Fokozottan emberszerű interfész fejlesztése számítógépes rendszerekhez

Részletes Rendszerterv

Merész Patrik

Tartalomjegyzék

[Absztrakt 3](#_Toc134365667)

[Bevezetés 3](#_Toc134365668)

[Motiváció a projekt elkészítéséhez 4](#_Toc134365669)

[Hasonló projektek 5](#_Toc134365670)

[Bot detektálás videójátékokban 5](#_Toc134365671)

[Viselkedés analízis 6](#_Toc134365672)

[Heurisztikus módszerek 7](#_Toc134365673)

[Rendszerterv 8](#_Toc134365674)

[Követelmények 8](#_Toc134365675)

[Funkciós lista 8](#_Toc134365676)

[Felhasznált eszközök 9](#_Toc134365677)

[Rendszerterv diagramm 10](#_Toc134365678)

[Fejlesztés 11](#_Toc134365679)

[Fejlesztés menete 11](#_Toc134365680)

[Adatgyűjtés 11](#_Toc134365681)

[Megvalósítás 12](#_Toc134365682)

[Eszközök: Python, LabelImg, Google Colab, YOLOv5, CUDA, PyTorch 12](#_Toc134365683)

[Adatok Feldolgozása 12](#_Toc134365684)

[Tanítás 12](#_Toc134365685)

[Célzás implementálása 13](#_Toc134365686)

[Rejtve maradás 13](#_Toc134365687)

[Hivatkozások 15](#_Toc134365688)

[Idegen szavak jegyzéke 16](#_Toc134365689)

# Absztrakt

A szakdolgozat rövid bemutatása, tények, eredmények közlése, tőmondatos jelleg.

A szakdolgozat tartalmazza azon preventív rendszerek bemutatását, amelyek képesek megkülönböztetni gépi, illetve emberi interakciókat, különös tekintettel a [bot](#bot) (Lásd: Idegen szavak jegyzéke) detektálás területén. A dolgozatban kifejtem, hogy milyen következményekkel jár, ha az említett ellenőrzés elmarad, továbbá ezen rendszerek előnyeit és hátrányait fogom taglalni. A szakdolgozat második részében egy olyan szoftver tervezéséről és megvalósításáról lesz szó, amelyet a fentebb említett rendszerek még nem megbízhatóan képesek eldönteni a generált válaszok valódiságát, az elkészült szoftvert összehasonlítom más ilyen jellegű tanulmányokkal, és egy olyan valós tesztkörnyezetben is kifogom próbálni, mint például az online számítógépes játékok világa, amelyben a modern bot felismerő rendszereket fogom próbatétel elé állítani. Egy olyan szoftvert fogok fejleszteni, amely úgy mozgatja az egeret, hogy az a lehető legemberszerűbb legyen és amely jelentősen megkönnyíti a célzási műveleteket. A program emberi voltát egy GAN típusú neurális hálóval, ami az egérmozdulatok generálásáért felel, szeretném garantálni. A neurális hálóval a saját egérmozgatási adataimmal feltanítva fogok emberszerű egérmozdulatokat generálni. A dolgozat végén összegzem a fejlesztés menetét és értékelni fogom a projekt sikerességét.

# Bevezetés

Érdeklődés felkeltése(kedvcsináló), a téma ismertetése, cél, motiváció megfogalmazása

Az ember-számítógép kapcsolatok (HCI) kutatása egyik kulcsterület napjainkban. Számos olyan alkalmazás jelenik meg, ahol a kommunikációs partner egy számítógép, és az szolgáltat emberszerű válaszokat. Ilyenek például a chatbotok, hangalapú asszisztensek vagy tágabb értelemben vett virtuális ügynökök. Ezek a rendszerek rengeteget fejlődtek az elmúlt pár évben és a korlátaik terén is jelentős előre lépés történt, köszönhetően az AI és machine learning technológiáknak. Azonban érdemes megjegyezni, hogy az alább felsorolt rendszerek még mindig nem tökéletesek és alkalom adtán képesek olyan választ adni, amely messze távol van egy emberi kimenettől. Találkozhatunk olyan esetekkel is amikor ezek a botok rendkívül kártékonyak lehetnek, mivel teljesen megtévesztik a kommunikáló felet és fontos információk kiadását vagy az egyén véleményének befolyásolására használják.

Fontos megjegyezni, hogy a botokat rengeteg területen használják, mind hasznos, illetve kártékony használatra, ezért szükségszerű, hogy tisztában legyünk a lehetséges kockázatokkal és sebezhetőségekkel és hogy megtegyük a megfelelő óvintézkedéseket az okozott károk mérsékléséhez. Az elmúlt években jelentősen megnövekedett a kártékony botok száma ez különösen veszélyes lehet az online közösségi médiában, ahol rengeteg kamu profillal találkozhatunk. Ezek a fiókok úgy vannak tervezve, hogy félretájékoztassák a felhasználót, amelyekkel befolyásolni tudják a közvéleményt, politikai kampányokat vagy akár a pénzügyi piacokat. A másik szegmens, amely világméretűvé nőtte ki magát és amiről dolgozatom keretein belül fogok foglalkozni az nem más, mint a videó játékok világa. Egyre népszerűbb a játékosok körében, hogy olyan eszközökhöz nyúljanak, amellyel valamilyen előnyre tudnak szert tenni a többi játékossal szemben ezáltal jelentős anyagi károkat okozva a játékfejlesztő cégeknek.

A legfőbb cél a detektálás és a megelőzés, a nevesebb játékfejlesztő cégek rengeteg pénzt fektetnek abba, hogy olyan rendszereket építsenek ki, hogy azok garantálni tudják a tisztességes játékteret és hogy megőrizhessék játékuk integritását. Ezek a rendszerek úgy működnek, hogy felismerik és megakadályozzák a játékfájlok vagy folyamatok olyan jogosulatlan módosításait, amelyek tisztességtelen előnyhöz juttathatják a játékosokat. Emellett képesek a játékosok viselkedésének megfigyelésére és a csalásra utaló minták, például az aimbotting vagy a wall hacking felismerésére. Kezdetben heurisztikus módszereket alkalmaztak, de ahogy fejlődni kezdtek a különböző csalási metodikák és egyre több adat állt rendelkezésre, úgy egyre jobban megkövetelte az ipar a megfelelő gépi tanulási módszerek bevetését.

A szakdolgozatban lévő kutatásommal arra a kérdésre keresem a választ, hogy mit tudnak kezdeni ezek a rendszerek, akkor, ha a szemben álló program mesterséges intelligenciával támogatva próbálja kijátszani azokat. Röviden összefoglalva a projekt lényegét, két neurális háló fog összecsapni, amelyben ez egyik fél a detektálásért lesz felelős ez lesz a játék csalásellenes szoftvere, a másik oldalon pedig egy olyan neurális hálózat fog állni, amely generálja a megfelelő inputokat. Ilyesfajta módszert már most is alkalmaznak a mesterséges neurális hálózatok fejlesztésénél, amely Generatív párharc hálózat névre van keresztelve, röviden [GAN](#GAN) (Lásd: Idegen szavak jegyzéke).

## Motiváció a projekt elkészítéséhez

Az elsődleges motivációm a kihívás, olyan program létrehozásán gondolkoztam, amelyet akár több projektben is hasznosítani lehet, amellyel meglehet oldani a repetitív feladatokat és energiát lehet megtakarítani a használatával.

Egy olyan projektet akartam készíteni, amelyben a szoftver specializációban hallgatott tárgyaim közül a lehető legtöbb tudást tudom kamatoztatni. Legyen szó képfeldolgozási alapokról, neurális hálózatok létrehozásáról, használatáról, illetve párhuzamosításról és legutolsó sorban a videókártya által nyújtott gyorsításról.

Beszeretném mutatni, hogy AI technológiákat felhasználva képesek vagyunk olyan rendszereket gyártani, amelyekről nem lehet teljes mértékben eldönteni, hogy gép vagy ember gyártotta a megfelelő kimenetet.

A videójáték tesztkörnyezetre azért esett a választásom, ugyan is gyerekkorom óta érdekelt ez a terület és rengeteg időt töltöttem velük. Ezenfelül rendkívül kíváncsi vagyok, hogy milyen sikereket lehet benne elérni, ha az ember munkáját szoftveresen támogatjuk. Kompetitív személyiségűnek tartom magam, gyerekkoromban sok versenyen részt vettem, mind sportban, tanulmányi, illetve e-sport szinten is. Tanulmányaim során nem jut elég idő a gyakorlásra az utóbbi területre, ezért a játékokban való teljesítményem is drasztikusan romlik, ez különösképpen akkor a legbosszantóbb, amikor egy közeli ismerős vagy egykori csapattárs hívja fel a figyelmed arra, hogy mennyire nem vagy formában. Megoldást ebben a projektmunkában látom.

Szeretnék egy olyan programot készíteni, amely valósidőben képes detektálni ellenfeleket a képernyőn és a pozíciójukat meghatározva a másodpercek tört része alatt 0.1-0.2 másodpercen belül becélozni azokat. Ezt a gyakorlatban úgy szeretném megvalósítani, hogy számítógépes játékokban, kifejezetten FPS (belső nézetes lövöldözős) típusúakban fogom implementálni. A mágikus 0.1-0.2 másodperces szám, egy átlagos fizetett profi játékos reakció ideje az ellenfél felismerésétől az első célzott lövés elsütéséig. A projekt másodlagos célja az, hogy mind ezt úgy hajtsam végre, hogy a profilunkat ne érje kitiltás. A projektet két játékon tervezem megvalósítani az egyik a Counter Strike: Global Offensive (CSGO) a másik játék pedig a Valorant lesz. A CSGO-ra az egyszerűség és a kevésbé kifinomult csalásellenes szoftvere miatt esett a választás, a Valorantra pedig a híresen erős csalásellenes szoftvere miatt a választottam. Mind a két játék ingyenes, így a tesztelés közben felmerülő anyagi károk valószínűleg nem lesznek súlyosak.

## Hasonló projektek

Az emberi egérmozgás metrikáit már számos területen nagy figyelem övezi, mivel ez az adat jóval több információval tud szolgálni, mint elsőre gondolhatjuk. Ilyen például a felhasználók azonosítása egérmozdulatok alapján [1], vagy demográfia adatokat kinyerése [2], esetleg stressz szint felmérése munkahelyi környezetben [3].

Az emberszerű egérmozgatás generálásáról kiváló kiinduló pontot nyújt a Sapientia Erdélyi Magyar Tudományegyetem által készített tanulmány, melynek címe SapiAgent: A Bot Based on Deep Learning to Generate Human-Like Mouse Trajectories. [1]

A videójátékokban való éles tesztelésről hivatalos forrásból származó projektekről nem találtam információt. Az interneten különböző fórumokon kutatva fellelhető pár hasonló megoldás, de ezek elég gyengére sikerültek hatékonyság szempontjából. A legalapvetőbb probléma az, hogy könnyedén detektálhatóak gépi vezérlés szempontjából más részről a teljesítményük is elmarad a kívánttól.

# Bot detektálás videójátékokban

Irodalom kutatás jelentős része, előnyök, hátrányok bemutatása, összegzés

A játékokban használt botok olyan automatizált programok, amely egy adott játék bizonyos részeit vagy a játék teljes egészét játssza egy emberi játékos nevében. Ezek a programok sokkal több játék valutát vagy tárgyat tudnak keresni, mint az emberi felhasználók, mivel az előbbiek szünet nélkül tudnak játszani. Ezen játék robotok használata illegálisnak tekinthető, használatukért jellemzően felhasználó fiókját éri tiltás, ilyen esetben a felhasználó elveszíti a játék használati jogát és a játékben megszerzett javait. A játékban megszerzett javak egy része jelentős valódi anyagi értékkel bír. A legdrágább digitális tárgy idáig egy bolygó volt, aminek az értéke hatmillió dollárért kelt el a Entropia's universe című játékban [2]. A játékokban lévő botok az emberi felhasználókat is zavarhatják, mivel folyamatosan fogyasztják a játék erőforrásait és ez súlyos játékon belüli inflációt okoz, továbbá lerövidíthetik a játék életciklusát és rossz hírnevet generálhatnak a játékfejlesztő cégről. [3]

A botok detektálásához és felderítéséhez számos tanulmányt mutattak be a tudományos és az ipari életben. Ezek a módszerek három kategóriába sorolhatók: kliensoldali, hálózatoldali és szerveroldali. A legtöbb játékgyártó cég a játékrobotok elleni elsődleges intézkedésként a kliensoldali észlelési módszereket alkalmazza, amelyek a játékrobotok digitális lenyomatát elemzik. Az ügyféloldali felismerési módszerek a botprogram nevét, a folyamat információkat és a memória állapotát használják elsősorban. Ez a módszer hasonló a számítógépes vírusokat felismerő vírusirtó programokhoz. A kliensoldali észlelési módszereket a játékrobotok fejlesztői könnyen megkerülhetik, ráadásul a számítógép teljesítményét is rontják. Emiatt számos, ezen a megközelítésen alapuló ellenintézkedés, például a kereskedelmi botellenes programok jelenleg nem preferáltak, illetve nincsenek elterjedve. A hálózati oldali észlelési módszerek, például a hálózati forgalom figyelése vagy a hálózati protokoll változások elemzése hálózati túlterhelést és késleltetést okozhat a játékban, ami jelentősen rontja a játékélményt. A kliens- és hálózatoldali észlelési módszerek ezen korlátjainak kiküszöbölésére számos online játékszolgáltató szerveroldali észlelési módszereket alkalmaz. A szerveroldali észlelési módszerek a játékszerverek naplóadatait elemző adatbányászati technikákon alapulnak. A legtöbb játékszerver eseménynaplókat generál, amikor a felhasználók olyan műveleteket végeznek, mint a vadászat, a gyűjtögetés és a csevegés. Ezért ezek a játékon belüli naplók megkönnyítik az adatelemzést, mint a játékrobotok felderítésének lehetséges módszerét. Az online játékokkal foglalkozó vállalatok a szerveroldalon elemzik a felhasználói viselkedéseket vagy egyéb adatcsomagokat, majd ennek hatására az online játékszolgáltatók szelektíven blokkolhatják azokat a bot felhasználókat, akiket ki akarnak tiltani, anélkül, hogy további programokat telepítenének a kliensoldalon. Ezért a legtöbb online játékszolgáltató a szerveroldali észlelési módszereket részesíti előnyben. Ezen túlmenően egyes online játékokkal foglalkozó vállalatok olyan nagy adatelemző rendszer-megközelítéseket vezettek be általában gépi tanulási algoritmusok formájában, amelyek adatvezérelt profilalkotást és észlelést alkalmaznak. Az ilyen megközelítések több mint 600 TB-nyi, a játékszerverek által generált naplófájlokat képesek elemezni, és nem okoznak semmilyen mellékhatást, például teljesítménycsökkenést vagy konfliktust más programokkal. [3]

## Viselkedés analízis

Az egyik leggyakoribb módszer a viselkedés analízis, amely magába foglalja a játékosok viselkedésének elemzését a játékokban, hogy azonosítani lehessen a botok tevékenységére utaló mintákat. Ez a gyakorlatban úgy jelenik meg, hogy azt a tényt feltételezzük, hogy a botok egy előre meghatározott úton közlekednek és ismétlődően ugyan azt az interakciót hajtják végbe, ellenben az emberi felhasználóval, ami kiszámíthatatlanul és változatosabb módon viselkedik. Előfordulhat olyan eset is, amikor anomáliákat keresünk, ilyenkor olyan kirívó adatokat figyelünk, amelyek nem jellemzőek az emberi játékosokra, mint például kiugró erőforrás gyűjtögetési értékek vagy túlzott szint növekedés [4].

A közösségi tevékenységelemzés a közösségi hálózat jellemzőit használja fel az emberi és a játékrobotok megkülönböztetésére [5]. Ebben a módszerben azt a lehetőséget aknázzák ki, hogy a botok jellemzően nem végeznek semmiféle interakciót a többi játékossal, ezáltal rendkívül hatékonyan szétválasztható a két csoport. Az interakciók az alábbiak lehetnek, chatelés, csoport hívások vagy esetleg bármilyen tárgy cseréje, illetve üzletelése.

A fentebb említett módszerekre jellemző, hogy rendkívül nagy mennyiségben kell az adatoknak rendelkezésre állniuk, hogy ezek a technikák hatékonyan működjenek. A legtöbb esetben klasszterező algoritmusokat vagy statisztikai eszközöket használnak. Egy másik kihívás, amellyel az adat elemzési módszerek szembenéznek az a számításigényesség, különösen a nagy mennyiségű adatok kezelésénél. Ezért fontos, hogy a játékfejlesztők optimalizálják az adatelemzési technikákat, hogy azok ésszerű idő alatt elvégezhetők legyenek és hogy a megfelelő intézkedés a játékos irányába minél hamarabban megtörténjen.

## Heurisztikus módszerek

A heurisztikus elemzés a botok észlelésének olyan módszere, amely a gyanús viselkedésminták vagy műveletek azonosítását foglalja magában, amelyeket valószínűleg botok hajtanak végre. Ez a technika előre meghatározott szabályokon vagy heurisztikákon alapul, amelyeket a bottevékenységre utaló viselkedések azonosítására használnak.

Például az egyik heurisztika, amelyet a játékban lévő botok felismerésére lehet használni, a játékosok mozgásának sebessége. Ha egy játékos természetellenesen gyors sebességgel mozog, ami nincs összhangban a játék fizikájával, az jelezheti, hogy botot használ. Hasonlóképpen, ha egy játékos ugyanazt a műveletet ismétlődően, tökéletes időzítéssel hajtja végre, az is jelezheti, hogy botot használ. [6]

A heurisztikus elemzés hatékony lehet bizonyos típusú botok azonosításában, például azokéban, amelyek bizonyos típusú viselkedési mintákra vagy a játékban található kihasználható résekre támaszkodnak. Ugyanakkor hamis pozitív eredményeket is produkálhat, mivel egyes legitim játékosok hasonló viselkedési mintákat mutathatnak. Ezért fontos, hogy a játékfejlesztők gondosan meghatározzák heurisztikáikat, és alaposan teszteljék azokat a hamis pozitív eredmények minimalizálása érdekében.

A heurisztikus elemzés egyik előnye, hogy valós időben végezhető, ami lehetővé teszi a játékfejlesztők számára, hogy gyorsan azonosítsák és eltávolítsák a botokat a játékból. A heurisztikus elemzés önmagában azonban nem biztos, hogy elegendő a botok minden típusának felismeréséhez, mivel a fejlettebb botokat úgy tervezték, hogy elkerüljék a heurisztikus felismerést.

Összefoglalva, a heurisztikus elemzés hasznos technika a videojátékokban használt botok észlelésére, mivel előre meghatározott szabályok vagy heurisztikák alapján gyorsan azonosítani tud bizonyos típusú botokat. Ugyanakkor hamis pozitív eredményeket is produkálhat, és nem biztos, hogy hatékony a fejlettebb botok felderítésére. Ezért a játékfejlesztőknek a heurisztikus elemzést más botfelismerési technikákkal együtt kell használniuk annak biztosítására, hogy a botokat felismerjék és eltávolítsák a játékból.

# Rendszerterv

## Követelmények

A neurális hálózatnak emberszerű egérmozdulattal kell visszatérnie, ami azt jelenti, hogy kerülnie kell az egyenes vonalakat és az irányváltoztatásoknál pedig nem kanyarodhat szögletesen, továbbá a rendszer az egérmozdulat generálását 0.4 másodpercen belül képes legyen elvégezni. Az emberszerű egérmozgás garantálásra készüljön egy másik neurális háló, ami az egérmozdulat lépéseinek az időzítését tanulja meg.

A tesztkörnyezetben az elkészített szoftvernek valósidőben nagy pontossággal fel kell tudnia ismerni a betanított objektumokat a képernyőn. Rendelkeznie kell egy grafikus interfésszel, amellyel a felhasználó személyre szabhatja a program működését. A rendszernek megfelelően le kell kezelnie azt, ha egyszerre több ellenfél is tartózkodik a képernyőn. Az egérmozgató funkciónak emberszerűnek kell tűnnie. A használatáért a játékos fiókját nem érheti kitiltás, kivételt képez ez alól, ha a programot olyan paraméterekkel indítják el, hogy a rendszer a maximális teljesítményét kihasználva működtetik. A rendszer kizárólag Windows környezetben működjön. Képesnek kell lennie olyan eszközön is működni, ami nem rendelkezik grafikus gyorsítóval.

## Funkciós lista

A szoftverben szabadon paraméterezhetőek az alábbi funkciók: játék kiválasztása, célpont területének kiválasztása, a képernyőn vizsgált terület meghatározása, az egérmozgatási funkció távolsága és sebességének állítása. Ezeket a beállítási lehetőségeket egy grafikus interfészben lehet állítani.

* A játékot kiválasztva a program betölti az adott játékhoz tartozó képfelismerési modellt
* Célpont terület, a felhasználónak lehetősége van kiválasztani, hogy a célpont testének melyik részét kívánja becélozni.
* Képernyőn vizsgált terület, a képernyő közepén elhelyezett rész ablak, amelyben a képfelismerő algoritmus vizsgálja a képet.
* Egérmozgatási funkció távolsága, megjelöli azt az euklideszi távolságot az egér kurzor és a célpont között, amelyen belül az egérmozgató automatika működésbe lép.
* Az egérmozgatás sebessége egy szorzó, amely megváltoztatja az egér maximális lépési távolságát a szorzó szorosával. A programban alapértelmezetten ez 10 pixel.

## Felhasznált eszközök

* Programozási nyelv: Python 3.9
* Főbb könyvtárak: Win32Api, Mss, Pandas, PyTorch, Numpy, Matplotlib, Tkinter
* Virtuális környezet: Google Colab
* Képfelismerés: Yolov5
* Szoftver: LabelImg
* Videókártya gyorsítás: CUDA, cuDNN, TensorRT
* Verziókövetés: Git
* Számítógép fontosabb hardverei:
  + Intel Core i5-11400f (6 mag, 12 szál, max 4.4Ghz)
  + RAM: ddr4 3600Mhz 16GB
  + Nvidia RTX 3060 12GB

A Python programozási nyelvre azért esett a választás mert rengeteg hasznos keretrendszerrel van ellátva, továbbá rendkívül sok dokumentáció lelhető felhozzá. Gyorsan lehet vele dolgozni, nem kell szenvedni az alap dolgok implementálásával és könnyen olvasható a kódja. A Pythont előszeretettel használják machine learning algoritmusok írására és automatizációra, amire pont nekem is szükségem van. Egyetlen egy hátránnyal szembesültem a Python használata alatt az pedig a sebesség, de ez egy kis párhuzamosítással és a beépített adatszerkezetek helyett Numpy könyvtár használatával jelentős sebesség növekedést tudtam elérni és így már a teljesítmény jelentősen javult.

A neurális hálózatokat PyTorch [7] keretrendszerrel szeretném készíteni, mert használata egyszerű és könnyű benne szerkeszteni a neurális hálózat architektúráját. Nyílt forráskódú ezért a használatáért nem kell fizetni. Rendelkezik videókártya gyorsítással amire a tanítás során és a modell kiértékelésénél vettem nagy hasznát. A Pytorch-nak nagy és aktív felhasználói közössége van, ami az én esetemben azt jelentette, hogy könnyen találtam megfelelő forrásokat, illetve kaptam segítséget, ha elakadtam, további nagy előnye, hogy rendelkezik debug funkcióval, ami megkönnyítette a hiba keresést és azok megtalálását.

Az objektumok detektáláshoz a Pythonhoz megírt Yolov5 [8] algoritmust használom, amit azért választottam ki mert rendkívül magas pontosságot és teljesítményt ígért. A Yolov5 egy nyílt forráskódú mélytanuláson alapuló megközelítés, ami a képeken vagy videókon lévő objektumok felismerését teszi lehetővé, amely különböző számítógépes látási feladatokhoz, például önvezető autókhoz, robotikához és biztonsági rendszerekhez használnak. Az egyik legnagyobb előnye, hogy saját adathalmazra is betanítható és hogy képes lekezelni a TensorRT modelleket is.

A Mathplotlib nevű könyvtárral az adatvizualizációs problémákat oldom meg. Ezzel a könyvtárral fogom készíteni a tanulási folyamatokat bemutató ábrákat, további az emberi egérmozdulatokat is ezzel a könyvtárral fogom grafikusan megjeleníteni.

A Pythonba közvetlenül beépített GUI keretrendszerét a Tkintert fogom használni, hogy legyen egy kézzel fogható interfész, amelyen keresztül belehet állítani a program paramétereit és amelyből egyszerűen kezelhető a program indítása vagy leállítása.

A Win32Api könyvtár azért felel, hogy a szoftverben kiadott egérmozgatási parancsok globálisnak hassanak, magyarán szólva, hogy a játékban is megjelenjenek. Sok könyvtárral ellentétben, mint például a PyautoGui,PyMouse melynek egérmozgatási parancsai DirectX környezetben már nem működnek. Egyetlen egy gond van vele, hogy már a Valorant nevű játék képes kiszűrni. A másik hátránya, hogy ez a könyvtár csak windows számítógépen működik és így a programom nem tud működni más operációs rendszeren

Az Mss könyvtár egy nagyon gyors platformfüggetlen képfelvevő szoftver [9], megfelelő hardverrel képes akár a 60fps fölötti gyorsasággal rögzíteni a képernyő tartalmát.

## Rendszerterv diagramm a tesztkörnyezetben való futtatásról

# Fejlesztés

## Képfelismerés

### Adatgyűjtés és tanítás

A projekt megvalósításának az első és legfontosabb lépése az adatgyűjtés, első teendőm az volt, hogy képeket gyűjtsek a felismerni kívánt objektumokról ezt a problémát úgy oldottam meg, hogy írtam egy scriptet, ami képernyő felvételeket készít a monitor tartalmáról. Kezdetben ezt úgy próbáltam kivitelezni, hogy a program bizonyos időközönként például 5 másodpercenként képet készít. De ezzel a megközelítéssel az volt a gond, hogy nagyon sok irreleváns kép keletkezett. Végül annál a megvalósításnál maradtam, hogy egy gomb lenyomására készítse a felvételeket. Ezt a kis programot játék közben futattam és a bal egérgombbal aktiváltam a képek készítését. Amikor meguntam a játékot, ugyan ezt a tevékenységet videókon alkalmazva végeztem, mindig, amikor egy ellenfél tűnt fel a képernyőn kattintottam. Két hónap kitartó munkája alatt sikerült 2137 képet gyűjtenem és megfelelően felcímkézni azokat. A képek címkézést LabelImg nevű szoftverrel végeztem. Ez a folyamat az adatgyűjtés legidőigényésebb része, de szerencsére csak egyszer kell megcsinálni, illetve annyiszor ahány játékra megszeretnénk írni programot. A képek megcímkézésére azért van szükség, hogy a Yolov5 modellt [supervised](#supervised) vagy másnéven felügyelt módon tanítsuk.

A Yolov5 modellt a dokumentációjában [8] leírt instrukciókat követve sikerült feltanítanom Google Colabon a saját adathalmazommal, mind össze 3 órán át tarott a tanítás egy P100-as videókártyát használva, mivel nekem a teljesítmény elsődleges szempont volt ezért a legkisebb modellt választottam, ami az „S” névre hallgat. Ez a modell a gyártó szerint egy V100-as videókártyával képes az átadott 1280 pixel széles képen 4.3 milliszekundum körül felismerni a megfelelő objektumot és visszaadni körbe író téglalap pozícióját. Természetesen ez a sebesség nagyban függ attól, hogy az átadott kép mekkora. A legjobb hatékonyságot 320x320-as képeken sikerült elérnem, ennél kisebb kép méretbe már nem férnek bele a felismerni kívánt objektumok. A nagyobb képméretnél a felismerés ideje nőtt drasztikusan. Egy 1920x1080-as képen 50 milliszekundum körüli értékeket kaptam. A megkapott adatok alapján könnyen kitudjuk számolni a felismert célpont középpontját, amire a célzást fogom implementálni.

## Emberszerű egérmozgás

Szakdolgozatomban az emberszerű egérmozgást több fajta megközelítéssel szeretném megvizsgálni és összehasonlítani, először is a Sapientia Erdélyi Magyar Tudományegyetem által készített tanulmány alapján, melynek címe SapiAgent: A Bot Based on Deep Learning to Generate Human-Like Mouse Trajectories [1] a későbbiekben csak SapiAgent néven fogok hivatkozni rá. Ebben a tanulmányban arra a kérdésre keresték a választ, hogy hogyan lehet emberszerű egérmozdulatsort generálni, illetve kellő pontossággal felismerni azt, kiszűrve a botokat csak az egérmozdulat sorra alapozva. A tanulmány arra is kitért, hogy az ő általuk készített adathalmaz első sorban azt a célt szolgálja, hogy a jelenlegi, illetve későbbi bot detektáló rendszerek továbbfejlesztését, javítását biztosítsa.

A SapiAgent 128 felhasználó egérmozgatási szokásaiból lett betanítva, fontos kiemelni, hogy ez a tanulmány három és öt perces időintervallumban vizsgálta a résztvevőket. A felhasználók 18 és 53 év közötti korosztályból származnak és a felmérésben résztvettek jobb és balkezes emberek is. A méréseket megközelítőleg 60Hz-es frissítéssel monitorozták és az egyes lépéseket egy fix méretű tömbbe mentették, amely 128 elemet tartalmaz, ha egy mozdulatsor ennél több lépésből állt akkor egyszerűen nem mentették el a felhasználó mozdulatait és ha a résztvevő fél kevesebb lépésből megoldotta a megadott egérműveletet akkor a létrejött tömböt addig növelték 0-s tagokkal ameddig 128 méretű lett. A mérés diverzitását tovább fokozza, hogy több fajta egérmozdulat sort vettek számításba, azon felül, hogy vizsgálták az elmozdulást, külön eseményeket is figyelembe vettek, mint például drag and drop, jobb és bal klikk lenyomásokat. Az összegyűjtött adatokból több neurális hálózatot tesztelve próbálták a legeredményesebb adathalmazt generálni. A legjobb eredményt egy sajátos módszerrel betanított konvolúciós autoenkóderrel érték el.

Összegezve a SapiAgent tanulmányt, a SapiAgent eltér a hagyományos tanítási megközelítésektől és komoly sikereket értek el a felismerésekkel összefüggésben. A tanító adathalmaz 120 alanytól származik és a botok észlelését anomália detektálásként fogalmazták meg. A SapiAgent által generált szintetikus egérmozdulatok sokkal valósághűbbek ez eddig ismert Bézier-görbékhez és más hagyományosan feltanított autoenkóderekhez képest.

A másik megközelítést saját magam szeretném megvalósítani, illetve implementálni GAN neurális hálózatokkal. Az adatokat a sajátegérmozgatási szokásaimból szeretném felépíteni és bizonyos szempontból egy jóval egyszerűbb, de másrészről sokkal precízebb környezetet szeretnék biztosítani a méréshez, mint az előbb említett SapiAgent tanulmányban, ugyan is a szakdolgozatomban, nekem nem lesz szükségem drag and drop vagy más klikkelési szokásokra szimplán csak az egérmozdulások lesznek mérvadóak. Viszont, hogy minél tisztább képet kapjak a valós emberszerű egérmozgásokról az én adathalmaz generáló rendszeremben jóval nagyobb 1000Hz-es frekvencián fogom mérni az egérműveleteket és egy sokkal nagyobb méretű adathalmaz tömbbe fogom eltárolni az adatokat. Erre azért lesz szükségem, hogy az elmozdulások mellett egy részletesebb képet kapjak az elmozdulások időbeliségéről és a sebesség változásairól.

Az adathalmaz generáló rendszerről tudni érdemes, hogy a Python Pygame könyvtárával készítettem és úgy működik, hogy a képernyőn egy véletlenszerűen generált ponton megjelenik egy pont, amelyre kattintva eltűnik majd egy újabb pozícióban megjelenik. Ezen folyamat közben a háttérben elmentjük az összes keletkezett egér elmozdulást és a program bezárásával az összegyűjtött elmozdulásokat kiírjuk egy csv-be. A kiírt elmozdulások tartalmazzák az egér útvonalának kezdő és vég koordinátáit és a köztük végbe menő lépéseket. Az összegyűjtött adathalmazból megállapítható, hogy egy átlagos egér elmozdulás 301.14 lépésből áll, ebből az értékből következtetni, tudunk, hogy egy emberszerű egérműveletnek átlagosan 0.3 másodperc alatt meg kell történnie, természetesen ennek a mértéke nagyban függ attól, hogy mekkora távolságot fed le az egér útvonala, de statisztikailag az előbbi állítás tapasztalható. A felmért adathalmazból érdekesség lehet, hogy a legnagyobb lépés 125 pixel volt és az egy – egy lépés között pedig 6.64 pixel van. Ezeket a mért eredményeket később össze tudjuk hasonlítani a neurális hálózat által kapott értékekkel.

IDE SZÚRD BE A DIAGRAMMOT

### Adatgyűjtés

fejlesztés menete: adatgyűjtés, képfelismerés implementálása, egérvezérléshez való adatgyűjtés, neurális háló fejlesztése, tovább fejleszthetőség, felmerülő problémák (determinisztikusság)

Megvalósítás

## Célzás implementálása

Most, hogy már tudjuk, hogy a képernyőn felismerve hol vannak az ellenfelek, már csak annyi a teendőnk, hogy kiválasszuk a megfelelő célpontot majd az egeret mozgassuk a megfelelő helyre. Először is ki kell találni, hogy mi alapján válasszunk elsődleges célpontot, főleg akkor amikor több ellenfél is a képernyőn van. Kétféleképpen történhet egyik megoldás, hogy az egérhez a legközelebbi célpontot válasszuk vagy azt, aki a karakterünkhöz a legközelebb van, ezt úgy lehet megállapítani, hogy megmérjük az adott ellenfelet körbe rajzoló téglalap kerületét és kiválasztjuk a legnagyobbal rendelkezőt. Az egér mozgatását egy betanított neurális halóval szeretném megvalósítani, ehhez írok egy egyszerű programot, amiben különböző pontokra kell kattintani és a program folyamatosan figyelni fogja az egér pozícióinak a változásait, amit majd kimentek és ezen adatok alapján betanítok [10]. Ha nem sikerülne a neurális hálóval megoldani az egér mozgatást akkor további megoldást látok a Bézier görbe alapú [11] illetve a WindMouse [12] algoritmusban.

## Rejtve maradás

A program írásakor ügyelni kell arra, hogy kerüljük azokat a jelenségeket, amelyek nem emberi tényezőre hajaznak, ilyen például az egyenes egérmozgás, az egér állandó sebessége, gyorsulása, továbbá emberfeletti reakció idő. Pontosan ezért van szükség a fentebb említett egér mozgató algoritmusokra, hogy ezt a problémát kiküszöböljük. A csalásellenes szoftverekről nagyon kevés információ áll publikusan rendelkezésre érthető okok miatt, ez legfőképp igaz a modernebbekre, de azért egy két dolgot sejthetünk a működésükről. Tekintsük például a CSGO Valve Anti Cheat (VAC) szoftverét, ami felhasználó oldalán fut különböző szűréseket alkalmazva vizsgálja a játék memória területét elváltozások után kutatva, ha talál valami gyanúsat akkor azt megjelöli és adatokat gyűjt róla. Az összegyűjtött adatok alapján összehasonlítja a többi adatbázisban lévő kártékony kódokkal, ha egyezés van azonnal tilt, ha nincs egyezés akkor további adatot gyűjt. Az elsődleges ellenőrzések után következik a második biztonsági protokoll, ami „Code Blacklisting” névre hallgat, ezen a részen a szoftver az összes betöltött DLL-t megvizsgálja különböző injektálások után kutatva, továbbá ellenőrzi a játékhoz tartozó DLL-ek sértetlenségét. A VAC csak alkalmazás szinten végzi a vizsgálatokat, kernel szintű műveleteket nem hajt végre [13]. A jó hír az, hogy nekünk ezekkel a dolgokkal nem kell foglalkozni, mivel nem módosítunk semmilyen játék fájlt és minden segítségünkre lévő eszközt kívülről érünk el. A ring másik oldalán a Valorant BattleEye (BE) rendszerével állunk szemben, amelyről még kevesebb tudunk, mint a VAC-ról, de az biztos, hogy a BE kernel szinten is fut és már a játékosok puszta viselkedését is elemzi valamilyen gépi tanulás algoritmussal. A kernel szintű védelem miatt a programom egyszerűen nem tud egér utasításokat küldeni, ugyan is a játék csalásellenes szoftvere egyszerűen letiltja azokat. Ezért ahhoz a megoldáshoz folyamodok, hogy egy Arduinohoz csatlakoztatott USB host shieldel [14] támogatom meg az egér vezérlést. Terveim szerint ezt úgy fogom megvalósítani, hogy a program által kiszámított koordinátákat átadom az Arduinonak amely ez alapján a megfelelő koordinátákkal elmozdítja az egeret, hogy ez működjön külön egér drivert kell készítenem. Ha

# Hivatkozások

|  |  |
| --- | --- |
| [1] | M. ANTAL, „SapiAgent: A Bot Based on Deep Learning to,” in *IEEE Access*, Department of Mathematics-Informatics, Faculty of Technical and Human Sciences, Sapientia Hungarian University of Transylvania, 540485 Targu Mures, 2021. |
| [2] | S. Ford, „mmorpg.com,” 27 01 2011. [Online]. Available: https://www.mmorpg.com/news/planet-calypso-sold-for-6-million-usd-2000073586. [Hozzáférés dátuma: 24 04 2023]. |
| [3] | A. J. S. M. Kang, „Multimodal game bot detection using user behavioral characteristics,” in *SpringerPlus*, 2016. |
| [4] | C.-y. P. N.-r. K. H. C. T. Y. H. L. J.-H. L. Yeounoh Chung, „Game Bot Detection Approach Based on Behavior Analysis and Consideration of Various Play Styles,” in *Wiley Online Library*, https://onlinelibrary.wiley.com/, 2013. |
| [5] | G. M. V. Matteo Varvello, „Second Life: a Social Network of Humans and Bots,” pp. 4-6, 2010. |
| [6] | N. Y. H. J. Y. P. A. Su-Yang Yu, „Aimbot Detection in Online FPS Games Using a Heuristic Method Based on Distribution Comparison Matrix,” in *Proceedings of the 19th international conference on Neural Information Processing - Volume Part V*, Doha-Qatar, 2012. |
| [7] | „PyTorch,” Open Source, 09 2016. [Online]. Available: https://pytorch.org/. [Hozzáférés dátuma: 27 09 2022]. |
| [8] | G. Jocher, „YOLOv5 documentation,” 18 Május 2020. [Online]. Available: https://docs.ultralytics.com/. [Hozzáférés dátuma: 8 12 2022]. |
| [9] | „Python MSS’s documentation,” [Online]. Available: https://python-mss.readthedocs.io/. [Hozzáférés dátuma: 13 12 2022]. |
| [10] | I. Arapakis és L. A. Leiva, „Learning Efficient Representations of Mouse Movements to Predict User Attention,” 25 07 2020. [Online]. Available: https://doi.org/10.1145/3397271.3401031. [Hozzáférés dátuma: 08 12 2022]. |
| [11] | J. Davies, 2010. [Online]. Available: https://www.jasondavies.com/animated-bezier/. [Hozzáférés dátuma: 8 12 2022]. |
| [12] | B. J. Land, „WindMouse, an algorithm for generating human-like mouse motion,” 25 4 2021. [Online]. Available: https://ben.land/post/2021/04/25/windmouse-human-mouse-movement/. [Hozzáférés dátuma: 8 12 2022]. |
| [13] | T. Curda, „Analysis and detection of online game cheating software BACHELOR THESIS,” Brno, 2014. |
| [14] | Arduino Corporation, „Arduino USB Host Shield,” Arduino, 7 12 2022. [Online]. Available: https://docs.arduino.cc/retired/shields/arduino-usb-host-shield. [Hozzáférés dátuma: 9 12 2022]. |
| [15] | T. W. V. D. K. A. B. S. a. A. A. B. A. Creswell, „Generative Adversarial Networks: An Overview,” *IEEE Signal Processing Magazine,* %1. kötet1, %1. szám35., p. 53, 2018. |

# Idegen szavak jegyzéke

Bot: Egy olyan szoftveres alkalmazás, amit arra terveztek, hogy különböző feladatokat oldjanak meg automatikusan, jellemzően azzal a céllal, hogy helyettesítse vagy utánozza az emberi beavatkozást. Ezek a feladatok általában ismétlődő jellegűek és sokkal hatékonyabban oldják meg, mint mi emberek.

GAN: Egy olyan neurális hálózat, amely két neurális hálózat versengéséből alkotja meg az ideális kimenetet. Az egyik alhálózat a generátor, amely előállítja a teszt kimenetet, a másik hálózat a diszkriminátor, ami elbírálja a generátorból származó kimenet valódiságát. A diszkriminátor úgy van feltanítva, hogy megtudja különböztetni a helyes és helytelen adatot. Kiemelten fontos, hogy generátornak fogalma sincs a valódi és mesterséges mintázatokról. Az egyetlen dolog, amiből tanulni tud az a diszkriminátortól származó eredmény. A végeredmény akkor számít jónak, ha a generátor már betudja csapni a diszkriminátort. [15]

Supervised learning: magyarul felügyelt tanulás a gépi tanulás egyik alkategóriája, amiben címkézett adathalmazokat használnak olyan algoritmusok betanítására, amelyek adatok osztályozására vagy eredmények pontos előrejelzésére szolgálnak. Ez úgy történik, hogy a bemeneti adatokat betáplálják egy modellbe, ami addig módosítja a súlyokat, amíg a modell megfelelően nem illeszkedik az elvárt eredményhez. [16]