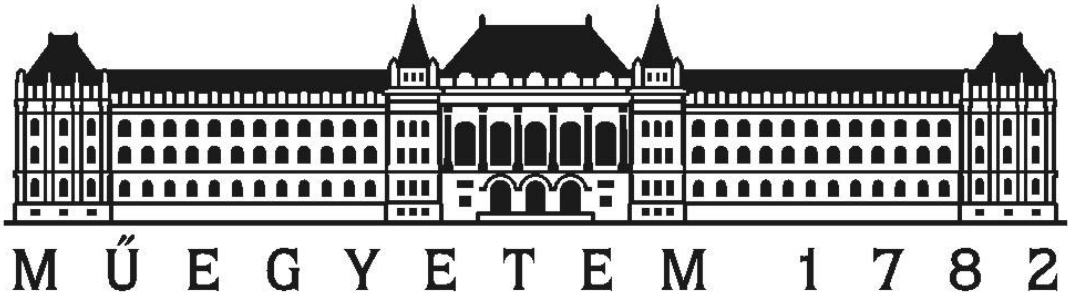


Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Hálózati Rendszerek és Szolgáltatások Tanszék

Okos otthonos funkciók fejlesztése képfeldolgozás és mesterséges intelligencia segítségével

TDK dolgozat

Készítette:

Wágner Bánk

Konzulens:

Dr. Szabó Sándor

2022

Tartalom

1 Kivonat

Az elmúlt évtizedben egyre több okosotthon rendszer, eszköz és fejlesztés jelent meg. A bővülő felhasználási mód mellé egyre több vezérlési, otthonirányítási lehetőség is elérhetővé vált, eleinte távirányítókkal, később okostelefonokkal lehetett kiadni a parancsokat. Ezen a területen a legújabb fejlesztés az, hogy már hangvezérléssel is tudunk parancsolni eszközeinknek.

Dolgozatomban egy eddig nem elterjedt irányítási módot fogok megvizsgálni, a képfeldolgozás alapú okosotthon irányítást. A módszerrel a célom, hogy könnyebben irányíthatóvá és felhasználó-barátibbá tegyem az okosotthon rendszerek kezelését. A megvalósítás során szem előtt tartom, hogy a rendszer a lehető legegyszerűbben, intuitív módon legyen kezelhető, nemzetközi kézjelekkel, mozdulatokkal. A rendszer szükségtelenné teszi a különböző bonyolult mobilalkalmazásokat és komplikált vezérlőfelületeket, használatához nincs szükség további eszközökre.

TDK dolgozatban egy olyan rendszert készítek el, amely az OpenCV keretrendszeren futva gépi tanulási metódussal dolgozza fel a kézjeleket, mivel eddig ilyen környezetben nincs használatban hasonló technológia. Összehasonlítom több különböző gépi tanulási módszer teljesítményét annak érdekében, hogy megtaláljam az adott feladat megoldására legalkalmasabb megoldást.

2 Abstract

In the past decade, more and more smart home systems, devices, and developments have appeared. In addition to the expanding methods of use, more and more control and home control options become available, at first commands could be issued with remote controls, and later with smartphones. The latest development in this area is that we can now command our devices with voice control.

In my thesis, I will examine a control method that has not been widespread until now, image processing-based smart home control. My goal with the method is to make the management of smart home systems easier to control and more user-friendly. During the implementation, I keep in mind that the system should be handled as simply and intuitively as possible, using international hand signals and gestures. The system eliminates the need for various complicated mobile applications and complicated control interfaces, and no additional devices are required for its use.

In my TDK thesis, I am creating a system that processes hand signals using a machine learning method running on the OpenCV framework, since no similar technology has been used in such an environment so far. I compare the performance of several different machine learning methods to find the best solution for solving a given task.

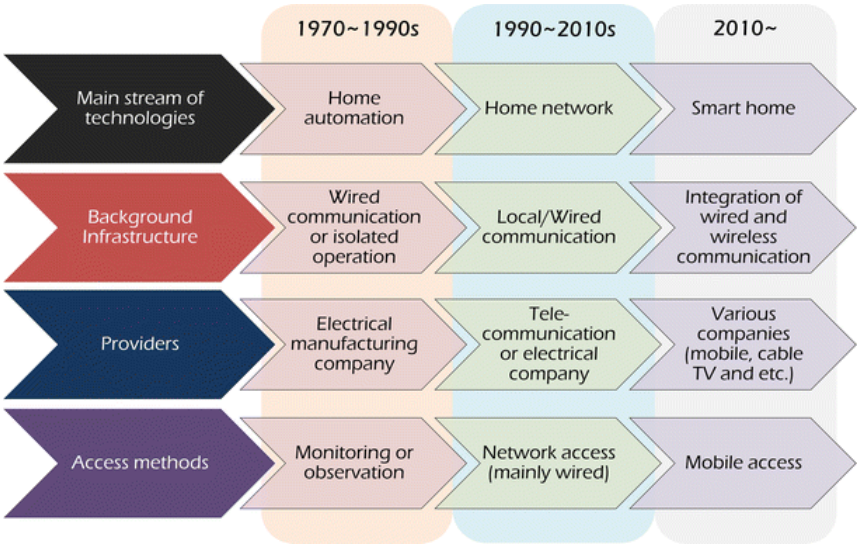
3 Háttér

3.1 Okosotthonok rövid története, technológiai fejlődése

Okosotthonokról csak nem olyan rég óta beszélhetünk, viszont már a 20. század elején is megjelentek automatizált eszközök. Ilyen volt például 1905-ben az első elektromos porszívó vagy 1907-ben az első elektromos mosógép. [1]

Az 1900-as évek második felére érkezett el a technológia olyan szintre, hogy már meg tudtak jelenni a mai okosotthonokban használt eszközök és technológiák első változatai. Ezek közé tartozik az első érintő szenzor, első kommunikációs sztenderdek, mint az X10 vagy az IPv4. Ebben az idősávban már bőségesen kezdtek megjelenni első generációs otthon automatizáló eszközök.

A 2000-2011 közötti időszak kezdte megteremteni a szükséges alapot az okosotthon technológiáknak. A 3G, majd a 4G megjelenésével az első stabil hálózat jelent meg, amivel biztosítani lehetett eszközöknek a vezérlését távolról. A Google Play és az Apple App Store megjelenésével könnyen elérhetővé váltak különböző telefonos alkalmazások. Ebben az időszakban jelent még meg több, elterjedt kommunikációs szabvány is, mint a Z-Wave, a Zigbee vagy a Bluetooth Low Energy.



ábra 3.1 *Okosotthon rendszerek fejlődése*

Azonban, 2011-től kezdődően jelentek meg a mai napra más elterjedt okosotthon eszközök. Megjelent az első igazán elterjedt okos villanykörte a Philips

Hue valamint az Amazon kiadta az első Echo, ami az első nagyon elterjedt virtuális asszisztens eszközét és a nagy cégek elkezdték felvásárolni a kisebb startupokat. Például a Google a Nest-et, ami egy beindulóban lévő intelligens termékeket gyártó cég volt, vagy az Amazon a Ring-et. Ahol okos kapucsengőt gyártottak. Ez nem véletlen, mivel ez a piac folyamatosan és nagyon gyorsan növekszik. Azért mutattam be ezt, hogy látható legyen mennyire növekvő iparágról van szó.

Az okosotthon piacon az öt legnagyobb bevétellel rendelkező ország a világon			
2017		**2022**	
Amerika	15 110	Amerika	31 450
Kína	3 994	Kína	23 630
Egyesült királyság	3 010	Egyesült királyság	7 850
Németország	2 755	Németország	6 619
Dél-Korea	2 306	Japán	6 521

táblázat 3.1 *Az értékek dollármillióban értelmezendőek.*

3.1-es táblázatban is látható, hogy az okosotthonban világ-első 5 országban csupán 5 év alatt majdnem 3-szorosára nőtt az iparon belüli bevétel.

3.2 Szakmai alapismeretek

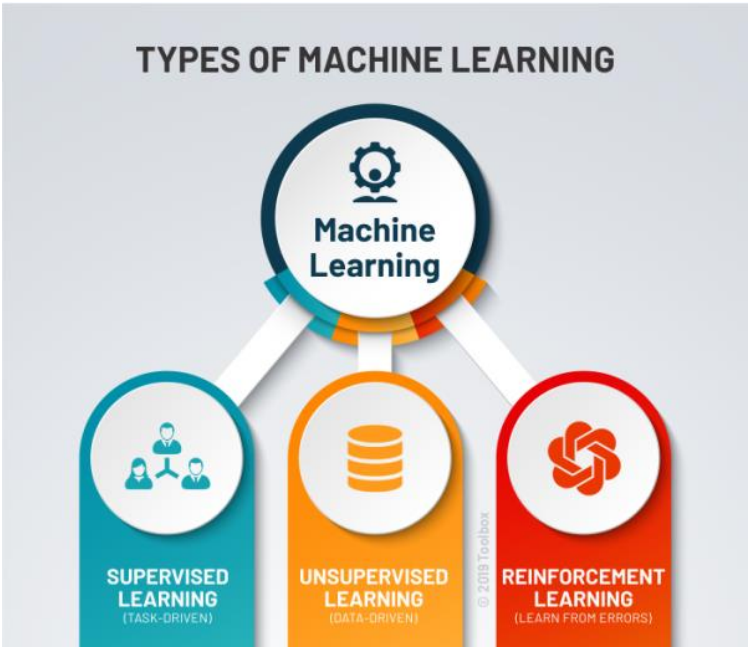
3.2.1 Gépi tanulás

A gépi tanulás a mesterséges intelligenciának az egyik ága. Metódusai, technológiái adatokat dolgoznak fel, majd ezek után fejlesztik a saját pontosságukat, emberi beavatkozás nélkül. A megadott halmazban mintákat, szabályszerűségeket keresnek, annak érdekében, hogy után képesek legyenek új adatokra vonatkozóan is megbecsülni a helyes kimenetet.

A terület nagyban kapcsolódik a számítástechnikai statisztikához, aminek a célja, hogy becslést adjon bizonyos eseményekre számítógépes erőforrások felhasználásával. Azonban a gépi tanulás nem minden ága így működik, van amelyik matematikai optimalizálást használ, vagy akár az emberi agy működését próbálja szimulálni.

A gépi tanulás három fő kategóriára bontható: a felügyelt tanulás, a felügyelett nélküli tanulás és a megerősítésesett tanulás. Míg a fFelügyelt

tanulásnál a tanulás maga már címkézett adatokon folyik, addig a felügyelet nélküli tanulás esetébennál nincsenek címkék. Ilyenkor az algoritmus feladata, hogy megtalálja a kapcsolatot az adatok között és ezek alapján adjon következtetéseket. Megerősített tanulásnál a rendszer a saját hibáiból tanul, mivel ilyenkor csak egy cél van kitűzve, nincsen adat, amin tanulhatna, csak egy adott cselekvés-halmaz, melyből véletlenszerűen választhat. Ebből kifolyólag eleinte sokat hibázik, de egy beépített ügynök segítségével folyamatosan egyre pontosabb lesz. Ennek az ügynöknek a célja, hogy minden választott cselekvést vagy cselekvés-sorozatot egy pontszámmal lásson el, az alapján, hogy az helyesnek bizonyult-e.



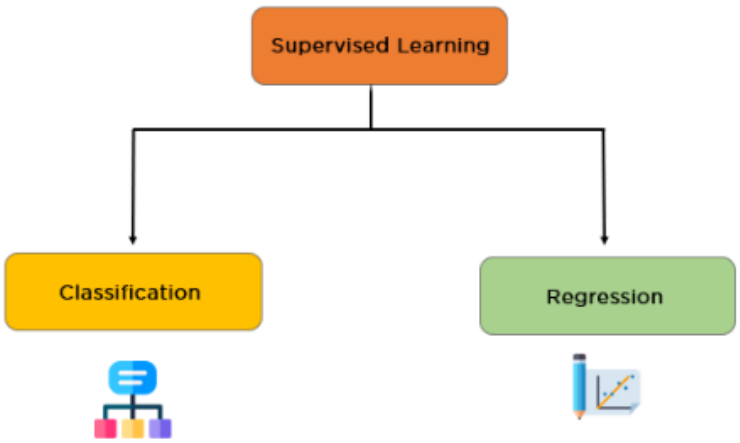
ábra 3.2 *A gépi tanulás három fő típusa*

A gépi tanulásnak rengeteg féle algoritmusa van, melyek a jobb hatás érdekében egymással kombinálhatóak is. Ebben a dolgozatban a felügyelt tanulásra fogok pontosabban kitérni, mivel a projektemben ezt a típusú gépi tanulásos algoritmust használtam.

3.2.2 Felügyelt tanulás

A gépi tanulásnak az egyikfajtája a felügyelt tanulás. Ebben az esetben rendelkezésünkre áll egy bizonyos mennyiségű adat, amiben a bemenetekre már megvannak a kimenetek, címke-példa párokban. Az algoritmus tanulása

arra összpontosul, hogy a bemeneten kapott példákat és a kimenetre adott eredményt összekösse egy függvénnyel. Ez a függvényt az algoritmus minden iterációt követően picit megváltoztatja, az alapján, hogy mennyire jó értéket mért. A felügyelt tanulásnak további kettő fajtáját különítjük el, ezek az regresszió és a klasszifikáció. [2]



ábra 3.3 *A felügyelt tanulás két típusa*

Regresszió használatakor meg akarunk becsülni egy folyamatos értékű kimenetet. A legegyszerűbb és egyik leggyakoribb fajtája a lineáris regresszió, amikor egy függvényt kapunk eredményül, ami a lehető legjobban illeszkedik az adatainkra. Egy jó példa regressziós feladatra, ha egy háznak a tulajdonságai alapján meg akarjuk becsülni az árát.

A másik változat pedig a klasszifikáció. Ebben az esetben a kimeneten egy konkrét osztályt kapunk meg. Ezt akkor használjuk, ha a bemeneten lévő adatokat valamilyen kategóriába szeretnénk besorolni. Ilyen kategorizálási felada lehet például adott ember tulajdonságai alapján annak megállapítása, hogy férfi-e vagy nő, vagy éppen, hogy milyen hajszíne lehet.

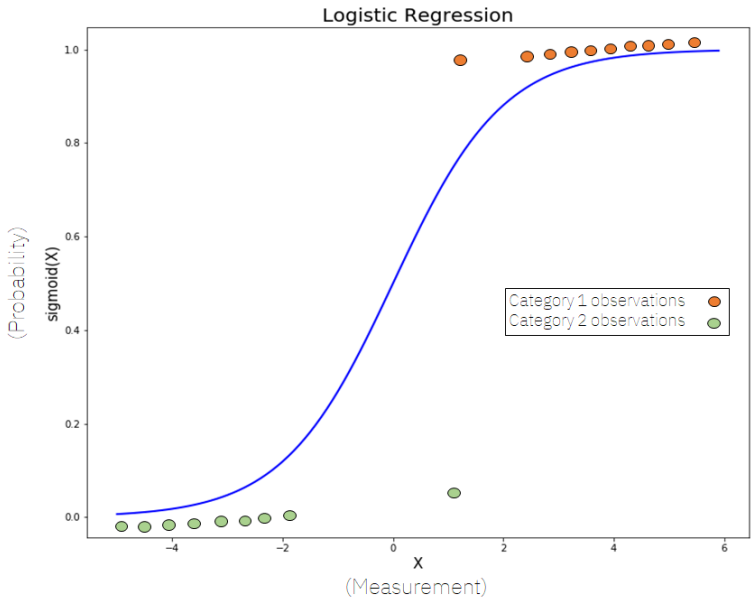
3.2.2.1 Logisztikus Regresszió

A logisztikus regresszió a gépi tanuláson belül az klasszifikáció a problémájával foglalkozik. A függvény maga egy valószínűségen alapuló becslést hajt végre. A modell egy komplex súlyfüggvényt használ, amit szigmoid

függvénynek is neveznek. Ennek a lényege, hogy a függvényt értékét 0 és 1 közé szorítja, annak érdekében, hogy az osztályba sorolást meg tudjuk valósítani. [3]

$$0 \leq h_\theta(x) \leq 1$$

egyenlet 3.1 *Logisztikus regresszió hipotézise*



ábra 3.4 *Logisztikus regresszió példája*

Tehát, ha meg akarjuk becsülni egy bizonyos értékekhez a hozzá tartozó valószínűséget, akkor szükségünk van a szigmoid függvényre. Ez a függvény minden bemenethez hozzárendel egy másik valós számot, ami 0 és 1 között található. Abba az osztályba fog tartozni az adott példa, amelyikhez közelebb van a szám, azaz 0.5 alatt a 0, ettől kezdve, pedig az 1-es osztályba.

Azonban mi van akkor, ha több osztályunk van, mint kettő és nem tudjuk a bináris klasszifikációt használni? Ebben az esetben minden egyes kimeneti osztályhoz készül egy szigmoid függvény. A modell célja az lesz, hogy minden egyes osztályra megbecsülje, hogy az adott minta mekkora esélyjel tartozik bele, és végül az válassza ki a kimenetnek, amely osztálynál ez a becslés a legnagyobb.

$$h_\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

egyenlet 3.2 A szigmoid függvénye a logisztikus regressziónak

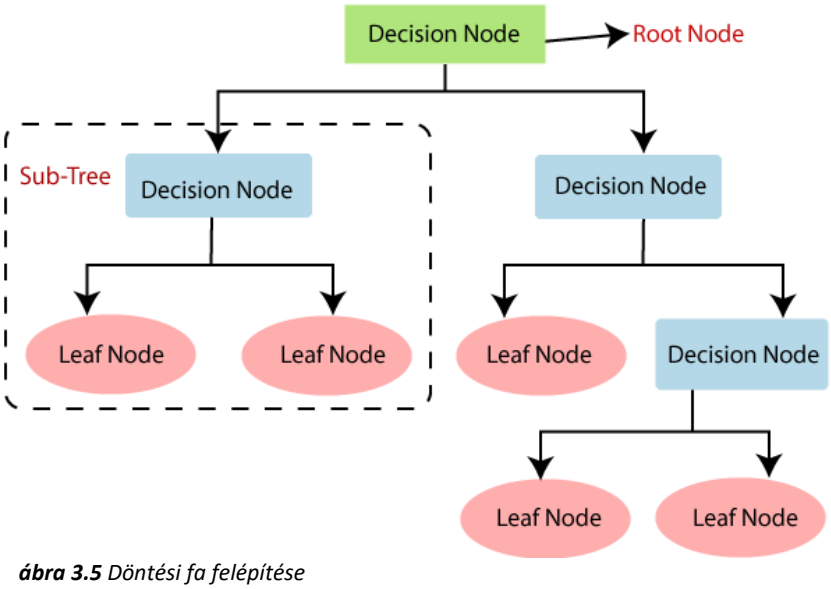
Ez által az algoritmus minden esetben képes lesz visszaadni egy, a bemenetet azonosító osztályt, amikor a beérkező adatot végig futtatjuk egy predikciós függvényen és megkapja az adott osztályokra a valószínűségétt.

3.2.2.2 Döntési fák

A döntési fák valószínűleg a legegyszerűbb algoritmusok a felügyelt tanulási algoritmusok közül. Az adatkészletet egyre kisebb adatcsoportokra bontják, amíg azok le nem lesznek írhatóak egy címkével. [4]

A döntési fák négy részből állnak össze:

- Gyökér csomópontból
- Belső csomópontokból
- Ágakból
- Levél csomópontokból



ábra 3.5 *Döntési fa felépítése*

Az adat ezeken a részeken halad végig a gyökér csomópontból indulva, amíg el nem ér egy levél csomópontot. Minden belső pontnál szétbontják az adatot úgy, hogy amik egy csoportba kerültek a lehető legjobban hasonlítsanak egymásra. Ezek mellet továbbá cél az is, hogy a különböző csoportok a lehető leginkább eltérőek legyenek.

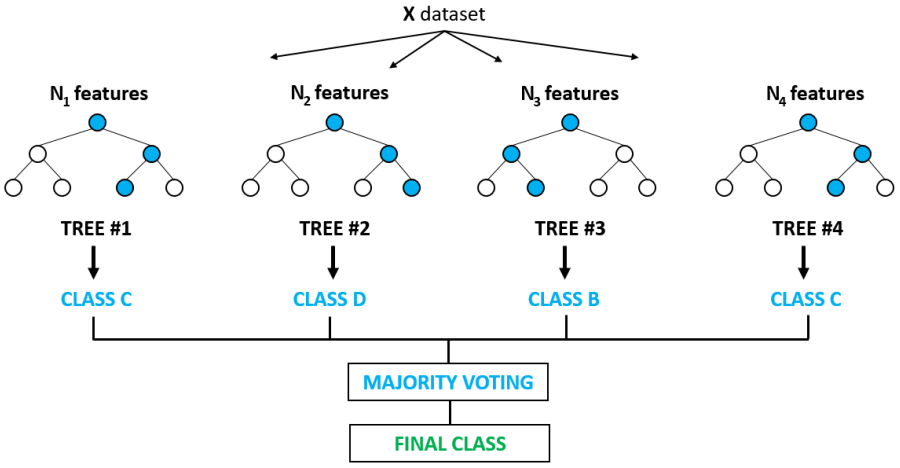
Az algoritmus az egyszerűség, és emberi szemmel is könnye értelmezhetősége miatt nagy népszerűségnek örvend. Könnyen ábrázolható és jól működik mind regressziós, mind klasszifikációs problémáknál.

Ennek ellenére megvannak a hátrányai is. Létrejöhetnek nagyon bonyolult fák is, amik túlságosan illeszkednek a tanulási adatokra. Ilyenkor nem működnek megfelelően új, még ismeretlen adatokra. Továbbá nem működnek jól olyan adathalmazokon sem, amik kiegyensúlyozatlanok, azaz, ha az egyik osztály jobban van reprezentálva, mint a másik. Ekkor mielőtt nekikezdnénk a fa tanításának szükséges az adathalmazt kiegyensúlyozni, vagy ha ez nem lehetséges akkor érdemes egy másik algoritmust használnunk a problémánkhoz.

3.2.2.3 Véletlen Erdők

A véletlen erdők módszer a döntési fákonalapul. Működése során több döntési fát épít, melyek eltérnek egymástól. Egy becslés során minden fán kap egy eredményt, majd klasszifikációs esetben azt választja végső kimenetnek, amit a legtöbb fa adott eredményül, regressziós esetben az átlageredményt továbbítja a kimentre. A legjobban klasszifikációs problémákra lehet használni. [5]

Az algoritmus a többség bölcsességének elvét használja fel, azaz, hogy ha több helyről kapunk vissza egy eredményt, akkor ez biztosabb lesz. Ha az embereket vesszük példának valószínűbb, hogy egy csoport ember jobb megoldásra fog jutni egy feladattal kapcsolatban, mint csak egy ember.



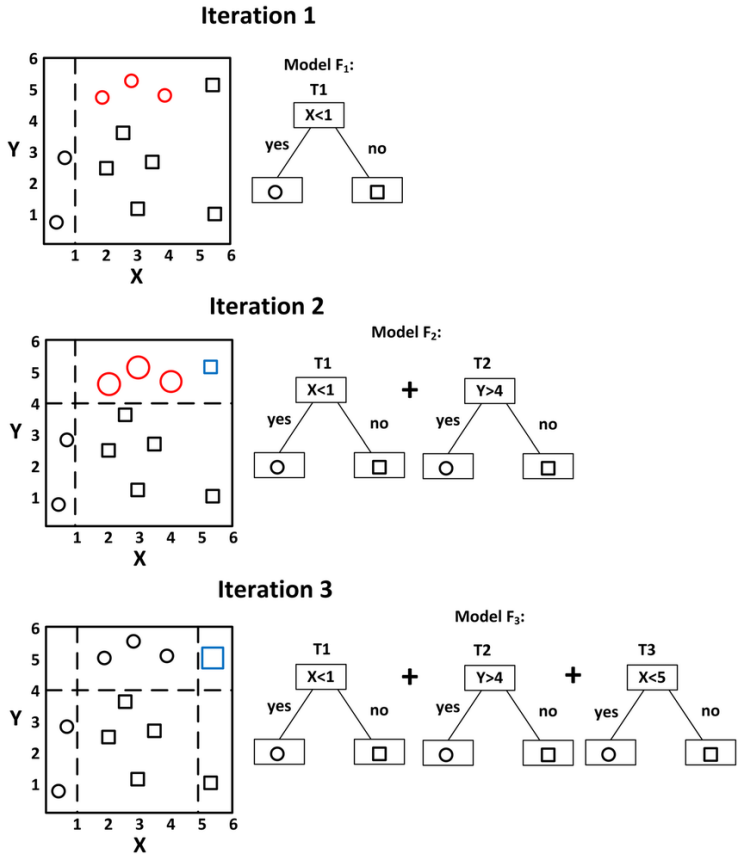
ábra 3.6 *Véletlen erdők modell felépítése*

Nagy előnye a döntési fákhoz képest, hogy képes kezelni kiegyensúlyozatlan adathalmazokat súlyozott fák segítségével. A döntési fák esetében azt figyeli a felosztási módszer, hogy a mintacsoportok mennyire keverednek, azonban figyelmen kívül hagyja a tényt, hogy egyes osztályok sokkal jobban vannak reprezentálva. Ezzel ellentétben a véletlen erdőknél lehetőség van költség érzékeny tanulást folytatni. Ilyenkor csak annyit kell tegyünk, hogy nagyobb büntetést adunk a kevesebb példát tartalmazó osztályoknál, így kiegyensúlyozva a modellünket.

3.2.2.4 Gradiens Erősítés

Az erősítés egy gépi tanulási metódus, amely alatt több, egyszerűbb modellt vonunk össze, hogy a nagy modellünk jobb eredményt adjon. Az egyszerű modelleket ebben a környezetben gyenge tanulóknak nevezzük. Regresszióra és osztályozásra ugyanúgy használható algoritmus. [6]

A modell esetében az erősítés matematikai optimalizálást jelent. Célunk a veszteségfüggvény minimalizálása azzal, hogy további gyenge tanulókat adunk a modellhez. Ezt a tanulási időszak alatt tesszük úgy, hogy a lehető legjobban javítsuk azokat a területeket, amik eddig gyengén teljesítenek.



ábra 3.7 *Gradiens erősítő működése*

Mint mindennek természetesen ennek is megvannak az előnyei és hátrányai. Egyrészről az algoritmus rendkívül flexibilis. Könnyen tudja használni azt is, ha hiányzik adat, valamint elérhető rengeteg fajta veszteségfüggvény vagy hiperparaméter keresés. Egyik legnagyobb előnye viszont abban rejlik, hogy nincs szükség az adatok előfeldolgozására, mivel mind kategorikus mind számszerű adatokkal tud dolgozni.

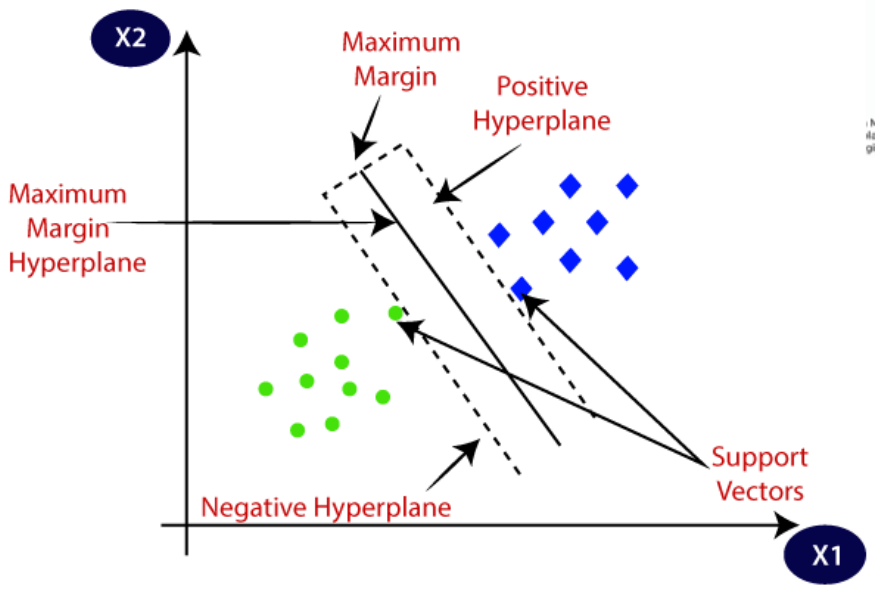
Hátránya viszont, hogy szintén előfordulhat túltanulás. Ennek következtében túl jól fog illeszkedni a tanulásban szereplő adatokra és nem lehet általános adatokkal megfelelően használni. Ezeken kívül nem lehet eltekinteni attól sem, hogy nagy az erőforrás szükségessége.

3.2.2.5 Szupport vektor gép

Az algoritmus célja egy olyan N-dimenziós hipersík létrehozása, ami a lehető legjobban elkülöníti a más-más osztályba tartozó adatpontokat. A maximum

távolság logikáját követve azt a síkot igyekszünk megtalálni, amelyik a lehető legtávolabb van az osztályoktól. [7]

A sík feladata, hogy ennek segítségével el tudjuk különíteni egymástól az osztályokat. Kettő osztály esetén, ha egy pont a sík egyik oldalára esik akkor az egyik csoportba fog tartozni, ha a mások oldalára akkor a másikba.



ábra 3.8 *Szupport vektor gép felépítése*

Amennyiben komplexebb problémákat akarunk megoldani a lineárisnál az algoritmus segítségével, akkor kerneleket használhatunk. Többféle kernel típus létezik, ezek közül pár példa a lineáris, a polinomiális, valamint a Gauss kernel.

A logikai regresszióval összehasonlítva jól látható, hogy a két módszer közel ugyanazt tűzi ki saját céljának. Amíg viszont a szupport vektor módszer megtalálja a lehető legjobb megoldást, addig a másik csupán egy nagyon jót talál. Ilyenkor fel is merül a kérdés, hogy melyiket használjuk.

Ha csak kevés példa áll rendelkezésre, de az adatainknak sok jellemzője van, akkor a legjobb, ha a logikai regressziót vagy a lineáris kernelt használjuk. Ennek ellenére közepes mennyiségű példánál, ahol kevés jellemző található, ott jobban járhatunk a gauss, vagy a polinomiális kernellel. Amikor sok példánk van relatíve kevés tulajdonsággal akkor a legjobb megoldás, ha további

tulajdonsággal látjuk el őket majd ezek után a logikai regressziót vagy a lineáris kernelt használjuk.

3.2.2.6 Neurális hálók

A neurális hálók az emberi agyat próbálják meg utánozni. Ezt úgy teszik, hogy az adatot egy sor algoritmuson küldik keresztül. A módszernek a célja, hogy bonyolultabb problémákat is meg tudjon oldani, mint például a korábban felsorolt, egyszerűbb algoritmusok. [8]

A háló logikai egységekből épül fel, amiket neuronoknak nevezünk. Ezek rétegekbe csoportosulnak, melyeknek három fajtája lehet:

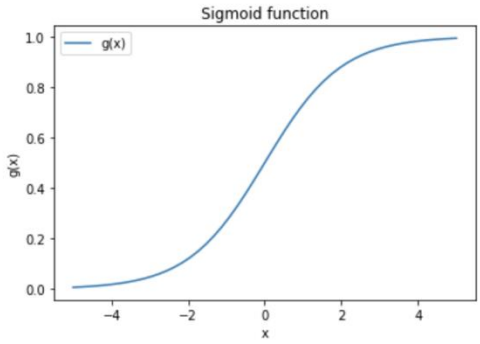
- bemeneti réteg
- rejtett réteg
- kimeneti réteg

Működését a legegyszerűbb példát mutatnám be, ami a három rétegű neurális háló. Ennek van három bemeneti réteg-beli neuronja, egy rejtett réteg-beli neuronja és egy kimeneti réteg-beli neuronja. A bemeneti rétegek megkapják az adatot, amit a rejtett rétegnek továbbítanak. Ezek után, kimeneti rétegen megjelenik a kapott predikció.

A neurális háló rétegeit egymásra pakolva kapunk ezekből egy egyedi hálózatot. Ezek a rétegek tartalmazhatnak különböző aktivációs függvényt vagy matematikai operációt minden egyes neuronban.

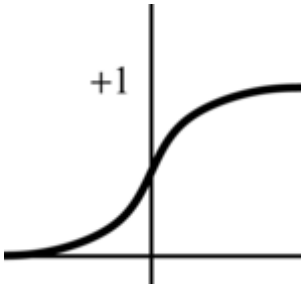
A neuronok kimenetét az aktivációs függvényük határozza meg. Sok elérhető ezek közül, de a legnépszerűbbek a következők:

- Szigmoid



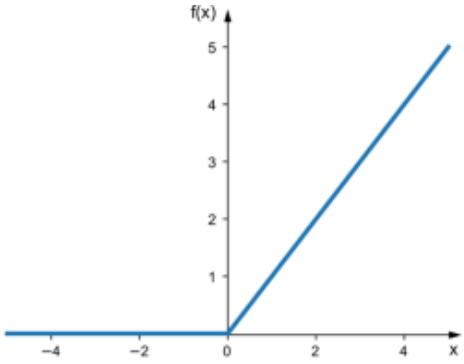
ábra 3.9 *Szigmoid függvény*

- Logisztikus függvény



ábra 3.10 *Logisztikus függvény*

- ReLU



ábra 3.11 *ReLU függvény*

3.2.2.7 CNN

A konvolúciós neurális háló, röviden CNN, a gépi látással összefonódott fogalom lett, ugyanis az algoritmus a bemenetére egy képet vár, amin a különböző aspektusoknak és objektumoknak súlyt ad. Elterjedését nagyban elősegítette, hogy minimális az előfeldolgozási igénye, hiszen bemenete akár egy kép is lehet. [9]

A modell architektúrája az emberi neuronokéhoz hasonló. Különböző neuronok csak különböző régióknak a stimulálására reagálnak. Ezek a területek átfedésben állnak egymással, hogy a kép egészet le tudják fedni.

Különböző filterek segítségével képes a CNN a képnek a térbeli és időbeli függőségeit kiemelni. Ezzel a célja az, hogy a képek méretét lecsökkentse úgy, hogy az algoritmus számára fontos dolgokat képes legyen megtartani. Így is garantálja a megfelelő becslést. Ez azért is fontos, mert nem csak képes felismerni a képeken lévő objektumokat, de nagy adatbázisra is használható.

Három fő rétegből épül fel:

- Konvolúciós, avagy kernel
- Összevonó
- Osztályozó, avagy a teljesen összekötött

A kernel réteg feladata, hogy a bemeneten érkezett képen végig lepkéd egy több pixelt lefedő mátrix segítségével. Ezt a mátrixot filternek nevezzük. Minden lépéskor összeszorozza ezzel a mátrixxal képnek az adott részét így kapva egy számot. Így lépked végig soronként, majd, ha sor végéhez ér lejjebb lép egy oszloppal egészen amíg a kép végéhez nem ér. A rétegnek a célja, hogy a kiemelje a fontos elemeket a képen, mint az objektumok szélei.

Az összevonó réteg nagyban hasonlít az előzőre olyan téren, hogy itt is a térbeli részt szeretnénk csökkenteni. A domináns jellemzőket kiemelve célunk, hogy csökkentsük a szükséges számítási erőforrást.

Az adatok megfelelővé alakítása után a képet egy oszlopvektorrá simítjuk. Ez a kimenet végül belekerül egy neurális hálóba, ahol egy visszacsatolásos modell segítségével megtanulja megkülönböztetni a domináns és az alacsony fontosságú tulajdonságokat. Ezek végeztével képes lesz a képet beosztályozni, amit a kimenetre visszaad.

4 Probléma definiálása

4.1 Bonyolult otthon irányítási felületek

Számos otthon irányítási lehetőség érhető el számunkra jelenleg a piacon, ám a legelterjedtebbek a következőek: falra rögzített okos eszköz, telefonos applikáció és hang alapú irányítás.

Falra rögzített okoseszköz esetében egy, általában tablet lesz felszerelve szobahelységek falaira. Ennek is van több változata. Vannak esetek amikor az eszköz maga mozgatható és használható másra is, viszont megtalálhatóak a piacon olyan megoldások is, amikor az adott helyről nem mozdítható el az eszköz és csupán a ház irányításához szükséges funkciókat képes ellátni, arra van dedikálva. Az ilyen megoldások már kimenőben vannak a piacról, de még mindig megtalálhatóak.

A jelenleg legelterjedtebb megoldás az okosotthonok irányítására a telefonos applikáció. Ilyenkor az esetek többségében egy adott rendszeren belüli eszközöket lehet a rendszerhez készült applikációval összefogni és vezérelni. Ilyen rendszer például az Apple HomeKit.

ábra 4.1 *Home Assistant logója*

Létezik olyan megoldás is, amikor több, különböző gyártótól származó eszközt fogunk össze egy felületen. Ez azért számít nagy dolognak, mivel a kezdetekben, amikor megjelentek a különböző gyártóknak az eszközei, azok nem funkcionáltak együtt más gyártók eszközeivel. Erre a problémára kínált megoldást a Home Assistant, ami egy nagyon személyre szabható felület és nagy mélységekig lehet eszközöket hozzárendelni. 2019 végén viszont az Amazon és a Google, valamint más, a témában érdekelt, nagy cégek bejelentették, hogy elindítják a Connected Home over IP (CHIP) projektet. Az összefogás célja az volt, hogy legyen egy egységesített protokoll, amivel minden eszköz tud majd kommunikálni egymással. Ebből az összefogásból alakult meg 2021-ben a matter protokoll.

ábra 4.2 *Matter protokoll logója*

A hang alapú irányítás legtöbbször, mint kiegészítő lehetőségként szerepel az okos otthon hálózatokban. A legelterjedtebb megoldás Amazonnak az Alexája, ami sok mindennel képes együtt működni.

4.2 Jelenlegi megoldások problémái

4.2.1 Falra rögzített okoseszköz

Mivel az egyik legelső implementációs megoldás ez volt, így nem csoda, hogy hagyott még kívánságokat maga után. A legnagyobb problémája, hogy a ház irányítására nincs akárhonnan lehetőség. A felhasználói élményt nagyban rombolja az, hogy például egy lámpa kapcsolásért szintén fel kell kelni az ágyból, mintha a hagyományos kapcsolót használnánk.

4.2.2 Telefonos irányítás

A telefonos applikációk kiküszöbölik a nagy hátrányát a falra rögzített eszközöknek, de még ezek sem nyújtanak tökéletes megoldást. Ugyan most

már bárhonnan és bármikor tudunk otthonunknak parancsot kiadni, ám még mindig szükségünk van egy plusz eszközre ahhoz, hogy ezeket véghez vigyük. Arról nem is beszélve, hogy van lehetőség arra is, hogy ez az eszköz lemerül, vagy nincs nálunk.

4.2.3 Hang alapú irányítás

A jelenleg használt talán legjobb megoldás, de kivetnivalót még mindig bőven hagy maga után. Az előző pontokban szedett hibákat mindet kiküszöböli, ugyanis bárhonnan bármikor plusz eszköz nélkül képesek vagyunk parancsokat kiadni.

Ennek ellenére új problémákkal állunk szemben. Probléma az, hogy ha bármi szól hangosan, akkor nem működik megfelelően a szolgáltatás. Legyen szó filmezésről, főzésről vagy családi összejövetelről a hang alapú szolgáltatás kiesik. Újdonsült probléma az is, hogy egy hangokat lejátszó alkalmazással egyszerűen tudunk jogtalanul parancsokat is kiadni.

4.3 Javasolt megoldások

A fent felsoroltokat figyelembe véve egy olyan megoldást kerestem, amivel szemben nem hozhatóak fel ilyen jellegű problémák. Az irányító rendszernek olyan követelményeket szabtam meg, amivel biztosíthatom a maximális felhasználói élményt. Ilyen követelmények a következők voltak: bárhonnan, bármikor adható ki parancs az otthonnak, nincs szükség plusz fizikai eszközre a parancs kiadásához, a zavaró tényezők kiküszöbölhetőek legyenek, használata egyszerű legyen és kényelmes, valamint beépíthetőek legyenek biztonsági funkciók is

4.3.1 Kézjelzés alapú irányítás

A kézjelzéseket az öt ujjunk és tenyerünk különböző módon való tartásával tudjuk elérni. Ezek a kéztartások lehetnek annyira egyszerűek is, mint a teljesen nyitott vagy csukott tenyér.

Léteznek viszonylag univerzális kézjelek. Azokat értem ez alatt, amiket a világ bármelyik részén nagy valószínűséggel megértenek, mint például a like vagy a telefon jel. Ennek ellenére, dolgozatomban olyan kézjeleket választottam, amikre találtam megfelelő mennyiségű és minőségű adatbázist a CNN algoritmushoz. A cél ezzel az volt, hogy jó össze tudjam hasonlítani a választott algoritmusokat az online forrásokból szerzett, valamint a saját magam által generált adatokon való tanítás esetében is.

4.3.1.1 Előnyök

A rendszer kiküszöböli a jelenlegi technológiák egyik legnagyobb hátrányát, azzal, hogy bárhol és bármikor használható. Nem eshetünk olyan hibába, hogy ne legyen nálunk az eszköz, amivel a parancsokat adjuk ki, ugyanis csak egy kamerára van szükség. Ezen kívül szintén nem fordulhat elő olyan időszak, amikor ne tudnánk kiadni parancsokat.

Maguknak a kézjeleknek egy nagyon fontos előnye, hogy közismertek és egyszerűek. Ezáltal nagyban meggyorsítja a felhasználóknak azt a folyamatát, hogy megtanulják kezelni, használni a rendszert. Nincsen bonyolult felület, vagy, leginkább az idősek számára, átláthatatlan kezelőpult.

További nagy előnye lehet a kézjeleknek, hogy viszonylag univerzálisak. A világ bármelyik pontján találkozhatunk hasonlóakkal., így nagyban megkönnyül a fejlesztési fázis.

Nagy segítséget jelenthet az is, hogy nem muszáj, hogy pontosan mutassuk a parancsokat a rendszernek. A hétköznapi működésben ez nagyon sokat tud segíteni, ugyanis nagyon kényelmesen tudjuk kiadni a jeleket.

4.3.1.2 Hátrányok

Ugyebár, minden jó dolognak van hátránya, így felmerülhet a kérdés, hogy a kézzel történő irányításnak is van-e. Alapos átgondolás után eszünkbe jut pár dolog. A kézjelek feldolgozásához szükségünk van arra, hogy láthatóak legyenek a kezeink. Ez sötétben jelenthet kihívást, ugyanis, ha nincsen elegendő fény, akkor az OpenCV könyvtár nem képes azokat felismerni.

Ezen kívül még két további probléma merült fel, melyek közül az egyik az, hogy a kézmozdulat is könnyen leutánozható. Ha valaki látta már egyszer a mozdulatot valószínűleg le tudná utánozni, ugyanis nem bonyolult mozdulatokról lenne szó. Így akár más is tudná irányítani az eszközünket.

Az előző példához szorosan kapcsolódik, hogy mivel nem bonyolult dolgokról van szó így véletlen is előfordulhat a parancsnak a kiadása. Mondjuk egy családi összejövetelnél, vagy nagy magyarázások közepette is előfordulhat ilyen.

4.3.1.3 Hátrányok megoldása

Ebben a pontban azt szeretném bebizonyítani, hogy ámbár vannak problémák jelenleg a szoftverrel, azokra van is már megoldás. Az éjszakai kamerák már egészen közismert eszközök. Ezeknek a segítségével el lehetne érni, hogy a kézjeleket akár teljes sötétségben is fel tudjuk ismerni.

A leutánozhatóságra egy szoftveres megoldást szeretnék nyújtani. Egy a piacon már megtalálható eszközhöz hasonlóan, ahol a hangszínből állapítja meg, hogy ki adja ki a parancsot, jelen helyzetben arcfelismerés alapján lehetne ugyan ezt megoldani. Ehhez bizonyos arcokhoz jogköröket is lehetne rendelni, hogy például a borpincéhez csak a szülőknek legyen joga bemenni.

Ismét egy más környezetben már bevált módszert felhasználva előzném meg a véletlen parancs kiadást. Ezt úgy oldanám meg, hogy a parancs érzékelése előtt szükséges lenne egy indikátor jelzést kiadni, amire felfigyelne a rendszer. Csak ez az indikátor jelzés után lehetne bármilyen parancsot kiadni. Ahhoz, hogy ez jól működjön az indikátor jelnek egy olyat választanék, amely nem bonyolult, de a hétköznapokban nem gyakran fordul elő egy átlagos háztartásban.

4.3.1.4 Jövőbeli fejleszthetőség

A dolgozatban fejlesztett szoftver további, a TDK keretein túlmutató fejlesztésekre szeretnék ebben a részben kitérni. Mivel a cél egy olyan program

elkészítése lenne, ami maximalizálja a felhasználói élményt, így ezt a szempontot tartottam előnyben.

Egyik ilyen fejlesztési lehetőség a parancsok teljes személyre szabhatósága lenne. Mivel jelenleg, a megfelelő adatbázis szükségessége miatt, jelnyelvben használt kézjeleket használok. Ez a gyakori előfordulás miatt nem biztos, hogy megfelelő lenne egy olyan családnak, ahol halláskorlátozottak is tartózkodnak. Valamint, ha adott környezetnek jobb ötlete van a bizonyos jelekre, amit maguk között egyszerűbben visznek véghez, akár sérülés miatt, vagy egyéb okokból. Akkor azokat a jeleket is be lehet állítani.
Ez a funkció nem csak azt segíti elő, hogy egyszerűbb legyen kiadni és megtanulni a jeleket. Hanem azt is, hogy azok ne legyenek olyan közismertek, ezzel biztonságosabbá téve a rendszert.

Az előző pontban már felvetett arcfelismerés és jogkör kiosztás is egy lehetséges továbbfejlesztés. Ezzel megelőzhető, hogy például a kisgyerekek ne rongálják össze a dolgozószobát, vagy ne tudják kinyitni a törékeny dolgokat tartalmazó szekrényeket. Ezen kívül számos segítséget nyújthat biztonsági területeken. Legyen szó a vendégekről, hogy ne tudjanak ők sem csak úgy bemenni helyekre, vagy arról, hogy a rendszer jelezzen, ha egy eddig idegen ember a házunk területéhez közel járkál.

Talán a képfeldolgozás segítségének az egyik legnagyobb előnye a helyzetmeghatározás lehetősége. A más jelenleg is használt könyvtárban, az OpenCV-ben van lehetőség kettő kamera segítségével térbeli helyet meghatározni. Ez több okból is nagy segítség lehet. Egy ilyen példa a lámpa felkapcsolása, ugyanis, ha éppen az íróasztalnál ülve egy könyvet olvasunk, akkor nem feltétlen szeretnénk azt, hogy a nappaliban az összes lámpa legyen felkapcsolva, csak az amelyik az asztalon található.
Ugyan ez eljátszható azzal, hogyha zenét hallgatunk, akkor is elég az éppen aktuálisan legközelebb található hangfalból szólnia a hangnak.

Jelenleg ezek a legjobb fejlesztési lehetőségei a rendszernek. Természetesen ez a folyamatos újdonságoknak és fejlesztéseknek köszönhetően akár a közeljövőben változhat is. De ha lesz is valamekkora változás ezek akkor is mérföldkövet jelenthetnek az okosotthonok fejlődésének a történetében.

5 Felhasznált adatok és keretrendszerek

A következő fejezetben bemutatnám, hogy milyen adathalmazok és külsős keretrendszerek segítségével végeztem el a kutatásomat.

5.1 Adathalmazok

Az alább leírt adathalmazokon tanítottam és teszteltem az algoritmusokat. Figyeltem arra is, hogy a saját készítésű adathalmazon ugyan azokat a kézjeleket mutassam, mint amelyiket a CNN-hez használtam.

Jelenleg kis adathalmazok voltak csak használhatóak az erőforrások szűkössége miatt. A kézjelek hétköznapi előfordulását nem lehetett teljes mértékben szem előtt tartani, mivel figyelni kellett arra is, hogy miből létezik már megfelelő adathalmaz a képekből.

5.1.1 Osztályozókhoz használt adathalmaz

Az előző pontban felhozott szempont alapján sajátkezűleg készítettem ezt az adathalmazt. Erre azért volt szükség, mivel az interneten nem volt elérhető olyan adatállomány, ahol ugyan azok a kézjelek voltak elérhetőek vektorizált állapotban. E mellett olyan adatokon akartam végezni a tanítást, amelyikkel az OpenCV is dolgozik, hogy a rendszeren belül ne legyen szükséges az adatok átalakítása.

Mivel az osztályozók vektor alakú adatokkal tudnak jól működni, ezért dolgoztam vektorizálttal. Ezt úgy oldottam meg, hogy az OpenCV által érzékelt huszonegy kézpontnak vettem a képen elhelyezkedő x és y koordinátáját. Ezeket a koordinátákat összegyűjtöttem, majd egy adatként kezeltem. Így összesen negyvenkettő szám tartozott egy kézjel egyik példájához.

Összesen hat különböző kézjellel dolgoztam. Igyekeztem ezeket úgy kialakítani, hogy körülbelül hasonló mértékben legyen adat minden kézjelhez.

Valamint ne térjen el sokkal attól sem, ahány kép adat volt a CNN algoritmushoz.

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	39	40	41	Label
0	468	261	431	244	404	214	399	179	418	157	...	223	497	183	480	166	478	189	481	210	0
1	466	262	429	248	401	217	396	184	412	162	...	225	492	186	476	167	475	190	478	211	0
2	464	262	428	248	401	216	397	184	413	163	...	225	494	187	477	168	475	190	478	211	0
3	463	262	428	249	401	217	396	185	411	162	...	226	493	186	477	168	475	191	478	213	0
4	459	262	424	247	396	216	391	184	407	162	...	226	488	186	472	168	471	192	474	214	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11652	196	238	225	224	243	196	237	170	223	155	...	184	182	176	193	162	201	179	204	195	5
11653	190	237	220	225	238	195	232	167	217	150	...	186	176	176	187	162	196	180	199	196	5
11654	181	238	212	226	230	196	225	167	208	151	...	188	165	176	176	162	185	181	188	197	5
11655	178	238	208	228	227	197	222	168	207	152	...	188	162	177	173	165	181	184	184	199	5
11656	177	241	207	230	225	199	220	170	205	152	...	189	160	179	171	166	179	185	181	200	5

ábra 5.1 *Az osztályozókhoz használt adathalmaz eleje és vége*

5.1.2 CNN-hez használt adathalmaz

CNN algoritmus képek feldolgozására lett kifejlesztve, ezért volt szükségem egy megfelelő adathalmazra kézjelekből. Erre a célra Muhammad Khalid Sign Language for Alphabets gyűjteménye volt a legalkalmasabb, amit a CC0ás liszensz alatt tett elérhetővé. Benne az angol abc összes betűjéhez található kézjel.

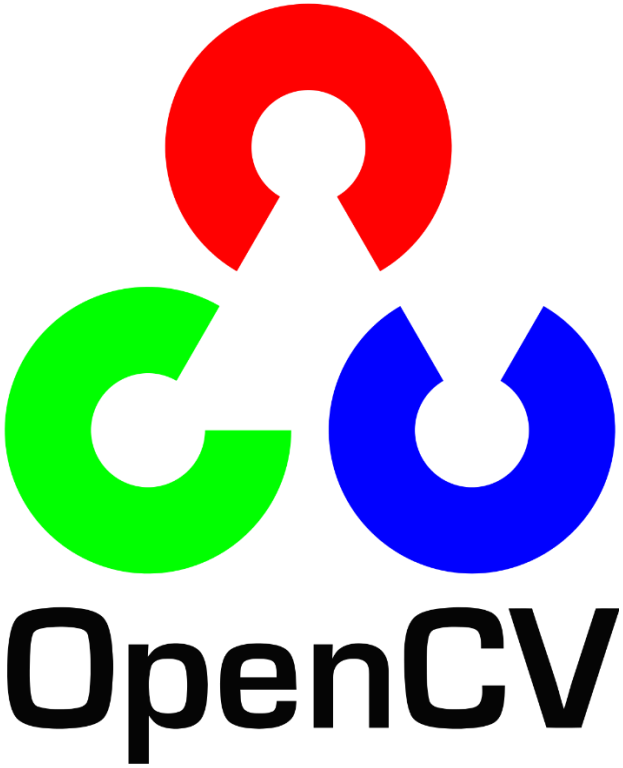
Kis mérete miatt a szűkös erőforrások is képesek voltak megbirkózni vele. Ez köszönhető volt annak, hogy a képek változó, de kis méretűek voltak. Ezzel szemben teljesen felismerhetőek és sokszínűek voltak a kézjelek, magas a halmaz minősége. Ezen kívül azért is volt alkalmas, mivel szürkítettek a képek. Ez azért fontos, mert így nem fog tudni az algoritmus véletlen olyat megtanulni, hogy számítson számára a bőrszín. Egy kézjelből ezerötszáz kép található meg, ezt már elégnek gondolom arra, hogy tudjon rajta az algoritmus tanulni.

ábra 5.2 *A CNN algoritmushoz használt adathalmazból egy kiválasztott kép*

5.2 Keretrendszerek

5.2.1 OpenCV

Az OpenCV egy 2000-ben megjelent képkezelő- és feldolgozó függvényeket tartalmazó könyvtár. Eredetileg Willow Garage és az Itseez fejlesztette az Intelnél, annak egy orosz részlegénél. Céljaik között volt a látáskutatás fejlesztése azzal, hogy a nyílt forráskód mellé az optimalizálva is lesz, egy könnyen olvasható és szállítható legyen, amire egyszerűen tudnak a fejlesztők fejleszteni, valamint a gépi látás alapú kereskedelmi alkalmazásokat szerették volna fejleszteni egy licenccel, amivel a kód ingyenesen használható legyen, ez a BSD licence.



ábra 5.1 *Az OpenCV logója*

A fő fejlesztési nyelve a C++, de eleinte C nyelven íródott, ami bár kevésbé átfogó és kitejedet de még mindig létező felület. Ezeken kívül elérhető hozzá API Java, Python és MATLAB/OCTAVE nyelveken.

Nagykörben elterjed a könyvtárnak a használata, manapság sok területen használják. A teljesség igénye nélkül ilyen területek az arcfelismerő rendszer,

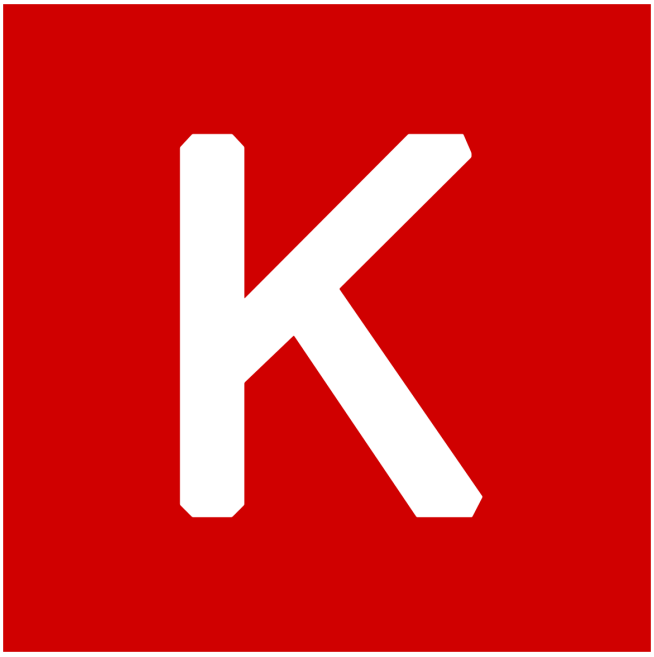
gesztus felismerés, ember-gépi interakció, sztereopszis vagy az objektum detektálás.

5.2.2 Keras és TensorFlow

A Keras egy nyílt forráskódú szoftver amelyik Python nyelvre szolgáltat egy interfészt mesterséges neurális hálók készítésére. Tengernyi implementációt tartalmaz a legelterjedtebben használt neurális hálók építő elemeiből, többek között optimalizálókból vagy aktiválási funkciókból. Ezeken kívül támogatja még a visszatérő és a konvolúciós neurális hálókat is. Mindezt a TensorFlow könyvtárra épülve teszi meg.

ábra 5.2 *TensorFlow logója*

A TensorFlow az egyik legelterjedtebb ingyenes és nyílt forráskódú könyvtár gépi tanuláshoz és mesterséges intelligenciához. Közfelhasználásra 2019 szeptemberében jelent meg, de a Google Brain Team által fejlesztett könyvtárat a Google belsős rendszereiben már 2015 óta használták kutatásra és fejlesztésre.



ábra 5.3 *A Keras logója*

Több elterjedt programozási nyelvben is lehet használni, a legelterjedtebbek a C++, Java, JavaScript és a Python. Mivel nagyon flexibilis ezért sok szektorban használják, ilyenek például a tanítás, közösségi médiák vagy az orvostudomány.

5.2.3 Scikit-learn

Korábban scikits.learn vagy máshogy sklearn-nek is ismert könyvtár Python nyelvhez készült David Cournapeau által. A Google Summer of Code keretein belül elinduló projektben sikeresen elkészült egy ingyenes szoftver, amiben gépi tanuláshoz szükséges algoritmusok nagy része megtalálható. Ilyenek például az SVM, k-means vagy a DBSCAN, mindezt a NumPy and SciPy tudományos és matektudományi könyvtárak segítségével végzi.

ábra 5.4 *Scikit-learn logója*

6 Eredmények

Több körben végeztem el a tesztelést a könyvtárak segítségével.

6.1 Modellek által kapott eredmények

A tanítás és tesztelés után a cél a lehető legjobb eredmény elérése volt.

Adathalmaz	*Modell*	*Teszt Eredmény*
Saját adathalmaz	*Logikai Regresszió*	*100.00%*
Saját adathalmaz	*Véletlen Erdők Osztályozó*	*99.91%*
Saját adathalmaz	*Gradiens Erősítő Osztályozó*	*99.87%*
Saját adathalmaz	*Szupport Vektor Gép*	*99.53%*
Muhammad Khalid - Sign Language for Alphabets	*CNN*	*Tanítás 100.00%* *Validáció 83.41%*

***táblázat 6.1** A kapott eredmények*

6.2 Eredmények összehasonlítása

A fenti táblázatban jól látható, hogy messze a legrosszabb eredményt a CNN érte el a validáció során. Ennek több indoka is lehet. Mivel nem voltak egységesítve az adatgyűjtemények így azoknak az eltérése is befolyásolhatta az eredményeket, valamint a CNN is tovább optimalizálható lenne, ha regularizációt alkalmaznánk.

7 Konklúzió

A dolgozatom célja az volt, hogy az okosotthon irányítást a lehető legkényelmesebb és jól működő módon meg bírjam oldani. Ehhez körül jártam az okosotthonok fejlődésének a történetét, hogy milyen irányzatokban történtek a fejlesztések a parancskiadást tekintve. Arra jutottam, hogy az idő haladtával a leginkább arra kezdtek koncentrálni, hogy a legkényelmesebb megoldást megtalálják plusz eszközök használata nélkül.

Az okosotthon piacot felkutatva arra jutottam, hogy jelenleg az egyetlen elérhető megoldás az, hogy hang alapon tudjuk a parancsokat kiadni. Ennek több problémája is van. Ezek közé tartozik a hangzavar, vagy az egyszerű feltörése a rendszernek.

Mivel a világon már vannak olyan helyszínek, ahol megfelelően működik arcfelismerés, vagy süketnéma kézjel felismerés úgy gondoltam ez egy jó irányt adhat a kutatásnak. A kézjelek ugyanis minden gondot megoldanak azzal, ami az eddigi okosotthon irányítási lehetőségekkel felmerült. Bárhonnan kiadható, nincs szükség plusz fizikai eszközre, egyszerűen irányítható, nehezen feltörhető, könnyen megtanulható és kevés akadályozó tényező létezik.

Ezek után a célom az volt, hogy megtaláljam a legmegfelelőbb modellt a kézjelek feldolgozására. Ezt úgy kezdtem el, hogy körbe néztem milyen a probléma jellege. Arra jutottam, hogy a legtöbb eljárás, ami hasonló irányban történt mind a CNN algoritmus segítségével volt elvégezve. Viszont a probléma jellege miatt kerestem egyéb, használható algoritmusokat. Úgy döntöttem, hogy osztályozó algoritmusok lesznek még számomra megfelelőek, de kontroll egységként egy regressziós algoritmussal is megpróbáltam a kutatást elvégezni. A dolgozatban használt algoritmusok mellett azért döntöttem, mivel mindegyik valamilyen okból kiemelkedő, vagy jelentős, elterjedt modell.

Legvégül a kiválasztott modelleket leteszteltem egy megfelelő adatbázis segítségével. Az eredmények azt mutatták, hogy a legrosszabb eredményt a CNN adta, ahhoz képest, hogy eddig a süketnéma kézjelek feldolgozására ez volt a legelterjedtebb megoldás. Jelen tesztek során a legjobb eredményt a logikai regresszió nyújtotta 100%-os pontossággal. Ez azt jelenti, hogy minden parancsot a teszt során megfelelően felismert. Ezzel meg sikerült találni a lehető legjobb megoldást a kézjelek felismerésére.

A jövőben magát az okosotthon rendszert még lehet majd plusz funkciókkal kiegészíteni, hogy a kézjel felismerés biztosan a lehető legjobb megoldás legyen. Ezek az extra funkciók az éjjel látó kamera beszerelése, hogy képes legyen teljes sötétségben is érzékelni a parancsokat. Az arcfelismerő rendszer hozzárendelése, erre azért van szükség, hogy biztonságossá tehető legyen a rendszer. Ugyanis ilyenkor jogokat lehet rendelni bizonyos emberekhez. A véletlen parancskiadások elkerülése érdekében meg be lehet vezetni egy kevésbé hétköznapi indikátor jelzést, ami elindítaná magát a parancs érzékelésének a folyamatát.

8 Irodalomjegyzék

[1] „smarthomepoint,” 31 október 2022. [Online]. Available: https://www.smarthomepoint.com/history/.

[2] „towardsdatascience,” 31 október 2022. [Online]. Available: https://towardsdatascience.com/what-is-supervised-learning-fd86d704d855.

[3] „analyticsvidhya,” 31 október 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/.

[4] „towardsdatascience,” 31 október 2022. [Online]. Available: https://towardsdatascience.com/the-complete-guide-to-decision-trees-28a4e3c7be14.

[5] „analyticsvidhya,” 31 október 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[6] „analyticsvidhya,” 31 október 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/gradient-boosting-machine-for-data-scientists/.

[7] „pub.towardsai.net,” 31 október 2022. [Online]. Available: https://pub.towardsai.net/support-vector-machine-svm-introduction-machine-learning-8c56b7da63f1.

[8] „analyticsvidhya,” 31 október 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/understanding-the-basics-of-artificial-neural-network-ann/.

[9] „towardsdatascience,” 31 október 2022. [Online]. Available: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529 .

[10] [Online].