Analisis Pengaruh Watermark Least Significant Bit Pada Objek Terdeteksi Sebagai Bentuk Adversarial Attack

Safiq Faray (13519145)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519145@std.stei.itb.ac.id

***Abstract*— Steganography adalah sebuah metode untuk menyisipkan pesan ke dalam suatu data lain seperti citra yang tidak kelihatan oleh kasat mata. Terdapat banyak jenis steganography yang dapat dilakukan dan beberapa diantaranya dapat digunakan sebagai bentuk adversarial attack secara black-box. *Steganography* dengan menyisipkan pesan pada *least significant bit* pada pixel citra merupakan metode *steganography* sederhana yang belum pernah dievaluasi kinerjanya untuk *adversarial attack* secara *black box*. Hasil yang didapat adalah metode ini tidak dapat digunakan untuk *adversarial attack*, karena memiliki 0% *fooling rate*.**

Keywords—Adversarial attack; black box; steganography; least significant bit

I. PENDAHULUAN

Adversarial attack merupakan bentuk serangan terhadap suatu model pembelajaran mesin atau pembelajaran mendalam untuk mempengaruhi hasil prediksi agar jauh dari target. Pada klasifikasi citra, *adversarial attack* digunakan agar membuat model salah memprediksi klasifikasi citra.

Terdapat dua jenis *adversarial attack*, yaitu *white-box* dan *black-box*. *Adversarial attack white-box* adalah serangan yang dilakukan dengan mengetahui gradien dari sebuah model yang menjadi target. Contoh tipe serangan *white-box* adalah *fast gradient sign method* (FSGM) [1] yang cara kerjanya adalah menghitung gradien dari sebuah fungsi *loss* berdasarkan masukan ke suatu model dan akan menciptakan masukan baru untuk memaksimalkan *loss*-nya. Tipe serangan *white-box* sangat efektif untuk mempengaruhi hasi prediksi model, tetapi tidak praktis karena tipe serangannya hanya berguna untuk satu model spesifik saja.

Tipe serangan *black-box* dari *adversarial attack* adalah tipe serangan yang tidak didesain untuk model spesifik, sehingga berfungsi untuk semua model. Contoh tipe serangan *black-box* adalah dengan ZOO [2].

Pada konteks citra, terdapat penelitian yang merancang *adversarial attack* secara *black-box* dengan *steganography* oleh Din et al [3]. *Steganography* yang dilakukan adalah berbasis *Discrete Wavelet Transform* dan *Singular Value Decomposition* dan memiliki kinerja yang baik, yaitu 80% *fooling rate*.

Terdapat metode *steganography* yang sederhana untuk menyisipkan pesan, atau *watermark*, ke dalam sebuah gambar, yaitu *steganography* berbasis *least significant bit*. Makalah ini bertujuan untuk menganalisis metode ini sebagai jenis *adversarial attack* secara *black-box*. *Watermarking* yang dilakukan adalah terhadap objek spesifik pada gambar yang segmentasi dengan deteksi tepi.

II. LANDASAN TEORI

A. *Deteksi Tepi*

Tepi adalah bagian dari citra yang terdapat perubahan intensitas pixel yang cukup tinggi dalam jarak yang singkat [4]. Tepi memiliki arah yang bergantung pada perubahan intensitas. Contoh tepi pada citra terdapat pada gambar II.1.

Gambar II.1 Contoh tepi citra (Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/18-Pendeteksian-Tepi-Bagian1-2022.pdf)

Deteksi tepi dapat dilakukan dengan turunan terhadap nilai pixel. Terdapat operator turunan pertama, operator turunan kedua (laplace), operator laplacian of gaussian (LoG), operator sobel, operator prewitt, operator roberts, dan operator canny. Semua operator ini pada dasarnya memiliki fungsi yang sama, yaitu deteksi tepi, tetapi dengan metode yang berbeda.

Operator yang populer digunakan untuk deteksi tepi adalah operator canny, yang menghasilkan tepi dengan ketebalan 1 pixel seperti pada gambar II.2. Langkah-langkah deteksi tepi dengan operator canny adalah dengan menghaluskan citra dengan penapis gaussian, menghitung gradien dan arah gradien setiap pixel dengan operator sobel, prewitt, atau roberts, lalu melakukan *thresholding* dengan dua jenis *threshold*, yaitu T1 untuk tepi kuat dan T2 untuk tepi lemah. Setelah itu, akan dilakukan proses *edge linking* agar tepi-tepi lemah tergabung dengan tepi-tepi kuat.

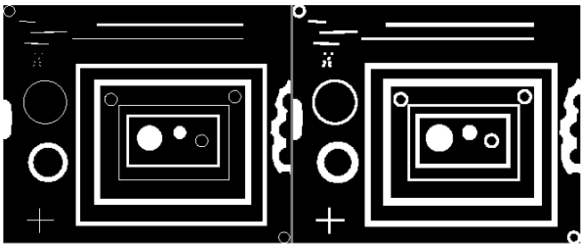
Gambar II.2 Contoh tepi citra (Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/19-Pendeteksian-Tepi-Bagian2-2023.pdf)

B. *Operasi Morfologi pada Citra Biner Untuk Segmentasi Objek*

Segmentasi objek pada citra dapat dilakukan dengan deteksi tepi. Namun, pada citra tepi terkadang terdapat tepi-tepi yang tidak terhubung dengan satu sama lain. Oleh karena itu, dapat dilakukan operasi morfologi pada citra biner untuk membantu menghubungkan tepi-tepi ini agar mempermudah segmentasi objek.

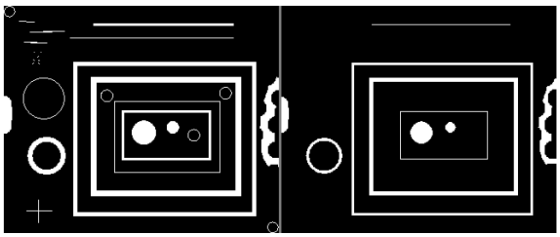
Operasi morfologi adalah operasi pada citra biner yang memanipulasi langsung pixel pada citra. Terdapat dua jenis operasi morfologi, yaitu dilasi dan erosi.

Dilasi adalah operasi yang bertujuan untuk memperlebar bentuk objek pada gambar. Contoh dilasi tertera pada gambar II.3.



Gambar II.3 Contoh operasi dilasi (Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/24-Citra-Biner-2023.pdf)

Erosi adalah operasi yang bertujuan untuk memperkecil bentuk objek pada gambar. Contoh erosi tertera pada gambar II.4



Gambar II.4 Contoh operasi erosi (Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/24-Citra-Biner-2023.pdf)

Kedua operasi ini dapat digabung untuk mencapai tujuan tertentu. Terdapat dua jenis operasi gabungan, yaitu operasi *closing* dan *opening*. Operasi *closing* adalah operasi dilasi yang diikuti dengan erosi yang bertujuan untuk mengisi lubang dan celah kecil, sedangkan operasi *opening* adalah operasi erosi yang diikuti dengan dilasi yang bertujuan untuk menghilangkan pixel di wilayah yang terlalu kecil untuk memuat elemen penstruktur.Operasi *closing* dapat membantu untuk menghubungkan tepi-tepi agar segmentasi objek yang dihasilkan lebih baik.

C. *Least Significant Bit (LSB) Steganography*

Steganography berbasiskan least significant bit (LSB) adalah teknik steganography sederhana yang dapat digunakan untuk watermarking. Teknik ini menyisipkan informasi rahasia di dalam sebuah citra dengan menyisipkan informasi tersebut pada least significant bit pada nilai pixel citra. Contoh citra beserta *watermark* tersembunyinya dapat dilihat pada gambar II.5.

Gambar II.5 Contoh citra dan *watermark* tersembunyi secara LSB

III. Implementasi

Terdapat beberapa implementasi yang harus dilakukan, yaitu pelatihan model *deep learning* untuk klasifikasi dan implementasi *watermark* secara LSB. Eksperimen akan dilakukan terhadap klasifikasi tipe kendaraan dengan jenis mobil, bus, truck, atau *motorcycle*.

A. *Fine-tune Model*

Model *deep learning* yang digunakan adalah model *fine-tune* dari ResNet18. Linear layer ditambahkan diatas model ResNet18 sebagai layer klasifikasi yang terdiri atas 4 neuron. Latihan akan dilakukan dalam dua tahap, yaitu pelatihan linear layer saja sebanyak 5 epoch, lalu pelatihan keseluruhan model sebanyak 2 epoch. Untuk hyperparameter yang digunakan dalam pelatihan model terdapat pada tabel III.1. Dataset yang

digunakan adalah “Vehicle Type Recognition” pada Kaggle, dengan jumlah data sebesar 400 dan terdiri atas 4 label. Pembagian training set dan validation set adalah 90% dan 10%.

Tabel III.1 Hyperparameter pelatihan model

Hyperparameter	Value
Batch size	32
Loss	Cross entropy loss
Learning rate	0.001
Output layer training epoch	5
All layer training epoch	2

Model dilatih di platform kaggle dengan GPU T4x2. Kinerja model setelah *fine-tune* tertera pada tabel III.2

Tabel III.2 Kinerja Model Setelah *Fine-tune*

Komponen Evaluasi	Nilai
Train loss	0.0516
Train accuracy	0.9861
Validation loss	0.0853
Validation accuracy	0.975

B. Implementasi LSB-based Steganography dan Segmentasi Objek

Berikut adalah implementasi *steganography* dengan LSB.

```
from PIL import Image
import numpy as np


def encode_lsb_image(
    image: Image, secret: Image, output_path:
str, lsb_bit_length: int = 1
):
    secret = secret.resize(image.size)
    output = Image.new("RGB", image.size)

    # rgb
    image = image.convert("RGB")
    secret = secret.convert("RGB")

    image_data = image.getdata()
    secret_data = secret.getdata()

    # encode lsb
    for i in range(len(image_data)):
        image_pixel = image_data[i]
        watermark_pixel = secret_data[i]

        r, g, b = watermark_pixel

        # modify lsb of image pixel
        bits_retained = (0xFF <<
lsb_bit_length) & 0xFF
        info_bits_length = 8 - lsb_bit_length
        new_r = (image_pixel[0] &
bits_retained) | (r >> info_bits_length)
        new_g = (image_pixel[1] &
bits_retained) | (g >> info_bits_length)
        new_b = (image_pixel[2] &
bits_retained) | (b >> info_bits_length)

        # new pixel
        new_pixel = (new_r, new_g, new_b)
        output.putpixel((i % image.width, i
// image.width), new_pixel)

    return output


def decode_lsb_image(encoded_image: Image,
output_path: str):
    # extract watermark from watermarked
image
    encoded_image_data =
encoded_image.getdata()
    secret_data = []
    for i in range(len(encoded_image_data)):
        encoded_pixel = encoded_image_data[i]
        watermark_pixel = (encoded_pixel[0] &
0x01) << 7
        secret_data.append(watermark_pixel)
    secret = Image.new("L",
encoded_image.size)
    secret.putdata(secret_data)
    return secret
```

Penyisipan informasi dengan metode LSB akan dilakukan terhadap semua objek yang disegmentasi dari sebuah citra. Implementasi dari segmentasi objek adalah sebagai berikut.

```
import cv2
import numpy as np
```

```
from PIL import Image
from stego_lsb import encode_lsb_image,
decode_lsb_image


def segment_and_encode(image, size, secret,
lsb_bit_length=1):
    # secret to pil
    secret = cv2.cvtColor(secret,
cv2.COLOR_BGR2RGB)
    secret = Image.fromarray(secret)
    watermarks = []
    # resize image
    image = cv2.resize(image, (size, size))
    gray_img = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)

    blurred = cv2.GaussianBlur(gray_img, (5, 5),
0)
    blurred = cv2.GaussianBlur(blurred, (5, 5),
0)

    edges = cv2.Canny(blurred, 50, 150)

    dilated = cv2.dilate(edges, None,
iterations=1)

    contours, _ = cv2.findContours(dilated,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    min_contour_area = 2000  # subject to change
    filtered_contours = [
        cnt for cnt in contours if
cv2.contourArea(cnt) > min_contour_area
    ]
    # Create a mask for each object and extract
it
    for i, cnt in enumerate(filtered_contours):
        x, y, w, h = cv2.boundingRect(cnt)

        # crop the image based on coordinates
        object_image = image[y : y + h, x : x +
w]
        object_image_pil =
cv2.cvtColor(object_image, cv2.COLOR_BGR2RGB)
        object_image_pil =
Image.fromarray(object_image)

        # encode image with secret
        object_image_encoded = encode_lsb_image(
            object_image_pil, secret,
f"watermarked_image{i}.png", lsb_bit_length
        )
        # decode
        watermarks.append(

np.asarray(decode_lsb_image(object_image_encoded,
f"watermarked_image{i}.png"))
        )
        object_image_encoded =
np.asarray(object_image_encoded)
        # in the original image, the extracted
object_image shall be replaced with
object_image_encoded
        # replace object_image with
object_image_encoded

        image[y : y + h, x : x + w] =
object_image_encoded

        # bounding box and text
    return image, watermarks[0]
```

C. Pengujian

Pengujian dilakukan terhadap 7 citra uji. Model akan memprediksi tipe kendaraan dari citra orisinil, lalu citra tersebut akan disisipkan citra kendaraan lain yang kemudian model akan memprediksi ulang tipe kendaraan pada citra tersebut. *Fooling rate* akan dihitung dengan perubahan prediksi pada suatu citra orisinil dan yang telah diubah dibagi dengan jumlah citra orisinil.

Untuk citra yang disisipkan adalah citra bus, yang diharapkan dapat mengubah hasil prediksi model terhadap jenis kendaraan pada suatu citra.

IV. HASIL DAN ANALISIS

Detail hasil eksperimen terdapat pada tabel IV.1.

Citra Orisinil	Citra *Watermarked* dengan Stego LSB	Citra tersembunyi	Apakah Hasil prediksi berubah antar citra?

			-
			-
			-
			-
			-
			-
			-

Dari eksperimen yang dilakukan, *fooling rate* yang didapatkan adalah 0%, yang artinya adalah *steganography* dengan LSB tidak dapat dijadikan metode untuk *adversarial attack* secara *black-box*. Hasil eksperimen yang dilakukan terhadap sampel kecil mungkin tidak representatif terhadap kinerja metode ini secara keseluruhan.

Usaha yang telah dilakukan untuk meningkatkan kinerja *fooling rate* adalah dengan menambah panjang informasi yang disisipkan dari satu *least significant bit* menjadi 4 bit, tetapi kinerja tidak berubah. Berbagai manipulasi proses dari *steganography* dengan LSB tidak dapat meningkatkan kinerja *adversarial attack.*

Terdapat beberapa penyebab kenapa metode ini tidak bekerja dengan semestinya walau bit yang dimanipulasi pada citra cukup banyak, yaitu citra yang disisipkan tidak terbaca oleh model karena pola manipulasi pixel citra terlalu sederhana dibandingkan dengan metode lain. Selain itu, terdapat jenis kendaraan pada citra orisinil dan citra *watermark* yang sama, yaitu bus. Kesalahan dalam eksperimen ini menyebabkan kinerja metode ini untuk *adversarial attack* secara *black-box* tidak terevaluasi secara utuh dan benar.

V. Kesimpulan

Steganography adalah sebuah metode untuk menyisipkan pesan ke dalam suatu data lain seperti citra yang tidak kelihatan oleh kasat mata. Terdapat banyak jenis *steganography* yang dapat dilakukan dan beberapa diantaranya dapat digunakan sebagai bentuk *adversarial attack* secara *black-box* [3]. Makalah ini bertujuan untuk mengevaluasi kinerja *steganography* yang sederhana, yaitu penyisipan pesan melalui LSB.

Eksperimen yang dilakukan adalah *adversarial attack* terhadap klasifikasi jenis kendaraan pada model *fine-tune* ResNet18. Eksperimen dilakukan terhadap 7 citra uji. Hasil yang didapatkan adalah metode ini tidak cocok untuk digunakan sebagai metode *adversarial attack* secara *black box*, karena memiliki hasil 0% *fooling rate*.

Terdapat beberapa kekurangan terhadap eksperimen ini yang dapat diperbaiki untuk penelitian selanjutnya. Selain itu, modifikasi terhadap *steganography* secara LSB dapat dilakukan agar meningkatkan kinerjanya untuk melakukan *adversarial attack* secara *black-box*.

Pranala Program

Pranala github : https://github.com/Haruray/citra-targeted-watermark

Pranala kaggle : https://www.kaggle.com/safiqfaray/pretrained-resnet-ujicoba-lsb-adversarial-attack

Ucapan Terima Kasih

Pertama-tama, puji syukur saya panjatkan kepada AllahSWT, yang senantiasa memberikan saya kesehatan untuk menuntut ilmu dan juga memberikan kesempatan untuk menyelesaikan makalah ini. Selain itu, saya berterima kasih kepada dosen Interpretasi dan Pengolahan Citra, Bapak Rinaldi Munir, yang senantiasa mengajar kami dengan sabar sehingga kami semua dapat mencapai titik ini, Saya juga ingin berterimakasih kepada keluarga dan teman-teman seperjuangan karena telah mendorong dan menyemangati satu sama lain.

Referensi

[1] Jaydip Sen and Subhasis Dasgupta, Adversarial Attacks on Image Classification Models – FGSM and Patch Attacks and their Impact. https://arxiv.org/ftp/arxiv/papers/2307/2307.02055.pdf

[2] Pin-yu Chen, et al. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. https://arxiv.org/pdf/1708.03999.pdf

[3] Salah Ud Din, et al. 2020. Steganographic universal adversarial perturbations.

[4] Munir, Rinaldi. “Pendeteksian Tepi (Bagian 1)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/18-Pendeteksian-Tepi-Bagian1-2022.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2023

Nama dan NIM