Cover): 33.4782, Average SSIM (Cover): 0.9957
 COCO – Average PSNR (Secret): 25.8709, Average SSIM (Secret): 0.9668

Hình 22: Model v4 average PSNR and SSIM

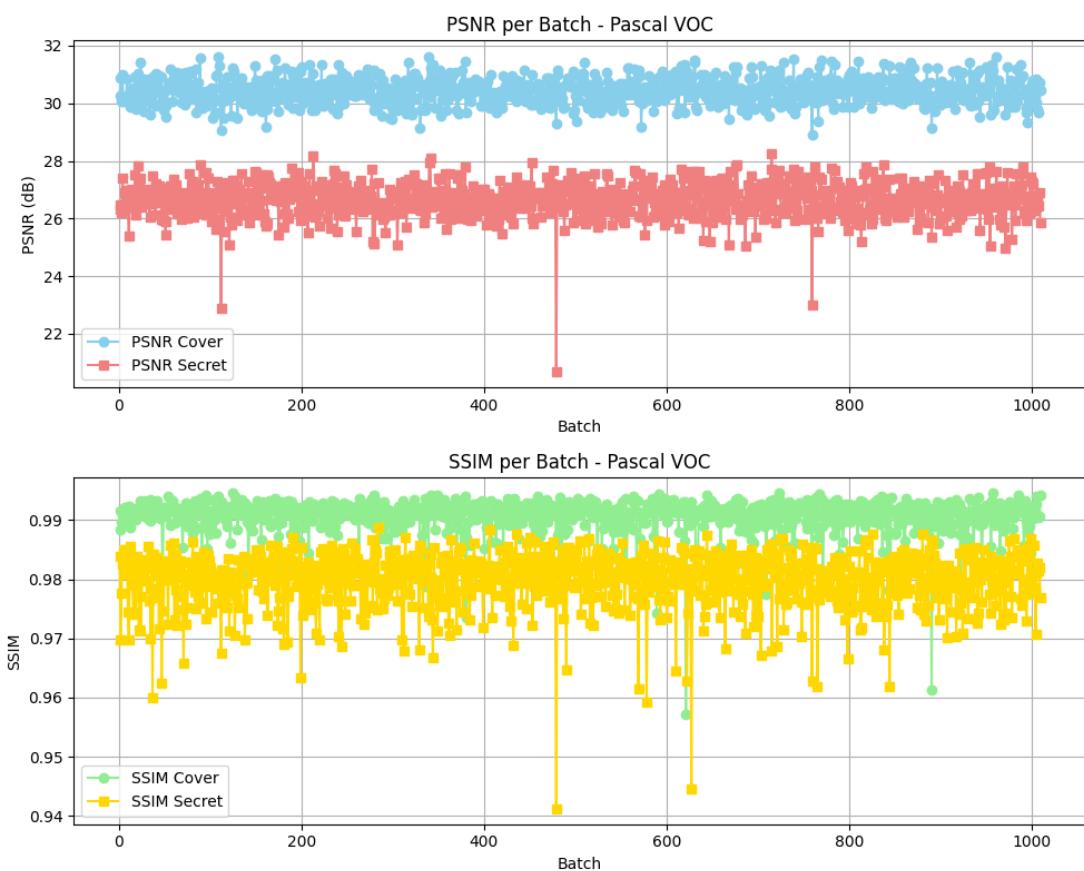
- Model v5:



Hình 23: Model v5 COCO



Hình 24: Model v5 ImageNET



Hình 25: Model v5 Pascal VOC

Trong bài báo sử dụng các model từ bài báo khác để so sánh với model của họ trên nhiều tập dataset khác nhau:

Model	Images dataset	Cover-stego		Secret-revealed	
		PSNR	SSIM	PSNR	SSIM
Rehman et al. [14]	LFW	33.7	0.95	39.9	0.96
	P.-V.12	33.7	0.96	35.9	0.95
Zhang et al. [16]	LFW	34.63	0.9573	33.63	0.9429
	P.-V.12	34.49	0.9661	33.31	0.9467
Proposed	LFW	38.35	0.9659	34.90	0.9505
$L_2 + L_2$	P.-V.12	35.94	0.9731	32.95	0.9532
Proposed	LFW	40.03	0.9797	33.13	0.9280
$L_1 + V + L_2$	P.-V.12	37.40	0.9790	30.80	0.9094

Tuy bài báo đạt được kết quả tốt với PSNR, SSIM cao nhưng bài báo training model với 150 epoch. Model của nhóm đạt SSIM tốt hơn so với bài báo và PSNR không kém nhiều lăm.

Chương IV: TRIỂN KHAI VÀ ĐÓNG GÓI

4.1 Chuẩn bị

- Yêu cầu hệ thống:
 - Python 3.12 hoặc tương thích với mô hình
 - Windows OS (khuyến nghị 64-bit)
 - Đảm bảo các file model (.h5 hoặc .pth) và dữ liệu phụ trợ (file cấu hình, asset ảnh, v.v.) có sẵn
- Cài đặt PyInstaller:

```
<python_path> -m pip install pyinstaller
```

Ví dụ:

```
E:\Pythons\Python312stego\python.exe -m pip install pyinstaller
```

4.2 Đóng gói với PyInstaller

Thực hiện đóng gói với lệnh sau:

```
<pyinstaller_path> --onefile \
--add-data "Model_v4/stegano_model_final.pth:results" \
--add-data "model/model_v2.py:model" \
-F datahiding.py
```

Ví dụ:

```
E:\Pythons\Python312stego\Scripts\pyinstaller.exe --onefile \
--add-data "Model_v4/stegano_model_final.pth:results" \
--add-data "model/model_v2.py:model" \
-F datahiding.py
```

Giải thích:

- <pyinstaller_path>: đường dẫn đến lệnh pyinstaller, ví dụ:
 - Nếu đã cài PyInstaller toàn cục: pyinstaller
 - Nếu dùng Python riêng: E:\Pythons\Python312stego\Scripts\pyinstaller.exe
- --onefile hoặc -F: tạo duy nhất một file .exe
- --add-data: thêm các file hoặc thư mục phụ thuộc vào file .exe. Cú pháp:
 - "source_path:target_folder" (Windows dùng dấu : còn Linux/Mac dùng : hoặc ;)

- `datahiding.py`: file mã nguồn chính chạy chương trình

Chương V: KIỂM THỬ

5.1 Mục tiêu kiểm thử

Mục tiêu của giai đoạn kiểm thử là đánh giá hiệu quả, độ tin cậy và tính an toàn của hệ thống ẩn thông tin sử dụng mạng CNN Auto-Encoder kết hợp với Generative Adversarial Network (GAN) trong truyền thông số. Cụ thể, quá trình kiểm thử hướng đến các mục tiêu sau:

- Đánh giá khả năng ẩn thông tin mà không gây biến đổi rõ rệt về mặt thị giác (tính không thể phát hiện bằng mắt thường).
- Đo lường độ chính xác của quá trình giải mã và khôi phục thông tin đã giấu sau khi truyền qua môi trường giả lập.
- Kiểm tra khả năng kháng lại các phương pháp tấn công phát hiện (detection attacks).
- Đánh giá hiệu suất thực thi của hệ thống trong điều kiện môi trường thực tế.

5.2 Môi trường và công cụ kiểm thử

Hệ thống được kiểm thử thông qua **file thực thi (.exe)** được đóng gói từ mã nguồn Python bằng công cụ **PyInstaller**, giúp việc triển khai và thử nghiệm trở nên dễ dàng, không phụ thuộc vào môi trường Python cục bộ của người dùng.

- Hệ điều hành:** Windows 11 Pro 64-bit
- CPU:** Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 4 Cores / 8 Threads
- RAM:** 20 GB
- Python build:** Đóng gói sử dụng PyInstaller tạo ra file thực thi .exe

Quy trình kiểm thử được chia thành bốn nhóm chính: kiểm thử chức năng, kiểm thử hiệu suất, kiểm thử tính ẩn và kiểm thử bảo mật.

5.3 Tiến hành kiểm thử

#	Title	Description	Input	Expect
1	Giấu text ngắn	Giấu "Hello World!" vào ảnh, upload, tải về, giải mã	Text ngắn + ảnh nền JPG	Giải mã ra đúng "Hello World!"
2	Giấu text dài	Giấu text ~75 ký tự, upload, tải về, giải mã	Text dài + ảnh nền JPG	Text đầy đủ, có thể đọc được.

3	Giấu file ảnh nhỏ	Giấu 1 file ảnh nhỏ (~15KB), upload, tải về, giải mã	File ảnh PNG + ảnh nền JPG	File ảnh giải mã thành công, mở xem được
4	Giấu file ảnh lớn	Giấu file ảnh 1MB vào ảnh nền	File secret 1MB + ảnh nền lớn	Thành công nếu ảnh nền đủ lớn, báo lỗi nếu không đủ
5	Upload nhiều lần	Upload lên web social → download → re-upload → re-download → decode	Text "Persistence test" + ảnh nền	Text không bị mất sau nhiều lần tải

Test case 1: Giấu text ngắn

Ảnh đầu vào: SonDoong.jpg



Thông điệp: "Hello World!"

Lệnh sử dụng:

```
.\datahiding.exe .\image\cover\SonDoong.jpg
.\image\secret\hello.txt
```

Hoặc

```
<python_path> .\datahiding.py encode .\image\cover\SonDoong.jpg
.\image\secret\hello.tx
```

Ảnh sau khi nhúng:



Hiệu năng khi chạy code:

Name	Status	83% CPU	41% Memory	0% Disk	0% Network
Python		62.7%	899.9 MB	0 MB/s	0 Mbps

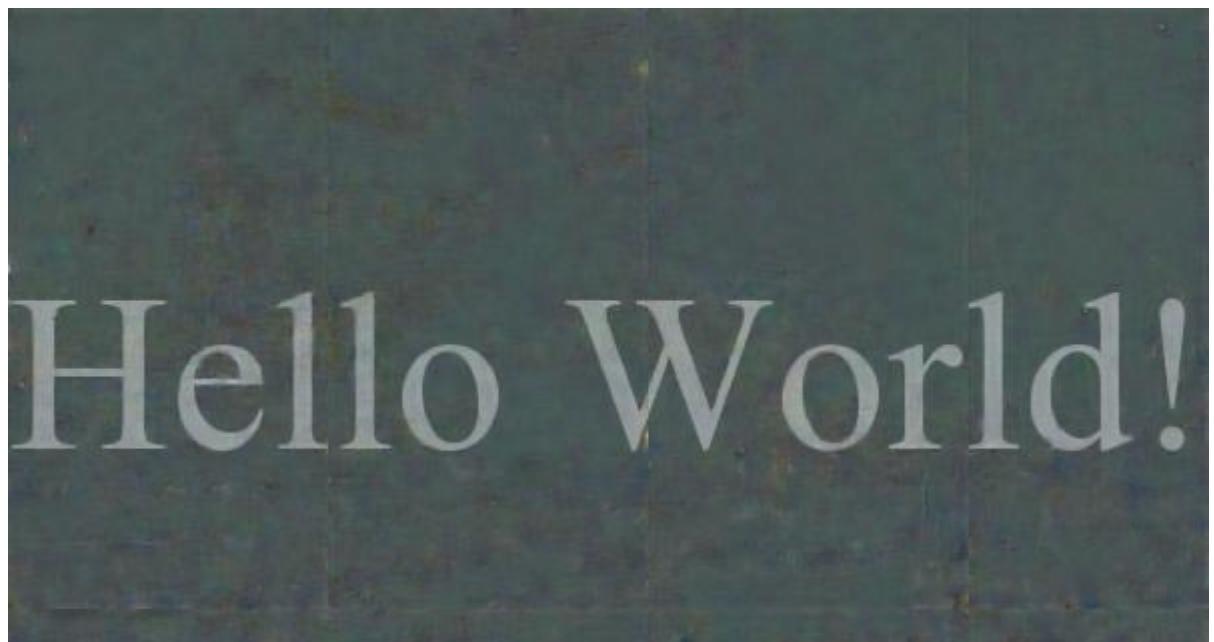
Lệnh sử dụng:

```
.\datahiding.exe decode .\dist\stego_image.png
```

Hoặc

```
<python_path> .\datahiding.py decode .\dist\stego_image.png
```

Secret được lấy ra:



Hiệu năng khi chạy decode:

Name	Status	60% CPU	39% Memory	0% Disk	0% Network
Python		48.9%	441.8 MB	0 MB/s	0 Mbps

Nhận xét:

Giải mã chính xác 100% nội dung

Đạt yêu cầu

Test case 2: Giấu text dài

Ảnh đầu vào: SonDoong.jpg



Thông điệp: " VT-9204ZX | deadline 72h | IP:10.10.55.88:8080 | Key:BXZ19v2 | Key:BXZ19v7"

Lệnh sử dụng:

```
.\datahiding.exe .\image\cover\SonDoong.jpg
.\image\secret\info.txt
```

Hoặc

```
<python_path> .\datahiding.py encode .\image\cover\SonDoong.jpg
.\image\secret\info.tx
```

Ảnh sau khi nhúng:



Hiệu năng khi chạy code:

Name	Status	75% CPU	44% Memory	0% Disk	0% Network
Python		67.3%	997.9 MB	0 MB/s	0 Mbps

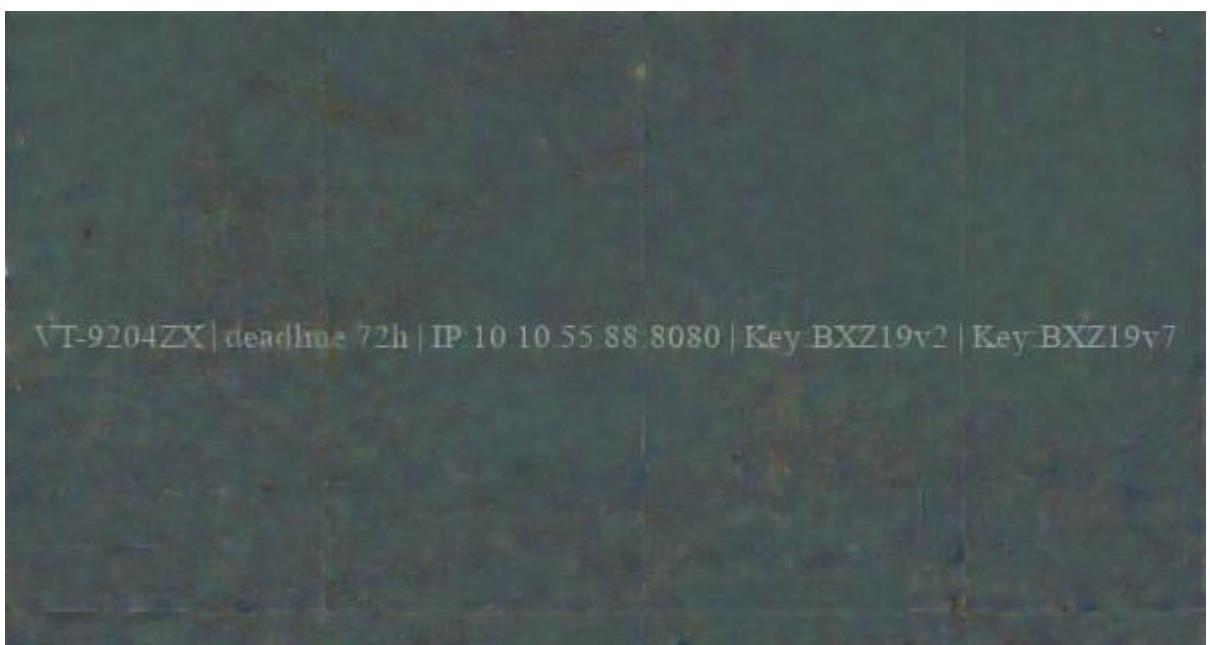
Lệnh sử dụng:

```
.\datahiding.exe decode .\dist\stego_image.png
```

Hoặc

```
<python_path> .\datahiding.py decode .\dist\stego_image.png
```

Secret được lấy ra:



Hiệu năng khi chạy decode:

Name	Status	71% CPU	42% Memory	0% Disk	0% Network
Python		58.7%	407.1 MB	0 MB/s	0 Mbps

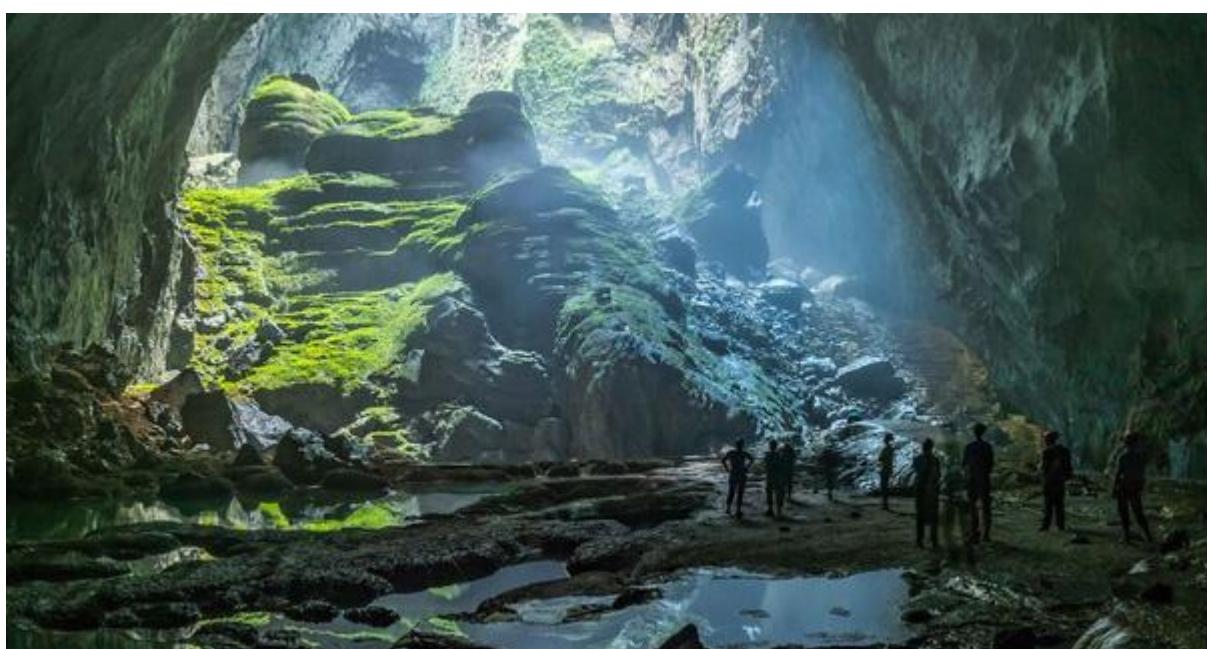
Nhận xét:

Giải mã chính xác 100% nội dung, có thể đọc được

Đạt yêu cầu

Test case 3: Giấu file ảnh nhỏ

Ảnh đầu vào: SonDoong.jpg



Ảnh secret:



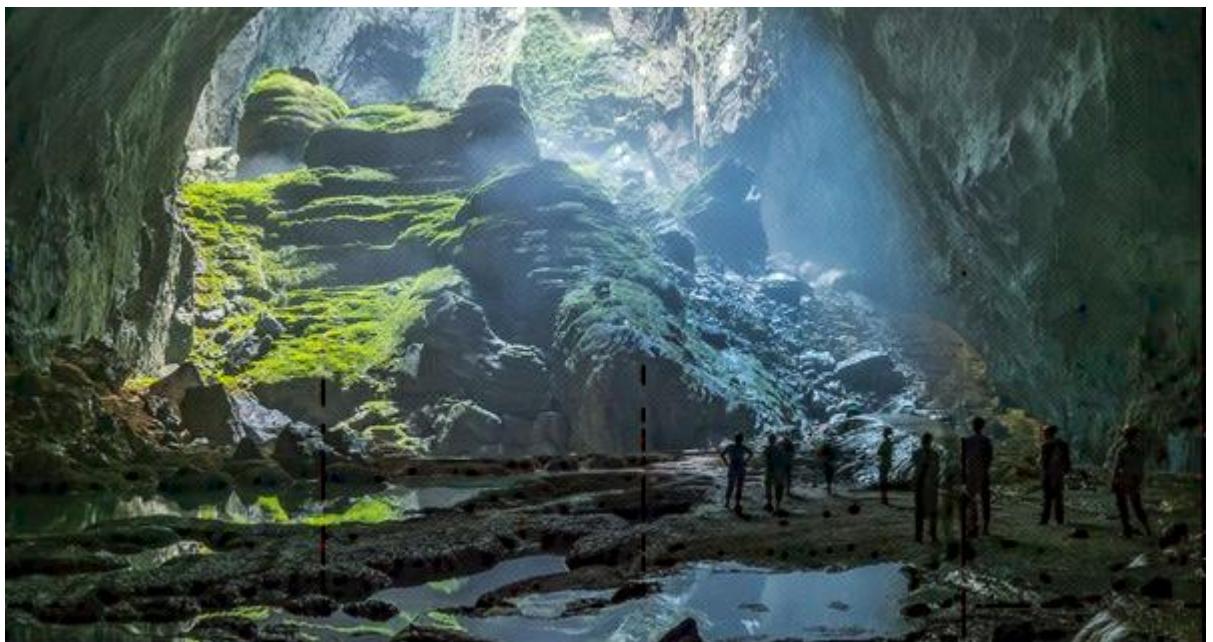
Lệnh sử dụng:

```
.\datahiding.exe .\image\cover\SonDoong.jpg .\image\secret\plane.jpg
```

Hoặc

```
<python_path> .\datahiding.py encode .\image\cover\SonDoong.jpg .\image\secret\plane.jpg
```

Ảnh sau khi nhúng:



Hiệu năng khi chạy code:

Name	Status	CPU		Memory		Disk		Network	
		76%	44%	0%	0%	0 MB/s	0 Mbps		
Python		70.3%	1,009.3 MB			0 MB/s	0 Mbps		

Lệnh sử dụng:

```
.\datahiding.exe decode .\dist\stego_image.png
```

Hoặc

```
<python_path> .\datahiding.py decode .\dist\stego_image.png
```

Secret được lấy ra:



Hiệu năng khi chạy decode:

Name	Status	36% CPU	41% Memory	1% Disk	0% Network
Python		23.4%	256.8 MB	0 MB/s	0 Mbps

Nhận xét:

Giải mã chính xác nội dung.

Đạt yêu cầu

Test case 4: Giấu file ảnh lớn

Ảnh đầu vào: SonDoong.jpg



Ảnh secret (1MB):



Lệnh sử dụng:

```
.\datahiding.exe .\image\cover\SonDoong.jpg .\image\secret\  
1mb.jpg
```

Hoặc

```
<python_path> .\datahiding.py encode .\image\cover\SonDoong.jpg  
.\\image\\secret\\1mb.jpg
```

Ảnh sau khi nhúng:



Hiệu năng khi chạy code:

Name	Status	76% CPU	44% Memory	0% Disk	0% Network
Python		70.3%	1,009.3 MB	0 MB/s	0 Mbps

Lệnh sử dụng:

```
.\datahiding.exe decode .\dist\stego_image.png
```

Hoặc

```
<python_path> .\datahiding.py decode .\dist\stego_image.png
```

Secret được lấy ra:



Hiệu năng khi chạy decode:

Name	Status	50% CPU	46% Memory	0% Disk	0% Network
Python		37.4%	477.3 MB	0 MB/s	0 Mbps

Nhận xét:

Giải mã chính xác nội dung, chất lượng hình ảnh thấp hơn ảnh secret ban đầu.

- Đạt yêu cầu

Test case 5: Upload nhiều lần

Thực thi như test case 1 nhưng reupload nhiều lần lên social:

<https://social.hoanganhngo.id.vn/>

Nhận xét:

Sau khi reupload nhiều lần vẫn giải mã chính xác 100% nội dung, có thể đọc được

- Đạt yêu cầu

Chương VI: KẾT LUẬN

Mô hình steganography dựa trên học sâu mà chúng tôi phát triển đã chứng minh được sức mạnh vượt trội. Nó có thể giấu một ảnh bí mật vào ảnh cover mà vẫn giữ cho ảnh stego trông gần như giống hệt ảnh gốc. Cho thấy ảnh stego không chỉ đẹp mắt mà còn cho phép trích xuất dữ liệu ẩn với độ chính xác đáng kinh ngạc.

Việc sử dụng các mô-đun convolution giãn nở và skip connections đã giúp mô hình tận dụng cả những chi tiết nhỏ lẩn bối cảnh tổng thể của ảnh. Đặc biệt, chiến lược huấn luyện với các biến đổi ngẫu nhiên và nhiễu bổ sung đã làm cho mô hình trở nên "cứng cáp" hơn, sẵn sàng đối phó với những thách thức thực tế như nén ảnh hay chỉnh sửa.

Tuy nhiên, mô hình vẫn còn một số điểm cần cải thiện. Nó chưa thực sự xuất sắc khi đối mặt với các cuộc tấn công phức tạp hoặc khi cần giấu lượng dữ liệu lớn hơn. Những hạn chế này sẽ là động lực để chúng tôi tiếp tục nghiên cứu và hoàn thiện.

Trong tương lai, chúng tôi dự định tích hợp các kỹ thuật tiên tiến như huấn luyện đối kháng (adversarial training) và cơ chế chú ý (attention) để tăng cường hiệu suất và bảo mật. Đồng thời, việc mở rộng khả năng xử lý các loại biến đổi đa dạng hơn sẽ giúp mô hình linh hoạt hơn trong các kịch bản thực tế.

Tóm lại, nghiên cứu này đánh dấu một bước tiến quan trọng trong lĩnh vực steganography, cho thấy tiềm năng to lớn của học sâu trong việc bảo vệ thông tin số. Mô hình không chỉ mang đến một giải pháp giấu tin an toàn mà còn mở ra nhiều triển vọng ứng dụng trong bảo mật, tình báo, và hơn thế nữa.

References

- [1] Kich, I., & Taouil, Y. (2021). CNN auto-encoder network using dilated inception for image steganography. *International Journal of Fuzzy Logic and Intelligent Systems*, 21(4), 358-368.
- [2] Zhang, K. A., Cuesta-Infante, A., Xu, L., & Veeramachaneni, K. (2019). SteganoGAN: High capacity image steganography with GANs. *arXiv preprint arXiv:1901.03892*.
- [3] Kich, I., Taouil, Y., & Benhfid, A. (2021, May). Image steganography scheme using dilated convolutional network. In *2021 12th international conference on information and communication systems (ICICS)* (pp. 305-309). IEEE.
- [4] Pin Wu 1 ID , Yang Yang 1 ID and Xiaoqiang L (2018, June). StegNet: Mega Image Steganography Capacity with Deep Convolutional Network.