

TDK-dolgozat

Szerző

Szőcs Barna

D.A.C. - Draw and Calc

- The Intuitive Calculator -

Szerző:

Szőcs Barna

Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika kar, Informatika szak, alapképzés, III. év

Témavezetők:

dr. Robu Judit egyetemi docens,

Babeş–Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika kar, Informatika tanszék

dr. Vajda Szilárd tudományos kutató,

TU Dortmund - Department of Electrical Engineering and Information Technology

Kivonat:

A hirtelen megnövekedett számú érintőképernyős telefonok, melyek mellőzik a hagyományos billentyűzet használatát, nagy igényt teremtettek az intuitív adatbeviteli formákra, mint pl.: az írásfelismerés. A dolgozatban egy olyan rendszer kerül bemutatásra, amely egy nap mint nap használt programot, a számológépet vértézi fel az írott számjegy felismerésével, megszüntetve így a hagyományos billentyűzetről való gépelést. Egy olyan mobil rendszer került kifejlesztésre, amely ötvözi az off-line és on-line karakterfelismerők előnyeit, úgy, hogy ez egy átlagos Android-alapú telefon számítási korlátait sem haladja meg. A rendszer nem csak a kézírás felismerését teszi lehetővé, hanem lehetőséget nyújt arra is, hogy ez utóbbi automatikusan igazodjon az éppen aktív felhasználó kézírási sajátosságaihoz, ezáltal létrehozva egy valódi ember - számítógép interakciót. A dolgozat külön részletezi ezen rendszer matematikai alapjait, szoftveres megvalósításait, és különös figyelmet fordít az eredmények bemutatására egy általunk valós élethelyzetekben gyűjtött kézírás-adathalmazon.

Kulcsszavak: gépi tanulás, kézírásfelismerés, neurális hálók, kifejezés kiértékelés, szerkezeti primitívek, ember-eszköz interakció.

Tartalomjegyzék

1 Bevezető.....	1
2 Előzetes munkák.....	2
2.1 Általánosságok	2
2.1.1 Off-line karakterfelismerők	3
2.1.2 On-line karakterfelismerők.....	3
2.2 Neurális hálók.....	4
2.2.1 A tanulási folyamat.....	5
2.3 Sablon alapú osztályozó	9
2.3.1 DTW algoritmus	10
3 A felhasznált technológiák	11
3.1 Android.....	11
4 Rendszer Felépítés	12
5 Eredmények	17
5.1 Az adatok.....	17
5.2 Eredmények ismertetése	17
6 Következtetések	20

1 Bevezető

A hirtelen megnövekedett számú érintőképernyős eszközök, melyek mellőzik egy billentyűzet használatát, nagy igényt teremtettek az intuitív adatbeviteli formákra. Ezáltal nagy szükséget teremtve újabb megoldások keresésére, kifejlesztésére, alkalmazására. Ezen eszközök fejlődési gyorsasága, úgy a sebesség mint a felbontás és felhasználó-barátság területén, lehetőséget teremtettek olyan alkalmazások fejlesztésére, ha csak részben is, melyeknek kivitelezése régebb elképzelhetetlen lett volna.



1. ábra – Bal felől egy Android alapú Samsung telefon, jobbra egy iOS rendszert futtató Apple Tablet

Az írott számjegyfelismerés merőben megkönnyíti egy, csakis érintőfelülettel rendelkező eszköz használatát, ugyanis a felhasználónak nem egy virtuális billentyűzetről kell bevennie a karaktereket, hanem a megszokott módon, mintha csak egy darab papírra írná azokat. Egy ilyen hétköznapi alkalmazás a számológép, mely kiegészítve egy rajzvászonnal megadja a lehetőségét egy intuitív rajzoláson keresztül matematikai kifejezés bevitelnek, ezáltal növelve az eszköz felhasználási élményét. Az Android operációs rendszerbe beépített számológép alkalmazás nem használja ki az érintőképernyő által nyújtott előnyöket. Összezsúfolva a virtuális billentyűzet elemeit, kis hely marad csupán a válaszok megjelenítésére, ugyanakkor bővíthetőség szempontjából is behatárolt. Feltételezve, hogy ezeket az eszközöket mindig ugyanaz a személy használja, lehetőség adódik az adott felhasználó személyes írásmódjának kiismerésére, az alkalmazás használat közbeni javulására, fejlődésére, mindezt úgy, hogy nem szükséges előzőleg bármilyen mintapéldány bekérése.

2 Előzetes munkák

Egy általános kézírás-felismerő rendszernek komplex mintákat kell képesnek lennie megtanulni, és később ezeket minél nagyobb hatékonysággal felismerni. Ahhoz, hogy ez lehetséges legyen, több matematikai alapokon nyugvó rendszert is kidolgoztak az elmúlt pár évtizedben. Ezek kerülnek a következőben bemutatásra.

2.1 Általánosságok

Az OCR (Optical Character Recognition) rendszerek megoldottnak tekintik a nyomtatott tagolt írás felismerését, nagyobb, mint 99,9 %-os pontosságot elérve [7]. Az egyes nyelvek, melyeknél nagyon magas a szavak száma és a különböző nyelvű kézírások felismerése azonban, még mindig aktív kutatási területnek számít [6].

Kézírásfelismerőnek nevezünk egy olyan rendszert, amely bemenetként egy képet vagy egy pontsorozatot kap (előbbi off-line, utóbbi on-line karakterfelismerő), a kimenete pedig egy konkrét karakter ASCII reprezentációja. A kézírásnak két formája lehetséges:

- összefolyó, folytonos írás
- elválasztott, tagolt írás

Mivel ez a dolgozat egyszerű matematika kifejezések kiértékelésével foglalkozik, az elválasztott, tagolt írás kerül előtérbe. Ahhoz, hogy egy rendszer automatikusan és helyesen felismerjen egy adott karaktert, nagy jelentősége van minél több és változatosabb minta-egyed begyűjtésének. Egy helyesen működő kezírásfelismerőnek a következő problémákat kell megoldania:

- különböző emberek különböző írásstílusait értelmezni
- az eltérő tulajdonságú eszközökön való használatot

Az első problémára egy kezdeti megoldás, egy minél nagyobb tanuló halmaz, amely stílusban eltérő mintaadatokat tartalmaz. Egy másik megoldás a rendszer alkalmazkodása a felhasználóhoz, amelyet az 4. fejezet bővebben tárgyal. A második probléma esetünkben a következő problémákra egyszerűsödik:

- az eszköz erőforrásainak behatároltsága
- az eszköz érintőfelületének felbontása és frissítési aránya

A 4. fejezetben tárgyalásra kerül a felhasznált és beépített rendszerek hatékonysága. A felbontásbeli és frissítési, mintavételezési különbözőségeket normalizálási eljárások küszöbölik ki (részletek a 4.dik fejezetben).

2.1.1 Off-line karakterfelismerők

Egy off-line karakterfelismerő rendszer adott képpont halmaz alapján becsüli meg, milyen karaktert is ábrázolhat a bizonyos kép. Ez a technológia általában akkor használatos, mikor már egy meglévő, leírt szöveg kerül feldolgozásra, amelyről semmilyen típusú mellékinformáció (az írás iránya, sebessége, stílusa, stb.) nem áll rendelkezésre. Jelen pillanatban ebben a kategóriában, ha csak az írott számjegy felismerésre fókuszálunk, nagyon magas pontosságú eredmények születtek. Az MNIST adatbázis [5,9] egy, az interneten ingyenesen közzé tett adatbázis, amely 60,000 írott számjegyből áll. Ehhez tartozik egy 10,000 írott számjegyből álló teszhalmaz is. Ugyanitt megtekinthetők ezen adatbázison elért legjobb eredmények. Neurális hálókkal sikerült eddig a legjobb eredményeket elérni, melyek közt szerepel az egészen elképesztő 99,65 % pontosság.

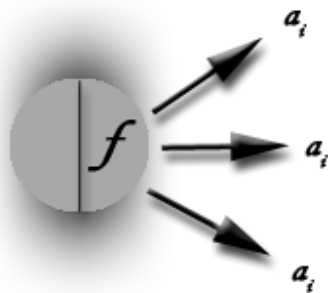
2.1.2 On-line karakterfelismerők

On-line karakterfelismerő rendszer esetében sokkal több információ áll rendelkezésünkre, ugyanis ilyenkor maga a rendszer felügyeli az írás folyamatát is, így arról a bevitel alatt is tud információkat gyűjteni. Ilyen létfontosságú információk például a pixelek sorrendje, az írás időbeni hossza és folyamatossága. Adva lévén az információk ilyen sokszínűsége, lehetőség adódik egy saját, egyedi rendszer kidolgozására, mely az adott információk valamelyikét vagy valamilyen kombinációját felhasználva épít egy felismerőt. Ugyanezen okból kifolyólag nem is hasonlíthatóak olyan nemes egyszerűséggel össze, mint ahogy az off-line típusúaknál lehetséges [13].

2.2 Neurális hálók

A neurális háló alapötletét az emberi agy felépítése adta. Úgy a biológiai neurális hálózatok esetében, mint a mesterséges intelligenciában használt megközelítésben egy olyan struktúráról van szó, amely információfeldolgozást végez. Teszi ezt több összefüggésben lévő alapegység, a neuronok segítségével. Az agyban a neuronok közti kapcsolatok a szinapszisok, melyek továbbítják a köztes adatokat egyik neurontól a másikig. Ettől a ponttól viszont a neurális háló kifejezés a mesterséges intelligenciában használt megközelítésre korlátozódik. Egy ilyen rendszer több alkotóelemből tevődik össze:

- bemeneti neuronok
- kimeneti neuronok
- súlyozott kapcsolatok a neuronok között

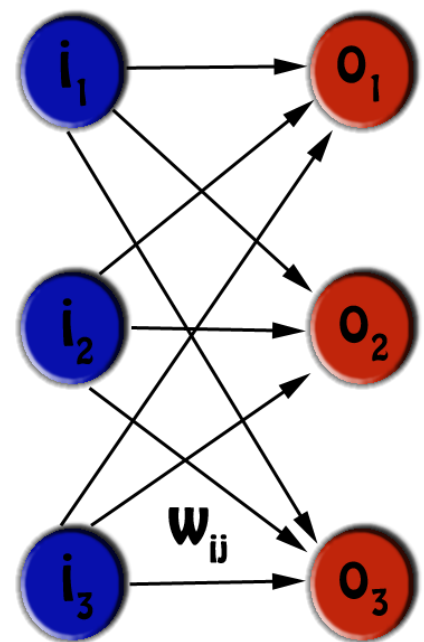


A bemeneti réteg neuronjait (lásd a 3. ábra kék neuronjait) inicializáljuk a megfigyeléseinkből kapott információkkal, és ezeket fogjuk továbbítani a kimeneti rétegnek. Az előbbi két réteg neuronjai közti súlyozott kapcsolatot egy w mátrix tartalmazza, melynek minden eleme meghatároz egy i -dik bemeneti neuronból (piros) a j -dik kimeneti

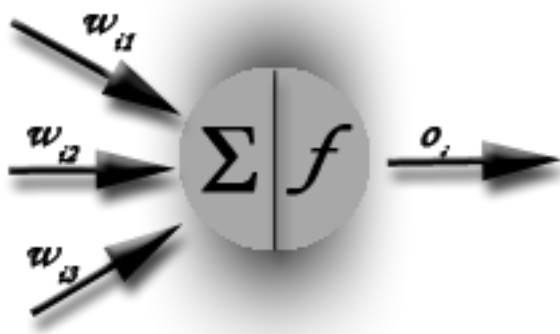
neuronhoz (kék)

2.ábra – Egy bemeneti neuron

kapcsolódó súlyértéket. A kézzel jelölt neuronokra I (input) réteggént, elemeire pedig i jelöléssel történik hivatkozás.



3.ábra - Egy egyszerű neurális háló: a kék neuronok a bementi neuronok, a pirosak meg a kimeneti neuronok



A kimeneti neuronok: összegzik a velük kapcsolatban lévő bemeneti neuronoktól érkező információkat a kapcsolatukon meghatározott súlyok szerint. Ezt az értéket nevezzük net bemenetnek (1-es képlet).

4.ábra – Példa egy kimeneti neuronra

$$net_i = \sum_j w_{ij} a_j \quad (1)$$

Ezen a súlyozott összegben elvégez egy feldolgozási, aktiválási függvényt, melynek kimenete a neuron aktiválási értéke ($a_i = f(net_i)$) lesz. A kimeneti neuronok aktiválási értékei azok, amelyek számunkra az értékes információt hordozzák. Megmondják, hogy az adott bemeneti értékekre, az előre meghatározott faktorokból melyek érvényesülnek és milyen mértékben. Az aktiválási függvény gyanánt gyakran a sigmoid vagy a tangens-hiperbolikus függvényt használják, de bizonyítva van, hogy bármilyen folytonos és nem folytonos függvény alkalmas erre a célra (lásd [1]). Minsky és Papert az 1969-ben megjelent könyvükben megmutatták, hogy egy ilyen rendszer nem képes megoldani nem-lineáris problémákat, például az XOR feladatot [3].

2.2.1 A tanulási folyamat

Minden rendszernek egyszer meg kell tanítani, egy adott mintahalmaz segítségével, hogy milyen adatokat fog elemezni. Ez alól a neurális hálók sem kivételek. De mi tanítható meg egy ilyen rendszeren? Mitől fognak ezek a neuronok úgy aktiválódni, ahogy mi szeretnénk?

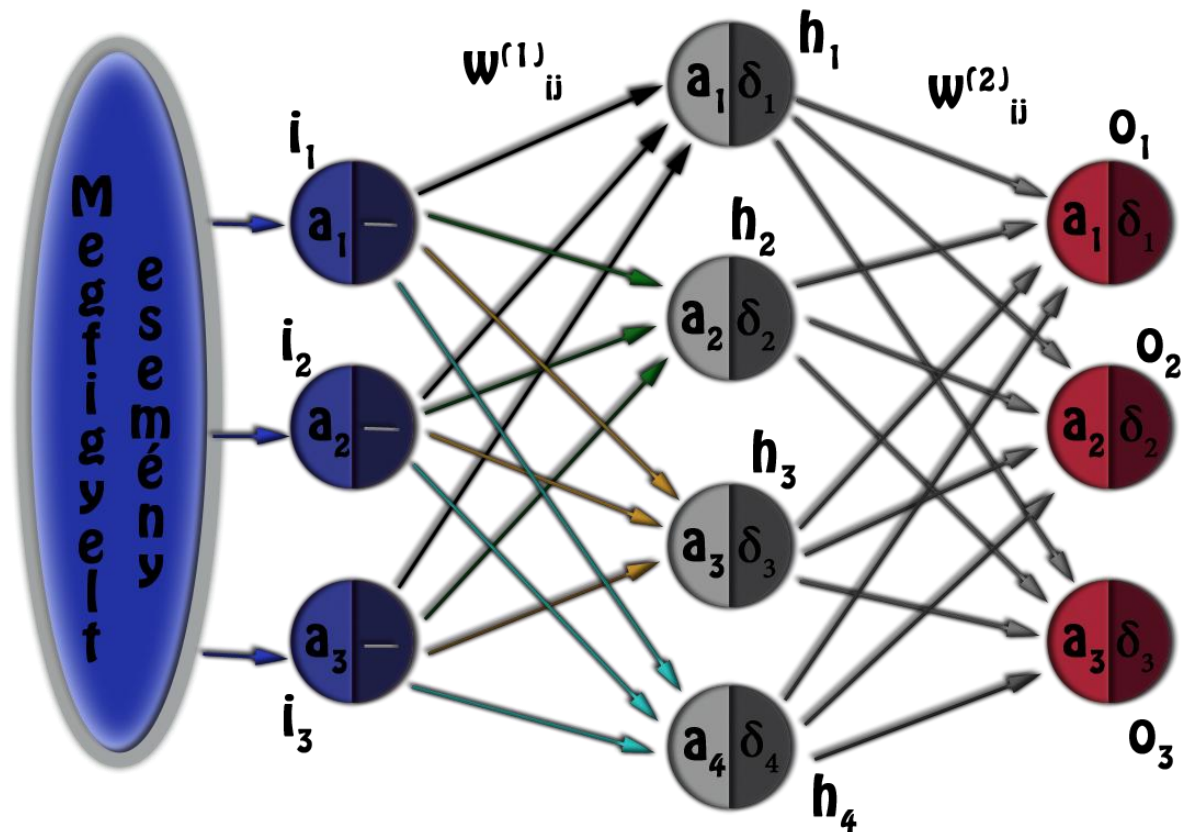
Nos, a válasz a kérdésre egy 1985-ban Rumelhart által kifejlesztett algoritmus [2]. Ennek az algoritmusnak az alapötlete a következő:

1. Határozzunk meg minden egyes típusú bemenetre egy olyan kimenetet, amilyent szeretnénk, hogy az adott bemenetre a neuronjaink generáljanak.
2. Számoljunk hibát a kimeneti neuronokban

3. Módosítsuk egy kicsit úgy az egyes kimenet neuronokhoz tartozó kapcsolat súlyokon, hogy az csökkentse az aktuális hibaértékeket
4. Tegyük ezt meg az összes kimeneti réteg alatt elhelyezkedő neuronokra, mégpedig úgy, hogy adott neuron hibaértékét a fölötte álló neuronok súlyozott összege adja

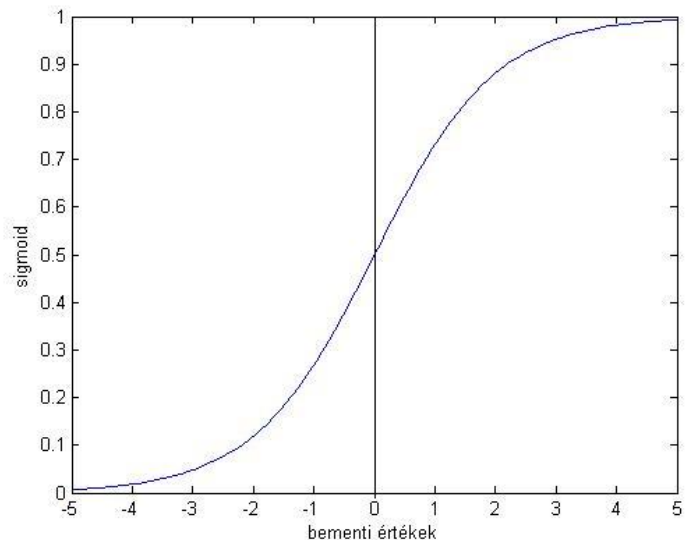
Ezzel az algoritmussal és egy plusz, köztes réteg hozzáadásával, már megoldhatóvá vált az XOR probléma.

“In other words, standard feed forward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function arbitrarily well in the P metric, regardless of the squashing function 'T' (continuous or not), regardless of the dimension of the input space r , and regardless of the input space environment . “ Hornik, 1989



5.ábra – Egy 3 rétegű neurális háló: a kékek a bementi neuronok, a szürkék a rejtett réteg neuronjai, míg a pirosak a kimeneti neuronok. A nyilak jelzik a neuronok közti topológiát és a rajtuk elhelyezett súlyokat

A megfigyelt eseménynek a számokkal mérhető tulajdonságai kerülnek a bemeneti réteg aktiválási értékeibe. A szürkével jelölt neuronok a rejtett, köztes réteg neuronjai, melyeknek értékei csak a rendszer futása közben fontosak. Ezek bemeneti értéke (net_i) a 1-es képlet szerint történik, ezután az adott aktiválási értéket az aktiválási függvény adja meg ($a_i = f(net_i)$). Ugyanezen módszerrel számoljuk ki a kimeneti neuronok (piros) aktiválási értékeit is. Ezen a ponton szükségünk van az adott bemenet típusától függően egy optimális/helyes kimenetre. Ahhoz, hogy ezt megtehesük, ismernünk kell az aktiválási függvény értéktartományait. Ha az aktiválási függvénynek a sigmoid függvényt használjuk, annak kimeneti



6.ábra – A sigmoid függvény

értékei mindig a $[0, 1]$ intervallumba esnek. A minimális érték (0) felel meg az inaktív neuronnak, míg a maximális érték (1) felel meg a teljesen aktív neuronnak. Tehát abban az esetben, például ha 3 db. kimeneti neuronunk van, ezt megfeleltethetjük 3 különböző osztálynak. Egy gyakori eljárás az, hogy minden egyes aktív kimeneti neuron egyetlen osztályt jelöl. Így minden egyes osztály esetében meg kell adnunk az összes kimeneti neuron kívánt értékét. Ez a fenti 3 osztály esetében, a következő képpen nézne ki:

1. osztály : $d_1 = [1 \ 0 \ 0]$
2. osztály : $d_2 = [0 \ 1 \ 0]$
3. osztály : $d_3 = [0 \ 0 \ 1]$

Látható, hogy minden osztály esetében egy adott neuron aktiválási értéket szeretnénk maximalizálni, míg a többiekét minimalizálni. Rumelhart elmélete alapján a kimeneti neuronokban hibaértéket számolunk a következőképpen:

$$\delta_i = (d_{ai} - a_i) * f'(net_i) \quad (2)$$

ahol: a az aktuális bemeneti osztályt jelenti, az f' pedig az aktiválási függvény deriváltját. Ezen az elven hibaértékeket számolunk a rejtett réteg neuronjainak is, méghozzá azzal a

különbséggel, hogy a különbozeti erték helyett, a rejtett neuron összes kapcsolatban lévő kimeneti neuronjával súlyozott hibaösszeget számol:

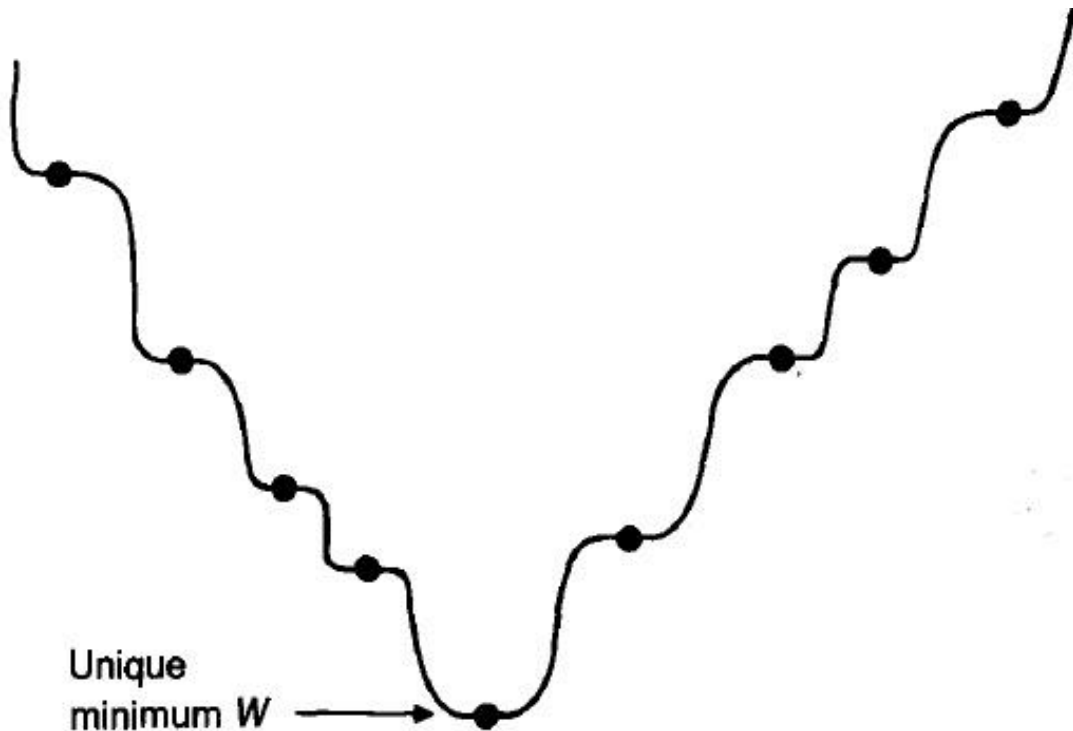
$$\delta_i = f'(net_i) \sum_k \delta_k w_{ki}^{(2)} \quad (3)$$

A bementi rétegen logikus okokból nem definiálható hibaérték. A kiszámolt hibaértékek alapján így már látható, hogy mely súlyok azok, melyek nem helyesen befolyásolják az egyes neuronok érdekeit.

$$w_{ij} + = \eta * \delta_i * a_j \quad (4)$$

η - egy előre meghatározott konstans, a *tanulási ráta*, mely szabályozza egy minta hatását a rendszerre

Rumelhart cikkében bizonyítja, hogy amennyiben a fenti képlet alapján módosítgatjuk a súlyokat, minden minta lefutása után, a rendszerünk konvergálni fog.



7.ábra – A globális minimum keresése

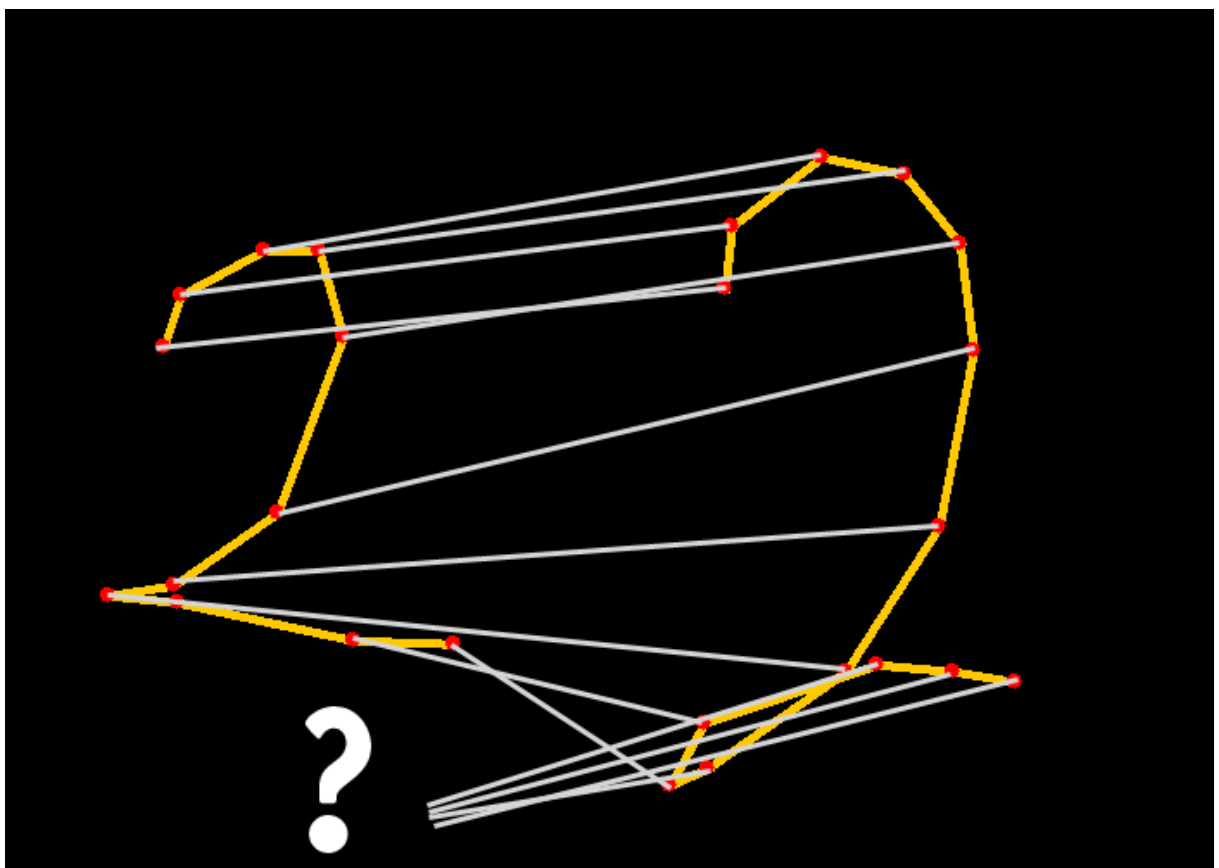
Ezen tanulási algoritmus futási ideje $O(W)$, ahol W a súlyparaméterek számát jelzi. Ez nagyon hatékony, lehetővé téve a kisebb erőforrással rendelkező készülékekben való felhasználást.

2.3 Sablon alapú osztályozó

A fentebb részletezett neurális hálók csupán egy kis része azon technikáknak, melyek lehetővé teszik, hogy felügyelt tanulást végezhessünk egy adathalmazon. Abban az esetben, ha egy jóval kézzelfoghatóbb karakterfelismerőt szeretnénk építeni, kiváló választás a sablon alapú mintaillesztés.

Ezek a rendszerek egy sorba rendezett (x, y) pontokból álló sorozattal dolgoznak, így nyilvánvaló, hogy az online karakterfelismerés területén előszeretettel használatosak [12], mivel itt ezek az információk azonnal adóttak, nem szükségesek extra technikák azok kinyerésére, mint pl. egy kép esetében.

Intuitívan jön az ötlet, hogy két ilyen pont halmaz esetében, számoljunk ezek pontjai közt különbségeket a következőképpen:



8.ábra - Sablonösszehasonlítás problémája

Látható, hogy a probléma akkor merül fel, mikor nem egyezik a mintavételezési pontok száma. Ez pedig jelentős probléma, hiszen ezt a méretet több faktor is befolyásolja:

- milyen gyorsan lett lerajzolva az illető minta

- milyen a frissítési sebessége az adott érzékelő felületnek

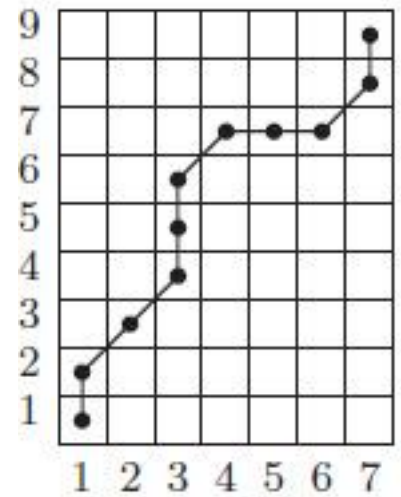
Olyan tényezők ezek, melyeken nem tudunk változtatni és számítanunk kell a probléma megjelenésére. A *DTW* (*Dynamic Time Warping*) algoritmus adja meg a választ a kérdéseinkre, hogy hogyan is hasonlítható össze két nem egyforma hosszúságú lista.

2.3.1 DTW algoritmus

A *DTW* egy dinamikus programozásra épülő módszer, mely a következő elven alapszik [12]:

- építsünk fel egy $n \times m$ mátrixot, ahol n az egyik, m a másik lista hossza
- minden cellába írjuk be a sorának és oszlopának megfelelő két elem különbségét

keressük meg azt az optimális utat a mátrix egyik sarkától a másikig, aminek összegezve az érintett cellák értékeit, ez minimális lesz.



9.ábra – Optimális út a különbség mátrixban

Mivel szükséges a minták összehasonlítás előtti normalizálása, mint az átméretezés, centerezés, esetlegesen újramintázás, és csak ezután futhat le a *DTW*, esetünkben nagyon megugrott a futási ideje a neurális hálókéhoz képest. Ennek fő oka, hogy míg a neurális hálók egy lefutással választ adnak, addig a *DTW* algoritmust annyiszor kell használni, ahány osztály/minta előre definiálva lett. Ez sajnos nagyban túllépte egy mobil eszköz erőforrásait, vagy nagyban károsította annak hatékonyságát. Ennek okán nem került bele a végső rendszerbe.

3 A felhasznált technológiák

A rendszer megvalósítása objektumorientáltan történik, Java alapokon nyugszik, mely később Android alkalmazásba lett beültetve.

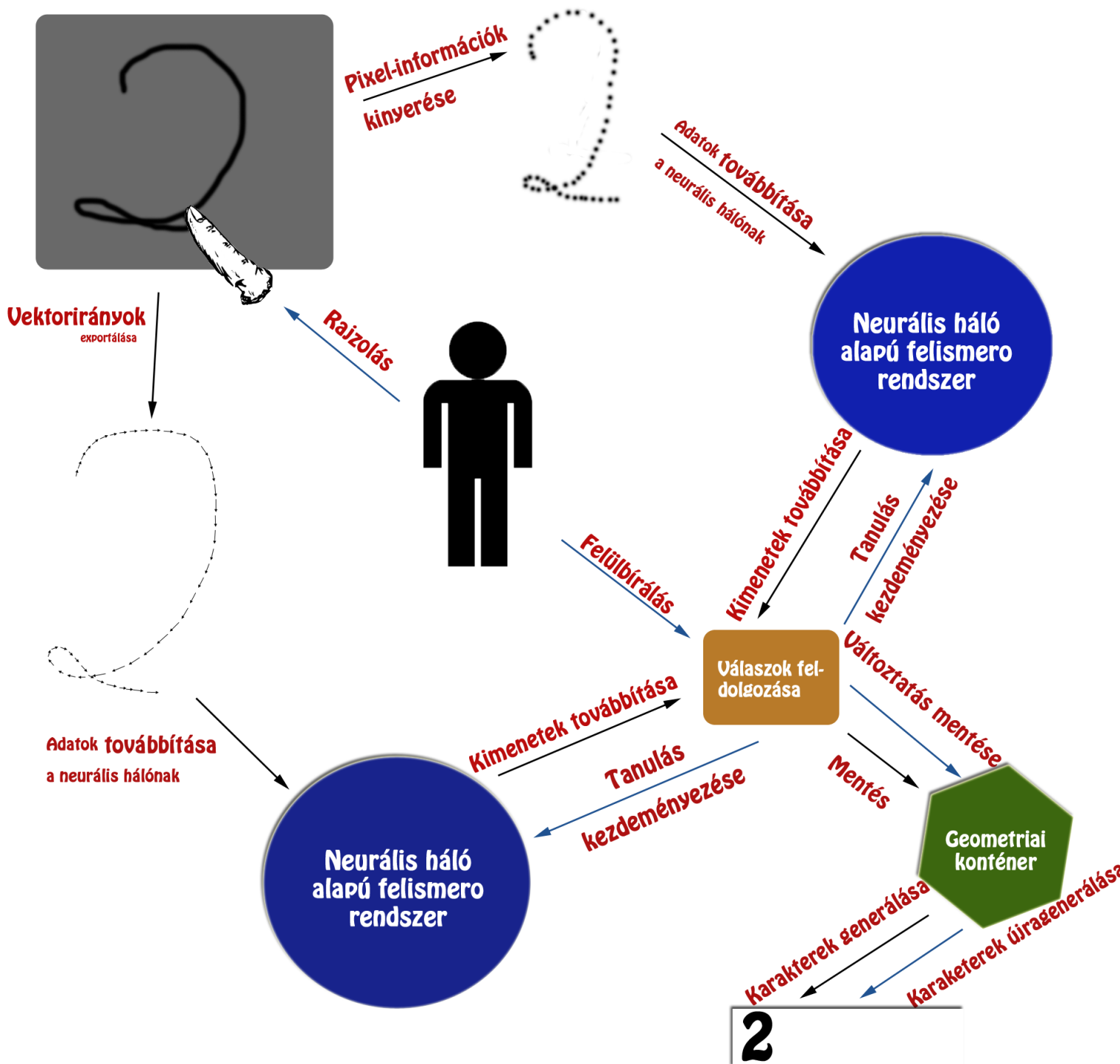
3.1 Android

Az Android [10] fejlesztés során már a legelső lépésektől fontos volt a implementációk optimális mivolta, mivel nagyban rontja a felhasználó élményt egy alkalmazás gördülékenysége hiánya. Másodsorban nagy figyelem fordult a függőségek, dependenciák minél nagyobb kizárására, így kijelenthető az alkalmazásról, hogy amennyiben az eszköz megfelelő, egy 1.6-os Android rendszeren is könnyedén használható.



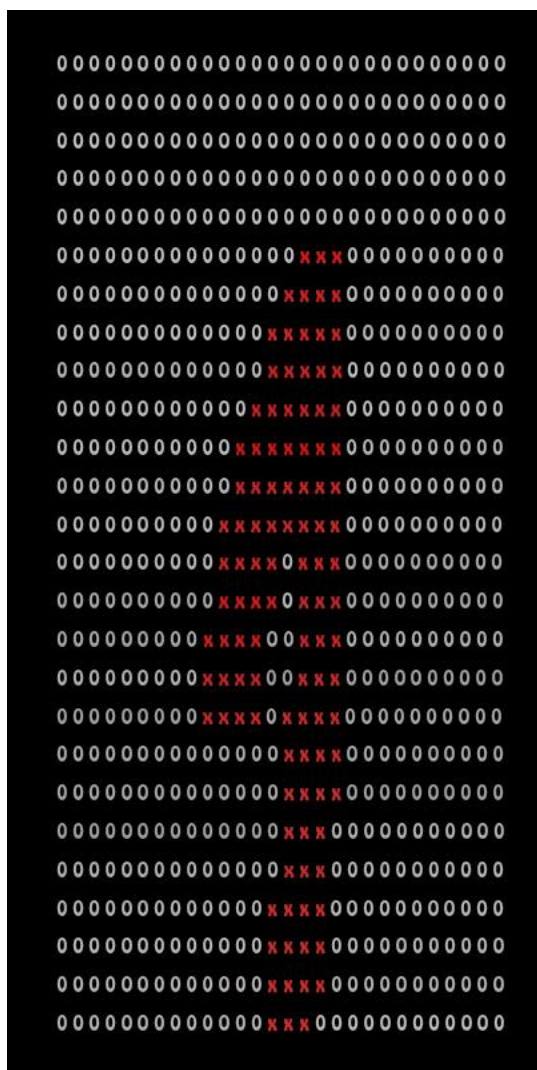
2. ábra – Az Android rendszer felépítése

4 Rendszer felépítés



10. ábra – A rendszer felépítése - A fekete nyilak automatikusan történő eseményeket jelölnek, míg a kékek a felhasználó közvetlen beavatkozása nyomán történő események. Szürkével jelölt felület az érintés érzékeny felületet, míg a színes alakzatok a felismerő alkalmazás részei.

Ahogy az ábrán is látszik, a feldolgozás akkor indul el, amikor a felhasználó befejezett egy mintaberajzolást. Minta alatt egy olyan alakzat értődik melyet egy érintéssel lerajzolhatunk anélkül, hogy felemelnénk az ujjunkat a felületről. A több alakzataból álló karakterek felismerésének megoldását később tárgyaljuk. Amint egy alakzatabot berajzolt a felhasználó,



elkezdődik az előfeldolgozás. Ennek keretein belül kerülnek kinyerésre a két felismerő rendszernek szükséges tulajdonságok. Normalizációs eljárások segítségével kerülnek kiküszöbölésre a különböző méretű és felbontású minták közti különbségek a következőképpen:

1. A bementi mintát újrageneráljuk --> több pontból álló bemenet
2. A mintát átméretezzük, megtartva az arányait, hogy pont beleférjen egy előre megadott konstans keretben
3. Ezután kinyerjük belőle az adott rendszer számára szükséges adatokat

Az utolsó lépés különbözik annak tekintetében, hogy melyik rendszernek készülnek az adatok. Az off-line felismerő rendszernek a pixelinformációk az életbevágóak, ezek alapján dönti el, hogy hová is sorolja az adott mintát. Egy pixelinformációt tartalmazó mátrix látható a 11-es ábrán.

11.ábra - Normalizált belső pixel-reprezentáció

Az on-line tulajdonságok közül a leghasznosabbnak bizonyultak a vektorirányok. Ezek segítségével van reprezentálva magának a ujjmozgásnak a dinamikája. Úgy is tekinthető, mintha az egyik rendszer, egy csak optikai felismeréssel foglalkozna, a másik pedig mint egy vak rendszer, a mögöttes információkat dolgozza fel.

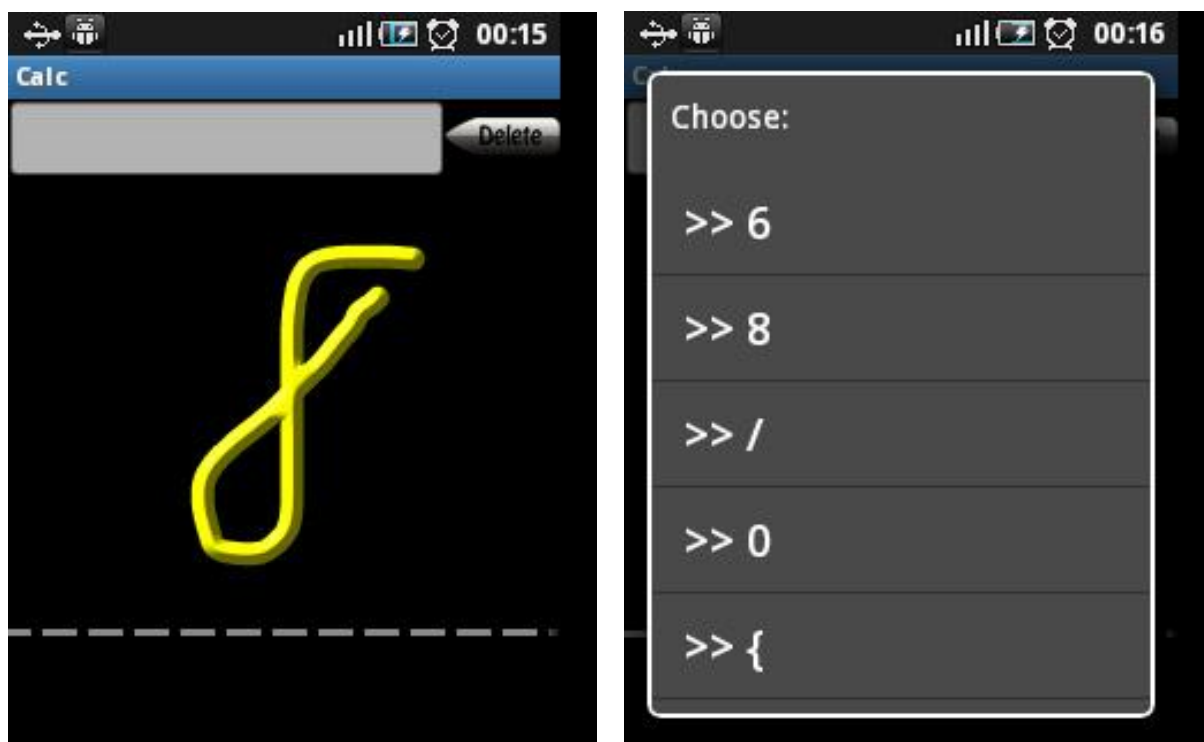
A rendszerek kimenetei egy előre meghatározott halmaz elemei. Ennek a halmaznak az elemei olyan karakterek, amelyek berajzolhatóak egy kézmozdulattal, vagy pedig egy komplexebb karakter részeit képezik. Így, mikor két részleges karaktert felismer a rendszer, azt egy előre meghatározott szabályrendszer alapján összehúzza, egy egységnek tekinti [4]. Ez a szabályrendszer, mielőtt összekapcsolna két részkaraktert, megvizsgálja azok egymáshoz

viszonyított geometriai helyzetét. Figyelembe véve például azt, hogy van-e metszetük egymáshoz képest, hogy vannak elhelyezkedve. Konkréten tekintsük például az '=' karakter esetét, amely nem rajzolható be egy kézmozdulattal, de két darab, relatíve egymás fölött lévő '-' karakterről biztosan állítható, hogy az egyenlőség jel. Emellett még a további szabályok:

- egymást metsző '|' és '-' karakterek '+' jelt alkotnak
- egymást metsző '7' és '-' jelek ugyancsak egy '7'-t jelentenek
- egymást metsző '/' és '\' jelek, szorzásjelet ('x') alkotnak
- egymást nem metsző '┌', '-' de relatív egymás alatt elhelyezkedő jelek, '['-t alkotnak

stb...

Amennyiben a felismerőrendszerek, felismerték az adott bemenetet a kimenet a geometriai szabályok alkalmazása után már generálva is lesz. Abban az esetben ha a két felismerő rendszer nem ugyanarra a karakterre teszi a voksát, részeredményeket generálunk a neurális hálók kimeneti neuronjainak az aktivitása alapján. Ezt úgy tesszük, hogy százalékosan mért aktivitás alapján sorba rendezzük a lehetséges kimeneteket, a két kimeneti listát pedig összesítjük. A listát pedig a felhasználó elé tárva, hagyjuk, hogy döntse el melyik is lett volna az igazi. A 12-es ábrán erre látható egy példa.



12.ábra - Bal oldalt látható maga az alkalmazás, ahová a felhasználó berajzolt egy mintát, a jobb oldalon pedig a lista amelyből választható a kívánt eredmény

Jól látható, hogy annak ellenére, hogy a rendszer nem teljesen biztos a döntésben (ezért is jelent meg a lista), a helyes megoldások az esetek nagy részében az első 3 megoldás közt szerepelnek, bővebben az 5. fejezetben.

A rendszer egyik nagy előnye abban rejlik, hogy az előbb feltárt esetben, mikor a felhasználó válaszol a rendszernek, hogy mi is lett volna az elvárása, a rendszer képes az adott példára és a adott válaszra futási időben, önmagát képezni. Így elérve azt a hatást, hogy a program alkalmazkodjon a felhasználó írási stílusához. Ezúton megoldható, hogy nem egy nagyon komplex rendszerre van szükség ami bármilyen felhasználó kezeiben és bármilyen környezetben hatékony kell legyen.

Ezúton megoldottuk a számjegyek és operátorok bevitelének kérdését, már csak az maradt hátra, hogy megoldjuk magát az aritmetikai kifejezést. Ennek a problémának a megoldására a [11]-ben javasolt eljárás, a fordított lengyel jelölés került felhasználásra. Ez bővítődött ki a magasabb szintű zárójelek bevezetésével. A módszer nagy előnye, hogy olyan formába írja át a kifejezést, amelyben már nem szerepelnek zárójelek. Egy ilyen formulát már sokkal könnyebb kiértékelni, mivel a műveletek prioritása előre adott. A következő egyszerű kifejezés, a $3 + 4 * 6$, a következőképpen fog kinézni: $346 * +$. Mikor már átalakításra került a kifejezés, egy nagyon egyszerű elv alapján értékeljük ki:

- vesszük a lista elemeit addig amíg nem kapunk egy operátort
- mikor van egy operátorunk kivesszük az operátort és az utána lévő két elemen azt végrehajtjuk, az eredményt pedig a kivett elemek helyére visszatesszük
- ismétljük amíg csak egy elemünk marad, az lesz a végeredmény

Az előbbi példa tehát egyszer végrehajtaná a szorzást $3\ 24\ +$, majd az összeadást, így megkapjuk a helyes végeredményt: **27**.

Lássuk, hogyan is működik ez az alkalmazásban:

- a felismert karakterek a felületen lévő pozíciójuk alapján generálódik a karakterlista
- ha ennek a listának az utolsó eleme az '=' jel, akkor kiértékelődik a karakterlista



13. ábra – Bevitelre kerülnek a karakterek 8,-,7,=



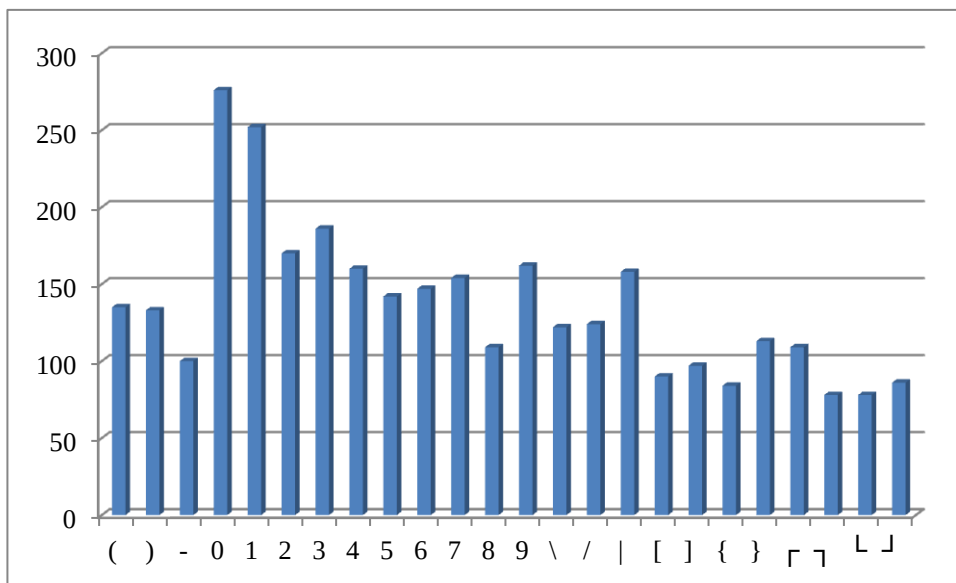
14. ábra – A 2. '-' jel után a rendszer visszakérdez mivel azt nem ismerte fel, viszont a helyes karakter kiválasztása után, megjelent az eredmény és törlődött az '=' jel a további felhasználás érdekében

5 Eredmények

Ebben a fejezetben kerül konkrét kiértékelésre a két felismerő rendszer, továbbá részletezve lesznek a kiképzési, illetve tesztelési adathalmazok.

5.1 Az adatok

A dolgozatban tárgyalt rendszerek mindenike, egy 35 fős csoport által írt, több mint 3500 mintát tartalmazó adathalmazon lett megtanítva és 5 mezős kereszt-megerősítéses



1.diagram – Az adathalmaz mintaeloszlása

módszerrel tesztelve. Minden iterációban két fele osztva az adatokat, 80% és 20% arányban, az előbbi véve tanulási halmaznak, míg az utóbbin teszteltük a hatásfokot. Így sikerült elérni a következő eredményeket:

5.2 Eredmények ismertetése

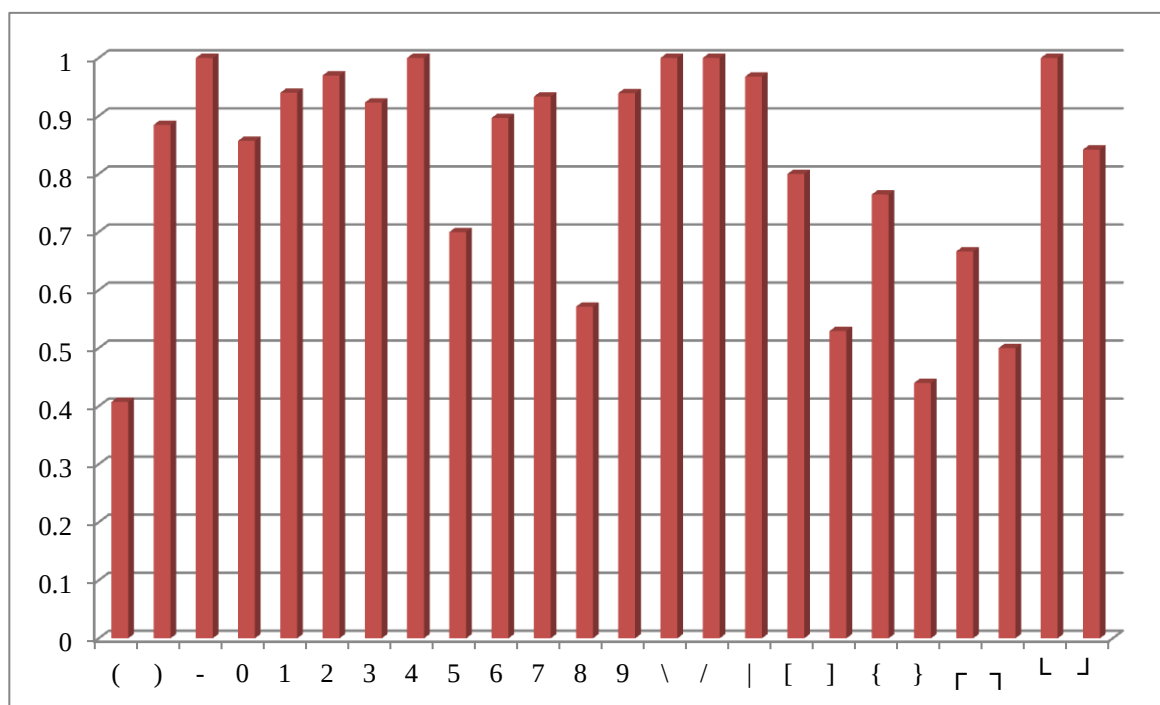
Az online felismerő rendszer esetében: tanulási hatásfok: **83.78%**

teszt halmazon elért hatásfok: **82.34%**

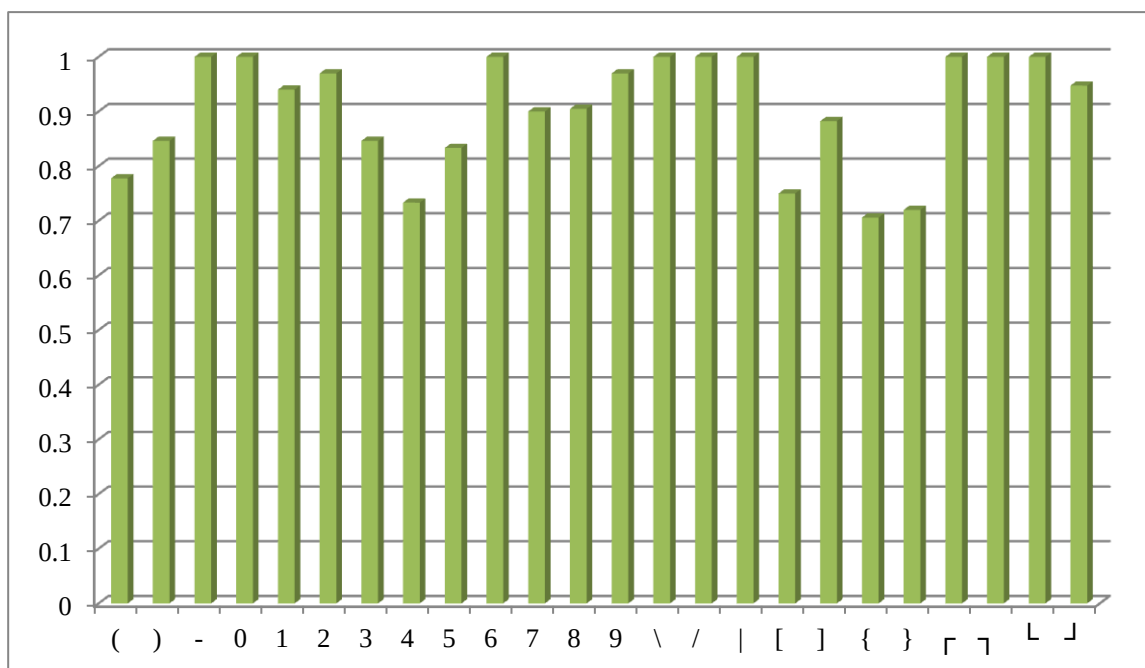
Az offline felismerő rendszer esetében: tanulási hatásfok: **95.92%**

teszt halmazon elért hatásfok: **90.53%**

Részletesebben látszanak a következő két diagramon, osztályokra lebontva a felismerési hatásfokok.



2.diagram - Az online felismerő rendszer osztályonkénti hatásfoka



3.diagram - Az offline felismerő rendszer osztályonkénti hatásfoka

Látható, hogy a két rendszer, amellett, hogy más-más osztályokban bizonyulnak jobbnak, kiegészítik egymást. Az offline felismerőrendszer robusztusabb, nem annyira érzékeny a kisebb eltérésekre, így a tesztelés során is kiegyensúlyozott eredményt produkál. Az online felismerő rendszer viszont már jóval precízebb, így ilyen „kis” halmazon kitanítva, nem ér fel az offline rendszer hatásfokával. Miért is van ez? Gondoljunk csak azokra a

karakterekre, melyet minden ember más- és másféleképpen ír. Például a '{,}', melyeknek írott formája rendkívül változó. Másodsorban az adathalmaz relatív kis mérete (egy normális 60.000-hez képest) nem ad elég információt arra, hogy magasabb hatásfokú rendszert építsünk rá. Ahogy az már bizonyításra került, egy ekkora paraméterhalmazzal rendelkező rendszer esetében, mint ez az online rendszer, a paraméterek számának a több mint 10-szeres mennyiségű begyűjtött minta szükségeltetik a rendszer stabilitásához hosszú távon. A jelenlegi halmaz ennek megközelítőleg a 2%-a. Tehát lehet, hogy a számok azt sugallják, hogy az offline rendszer a jobb, de kétségtelen, hogy további minták begyűjtésével az online rendszerrel magasabb hatásfok érhető el.

Bizonyítva van, hogyha több olyan felismerő rendszert egyesítünk, amelyek ugyanazon eseményt elemzik, de más és más szemszögből, együtt magasabb hatásfokot elérhetnek, mint az külön-külön tennék [8]. Tehát ha meg is történik, hogy nem ugyanarra szavaznak egy adott minta esetében, nagyon valószínű, hogy a válasz benne lesz a találati lista első 3 helyezettjében.

6 Következtetések

Bemutatásra került egy olyan karakter- és számjegyfelismerő rendszer, mely megkönnyíti a számológépet használó emberek életét, kibővíti az erre alkalmas készülékek lehetőségeit. Ezen alkalmazással sokkal ember közelebbé tettük az érintő felülettel rendelkező eszközöket, mivel ilyen alkalmazás még nem található meg a szoftverpiacon.

Az eredmények magukért beszélnek. Az adott halmazon elért hatások minden egyes mintával, minden kézbevitellel csak javulni fog. A felhasználói visszajelzések pozitív mivolta pedig arra enged következtetni, hogy még egy kis erőfeszítéssel a nap mint nap használt alkalmazások közé verekedheti be magát.

A jövőben fontos fejlesztés lenne, hogy a felhasználók megoszthassák egy központi szerveren írásstílusukat, így az azután érkező friss felhasználóknak még hatékonyabb rendszer állna rendelkezésükre. Elengedhetetlen további matematikai műveletek támogatása és bonyolultabb függvények (*sin*, *cos*, *exp*, *log*) beépítése. Ezek maguk után vonják újabb felismerő rendszerek alkalmazását, melyek képesek folyamatosan írt, egymáshoz kapcsolódó karaktereket elemezni, felismerni.

Mindezen terveknek és fejlesztéseknek akkor van értelme, ha megosztásra kerülnek. Tervben van az alkalmazás publikálása a *Google Play* szoftverpiacon.

Irodalom:

- [1] K. Hornik, M. Stinchcombe, H. White: *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, Vol. 2, pp. 359-366, 1989, USA
- [2] D. E. Rumelhart, G. E. Hinton, R. J. Williams: *Learning Internal Representations By Error Propagation*, September 1985, ICS Report 8506
- [3] M. Minsky, S. Papert: *Perceptrons*, MIT Press, 1969
- [4] A. Ahmed, H. Maen: *Recognition of On-line Handwritten Arabic Digits Using Structural Features and Transition Network*, Informatica 32 (2008) pp. 275-281
- [5] E. Kussul, T. Baidyk: *Improved method of handwritten digit recognition tested on MNIST database*, Image and Vision Computing 22 (2004) pp. 971-981
- [6] C. C. Tappert, C. Y. Suen, T. Wakahara: *The State of the Art in On-Line Handwriting Recognition*, Pattern Analysis And Machine Intelligence Vol. 12, No. 8, August 1990
- [7] S. Impedovo, L. Ottaviani, S. Occhinegro: *Optical Character Recognition — A Survey*, International Journal of Pattern Recognition and Artificial Intelligence Vol. 5, Issues:1-2 (1991) pp. 1-24
- [8] Rahman, A.F.R., Fairhurst, M.C: *Multiple classifier decision combination strategies for character recognition: A review*, International Journal on Document Analysis and Recognition Vol. 5, (2003) 166-194
- [9] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE, 86(11):2278-2324, November 1998
- [10] Android honlap: <http://www.android.com> látogatva: 2012. április 19.

- [11] A. W. Burks, W. Warren, J.B. Wright: *An analysis of a Logical Machine Using Parenthesis-Free Notation*, Mathematical Tables and Other Aids to Computation Vol. 8, No. 46 (Apr., 1954), pp. 53-57
- [12] P. Senin: *Dynamic Time Warping Algorithm Review*, Honolulu, USA, December 2008
- [13] R. Plamondon: *Online and off-line handwriting recognition: a comprehensive survey*, Pattern Analysis and Machine Intelligence Vol. 22 Issue: 1 pp. 63 – 84, 2000