|  |  |  |
| --- | --- | --- |
|  | **NEUMANN JÁNOS**  **INFORMATIKAI KAR** | NIK_cimer.jpg |

**SZAKDOLGOZAT**

|  |  |  |
| --- | --- | --- |
| **OE-NIK**  **2023** | Hallgató neve:  Hallgató törzskönyvi száma: | **Merész Patrik**  **T010001/FI12904/N** |

Óbudai Egyetem

Neumann János Informatikai Kar

Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

|  |  |
| --- | --- |
|  |  |
| Hallgató neve: | **Merész Patrik** |
| Törzskönyvi száma: | T010001/FI12904/N |
| Neptun kódja:        A dolgozat címe: | VH3Q8P |

**Fokozottan emberszerű interfész fejlesztése számítógépes rendszerekhez Development of enhanced human-like interface for computer systems**

|  |  |
| --- | --- |
| Intézményi konzulens: | Dr. Vámossy Zoltán |
| Külső konzulens: |  |
| Beadási határidő: | 2023. december 15. |
| A záróvizsga tárgyai: | Számítógép architektúrák  Szoftvertervezés és -fejlesztés specializáció |

**A feladat:** 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. A hallgató feladata egy olyan rendszer megtervezése és fejlesztése, amikor a felhasználó segédeszközeinek szimulálásával (elsősorban egér, kisebb részben billentyűzet helyettesítésével) generál olyan életszerű választ, amit a fogadó partner emberszerűnek tapasztal.

Vizsgálja meg, hogy milyen következményekkel járhat, ha nem vagyunk képesek egy botot detektálni. A szakirodalom tanulmányozása alapján ismerje meg az olyan rendszereket, amelyek képesek eldönteni, hogy a felhasználó gép vagy ember, fejtse ki ezek működési elvét, taglalja ezen rendszerek hátrányait és előnyeit. Tervezze meg és készítse el saját interfész rendszerét, ahol a generált felhasználói válaszokat a gyakorlatban is elérhető ilyen rendszerek még nem képesek megkülönböztetni az emberi válaszoktól. Eredményeit hasonlítsa össze más ilyen célú rendszerekkel. Prezentálja élő demonstrációval a rendszer hatékonyságát.

**A dolgozatnak tartalmaznia kell**:

* a feladat leírását,
* szakirodalomból megismerendő releváns fejlesztések ismertetését, valamint azok értékelését,
* a megvalósításhoz szükséges módszerek tömör ismertetését,
* a megvalósítandó rendszer tervét,
* tesztelési, mérési adatokat,
* az eredmények értékelését, más megvalósításokkal való összehasonlítását,
* továbbfejlesztési lehetőségek ismertetését,
* a dokumentációt, a programot, a szükséges inputadatokat, valamint a rendszert bemutató prezentációt elektronikus mellékleten.

(PH.)

..........……………….

Dr. Vámossy Zoltán

intézetigazgató

|  |  |
| --- | --- |
| A szakdolgozat elévülésének határideje: **2025. december 15.** (OE TVSz 55.§ szerint)    A dolgozatot beadásra alkalmasnak tartom: |  |
| ……………….. | ..……………………. |
| külső konzulens | intézményi konzulens |

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 202... ..........................................

|  |  |
| --- | --- |
|  | hallgató aláírása |

|  |  |
| --- | --- |
|  |  |

Tartalomjegyzék

[1. Absztrakt 7](#_Toc146898502)

[2. Bevezetés 7](#_Toc146898503)

[2.1 Motiváció a projekt elkészítéséhez 8](#_Toc146898504)

[2.2 Hasonló projektek 9](#_Toc146898505)

[3. Bot detektálás videójátékokban 11](#_Toc146898506)

[3.1 Viselkedés analízis 12](#_Toc146898507)

[3.2 Heurisztikus módszerek 12](#_Toc146898508)

[3.3 Összegzés 13](#_Toc146898509)

[4. Rendszerterv 14](#_Toc146898510)

[4.1 Követelmények 14](#_Toc146898511)

[4.2 Funkciós lista 14](#_Toc146898512)

[4.3 Felhasznált eszközök 15](#_Toc146898513)

[4.4 Rendszerterv diagramm 16](#_Toc146898514)

[5. Fejlesztés 17](#_Toc146898515)

[5.1 Képfelismerés adatgyűjtés és tanítás 17](#_Toc146898516)

[5.2 Emberszerű egérmozgás 18](#_Toc146898517)

[5.3 GAN neurális hálózat fejlesztése 21](#_Toc146898518)

[5.4 Tesztelés és fejlesztések 23](#_Toc146898519)

[5.5 Összefoglalás 24](#_Toc146898520)

[6. Konklúzió 25](#_Toc146898521)

[6.1 A játékteljesítményre gyakorolt hatás 25](#_Toc146898522)

[7. Hivatkozások 26](#_Toc146898523)

[8. Ábrajegyzék 28](#_Toc146898524)

# 1. Absztrakt

A szakdolgozat tartalmazza azon preventív rendszerek bemutatását, amelyek képesek megkülönböztetni gépi, illetve emberi interakciókat, különös tekintettel a bot 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 viselkedéséhez egy GAN típusú neurális hálózatot fogok használni, ami az egérmozdulatok generálásáért fog felelni. A neurális hálóval a saját egérmozgatási adataimmal feltanítva fogok emberszerű egérmozgást generálni. A dolgozat végén összegzem a fejlesztés menetét és értékelni fogom a projekt sikerességét.

# 2. Bevezetés

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 észben tartani, 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.

A 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.

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 félre tájékoztassák 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 videójátékokban a legnagyobb kihívás 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.

## 2.1 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. Ezen felül 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) 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.

## 2.2 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] ezt mind pusztán az egérmozdulatok vizsgálata alapján történik.

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 [4].

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 más részről a teljesítményük is elmarad a kívánttól.

# 3. Bot detektálás videójátékokban

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 [5]. 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 [6].

A botok detektálásához és felderítéséhez számos tanulmányt mutattak be a tudományos és az ipari életben. Ezek a módszerek kettő kategóriába sorolhatók: kliensoldali 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 információit és a memória állapotát vizsgálják 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.

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 [6], a játékszerverek által generált naplófájlokat képesek elemezni, és nem okoznak semmilyen mellékhatást, például teljesítménycsökkenést vagy konfliktust más programokkal.

## 3.1 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 [7].

A közösségi tevékenységelemzés a közösségi hálózat jellemzőit használja fel az emberi és a játékrobotok megkülönböztetésére [8]. 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.

## 3.2 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 [9].

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.

## 3.3 Összegzés

A botok felderítése elengedhetetlen a jelenlegi videójáték iparban, biztosítani kell a tisztességes játékmenet fenntartását, a játékosok elégedettségének növelését, a gazdaság és a monetizációs rendszerek védelmét, a biztonság fokozását, a közösség bizalmának erősítését, valamint a játékfejlesztők pénzügyi javainak érdekét. Egy fejlett botfelismerő rendszer a fentebb említett technikákát ötvözve használják. Mindegyik módszernek meg van a saját előnye és hátránya. A legnagyobb problémát az jelenti, hogy maximalizálják a pontosságot és minimalizálják a fals pozitív eredményeket, továbbá, hogy ne rontsák a játék teljesítményét. A felhasznált bot detektáló rendszerek kombinációját jelentősen meghatározza, hogy milyen a játék felépítése és a botok fejlesztésének fejlődő jellegétől. Például egy modern mélytanuló algoritmuson alapuló bot detektálásához egy teljesen más eljárású rendszerre van szükség, mint a korábbi csalási módszerek kiszűréséhez. Fontos a folyamatos fejlődés hangsúlyozása és az elmúlt évek tapasztalataiból kiindulva elmondható, hogy ez a folyamat egy soha véget nem érő macska-egér játék.

# 4. Rendszerterv

## 4.1 Követelmények

A neurális hálózatnak emberszerű egérmozdulattal kell visszatérnie, ami azt jelenti, hogy kerülnie kell az egyenes vonalakat és az irányváltoztatásoknál sem látszódhat semmi gépre utaló jel, továbbá a rendszer az egérmozdulat generálását 0.4 másodpercen belül képes legyen elvégezni. Az emberszerű egérmozgás generálásához a neurális hálózatnak, meg kell tanulnia az egérmozdulat sorokban lévő időzítéseket, gyorsulásokat, lassulásokat, pusztán a megfelelő egér pálya nem megfelelő.

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.

## 4.2 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. A felhasználónak lehetősége van kiválasztani, hogy a célpont objektumnak melyik részét kívánja célozni. A grafikus interfészben beállítható a képernyőn vizsgált terület mérete, amely a képernyő közepén van elhelyezve. Ebben a rész ablakban számolja ki a képfelismerő algoritmus a releváns értékeket. A program lehetőséget ad a használónak, hogy megváltoztassa az egérmozgatás paramétereit, mint például a távolságot, amelyen belül a program aktiválódik ez az érték megjelöli azt az euklideszi távolságot az egér kurzor és a célpont között, amelyen belül az egérmozgató automatika működésbe lép. További opciót nyújt a program arra, hogy az egér sebességét változtatni tudjuk. Az egérmozgatás sebessége egy szorzó, amely megváltoztatja az egér maximális lépés távolságait a szorzó szorosával. A programban alapértelmezetten ez 10 pixel.

## 4.3 Felhasznált eszközök

Ebben a fejezetben bemutatom a felsorolt technológiákat és az eszközöket, valamint azt, hogy milyen szerepük és jelentőségük van ebben a projektben.

A megvalósítandó programot Python nyelvben fogom megírni. A Python programozási nyelv 3.9-es verzióját fogom használni, mivel ez támogatja megfelelően a PyTorch és CUDA könyvtárakat. A programban az egérvezérlést az a Win32Api könyvtárral fogom végrehajtani, ami lehetővé teszi a Windows operációs rendszer API-jának közvetlen hívását. A képernyőmentéseket Mss (Multi Screen Shot) nevű könyvtárral fogom megoldani. Az adatkezelést Pandas és Numpy-jal valósítom meg.

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 [10] 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 [11] 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 [12] 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, illetve a feltanított modellek kimentelét.

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 és a PyMouse melyek 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 könyvtár [13], megfelelő hardverrel képes akár a 60fps fölötti gyorsasággal rögzíteni a képernyő tartalmát. Leggyakrabban automatizált tesztek és videófelvételekre használják.

## 4.4 Rendszerterv diagramm

Az alábbi rendszerterv diagramm (Lásd 1. ábra) ábrázolja a rendszeren belüli komponenseket, kapcsolataikat és az adatáramlást. Segíti a fejlesztés menetét azáltal, hogy egy egyszerű összefoglaló képet ad a rendszer működéséről.

A rendszer kiindulási pontja a grafikus felhasználó interfész, ahol a program paramétereit tudjuk változtatni. A paraméterekkel képesek vagyunk kiválasztani a megfelelő képfelismerő modellt és az egérmozgató algoritmus bizonyos a beállításait módosítani. Ezután a rendszer egy ciklus folyamatba lép.



1. ábra Rendszerterv diagramm

# 5. Fejlesztés

## 5.1 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 vagy másnéven felügyelt módon tanítsuk.

A 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 [18].

A Yolov5 modellt a dokumentációjában [11] leírt instrukciókat követve sikerült feltanítanom Google Colabon a saját adathalmazommal, mind össze 2 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.

## 5.2 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 [4] 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 részt vettek 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.

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 két mintavételezés között az átlag pedig 6.64 pixel volt. Ezeket a mért eredményeket később össze tudjuk hasonlítani a neurális hálózat által kapott értékekkel. Az alábbi képeken (Lásd 2. ábra és tovább) egy-egy mérési eredmény látható. A mérések a 0,0 pontból indulnak. Jelenleg az adathalmaz 2819 ehhez hasonló mozdulatsorból áll.



2. ábra Saját egérmozdulat mérési eredménye



. ábra Saját egérmozdulat mérési eredménye



. ábra Saját egérmozdulat mérési eredménye



5. ábra Saját egérmozdulat mérési eredménye

Mivel a szakdolgozatom írása közben többször felmerült bennem az a gondolat, hogy ágyúval lövök verébre amiért az emberszerű egérmozgás generálását neurális hálózattal oldom meg. Ezért szeretném a keletkezett eredményeket összehasonlítani egy harmadik módszerrel, ami egy jóval egyszerűbb algoritmussal oldja meg ezt a problémát, ezt az algoritmust a szerzője WindMouse [14] névre keresztelte. Ez a program lényegében egy 50 kódsorból álló függvény, amely középiskolai fizika ismereteket felhasználva random zajjal generál emberszerű egérmozdulatot. Az eredmény az alábbi ábrán látható (Lásd 3. ábra).



. ábra WindMouse által generált egérmozdulat

## 5.3 Lineáris regresszión alapuló neurális hálózat fejlesztése

Az első neurális hálózat, amivel próbálkoztam, lineáris regresszión alapul, mivel a megvalósítandó problémát úgy képzeltem el, hogy egy adott ponthalmazt kell összekötnöm egy képzeletbeli vonallal csak azzal a megkötéssel, hogy az egyenes ne az egész ponthalmazra illeszkedjen, hanem csak egy kisebb szeletére. Főleg azért választottam ezt a típusú neurális hálózatot, mert az órákon sokat tanultam róla és az implementálása meglehetősen egyszerű.

A modellt úgy implementáltam, hogy inputként egy koordinátapárt várjon (x,y)-t, ami egy irányvektornak feleltethető meg. Kimenetként pedig egy olyan (Δx, Δy) elmozdulást vártam, ami a tanító adathalmazból a legnagyobb fitnesz értékű eredményt adja.

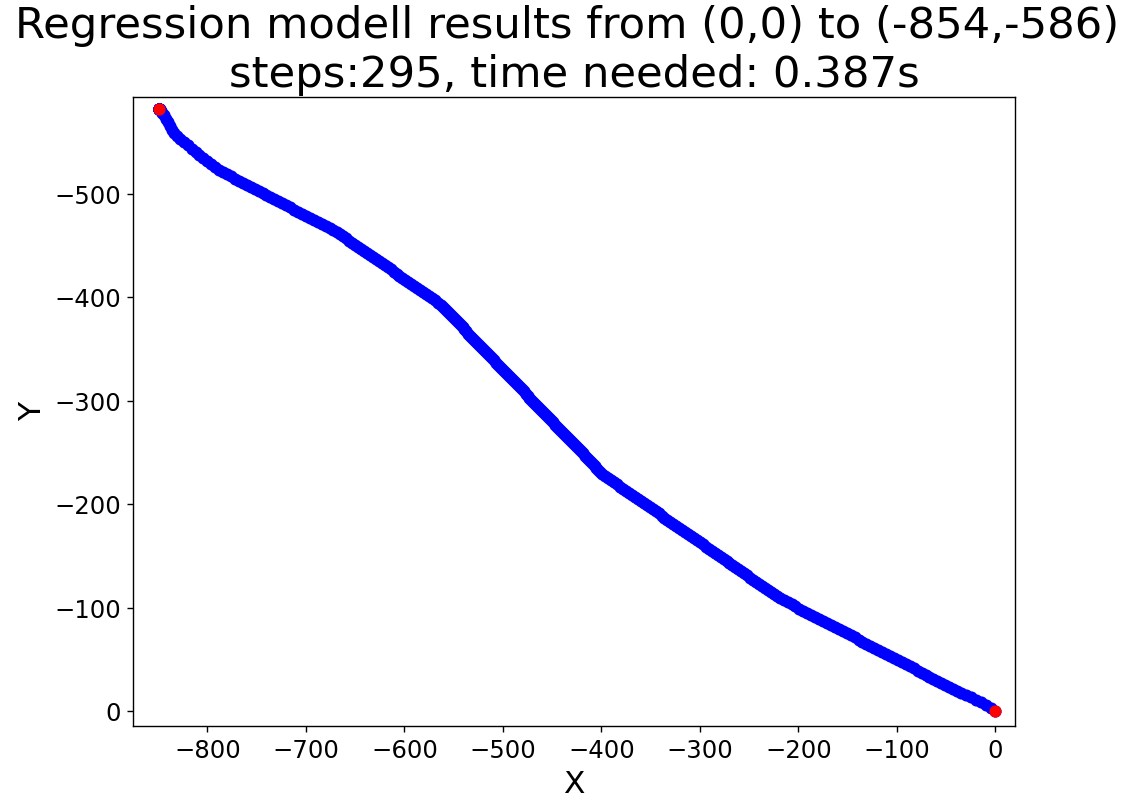
A gyakorlatban a működést úgy kell elképzelni, hogy a feltanított modellnek átadok egy (100,100) koordináta párt, majd a kimenet visszaadja a legjobb elmozdulást, ami (0,0) pontból a (100,100) pontba mutat, ami legyen jelen esetben (9,8) ezt a kimenetet kivonom az előző bemeneti paraméterből. Most a modellnek a (100-9,100-8) pontra kell megkeresni a legjobb elmozdulást és ezt a folyamatot addig folytatom egy ciklusban ameddig a kimeneti elmozdulások összege nem lesz egyenlő a megadott kezdőponttal vagy is a (100,100) ponttal.

Ennek a megvalósításnak vannak előnyei és hátrányai is. Az előnyök közé felsorolnám azt, hogy a viszonylag kevés tanító adathalmazzal is jól működött a modell és hogy a generált egérmozdulatok első ránézésre tényleg emberszerűnek hatnak, de ez csak olyan esetekre igaz, amikor a bemenethez létezik hasonló tanító minta. Továbbá kiemelném a többi megvalósítással ellentétben, hogy ez a modell képes olyan mozdulatsor halmazt generálni, amelynek elemszáma viszonyul a megtett távolsághoz, röviden megfogalmazva rövid egérmozdulathoz, kis lépésszámú kimenet tartozik és az eredmény pontosan oda mutat, amelyre a bemeneti paraméter jelzi. (Túl tanulás veszélyei)

A hátrányok közé megemlítem azt, hogy a modell elveszíti az emberre jellemző randomitást és a tipikus jellemzőket. A modell egy bementre csak egy útvonalat képes generálni. Ezzel szemben, ha egy embert megkérünk arra, hogy mozgassa az egeret 100 pixellen keresztül egy adott irányban, akkor minden próbálkozás során hasonló, de egy másik megoldást fogunk kapni. A modell továbbá elveszti azokat a tulajdonságokat, amit emberi egérmozdulatok során tapasztalhatunk, miszerint a mozdulatsor általában egy gyorsuló szakasszal indul, ahol a lépések közötti távolság folyamatosan növekszik, majd egy állandó sebességű szakasz következik és végül a cél felé közeledve egy lassuló mozgás figyelhető meg. A modell a működéséből adódóan először az adható legnagyobb lépésszámokat generálja és a távolság csökkenésével változik a kimeneti távolságok mértéke is. A legfőbb probléma pedig az, hogy a generált kimenet sok-sok iteráció alapján készül és ez negatív hatással van a rendszer teljesítményére. Általánosságban elmondható, hogy egy egérmozdulatsor 0.3 másodperc körül generálódik, amely az élesben való teszteléshez nagyon lassúnak mondható.

Összefoglalásként azt szeretném kiemelni, hogy a lineáris regresszió egy könnyen implementálható, egyszerű megoldást nyújtott a problémára, de sajnos közel sem optimálisat és inkább csak kísérletezés szempontjából emelném ki ezt a felépítést. A generált eredmények az alábbi felsorolt ábrákon lesznek láthatók.

## 



## 5.3 GAN neurális hálózat fejlesztése

A Generative Adversarial Network röviden GAN nevezetű eljárást 2014-ben Ian J. Goodfellow és a kollégái dolgozták ki. Ez egy olyan neurális hálózatot jelent, 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 olyan hiteles eredményt tud produkálni, amelyről a diszkriminátor nem tudja eldönteni, hogy valódi, illetve mesterségesen generált [17] [15].

A GAN típusú neurális hálózatokat főleg azokon a területeken használják, ahol a bementi adatból valamilyen szabály jelleget szeretnénk felismerni és reprodukálni. Ilyen problémák lehetnek az alábbiak, mint például kép alkotás, szöveg generálás [15] vagy esetleg emberszerű egérmozdulat reprodukálás is. A GAN típusú hálózatok egyik fő ismérve, hogy unsupervised módon tanulnak és képesek egy adott kimenetre több megoldást adni, szemben a többi mélytanuláson alapuló módszerrel. Kiváló példa erre egy GAN típusú képgenerátor, ami, ha paraméterül egy macska szót adunk, akkor képes akár több különböző macskát ábrázoló képet generálni, ahelyett, hogy csak egy adott képet tudna visszaadni.

Az unsupervised learning: a supervised learning ellentéte, amelyben címkézetlen adathalmazok elemzését végzik. Ezek az algoritmusok emberi beavatkozás nélkül fedezik fel a mintázatokat és adatcsoportosításokat [19].

A GAN neurális hálózattal a célom, hogy a generálni tudjak emberszerű egérmozdulatokat az alapján, hogy a modell felismeri az egérmozgatásban lévő mintázatokat. Az egérmozdulatokról többnyire megállapítható az a tulajdonság, hogy van benne egy gyorsulási szakasz, majd ezt követi egy viszonylag lineáris mozgás és az utolsó pillanatban amikor a mozdulatsor a végcélhoz közeleg akkor lassulás figyelhető meg.

A neurális hálózat két komponensből tevődik össze, egyik a generátor a másik a diszkriminátor. A diszkriminátor az a neurális hálózat, amelyet a tanító adathalmazzal feltanítunk és ezáltal képes eldönteni a következő bejövő adatról, hogy valós vagy mesterséges. Szemben a másik alhálózattal, amit generátornak hívnak, ami azért felel, hogy a generátorból kapott válaszok alapján úgy változtassa a modell súlyait, hogy az általa generált kimenet a legkevesebb veszteséget eredményezze vagy olyan eredményeket adjon, amelyről a diszkriminátor helytelenül állapítja meg a kimenet valódiságát, más szóval becsapva azt. A neurális hálózat implementálását PyTorchban fogom megoldani.

A projekt előrehaladásával a GAN neurális hálózat tanítási és optimalizációs folyamatait fogom részletezni. Tárgyalni fogom a hiperparamétereket, tanulási sebesség mértékét, a batch-ek méretét és az alhálózatok architektúráit. Továbbá ezen pontok hatásait a generált egérmozgás minőségére. Legutolsó sorban pedig a fejlesztés közben felmerülő kihívásokat és kezelésüket fogom bemutatni.

Az elkészült modell által generált kimenet kvalitásainak meghatározásához különböző metrikákat fogok alkalmazni. A mérési módszerekhez tartozhat vizuális vizsgálat vagy az olyan mérési számok alkalmazása, mint például átlagos négyzetes hiba és hasonlósági mutatók.

Ez a fejezet ki lesz egészítve amint leprogramozom a GAN neurális hálót…

## 5.4 Tesztelés és fejlesztések

A szakdolgozat jelenlegi állása alapján a képfelismerést a WindMouse algoritmussal kiegészítve tudom tesztelni a videójátékokban. Kezdetben a tesztelést offline üzemmódban, hajtottam végre, hogy elkerüljem az esetleges szankciókat. Legelőször azzal a problémával szembesültem, hogy a képfelismerés nagyon lassan futott átlagosan 20 fps környékén. Tudni érdemes, hogy a rendszer teljesítményét jelentősen befolyásolja a képfelismerés sebessége, ugyan is ez a folyamat frissíti a célpont koordinátákat és csak ezután lehet végre hajtani az egérvezérlést, ha lassan frissül a kiszámolt pozíció akkor szaggatott hatást fogunk tapasztalni.

Az elsődleges megoldás erre az volt, hogy redukáltam a felvett vizsgálni való képek méretét, kezdetben 1920x1080-as felbontásban végeztem a tesztelést és most már belátom, hogy jóval kisebb méret is 320x320-as méret is teljesen megfelel a célnak. Ezzel a módszerrel jelentős teljesítmény növekedést értem el és mivel tapasztalt videójátékosként tudom, hogy ha a célkeresztet megfelelő pontokra, illetve sarokpontokra helyezem akkor egyáltalán nem jelent gondot az, hogy a képfelismerő szoftver kisebb területet vizsgál a rendszer összteljesítményét tekintve.

A második nagy előrelépést az jelentette, hogy sikerült szert tennem egy nagyobb teljesítményű videókártyára, amellyel a kétszeresére tudtam növelni a képfelismerés gyorsaságát. Ezekkel a módosításokkal a képfelismerés már képes volt közel 60fps frissítési értéket produkálni.

Az utolsó nagy áttörést annak köszönhettem, hogy az irodalom kutatásom alatt rátaláltam a TensorRT SDK-ra [12] amit az Nvidia fejlesztett. Ez egy olyan könyvtár, ami optimalizálja a gépi tanuláson alapuló modellek kiértékelő teljesítményét. Az én esetemben az annyit jelent, hogy a képfelismerés 60fps helyett mást már 110 és 120 közötti fps értkékeket produkál. Hátránya az, hogy csak Nvidia kártyákat támogat.

A program futtatása közben egy másik problémával is szembesültem. A kihívás abban rejtőzik, hogy a mozgó célpontokat a programom nehezen tudja lekövetni, mivel az objektum aktuális pozícióját meghatározva egy egérmozdulat szimulálása 0.2 – 0.3 másodperc alatt megy végbe és mire az egér a megfelelő koordinátákra kerül, addig már elmozdult a célpont. Természetesen ezt a problémát meglehet oldani úgy, hogy az egérmozgatást villámgyorsan lefuttatjuk és akkor kevés ideje marad a célpontnak az elmozdulásra, de ebben az esetben a programunk elveszíti az emberszerű jellegét és nagy valószínűséggel elveszítjük a fiókunkat is.

A megoldást a fentebb említett problémára a Kálmán-filter alkalmazását látom. A Kálmán-filter vagy másnéven Kálmán szűrő egy olyan matematikai formula, amellyel képesek vagyunk egy objektum előző helyzeteiből kiindulva megbecsülni a jövőbeli állapotát [16].

## 5.5 Összefoglalás

Ebben a fejezetben az emberszerű egérmozgatásért felelős rendszerek tulajdonságait fogom röviden bemutatni. Az összehasonlítás szempontjait egy táblázatban foglaltam össze (Lásd 4. ábra) a könnyebb átláthatóság miatt. A táblázatban még nincsen kitöltve minden egyes mező, mivel még további méréseket kell alkalmaznom a rendszerek teljesítményéhez kapcsolódóan.

|  |  |  |  |
| --- | --- | --- | --- |
| Eljárás neve: | SapiAgent autoenkóder | Saját GAN | WindMouse |
| Tanuló adathalmaz: | 128 ember, 3 és 5 perces mérések | Általam készített  2818 minta | Nem rendelkezik |
| Monitorozás frekvenciája: | 60Hz | 1000Hz | Nem rendelkezik |
| Szükséges idő egy generált adathalmaz létrehozásához: | Kevesebb mint 1ms a tanulmány alapján | Később kerül kitöltésre | 3.6ms |
| Átlagos lépésszám egy mozdulatsorban: | Később kerül kitöltésre | Később kerül kitöltésre | 218 |
| Egy lépés átlagos mérete: | Később kerül kitöltésre | Később kerül kitöltésre | 10.2 pixel |
| Használatáért a felhasználót kitiltás érte: | Később kerül kitöltésre | Később kerül kitöltésre | Még nem |

6. ábra Az összefoglaló táblázat az egérmozdulat generáló módszerekről

Itt részletesebben kifejtem az összehasonlítást…

# 6. Konklúzió

## 6.1 A játékteljesítményre gyakorolt hatás

A program használata mellett a játékok átlagosan 5-10%-os teljesítmény csökkenéssel futnak. Felhasználói szinten az alábbi dolgokat tapasztaltam 23 CS:GO mérkőzés adataiból kiindulva. Javult a pontosság, a fejlövési arány 22%-ról 50%-ra emelkedett, viszont ölési/halálozási arány érdemben nem javult. Továbbá észrevettem magamon, hogy a program használatával jobban izgulok a játék közben és a kognitív készségek, mint például a taktikai döntések, helyezkedések sokkal inkább a háttérbe szorulnak. A fentebb leírt eredményeket egy online profil alapú statisztikai oldal adatait használtam fel.

Összességében elmondható, hogy a program bizonyos szempontok alapján növelni tudja a játékos teljesítményét, de egy amatőr játékosból nem fog világklasszist készíteni.

# 7. Hivatkozások

|  |  |
| --- | --- |
| [1] | C. Feher, Y. Elovici, R. Moskovitch, L. Rokach és A. Schclar, „User identity verification via mouse dynamics,” in *Elsevier*, Academic College of Tel-Aviv, Isreal, 2012. |
| [2] | L. A. Leiva, L. Arapakis és C. Iordanou, „My Mouse, My Rules: Privacy Issues of Behavioral User Profiling via Mouse Tracking,” *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval,* pp. 51-61, 2021. |
| [3] | N. Banholzer, S. Feuerriegel, E. Fleisch, G. F. Bauer és T. Kowatsch, „Computer Mouse Movements as an Indicator of Work Stress: Longitudinal Observational Field Study,” *JMIR Publications,* %1. kötet23, %1. szám4, 2021. |
| [4] | A. Margit, B. Krisztian és F. Norbert, „SapiAgent: A Bot Based on Deep Learning to Generate Human-Like Mouse Trajectories,” in *IEEE Access*, Department of Mathematics-Informatics, Faculty of Technical and Human Sciences, Sapientia Hungarian University of Transylvania, 540485 Targu Mures, 2021. |
| [5] | 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]. |
| [6] | A. R. Kang, S. H. Jeong, A. Mohaisen és H. K. Kim, „Multimodal game bot detection using user behavioral characteristics,” in *SpringerPlus*, SpringerPlus, 2016. |
| [7] | Y. Chung, C.-y. Park, N.-r. Kim, H. Cho, T. Yoon, H. Lee és J.-H. Lee, „Game Bot Detection Approach Based on Behavior Analysis and Consideration of Various Play Styles,” in *Wiley Online Library*, https://onlinelibrary.wiley.com/, 2013. |
| [8] | G. M. V. Matteo Varvello, „Second Life: a Social Network of Humans and Bots,” pp. 4-6, 2010. |
| [9] | S.-Y. Yu, N. Y. Hammerla, J. Yan és P. Andras, „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. |
| [10] | „PyTorch,” Open Source, 09 2016. [Online]. Available: https://pytorch.org/. [Hozzáférés dátuma: 27 09 2022]. |
| [11] | G. Jocher, „YOLOv5 documentation,” 18 Május 2020. [Online]. Available: https://docs.ultralytics.com/yolov5/. [Hozzáférés dátuma: 8 12 2022]. |
| [12] | Nvidia, „TensorRT,” Nvidia, 2020. [Online]. Available: https://developer.nvidia.com/tensorrt. [Hozzáférés dátuma: 03 04 2023]. |
| [13] | „Python MSS’s documentation,” [Online]. Available: https://python-mss.readthedocs.io/. [Hozzáférés dátuma: 13 12 2022]. |
| [14] | 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]. |
| [15] | A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta és A. A. Bharath, „Generative Adversarial Networks: An Overview,” *IEEE Signal Processing Magazine,* %1. kötet1, %1. szám35., p. 53, 2018. |
| [16] | E. R. Kalman, „A New Approach to Linear Filtering,” in *Research Institute for Advanced Study*, Baltimore, Md, 1960. |
| [17] | I. J. Goodfellow és J. Pouget-Abadie, „Generative Adversarial Nets,” in *Departement d’informatique et de recherche op ´ erationnelle*, Universite de Montr ´ eal, 2014. |
| [18] | IBM, „IBM - Supervised Learning,” [Online]. Available: https://www.ibm.com/topics/supervised-learning. [Hozzáférés dátuma: 06 05 2023]. |
| [19] | IBM, „IBM - What is unsupervised learning,” International Business Machines, [Online]. Available: https://www.ibm.com/topics/unsupervised-learning. [Hozzáférés dátuma: 19 05 2023]. |

# 8. Ábrajegyzék

[1. ábra Rendszerterv diagramm 11](file:///C:\Users\User\Desktop\AimAssistUsingAi-main\Részletes%20Rendszerterv.docx#_Toc144751907)

[2. ábra Saját egérmozdulat mérési eredménye 14](file:///C:\Users\User\Desktop\AimAssistUsingAi-main\Részletes%20Rendszerterv.docx#_Toc144751908)

[3. ábra Saját egérmozdulat mérési eredménye 15](#_Toc144751909)

[4. ábra Saját egérmozdulat mérési eredménye 15](#_Toc144751910)

[5. ábra Saját egérmozdulat mérési eredménye 15](#_Toc144751911)

[6. ábra WindMouse által generált egérmozdulat 16](#_Toc144751912)