Trăistaru Vlad-Viorel
421-TAID

Tehnici de verificare a autenticității imaginilor – ELA (Error Level Analysis)

Cuprins

Figuri

Introducere

„Unul dintre aspectele problematice în criminalistica imaginilor digitale este explicarea problemelor tehnice care sunt greu de înțeles pentru oamenii obișnuiți. Chiar dacă dovezile falsificării sunt complet clare, documentele tehnice pure nu sunt de înțeles pentru publicul larg. Există probabilitatea ca textele de descriere complexe extinse să nu creeze acceptare. În cel mai rău caz, provoacă o atitudine negativă ofensivă.

Pentru munca de recunoaștere care implică publicul, vizualizarea este unul dintre cele mai eficiente mijloace, dar nu trebuie să fie cel mai bun. O metodă vizuală care și-a găsit drumul în criminalistica imaginilor digitale se numește analiza nivelului de eroare. Cu toate acestea, există riscul unei aplicări greșite. Utilizarea metodologiei ELA trebuie întotdeauna luată în considerare cu mare atenție.” [1]

Error Level Analysis

„Analiza nivelului de eroare (ELA) permite identificarea zonelor dintr-o imagine care se află la diferite niveluri de compresie. Cu imaginile JPEG, întreaga imagine ar trebui să fie aproximativ la același nivel. Dacă o secțiune a imaginii se află la un nivel de eroare semnificativ diferit, atunci probabil că indică o modificare digitală.

ELA evidențiază diferențele în rata de compresie JPEG. Regiunile cu colorare uniformă, cum ar fi un cer albastru solid sau un perete alb, vor avea probabil un rezultat ELA mai scăzut (culoare mai închisă) decât marginile cu contrast ridicat. Lucruri de căutat:

- Marginile similare ar trebui să aibă luminozitate similară în rezultatul ELA. Toate marginile cu contrast ridicat ar trebui să arate similare între ele, iar toate marginile cu contrast scăzut ar trebui să arate similare. Cu o fotografie originală, marginile cu contrast scăzut ar trebui să fie aproape la fel de luminoase ca marginile cu contrast ridicat.
- Texturi similare ar trebui să aibă o colorare similară sub ELA. Zonele cu mai multe detalii de suprafață, cum ar fi un prim plan al unei mingi de baschet, vor avea probabil un rezultat ELA mai mare decât o suprafață netedă.
- Indiferent de culoarea reală a suprafeței, toate suprafețele plane ar trebui să aibă aproximativ aceeași colorare sub ELA.

Se identifică diferitele margini cu contrast ridicat, margini cu contrast redus, suprafețe și texturi. Se compară acele zone cu rezultatele ELA. Dacă există diferențe semnificative, atunci se pot observa zonele suspecte care ar fi putut fi modificate digital.

Resalvarea unui JPEG elimină frecvențele înalte și are ca rezultat mai puține diferențe între marginile, texturile și suprafețele cu contrast ridicat. Un JPEG de calitate foarte scăzută va apărea foarte întunecat.” [2]

Descrierea aplicației

Implementarea ce urmează a fi discutată a fost realizată în limbajul de programare Python cu ajutorul bibliotecii Tkinter pentru interfață și scikit-image pentru procesarea imaginilor.

```
#importing libraries
import tkinter as tk
from tkinter import filedialog as fd

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import (
    FigureCanvasTkAgg,
    NavigationToolbar2Tk
)

import skimage
from skimage.util import compare_images
from skimage import img_as_ubyte

import numpy as np

import io
import imageio
import os.path
import copy
```

Figură 1: Bibliotecile importate pentru proiect

```
#creating main program window
root_window = tk.Tk()
root_window.title("ELA")

#variables for the image handles
original_image = None
original_file_name = None
original_image_blurred = None

#variables for altering the image
to_blur_var = tk.IntVar()
compress_quality_var = tk.IntVar()
```

Figură 2: Crearea ferestrei principale și a imaginilor de procesat

```
#setting up the interface
open_image_button = tk.Button(
    root_window,
    text = "Open Image",
    command = openImage
)

use_blurred_image_checkbox = tk.Checkbutton(
    root_window,
    text = "Use blurred image",
    command = lambda : processImages(),
    var = to_blur_var
)

compression_label = tk.Label(
    root_window,
    text = "Compression level: "
)

compression_slider = tk.Scale(
    root_window,
    from_ = 0,
    to = 100,
    length = 300,
    tickinterval = 10,
    orient = tk.HORIZONTAL,
    var = compress_quality_var
)

#default compression level
compression_slider.set(90)

compression_slider.bind("<ButtonRelease-1>", lambda event: processImages())
```

Figură 3: Definirea elementelor interfeței

```
#binding matplotlib figure to the canvas
figure = plt.figure()
figure_canvas = FigureCanvasTkAgg(figure, master = root_window)
NavigationToolbar2Tk(figure_canvas, root_window)
sp1 = figure.add_subplot(221)
sp2 = figure.add_subplot(222)
sp3 = figure.add_subplot(223)
sp4 = figure.add_subplot(224)

#pack the interface
open_image_button.pack()
use_blurred_image_checkbox.pack()
compression_label.pack()
compression_slider.pack()
figure_canvas.get_tk_widget().pack(side = tk.TOP, fill = tk.BOTH, expand = 1)

#start the main loop
root_window.mainloop()
```

Figură 4: Crearea elementelor vizuale pentru interfață

```
#function for loading the image
def openImage():
    file_types = [
        ('JPG', '*.jpg')
        ]

    file_path = fd.askopenfilename(
        title = 'Open Image',
        filetypes = file_types
        )

    if file_path:
        #load the image and its' info in global variables
        global original_image
        global original_file_name
        global original_image_blurred

        image = skimage.io.imread(file_path)
        original_image = copy.deepcopy(image)

        _, file_name = os.path.split(file_path)
        original_file_name = file_name

        #process the image with gaussian blur
        image_blurred = skimage.filters.gaussian(image,
                                                 sigma=2,
                                                 multichannel = True)
        original_image_blurred = copy.deepcopy(image_blurred)

        #once the images are loaded, process them
        processImages()
```

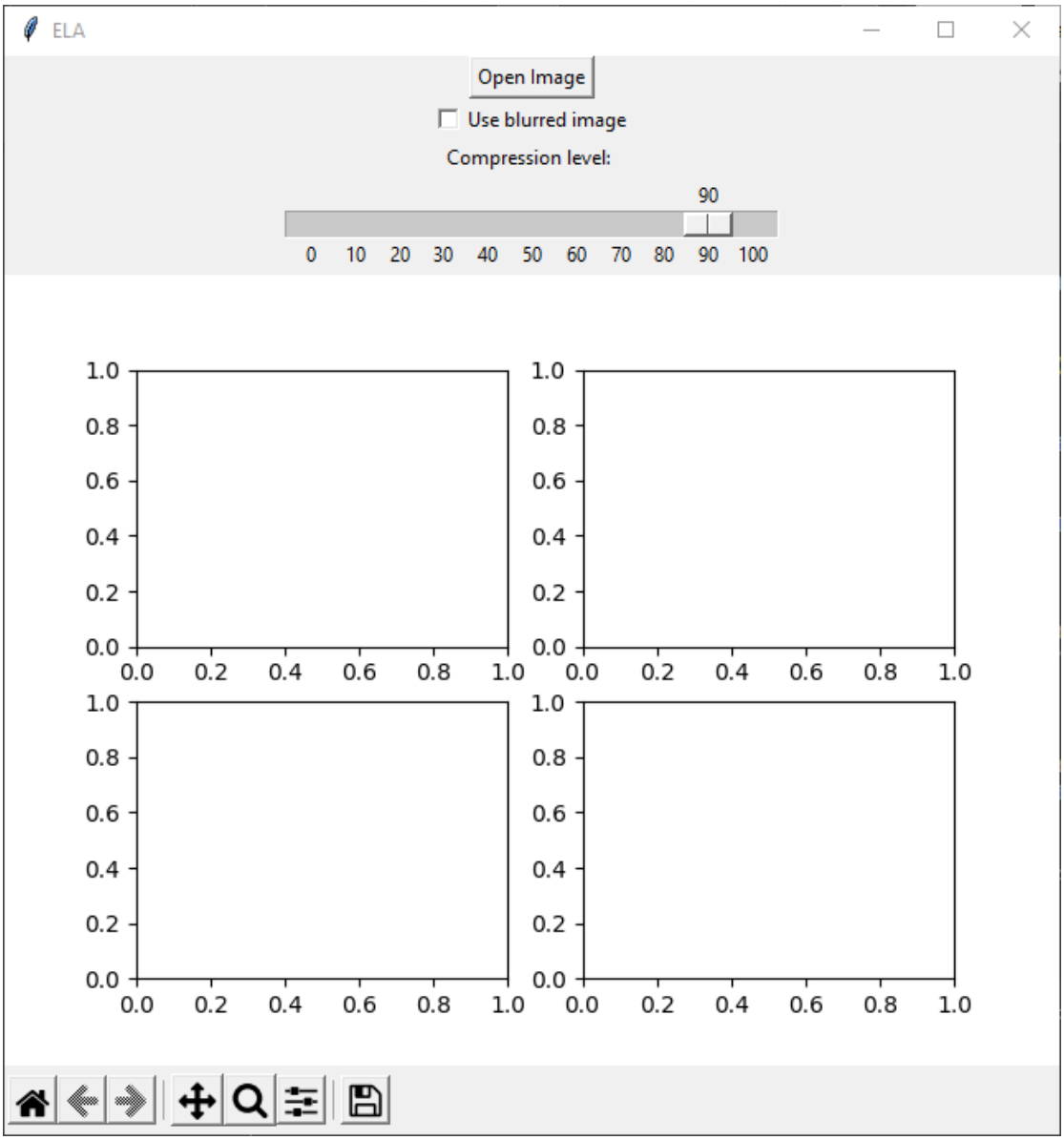
Figură 5: Funcție pentru deschiderea și aplicarea de blur pe o imagine

```python
#main work function
def processImages():
    #check if the image is loaded
    if original_image is not None:
        #creating memory stream for the compressed image
        buffer = io.BytesIO()

        #compress the image using JPEG in-memory
        #use the normal image or the blurred image depending on the checkbox
        imageio.imwrite(buffer,
                        original_image if to_blur_var.get() == 0
                        else original_image_blurred, format = 'jpg',
                        quality = compress_quality_var.get())
        image_compressed = imageio.imread(buffer.getbuffer(), format='jpg')

        #subtrack the original image from the compressed image
        image_difference = compare_images(original_image, image_compressed, 'diff')
        image_difference = img_as_ubyte(image_difference)

        #get the maximum difference value
        max_diff = np.amax(image_difference)

        #enhance the brightness of the difference image
        image_ELA = skimage.exposure.adjust_gamma(image_difference, max_diff / 255.0)

        #calculate the inverse of the ELA image
        inverted_image_ELA = skimage.util.invert(image_ELA)

        #setting the figures
        sp1.title.set_text(f'Original image: \n{original_file_name}')
        sp1.imshow(original_image)

        sp2.title.set_text(f'Blurred image: \n{original_file_name}')
        sp2.imshow(original_image_blurred)

        sp3.title.set_text(f'ELA on ' +
                           f'{"original" if to_blur_var.get() == 0 else "blurred"} ' +
                           f'image: \n{original_file_name}')
        sp3.imshow(image_ELA)

        sp4.title.set_text(f'ELA on ' +
                           f'{"original" if to_blur_var.get() == 0 else "blurred"} ' +
                           f'inverted image: \n{original_file_name}')
        sp4.imshow(inverted_image_ELA)

        figure.suptitle(f'Compression level used: {str(compress_quality_var.get())} %')

        #update the canvas
        figure_canvas.draw()
```

Figură 6: Funcție pentru procesarea imaginii folosind metoda ELA

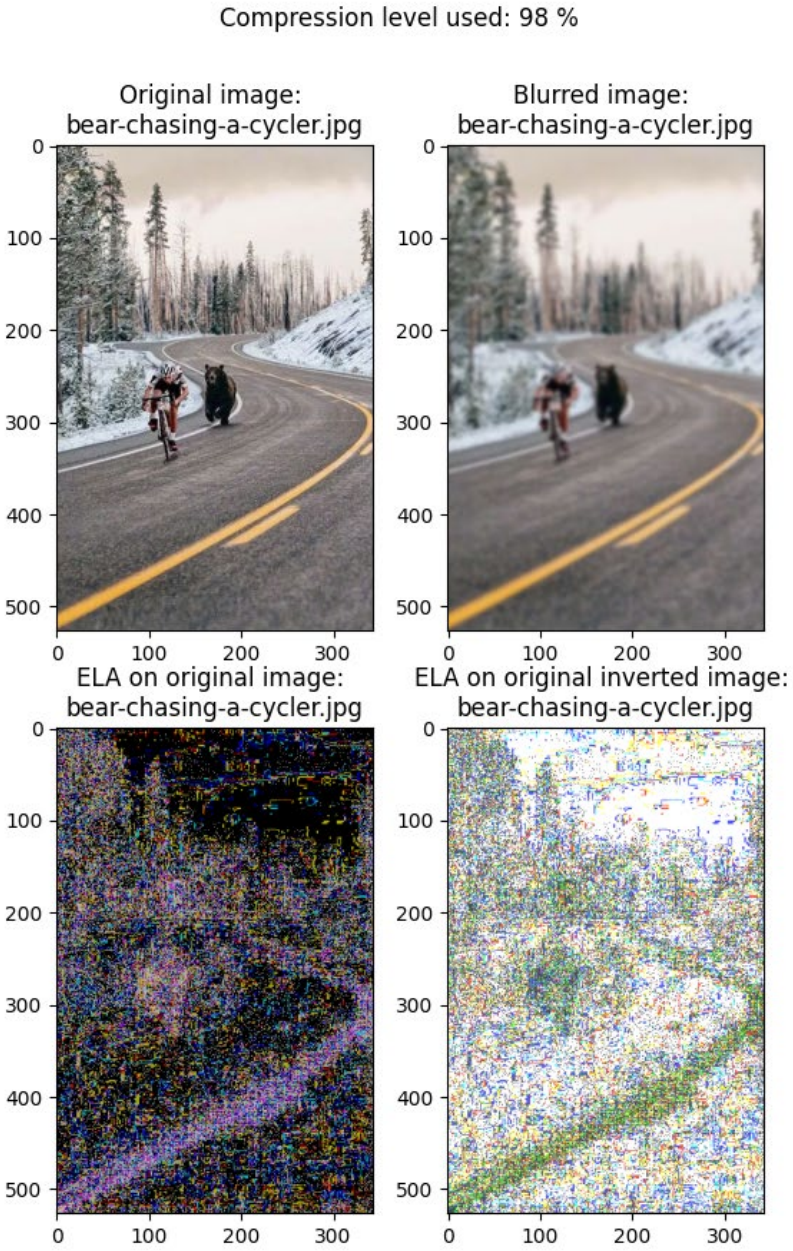
Algoritmul de procesare al imaginilor în această aplicație se ocupă cu următoarele:

- Se alege dacă să se proceseze imaginea originală sau cea blurată (se poate alege cea blurată atunci când este nevoie de informații în plus)
- Se salvează imaginea cu compresie JPEG la nivelul setat de utilizator
- Se face diferența între imaginea originală și cea resalvată cu JPEG
- Se caută pixelul cu cea mai valoare din imaginea de diferență
- Se crește luminozitatea imaginii de diferență cu o scală gamma egală cu valoarea pixelului discutat mai sus pentru a avea imaginea ELA
- Se inversează imaginea ELA pentru a avea o perspectivă în plus asupra analizării imaginii



Figură 7: Interfața aplicației pentru vizualizarea imaginii ELA

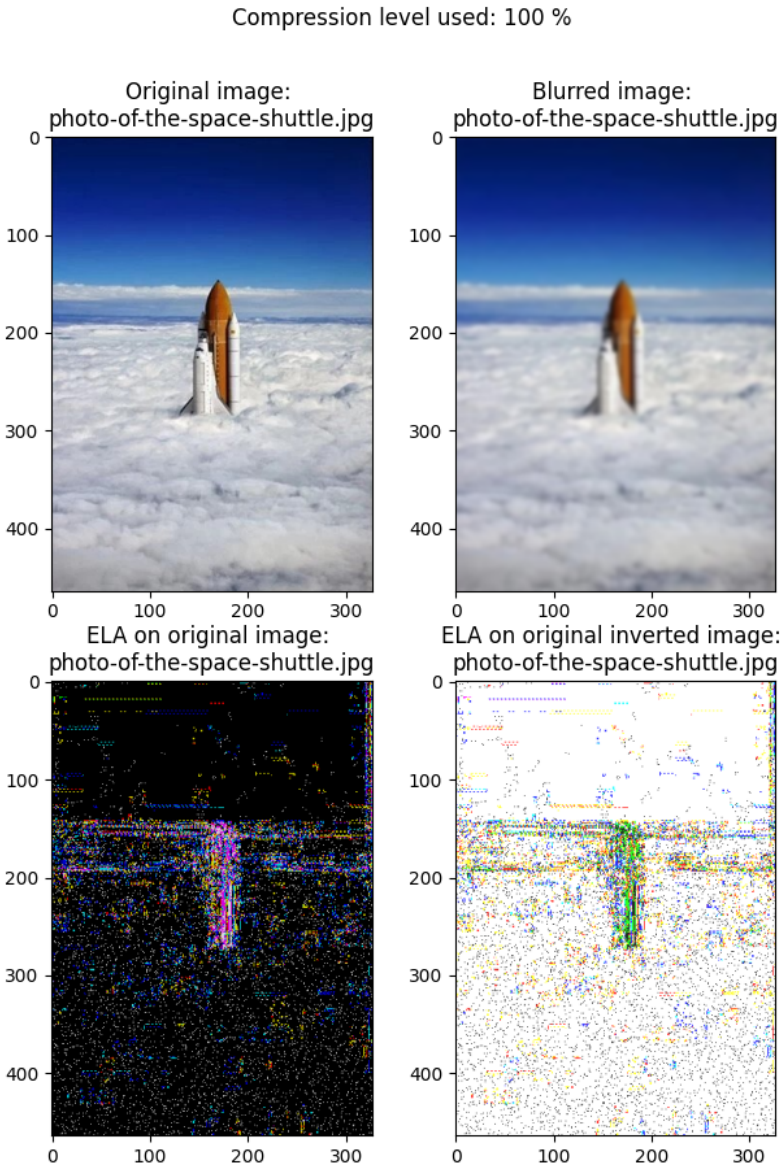
Rezultate



Figură 8: Analiza imaginii cu un ciclist urmărit de un urs

[3]

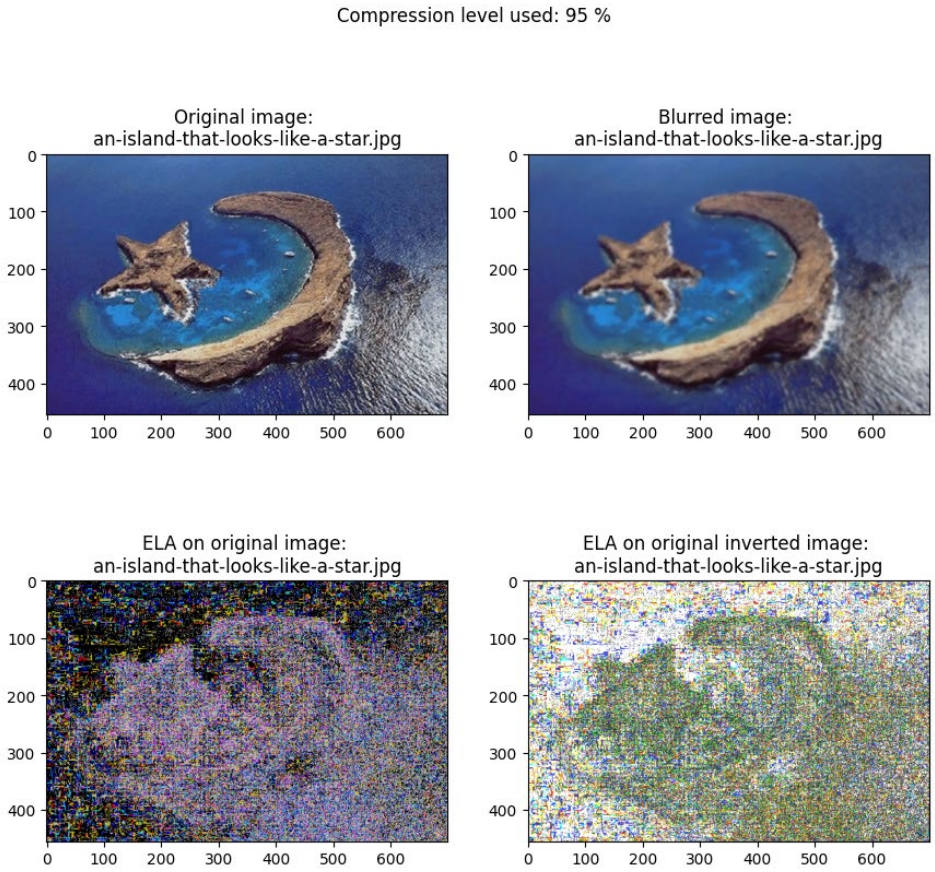
În această imagine, cu un nivel de calitate al recomprimării de 98%, algoritmul ELA detectează că atât ciclistul din poză cât și delimitatorul de pe drum ies în evidență. Acest lucru se poate observa foarte bine în imaginea inversată. Astfel, se poate ajunge la concluzia că ciclistul este adăugat digital, lăsând la o parte marginile delimitatorului din moment ce acestea au o culoare diferită față de restul drumului.



Figură 9: Analiza imaginii cu o rachetă trecând prin nori

[4]

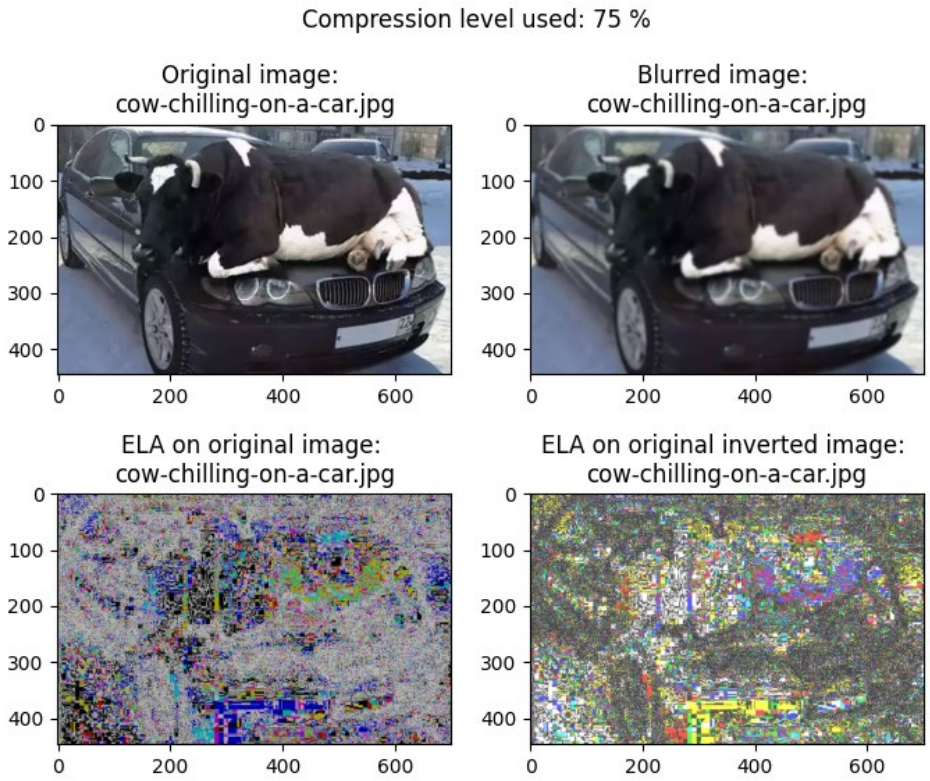
Racheta din această imagine a fost adăugată digital ca să pară că aceasta a fost capturată în momentul în care a trecut prin nori. La o privire simplă asupra imaginii procesate cu metoda ELA, care a fost recomprimată JPEG cu o calitate maximă, se poate observa că forma rachetei este evidențiată cu un nivel de eroare foarte mare ceea ce confirmă faptul că se lucrează cu o imagine doctorată.



Figură 10: Analiza imaginii cu două insule în formă de stea și lună

[5]

Imaginea de mai sus, recomprimată JPEG cu un nivel de calitate de 95%, dă de crezut că ambele insule de forme diferite nu ar fi reale sau că ar fi adăugate digital. Adevărul, însă, este că numai insula în formă de stea a fost adăugată peste imaginea originală. Se poate spune totuși că ELA a luat în considerare întreaga zonă a insulei în formă de stea față de cea în formă de lună.



Figură 11: Analiza imaginii cu o vacă care stă pe o mașină

[6]

În imaginea ELA, recomprimată JPEG cu un nivel de calitate de 75%, se poate observa destul de bine, mai ales în varianta cu imaginea inversată, conturul vacii care stă pe mașină. În acest caz, nivelul de eroare în jurul vacii nu este unul destul de mare dar poate arăta cu ușurință că imaginea originală nu este una reală.

Concluzie

Tehnica ELA poate fi folosită pentru a determina nivelul de adevăr a imaginilor JPEG, prin intermediul analizei nivelelor de compresie. Această metodă este foarte folositoare din moment ce suntem înconjurați zilnic de foarte multe imagini falsificate pe internet sau pe rețelele de socializare. Chiar și așa, în pozele care se prezintă rezoluție slabă sau contrast puternic, metoda ELA are niște limitări când vine vorba de procesarea acestor tipuri de imagini.

Bibliografie

[1] map-base, „Image Forensics | Error Level Analysis,” [Interactiv]. Available: https://forensics.map-base.info/report_2/index_en.shtml.

[2] FotoForensics, „FotoForensics - Tutorial: Error Level Analysis,” Hacker Factor, [Interactiv]. Available: https://fotoforensics.com/tutorial-ela.php.

[3] Rugile, „30 Fake Viral Photos People Believed Were Real - #15 Bear Chasing A Cycler,” Bored Panda, February 2019. [Interactiv]. Available: https://www.boredpanda.com/fake-news-photos-viral-photoshop/#:~:text=%2315%20Bear%20Chasing%20A%20Cycler.

[4] Rugile, „30 Fake Viral Photos People Believed Were Real - #18 Photo Of The Space Shuttle,” Bored Panda, February 2019. [Interactiv]. Available: https://www.boredpanda.com/fake-news-photos-viral-photoshop/#:~:text=%2318%20Photo%20Of%20The%20Space%20Shuttle.

[5] Rugile, „30 Fake Viral Photos People Believed Were Real - #22 An Island That Looks Like A Star,” Bored Panda, February 2019. [Interactiv]. Available: https://www.boredpanda.com/fake-news-photos-viral-photoshop/#:~:text=%2322%20An%20Island%20That%20Looks%20Like%20A%20Star.

[6] Rugile, „30 Fake Viral Photos People Believed Were Real - #23 Cow Chilling On A Car,” Bored Panda, February 2019. [Interactiv]. Available: https://www.boredpanda.com/fake-news-photos-viral-photoshop/#:~:text=%2323%20Cow%20Chilling%20On%20A%20Car.