Rezumat

Odata cu aparitia internetului, Securitatea Sistemelor Informationale a devenit o necesitate de baza a oricarei aplicatii, iar pe parcursul anilor, implementarea acesteia a devenit din ce in ce mai puternica. De la algoritmi de criptare a datelor utilizatorilor, pana la diverse masuri de protectie anti-spam, astfel incat interfata si experienta utilizatorilor sa fie cat mai simpla, iar libertatea programatorilor, cat mai restrictionata. Una dintre aceste masuri de protectie poarta numele de CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart).

In cadrul acestei lucrari, ce poarta denumirea ”Breaking CAPTCHA” ne vom axa pe tipul de CAPTCHA bazat pe text si vom observa cum putem cu ajutorul Vederii Artificiale sa construim un program care sparge CAPTCHA-urile respective. Se va explica in detaliu fluxul de lucru, obiectivul programului, algoritmii implementati, si pasii efectuati, astfel incat la final sa obtinem un soft cat mai curat si ideal. Programul va fi implementat cu ajutorul limbajului de programare Python, iar pe langa bibliotecile secundare vor fi utilizate 3 biblioteci principale dezvoltate special pentru lucrul cu Inteligenta Artificiala: NumPy, PyTorch, OpenCV. Seturile de date cu imagini CAPTCHA, descrise in lucrare, vor fi generate din librarii implementate in diverse limbaje, cum ar fi JavaScript, Python, Rust, etc.

Spre final vom incerca sa imitam un scenariu din viata de zi cu zi, si sa punem in practica bot-ul creat, utilizand o aplicatie secundara cu scopul de a observa comportamentul programului, remarca progresele, si de a formula concluzii asupra masurii de protectie, revenind cu o potentiala extensie a studiului efectuat, posibile imbunatatiri ale sistemului anti-CAPTCHA creat, dar si cu o sinteza precum si o serie de recomandari in care vor fi exemplificate tehnicile de protectie care opun o rezistenta mai mare, unor astfel de tipuri de atac.

Cuprins

Introducere

Odata cu dezvoltarea internetului si cresterea numarului de persoane care au acces la platformele web, a fost necesara introducerea unor sisteme care impiedica sau provoaca inconveniente in automatizarea proceselor, cum ar fi creare excesiva de conturi, colectare de date, reclama fortata, cu alte cuvinte, sisteme care implementeaza masuri de protectie impotriva botilor. Astfel a fost introdusa CAPTCHA.

CAPTCHA este prescurtare de la Completely Automated Public Turing Test to Tell Computers and Humans Apart.

Ce este Testul Turing? Testul turing original denumit si ”jocul de imitatie”, creat de Alan Turing, este un experiment implementat pentru a testa capacitatea masinii de a manifesta un comportament inteligent, echivalent, sau chiar si indisctintibil fata de cel al oamenilor. [1]

Termenul de CAPTCHA a fost inventat in anul 2003 de catre Luis von Ahn, Manuel Blum, Nicholas J.Hopper, si John Langford, iar primul si cel mai comun tip de CAPTCHA a fost inventat prima data in 1997 de catre 2 grupe de oameni de stiinta lucrand in paralel [2]. Mai jos se poate observa prima versiune a CAPTCHA-ului.

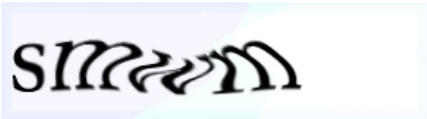

Figure 1: First Version of CAPTCHA

Dupa cum se poate observa, prima versiune de CAPTCHA este una bazata pe text, dar pe langa aceasta pe parcusul evolutiei au fost dezvoltate si alte tipuri cum ar fi, bazate pe imagini, sau audio. Initial s-a constatat ca CAPTCHA-urile puteau fi folosite in scopuri de securitate deoarece nu se stia, nu era descoperit, sau inventat un program care ar reusi sa rezolve problema

intampinata

O vulnerabilitate majora, care nu era considerata la momentul crearii CAPTCHA-urilor datorita nepopularitatii si progreselor minore, era si ramane un alt subdomeniu al informaticii, Inteligenta Artificiala, si anume Vederea Artificiala (Computer Vision), tinta caruia este sa imite comportamentul uman, sa detecteze, si sa identifice obiecte de interes intr-o imagine, secventa video, etc.

In aceasta lucrare ne vom axa pe CAPTCHA-uri de tip text, de altfel cum am si mentionat mai sus, cel mai comun tip de CAPTCHA, vrem sa evaluam starea curenta a tehnologiilor AI experimentand cu diverse dataseturi de CAPTCHA bazate pe text si sa determinam daca inca mai este valida ipoteza creata, si mai exact, daca CAPTCHA-urile de tip text manifesta o rezistenta puternica impotriva programelor automatizate, boti.

Aplicatia principala este un script de python care ruleaza pe fundal face http-request catre aplicatia secundara, care e o platforma web, si creeaza in continuu conturi pentru platforma respectiva, folosindu-se de AI-ul antrenat pentru rezolvarea CAPTCHA-ului corespunzator

Aplicatia secundara este o aplicatie web, vulnerabila, simpla, creata cu ajutorul framework-urilor Nuxt.js pentru FrontEnd si .Net Framework pentru BackEnd, care serveste ca scop, ilustrarea functionalitatii aplicatiei principale

Structura lucrarii este urmatoarea:

Capitolul I: Prezentarea generala a CAPTCHA-urilor. Vor fi prezentate librariile de Captcha utilizate pentru generarea seturilor de date, si folosite pe parcursul experimentelor, vor fi ilustrate tehnicile de protectie care sunt folosite, evolutia generala a lor, si un tabel cu situatia actuala.

Capitolul II: Prezentare generala a Invatarii Automate Adanci. Vor fi descrise bibliotecile folosite, va fi explicat fluxul de lucru in privinta realizarii aplicatiei.

Capitolul III: Experimentele realizate si rezultatele respective. Alegerea generatoarelor de imagini CAPTCHA tinta, urmata de justificarea modalitatii de spargere, proiectarea sistemului AI si compararea acestuia cu SOTA existent.

Capitolul IV: Vor fi structurate sub forma de sinteza concluziile la care am ajuns, problemele pe care le-am intampinat, niste propuneri cu ce se poate imbunatati, cat si recomandari personale in privinta alegerii masurilor de protectie.

In continuare vom vorbi doar despre CAPTCHA bazat pe text, si pentru simplicitate ii vom spune doar CAPTCHA

Capitolul 1 - Prezentare CAPTCHA

Istoric CAPTCHA

Primul CAPTCHA (Figura 1) a fost folosit in 1997 de catre motorul de cautare AltaVista si a impiedicat boti sa adauge URL-uri (Uniform Resource Locator) la motorul lor web [3]. Putem observa masurile de protectie intreprinse in cazul respectiv, caracterele sunt distorsionate, iar imaginea de fundal are cromatica unui gradient. In trecut doar aceste lucruri erau de ajuns pentru asigurarea protectiei, impotriva programelor rau intentionate, dar datorita progresului in stiinta, bariera a fost ocolita de catre un soft OCR (Optical character recognition) implementat in anii 2000 [3].

CAPTCHA-urile au devenit mai complexe, diverse nivele de zgomot si distorsii au fost adaugate la imagini, astfel incat OCR-urile populare au devenit ineficiente iar taskul pentru spargerea CAPTCHA-urilor a devenit costisitor pentru partea atacanta. Developerii de CAPTCHA in schimb trebuiau sa aiba grija cu nivelul de anti OCR aplicat imaginilor intrucat uneori se producea o inconvenienta mai mare pentru om decat pentru calculator sa le recunoasca [3].

O versiune mai complexa a primului CAPTCHA a fost implementata in Rusia, pentru platforma web Yandex.com.

Figure 2: Improved CAPTCHA

Remarcam un nivel de distorsie mai mare a caracterelor in imagine precum si diferentierea in culori a textului combinat cu imaginea de fundal ce corespunde unei valori aleatoare intr-un

spectru (0 - 255) de la negru la alb. Deasemenea imaginea de fundal este inconsistenta, tehnica care eroneaza si previne atacuri de spargere care utilizeaza anumite euristici precum diferenta neschimbata intre culorile imaginii, aceasta fiind cea mai mare problema a atacatorului - separarea caracterelor de imaginea de fundal. Astfel pentru a avansa in spargerea CAPTCHA-ului este necesara identificarea fonului si setarea fiecarui pixel al sau cu 0 (negru) iar fiecare pixel care apartine unui caracter cu 1 (alb) procedeu care se numeste binarizarea imaginii [3].

In 2005 isi face debutul primul CAPTCHA creat de Google, cu numele reCAPTCHA.

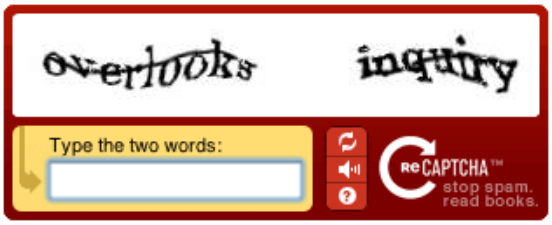

Figure 3: Google reCAPTCHA

Ca masuri de securitate, putem remarca multiple cuvinte, forma de valuri a acestora, precum si o linie peste caractere care impiedica segmentarea usoara a lor. reCAPTCHA-ul se deosebeste prin modul in care a fost programat, cuvintele fiind scanate dintr-o carte aleasa de adminului programului, si alese prin ajutorul Crowd-Funding-ului.

Programul reCAPTCHA alegea cuvintele in felul urmator, unul care e lizibil de catre OCR si celalalt nu. Apoi CAPTCHA-urile erau pasate unui grup de utilizatori, care pe rand rezolvau testele respective. Daca primul cuvant era introdus corect, programul presupunea ca si al doilea cuvant este corect, al doilea cuvant fiind pasat urmatorilor utilizatori ca fiind primul in testele noi. Programul mai apoi compara raspunsurile si avea destule dovezi pentru a recunoaste care cuvinte au fost introduse corect.

Astfel programul avea un scop dublu:

- Verificarea utilizatorului - Om/Robot?
- Verificarea cuvantului care nu putea fi citit de catre OCR si digitalizarea acestuia pentru cunoasterea umana. [4]

Ideea Google-ului era sa digitalizeze un numar cat mai mare de carti, ajungand la un numar de 40 milioane in 2019, potrivit wikipedia[5], si de a determina oamenii sa identifice caractere si cuvinte care nu puteau fi identificate de cei mai buni algoritmi de procesare a imaginilor existente la acel moment [4]

In acelasi timp multi programatori construiau diverse tipuri de CAPTCHA implementand multiple tehnici de protectie, care pot fi urmarite in imaginea de mai jos:

Figure 4: Tehnici de Protectie Captcha

Tehnici de Protectie

- Distorsiunea Caracterelor - Orice fel de modificare a caracterului cum ar fi rotirea sau deformarea acestora impiedica botii inspre utilizarea potrivii sabloanelor, cu niste imagini ale caracterelor prestabilite, pentru identificarea rapida a caracterelor distorsionate.

- Distorsiunea Imaginii de Fundal - Aplicarea diferitor manipulari asupra fonului, cum ar fi diverse culori, sau imagini inconsistente, acopera o mare vulnerabilitate in spargerea CAPTCHA-ului, problema de baza, si cea mai complicata fiind separarea fonului de caractere.

- Introducerea zgomotului in imagine - Zgomotul in imagine aplicat corect reuseste sa uneasca caracterele si imaginea de fundal astfel acestea raman lizibile pentru om dar foarte greu de distins de catre calculator, iar incercarea eliminarii acestuia, duce catre eliminarea unor componente foarte importante in CAPTCHA, cum ar fi portiuni de caractere sau chiar caractere intregi.

- Utilizarea diferitor fonturi - Stilul, marimea si originea fontului este o aplicatie importanta, fiindca adauga complexitate si provoaca inconveniente botilor in replicarea si identificarea caracterelor .

- Amplasarea aleatoare a caracterelor - Pozitia caracterelor in CAPTCHA, pot cauza erori in programul malitios si euristica utilizata de catre atacator.

- Concatenarea caracterelor - Atacatorul trebuie sa depuna efort inspre proiectarea unui program care separa caracterele concatenate, astfel incat imaginea este segmentata corect.

- Adaugarea elementelor secundare - Elemente secundare cum ar fi puncte, linii, figuri geometrice aleatoare in imagine sunt si ele considerate fiind zgomot, dar e mai putin aleator,

si reprezinta o piedica in segmentarea imaginii, detectia precum si recunoasterea caracterelor.

- Lungimea textului - O tehnica simpla dar puternica, se bazeaza pe faptul ca OCR-ul nu va reusi sa identifice corect toate caracterele din CAPTCHA, prin urmare daca lungimea textului este 15, iar OCR-ul a ghicit doar 14 caractere, testul va fi considerat incorect

- Combinatii intre tehnici - Evident combinarea a doua sau mai multe procedee descrise mai sus, sporeste complexitatea testului prin urmare si dificultatea intampinata de algoritmii automatizati este mai mare

In continuare atasez un tabel cu generatoarele de CAPTCHA care au fost analizate si in cadrul carora au fost executate experimentele din lucrarea respectiva. Librariile prezentate au fost construite de alti programatori, si se pot gasi open-source, aferent link-urilor atasate acestora.

table goes here

Capitolul 2 - Introducere Deep Learning

Istoric Deep Learning

Inteligenta Artificiala este un domeniu al informaticii ce si-a luat nastere in 1940 si avea ca scop principal simularea cu o precizie cat mai mare a comportamentului uman. Astfel erau creati si studiati algoritmi de creare a unor programe speciale incat la interactiunea lor cu omul, persoana sa nu isi dea seama daca interactioneaza cu o alta persoana sau cu un robot. Progresul urmeaza in 1958 cand a fost creata prima Retea Neurala Artificiala de catre psihologul Frank Rosenblatt. Reteaua era denumita Perceptron si a fost destinata să modeleze modul în care creierul uman procesa datele vizuale și învață să recunoască obiectele. Alți cercetători au folosit de atunci ANN-uri similare pentru a studia cogniția umană.[6]. Mai apoi in 1967 inspirata din ideea Retelelor Neurale Artificiale (ANN) apare primul algoritm functional, care era in sine un Perceptron cu mai multe straturi, alaturi de algoritmul Stochastic Gradient Descent, cu ajutorul caruia erau calculate ponderile in Reteaua Neurala. De atunci pana in prezent se realizeaza multe descoperiri si ajungem cu o arie de studiu intreaga pe care se focuseaza multi oameni de stiinta, numita Vederea Artificiala.

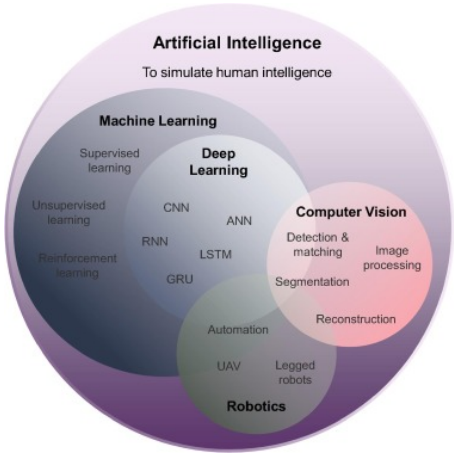

Figure 5: Ilustrare Deep Learning

Instrumente de lucru in Vederea Artificiala

Python este limbajul de programare de baza in rezolvarea problemelor de Inteligenta Artificiala, de altfel si algoritmii descrisi in aceasta lucrare sunt implementati in Python. Datorita lizibilitatii codului dar si contributiei a unui grup enorm de oameni, au fost create si mai apoi optimizate multe librarii ce ofera simplitate in lucrul cu concepte complexe.

- NumPy - este pachetul fundamental pentru calculul științific în Python. Este o bibliotecă Python care oferă obiecte si matrici multidimensionale, diverse obiecte derivate (cum ar fi matrice și array-uri mascate) și o varietate de operații rapide pe matrice, inclusiv matematice, logice, manipulare a formelor, sortare, selectare, I/O , transformate Fourier discrete, algebră liniară de bază, operații statistice de bază, simulare aleatorie și multe altele.[7]
  Deasemenea un avantaj in utilizarea bibliotecii NumPy este ca aceasta e implementata in limbajul de programare C/C++, astfel incat programele ce utilizeaza NumPy ruleaza mult mai rapid decat cele care utilizeaza functiile si sintaxa obisnuita din python.
  Mai jos sunt descrise niste functii care apartin NumPy si au fost utilizate in constructia aplicatiei:
  - np.array(*list*): transforma listele de orice tip in array-uri numpy - util cand avem nevoie sa manipulam imaginea in anumite feluri
  - np.transpose(*array*): permite modificarea/transpunerea dimensiunilor matricii
  - np.ones(*size*) / np.zeros(*size*): permite crearea matricilor care contin pe toate pozitiile fie valoarea 1, fie valoarea 0 - utile pentru eliminarea sau adaugarea elementelor in imaginea de baza.

- OpenCV - Biblioteca open-source care include cateva sute de algoritmi pentru Computer Vision. Pachetul este divizat in mai multe sectiuni, partea pe care se focuseaza in lucrarea respectiva fiind **Image Processing**. Ca si in NumPy algoritmii sunt si ei implementati si

optimizati in C/C++.

Mai jos sunt descrise niste functii care apartin OpenCV si au fost utilizate in constructia aplicatiei:

- cv.imshow(image) - functie permite afisarea pe ecran a imaginii, este utila in perioada debugging-ului.

- cv.cvtColor(image, ...*other parameters*) - functie care manipuleaza cu spatiul de culori al imaginii. Util cand e necesara eliminarea celor 3 dimensiuni RGB si se converteste imaginea in grayscale.

- cv.bitwise_not(image) - functie care inverseaza culorile imaginii, utila cand vrem sa separam obiectele de interes de fundal, si vrem sa fim consistenti in procesul dezvoltarii

- cv.erode(image) / cv.dilate(image) - functii extrem de importante utilizate in preprocesarea imaginii, permit eliminarea sau accentuarea detaliilor in imagini.

- cv.contours(image) - functie cu ajutorul carora se extrag contururile intr-o imagine, este utila in procesul segmentarii si descoperirii coordonatelor caracterelor

- cv.boundingRect(contours) / cv.rectangle(coordinates, image) - functii cu ajutorul carora se extrag si se deseneaza sub forma de dreptunghi caracterele care au fost segmentate.

- PyTorch - Biblioteca care implementeaza algoritmi de deep learning, permite construirea retelelor neurale intr-un mod cat mai usor si lizibil pentru programator, este optimizata pentru lucrul cu tensorii folosind GPU-uri si CPU-uri Mai jos atasez functionalitati utile care au fost folosite in constructia aplicatiei:

  - torch.nn.Conv2d() - Constructor pentru amplasarea in retea a unui strat convolutional

  - torch.nn.Linear() - Constructor pentru amplasarea in retea a unui strat linear

  - torch.nn.ReLU() - Rectified Linear Unit, Functie de activare pentru introducerea non-liniaritatii

Fluxul de lucru

Orice aplicatie bine implementata este construita in asa mod astfel incat sa restrictioneze, cat se poate de mult actiunile utilizatorului. Acesta insa, nu este scutit de automatizarea unor procese care la prima vedere ar parea complet legale, cum ar fi colectarea de date, sau chiar crearea a mai multor conturi pentru un singur website. Mai jos va fi descris procesul de gandire si implementare a automatizarii crearii conturilor pe o pagina web care contine sistemul de protectie de tip CAPTCHA.

Pentru inceput partea atacanta trebuie sa colecteze informatii despre cum arata pagina web, si in prealabil sa verifice daca CAPTCHA-urile prezente sunt asemanatoare intre ele sau apartin unor generatoare diferite. Odata ce avem destule informatii despre tinta respective putem incepe sa pregatim date pentru antrenarea sistemului de inteligenta artificiala. Daca generatorul de CAPTCHA este open-source acesta poate fi gasit usor, spre exemplu cu ajutorul functionalitatii ”search image with google” care este foarte util in cautarea imaginilor cu sabloane asemanatoare, sau pe diverse platforme, cum ar fi github. Daca insa generatorul nu este open-source, o varianta ar fi automatizarea procesului de extragere a imaginii din webpage, cum ar fi un script care reincarca pagina web, preia CAPTCHA-ul generat si il salveaza intr-un folder in local, apoi se adnoteaza manual setul de date, proces care ar putea sa dureze un pic mai mult. Cu setul de date pregatit, putem sa trecem la antrenarea sistemului de inteligenta artificiala. CAPTCHA-ul trebuie studiat, si depistate vulnerabilitati in masurile de protectie intreprinse, astfel incat spre final sa fie creata o functie care va putea separa caracterele (obiectele de interes) de imaginea de fundal. Functia respectiva o vom numi preprocess_image(). Odata ce avem separate caracterele de imaginea de fundal, incepem sa construim Reteaua Neurala care va fi antrenata. Din moment ce lucram cu imagini si trebuie sa depistam asemanari intre sabloane, evident ca vom utiliza straturile convolutionale, astfel vom avea o retea neurala convolutionala. In faza antrenarii, se va experimenta cu mai multi optimizatori, diversi parametri, si se varianta ca mai optima, se

va salva intr-un folder in local. Dupa ce avem modelul salvat, urmeaza crearea scriptului de atac care parseaza pagina web tinta si completeaza toate casetele de tip input, cu valori prestabilite, sau generate random, extrage CAPTCHA, o rezolva si apasa pe butonul submit pentru completarea actiunii, dupa care reia actiunea.

Deoarece nu este etic sa cream boti astfel incat sa provocam atacuri de tip spam impotriva oricarui website existent, aplicatia principala va fi testata pe aplicatia secundara deasemenea implementata pentru aceasta lucrare.

Google Colab

De asemenea in cadrul acestei lucrari vom lucra in environmentul Google Colab. Colab este un produs de la Google Research. Colab permite oricui să scrie și să execute cod python arbitrar prin browser și este deosebit de potrivit pentru machine learning, analiza datelor și educație. Din punct de vedere tehnic, Colab este un serviciu de Jupyter-notebook găzduit care nu necesită configurare pentru a fi utilizat, oferind în același timp acces gratuit la resursele de calcul, inclusiv GPU-urile, si este gratuit.

Capitolul 3 - Implementarea Aplicatiei, Experimente Realizate

Prezentare - Seturi de date CAPTCHA

Intrucat stim deja cum va arata aplicatia tinta, vom trece direct la extragerea si generarea dataset-urilor precum si prezentarea in detaliu a acestora.

SVG-Captcha

SVG-Captcha este o librarie CAPTCHA open-source, implementata pentru javascript, si dupa cum sugereaza numele, CAPTCHA-ul respectiv in momentul generarii este encodat in format SVG. Codul utilizat pentru generare este urmatorul:

```
var svgCaptcha = require('svg-captcha');
const fs = require('fs')

for (let i = 0;i<10000;i++){
    var captcha = svgCaptcha.create();
    fs.writeFile(`./../captchas/svg_captcha/images/
        ${captcha.text}.txt`, captcha.data,(err) =>{
        if (err) throw err
    })
}
```

Listing 1: SVG-CAPTCHA generation code

Se import modulele svg-captcha(pentru CAPTCHA) si fs(pentru manipularea cu fisierele), dupa care se itereaza de un numar prestabilit de ori, iar la fiecare iteratie, se apeleaza functia svg-Captcha.create(), dupa care salvam obiectul generat intr-un folder sub forma de fisier txt, iar numele fisierului va fi insusi eticheta / textul din imagine. Odata ce avem imaginile stocate intr-un folder, acestea trebuie transformate in format png sau jpg, pentru a lucra cu ele sub forma de matrici. Pentru a face asta folosim doua tool-uri implementate pentru python, software-ul

ImageMagick (gama de software gratuită, open-source, utilizată pentru editarea și manipularea imaginilor digitale. Poate fi folosit pentru a crea, edita, compune sau converti imagini bitmap și acceptă o gamă largă de formate de fișiere, inclusiv JPEG, PNG, GIF, TIFF și PDF. [8]) si libraria Wand (bibliotecă de funcții ctype care realizeaza conexiunea simplă intre ImageMagick pentru Python.).

```
from wand.image import Image
for idx in range(len(image_files)):
    fn = image_files[idx]
    ny = Image(filename = fn)
    fn = fn[:-4]
    ny_convert = ny.convert('png')
    ny_convert.save(filename = f'./{formated_svg_path}/{fn.rsplit("/", 1)
    [-1]}.png')
```

Listing 2: SVG-CAPTCHA generation code

Se itereaza prin folderul cu imagini, la fiecare iteratie se creaza un obiect de tip wand.Image cu fisierul propriuzis, se realizeaza conversia, dupa care se salveaza imaginea in formatul dorit in alt folder. Odata ce avem imaginile in formatul potrivit, dorim sa le mutam in environmentul cu care o sa lucram in continuare, google colab. Deoarece google colab este o platforma online, informatiile cu care lucram vor fi stocate in cloud. Pentru a nu consuma mult spatiu si pentru a fi cat mai concisi, vom transporta imaginile CAPTCHA sub forma de fisier .npy.

```
captcha_array = []
captcha_labels = []

for captcha_file in captcha_files:
    image = cv.imread(captcha_file)
    captcha_array.append(image)
    captcha_labels.append(captcha_file[captcha_file.rfind('/')+1:]
    .split('.')[0])
```

```
9
10 np.save("svg_formated_test_array",captcha_array)
11 np.save("svg_formated_test_labels",captcha_labels)
```

Listing 3: Conversie SVG in PNG

Figure 6: Exemple SVG CAPTCHA

Lepture-CAPTCHA

Lepture-CAPTCHA este libraria default CAPTCHA implementata pentru python, si are constructori pentru generare de CAPTCHA atat audio, cat si sub forma de imagini.

```
1 from captcha.image import ImageCaptcha
2 import random
3 import string
4
5 image = ImageCaptcha()
6 for i in range(1,11,1):
7     label = ''.join(random.choices(string.ascii_lowercase + '0123456789',
8                     k=4))
9     data = image.generate(label)
10    image.write(label, f'./captchas/lepture_captcha/images/{label}.png')
```

Listing 4: Lepture-CAPTCHA generation code

Label-ul / Textul din imagine este generat cu ajutorul librariei *random* implementate pentru python, si functia random.choices() primeste ca parametri o structura de tip lista din care se aleg caracterele, iar k este parametrul ce denota lungimea listei returnate, prin urmare in urma rularii scriptului respectiv vom obtine string-uri a cate 4 caractere ce contin litere mari, litere mici si

cifre. Acelasi principiu ca la SVG, doar ca de data asta primim imaginile in format .png direct, deci procesul de creare este mult mai usor, urmeaza sa stocam imaginile generate intr-un numpy array, si sa le transportam in memoria storage din cloud.

Figure 7: Exemple Lepture-CAPTCHA

Samir-CAPTCHA-rs

Samir's CAPTCHA-rs este o librarie de generare CAPTCHA implementata pentru limbajul de programare Rust

```
let captcha = CaptchaBuilder::new()
    .length(length)
    .width(200)
    .height(150)
    .dark_mode(mode)
    .complexity(complexity) // min: 1, max: 10
    .compression(compression) // min: 1, max: 99
    .build();

image_file.write_all(captcha.to_base64().as_bytes()).expect("unable to write
    data to image file");
```

Listing 5: Samir-CAPTCHA-rs generation code

Odata ce cream obiectul de tip CaptchaBuilder, aplicam functiile care construiesc imaginea finala:

length - Reprezinta lungimea CAPTCHA-ului generat

width / height - Dimensiunile Imaginii generate

dark_mode - In dependenta de valoarea booleana true sau false va genera imaginea cu fundal negru sau alb

complexity - Reprezinta nivelul de distorsiune aplicat CAPTCHA-ului, intr-o gama de nivele de la 1 la 10

compression - Reprezinta nivelul de comprimare a zgomotului din Imagine, intr-o gama de nivele de la 1 la 99

Imaginea Captcha este encodata in format base64 si este stocata in fisiere de tip txt. Urmatorul pas este sa transformam imaginea din base64 in format obisnuit - png sau jpg/jpeg. Pentru a converti imaginea o sa folosim urmatoarea functie:

```
def base64_to_image(base64_string):
    def pil_to_cv(image):
        if image.mode != 'RGB':
            image = image.convert('RGB')
        cv_image = cv.cvtColor(np.array(image), cv.COLOR_RGB2BGR)
        return cv_image
    if base64_string.startswith('data:image'):
        base64_string = base64_string.split(',')[1]
    image_data = base64.b64decode(base64_string)
    image = Image.open(BytesIO(image_data))
    return pil_to_cv(image)
```

Listing 6: Samir-CAPTCHA-rs generation code

Functia respectiva utilizeaza pachetul pillow, implementat pentru manipularea imaginilor, si ofera functionalitati asemanatoare cu pachetul OpenCV, si biblioteca base64, datorita careia putem decoda imaginea obtinuta in faza generarii. Astfel, se decodeaza imaginea, si se stocheaza intr-o variabila ca obiect de tip pillow.Image, dupa care o convertim in numpy array, pentru a putea efectua operatii din libraria OpenCV si returnam imaginea finala.

Preprocesarea imaginilor

Preprocesarea imaginii are o importanta enorma in perspectiva, si rezultatele finale, de aceea in mod ideal vrem sa separam perfect elementele de distorsiune din imagine, astfel incat sa ramanem doar cu zonele de interes computate corect. Dat fiind faptul ca avem mai multe generatoare de captcha, evident ca va trebuie reimplementata functia de preprocesare pentru toate dintre ele. In cadrul experimentelor au fost implementate preprocesarea imaginilor pentru cele 3 CAPTCHA, descrise mai sus. Cele mai importante functii utilizate

References

[1] https://en.wikipedia.org/wiki/Turing_test

[2] https://en.wikipedia.org/wiki/CAPTCHA

[3] https://habr.com/en/articles/670520/

[4] History and Evolution of CAPTCHA, Kishan Pandey

[5] https://en.wikipedia.org/wiki/Google_Books

[6] Artificial Neural Networks, Alexx Kay

[7] https://numpy.org/doc/stable/

[8] https://imagemagick.org/index.php