Debreceni Egyetem
Informatika Kar

Kézzel írt szövegek feldolgozása tanuló algoritmusokkal

Neurális hálózatok szerepe az optikai karakterfelismerésben

Készítette:
Szabó Csilla
programtervező matematikus hallgató

Témavezető:
Kósa Márk
számítástechnikai munkatárs

Debrecen
2007

Tartalomjegyzék

Ábrajegyzék

Táblázatjegyzék

1. Bevezetés

Napjainkban egyre inkább megfigyelhető, hogy az informatika két ága, a képfeldolgozás és a mesterséges intelligencia több alkalmazási területen is egybefonódik. Ez főként azzal magyarázható, hogy a képfeldolgozásban olyan új ötletek és célok fogalmazódnak meg, amelyek megvalósításához már nem elegendőek az egzakt matematikai képleteken alapuló képfeldolgozási módszerek. Ezen területeken csak olyan módon érhetünk el tényleges eredményeket, ha megpróbálunk valamiféle logikát, gondolkodást és fejlődőképességet vinni a programjainkba.

Ha túllépünk a képfeldolgozásban alapfeladatnak számító képjavítási célokon és elkezdjük a képen lévő információk vizsgálatát, hamarosan azzal a ténnyel szembesülünk, hogy a feladatok összetettsége megköveteli az eddig alkalmazott módszerek kiegészítését, illetve átgondolását. Abban az esetben, amikor egy képen négyzeteket, köröket vagy egyéb egyszerű alakzatokat akarunk felfedezni, viszonylag könnyű dolgunk van, mivel ezek leginkább méretben, színben vagy elforgatásban különbözhetnek. Azonban ha a probléma ennél jóval bonyolultabb, mondjuk arcokat keresünk a képen vagy karaktereket akarunk felismerni, akkor már sokkal adaptívabb és rugalmasabb rendszerre van szükségünk.

Az írásfelismerés fontossága többek között abban rejlik, hogy az írásunk szinte olyan pontossággal azonosít minket, mint az ujjlenyomatunk vagy a DNS-ünk, sőt, esetenként jobban is, hiszen még az egypetéjű ikrekre sem jellemző, hogy tökéletesen azonos lenne az írásképük, míg ez a DNS-szerkezetükre és az ujjlenyomatukra teljesül.

A kézírást már régóta használják azonosítási célokra az informatikában is. Az első aláírás-felismerő rendszert 1965-ben fejlesztették ki, ezt pedig további fejlesztések követték. A Magyar Grafológiai Társaság például saját aláírás-felismerő célhardverrel is rendelkezik. Az aláírások felismerése azonban még nem okoz akkora nehézséget, mint a karakterfelismerés, mivel az ember aláírásában az azonosságokat vizsgáljuk és nem törődünk azzal, hogy ténylegesen milyen betűkből áll össze a név. Amikor azonban az egyes betűket kell felismernünk, figyelembe kell vennünk, hogy emberenként, sőt, még az egyes embereknél időben is szembetűnő eltérések tapasztalhatók az írásképben. Ezt figyelembe véve pedig nem

állíthatjuk, hogy egy betű kinézetét egyetlen vagy akár egy tucat mintapélda alapján meg tudjuk határozni, és ezek segítségével egy leírt karaktert valamilyen osztályba be tudunk sorolni.

Ahhoz, hogy a karakterfelismeréshez megfelelő programot készíthessünk, meg kell vizsgálnunk, hogy mi magunk hogyan állapítjuk meg egy leírt betűről, hogy pontosan micsoda. Az emberi agyban az egyes betűkhöz különböző tulajdonságok rögzülnek, amelyeket viszonylag könnyedén és gyorsan vagyunk képesek alakítani a konkrét feladat jellemzőihez. Ezt az alkalmazkodási képességet azonban már nem lehet pusztán képfeldolgozási algoritmusokkal leprogramozni.

Az emberi agy modellezése egy olyan terület, amely a tudomány több területén is régóta foglalkoztatja a kutatókat. Az informatikában az agy működését úgynevezett neurális hálókkal vagy hálózatokkal szimuláljuk, amelyek belső működése komoly statisztikai alapokon nyugszik. A neurális hálózatok nagy előnye, hogy képesek alkalmazkodni az adott problémához és úgynevezett tanuló rendszerek, vagyis lehetőséget biztosítanak arra, hogy az adott területen javítsuk a teljesítményüket.

A neurális hálózatok egyik dinamikusan fejlődő felhasználási területe az optikai karakterfelismerés (OCR), melynek célkitűzése a nyomtatott vagy írott szövegek karaktereinek felismerése. Az ilyen rendszerek működésénél a képeldolgozási módszereket előfeldolgozásra, a hálózat bemenetének előállítására használjuk, míg a tényleges „felismerést”, osztályozást a neurális háló végzi. Ezáltal ezek a rendszerek inkább tekinthetők mesterséges intelligenciai megvalósításnak, mint képfeldolgozó rendszernek annak ellenére, hogy a megfogalmazott feladatkör valójában képfeldolgozási probléma.

A karakterfelismerés egyre nagyobb teret hódít az informatikában, ezt az a tény is jelzi, hogy több nagy informatikai cég, pl. a Google és a Microsoft is nagymértékben támogatja az ezen a területen történő kutatásokat és fejlesztéseket.

Jelen dolgozatomban igyekszem feltárni az optikai karakterfelismerés jelenlegi eredményeit és feltárni azok hiányosságait, illetve gyenge pontjait. Célom több módszer

bemutatása és összehasonlítása után olyan lehetőség(ek) megtalálása, amelyekből kiindulva a későbbiekben folyóírásos szövegekre is hatékonyan alkalmazható megoldás hozható létre.

A dolgozat elkészítéséhez főként angol nyelvű cikkeket dolgoztam fel, melyek közül válogatva lehetőségem nyílik arra is, hogy a számjegyek felismerésétől, a nyomtatott karaktereken át a kézzel írt karakterek felismeréséig vezető utat is megvilágítsam. Dolgozatom felépítése ennek megfelelően alakul. Először röviden áttekintem a felhasználni kívánt két terület, az alakfelismerés és a neurális hálózatok alapjait, azután pedig lépésről lépésre bemutatom az optikai karakterfelismerés egyre összetettebb feladatait. A dolgozat végén áttekintést adok a Google saját karakterfelismerési projektjének jelenlegi állásáról és megvizsgálom a karakterfelismerés két továbblépési lehetőségét, a magyar nyelvű karakterek felismerését és a folyószövegre való általánosítás lehetőségeit.

2. Az alakfelismerés célkitűzései

A képfeldolgozás az informatika egyik dinamikusan fejlődő területe, amely alkalmazási lehetőségeit tekintve sok egyéb tudományterület feladatainak megoldása illetve megoldásának segítése céljából felhasználható.

A képfeldolgozás valójában többféle célkitűzést is megvalósít, ezáltal több különböző ágazata ismert. Ezek a részek többnyire egymásra épülve azt az eredeti feladatot hivatottak megoldani, hogy számítógép segítségével pontos információkat tudjuk adni arról, hogy mi található egy képen.

Ha a képfeldolgozási feladatokat szintenként tekintjük, akkor beszélhetünk *fizikai szintről*, amely a képátalakítások műveleteit foglalja magába, *elemzési szintről*, amelybe az alakfelismerés és a textúraelemzés tartozik, illetve a *képosztályozási szintről*, amely értelemszerűen a képek osztályozását, a teljes képfelismerést tartalmazza.

Az elemzési szinthez tartozó területekről elmondható, hogy lényegük az objektumok sajátosságai alapján történő osztályozás. Az alakfelismerés a kép makrostruktúrájának elemzésén alapul; a képen bizonyos objektumok jelenlétét vizsgálja.

Az egyes objektumok képeken való keresése során többféle nehézséggel találkozhatunk. Az objektumokat általában valamilyen szignifikáns tulajdonságuk vagy tulajdonságaik alapján reprezentáljuk és a reprezentáció alapján keressük őket a képeken. Egy-egy objektum azonban többféle színben, méretben és helyzetben jelenhet meg akár egy képen belül is. Ennek megfelelően olyan algoritmusokat kell kidolgoznunk, amelyek ezektől az eltérésektől függetlenül képesek egy alakzatot megtalálni egy képen.

Az optikai karakterfelismerés feladatköre az alakfelismerésnél megjelenő problémákat továbbiakkal bővíti, amelynek legfőbb oka a kézírások közötti meglehetősen nagy eltérés. Mivel egy betű leírása minden embernél más és más, nehezen adható olyan reprezentáció, amellyel tetszőleges ember kézírásával készült karakter pontosan besorolható a megfelelő csoportba. De nem is kell ilyen messzire mennünk, hiszen már a különböző betűkészletből vett nyomtatott karakterek között is megfigyelhető olyan mértékű eltérés, amely, ha nem is a

kézírással azonos mértékben, de még így is meglehetősen bonyolulttá teszi az egzakt reprezentációt.

Ha figyelembe vesszük az előbb említett nehézségeket, világossá válik, hogy a képfeldolgozási ismeretek mellett másféle módszerekre is szükségünk lesz, amennyiben az optikai karakterfelismerésben értékelhető eredményeket kívánunk felmutatni.

Mivel az emberi agy számára a karakterfelismerés viszonylag egyszerű feladat, annak megfelelő modellezésével megpróbálkozhatunk a karakterek gépi felismerésének eredményességét javítani. Ennek tükrében a következőkben az agy modellezésére kialakított neurális hálózatokkal fogunk foglalkozni.

3. Neurális hálózatok

Az emberi agy modellezésének gondolata már évtizedek óta foglalkoztatja a tudósokat. Ha körültekintően szemügyre vesszük agyunk felépítését, azt tapasztaljuk, hogy az neuronokból és közöttük felépülő kapcsolatokból áll össze.

A külvilágból és a szervezetből folyamatosan érkező ingerek az agy receptorait állandóan alakítják. Ezt a folyamatot nevezzük tanulásnak. Az agy későbbi döntéseit az ilyen tanulással beszabályozott receptorok fogják meghatározni.

A kutatók az agy felépítését vizsgálva egy olyan matematikai modellt dolgoztak ki, amely reprezentálni próbálja az agyban található neuronokat és a közöttük lévő kapcsolatokat. Ezt a modellt nevezzük neurális hálónak vagy neurális hálózatnak.

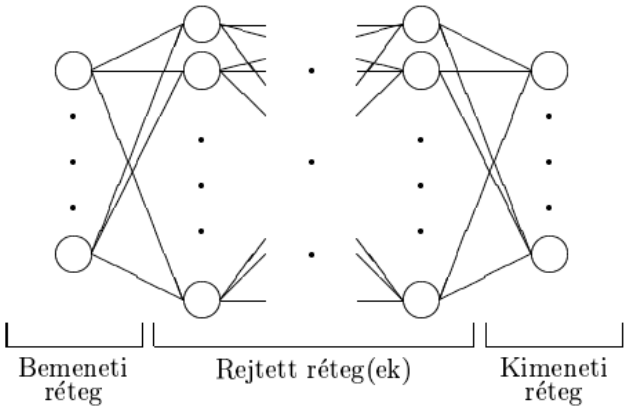
Ahhoz, hogy a célkitűzésünket, az optikai karakterfelismerést meg tudjuk valósítani, közelebbről is meg kell vizsgálnunk a karakterfelismerő rendszerekben alapelemként használatos neurális hálózatokat és azok működését.

Akármilyen jó és bonyolult modellt alkottak is a tudósok, a neurális hálózat mégsem lett alkalmas az agy gondolkodásának rekonstruálására, csupán a tanulás képességét „örökölte”. Amennyiben azonban kellően nagy tanítási adatbázissal rendelkezünk, a betanított hálózat olyan adatokról is képes információkat adni, amelyekkel korábban még nem találkozott.

A neurális hálózatot alkotó neuronok úgynevezett rétegekbe rendeződnek. Háromféle réteget különbözetünk meg, a *bemeneti,* a *kimeneti* és a *rejtett réteget*. Bemeneti és kimeneti rétegből minden hálózatban egy darab van, rejtett rétegből azonban tetszőleges számú lehet [1][5].

A hálózatban a rétegeket élek kötik össze egymással, amelyekhez egyenként egy-egy *súly* tartozik. A neuronok a bemeneti éleiken kapott értékek és a súlyok segítségével bizonyos műveleteket végeznek el, majd az eredmény a kimeneti éleiken keresztül továbbítják a következő réteg neuronjai felé.

A *tanítási folyamat* elvégzésekor a hálózatba olyan bemenetet juttatunk, amelyhez tartozó kimenet ismert. A bemenetet végigfuttatjuk a hálózat rétegein, majd a kimeneti réteg által szolgáltatott eredményt összehasonlítjuk a kimenet várt értékével. A kér érték közötti eltérést a hálózat *hibájának* nevezzük. A tanítás folyamán a hálózat súlyait úgy változtatjuk, hogy ez a hiba lehetőleg minél kisebb legyen[1]. A hálózat betanítása után már olyan bemeneteket is megadhatunk, amelyeknek már nem ismerjük a kimenetét, és a hálózat ezekre is képes hibahatáron belüli kimenetet produkálni.



1. ábra: Egy általános felépítésű neurális hálózat

3.1. Perceptron

A neurális hálók általános felépítésének megismerése után vizsgáljunk meg egy speciális „hálózatot”, amely mindössze egy darab neuronból áll. Az ilyen neurális hálót *perceptronnak* nevezzük.

Egy perceptron hat részből tevődik össze:

- *Bemenet*: Egy ismert m dimenziós vektor *x(n)*.
- *Súlyok:* A valódi súlyokat nem ismerjük, csak azok n-edik közelítését, amelyek rendre a $w_1(n), \ldots, w_m(n)$ értékek.
- *Torzítás:* A torzítás valódi értékét nem ismerjük, helyette annak n-edik közelítését, a *b(n)*-t használjuk.

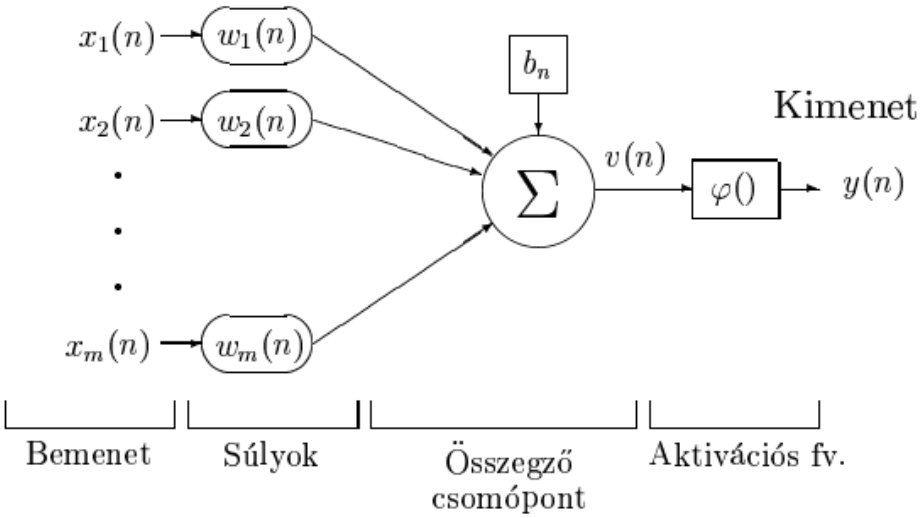
[1] Valójában általában a hiba négyzetének minimalizálására törekszünk.

- *Összegző csomópont:* A bemenet adatait és a súlyokat felhasználva elkészít egy *v(n)* összeget, amit továbbít:

$$v(n) = b(n) + \sum_{i=1}^{m} w_i(n)x_i(n)$$

- *Aktivációs függvény:* Egy olyan φ függvény, amely a *v(n)* értéket leképezi a feladat szempontjából ismert intervallumba.

- *Kimenet:* Ez az az érték, amelyet a perceptron az *x(n)* bemenethez rendel, valójában a kimenet: $y(n) = \varphi(v(n))$ [5].



2. ábra: Egy perceptron felépítése

Általában a *b(n)* torzítást is súlyként szoktuk értelmezni és $w_0(n)$-ként hivatkozunk rá. A torzításhoz tartozó bemeneti értéket pedig $x_0(n) = 1$-nek vesszük.

Amennyiben az *x(n)* bemenethez tartozó ideális kimenetet *d(n)*-nel jelöljük, illetve *y(n)* jelenti a hálózat által az *x(n)* bemenetre adott kimenetét, a neurális hálózat négyzetes hibáját a következőképpen értelmezzük: $\varepsilon = (d(n) - y(n))^2$

Ezt a hibát akarjuk a tanítási eljárás során minimálisra csökkenteni. Természetesen az lenne az ideális, ha a hibát egészen nullára tudnánk redukálni, de ez általában nem sikerül, ezért meg kell elégednünk egy kellően kicsiny hibaküszöbbel.

Aktivációs függvényként tetszőleges balról illetve jobbról folytonos eloszlásfüggvény megfelel, azaz olyan függvényt kell választanunk, amely

- monoton növekvő,
- balról/jobbról folytonos,
- határértéke $+\infty$-ben 1, $-\infty$-ben 0.

Aktivációs függvénynek általában a szigmoid, azaz S alakú, függvényeket használjuk (pl. logisztikus, tangens hiperbolikus stb.).

A perceptront olyan feladatok megoldásánál alkalmazzuk, amikor az egyes értékeket a *k* dimenziós térben ($1 \leq k < \infty$) diszjunkt halmazok egyikébe kell besorolni, azaz osztályozni. Az ilyen jellegű feladatokat *lineáris szeparálási feladatoknak*[2] nevezzük. A feladat megoldását úgy kell elképzelni, hogy a perceptron a *k* dimenziós teret *k-1* dimenziós hipersíkokkal osztja szét.

A *perceptron konvergencia tétele* szerint a tanítóeljárás véges sok lépésben megtalálja a tanulópontokat teljesen szeparáló hipersíkot.

A perceptron konvergenciáját többféle módszer segítségével is elérhetjük. Ezek a numerikus matematikában jól ismert módszerek, mint például a Gradiens-módszer, a Newton-módszer, a Gauss-Newton módszer illetve a legkisebb négyzetek módszere.

[2] A lineáris szeparálhatóságnak több matematikai feltétele is van, amelyekkel egy feladatról eldönthető, hogy az lineárisan szeparálható-e.

Amennyiben elértük a kellő konvergenciát, vagyis a megfelelő érték alá szorítottuk a perceptron négyzetes hibáját, úgy tekinthetjük, hogy a perceptron alkalmas az eredetileg megfogalmazott lineáris szeparálási feladat megoldására.

3.2. Többrétegű perceptron

Amennyiben olyan feladattal van dolgunk, amely nem lineárisan szeparálható, egyetlen perceptronnal nem leszünk képesek azt megoldani. Abban az esetben azonban, ha egy helyett több perceptront használunk, az eredeti lineáris szeparálásnál jóval bonyolultabb felépítésű feladatokat is meg tudunk oldani.

A többrétegű perceptron ötlete onnan ered, hogy a lineáris szeparálásra jól használható perceptronokat egy nagyobb hálózattá kapcsoljuk össze, és ez a hálózat fogja majd betölteni a feladatmegoldó szerepét.

A többrétegű perceptronháló az általános felépítésű neurális hálózatnál megismert három fő részből tevődik össze. Itt is igaz, hogy pontosan egy bemeneti és kimeneti rétegünk van, illetve a hálózat tartalmazhat tetszőleges számú rejtett réteget[3]. A bemeneti réteg a hálózat bal oldalán található. A jel balról jobbra áramlik a hálózatban, míg el nem éri a jobb oldalon található kimeneti réteget. A rétegekben tetszőleges számú perceptron található.

Az egyes rétegek perceptronjait többféleképpen kapcsolhatjuk a következő rétegben találhatóakkal. Lehetőségünk van arra, hogy egy perceptront a következő rétegben található összes perceprtonnal összekapcsoljunk, azaz a perceptron kimenetét a következő réteg összes perceptronja felé továbbítsuk. Azonban olyan hálózatot is létrehozhatunk, amelyben egy perceptron a következő rétegből csak bizonyos perceptronokkal van kapcsolatban. A perceptronok közti kapcsolatok meghatározzák a hálózat működését, így természetesen nem mindegy, hogy hogyan kapcsoljuk össze az egyes rétegeket egymással.

A többrétegű perceptron tanítási eljárása hasonlóképpen működik, mint a perceptroné. A tanulópéldát végigvezetjük a hálózaton, a kimenetet pedig összevetjük a várt kimenettel. Ezután a hálózat négyzetes hibáját kiszámolva azt egy úgynevezett visszaáramoltatási

[3] Általában legfeljebb két rejtett réteggel megoldhatók a neurális háló alkalmazásával elvégzett feladatok.

módszerrel (hátulról előre haladva) visszavezetjük a hálózatba és ennek az eljárásnak a segítségével módosítjuk a hálózat súlyait. A hiba-visszaterjesztés módszerét addig alkalmazzuk, amíg a hálózat hibája egy kívánt küszöb alá nem esik, illetve addig, amíg a két tanítási folyamat (*epoch*[4]) közötti hibaeltérés nem válik elhanyagolhatóvá.

Amíg a perceptron működését még viszonylag egyszerűen meg lehet érteni, a hálózat felépítésének bonyolításával a működés elmagyarázása és megértés egyre nehezebb feladattá válik. Ennek megfelelően a többrétegű perceptronhálót általában egy fekete dobozként szoktuk tekinteni, amely egy meghatározott bementre egy megfelelő kimenetet produkál, de nem próbáljuk megérteni a belső működését.

A tanítási eljárás során figyelnünk kell arra, hogy a hálózat ne váljon *túlillesztetté.* Ez akkor következik be, amikor túlságosan sok tanulópéldát használunk és a hálózat ezekhez a tanulópontokhoz a lehető legjobban közelítő függvényt próbálja meghatározni. Ekkor előfordul, hogy a függvény jól közelíti a tanulópontokat, de közöttük fölöslegesen hullámzik, aminek következtében félő, hogy a tesztadatokra majd hibás eredményeket fog produkálni.

A hálózatok ideális méretének meghatározására sajnos még nem dogoztak ki általános eljárást, ennek megfelelően főként próbálgatásos módszerrel határozhatjuk meg, hogy egy adott probléma megoldásához mekkora és milyen felépítésű hálózatra van szükségünk. Alapvetően kétfajta lehetőségünk van. Kiindulhatunk egy minimális méretű hálózatból és azt bővítve próbálhatjuk megtalálni a feladat megoldása szempontjából már megfelelő hálót vagy egy nagyobb, a feladatot megoldani képes hálózatot csonkolhatunk addig, amíg az még mindig alkalmas marad a probléma megoldására.

A neurális hálózat ideális méretének megtalálása azért szükséges, mert a hálózat méretének növekedésével csökken a hálózat sebessége. Ezért nem ajánlatos az első olyan hálózatnál megállni, amely már meg tudja oldani a kijelölt feladatot, hanem meg kell próbálnunk a probléma megoldására alkalmas lehetőleg legkisebb méretű hálózatot megtalálni.

[4] Az *epoch* a teljes tanítóhalmaz egyszeri végigáramoltatása a rendszeren.

A neurális hálózatok az informatikában széles körben használatosak. Főként olyan területeken alkalmazzák őket, ahol az információt nagy tömegű adatmennyiségből kell kinyerni.

A jelenleg célul kitűzött optikai karakterfelismerés megoldására a többrétegű perceptronhálók meglehetősen jó eredményeket produkálnak, ezért a továbbiakban olyan módszereket és programokat mutatok be, amelyek ezzel a hálózattal oldják meg a karakterfelismerés feladatát.

4. Optikai karakterfelismerés

A különböző szövegek digitalizált – lapolvasóval beolvasott – képének elemzése illetve a rajtuk szereplő karakterek felismerése régóta foglalkoztatja az informatikusokat és fokozott érdeklődésre tart számot a képfeldolgozás, az alakfelismerés és a mesterséges intelligencia területén is. A vizuális (optikai) karakterek felismerésének problémája összetettségben vetekszik az arcfelismerés problémakörével.

A karakterfelismerést többféle módon is csoportosíthatjuk. Ha az írás folyamatáról rendelkezünk valamiféle információról (irányítottság), akkor *on-line* karakterfelismerésről beszélünk. Ezt a módszert használják például a PDA programok, illetve az intelligens táblák szövegfelismerő rendszerei. Ha semmilyen információnk nincsen az írás folyamatáról, csak a leírt karakterek képei állnak rendelkezésünkre, akkor beszélünk *off-line* karakterfelismerésről.

Az off-line karakterfelismerés további csoportokra osztható a szerint, hogy nyomtatott vagy írott karaktereket tartalmaz-e, illetve, hogy hányféle és milyen betűtípusokat használ. Ennek megfelelően a következő csoportokat különíthetjük el:

- Fixed Font OCR – egy vagy meghatározott számú betűtípust tartalmazó szövegek karaktereinek felismerése.
- Omnifont OCR – tetszőleges nyomtatott betűtípust tartalmazó szövegek karaktereinek felismerése.
- Handwriting OCR – kézzel írt önálló karakterek felismerése.
- szkript OCR – megkötés nélküli kézírásos szövegek felismerése.

Jelen dolgozatomban a Handwriting OCR területét vizsgálom meg részletesen, emellett kitérek a karakterfelismerés alapfeladatát jelentő nyomtatott karakterek, illetve számjegyek felismerésére és felvázolom a folyóírásos szövegek felismerésének problémáit, valamint érintőlegesen vizsgálom, hogy milyen nehézségekkel jár, ha a karakterfelismerést kiterjesztjük magyar nyelvű, ékezetes karakterekre is.

Az alakfelismerés klasszikus módszereinek alkalmazása a karakterek felismerésében nem hozza azokat az eredményeket, amelyeket az egyéb problémák esetén megszoktunk. Ennek számos oka lehetséges, de a legfontosabbak talán a következők:

- „Ugyanazon" karakterek emberenként meglehetősen különbözőek lehetnek méretben, alakban és stílusban, emellett még ugyanazon ember esetén is megfigyelhetők bizonyos eltérések a különböző időpillanatokban leírt karakterek között.

- Minden más képpel egyetemben a karaktereket illetve szövegeket tartalmazó képek esetén is számolnunk kell a képalkotás – digitalizálás – folyamán, illetve az egyéb okokból a képre került zajokkal és egyéb képhibákkal (pl. elmosódás).

- A karakterek kinézetének nincsenek megszeghetetlen, tudományosan lefektetett alapszabályai. Ennek megfelelően csak minták alapján tudunk valamiféle szabályosságot megállapítani az egyes karaktertípusok kinézetével kapcsolatban.

Az automata vagy félautomata optikai karakterfelismerő rendszerek készítésének igénye már évtizedekkel ezelőtt megfogalmazódott. Manapság számos erre a célra készített alkalmazás létezik, és mindegyiknek megvannak az erősségei és a gyengéi.

A karakterfelismerés számára készített dokumentumtól elvárjuk, hogy nyomtatott vagy kézzel írt karaktereket, szavakat tartalmazzon. Egy dokumentum azonban általában a szövegen kívül másféle információt is tartalmazhat, például képeket vagy színeket, amelyek a karakterfelismerés szempontjából lényegtelenek, sőt esetenként meg is nehezítik azt. Ennek megfelelően tehát először az OCR számára lényegtelen információt kell eltávolítanunk a képekről, és csak utána kezdhetünk bele a tényleges karakterfelismerésbe.

Amennyiben a bemeneti kép már csak a lényeges információt, vagyis a karaktereket tartalmazza, még mindig szembesülhetünk az általános alakfelismerésnél is jelen levő problémával, a kép zajosságával. Ezeket a hibákat a képfeldolgozásban általában előfeldolgozással el szoktuk tüntetni, itt viszont látni fogjuk, hogy a vizsgált módszerek használatával ez a lépés szükségtelenné válik.

A gyakorlat azt bizonyítja, hogy neurális hálózatok, különösen a többrétegű perceptronhálók meglehetősen hatékonyak a kézzel írt karakterek felismerésében. Az így felépített rendszerek azonban meglehetősen érzékenyek a hálózat méretére és felépítésére. Egy többrétegű perceptronháló feladat megoldási sebessége a hálózatban lévő neuronok számának növelésével növekszik. Emellett azonban az is megfigyelhető, hogy minden problémakör esetében megvan az a minimális méretű hálózat, amelyre szükség van ahhoz, hogy egyáltalán valamilyen eredményt fel tudjunk mutatni a feladat megoldásában. ennek megfelelően a hálózat méretének és szerkezetének megválasztása kulcsfontosságú a feladatmegoldás sikerességének biztosításában.

5. OCR megvalósítások

A következőkben bemutatok néhány olyan alkalmazást, amelyek az optikai karakterfelismerés egyes részeit hivatottak megvalósítani. Ezek között szerepelnek olyanok, amelyek meglehetősen szűk témakörrel, például számjegyek felismerésével foglalkoznak, de lesznek olyanok is, amelyek meglehetősen komoly eredményeket képesek felmutatni a kézírásos karakterek felismerése terén.

Minden esetben részletesen kitérek arra, hogy az alkalmazás készítése során milyen megfontolások vezettek az adott neurális hálózat felépítésének megválasztásához, bemutatom az egyes területeken megjelenő és kezelendő problémákat, illetve összehasonlításokkal és adatokkal támasztom alá az egyes rendszerek hatékonyságát és alkalmazhatóságát a kijelölt problémára.

5.1. Számjegyek felismerése zajos képeken

A zajos számjegyek és karakterek a rendszámtáblák felismerésénél általános jelenségnek számítanak, mivel itt a kamera felvétele általában mozgó járműről készül, és a zajt tovább növelheti a piszok, a nedves, esetleg esős időjárás, éjszakai sötétség, a tükröződés stb.

Jelen esetben tekintsünk el a különböző felvételi körülmények által okozott természetes hibáktól és vizsgáljuk meg, milyen eredménnyel ismerünk fel mesterségesen eltorzított képeken számjegyeket. A kísérlethez *Times New Roman* betűtípussal nyomtatott 96-os betűméretű számjegyeket használunk. A beolvasott képeket a következő módszerekkel tesszük zajosabbá:

- Gauss zajok alkalmazásával.
- Képfeldolgozási eszközök használatával (elmosás).
- Véletlenszerűen részletek kitörlésével.

A számokról készült képek mérete minden esetben 50 x 70 pixel. Amennyiben ettől eltérő méretű a bemeneti kép, azt erre a méretre nagyítjuk, illetve kicsinyítjük. A számokról készített

képeket a méretezés után 5 x 7 részre osztjuk, ahol értelemszerűen minden rész 10 x 10 pixelt tartalmaz.

A feladat megvalósításához többrétegű perceptronhálót használunk, amelynek bemenete a kép egyes szegmenseihez tartozó szürkeségi fokozat (a fekete pixelek százalékos értéke a szegmensen belül). A hálózat, a kép felosztásának megfelelően 35 bemeneti neuront tartalmaz. A hálózatban minden számjegynek megfelel egy kimeneti neuron, tehát ezekből összesen 10 darab van.

Mivel a program hatékonyságát nagymértékben befolyásolja a neurális hálózat felépítése, gondosan meg kell terveznünk, hogy milyen hálózatot alkalmazunk az egyes problémák megoldásához. Jelen esetben az átlagos négyzetes hiba (MSE) segítségével értékeljük a hálózat hatékonyságát és több hálózat teljesítményét összevetve választjuk ki, hogy milyen felépítést használunk a jelenlegi problémánkhoz.

Első lépésben ki kell választanunk a rejtett és a kimeneti réteg neuronjai által alkalmazott aktivációs függvényt. Jelen esetben a rejtett rétegbeli neuronoknál tangens szigmoid, a kimeneti réteg neuronjainál pedig logaritmus szigmoid függvényt alkalmazunk.

Természetesen lehetőségünk lenne akár minden rejtett rétegbeli neuronnál, illetve minden kimeneti neuronnál ugyanazt vagy akár mindegyiknél különböző aktivációs függvényt használni, de jelen esetben, a négyzetes hiba mértékét figyelembe véve ez a két függvény megfelelőnek látszik arra, hogy a segítségével a neurális hálózatunk nagy hatékonysággal ismerje fel az egyes számjegyeket.

Szintén próbálgatásos módszer segítségével a rejtett neuronok számát 9-ben határoztuk meg. Rögzített 50 tanítási epoch mellett a legkisebb négyzetes hibát adó rejtett rétegbeli neuronszámot választottuk.

A következő táblázatból látszik, hogy nem lehet egyértelműen megmondani, hogy az egyes neuronszámok mellett milyen hibaértékek várhatók. A táblázatból az is kitűnik, hogy 9 rejtett neuronnal a négyzetes hibát látványosan kisebbre lehetett visszaszorítani [9].

Rejtett neuronok száma	MSE
1	0,1424
2	0,0485
3	0,0508
4	0,0745
5	0,0809
6	0,0933
7	0,1183
8	0,0386
9	$1{,}9509*10^{-12}$
10	0,0039
11	0,1691
12	0,0025
13	0,0014
14	0,0014

1. táblázat: A négyzetes hiba alakulása a rejtett neuronok számának növekedésével

Ugyanezzel a módszerrel, immár a rejtett neuronok számát rögzítve (9 db) meghatározhatjuk a tanításhoz szükséges epochok ideális számát. Jelen esetben ezt a számot 50-ben határoztuk meg.

A neurális hálózat ideális felépítésének meghatározása után a következő lépés a tanulóhalmaz összeállítása. A számjegyek felismerésére készített hálózatot a következő tanulópéldákkal tanítjuk be:

- Normál számok.
- Háromszor és tizenkétszer elmosott számok.

- Öt és harminc közötti intenzitású Gauss zajjal módosított számok.

A tanítóeljárást 50 epochban Levenberg-Marquardt[5] algoritmus segítségével végezzük. A tanítóeljárás után a négyzetes hiba olyan mértékben lecsökken, hogy a hálózat meglehetősen jó eredményeket produkál a tanulóhalmazban nem szereplő tesztesetek esetében is.

A betanított hálózatot a következőféle példákon teszteltük:

- Normál számok.
- Többszörös elmosással kezelt számok (3, 6, 9 és 12-szeres elmosás).
- Többféle intenzitású Gauss zajjal kezelt számok (5 és 75-ös intenzitás között).
- Részben törölt számok.

A fenti tesztesetek alapján elmondható, hogy a feladathoz használt többrétegű perceptronháló a normál számokat 100%-ban, az elmosott számokat átlagosan 95%-ban, a Gauss zajjal kezelt számokat átlagosan 94%-ban, a részben törölt számokat pedig átlagosan 81%-ban felismerte.

Az eredmények alapján megállapítható, hogy a nyomtatott számjegyek felismerése szembetűnően jó eredményekkel elvégezhető egy jól felépített neurális hálózat segítségével.

5.2. Két 8 x 8-as karakterkészlet felismerése neurális hálózattal

A számjegyek felismerésének problémája látványosan egyszerűbb, mint az általános karakterfelismerés által támasztott feladat. Ennek megfelelően a következőkben egyre bonyolítjuk a megoldandó feladatot, vagyis minden alkalommal egy lépéssel közelebb kerülünk az eredeti cél, a kézzel írt karakterek felismeréséhez. Első lépésként vizsgáljuk meg, hogyan és milyen hatékonysággal ismerhetünk fel nyomatott karaktereket.

Ebben a részben két speciális betűtípussal (*Alt, Ult*[6]) nyomtatott karakterek felismerését tekintjük át egy neurális hálózatot alkalmazó program működésén keresztül. Mindkét

[5] A Levenberg-Marquardt algoritmus a Newton-módszerhez hasonló közelítési módszeren alapul, és segítségével sokkal gyorsabb konvergenciát érhetünk el a neurális hálózatunknál, mint például egy egyszerű hiba-visszaterjesztéses módszerrel.

betűtípus 84 karaktert tartalmaz, a karakterek pedig 8 x 8-as felbontásúak. Mindkét karakterhalmazban megtalálhatók az angol ábécé kis- és nagybetűi, számjegyek és a fontosabb írásjelek is.

A feladat megoldásához használt hálózatban 64 bemeneti neuront alkalmazunk. A hálózat általános előrecsatolt neurális háló. A 64 bemeneti neuron mindegyike a 8 x 8-as karakter egy-egy pixeléhez van kötve. A bemenet fekete pixelek esetén 1, egyéb esetekben 0 értékű. A kimenet meghatározásához egy hétbites címkét használunk (84 karaktert 7 biten tudunk egyedileg ábrázolni). A címkéket nullától 83-ig számozzuk.

Ennek megfelelően a hálózatban hét kimeneti neuron található. A kimeneti neuron tüzelése a címkében 1-es bitértéket, a tüzelés hiánya 0-ás bitértéket jelent. A hálózat egyetlen rejtett rétegében található neuronok számát önkényesen, a bemeneti neuronok számának 1,5-tel való megszorzásával 96-ra állítjuk.

Mivel a neurális hálózatok hatékonysága nagymértékben függ a hálózat méreteitől, elmondhatjuk, hogy a fenti módszerrel felépített hálózattól nem várhatunk túlságosan hatékony működést. Természetesen egy ilyen hálózat is betanítható oly módon, hogy szembetűnő eredményeket mutasson fel, de ehhez lényegesen több erőforrásra és időre van szüksége, mint az előző példában bemutatott, jóval kisebb hálózatnak.

Jelen esetben azért nem törekszünk a hálózat ideális felépítésének precízebb meghatározására, mert az eredeti célkitűzés, a kézírásos karakterek felismerése szempontjából a nyomtatott karakterek felismerésének megvalósítása csupán egy kezdeti lépés. Jelen példát azért érdemes mégis vizsgálni, mert jól demonstrálja, hogy egy nem kellő körültekintéssel felépített neurális hálózat nem hozza a tőle elvárt hatékonyságot. Ez a rendszer konkrétan egy Sun SPARC munkaállomáson több száz rendszerórás betanítás után is alig több mint 50%-os hatékonyságot volt képes elérni.

A rendszer teljesítményének meghatározásához először külön-külön, aztán összesítve is megvizsgáljuk a két választott karakterkészletből vett minták felismerési arányát. Az

[6] Ez a két betűtípus a régi CBM C128-as számítógépen használt betűtípusokhoz tartozik, nem konkrétan karakterfelismerési célokra készült.

értékelésnél külön vesszük a rendszer által ismeretlennek titulált és a rosszul felismert karaktereket.

Az eredmények tükrében elmondható, hogy az *Alt* karakterkészlet elemeit a rendszer nagyobb biztonsággal (körülbelül kétszer olyan hatékonyan) ismeri fel, mint az *Ult* karakterkészletből választott betűket. Az is megfigyelhető, hogy csak kis százalékban fordul elő, hogy a rendszer teljesen „tanácstalan" egy karakter besorolását illetően és ennek megfelelően azt ismeretlennek jelöli meg.

A két karakterkészlet összesített eredményeiből levonhatjuk azt a következtetést, hogy egy neurális hálózat számára valóban jóval nagyobb nehézséget jelent, ha nem „tudja", hogy pontosan milyen karakterekre számítson. Ez a feladat már előrevetíti, hogy milyen nehézségekkel kell szembenéznünk, ha nem egy fix karakterkészletből választott betűket akarunk a rendszerünkkel felismertetni, mint ahogy ez a kézzel írt karakterek esetében is történik. Mindemellett a különböző betűkészletből vett nyomtatott karakterek felismerése annyiban tűnhet egyszerűbbnek, hogy ha fix számú betűtípust használunk, akkor még mindig tekinthetjük úgy, hogy fix karakterkészlettel van dolgunk és elégséges a hálózatnak minden betűtípusból egy-egy mintát megtanítani az egyes betűkből [7].

A táblázatból megfigyelhetjük, hogy ez a neurális hálózat alkalmazás ugyan nem vall teljesen kudarcot a kitűzött feladat elvégzésekor, azonban az összesített eredményeket tekintve mindössze alig több, mint a tesztesetek felében ismeri fel a karaktereket.

Az *Alt* betűkészleten elért eredmények		
Felismert karakterek	66 db	79%
Ismeretlen karakterek	2 db	2%
Rosszul felismert karakterek	16 db	19%
Összes ismeretlen vagy rosszul felismert karakter	18 db	21%
Az *Ult* betűkészleten elért eredmények		
Felismert karakterek	33 db	39%
Ismeretlen karakterek	3 db	4%
Rosszul felismert karakterek	48 db	57%
Összes ismeretlen vagy rosszul felismert karakter	51 db	61%
Összesített eredmények		
Felismert karakterek	99 db	54%
Ismeretlen karakterek	5 db	3%
Rosszul felismert karakterek	64 db	38%
Összes ismeretlen vagy rosszul felismert karakter	69 db	41%

2. táblázat: A neurális hálózat eredményei a két választott betűtípus esetében

5.3. Kézzel írt karakterek felismerése

Az előbbi két példából látszik, hogy a számjegyek felismeréséhez képest a tetszőleges nyomtatott karakterek felismerése jóval nehezebb feladat még abban az esetben is, ha csak limitált számú, megadott betűtípusokból vesszük a bemeneti adatokat. A kézírásos karakterek felismerése még ennél a pontnál is jóval tovább megy, hiszen gyakorlatilag nem támaszkodhatunk arra, hogy egy adott karakterkészletből kell kiválasztanunk a megfelelő betűt.

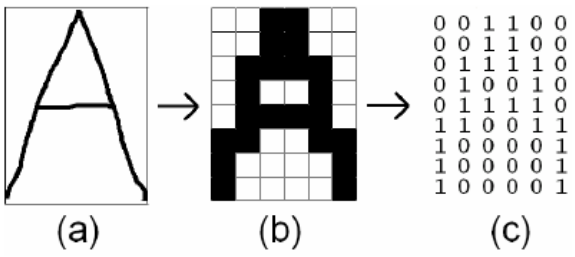
A kézírások változatosságából adódóan nagyon körültekintően kell megválasztanunk, hogy milyen módszert, illetve milyen felépítésű neurális hálózatot használunk, hiszen nem csak a

felismerés hatékonyságát, hanem a rendszer betanítási és futási idejét is figyelembe kell vennünk.

Ebben a részben egy olyan neurális háló alkalmazással ismerkedünk meg, amely már kézzel írt karaktereket kap bemenetként és ezek felismerésére tesz kísérletet.

A kézzel írt karakterek felismerésének első lépéseként a felismerni kívánt karaktereket szeparálni kell a képen található többi információtól, illetve a szavakat, szókapcsolatokat karakterekre kell bontani. A kép ilyen formájú feldolgozása után az egyes karaktereket tartalmazó képeket digitalizálni kell. A digitalizálás folyamán a képekből egy bináris mátrix készül. Minden fekete pixelnek egyest, minden fehérnek nullát feleltetünk meg[7].

A következő ábrán látható, hogy milyen lépések folyamán jutunk el a kézzel írt *A* betűtől addig a bitmátrixig (*I*), amelyet a neurális hálózat már értelmes bemenetként képes kezelni. A digitalizálás folyamán minden karaktert ugyanakkora méretűvé alakítunk, így a hálózat minden esetben egy előre meghatározott méretű bitmátrixszal dolgozik majd [6].



3. ábra: A kézzel írt *A* betű átalakítása a neurális hálózat számára értelmes bemenetté

A karaktereket ennél a programnál felügyelt tanulással tanítjuk be a rendszernek. Minden karakterhez tartozik egy *címke*, amely azt határozza meg, hogy az adott kép milyen betűt reprezentál. Minden címkéhez számos karaktert használunk fel a tanítási folyamat során, így a rendszer többféle variációban is „látja” az egyes betűket. A tanításhoz egy *M* bemeneti mátrixot használunk, amelyet a következőképpen állítunk elő:

[7] Jelen esetben nem térek ki a kép előfeldolgozására, melynek során a képről eltüntetjük a színeket, illetve különböző képjavítási eljárásokkal kiszűrjük a felesleges információkat, zajokat.

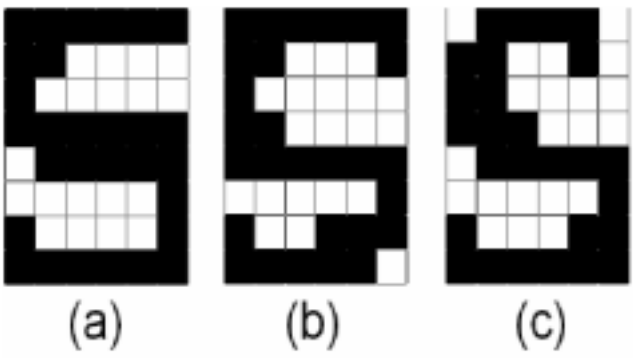
- Ha $I(i,j) = 1$, akkor $M(i,j) = 1$
- Ha $I(i,j) = 0$, akkor $M(i,j) = -1$

A *k*-adik megtanulandó karakterhez tartozó súlymátrixot W_k-nak nevezzük. Ez a mátrix alapértelmezés szerint nullmátrixként van inicializálva. A súlymátrix változtatására egy egyszerű algoritmust használunk. Veszünk egy mintát és jelezzük a rendszernek, hogy ezt a mintát a tudásbázis *k*-adik karaktereként kívánjuk azonosítani. Ennek megfelelően a mintához a *k* címkét rendeljük hozzá. Ezek után a a *k*-adik súlymátrix elemeit a következő algoritmus szerint módosítjuk, ahol x és y a W_k és az *M* mátrixok dimenziója:

```
for i <- 1 to x do
        for j <- 1 to y do
                Wk(i,j) <- Wk(i,j) + M(i,j)
        end for
end for
```

Ezt az algoritmust minden egyes tanulópélda esetén lefuttatjuk, ezáltal kialakítjuk a neurális hálózatban a felismeréshez használt súlyokat. Ez a módszer nem illeszkedik a neurális hálózatok elméleténél bemutatott súlybeállítási módszerekre, mindemellett meglehetősen hatékonynak bizonyult az alkalmazás tesztelése során, illetve az is elmondható, hogy ez a módszer lényegesen egyszerűbb, mint például a hiba-visszaterjesztéses módszer.

Példaként tekintsük meg az *S* betű három eltérő előfordulását és vizsgáljuk meg az *S* betűhöz tartozó súlyvektort a tanítási fázis után.



4. ábra: Tanulópéldák *S* betűre

Az ábrán látható, hogy az *S* betűt többféle módon is le lehet írni. Ezek közt a betűk közt az emberi agy ugyan érzékeli a különbséget, azonban elmondható, hogy még ennél sokkal nagyobb eltérések esetén is képesek lennénk felismerni egy *S* betűt. Egy általános alakfelismerő programnak azonban már nagy valószínűséggel nehézséget okozna megállapítani, hogy a három kép ugyanazt a karaktert ábrázolja, mivel a három közül bármelyiket választjuk is ki, mint mintapéldát, az egzakt reprezentációjuk (vagyis a hozzájuk rendelt bitmátrix) olyan mértékben különbözik, hogy egy általános illesztő algoritmus már nem találná őket azonosnak.

Ha jobban megfigyeljük a betűket, észrevehetjük azokat a szemmel látható hasonlóságokat, amelyek alapján az agyunk megállapítja, hogy ezek a képek *S* betűket ábrázolnak. Ha egy *S* betűt magunk elé képzelünk, mindenképpen egy jobb felső sarokból induló görbét látunk, amely valahol középen irányt változtat és a bal alsó sarokban fejeződik be. Az ábrán látható három *S* betű magán viseli ezeket a jellegzetességeket, habár ezeket konkrét reprezentáció esetén csak nehezen tudjuk leírni. A korábban leírt tanítási algoritmus során a súlyvektorban kiemeljük azokat a részeket, amelyek leghangsúlyosabbak az egyes karakterek előfordulásainál.

Ha ezzel a három tanítópéldával elvégezzük a fentebb említett tanítási eljárást, az *S* betűhöz tartozó W_s súlymátrix a következőképpen fog kinézni:

$$W_s = \begin{pmatrix} 1 & 3 & 3 & 3 & 3 & 1 \\ 3 & 3 & -3 & -3 & -1 & -1 \\ 3 & -1 & -3 & -3 & -3 & -3 \\ 3 & 3 & 1 & -1 & -1 & -1 \\ -1 & 3 & 3 & 3 & 3 & 3 \\ -3 & -3 & -3 & -3 & -3 & 3 \\ 3 & -3 & -3 & -1 & 1 & 3 \\ 3 & -3 & -3 & -1 & 1 & 3 \\ 3 & 3 & 3 & 3 & 3 & 1 \end{pmatrix}$$

5. ábra: Az *S* betűhöz tartozó súlymátrix

Erről a mátrixról elmondható, hogy a nagyobb (pozitív) számok jelzik benne azokat a pixeleket, amelyek a leggyakrabban előfordulnak a karaktert ábrázoló mintákban, a kisebb vagy negatív elemek pedig a betű képében kevésbé megjelenő pixeleket jelentik.

A rendszerben használt hálózat bemenete egy bitmátrix (tanulópéldák esetén *M*). A bemenetet *n* darab különböző karakter felismerésének tanítása esetén *n* darab neuronnak adjuk át. A mintafelismerés a következőkben leírt módon történik.

Vegyük a következő módon megkapható függvényt:

$$\psi(k) = \sum_{i=1}^{x} \sum_{j=1}^{y} W_k(i,j) * I(i,j)$$

Ez a függvény minden *I* input esetén minden *k* karakterre kiszámít egy pontszámot, amivel a *k*-adik súlymátrixra való illeszkedést jelzi. Megfigyelhetjük, hogy a betanítási fázissal ellentétben a felismerésnél már a képből készített eredeti *I* mátrixot használjuk a hálózat bemenetének.

A ψ függvény kiszámítása után a következő algoritmus segítségével adjunk meg egy másik függvényt, amely az egyes címkékhez tartozó súlymátrixok pozitív elemeinek összegét számolja ki.

```
for i <- 1 to x do
        for j <- 1 to y do
                if Wk(i,j) > 0 then
                        μ(k) <- μ(k) + Wk(i,j)
                end if
        end for
end for
```

Ezen két függvény megadása után már ki tudjuk számolni az úgynevezett felismerési hányadost (*Q(k)*), amelyet a következő összefüggés alapján értelmezünk:

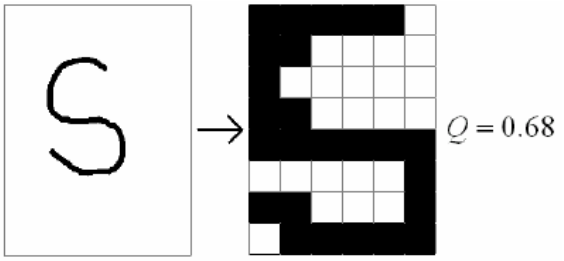
$$Q(k) = \frac{\psi(k)}{\mu(k)}$$

A felismerési hányados azt mutatja meg, hogy a rendszer milyen mértékben tudja a bemenetet az egyes megtanult karakterekkel azonosítani. Minél nagyobb a *Q(k)* értéke, annál valószínűbb a rendszer számára, hogy a bemenet a *k*-adik karakter ábrázolja.

Az osztályozás ezek után a következő egyszerű algoritmus alapján megy végbe a rendszerben:

1. Minden bemeneti *I* mátrix esetén kiszámítjuk az összes megtanult karakterhez tartozó *Q(k)* hányadost.
2. Megkeressük, hogy a *k* melyik értékére adott a program maximális értéket a *Q(k)*-ra.
3. Ha a *Q(k)* maximális értéke túl kicsi (ebben az esetben kisebb, mint 0,5), nem állapítható meg egyértelműen, hogy milyen karakterrel van dolgunk. Ilyenkor két lehetőség adódik:
    - Megállapítjuk, hogy a vizsgált karakter nem található meg a megtanult karakterkészlet elemei között.
    - A vizsgált karaktert addig tanítjuk a rendszernek, amíg a *Q(k)* értéke megfelelő nem lesz.
4. Megállapítjuk, hogy a vizsgált karakter a megtanult karakterkészlet *k*-adik eleme vagy hatékonyabb működés elérése érdekében a rendszernek megtanítjuk a vizsgált karaktert.

Vizsgáljuk meg az *S* betű egy olyan képét, amely nem illeszkedik teljesen egyik megtanult képre sem és figyeljük meg, hogy ebben az esetben mekkora lesz a felismerési hányados értéke.



6. ábra: A felismerési hányados egy eddig nem ismert *S* esetében

Az ábrán látható, hogy bár a vizsgált *S* betű egyik megtanult *S* mintával sem mutat teljes egyezőséget, a hálózat mégis 0,68-as felismerési hányadost számolt ki hozzá, amelynél már megállapíthatjuk az egyezést.

Ha ugyanennyi tudással rendelkező rendszerünknek egy másik betűt, mondjuk egy *P*-t adunk meg bemenetként, abban az esetben a felismerési hányados jóval kisebb (0,21) lesz, tehát a csupán *S* betűket ismerő rendszer biztosan nem fogja felismerni.

Az előbbiekben felvázolt rendszerről elmondható, hogy meglehetősen adaptív és, ahogy azt a példa is bizonyította, kisebb hibákra illetve mintaváltozásra érzéketlen. Mindemellett a tudásbázisa könnyedén bővíthető új karakterek, illetve a már megtanult karakterek újabb változatainak megtanításával. Maga a rendszer meglehetősen általános és a karakterek méretére és arányaira invariáns.

Egy ilyen, teszteléses alapokon felépített neurális hálózat alkalmazás esetén az is elmondható, hogy 32 x 32-es mátrixokkal már a teljes angol ábécét képes nagy hatékonysággal felismerni.

5.4. Többrétegű perceptronok kombinálása

A következőkben egy olyan rendszert vizsgálunk meg közelebbről, amelyben nem egy, hanem két különböző módszeren alapuló neurális hálózat előnyeit egyetlen rendszeren belül hasznosítjuk. Első lépésben megismerkedünk a két hálózattal és az általuk alkalmazott módszerekkel, aztán pedig megvizsgáljuk, hogy milyen módszerekkel lehet a két hálózat előnyeit egy rendszeren belül ötvözni jelentős lassulás nélkül, illetve a rendszer sebességének és hatékonyságának növekedésével [8].

5.4.1. Pixelátlagolásos módszer

A pixelátlagolás a legegyszerűbb módszer, amellyel egy lapolvasóval beolvasott képet digitalizálhatunk. A módszer nagyban hasonlít az előző részben ismertetett rendszernél használatos digitalizálási módszerre. Ebben az esetben a karakter által kifeszített ablakot egy 16 x 16-os mátrixra fogjuk leképezni, melynek minden egyes szegmense a benne található pixelek értékének átlagát veszi fel értékül[8].

A kép eredeti magasságának és szélességének arányát szintén kiszámítjuk és ezt az adatot is felhasználjuk a neurális hálózat bemeneteként. Ennek megfelelően a neurális hálózat bemeneti vektora 257 elemből áll össze.

5.4.2. Normalizált kontúrelemzés

Ezt a módszert az előző módszerrel előállított normalizált karaktermátrixokon hajtjuk végre. Többféle irányból indulva megvizsgáljuk az egyes kontúrvonalak hosszát (az első háttérpixelig tartó részt vesszük) és ezután a hosszokat nulla és egy közé normáljuk. A karakter külsejéből indulva nyolc, a belsejéből kifelé haladva további két irányból vizsgáljuk a kontúrokat. Ezeken felül a karakter vízszintes és függőleges vonalakkal vett metszéspontjainak a számát, illetve a betű oldalainak eredeti arányát is felhasználjuk a hálózat bemeneti vektorában. Ennek megfelelően a bemeneti vektor 193 komponensből áll.

5.4.3. A lényeges tulajdonságok kiszűrése

Mindkét módszer esetében elmondhatjuk, hogy az alap bemeneti vektoraik meglehetősen nagyméretűek. Ahhoz, hogy ezeket a vektorokat valamennyire lekicsinyítsük, el kell tudnunk dönteni, hogy mely komponenseik lényegesek és melyek nem. Az általánosított Fisher kritérium segítségével lehetőségünk van kiszámolni mely elemek milyen leíróképességgel rendelkeznek és kiválasztani ezek közül azokat, amelyek szükségtelenek a felismerés szempontjából.

A karakterfelismeréshez egy többrétegű perceptronhálót használunk egy rejtett réteggel. A pixelátlagolásos módszerrel előállított bemeneten 97,8%-os felismerést ér el a program,

[8] A kiindulási képet szürkeárnyalatosnak tekintjük, tehát jelen esetben egy pixel értéke a szürkeségi értéket jelenti.

miközben az eredeti 257 tulajdonság közül mindössze 46-ot használ. A normalizált kontúrelemzés módszerével kapott bemeneten a hálózat 98,7%-os eredményt ér el és ehhez a 193-ból 69 tulajdonságot vesz figyelembe.

Az első esetben a hálózat 120 rejtett neuron tartalmaz, míg a második hálózatban feleannyi rejtett neuron található. A második hálózat a méretéből adódóan 55%-kal gyorsabb, mint az első.

Annak ellenére, hogy a második hálózat teljesítménye és sebessége is jobb, mint az elsőé, megfigyelhető, hogy a két hálózat által helytelenül felismert karakterek halmaza nem egyezik meg. Ennek ismeretében célszerűnek látszik a két hálózat valamilyenfajta kombinációját alkalmazni, hogy azok együttesen még nagyobb hatékonyságot mutatva ismerjék fel a karaktereket. A hálózatok kombinációjára kétfajta lehetőségünk van. Választhatunk a két hálózat párhuzamosan működtetése, illetve a között, hogy a kevésbé hatékony hálózatot csak akkor használjuk, ha a másik nem ismert fel egy karaktert.

5.4.4. Többrétegű perceptronhálók párhuzamos működtetése

Az előzőekben elkészített két perceptronhálót kétféleképpen is párhuzamosíthatjuk. Az egyik megoldás az, hogy a két hálózatot gyakorlatilag függetlenül működtetjük minden bemeneten, ezután vesszük a kimeneti vektorok skaláris szorzatát és a maximális értékhez tartozó karakterként azonosítjuk a bemenetet. Ezzel a módszerrel a tesztesetek 99,6%-ára helyes eredményt ér el a két hálózat kombinációja.

A két perceptronhálózatot párhuzamosan működtethetjük úgy is, hogy figyelembe vesszük azt a tényt, hogy a kimeneti réteg által kapott bemenet valójában a rejtett réteg által szolgáltatott kimenet. Ennek megfelelően egy olyan új neurális hálót hozunk létre, amelynek kimeneti rétege a két eredeti hálózat rejtett rétegeihez van kapcsolva. Ez a módszer újabb tanítási eljárás lefuttatását igényli, de a bemenet mindkét hálózathoz tartozó neuronok esetében az eredeti reprezentáció szerint alakul, tehát csak a kimeneti réteg súlyait kell változtatnunk, így a tanítási folyamatot kevesebb iterációban elvégezhetjük. Az így kialakított neurális hálózat 99,8%-os hatékonyságot produkál a teszthalmazon.

Annak ellenére, hogy a hálózatok párhuzamosítása nagymértékben lecsökkenti a felismerés hibaarányát, az így előállított perceptronháló méreténél fogva sokkal számításigényesebb, mint a két elődje. Ez egészen pontosan azt jelenti, hogy az új hálózat 2,5-szer lassabb, mint az önállóan működtetett hálózatok gyorsabbika, azaz a normalizált kontúrelemzés módszerét használó rendszer.

A hálózat működésének gyorsítása érdekében annak méretét szükséges lecsökkentenünk, ahogyan azt az egyes eredeti hálózatoknál is tettük. Az átlagos 99,8%-os hatékonyság megtartása mellett a rendszer pixelátlagolásos módszert használó részében a rejtett neuronokat 60-ra, a normalizált kontúrelemzést alkalmazó rész rejtett neuronjait pedig 30 neuronra lehet csökkenteni. A bemenetként kapott tulajdonságokat semmiben nem változtatjuk. Az így kialakított hálózat már csak 40%-kal lassabb, mint az önmagában használt normalizált kontúrelemzésen alapuló perceptronháló.

5.4.5. Többrétegű perceptronhálók kaszkád kombinációja

A neurális hálózatok párhuzamos működtetése a fenti eredmények szerint valóban nagyobb teljesítményt produkál a kézírásos karakterek felismerésében, mint bármelyik önállóan működtetett hálózat. Ennek ellenére a 40%-os sebességcsökkenés nagymértékben csökkenti az új hálózat hatékonyságát, ezért olyan módszert kell keresnünk a két hálózat előnyeinek egyesítésére, amelynél nem jelentkezik ekkora számításigény.

Abból kiindulva, hogy a normalizált kontúrelemzéses módszert használó hálózat 40%-kal gyorsabban ismeri fel azokat a karaktereket, amelyeket felismerni képes, mint a párhuzamos működésre tervezett háló, egy olyan új hálózatot készítünk el, amelyben mindaddig a gyorsabb perceptronhálót működtetjük, amíg az eredményes. Ezek után, ha a normalizált kontúrelemzéses háló visszadobja a bemenetet, a felismerést átadjuk a párhuzamos működésű hálózatnak. Tehát csak abban az esetben használjuk a párhuzamosan összekapcsolt hálózatokat, ha az önállóan működő perceptronháló nem ismerte fel a karaktert.

A normalizált kontúrkövetéssel működő hálózatrész 15,3%-os elutasítási aránnyal dolgozik, de ezzel együtt a kombinált hálózat 99,8%-os felismerési rátát produkál. Más szavakkal ez az új hálózat a párhuzamosan működtetett hálózat teljesítményét nyújtja a

karakterek felismerésében, de az esetek 85%-ában mindössze az egyik, a hatékonyabb hálózatot működteti. Az átlagos felismerési idő 50%-al gyorsabbá vált ezzel a módszerrel és a hálózat több, mint 30%-kal kisebb, mint a legjobb önállóan dolgozó neurális hálózatok egyike.

Az eredmények tükrében elmondhatjuk, hogy amennyiben valamilyen módon kombinálunk karakterfelismerésre készített neurális hálózatokat, lehetőségünk van egy olyan rendszert létrehozni, amely nagyobb hatékonysággal végzi el az optikai karakterfelismerést és esetenként még nagyobb sebességgel is dolgozik.

6. Az OCRopus projekt

A Google cég manapság már nem csupán az internetes keresőprogramjáról híres, hanem számos egyéb szolgáltatásáról is, amelyek között találunk levelezőrendszert, szövegszerkesztőt és táblázatkezelőt, illetve online határidőnaplót is. Mindemellett helyet biztosít egy olyan közösségnek, amelynek tagjai sokféle célú, nyílt forráskódú programokat fejlesztenek.

A Google Code[9] egyik legfiatalabb projektje az OCRopus, amelynek célkitűzése egy általános dokumentumelemző és nyelvtől független optikai karakterfelismerést megvalósító rendszer elkészítése. A projekt két kutatási projekten alapul, az amerikai Census cég által a 90-es évek közepén kifejlesztett kézírás-felismerő rendszeren és egy új nagyteljesítményű elrendezés-elemző motoron [11].

A projekt jelenleg x86-os Linux, illetve x86-os és x64-es Windows platformot támogat; jelenleg készül az Ubuntu 6.10-es verziójának támogatása. Ezeken felül a kód a többi Linux platformra is könnyen átültethető lehet, ennek megvalósításához a készítők folyamatosan várnak segítséget a nyílt forrás világának közösségétől.

A rendszer főbb komponensei C++ nyelven íródtak, illetve van néhány olyan rész, amely a Python nyelvet használja. A verziókövetést Subversion alapokon valósítják meg.

Az OCRopusnak most készül a saját beépített szkriptnyelve, amely az Iua nevet viseli majd.

A rendszer jelenleg a következő elemeket tartalmazza:

- Tesseract karakterfelismerő rendszer
- RAST elrendezéselemző
- Egyszerű aspell alapú nyelvi modell
- Alapvető tesztelő és értékelő modulok

[9] A Google Code projektjeihez a http://code.google.com webcímen csatlakozhatunk.

A további tervek szerint az alfa verzió még 2007-ben megjelenik és sok további, eddig még meg nem valósított modult tartalmaz majd. Ezek között megtalálható lesz egy szöveg, illetve képszegmentáló modul, egy többrétegű perceptronhálón alapuló karakterfelismerő program, egy OpenFST alapú nyelvmodellező, képjavító és feldolgozó eljárásokat tartalmazó csomagok és további tesztelő és értékelő modulok.

A 2008-ra tervezett béta verzióban már további szkriptek és nyelvek támogatásának megvalósítása is tervbe van véve. Az 1.0-ás hivatalos verzió megjelenését szintén 2008-ra tervezik és ebben a verzióban már hangsúlyt fektetnek majd a megjelenésre is, vagyis az OCRopus 1.0 már komplett grafikus felhasználói felülettel fog rendelkezni.

Az 1.0-ás verzió után a fejlesztési lehetőségek száma szinte végtelen. Mindenképpen nagy hangsúlyt kívánnak fektetni a különböző nyelvek karakterkészleteinek felismerési támogatásra, a korábbi Google rendszerekkel (pl. Google Desktop Search) való integrációra, illetve a külső megjelenés további fejlesztésére is.

Mindezen tervek mellett azonban természetesen az OCRopus projektről is elmondható, hogy mint minden nyílt forráskódú rendszer, természetesen ez is szabadon bővíthető és felhasználható, így bárkinek lehetősége van a saját ötleteit megvalósítani és a Google rendszerébe beépíteni.

7. Magyar nyelvű karakterek felismerésének nehézségei

Amennyiben megértettük a kézzel írt karakterek felismerésének nehézségeit, valószínűleg már könnyedén el tudjuk képzelni, hogy milyen további problémákat vet fel, ha nem az angol ábécé betűit próbáljuk megtanítani egy programnak, hanem egy jóval bonyolultabb, ékezetes betűket is tartalmazó nyelvéit, mint amilyen például a magyar nyelv.

Anyanyelvünk több olyan betűt tartalmaz, amely képfeldolgozási szempontból nem egy, hanem két vagy három objektumnak felel meg. Ezek a betűk az ékezetes magánhangzók.

A karakterfelismerés lehetőségeit az ékezetek megjelenése természetesen tovább nehezíti, hiszen ezek olyan, a betűkhöz szorosan hozzátartozó jelek, amelyek azonban képfeldolgozási szempontból nem kapcsolódnak[10] a betű testéhez. Ennek megfelelően az alapvető szegmentálási módszerek alkalmazásával nem biztos, hogy megfelelően fel fogjuk tudni ismeri a karaktereket.

Az ékezetek felismeréséhez többféleképpen kezdhetünk hozzá. Egyik esetben olyan szegmentálási módszert kell találnunk, amely egyetlen objektumnak képes tekinteni az ékezetes betűt, azaz a felismerendő karakter reprezentációjában meg fog jelenni az ékezet is. A másik lehetőségünk, hogy a szegmentálás során leválasztjuk a betű alapját, azt a neurális hálózat segítségével felismerjük, majd az eredeti környezetbe visszahelyezve megvizsgáljuk a betű fölötti területen található képrészletet és ott keressük az esetlegesen megjelenő ékezeteket.

Az ékezetek megjelenésével a további nehézséget az is jelentheti, hogy például az *ö* betű esetén a kétpontos ékezet az esetleges előfeldolgozási folyamat, a zajszűrés esetén teljes mértékben eltűnhet a képről, amely gyakorlatilag visszafordíthatatlan információvesztést okoz a képen.

A magyar nyelv esetében egy másik nehézséggel is szembe kell néznünk, ha karakterfelismeréssel kívánunk foglalkozni. Ez a nehézség a nyelvünkből adódó kettős, illetve

[10] Egy ponthalmazt a képfeldolgozásban *kapcsolódónak* nevezzük, ha az adott szomszédsági struktúra szabályai szerint bármely két pontja között kapcsolódási reláció áll fenn, azaz bármely két pontja között létezik a halmaz pontjaiból álló út.

hármas betűk megjelenése. Természetesen megpróbálhatjuk olyan módon megközelíteni ezt a kérdést, hogy nem vesszük figyelembe az esetleges kettős és hármas betűket, hanem azokat két, illetve három különálló karakterként kezeljük, ebben az esetben azonban megfelelő szegmentálási módszereket kell alkalmaznunk, amelyek biztonsággal elválasztják az egyes betűrészleteket egymástól.

Amennyiben a kettős és hármas betűket egységként kívánjuk kezelni, ezt a szegmentálás folyamatában figyelembe kell venni. Ez a módszer azonban a nyelvünk szavainak felépítése miatt nem javallott, hiszen kérdéses hogy egy általános szegmentáló módszer mit kezd például a *készség* szavunkkal.

Anyanyelvünk sajátosságait figyelembe véve elmondható, hogy abban az esetben érdemes elkezdenünk a speciális magyar karakterek felismerését, ha a fenti „anomáliákra" már konkrét megoldási elképzeléssel rendelkezünk.

Általánosságban az figyelhető meg, hogy ha egy adott nyelv speciális karaktereit kívánjuk felismerni, nem érdemes egy korábbi, mondjuk az angol ábécé betűit már jól felismerő rendszerből kiindulni, hiszen az ékezeteket nem tartalmazó nyelv karaktereihez nem biztos, hogy olyan reprezentációt alkalmaztunk, amely az ékezetes betűk reprezentálására is alkalmas lesz.

Természetesen a magyar karakterek felismerésére készített rendszernél is igazak azok a korábbi megállapítások, melyek szerint egy gondosan megtervezett és felépített neurális hálózat, illetve reprezentáció nagymértékben elősegíti a későbbi felismerés hatékonyságát és sebességét.

8. Általánosítás folyóírásra

Az optikai karakterfelismerés legbonyolultabb feladata az úgynevezett szkript Recognition, vagyis a folyószöveg felismerése. A folyóírással készült szövegek felismerésének problémája visszakanyarodik az alapvető képfeldolgozási módszerek használatához. Mivel egy nyelv szavainak halmaza gyakorlatilag végtelen (különös tekintettel a magyar nyelvben jelenlevő toldalékolási rendszerre), lehetetlen vállalkozás olyan neurális hálózatot tervezni, amely az összes létező szót képes lesz felismerni. Ennek tükrében a folyóírás felismerésének feladatát vissza kell vezetnünk egy olyan problémakörre, amelynek megoldására már léteznek hatékonyan működő rendszerek.

A folyószövegek felismerését egy látszólag könnyű módszerrel, a szavak betűkre bontásával egyszerűsíthetjük le az optikai karakterfelismerés szintjére. A karakterekre bontás azonban egyáltalán nem triviális, sőt, esetenként részben vagy teljes egészében megoldhatatlan feladat. Némely esetben még az emberi szem sem képes egy szót betűkre bontani; ezeknél a szavaknál természetesen nehezen várható el, hogy egy program, automatizált módszerekkel képes legyen ezt a feladatot megoldani.

A karakterekre bontás legnagyobb nehézségét abban kereshetjük, hogy az egyes betűknek nincs meghatározott, állandó szélességük, illetve a folyóírás során a betűk közé úgynevezett betűkapcsoló elemek is bekerülnek, amelyeket ki kell tudnunk szűrni, hogy ne zavarják meg a betűfelismerésre betanított neurális hálózatot.

A folyószöveget tartalmazó képek feldolgozása a képjavítási eljárások elvégzése után az úgynevezett szegmentálással folytatódik, mely jelen esetben a szöveg szavakra, illetve a szavak betűkre bontását jelenti. A sikeres szegmentáláshoz olyan adaptív, visszacsatolással rendelkező rendszerre van szükség, amelyben ha egy karaktert a neurális hálózat nem ismer fel, lehetőséget kell adni arra, hogy újabb szegmentálási lépéssel egy szélesebb területet tekintve egy betűnek újra megpróbálhassuk a karakterfelismerést.

A szegmentálási feladatban az jelenti az igazi nehézséget, hogy a rendszer fel tudja-e ismerni, hogy a leválasztott, de a hálózat által fel nem ismert karakter esetén a karakter

szélességével van probléma, vagy csak a betű kidolgozatlansága okozza a rossz felismerési eredményeket. Amennyiben egy karaktert sikeresen felismert a rendszer, oda kell figyelni a karakter után következő betűkapcsolás jelenlétére, illetve hiányára, ami megzavarhatja a rendszert és ronthatja a felismerési teljesítményt.

Ezen ismeretek tükrében elmondhatjuk, hogy a folyóírásos szövegek felismerési feladatában a karakterekre bontás jelenti a valódi nehézséget. Amennyiben sikerül egy megfelelő szegmentálási módszert kidolgoznunk, a folyószövegek felismerése az önálló karakterek felismerésével egyenértékű feladattá válik.

9. Összefoglalás

Jelen dolgozatom célja az optikai karakterfelismerés (OCR) területének és megvalósításainak behatóbb vizsgálata volt, melynek során többféle különböző reprezentációs módszereken alapuló neurális hálózatot használó alkalmazás eredményeit mutattam be.

A megvizsgált rendszerek működéséből több tanulság is levonható. A példák tükrében jól látszik, hogy a többrétegű perceptronhálók nagymértékben alkalmasak hatékony karakterfelismerésre, azonban nem feledkezhetünk meg arról, hogy a hatékonyság mellett a hálózatok sebessége is fontos. Ennek megfelelően arra kell törekednünk, hogy minél kisebb méretű neurális hálózattal oldjuk meg a feladatokat.

A bemutatott rendszerek mindegyikénél megfigyelhető volt, hogy azok egy rejtett réteget használtak, melynek méretét általában próbálgatásos, a négyzetes hiba redukálásán alapuló módszerekkel határozták meg. Egyik bemutatott példámban a rejtett réteg mérete önkényesen, a bemeneti réteg neuronjainak számából származtatva lett meghatározva, ennek megfelelően a hálózat nem hozta a kívánt hatékonyságot.

A vizsgált programok általában valamilyen saját elképzelésen alapuló reprezentációt használtak, amelyek közül több is sikeresnek bizonyult a felismerési feladat modellezésére. Általánosságban elmondható, hogy a karakterek egyes részeinek szürkeségi értékein alapuló reprezentációs mátrixok viszonylag egyszerű és hatékonyan alkalmazható megvalósítások a neurális hálózat bementének előállítása során.

A különböző eredményekkel dolgozó neurális hálózatok kombinálása is ésszerű megoldásnak tűnik a karakterfelismerés hatékonyságának növelése érdekében. Az erre látott példákból azt a következtetést szűrhetjük le, hogy célszerű a hatékonyabban működő hálózatot mindaddig önmagában üzemeltetni, ameddig az eredményes és azokban az esetekben, amikor ez a hálózat visszadobja a bemenetet, a két választott hálózat párhuzamosításával ismerhetjük fel a megadott karaktert.

A konkrét felismerési rendszerek bemutatása után dolgozatomban röviden kitértem a Google cég dokumentumelemző és karakterfelismerő kezdeményezésére, az OCRopus projektre is, mely jelenleg még meglehetősen kezdeti stádiumban van.

A bemutatott rendszerek azonban nem adnak kimerítő választ az optikai karakterfelismerés minden kérdésére. Ennek megfelelően nagyvonalakban felvázoltam a karakterfelismerés két általánosítási lehetőségét, az ékezetes karaktereket tartalmazó magyar nyelvű és a folyóírásos szövegek felismerésének problémáit.

A következő lépés az optikai karakterfelismerésben tehát az lehet, hogy megtaláljuk azokat a módszereket, amelyekkel tetszőleges nyelvű, illetve folyamatosan írt szövegek felismerését is meg tudjuk valósítani.

Irodalomjegyzék

[1] Norvig, Peter; Russell, Stuart J. – Mesterséges Intelligencia – Modern megközelítésben, Panem Kiadó, 2000

[2] Fazekas, Attila; Kormos, János – Digitális képfeldolgozás matematikai alapjai (egyetemi jegyzet)

[3] Fazekas, Attila; Kormos, János – Alakfelismerés (egyetemi jegyzet)

[4] Fazekas, István – Neurális hálózatok (egyetemi jegyzet)

[5] Horváth, Roland – A neurális hálók alapjai és alkalmazásaik (diplomamunka)

[6] Araokar, Shashank – Visual Character Recognition using Artificial Neural Networks, 2005

[7] Brown, Erik W. – Applying Neural Networks to Character Recognition, 1992

[8] Gosselin, Bernard – Optimal Neural Network Combination for Handwritten Character Recognition, 1996

[9] Mashor, Mohd Yusoff; Sulaiman, Siti Noraini – Recognition of Noisy Numerals using Neural Networks, 2001

[10] Perez, Juan-Carlos; Vidal, Enrique; Sanchez, Lourdes – Simple and Effective Feature Extraction for Optical Character Recognition, 1994

[11] http://code.google.com