ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Ferenci Erik**
2023	Hallgató törzskönyvi száma:	**T/005624/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Ferenci Erik**
Törzskönyvi száma: T/005624/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Folyóírás feldolgozása mélytanuláson alapuló rendszerekkel
Handwriting processing based on deep learning systems

Intézményi konzulens: Balázs Elemér
Külső konzulens:

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Szoftvertervezés és –fejlesztés specializáció

A feladat

A digitalizáció korában az egyik fontos szempont az analóg adathordozók (papír) digitális formába történő átemelése írás tekintetében. Nyomtatott írásfelismerés napjainkban már előrehaladottnak számít, azonban a folyóírás elemzése kisebb támogatottsággal bír.
A hallgató feladata a folyóírás-felismerés aktuális eredményeinek feltárása, bemutatása, különböző módszerek összehasonlítása, majd egy olyan rendszer tervezése és elkészítése, mely képes magyarnyelvű folyóírás feldolgozására és digitalizálására. A megvalósítás során képfeldolgozás mellett mélytanuláson alapuló megközelítést alkalmazzon.

A dolgozatnak tartalmaznia kell:

- a feladat leírását,
- a kapcsolódó szakirodalom feldolgozását,
- hasonló rendszerek jellemzését,
- a felhasznált algoritmusok, módszerek bemutatását és az alkalmazásuk indoklását,
- a rendszer részletes tervét,
- az elért eredmények értékelését és az alkalmazás tesztelési jegyzőkönyvét,
- továbbfejlesztési lehetőségeket,
- a dokumentációt, a programot, a szükséges inputadatokat, valamint a rendszert bemutató honlapot és prezentációt CD mellékleten.

Ph.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

HALLGATÓI NYILATKOZAT

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

Budapest, 2022. 12. 12.

Ferenczi Ede

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: FERENCI ERIK

Neptun kód: [redacted]

Tagozat: NAPPALI

Telefon: [redacted]

Levelezési cím (pl: lakcím): [redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

FOLYÓJÁRÁS FELDOLGOZÁSA MÉLYTANULÁSON ALAPULÓ RENDSZEREKKEL

Szakdolgozat / Diplomamunka[2] címe angolul:

HANDWRITING PROCESSING BASED ON DEEP LEARNING SYSTEMS

Intézményi konzulens: BALÁZS ELEMÉR

Külső konzulens: ———

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2021. 02. 10.	PROJEKT INICIALIZÁLÓ MEGBESZÉLÉS	
2.	2021. 03. 20.	IRODALOMKUTATÁS ÁTTEKINTÉSE	
3.	2021. 04. 18.	HASONLÓ RENDSZEREK ÖSSZEVETÉSÉNEK BEMUTATÁSA	
4.	2021. 05. 07.	RENDSZERTERV VÉLEMÉNYEZÉSE	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplomamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 20 21. 05. 07.

intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: FERENCI ERIK

Neptun Kód: [redacted]

Tagozat: NAPPALI

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat / Diplomamunka[5] címe magyarul:

FOLYÓÍRÁS FELDOLGOZÁSA MÉLYTANULÁSON ALAPULÓ RENDSZEREKKEL

Szakdolgozat / Diplomamunka[6] címe angolul:

HANDWRITING PROCESSING BASED ON DEEP LEARNING SYSTEMS

Intézményi konzulens: BALÁZS BLOMÉR

Külső konzulens: ———

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2022.09.06.	TANÍTÓMINTA ÖSSZEÁLLÍTÁSÁNAK MEGBESZÉLÉSE	
2.	2022.10.04.	KEZDETI EREDMÉNYEK BEMUTATÁSA	
3.	2022.10.18.	EGYEZTETÉS A KÉPFELDOLGOZÁS MÓDSZEREIRŐL	
4.	2022.11.27.	ELÉRT EREDMÉNYEK ISMERTETÉSE, AZ ELKÉSZÜLT DOKUMENTUM ÁTTEKINTÉSE, VÉLEMÉNYEZÉSE	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat / Szakdolgozat I. / Szakdolgozat II. / Diplomamunka II. / Diplomamunka III. / Diplomamunka IV. / Projeklabor 2. / Projektlabor 3. / Záródolgozati projekt[7] tantárgy követelményét teljesítette, beszámolóra / védésre[8] bocsátható.

Javasolt érdemjegy: 5

Intézményi konzulens

Budapest, 2022.11.28.

---

[5] Megfelelő aláhúzandó!
[6] Megfelelő aláhúzandó!
[7] Megfelelő aláhúzandó!
[8] Megfelelő aláhúzandó!

TARTALOMJEGYZÉK

1. Bevezetés

Napjainkban az írásdigitalizálás egyre inkább elterjedté válik az informatika fejlődésével, egyúttal a folyamatosan szélesedő felhasználási területekkel az erre irányuló kutatások, fejlesztések száma is növekedésnek indult. Nyomtatott írásfelismerés már előrehaladottnak számít, azonban a folyóírás elemzése kisebb támogatottsággal bír. Ezen technológiák megismerése lesz a cél a következő fejezetekben.

Az alapvető képjavítási módszereken túl, a képen lévő adatok vizsgálata már sokkal összetettebb feladat. Az eddig ismert eljárások bővítését, átgondolását is megköveteli ez a fajta képelemzés. Vannak esetek, amikor az egyszerűbb alakzatok, (négyzetek, körök) detektálása a célunk, ekkor viszonylag könnyű műveletről beszélünk, mivel ezek a formák nagyrészt csak színben, méretben, vagy állásukban, irányukban különbözhetnek. Viszont ha az elemezni kívánt adat ennél jóval bonyolultabb, legyen szó karakterfelismerésről vagy arcok detektálásáról egy képen, az ilyen esetekben indokolt egy jóval rugalmasabb és adaptívabb egység felépítése.

A kézírásos szövegek felismerésének módjait két csoportba sorolhatjuk: ez lehet online, illetve offline detektálás. Előbbi esetén a karaktereket az írás folyamatában elemezzük, ilyenkor az elemzéshez felhasználhatjuk a folyamatot, hogy az egyes betűk, vonalak, miként lettek leírva, hogyan képezték ezeket. Míg az utóbbi használatakor az írást követően egy képről érkeznek az adatok. Ezt még tovább lehet bontani alcsoportokra úgy, mint gépelt, illetve kézzel írt szövegek. Ezen felül a kézírást lehet csoportosítani nyomtatott betűkkel írt szövegre továbbá írott folyószövegre.

Ezeknek a nagy részére ma már jó eséllyel találhatunk elérhető rendszereket, amelyekkel a különböző szövegek detektálása kivitelezhető. Valamennyi táblagép és okos telefon képes az online szövegfelismerésre, amiket igen pontos eredménnyel képes is teljesíteni. Az offline, nyomtatott, illetve kézzel írt, azon belül a szeparált nagybetűkkel írt szavak detektálására is léteznek már elterjed eszközök. Ezek a rendszerek döntő többségben optikai karakterfelismerést (OCR) alkalmaznak, melyekhez már könnyen hozzá lehet férni.

Ellenben az írott folyószöveg azonosítására már nem feltétlenül találunk egyszerűen megoldást, ugyanis a meglévő lehetőségek közül sok szinte elérhetetlen, vagy csak magas ráfordítás ellenében elérhető. Hiszen ezek az algoritmusok sokkal összetettebbek és a különböző kézírások közötti jelentős eltérések miatt nehezebb egy jól működő szoftver kiépítése. Legfőbb oka az, hogy a betűk írása emberenként más és más, nem könnyű az olyan reprezentáció megalkotása, amellyel tetszőleges kézírással készült karakterek pontosan besorolhatók, felismerhetők.

1.1. Írásdigitalizálás

A képfeldolgozás egy igazán komplex területe az informatikának, hiszen több célkitűzést is megvalósít, számos megoldást nyújt egymástól akár jelentősen eltérő témakörökben is. A cél összeségében nem változik, számítógép bevonásával szerzünk pontos információt a bemeneti képekről.

A képen való keresés folyamán a különböző objektumok eltérő nehézségeket eredményeznek. Maguk az objektumok rendelkeznek valamilyen szignifikáns tulajdonsághalmazzal, amely segítségével képesek vagyunk ezeket reprezentálni. Keresés közben ezekre fókuszálva vizsgálódunk a képen. Az objektumok azonban többféle alakban jelenhetnek meg, ezek az eltérések színben, méretben és elhelyezkedésben is jelentkezhetnek, akár egy képen belül is.

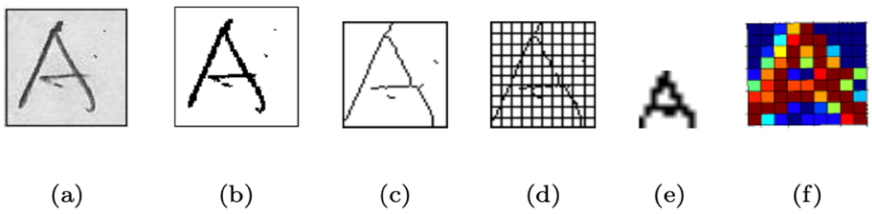
Karakterfelismerés során ennek ismeretében egy olyan algoritmus kidolgozása a cél, amely az eltérésektől szubjektívan képes felismerni az egyes alakzatokat újra és újra. Így az írásfelismerés legfőbb nehézége a kézírások közti merőben nagy különbség kezelése, hiszen még egy személy vizsgálata során is felmerül, hogy valamely írott betűi között nagy a differencia. [1]

Ezeken továbblépve más feladatok is felmerülhetnek. Amikor egy egész oldal, szöveg felismerése a cél, akkor ahhoz, hogy a külön vizsgálható betűkig eljussunk először szükséges meghatározni a sorokat, majd ezeken belül a szavakat. Amint fellelhetők a szavak, az egyes betűket kell detektálni, amennyiben karakterenkénti felismerés a célunk.

Léteznek olyan algoritmusok is, ahol a szótagok, szórészek vagy a szavak felismerése egyben történik. [2] Az írásdigitalizálásra ma már nagyszámban léteznek megoldások, azonban a legelterjedtebbek a neuronhálós megvalósítások.

További ismert eljárások:

- *Statisztikai, geometriai megközelítés*
- *Pozíció-szélesség-impulzus algoritmus* [3]
- *Egyenesek és görbék vizsgálatával történő algoritmus*
- *Fúziós eljárás ábrázolása - 1.1.ábra* [4]

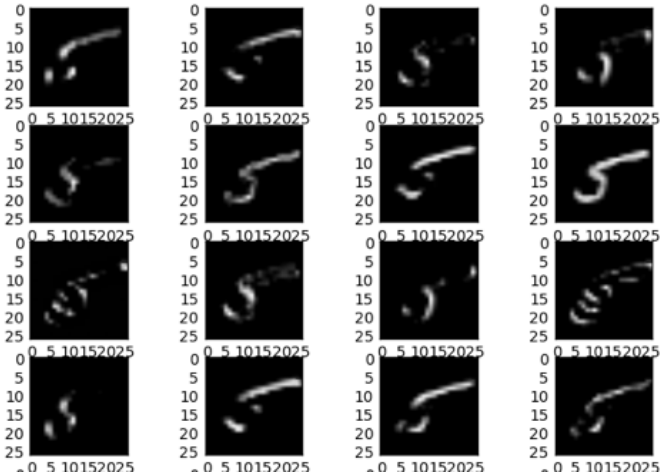


1.1. ábra *Fúziós eljárás első lépései*

a) Eredeti kép (b) Zajcsökkentett kép (c) Képillesztés után (d) 10 × 10 rácsszerkezet (e) Rekonstruált kép (f) Az „A” képi ábrázolása [5]

A karakterfelismerés az emberi agynak relatíve egy egyszerű feladat, ennek modellezésével megpróbálhatjuk javítani a karakterek gépi felismerésének eredményeit. Itt már beszélhetünk az agy modellezésére létrehozott neurális hálózatokról. Ezek előnye, hogy alkalmazkodni tudnak az aktuális problémához és tanuló rendszerek segítségével az adott területen javítani tudjuk a teljesítményüket.

A neurális háló segítségével működő algoritmusnak is sokféle variánsa ismert, hiszen eltérő neurális hálózat kiépítésével, vagy ezeknek a bemeneteinek változtatásával más és más kivitelezéshez jutunk. Az 1.2. ábrán látható egy konvolúciós neurális háló kimenetei, amely az irodalomkutatás során bővebben is kifejtésre kerül.



1.2. ábra *Konvolúciós neurális háló – élek, görbék tanulása [6]*

A célunk, hogy a lehetséges megvalósítások közül a legjobbnak vélt eljárásokat összevetve feltérképezzük, hogy az írott folyószöveg detektálására melyik algoritmus használata a célszerűbb, a felállított támpontoknak eleget téve.

A karakterfelismerésnek léteznek olyan ágai is, amelyek a különböző nyelvek sajátos karaktereivel foglalkozik, ilyenek lehetnek a japán, kínai, vagy akár az arab írásjelek detektálására készült programok. A magyar nyelvben is vannak eltérések az angolhoz viszonyítva, ilyenek az ékezettel rendelkező betűk, amik szintén eltérést jelentenek a megvalósítás során. [6]

1.2. A szakdolgozat célja

Jelen dolgozatomban törekszem az írott folyószöveg detektálására fellelhető aktuális eredményeket feltárni és elemezni. Célkitűzésem, az eltérő megvalósítások vizsgálata és összehasonlítása után egy olyan alternatíva létrehozása, mely hatékonyan alkalmazható a folyóírás felismerésére. A magyar nyelv sajátosságaira szintén vizsgálok megoldásokat, az ékezetes betűk nehézségeire és a toldalékolás hátrányaira is összpontosít a tervezett megvalósítás.

Célom, hogy a magyar nyelven írt folyószövegről készült képet algoritmusok segítségével elemezzem és egy olyan kimenethez jussak, ami a lehető legpontosabban reprezentálja a bemeneti képen lévő szavak tartalmát.

A szakdolgozat elkészítéséhez magyar és angol nyelvű könyveket, cikkeket és tanulmányokat egyaránt felhasználtam. Ezek segítségével haladhattam előre az egyszerűbb képfelismerési technikák megismerésétől a neurális hálók használatáig. A munkám szerkezete pedig ennek függvényében alakult. Először áttekintésre kerülnek a dolgozat alap területei, az írásdigitalizálás, karakterfelismerés és a neurális hálózatok, valamint a mélytanulás. Majd a fellelhető megvalósítások elemzése után egy saját rendszerterv összeállítása történik. Az ehhez szükséges technológiai ismeretek felhasználásával készül el a tervezet. Az implementáció és a tesztelés gyakorlatias leírása is szerepel a szakdolgozatban, ezeknél számos szempontot fontos figyelembe venni a megfelelő megvalósítás érdekében.

Az az állítás került előtérbe, hogy a folyóírás felismerésére egyre több kutatás fókuszál, illetve számos olyan technológia létezik napjainkban amivel már eredményesnek mondható kimenetekhez juthatunk. Mindazonáltal ezt a témakört fogjuk körbe járni, hogy valóban elérhetőek-e ezek a szolgáltatások, illetve módszerek.

2. IRODALOMKUTATÁS

Az irodalomkutatás során kifejtésre kerülnek a projekt megvalósításához szükséges ismeretek. A fejezet ismerteti a képfeldolgozásban használt karakterfelismerés módjait. Az írás detektálása kapcsán, megjelennek a magyar nyelv sajátosságai, nehézségei és a folyóírás problémái. Ezen kívül az íráselemzéshez használt mesterséges intelligencia alapját képező neurális hálózatok felépítését és tanítási lehetőségeit is részletezi a fejezet.

2.1. Karakterfelismerés

A karakterfelismerés az informatikában egyre inkább elterjedtebbé válik, ezt bizonyítja az a jelenség is, hogy több tekintélyes számítástechnikai vállalat kiemelten támogatja azokat a kutatásokat és fejlesztéseket, melyek erre a területre koncentrálódnak.

2.1.1. OCR

Az Optical Character Recognition (OCR), magyarul optikai karakterfelismerés, az egyik legsűrűbben használt módszer a gépelt, kézzel írott vagy nyomtatott szöveg képeinek elektronikus vagy mechanikus konvertálására, gépi kódolású szöveggé. Ez történhet akár beszkennelt dokumentumból, egy fotóból, vagy egy felvételre feliratozott szövegből. Széles körben használják a nyomtatott papír-nyilvántartások digitális felhasználása érdekében, legyen szó útlevelekről, számlákról, bankszámlakivonatokról, számítógépes nyugtákról, névjegykártyákról, vagy bármilyen megfelelő dokumentációról. Az OCR-programok technikáikban változhatnak, de általában egy karakter, szó vagy szövegblokk detektálását jelentik.

A beolvasott képet vagy bittérképet elemzik világos és sötét területek szempontjából, a sötét területeket karakterekként azonosítják, amelyeket fel kell ismerni, a világos területeket pedig háttérként. A sötét területeket ezután további műveletekkel dolgozzák fel, hogy ábécé betűket vagy numerikus számjegyeket nyerjenek ki belőle. A karakterek azonosítása ezután a két algoritmus egyikével történik:

Mintafelismerés - Az OCR-programok különböző betűtípusokkal és formátumokkal ellátott szövegekkel dolgoznak, amelyeket a beolvasott dokumentum karaktereinek összehasonlítására és felismerésére használnak.

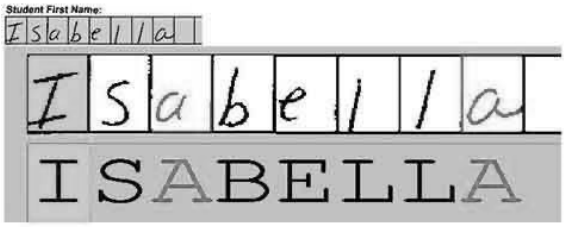
Jellemzők észlelése - Az OCR-programok szabályokat alkalmaznak egy adott betű vagy szám jellemzőire vonatkozóan. A jellemzők között szerepelhet vonalak vagy görbék száma, helyzete. Például az „A” nagybetű két átlós vonalként tárolható, amelyek a középső részen vízszintes vonallal találkoznak. [7]

2.1.2. ICR

ICR (Intelligent Character Recognition), működésére példa egy űrlap, amelyet kézzel kitölthetünk és az ICR segítségével kereshető szöveggé alakíthatunk. OCR kibővített technológiája.

Az intelligens karakterfelismerés (ICR) nem összekeverhető az OCR-rel, az előbbi egy olyan technológia, amely felismeri a kézírásos szöveget. Ennek a technológiának azonban vannak korlátai.

Az ICR programok képesek felismerni az írott karaktereket, amelyek strukturáltak és egyenletesen vannak elosztva, 2.1. ábra szemlélteti ezt. Példa erre az űrlap, amelyen az információkat mezőkbe írjuk, így egyértelműen megadva azok helyét, elosztását. A strukturálatlan vagy szabad formájú kézíráshoz megfelelő módszer lehet, az intelligens szófelismerés (IWR), hiszen itt az egész szót megkísérli felismerni a program, az egyes karakterek helyett. [8]



2.1. ábra *Mezőbe írt szöveg, ICR megvalósítás példája [27]*

2.1.3. IWR

IWR (Intelligent Word Recognition), azaz intelligens szófelismerés, a kézzel írott vagy nyomtatott szavakat felismerésére alkalmas program. Az IWR mesterséges intelligenciát használ a dokumentumok szövegének detektálására, egész szavakat próbál meg felismerni, az egyes karakterek helyett.

Az OCR-hez képest ez egy kontrasztos megvalósítás, hiszen az OCR karakterenként ismeri fel a szavakat. Például, használatakor a „fiú” szó detektálásához a rendszer felismeri az „f”, „i” és „ú” karaktereket. Az IWR a betűket egy szótárhoz illeszti és a különböző algoritmusok alapján észleli a teljes „fiú” szót.

Ez nevezhető egy nyelv specifikus megvalósításnak is, hiszen a módszer lényege, hogy az adott nyelven írt és a rendszer számára elérhető szavakhoz próbálja illeszteni a vizsgált szót és egyezést keresni köztük. [9]

2.2. Magyar nyelv sajátosságai

Anyanyelvünk egyik sajátossága az ékezetes betűk használata, amely a képfeldolgozás szemszögéből nem minden esetben feleltethetők meg egy objektumnak, hanem kettőnek vagy olykor háromnak is értelmezhetők.

A karakterfelismerési képességeket tovább kell fejleszteni, pontosítani kell az algoritmust az ékezetek megjelenésével. Hiszen ezeknek a jeleknek szorosan kell a betűkhöz kapcsolódniuk, hogy az írásfelismerés sikeres legyen. Azonban a képfeldolgozás szempontjából nem csatlakoznak a vizsgált karakterek „testéhez", így könnyen értelmezhetők külön objektumként. Ez azonban hibához vezet.

Az ékezetes karakterek detektálását többféleképpen is megvalósíthatjuk. Egyik megoldás, ha találnunk egy olyan szegmentálási algoritmust, amely az ékezetes betűt egyetlen objektumként észleli, azaz a vizsgálandó objektum reprezentációjához az ékezet is hozzátartozik. Egy másik opció, hogy szegmentálással a betű alapját nyerjük csak ki a képből, majd ezt a neurális hálózat működtetésével felismerjük, később pedig az eredeti közegbe visszatéve megvizsgáljuk a karakter feletti területet. Az ott található képrészletet vizsgálva ékezeteket keresünk és ezek ismeretében módosítjuk a felismert objektumot.

A pontok és vesszők, mint ékezetek és ezek megjelenése a képen további nehézséget is jelenthet. Például az „ü" karakter vizsgálatakor a két pont az előfeldolgozási műveletek során úgy, mint zajszűrés, akár el is tűnhet a képről. Ez olyasfajta információvesztéshez vezet, amit gyakorlatilag nem lehet visszafordítani és szintén hibás megvalósításhoz juthatunk.

A magyar nyelvnek további nehézségei vannak a karakterfelismerés terén, ez a probléma kettős vagy hármas betűk megjelenése miatt áll fent. Értelemszerűen ezt tudjuk úgyis megközelíteni, hogy nem értelmezzük dupla és hármas karaktereknek ezeket, hanem egyszerűen különálló objektumként kezeljük. Ilyenkor a szegmentálási algoritmus pontosításával elérhetjük a kívánt hatást.

Ha a megvalósítás során az egész szavak felismerését választjuk ki, mint kiépítendő egységet, akkor itt is kell számolni a magyar nyelv bizonyos sajátosságaival. Hiszen a toldalékolás, a ragok képzése nagy mértékben befolyásolja szavaink összetételét. Az angolhoz képest ez is nehezíti az általunk kigondolt program létrehozását, ugyanis a rendszernek így sokkal több szót kell ismernie a helyes működés érdekében, nagyobb tanítóhalmazra lesz szükségünk.

2.3. Folyóírás felismerésének nehézségei

Az írott folyószöveg detektálásának problémája visszavezethető az alapvető képfeldolgozási technikák alkalmazásához. Ugyanis ha azt a módszert választjuk, hogy a képen fellelhető karaktereket akarjuk felismerni, majd szöveggé alakítani, akkor egy jól kiépített szegmentáló algoritmusra lesz szükségünk és így már egy egyszerű karakterfelismerésről beszélünk. Amennyiben belegondolunk, mennyi szót tudunk alkotni a magyar nyelvben, a toldalékolási rendszerünknek köszönhetően, könnyen felismerhetővé válik a probléma, hogy az összes létező szó betanítására és felismerésére alkalmas neurális hálózat tervezése igen fáradalmas művelet lenne. Ezen ismeretek tudatában kell egy programot megalkotni, ami a toldalékok adta sokoldalú szavaink felismerését is helyesen elvégzi.

Ránézésre a fennálló nehézséget egy egyszerű eljárással megoldhatnánk, a szavak karakterekre bontásával. Ugyanakkor ez nem ennyire kézenfekvő művelet, egy megfelelő szegmentálási algoritmus kidolgozása igen nehézkes lehet. Vannak esetek amikor még az emberi agy sem képes a szavakat különbontani, ilyen esetekben egy programtól sem várhatjuk el a helyes működést.

A betűkre különítés problémája abban rejlik, hogy az egyes karaktereknek nincs állandó, meghatározható szélességük. Ezenfelül a folyóírással írt szavakban a karakterek között betűkapcsoló szegmensek is fellelhetők, amelyeket nekünk fel kell ismerni és elvonatkoztatni tőlük. Lehetséges megoldás, egy olyan rendszer kiépítése, ahol a neurális hálózat sikertelen detektálását követően a szót ismét karakterekre bontjuk és ezúttal egy másik elosztást használva a betűk kinyerésére. Ennél a megvalósításnál az jelent nehézséget, hogy a program megfelelően döntse el a hiba keletkezésének okát. Vagyis az objektum kidolgozatlansága jelenti a problémát, vagy pedig a szegmentálás során történ a helytelen karakter leválasztás.

Hogyha azonban a szófelismerésnél maradunk szükségünk van egy nagyméretű tanítóhalmazra, amivel a felismerés pontosságát javítani tudjuk. Viszont ezzel a módszerrel nem tudunk minden szót lefedni, elvárás lesz egy adaptív egység bevezetése, amelyet olyan esetekben használunk, amikor a felismerni kívánt szó idegen a rendszer számára. Erre is többféle megoldás készülhet, például, amikor a szótő sikeresen detektálásra kerül a rendelkezésre álló szótár segítségével, viszont a ragok miatt a működést módosítani kell. Esetlegesen előre megadott ragok, végződések, illetve toldalékok, igekötők között vizsgálódva érjük el a kívánt kimenetet. Másik alternatíva lehet, hogy az ismeretlen szótagokat, szórészeket szeparáljuk és karakterenként ismerjük fel, ilyenkor csak ezekre alkalmazzuk a betűkre bontást.

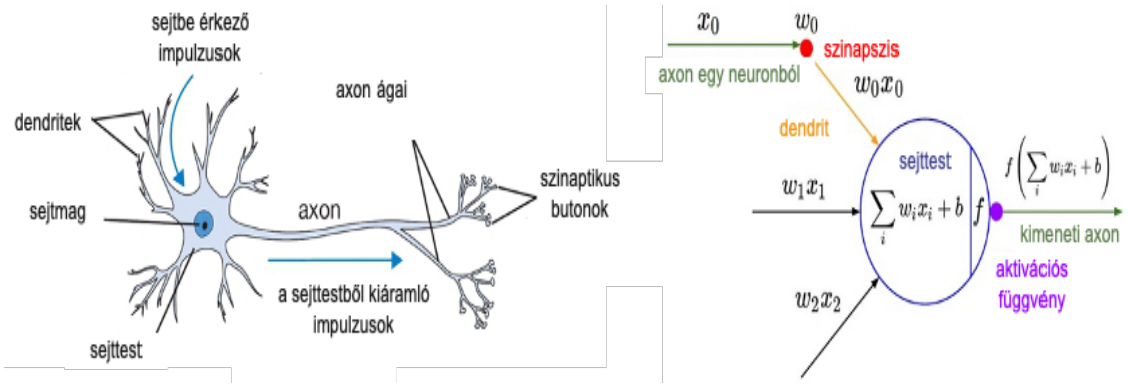
2.4. Neurális hálózatok

Ez a fejezet főként Fazekas I. Neurális hálózatok [1] című és Szabó Cs. Kézzel írt szövegek feldolgozása tanuló algoritmusokkal [10] című munkák alapján készült. Megfigyelhető, hogy a számítástechnika két jelentős része, a képfeldolgozás és a mesterséges intelligencia egyre több területen egyesül. Ez azzal magyarázható, hogy a képfeldolgozásban olyanféle új célok jelennek meg, amelyek megvalósításához elengedhetetlen a rendszer fejlődőképességének javítása, amihez egyfajta gondolkodásmódot, logikát kell a programjainkban tenni.

Manapság a neurális hálózatok felhasználási területe igencsak kibővült, többek között alkalmasak, jelfeldolgozási, karakterfelismerési, képfeldolgozási feladatok megoldására, de jól használhatók olyan összetett nehézségek megoldására, amelyek visszavezethetők két alapvető feladatra: osztályozásra és regressziószámításra.

2.4.1. Neuron

A mesterséges neurális hálók alapegységei a számításokat végző neuronok, ezeket az emberi agy analógiájára próbáltak meg létrehozni, ennek szemléltetése a 2.2. ábrán látható. A számítógépes neurális hálózatok háromféle rétegbe rendeződnek, bemeneti, a számító vagy rejtett réteg és a kimeneti réteg. Ezeket a biológiai idegrendszer fő részeivel feleltethetők meg: a receptorok, a központi idegrendszer és maga az inger. Azonban csak az általános koncepciót sikerült imitálni, a létrehozott neurális hálózatokkal nem lehet a biológiai rendszerek mélyebb tulajdonságait reprezentálni. Ennek ellenére a tanulás képesség nagyon is jól működik a megalkotott rendszerekben.



2.2. ábra Egy biológiai neuron rajza (1) és a mesterséges neuron matematikai modellje (2) [26]

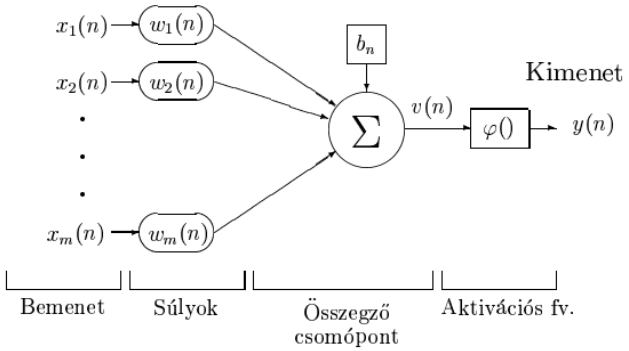
Neurális hálózatot kapunk akkor, ha a számításokat végző alapegységeket, a neuronokat összekapcsoljuk. A neuronok és processzorok közti különbség az, hogy amíg a processzorokat a megfelelő módszerekkel programozzák, addig a neuronok tanítási folyamatokon mennek keresztül.

A hálózatban a rétegek között súlyozott élek találhatók, a neuronok a bemeneten kapott súlyok és értékek segítségével műveleteket végeznek el és a kapott eredmény függvényében továbbítják az ingert a következő réteg felé. Működése oly módon történik, hogy az adott kimenethez tartozó súlyokkal szorozza a bemeneti adatokat, majd az összegükből kiszámolja a neuron aktivációs potenciálját. Ezt már az aktivációs függvény állítja elő a kapott összegből. Majd a meglévő értékhez kiszámoljuk az átviteli függvény értékét és így megkapjuk az eredményt az adott kimenetre. Az aktivációs függvény határozza meg, hogy a neuron az adott bemenetek hatására milyen mértékben érvényesül.

Mindegyik hálóban megtalálható az input réteg, ebbe a rétegbe kerülnek a bemeneti adatokat és ez juttatja el a többi réteg neuronjai felé az adatokat. Minden esetben megtalálható minimum egy rejtett réteg, ami a valós számolásokat végzi. A legvégén áll az output réteg, ami a rejtett rétegek értékeit összegyűjti.

Az egyetlen neuronból álló neurális hálózatot perceptronnak nevezzük, ennek a szemléltetését a 2.3. ábra végzi, ezek felhasználási területe a lineáris szeparálási feladatok megoldása. A perceptron a következő részekből áll össze:

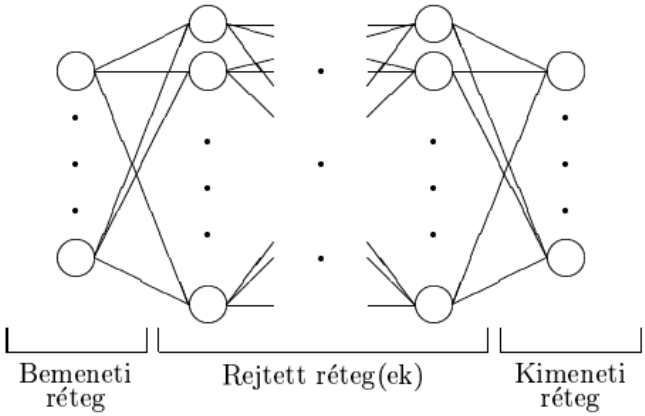
- **Bemenet***: Egy ismert m dimenziós vektor x(n)*
- **Súlyok***: A valós súlyok nem ismertek, ezek csak az n-edik közelítései a valódi súlyoknak, amelyek rendre a $w_1(n)$, ..., $w_m(n)$ értékek*
- **Torzítás***: A torzítás igazi értékét sem ismerjük, helyette annak n-edik közelítését, a b(n)-t használjuk*
- **Összegző csomópont***: $v(n) = b(n) + \sum_{i=0}^{m} w_i(n)\, x_i(n)$ – alábbiak szerint a bemenet adatainak súlyozott összegét képezi*
- **Aktivációs függvény***: Egy olyan φ függvény, amely a v(n) összegzett értékét alakítja át, érteke pedig a feladat szempontjából alkalmas intervallumba esik*
- **Kimenet***: Ez a neuron által a bemeneti x(n) értékhez rendelt érték*
  y(n) = φ(v(n))



2.3. ábra: A perceptron felépítése *[1]*

Vannak olyan esetek amikor egyetlen perceptronnal nem tudjuk megoldani az adott problémát. Ekkor az az eljárás, hogy egy helyett több perceptront építünk ki, ezt a felépítést a 2.4. ábra jeleníti meg. Így az eredeti lineáris szeparálásnál jóval bonyolultabb feladatokat is meg tudunk oldani.

Az egyes rétegek kimenetei kapcsolását többféleképpen is megoldhatjuk. Lehetséges módszer, hogy a következő rétegben található összes bemenetre rákötjük az előző réteg kimeneteit. Azonban olyan kapcsolást is létrehozhatunk, amelyben egy perceptron a következő rétegből csak bizonyos perceptronokkal van kapcsolatban. Ezek a kötések nagyban meghatározzak a rendszer működését, ennél fogva nem mindegy, hogyan alakítjuk ki a rétegek kapcsolatát. Lényeges megemlíteni azt is, hogy minden neuronnak megvan a maga aktivációs függvénye és a súlyai.



2.4. ábra: Egy általános felépítésű neurális hálózat *[10]*

Jelenleg még nem létezik általános eljárást a hálók optimális méretének meghatározására, ebből adódóan leginkább próbálgatásos módszerrel tudjuk meghatározni, hogy az adott feladat megoldásához mekkora és milyen felépítésű háló kiépítésére van szükségünk. A kiinduló mintát lényegében kétféleképpen határozhatjuk meg. Egyik lehetőség, hogy egy kisméretű hálóból indulunk ki és azt bővítjük, amíg a feladat megoldása szempontjából már megfelelő a hálózat, elértük a kívánt működést és hatást. Másik lehetőség egy nagyobb kiinduló háló választása, ami már képes a feladatot megoldani és ezt csonkolhatjuk addig, amíg még mindig megfelelő marad a feladat megoldására.

A neurális hálók megfelelő méretének kiválasztása azért fontos, mert a háló méretének növelésével csökkenhet a sebesség. A cél az a legkisebb méret megtalálása, ami még alkalmas a megoldás előállítására.

2.4.2. Hálók felépítése

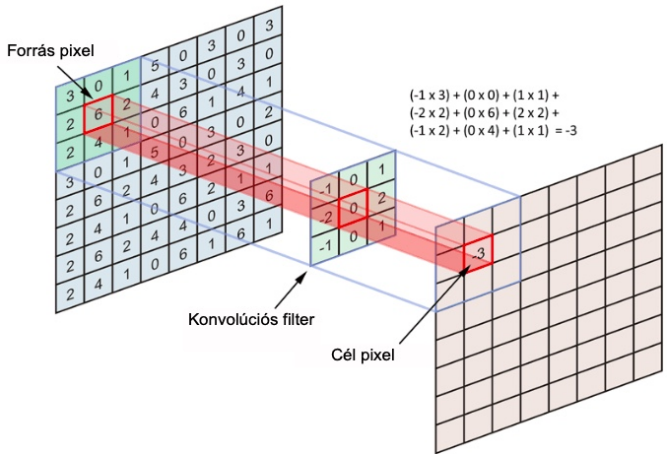
A neurális hálózatok a neuronok összekapcsolásával hozhatók létre, így ezekben a hálókban rétegezve jellenek meg a neuronok. A megegyező rétegben lévő neuronok vagy a kezdeti értékektől kapják az inputot, vagy egy előző rétegtől. A kimeneteik pedig vagy a következő réteg input adatai, vagy az egész háló kimeneti értékei. A hálók funkcióját az összeköttetések szerkezete és a súlyok értékei szabják meg.

Összességében elmondható, hogy egy neurális hálót jellemző hiperparaméterek megtervezése és kiválasztása a legkörülményesebb feladat. A hiperparaméter definíció szerint a paraméter paraméterei, vagyis jelen esetben azok a tényezők, amelyek a rendszer felépítését adják. Hiperparaméter akkor van jelen, ha egy feladat megoldásához szükség van egy paraméter ismeretére, de ezt a paramétert, csak az értékének meghatározására szolgáló egyenletben lévő paraméterek (hiperparaméterek) megoldásával lehetne megtenni. Ezalatt érthetjük, a használni kívánt rétegek számának meghatározását, illetve, hogy melyik rétegbe mennyi neuron kerül. Emelet az adott rétegek aktivációs függvényeinek definiálása és az esetlegesen alkalmazott szűrő méretének és felépítésének megadása is ide tartozik. A paraméterekkel általában a modell adatokra való illeszkedését, a hiperparaméterekkel pedig a modell flexibilitását, reprezentációs erejét tudjuk állítani. A paramétereket a tanítóalgoritmus hangolja be, a hiperparamétereket általában mi választjuk meg kézileg, valamilyen heurisztika alapján.

A hálózatokat a kapcsolatok szerint kettő kategóriába csoportosíthatjuk. Az előrecsatolt hálózatokban (feed forward neural network) minden rétegből a függés csak az utána lévő rétegbe mutathat. A jelek egy irányba haladnak végig a hálón, a bemeneti rétegekből a kimeneti réteg felé. A másik csoport a visszacsatolt háló (recurrent neural network – RNN), ez a fajta megvalósítás annyiban tér el az előre csatolt hálótól, hogy hurok található benne, képes a korábbi bemenetekre adott választ is figyelembe venni. Azaz egy adott réteg neuronjának az input adatai nem csupán egy előző réteg kimeneteiből származhat, hanem akár ugyanaz a neuron egy régebbi kimentéből is lehet, vagy egy azonos rétegen belüli neuron kimentéből vagy akár egy későbbi réteg kimeneti értékéből.

Funkcionalitás szerint azonban már több típusa létezik a neurális hálóknak, az előbb említett hálók is eltérő használati móddal rendelkeznek. Az utóbbi években számos olyan hálóstruktúrát alakítottak ki, amelyek kifejezetten egy-egy problémakör megoldására lettek létrehozva. Ezek közül, amik számunkra említésre érdemesek, azok a konvolúciós neurális hálózatok (convolutional neural network – CNN), mivel ezek elsősorban képfeldolgozási funkciókkal rendelkeznek, azonban egyéb inputot is képesek értelmezni (videó, hang). Használatakor a bemeneti kép adatait pixel szinten keresztül küldjük a hálón, mellyel végső célunk a képről szerzett információk osztályozás.

A CNN hálók működését kettő részre lehet bontani, a jellemzők felderítésére és az osztályozásra. Az előbbi esetén a kép egységein átlagolásokat és összevonásokat folytatunk, melyek végeztével megkapjuk a képen lévő speciális tulajdonságokat. Ezek megléte után következik asz osztályozás, azaz a jellemzők vizsgálatával értelmezzük a képet. Ezekben a hálókban az inputként érkező adatot nem egész terjedelmében elemezzük, hanem csak egy-egy része kerül bemásolásra, amíg végül nem végzünk a teljes bemeneti adathalmazzal, ennek a működését a 2.5. ábra vázolja fel. Például, ha egy 1 000 x 1 000 pixeles kép a beviteli adat, akkor nem egy 1 000 000 neuronból felépített réteget építünk ki, hanem egy kisebb méretű réteget használunk, ahova egy szűrőn keresztül áramlik az adat. Ilyen lehet egy 100 x 100 képpontos szűrő, ami részleteiben továbbítja a bemeneti képről az információkat. [11]



2.5. ábra *A konvolúciós neurális hálózatok által használt szűrőegység működése [11]*

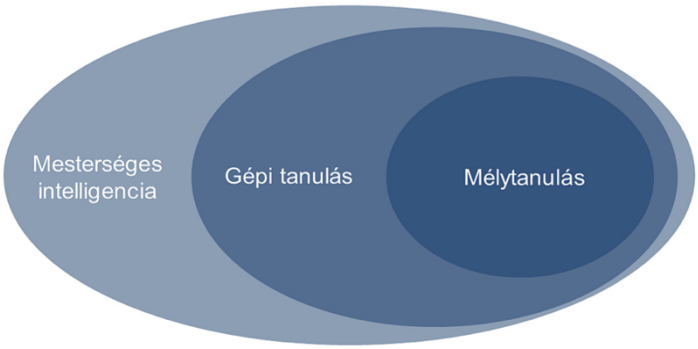
2.4.3. Mélytanulás

Klasszikus értelemen a mesterséges intelligenciáról elmondható, hogy képes a környezete ismeretében döntéseket hozni, mely döntések a természetes intelligenciához hasonló eredménnyel rendelkeznek, ezen felül képes alkalmazkodni a környezet változásához. A mesterséges intelligenciának hangsúlyos ága a gépi tanulás, a mélytanulás pedig a gépi tanulás egy szegmense, amely mesterséges neurális hálózatokon alapszik.

A gépi tanulásnál megadott kritériumok szerint tudunk historikus adatokból predikciót előállítani és döntést hozni. A gépi tanulásnál az mesterséges intelligencián alapuló rendszerek betanítása zajlik oly módon, hogy azok a szerzett tapasztalatokból felismerjék a mintákat, illetve javaslatokat tegyenek és alkalmazkodjanak.

A mélytanulás megjelenésével a digitális rendszerek nem csak az előírások alapján képesek reagálni, hanem a mintapéldákból ismereteket szerezve az emberekéhez hasonló reagálásra, viselkedésre és hatékonyságra is képesek. Használatával akár olyan minták is előtérbe kerülhetnek egy vizsgált adathalmazban, melyek az emberek számára kevésbé észrevehetők.

A mélytanulás, így a neurális hálózatok is a gépi tanulás egy részének tekinthetők, alkalmazási területeik és felépítésük között azonban jelentős különbség figyelhetők meg. A neurális hálózatokról elmondható, hogy felépítésben összetettebbek, illetve sokkal inkább képesek önállóan működni. Egy neurális hálózat képes lehet megállapítást tenni arról, hogy az eredményei, előrejelzései milyen pontossággal rendelkeznek, míg egy gépi tanulási modell esetében ezt emberi beavatkozással érhetjük el. Továbbá a neurális hálózatok képesek tanulni és tanult döntéseket hozni. A gépi tanulási modellek olyan döntéshozatalra képesek, amire már előzetesen betanították. A 2.6. ábra a mesterséges intelligencia, a gépi tanulás és a mélytanulás kapcsolatát ábrázolja.



2.6. ábra *A mesterséges intelligencia, a gépi tanulás és a mélytanulás kapcsolata [12]*

A hagyományos és a mély neuronháló strukturálisan annyi differenciával bír, hogy hatványozottan több rejtett réteg van a mély neurális hálózatokban és ezekben a rétegekben nagy számban vannak tanítható paraméterek. A mélytanulási modellek betanításához gyakran nagy mennyiségű adatra van szükség, előnyei is csak sok tanító adat esetén mutatkoznak. Ha nincs elégséges számú minta, akkor lehet, hogy az ismert adatokra túlságosan illeszkedni fog a háló és így nem ad egy megfelelően általánosított modellt. A mély neuronháló tanítása számításigényes, ezáltal csúcskategóriás számítási erőforrásokra (GPU) és hosszabb betanítási időre van szüksége. A betanítási lehetőségéről bővebben az ezt követő fejezetben lesz szó. A mély hálók jelenlegi sikeréhez új algoritmusok, sok tanító adat és a számítási kapacitás megfelelő együttállása volt a kitétel. [12]

2.4.4. Betanítási folyamat

Amint a neurális hálózat felépítésével végeztünk áttérhetünk a tanítás fázisára, azaz a súlyok és a küszöbértékek apró módosításaival elérhetjük, hogy a hiba értékét csökkenteni tudjuk, jobban mondva a kapott eredmény és az elvárt érték közti differenciát redukáljuk. A mély tanulási modellek betanítására különféle stratégiákat és módszereket lehet alkalmazni. A fejezetben leírtak főként Horváth G., Patak B. Neurális hálózatok [13] című munka alapján készültek.

Mivel a neurális hálók az emberi agyat reprezentálják, ezért a tanuló algoritmusok is hasonlóképpen működnek, akárcsak az emberek esetében. Ez a folyamat nagy mennyiségű és egyenértékű utasítást jelent eltérő adatokon. A neuronháló rendeltetéséből adódóan jól alkalmazható általánosításra, továbbá olyan bemenetre is képes kielégítő kimenetet adni, amit a tanítás során nem látott, így a hibás, zajos adatokkal is megfelelően tud dolgozni.

A tanítási módszereket többféle osztályba tudjuk sorolni, ezek közül a kettő legelterjedtebb: felügyelt, ellenőrzött tanulás és a nem felügyelt, nem ellenőrzött tanulás. Ezek közti különbség a kimeneti adatok rendelkezésre állása, amíg a felügyelt tanulás esetében a be- és kimenő adatpárok elérhetők, addig ezek a nem felügyelt tanulás esetén elmaradnak, csak az input adatok vannak a birtokunkban.

A felügyelt tanulás közé olyan módszereket sorolunk, amelyekhez külső beavatkozás szükséges, legyen ez egy betanítást végző személy vagy egy átfogó információhalmaz használata. Ilyen módszer lehet a megerősítő tanulás, ahol a hálózat egyes súlyait a kimeneti érték jóságának függvényében növeljük vagy csökkentjük. Egy további hasonló jellegű módszer a hibajavító tanulás, ahol a kapott output és az elvárt érték eltérése alapján módosítjuk a súlyokat.

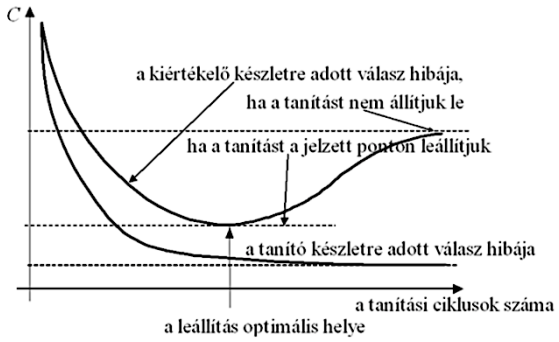
A súlyok előnyös kalibrálásához több szabály is létrejött az utóbbi években. A delta-szabály megváltoztatja a súlyokat a kimenet és a cél állapot közti különbség hibájának figyelembevételével, vagyis az eltérés minimalizálása a célkitűzés. A backpropagation, vagyis visszaterjesztés alkalmazásánál a hibafüggvény minimumhelyét keressük, tehát azokat a súlyokat, ahol a hibafüggvény a legkisebb, ezt gradiens módszer felhasználásával érhetjük el.

A felügyelet mentes tanítás során nem áll rendelkezésünkre az adott bemenethez várt kimeneti érték, így itt a neurális hálónak kell egyfajta viselkedést kialakítania, mindezt úgy, hogy semmiféle visszajelzés nem érkezik a környezetéből. Ilyen esetben a hálónak kell összefüggéseket keresni a bemeneti adatok között. Fel kell deríteniük, hogy az input jelekben fellelhető-e valami hasonlóság, adatok közötti kapcsolat. Ezek ellenére a rendszer képes a saját maga kalibrálására, hogy a kategorizálás eredményesebb legyen.

A megerősítő tanulási módszer főként gépi tanulás kapcsán fordul elő, ahol a modell próbálkozásos módszer használatával oldja meg a feladatokat. Nem adathalmazokat használ, hanem olyan információkat, amelyeket a környezetből gyűjt össze.

Egy neurális hálót abban az esetben nevezünk betanítottnak, amikor a tanító adatokra megfelelő kimenettel válaszol. Azonban még megtörténhet, hogy a tesztadatokra téves viselkedést produkál, ez értelemszerűen problémát jelent, ilyenkor még finomítani lehet a rendszeren.

Az általánosítóképesség feltárásához az adatainkat három részre célszerű osztanunk, így beszélhetünk tanító, validációs és teszt halmazokról. Magától értetődően a tanító készletet a modell felépítésére használjuk, a tanításra. A kiértékelő készlet felhasználásával hangolhatjuk a hálót, így csökkentve a túltanulás lehetőségét, ennek fontosságát a 2.7. ábra demonstrálja. A teszthalmazon pedig kiértékelhetjük a teljesítményt, így ezt a készletet ideális esetben egyszer vesszük igénybe, méghozzá a tanulási művelet végén. [13]



2.7. ábra *Tanulási görbe: a korai leállítás szerepe a túltanulás elkerülésében [13]*

Overtraining, vagyis túltanítás olyan esetben következik be, ha a tanító halmaz adataira már csak minimális hibájú visszajelzéseket kapunk, mialatt a validációs halmazra egyre nagyobb hibával reagál a modell. Ez akkor jön létre, amikor a modell válaszai drasztikusan beleillenek a véges számú tanító pont által megszabott régióba. A korai leállás célja a túltanulás elkerülése. Akkor kell a tanítást leállítani, hogyha a kiértékelő halmazra kapott válasz hibája növekedni kezd, tehát mielőtt a háló általánosító képessége romlásnak indulna.

2.5. Technológiai áttekintés

A neurális hálók létrehozása egy mély és összetett téma, a mesterséges neuronhálók alapvető koncepciója az algoritmusok használata az adatelemzéshez és a modellek létrehozásához. Az algoritmusok megírásához szükséges a hozzáillő programozási nyelv kiválasztása, amivel egyszerűen létre tudjuk hozni a szükséges modulokat. Szükség van olyan mélytanulási keretrendszerekre, amelyek leegyszerűsítik a neurális hálózatok betanítását és az adatok gyűjtésének folyamatát. Ebben a fejezetben olyan technológiák kerülnek bemutatásra, amik segítségével gördülékenyenebben tudjuk létrehozni a tervezett neurális hálózatunkat és használatuk elengedhetetlen lesz a későbbiekben.

2.5.1. Programozási nyelv

Számos olyan programozási nyelv adott számunkra mellyel elkezdhetjük egy mesterséges neuronháló megalkotását. Jelenleg a legnépszerűbb ezek közül a Python, ami nem csoda, ha figyelembe vesszük a mélytanulási Python keretrendszerek fejlődését az elmúlt években. Emellett szokták említeni a C és a C++ nyelvet, illetve az R nyelvet is.

A C++ egy magas szintű objektum-orientált programozási nyelv, gyorsabb végrehajtási idővel, mint a legtöbb más nyelv. Így ideális a gépi tanuláshoz, ahol a gyors és következetes visszacsatolás hangsúlyos. A C++ számos könyvtári támogatást is kínál a gépi tanuláshoz.

Az R nyelvet numerikus és statisztikai kérdések megoldására terveztek. A mesterséges intelligencia megjelenésével a gépi tanulás és az adattudomány rendkívüli módon megnövelte vonzerejét. Nem igényel magas szintű szakértelmet, hiszen számos könyvtárat és erőforrást tartalmaz, amelyek gyakorlatilag a szoftverfejlesztési folyamat minden fázisában segítséget nyújthatnak.

A Python általános célú, nyíltforrású, magas szintű programozási nyelv. Ez egy egyszerű, könnyedén tanulható nyelv, ahol nincsenek fordítási lépések. A Python nagyméretű standard könyvtárral rendelkezik, az ehhez készült mesterséges intelligencia és gépi tanulás kódtárak nagy mennyiségű és egyre növekvő ökoszisztémával rendelkeznek. [14] Felhasználása, módosítása és terjesztése ingyenes, egyúttal korlátozás nélkül használható kereskedelmi projektekben is. Egyaránt megfelel néhány soros scriptek és több tízezer soros komplex projektek létrehozására is. A nyelven írt kód tömör és jól olvasható, folyamatosan fejlődő nyelv. Rengeteg fejlesztő használja világszerte, ezért az interneten rengeteg példaprogram, dokumentáció elérhető, ami segít a fejlesztésben és a tanulásban. [15]

A Python sokoldalúságát jellemzi, hogy több mint 125 000, harmadik féltől származó könyvtár érhető el az interneten. Ezeket főként speciális célokra használják: webfejlesztés, szövegfeldolgozás, mesterséges intelligencia, gépi tanulás. Mivel az adatelemzés és mesterséges intelligencia egy sarokköve ez a nyelv, ezért az ilyen jellegű projektek esetén nélkülözhetetlen eszközök lehet a NumPy, matplotlib vagy a pandas könyvtárak használata. Illetve a TensorFlow, ami egy szoftverkönyvtár a gépi tanulási algoritmusok leírására és végrehajtására. [16]

2.5.2. Keretrendszer

A mélytanulás során nagy mennyiségű strukturálatlan adat kerül feldolgozásra, ezért olyan keretrendszerre van szükség, amely irányítja a rétegek közötti interakciót, emellett a bemeneti adatokból való tanulással és az önálló döntések meghozatalával gyorsítja a modellfejlesztést. A mélytanulás keretrendszere interfészek és könyvtárak kombinációjából áll, amivel lehetőséget kapunk a modellek gyors és pontos meghatározására, illetve betanítására. Ezen a területen a legmeghatározóbb tanulási keretrendszereknek a TensorFlow-t és a PyTorch-t tekintjük.

A PyTorch egy ingyenes és nyílt forráskódú szoftver, segítségével egyszerűbb a neurális hálózat fejlesztése és betanítása, még mélytanulási tapasztalat nélkül is. A PyTorch keretrendszert leginkább olyan alkalmazásokhoz használják, mint például a számítógépes látás és a természetes nyelvi feldolgozás. Python alapú, egyszerű és gyors, ez a keretrendszer megfelelő a tanuláshoz, használathoz és hibakereséshez.

A TensorFlow egy szoftverkönyvtár a gépi tanulási algoritmusok leírására és végrehajtására, amikor a mélytanulásról beszélünk a TensorFlow gyakran az elsőként említett keretrendszer. A TensorFlow magas és alacsony szintű API-k fejlesztésére használható, lehetővé téve az alkalmazások szinte bármilyen eszközön történő futtatását. Bár a Python az elsődleges nyelve, más programozási nyelvekkel is elérhető és vezérelhető, például C++, Java és JavaScript használatával. Roppant flexibilis, széles körű algoritmusok megvalósítására alkalmas, beleértve a sokrétegű neurális hálózat működtetéséhez szükséges algoritmusokat. Használják, például a beszédfelismerésben, a számítógépes látásban, a robotikában, az információ kinyerésben. A TensorFlow rendszer sokrétegű neurális háló modellek készítésén kívül széles körben alkalmazható más célokra is, ideértve a más jellegű gépi tanulási algoritmusok és a különféle numerikus számítások implementálását. [17]

Keras is egy Python alapú könyvtár, a Keras API magas szinten működik és lehetővé teszi az integrációt alacsony szintű API-kal, mint például a TensorFlow. Különféle megvalósításokat tartalmaz a neurális hálózatok alap elemeire úgy, mint a rétegek, célfüggvények, aktiválási függvények és matematikai optimalizálók. Mindemellett támogatja a konvolúciós neurális hálózatokat és az ismétlődő neurális hálózatokat. [4]

2.5.3. Futtatókörnyezet

A neurális hálók felépítéséhez és tanításához egy olyan futtatókörnyezetre van szükségünk, ahol maga a szoftverkörnyezet, vagyis az operációs rendszer, illetve a telepített programkönyvtárak, segédprogramok, esetenként rendszerbeállítások képesek ellátni a modell megalkotásához szükséges feladatokat. Mindemellett a modell tanítása bizonyos elvárásokat támaszt a hardver felé is, hogy az képes legyen a művelet végrehajtására. Tehát a megfelelő memória, GPU jelenléte is elvárás.

Ezek mindegyikére megfelelő alternatívaként szolgál a Google Colab, a Colaboratory vagy röviden „Colab" a Google Research terméke, ami egy ingyenes felhő szolgáltatás. A Colab lehetővé teszi, hogy bárki tetszőleges Python kódot írjon és futtasson a böngészőn keresztül, különösen alkalmas gépi tanulásra, mély neurális hálózatok fejlesztésére és adatelemzésre.

Technikai szempontból a Colab egy hosztolt Jupyter notebook szolgáltatás, amelynek használata nem igényel beállítást, ugyanakkor ingyenes hozzáférést biztosít a számítási erőforrásokhoz, beleértve a GPU-kat is. Nincs szükség telepítőre sem, a Google Drive-ból érhető el minden. Lehetőséget nyújt a Google Drive-on keresztül a képek, szövegek és egyéb, a modell tanítására és tesztelésére alkalmas adatok tárolására, illetve egyszerű felhasználására. Számos előre telepített adat könyvtárat tartalmaz, például pandas, NumPy, Matplotlib, OpenCV. Mindezek mellett több előre telepített gépi tanulási könyvtárat is biztosít, mint például a Keras, a TensorFlow és a PyTorch. [18]

2.5.4. Konklúzió

A Google Colab használata számunkra mindenképp célszerű, hiszen rengeteg segítséget nyújt mind a kezdeti mind a későbbi fázisokban. A telepített adatkönyvtárak mint pandas, NumPy, Matplotlib jó eséllyel szükségesek lesznek a projekt során. Illetve az OpenCV könyvtár, ami a képek előfeldolgozásában szolgálhat segítségűl. Ezek mellett az adatok feldolgozására és a modell betanítására a Colab által használatba vehető GPU is jelentős támogatás ad. A tanító adatok tárolására és felhasználására a Colab-on keresztül egyszerűen elérhető Google Drive ad megfelelő környezetet. A TensorFlow és a Keras olyan könyvtárakm amelyekkel a hálózat felépítése, tanítása és használata egyszerűen kivitelezhetők, továbbá a Colab használatában nehézség nélkül elérhetők. Ezáltál egyértelművé válik a Python, mint programozási nyelv kiválasztása, mindezek kominációja pedig egy optimális kiindulási pont a mély neurális háló megalkotásához.

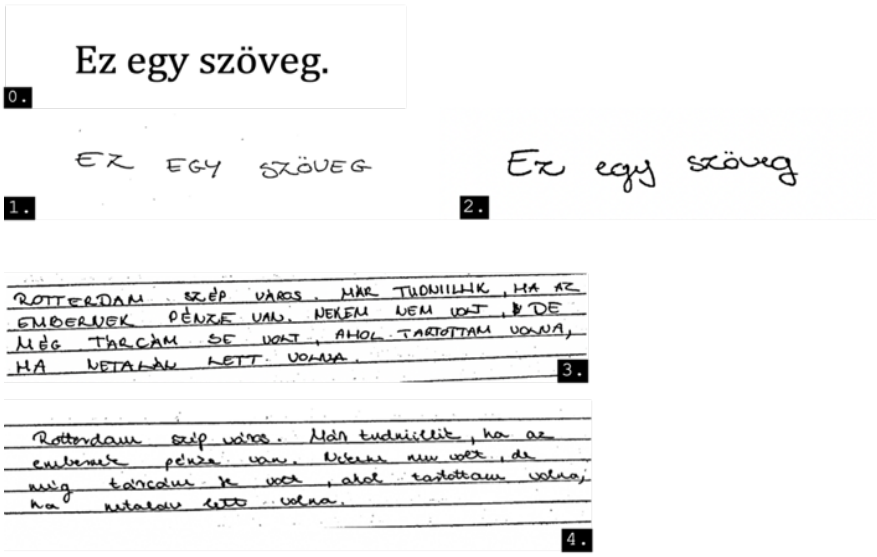
3. ÍRÁSDIGITALIZÁLÁS A GYAKORLATBAN

Ebben a fejezetben olyan írásfelismerő programok tesztelése és összehasonlítása történik, melyek az átlag felhasználó számára a legkönnyebben elérhetők és jellemzően ez első helyen szerepelnek a keresési listákon. A programok vizsgálata többféle bemeneti kép felhasználásával valósult meg. A cél az volt, hogy felmérjük a felkutatott és kipróbált rendszerek milyen hatással képesek felismerni a folyóírást.

3.1. Célkitűzés

Jelenleg sok elérhető adatkinyerő, szövegfelismerő program létezik, ami a szkennelt dokumentumok vagy képek elemzésére hivatott. Azonban ezek nagy része csupán a nyomtatott, illetve nagybetűkkel írt szavak detektálására alkalmas. Elsősorban olyan szoftver keresése volt a cél, ami megfelel az egyes kritériumoknak, azaz ismerje fel a magyar szöveget és a folyóírással írt szöveget megfelelően tudja detektálni. Több lehetséges alternatíva is megvizsgálásra került, jelentős szempont volt azonban az is, hogy a program hozzáférése ingyenes legyen. Mindezek alapján elmondható, hogy a legjobb eredményeket nem az ingyenes szoftverekkel tudjuk elérni.

A programok tesztelése a 3.1. ábrán látható képekkel történt. A bemenetnek átadott képek fokozatosan „bonyolódnak”, hiszen az elején átadott kép egy egyszerű nyomtatott szöveg, majd egy nagybetűs szöveg zaj nélkül, míg az utolsó képen a szöveg már folyóírással írt, ezen kívül a vonalazás is nehezítheti a kép elemzését. A végcél egy olyan program találása volt, amely képes a 3.1. ábrán látható utolsó, 4. képen lévő szöveget megfelelően detektálni. A következő fejezetek az egyes programok kimeneteit mutatják be.



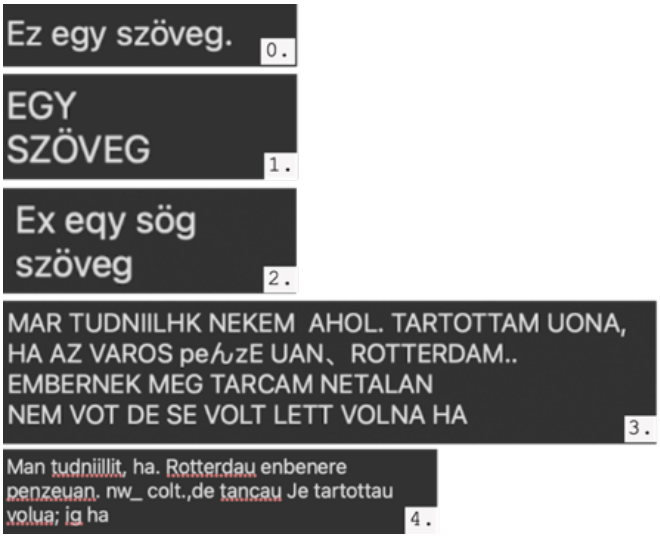
3.1. ábra Saját képek használata bemenetként

3.2. Teszteredmények

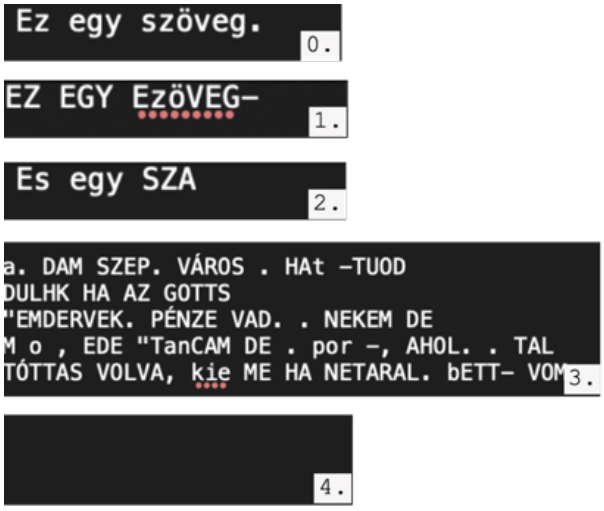
A választott programok kipróbálása az általam leírt, majd lefényképezett szövegekkel zajlott. A szoftverek használata minden esetben egyszerűnek mondható, bemenetként képet, illetve pdf formátumot fogadnak. Az írások egyesével kerültek a szövegfelismerő programok elé, amiknek a kimeneteit a következő ábrák szemléltetik. Ezek elemzésre majd összehasonlításra kerültek.

PDFF OCR X eredményei a 3.2. ábrán láthatók. Ezekből arra a következtetésre jutunk, hogy a nyomtatott szöveg felismerése részben sikeresnek mondható, azonban a folyóírást már nehezen kezeli a program. Ezenkívül a képen található zajok kiszűrésére nincs megoldása a szoftvernek, így ezeket egy előfeldolgozással kellene eltüntetnünk. **Text Scanner OCR** kimenetei a 3.3. ábrán a láthatók. Ez szintén egy OCR program, ami a gépelt, kézzel írt vagy nyomtatott szövegek felismerését végzi. A folyóírás felismerése itt sem érte el a várt eredményt. A legrosszabb eredményt a **LightPDF** programmal értük el, hiszen ennek a használata során mindössze a nyomtatott szöveg képéről tudott érdembeli detektálást végezni. A többi bemenet esetén a program nem adott értékelhető kimenetet. A **Sejda** szoftverével szintén megragadtunk az egyszerűbb OCR-ek szintjén, a nyomtatott szöveget egyszerűen fel tudta ismerni, azonban a kézzel írt szövegekkel már nem tudott a program az elvárásainknak megfelelően működni. A kimeneti szövegek a 3.4. ábrán láthatók. A **Pen to Print** egy telefonos szoftver, ami kézírásos OCR-t használ, azonban még itt sem teljesül az elvárt írásfelismerés, az eredményeket a 3.5. ábra szemlélteti.

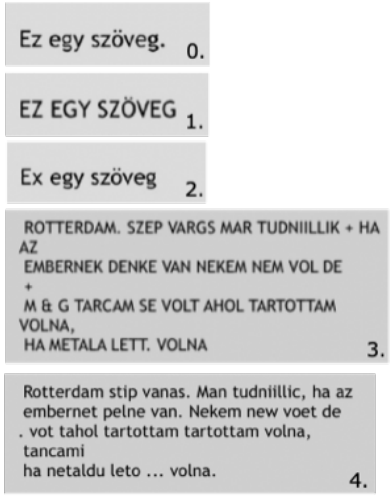
3.2. ábra PDFF OCR X kimenetei



3.5. ábra Text Scanner OCR kimenetei



3.3. ábra Sejda OCR kimenetei



3.4. ábra Pen to Print Handwriting OCR kimenetei

3.3. Áttekintés

A felkutatott szoftverek mindegyike OCR technológián alapul, ez is azt mutatja, hogy a gépelt vagy nyomtatott szöveg képeinek gépi kódolású szöveggé való konvertálása tekintetében az egyik legsűrűbben használt módszernek tekinthetjük.

Az eredmények tanulmányozása után azt a konklúziót vonhatjuk le, hogy a nyomtatott szöveg felismerése sikeresnek mondható valamennyi program esetében. Oly módon, hogy azok képesek voltak egy akár optimális érték kivitelezésére, azonban a további tesztbázisokra már eltérően reagáltak a programok. Az OCR által alkalmazott mintafelismerés, illetve a jellemzők észlelése megfelelő a nyomtatott szöveg detektálására, viszont a továbbiakban célszerű lenne áttérni az intelligens szófelismerésre, ami mesterséges intelligencia használatán alapszik. Azonban ilyen szoftverek jelenleg nem állnak rendelkezésre hétköznapi felhasználásra, nincsenek programok, amelyek ingyenesen elérhetőek lennének. A másik akadály, hogy a magyar nyelvre alkalmazható szoftverek is csak véges számban vannak jelen a piacon.

Az összegyűjtött anyagok elemzését a 3.6. táblázat demonstrálja, illetve mutatja a kipróbált szoftverek eredményeit és a főbb technikai adatait. Ebből látszik, hogy nehéz olyan szoftvert találni, ami képes a folyóírással írt szöveg felismerésére, hiszen a legtöbb program OCR technológiával oldja meg a problémát, ami pedig a nyomtatott szöveg detektálásán túl nem alkalmas másra. A nagybetűkkel írt szöveg esetében voltak eredmények melyek sikeresnek mondhatók, azonban teljes megoldást nem tudtunk elérni. A folyószöveg esetén a kimenetek távol estek az elvárt eredménytől, még ha voltak is jobb kimenetellel rendelkező programok.

	sikeres detektálás				
	nyomtatott szöveg esetén	nagybetűkkel írt szöveg esetén	írott folyószöveg esetén	input formátum	ingyenes
PDF OCR X	igen	részben	nem	image/pdf	igen
Text Scanner OCR	igen	részben	nem	image	részben(4scan/nap)
LightPDF	igen	nem	nem	image/pdf	igen
Sejda	igen	nem	nem	pdf	részben(3scan/óra)
Pen to Print	igen	részben	részben	image/pdf	igen

***3.6. táblázat** OCR-ek összehasonlítása*

4. RENDSZERTERV

Ebben a fejezetben a rendszerünk tervezete kerül meghatározásra a projekt által eddig összegyűjtött ismeretek alapján. A szakdolgozat elején kifejtett célok szerint a feladatunk az írott folyószöveg detektálására fellelhető aktuális eredmények feltárása és elemzése után egy olyan rendszer létrehozása, amely alkalmazható a folyóírás felismerésére. A végcél, hogy a magyar nyelven írt folyószövegről készült képet algoritmusok segítségével elemezni tudjuk és egy olyan kimenethez jussunk, ami a lehető legpontosabban reprezentálja a bemeneti képen lévő szöveg tartalmát.

4.1. Tervezés

A megvalósításhoz elengedhetetlen az ideális tervezés kivitelezése, ehhez pedig az elvégzendő célok definiálása szükséges. A kitűzött cél egy olyan rendszer felépítése, amely a bemenetként kapott kép fájlból képes olyan szöveget kinyerni, ami a lehető legjobban megközelíti a kép tényleges tartalmát és kereshető .txt formátumban visszaadja azt.

Az irodalomkutatás segített rávilágítani minden olyan részletre, ami a rendszer tervezése során a segítségünkre lesz. Az egyik fő kérdés a detektálást végző neurális hálózat kialakítása, erre kétféle lehetséges megvalósítás is tanulmányozásra került. Ezek pedig a karakterekre bontással létrehozható egyszerű karakterfelismerő, illetve a szófelismerésre kiépített neurális háló.

A szavak karakterekre bontásával visszakanyarodunk a karakterfelismeréshez, ennél a megvalósításnál a folyóírás szegmentálása válik a projekt egyik meghatározó elemévé. Hiszen a meglévő szavakat betűkre kell különíteni, amit aztán karakterfelismerés segítségével detektálni tud a rendszer. A szegmentálás során kialakítható a betűk strukturált és egyenletes elosztása, így azok egyesével megvizsgálhatók lesznek. A magyar nyelv adta nehézségek ezáltál már kevésbé lesznek jelentőségteljesek, hiszen a toldalékok nem befolyásolják a programot a felismerésben. Az ékezetes betűk detektálása pedig egyszerűen megoldható a megfelelő előfeldolgozási lépések beiktatásával, továbbá elégséges adat megléte esetén a háló tanítása szokványos műveletté válik.

Azonban egy megfelelő szegmentáló algoritmus kiépítése akár egy külön projekt terjedelmét is magába foglalhatná, illetve a lényeget is egy másik irányba vinné, ami jelenleg a mély neurális hálók használata. Igaz elviekben a későbbi munkánkat megkönnyítené, ha a szavak karakterekre bontva állnának a rendelkezésünkre, de még így is egy jelentős akadályt kellene leküzdenünk. Mindez ismeretében nem célszerű a rendszerünket ezek mentén felépíteni, szükségszerű lehet a másik alternatíva felvázolása.

Amennyiben az egész szavak felismerését válasszuk ki, mint kiépítendő egységet, akkor szükségünk lesz egy nagyméretű tanítóhalmazra, amivel a felismerés pontosságát javítani tudjuk. Viszont ezzel a módszerrel nem tudunk minden szót lefedni. Itt is számolni kell a magyar nyelv adta eltérésekkel úgy, mint a toldalékolás, illetve a ragok képzése. Az angolhoz képest ez is nehezíti az általunk kigondolt program létrehozását, ugyanis a rendszernek így sokkal több szót kell ismernie a helyes működés érdekében, nagyobb tanítóhalmazra lesz szükségünk.

A „Build a Handwritten Text Recognition System using TensorFlow" [2] című webhelyen egy olyan neurális hálózat felépítése látható, ahol a kézzel írt angol szavak felismerése szófelismerésen alapul. Az előbb említett rendszer megvizsgálása után a következőket állapíthatjuk meg. Ennek a hálónak a tanítását 115 320 izolált és feliratozott szó felhasználásával végezték, ebből jól látszik, hogy mekkora adatkészletre is lenne szükségünk a magyar szavak felismeréséhez. A korábban tárgyalt okok miatt, azaz a magyar nyelv speciális toldalékolási rendszere csak még jobban nehezíti ezt a fajta megvalósítást, hiszen ennél akár jóval több adat is szükséges lehet a háló megfelelő betanítására. Ilyen adatbázis hiányában, magunknak kell egy kisebb méretű halmazt megalkotnunk, amivel megfelelően reprezentálhatjuk a rendszerünk működésének alapjait. Az általunk megalkotott tesztbázis már potenciális alapot nyújthat a modell felépítésére és tesztelésére. Hogyha nem is a magyar nyelv egészére, de egy kis részének a használatára alkalmas programmal egy általános képet kaphatunk a problémakörről. Így a továbbiakban ezzel a megvalósítási stratégiával haladunk tovább és ezek alapján építjük ki a tervezett rendszerünk alapjait.

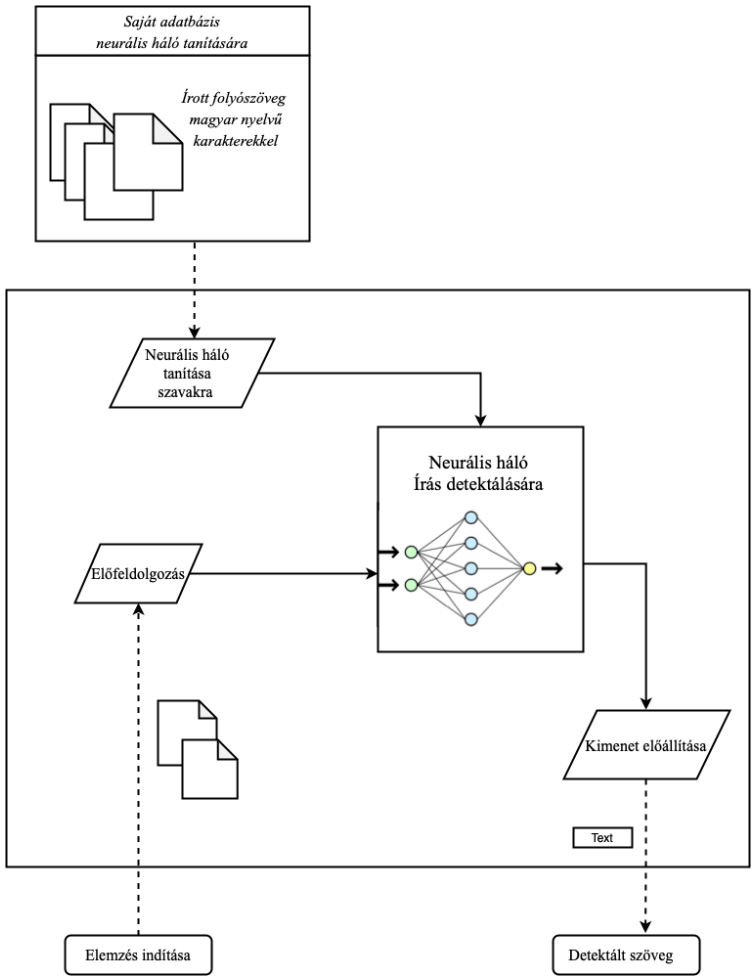
Az irodalomkutatásból kiderült az is, hogy a neurális hálók, illetve a mély neurális hálózatok egy jóval rugalmasabb és adaptívabb egység felépítésére alkalmasak, hiszen ezen rendszerek a fejlődőképességük javítása érdekében egyfajta gondolkodásmóddal, logikával rendelkeznek. A hálók a képfeldolgozás műveleteivel szorosan együtt tudnak működni, illetve mint körvonalazódott a konvolúciós neurális hálózatok a bemeneti képet pixel szinten elemezik és osztályozzák, ami a tervezett rendszer megvalósítása során fontos szerepet játszik. Minderre szükségünk is lesz a modell megalkotása során, így a mélytanulás lesz a programunk fő modulja.

A gyakorlati megvalósításhoz a legjobb eszközöknek a Python, a TensorFlow és a Keras bizonyult. Python nyelven sok olyan könyvtár elérhető mellyel nehézségek nélkül kezdhetünk bele a rendszer felépítésébe. A Google Colab használata számunkra mindenképp előnyös, hiszen számos olyan funkciója van melyet érdemes használnunk a későbbiekben. A TensorFlow és a Keras könyvtárak pedig a sokrétegű neurális háló modellek készítésén kívül széles körben alkalmazhatók más célokra is, ideértve a más jellegű gépi tanulási algoritmusokat is.

4.2. Rendszerterv bemutatása

Mindezek után már képesek vagyunk egy rendszer tényleges felállítására, a konkrétabb tervezés megkezdésére. Minden szükséges információ birtokában az egyes komponensek megtervezése is megkezdődhet. Az eddigi tervek alapján létrejövő rendszert a 4.1. ábra szemlélteti. Ezen az ábrán minden főbb modul látható, ezek a későbbiekben kifejtésre kerülnek.

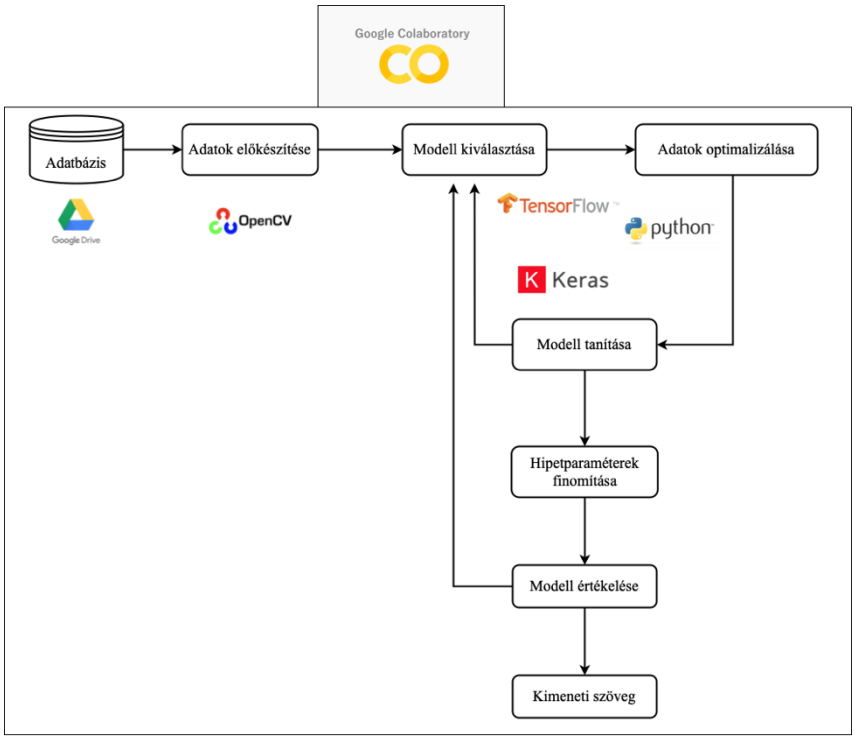
Ahhoz, hogy egy elemzés megkezdésétől eljussunk a detektált szövegig látszólag egyszerű a dolgunk, azonban egy igen összetett folyamat zajlik a háttérben. Maga a neurális háló tanítása az általunk összegyűjtött és címkézett adatokkal történik. Az írás detektálását végző háló előzetesen összegyűjtött kézírásos szavakat fogad, ezzel végezzük a tanítási fázist. Az elemzés indítását követően a bemeneti kép előfeldolgozásra kerül, ezt követően lép működésbe a neurális hálózat, ahol a szövegfelismerék zajlik. Mindezek pontos ismertetése a következő fejezetben történik.



4.1. ábra *Rendszerterv*

4.3. Megvalósítandó folyamatok

Annak érdekében, hogy a bemeneti képből sikeresen szöveget tudjunk alkotni a 4.2. ábrán látható folyamatokon kell végig haladnunk a modell megalkotása során. Ezek a modulok a végső rendszer legfontosabb szegmenseit képezik. Mindezek egymásután kifejtésre kerülnek a fejezetben, illetve bemutatásra kerülnek a lépések indokoltságai és a mögöttük lévő indíttatás. Az alábbi egységekről lesz szó: a megfelelő adathalmaz létrehozása, adatok előfeldolgozása a további műveletekhez, a modell kiválasztása, majd a modell implementálása, hiperparaméterek finomítása, végül pedig az eredmények kiértékelése. Ezek az eljárások többnyire fellelhetők más modell építési folyamatok során is, azonban vannak kiváltképp a neurális hálók kiépítésére jellemző lépések.



4.2. ábra A rendszer számára szükséges modulok működésének menete

4.3.1. Adatbázis létrehozása

A célokat definiálása és kijelölése után be kell szereznünk a szükséges információ halmazt. Az információ, mint tudás, adatokban testesül meg, vagyis szükségünk van a céljainkhoz illeszkedő adatbázisra. Az adathalmaz, amivel a neurális hálót tanítani fogjuk meghatározó része a programnak, ugyanis a rendszer működését a rendelkezésünkre álló gyűjtemény függvényében kell kialakítanunk. Az állomány, amin dolgozni fogunk lehet adott, használatra készen vagy lehet, hogy nekünk kell megalkotnunk.

Hosszabb kutatások után azt állapíthatjuk meg, hogy a dolgozat idejében nem áll rendelkezésre megfelelő adatbázis. Nincs jelenleg a magyar szavakra fellelhető gyűjtemény, ami címkézetten tárolná a folyóírással készült szavak képeit, így saját állomány létrehozása lesz a cél. Mivel angol nyelven fellelhető ilyen adathalmaz, illetve olyan hálózat megvalósítása is, ami ezeket az adatokat használja, ezáltal könnyedén létre tudunk hozni egy saját adatbázist.

Maga a megtervezés egyszerűen megoldható, hiszen az eddigi ismeretink alapján tudhatjuk, hogy szükség lesz három gyűjteményre. Ennél fogva beszélhetünk tanító, validációs és teszt halmazokról. A tanító készletet a modell felépítésére használjuk, a tanításra. A kiértékelő készlet felhasználásával hangolhatjuk a hálót, így csökkentve a túltanulás lehetőségét. A teszthalmazon pedig kiértékelhetjük a teljesítményt. A megvalósításhoz mindegyik állomány számára egy külön könyvtárat hozunk létre a Google Drive-on belül. A Drive elérése, illetve az adatok használata a Colab segítségével nehézség nélkül kivitelezhető. A három gyökér könyvtár belsejében ugyan azok a további könyvtárak lesznek elérhetők, hiszen minden tanítandó szóra készül egy validációs, illetve egy teszt halmaz is, értelem szerűen a tanító halmaznak kell a legnagyobb gyűjteménnyel rendelkeznie. A belső könyvtárak címkézve kerülnek eltárolásra, ezáltal egyértelműen következtethető a tárolt szavak képeinek valós tartalma is.

Az adatok összegyűjtése már okoz némi nehézséget, hiszen be kell látnunk, hogy egy nagy méretű tanító halmaz összerakása rengeteg időt igényel. Egy ilyen méretű kézírásos gyűjtemény összeállítása nem feltétlenül szükséges a megfelelő eredmény eléréséhez. Elégséges egy kisebb méretű halmazt megalkotnunk, amivel már megfelelően lehet a rendszerünk működésének alapjait létrehozni. Ha nem is a magyar nyelv egészére, de egy kis részének a használatára alkalmas programmal egy általános képet kaphatunk a problémakörről. Véges mennyiségű adatra alkalmazható program kialakítása lesz ezáltal a cél. A felismerendő szavak kiválasztásánál célkitűzés lehet, hogy alakban hasonló szavak kerüljenek az adatállományba, hiszen ezek felismerési arányából, illetve a tévesen felismert adatokól különböző vizsgálatokat tehetünk.

Mindemellett a különböző kézírásos szövegek összegyűjtésének menete is sok kérdést felvethet. Azaz össze tudunk-e szedni kielégítő mértékű kézírásos szöveget. Meg kell találni azt a mennyiséget, amivel a háló tanítása már megfelelően kivitelezhető és számunkra is elérhető. Ennek megoldásaként létrehozhatunk egy mesterséges környezetet, ahol az emberi folyóírást kézírásos betűtípusok használatával helyettesíthetjük. A kutatás során pedig ebben a környezeten értelmezett kézírás felismerése válik a kitűzött céllá. A fontok segítségével képesek vagyunk könnyedén több száz szürke árnyalatú képet rendelni minden szóhoz, amivel a neurális hálózat betanítása könnyedén elvégezhető.

4.3.2. Adatok előfeldolgozása

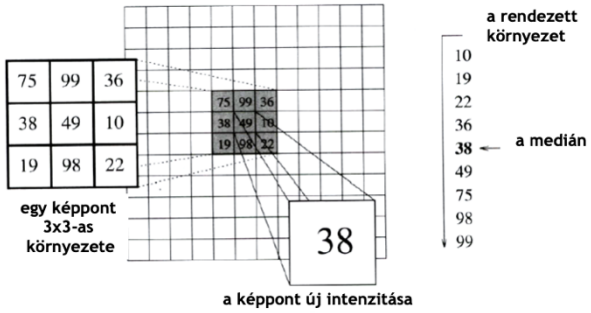
A rendszer számára szükséges információkat a gyűjteményünk elemei biztosítják, ezért kiemelkedően fontos, hogy megértsük a bennük lévő összefüggéseket és átlássuk a legfőbb jellemzőket. Ahhoz, hogy az általunk létrehozott állományban lévő adatokat használatba tudjuk venni, előbb el kell érni minden elem esetén a megfelelő állapotot. Bizonyos műveletek elvégzése válik szükségessé a halmaz komponensein. Előfeldolgozással a célunk, hogy a kiindulási állapotból egészen a kívánt állapotig módosítsuk az adatainkat úgy, hogy mindeközben a tartalmazott információ ne sérüljön, vagy módosuljon. Elmondható, hogy rendszerint sok időt követel az adatok megfelelő állapotba juttatása. Az előfeldolgozásra azért lehet szükség, mert legtöbb esetben a rendelkezésre álló adathalmaz zajos, hiányos, illetve a további műveletek elvégzésére nem megfelelő.

Szükség lehet az adatokon átalakításokat végezni, ezzel eleget téve bizonyos feltételeknek. Erre vonatkozó művelet lehet az adatok értékkészletének meghatározása, így bizonyosan egy adott intervallumon belüli értékeket képviselhetnek. Neurális hálózatoknál azért is kell átalakításokat végezni, hogy az értékkészlet illeszkedjen az aktivációs függvény lehetséges kimenetéhez. Amennyiben erről nem gondoskodunk a tanítás nem lesz eredményes. A bemeneti adatokat úgy kell módosítanunk, hogy a kialakítandó modell számára egy általánosított képet adjunk. Az első problémát jelentheti, ha a forrás adatok nem a megfelelő formátumban állnak rendelkezésre, ilyenkor el kell végezni a szükséges konverziót. Másik alapvető lépés a képek átméretezése, hiszen a neurális hálózat tanításánál és később a detektálás során így érhetjük el, hogy mindig hasonló területeket aktiváljon egy-egy objektum. Amennyiben nem azonos méretű képeket adunk inputként, akkor a háló rossz működést eredményezhet.

A képfeldolgozás egyik alapvető kihívása a képeken lévő zaj szűrése. A zajtalanítás az eredeti kép becslése a zaj elnyomásával. A képzajt különböző források okozhatják, amelyeket gyakran nem lehet elkerülni a gyakorlati helyzetekben, ezért a kép szűrése fontos tényezővé válhat. A gyakorlatan elterjedt módszerek a medián szűrő, átlagoló, illetve a Gauss szűrő.

A medián szűrő egy nemlineáris digitális szűrési technika, amelyet gyakran használnak a kép vagy a jel zajának eltávolítására. Az ilyen zajcsökkentés tipikus előfeldolgozási lépés a későbbi feldolgozás eredményeinek javítása érdekében. Bizonyos körülmények között megőrzi az éleket, miközben eltávolítja a zajt. A mediánszűrő fő gondolata az, hogy a képpontokat végigfuttatva, minden képpontot a szomszédos képpontok mediánjával kell helyettesíteni. A szomszédos képpontokat "ablaknak" hívják, amely a vizsgált pont 3x3-as környezetét jelenti.

A medián szűrő megfelelően alkalmazható a „só / bors” zaj eltüntetésére, emellett eltünteti a kis méretű kiugró értékeket. Nem változnak az intenzitásértékek a szűrő használatával, illetve az éleket jobban megtartja, mint az átlagoló vagy a Gauss szűrő. A medián szűrőnél lehet használni a 3x3-as keret helyett, például 5x5, 7x7-es vagy akár nagyobb keretet is. Működését a 4.3. ábra szemlélteti. [19]



4.3. ábra *Medián szűrő működési elve [20]*

Mindemelett szükséges lehet a képek alakítása szürkeárnyalatossá, a szürkeárnyalatos képek a feladat elvégzéséhez teljesen elegendők. Ezáltal kevesebb információt kell megadnunk minden egyes képpontról, így nincs szükség bonyolultabb és nehezebben feldolgozható műveletvégzők használatára sem. Emellett a kép binárissá alakítása is segítséget jelenthet a későbbieken.

Az egyes képeken lévő szavak vonalainak vastagsága szintén különbözhet, így szükséges lehet ezeket egységesíteni. Ennek eléréséhez egy megoldást adhat, ha a vonalakat a halványabb szélek mentén vastagítjuk. Ami azt jelenti, hogy addig növeljük a fekete pixelek számát, amíg el nem érünk a kívánt vastagságig. A kívánt vastagság a fekete pixelek arányát jelzi az összes pixelhez képest. A morfológiai transzformációkkal a kép alakja alapján végezhetünk műveleteket, általában ezeket bináris képeken hajtják végre. Két bemenetre van szüksége, az egyik az eredeti képünk, a másik az úgynevezett strukturáló elem vagy kernel, amely meghatározza a művelet jellegét. Az erózió és a dilatáció olyan műveletek, melyekkel a fekete és fehér pixelek arányát változtathatjuk, képesek a fehér zaj kiszűrésére, illetve hasznosak az objektumok méretének csökkentésére és növelésére is.

Az előfeldolgozás lépései számos algoritmus használatát követelik meg, melyek számottevően összetett feladatnak számítanak. Az általunk használatba vett Google Colab-on keresztül az egyik legnépszerűbb gépi látással foglalkozó könyvtár az OpenCV is könnyedén elérhető. Az OpenCV egy ingyenes, nyílt forráskódú függvénykönyvtár, amely a gépi látás megvalósításához szükséges programozási funkciókat tartalmaz. Több platformon és több programozási nyelvvel használható, leggyakrabban Python vagy C++ nyelven írt programokban alkalmazzák. A széles körben elérhető könyvtár, nagy mennyiségű optimalizált algoritmust tartalmaz, főként a gépi látás területein jelentkező problémák megoldására. [20]

4.3.3. A modell kiválasztása, implementálása

A modell definiálása esetünkben azt jelöli, hogy megadjuk milyen típusú neurális hálóval kezdjük meg a rendszer felépítését, illetve milyen architektúra alkalmazása lesz célszerű a számunka. Az eddig megismert építőelemekkel egy jól működő neurális hálózat létrehozása nehézségek nélkül megkezdhető.

A hálózat következetesen szekvenciális Keras modellre épül. Az első rétegnek szükséges előírni a bemeneti méretek számát, ami aztán a továbbiakban a belső rétegekben halad tovább. A hálózat belső rétegeinek deklarálása egy meghatározó lépés, hiszen sok egyéb paramétert, illetve jellemzőt is itt adunk meg. A számunkra releváns konvolúciós réteg esetén több jellemző megállapítása szükséges, ilyen a szűrők száma mellett az aktivációs függvény, illetve a kernel mérete is. A teljesen összekötött rétegek esetén a réteg neuronjainak száma és az aktivációs függvény definiálása a feladat. Az aktivációs függvények a köztes rétegek esetén ReLu függvényt alkalmazzák, míg az utolsó réteg esetén Softmax, amely a válasz kiértékelésében segít. A pooling rétegek esetén a lépés nagyságát és a pooling méretét kell megadni, ezzel a kinyert jellemzők számának csökkentését végezhetjük.

Amennyiben kiválasztottuk, hogy milyen típusú modellt, és a neurális hálózaton belül milyen hálózati architektúrát építünk, illetve az adatok a megfelelő formában rendelkezésre állnak az előfeldolgozás elvégeztével, úgy a következő lépés már az implementáció. Ennek során a Google Colab nyújt számos segítséget.

4.3.4. Hiperparaméterek finomítása

A neurális hálómodellek tekintetében, bármelyik változatot is használjuk, indokolt definiálni a hiperparaméterek induló értékét, amit az ezt követő iterációs műveletek alkalmával módosítani tudunk. A folyamatok között az egyik legidőigényesebb feladatnak a hiperparaméterek optimalizációját tekinthetjük. Hiszen egy terjedelmes hiperparaméter tér vizsgálatáról van szó, elméletileg minden egyes alternatívát ki kellene próbálni, hogy megbizonyosodjunk arról, hogy elértük az optimális modellt. Az optimumról ugyan le kell mondani, de a lehető legjobb megoldás megtalálására így is törekedni kell. Azonban léteznek eszközök ezen műveletek gyorsítására.

4.3.5. Eredmények kiértékelése

Az egyik utolsó és kihagyhatatlan lépése a modell létrehozásának, az eredmények kiértékelése. Ezt hiba vagy pontossági mutató felhasználásával érhetjük el, ezzel mérlegelni tudjuk a háló és annak eredményeinek a jóságát. Ezt elvégezhetjük az elvárt és a tényleges kimeneti értékeket összevetjük, vagy pedig sokszor egyéb hasonló felépítésű rendszer által produkált végeredményhez viszonyítjuk a saját rendszerünket.

5. MEGVALÓSÍTÁS

A fejezetben a projekt során elvégzett feladatok gyakorlatias leírása kerül. Az ismertetés menete egyrészt az előzőekben kifejtett követelmények alapján zajlik, másrészt pedig a megvalósítás lépéseinek mintájára. Bemutatásra kerülnek a fejlesztés során felmerült problémák és mindemellett az egyes ajánlások, megoldások, illetve a modellek megvalósítása is. A célkitűzésem a feladatkiírásban vállaltak abszolválása volt.

A rendszer fejlesztésének menete kettő főbb szegmensre bontható. Az első nagy alkotórész a mély neurális hálózatunk betanításához elengedhetetlen tanítóminták létrehozása, rögzítése. Ezen gyűjtemények előfeldolgozása, illetve a tanításhoz szükséges formátum elérése is ide tartozik. A második mérföldkő a modell felépítése, a megfelelő paraméterek beállítása. Az ezt követő szakaszban pedig már a meglévő adatbázis felhasználásával történik a neurális hálózat betanítása, mindezt a TensorFlow, illetve a Keras könyvtárak segítségével.

A megvalósítás során a paraméteres eljárásoknál fontos szerepet kapott a változók optimalizálása. Ez egy energiaigényes művelet, ugyanis számos iteráció elvégzése szükséges a megfelelő eredmény eléréséhez. A dolgozatban szereplő valamennyi komponenshez végeztem optimalizációs lépéseket. Összességében nehéz megmondani, hogy az egyes modellek paramétereinek hány változata került tesztelésre és azok milyen messze vannak az optimális eredményektől. Ennek magyarázata, hogy sok manuális futtatás került elvégzésre, illetve a paraméterek száma modellenként merőben eltérő. Igyekeztem a komponensek esetén a legjobb tudásom szerinti beállításokat elvégezni, megannyi kísérletet végrehajtani.

5.1. Tanítóminták létrehozása

A folyószöveg elemzésének megkezdése előtt megannyi lépést kell megtennünk, melyek között az első a neurális háló tanításához szükséges címkézett adatok összegyűjtése. A mély neurális hálózat tanítása nagy mennyiségű tanítómintát követel, melyeket először elő kell állítani.

Az előző fejezetekben kifejtettek alapján látjuk, hogy adatbázis hiányában magunknak kell egy kisebb méretű halmazt megalkotnunk, amivel megfelelően reprezentálhatjuk a tervezett működést. A jelenlegi alkalmazásban elégséges a magyar nyelv kis részének a használatára alkalmas programmal teszteléseket végezni. A kép állományok, a tanító, a validációs és a teszt halmaz formájában álltak elő. A megvalósításhoz mindegyik gyűjtemény számára egy külön könyvtárat hoztunk létre a Google Drive-on belül, hiszen a Drive elérése, illetve az adatok használata a Colab segítségével könnyedén kivitelezhető.

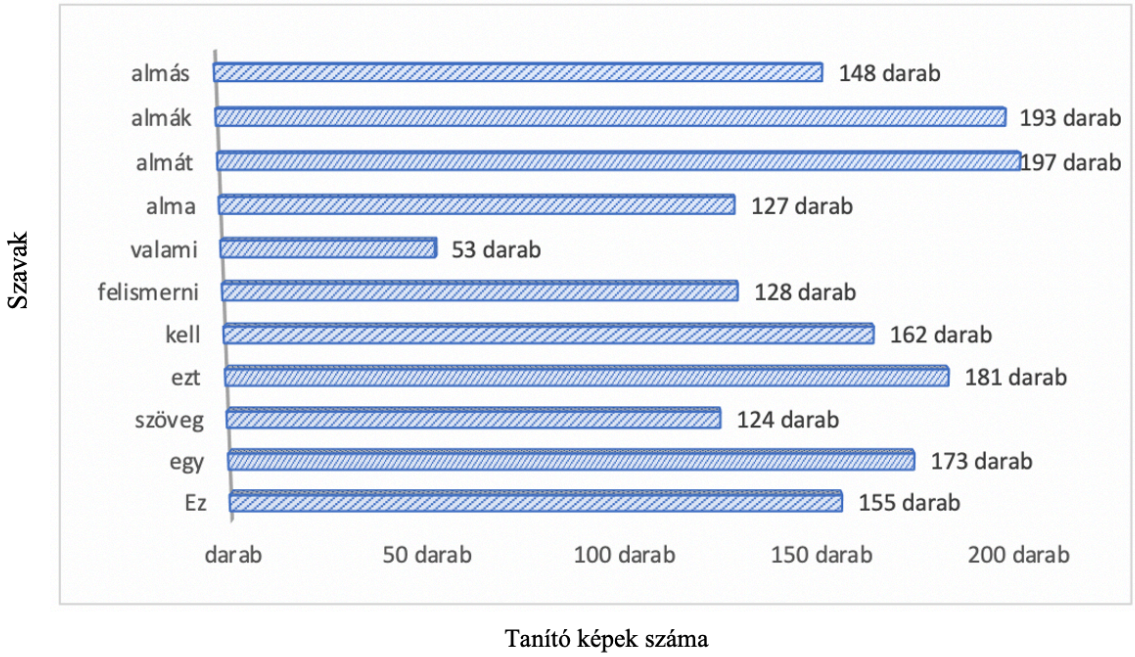
A három könyvtár, vagyis a Words, Tests, és a Validations mappák ugyanazokat a további elemet tartalmazzák, hiszen minden tanítandó halmazra készül egy validációs, illetve egy teszt halmaz is. A felismerendő szavak kiválasztásánál számos dolog közrejátszott. Alapvetően 10-15 szó kijelölése volt cél ugyanis, ha ezek mindegyikét pár száz kép felhasználásával tanítjuk be, máris egy nagyobb adathalmaz készítése válik elkerülhetetlenné. Maradva a csekélyebb szókészlet mellett, a kiválasztásra került szavak közé kerültek rövidebb, illetve hosszabb szavak, ékezetes szavak, továbbá ami a legfontosabb, hasonló alakú szavak is.

A tervezés során kialakított struktúra alapján egy mesterséges környezet létrehozása volt a feladat, ahol az emberi folyóírás kézírásos betűtípusok használatával került helyettesítésre. A fontok segítségével könnyedén több száz tanító minta rögzítése kezdődhetett meg. Azonban itt már jelentkezett is az első felvetés, vagyis miként lehetséges a nagy mennyiségű tanító állomány hatékony megalkotása. A már jól ismert technológiákkal, mégpedig a Python nyelv a Colab kettősének segítségével ezt nehézségek nélkül kivitelezhetjük.

A PIL a Python Imaging Library rövidítése, ez az eredeti könyvtár, amely lehetővé tette a Python számára a képek kezelését. Ezt volt a segítségünkre a tanító állomány rögzítésé során, hiszen ezt felhasználva képesek voltunk egyszerű saját képek készítésére. Az algoritmus működéséhez szükség volt .ttf kiterjesztésű betűtípusok gyűjteményére, amit nehézségek nélkül sikerült is összegyűjteni. Körülbelül 200 darab kézírásos font állt a rendelkezésünkre az adathalmaz megalkotásához, amik között voltak merőben eltérő és igencsak hasonló betűtípusok is. Fontos volt, hogy magyar ékezetes betűket tartalmazzák a kiválasztott font elemek. A .ttf állományok egy Drive mappán belül kerültek eltárolásra, ahonnan gond nélkül az algoritmus rendelkezésére álltak. A tanítandó szavakat egy listában tároltuk el, majd ezen lista elemein végig haladva minden szóra elkészítésre kerültek a tanítóminták a korábbiakban kifejtett struktúra alapján. A képek általános felépítését az 5.1. ábra illusztrálja.

5.1. ábra *A létrehozott képek bemutatása*

A tanításra szánt elemek darabszámát illetően, a természetes adatbázisok mintája került előtérbe. Vagyis nem minden szó került előállításra minden egyes meglévő betűtípus felhasználásával, így előállt egy vegyes szerkezetű halmaz. Voltak előzetesen összegyűjtött szavak képei is, amik szintén bent maradtak az állományban. A tanító képek gyűjteményének végső méretét az 5.2. ábra ismerteti.



5.2. ábra *A tanításra szánt adathalmaz méretének grafikonja*

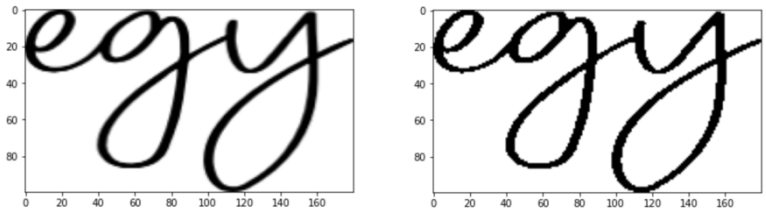
A modellünk megalkotása során az általánosítóképesség javításához szükség van validációs, illetve teszt adatok használatára is. A kiértékelő készlet felhasználásával hangolhatjuk a hálót, így csökkentve a túltanulás lehetőségét. A teszthalmazon pedig a végső teljesítményt értékeljük. A tanító halmaznál egy jóval kisebb adatállomány felhasználásával már működtethető az előbb említett kettő gyűjtemény. Végül szavanként 15-15 kép került felvételre mind a validációs, mind a tanító halmaz esetén. Ezek olyan betűtípussal készültek, melyek nem lettek felhasználva a tanítóminták rögzítése során.

Miután rögzítettük a megfelelő adatokat tanító, kiértékelő illetve tesztelő elemekként, megkezdődhetett a részletek átformálásához szükséges algoritmusok implementálása, vagyis az előfeldolgozási műveletek kidolgozása. Az általunk létrehozott állományban lévő adatok használatbavételéhez elsőként a megfelelő állapot elérése volt az elvárás. Előfeldolgozás vált szükségessé a halmaz komponensein úgy, hogy ezalatt a tartalmazott információ ne sérüljön, ne módosuljon. Az előző fejezetekben tárgyalt adatfeldolgozó technikák, műveletek egy megfelelő kezdő állapotot biztosítottak számunkra. Ezek mentén haladva készült el a program számára fontos szegmens, az előfeldolgozó egység kiépítése. Az előfeldolgozás algoritmusaihoz az OpenCV-nek a függvényei kerültek felhasználásra.

Annak érdekében, hogy egy mély neurális hálózat betanítása gond nélkül végbemehessen, szükség van egy általános bemeneti struktúra megalapozására. Esetünkben az első megkötés az adatok megfelelő formátuma, ilyenkor el kell végezni a szükséges konverziót. Számunkra ez a kép formátumú adatok használatát jelenti, azonban a tanítóminta nagyrészt saját fájlok előállítása során készült, ahol megfelelő formátumban kerültek elmentésre az elemek. Másik alapvető lépés a képek átméretezése, hiszen mint kiderült a neurális hálózat tanításánál és később a detektálás során így érhetjük el, hogy mindig hasonló területeket aktiváljon egy-egy objektum. Számos kísérlet után optimális választásnak egy 180x100 pixel nagyságú téglalap bizonyult. Mind a hosszabb, mind a rövidebb szavak esetén megfelelő a méretezés, nem torzul a szöveg a képeken. A karakterek szélei igaz kevésbé lesznek élesek, de ezt a szűrések során javítani tudjuk. Az 5.3. ábrán az átméretezés eredménye látható az eredeti kép mellett.

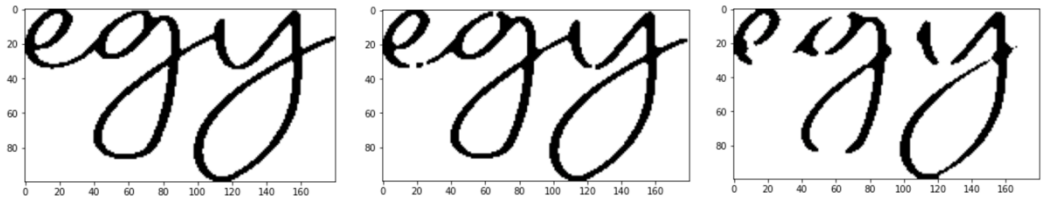
5.3. ábra *Eredeti kép (bal), illetve az átméretezett kép (jobb) szemléltetése*

Az irodalomkutatás során kiderült, hogy a bináris képek a feladat elvégzéséhez teljesen elegendők, kevesebb információt kell megadnunk minden egyes képpontról, illetve a további folyamatok elvégzése egyszerűbbé válik. A kép binarizálása a szürkeárnyalatos kép fekete-fehér képpé alakítását jelenti, amit a küszöbértéknek nevezett érték segítségével lehet elérni. A küszöbérték jelenlétekor minden pixel 0-ra vagy 255-re lesz konvertálva attól függően, hogy a vizsgált pixel értéke nagyobb vagy kisebb, mint a küszöbérték. Normál esetben, hogyha a pixel értéke nagyobb, mint a küszöbérték, akkor a rendszer 255-re, ellenkező esetben 0-ra konvertálja az elemet. Esetünkben a kiválasztott küszöbérték az intervallum közepe, vagyis 127. Mivel a képeken nincsenek kiugró értékek, lokális különbségek, ez a megközelítés megfelelő számunkra. Az eddig előállt képeken lévő karakterek szélei a binarizálás következtében megtörtek, az értékek a szélek mentén ingadoznak, ezt az 5.4. ábra szemlélteti. A bináris képek előállítása után az előfeldolgozási lépések soron lévő algoritmusainak az ismertetése következhet.



5.4. ábra *Átméretezett kép (bal), illetve a bináris kép (jobb) szemléltetése*

A medián szűrő képes eltüntetni a kis méretű kiugró értékeket, nem változnak az intenzitásértékek a szűrő használatával, illetve az éleket is képes megtartani. A szűrő esetén több méretű keret használata is lehetséges. Esetünkben az előállított képeken alapértelmezetten nem jelenik meg zaj, azonban lehetnek alkalmak amik során az eddigi előfeldolgozási lépések hatására a képeken szükség lehet a szűrő használatára. A tesztelések alkalmával gyorsan világossá vált, illetve amit sejteni is lehetett, hogy a nagy méretű keret számunkra nem lehet megfelelő. A vonalak vastagsága a képeken csupán néhány pixel, ezért a nagyobb keret hatására a vonalak könnyen elvékonyodhatnak, illetve akár teljes egészében eltűnhetnek. Az 5.5. ábrán a különböző keretek használatának kimenetei láthatók. Végül egy 3x3-as keret került használatba.

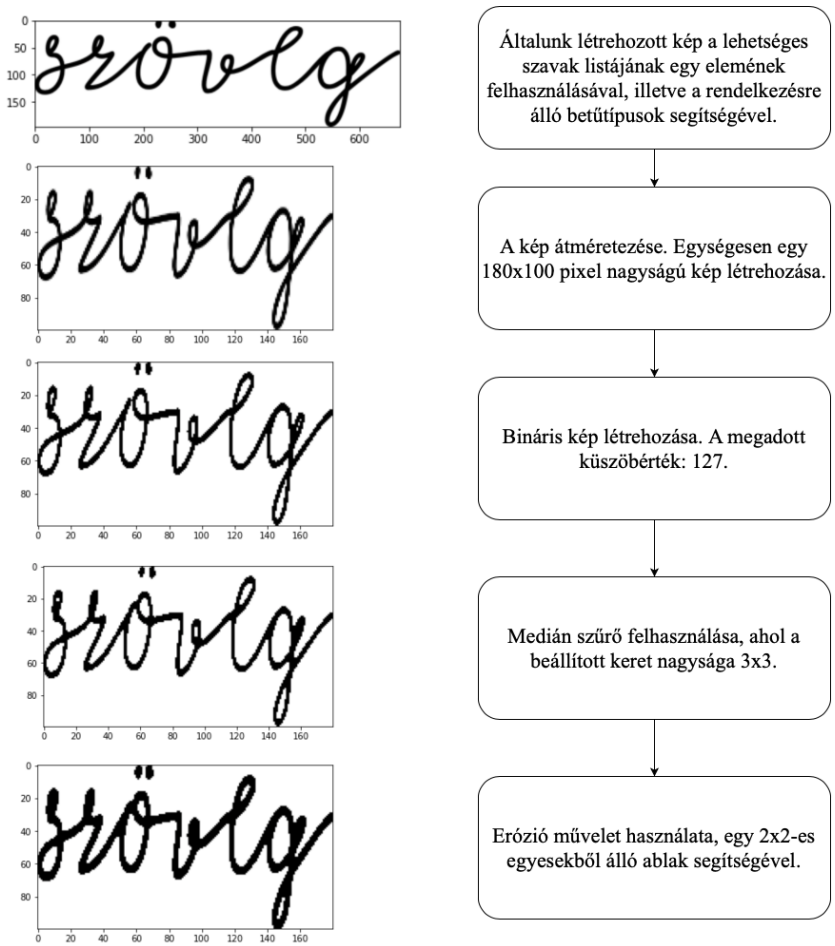


5.5. ábra *Medián szűrő kimenetei más-más keret esetén:*
3x3 (bal), 5x5 (közép), 7x7 (jobb)

Morfológiai transzformációkat általában bináris képeken végezhetünk. Számunkra az erózió művelete bizonyult hasznosnak, mellyel a fekete és fehér pixelek arányát változtathatjuk, esetünkben az objektumok méretének növelésére alkalmazhatjuk. Az erózió során ez eredeti kép egy pixele csak akkor tekinthető egyesnek, ha a kernel alatti összes pixel értéke 1, ellenkező esetben nullás értéket kap. Használatával a fehér terült csökken, a fekete nő, vagyis a kernel méretétől függően a határ közelében lévő pixelek értékei változhatnak. Több kernel méretet kipróbálva az látszik, hogy egy lehetséges optimális kimenethez juthatunk egy egyesekkel teli 2x2 méretű kernelt használva. Az OpenCV algoritmusával felhasználva olyan kimenetet kapunk, ahol a vékonyabb vonalak is kiegészülnek, növekednek, összességében pedig a képen lévő karakterek hangsúlyosabbak lesznek.

Miután sikeresen megtörtént mind a három gyűjtemény rögzítése, illetve kirajzolódtak az előfeldolgozás megfelelő algoritmusainak a lépései, úgy képesek vagyunk a tanító, a validációs, illetve a teszt halmaz létrehozására. A korábban leírt struktúra alapján létrejött adatállományok elemein végighaladva megalkotásra kerültek a valós tanító adatok. Ezek azok az adatok, melyekkel a felépített neurális hálózat dolgozik.

A megvalósítás lépései átláthatónak mondhatók, egyszerűen a halmazok adatain kellett az előfeldolgozás lépéseit lefuttatni, majd az eredményeket egy közös halmazban eltárolni. Megvizsgálva a tanító halmaz létrehozását, a következő műveletek zajlottak le. Létrehozásra került egy gyűjtemény, amibe a később feldolgozott képek kerültek. A Drive mappáin végighaladva megkaptuk ez egyes szavakhoz tartozó tanításra szánt képeket, melyeket egyesével feldolgoztunk. Ezután kerültek elmentésre a közös gyűjteményben, ahol a képek mellett egy, a valós tartalomra utaló index is mentésre került. Miután elkészült a halmazunk, a bennne lévő adatok összekeverésre kerültek, ennek következtében nem egymás után, nem sorban lettek elhelyezve az azonos szavakat tartalmazó képek. Az imént ismertetett műveletek mind a három állomány esetén lefuttatásra kerültek, ezáltal elkészült minden adat a modell felépítéséhez. A véglegesített előfeldolgozási algoritmus egyes lépéseit az 5.6. ábra mutatja be.



5.6. ábra *Előfeldolgozási algoritmus egyes lépései*

5.2. Modell megalkotása

A betanítási minták sikeres előállítása után az implementáció soron következő lépése a mély neurális hálózat létrehozása és tanítása. Az irodalomkutatásból kiderült, hogy ennek a megvalósítására a legnagyobb segítséget a TensorFlow és a Keras nyílt forráskódú könyvtárak biztosítják. Mint az eddigi műveletek során, most is a Colab használata az értelemszerű választás számunkra.

Egy kezdeti lépés az értékkészlet meghatározása az aktivációs függvény lehetséges kimenetei számára. Ez szükséges ahhoz, hogy a neurális hálók képesek legyenek bizonyosan egy adott intervallumon belüli értékeket képviselni. Amennyiben erről nem gondoskodunk, úgy a tanítás nem zárulhat eredményesen. Esetünkben ez a tanítóhalmaz létrehozásával, illetve a szavak kiválasztásával meg is történt. Hiszen ezek lesznek azok a szavak, amiket a rendszerünk képes lesz értelmezni.

Vannak paraméterek melyeket a modellépítés indulásakor már létre tudunk hozni a rendelkezésünkre álló adatok felhasználásával. Az *x_train*, illetve az *y_train* olyan jellemzők, melyek a háló betanítása során alkalmazásra kerülnek. Az x azon adatok összesége, melyeket a modell bemeneteként használ. Az y pedig a várt eredmények, címkék reprezentálása, hiszen a felügyelt tanulásnál ismerni kell az elvárt eredményt és ehhez viszonyítva értékelni kell a modelleket. Ezáltal a létrejövő paraméterek a következők*: x_train alakja: (1641, 180, 100, 1*) , *y_train alakja: (1641, 11)*. Mindkét esetben az első változó a tanító elemek számát képviseli. Az *x* további attribútumai a bemeneti adatok jellemzői. A bemeneti képek méretének meghatározása elengedhetetlen, ami jelen esetben 180x100 pixel nagyságot jelent, hiszen az előfeldolgozás során a saját képeinknek az átméretezése is ezekkel a méretekkel történt. Az utolsó paraméter pedig a pixelenkénti információt hordozza magával, ugyanis a fekete-fehér képekről elegendő egy információt megadni a pixelenként. Az *y* második paramétere pedig a kimeneti halmaz méretét adja meg.

Következő lépés a mély neurális hálózatunk meghatározása volt. Esetünkben a modell definiálása azt jelöli, hogy megadjuk milyen típusú neurális hálóval kezdjük meg a rendszer felépítését, illetve milyen struktúra mentén építjük fel a hálót. A kiindulási pont egy konvolúciós neurális hálózat architektúrája volt, a szerkezet hangolása sok időt vett igénybe, számos lehetőség kipróbálásra került. A konvolúciós rétegek paramétereinek beállítása is több szempont alapján zajlott.

A konvolúciós réteg esetén a megadott szűrők számának kezdetben alacsonynak kell lennie, hogy felismerje az alacsony szintű jellemzőket, amelyek segítenek megkülönböztetni az osztályokat. A konvolúció szűrője azt méri, hogy egy bemeneti folt mennyire hasonlít egy jellemzőhöz. A szűrők száma általában növekszik vagy változatlan marad, miközben a konvolúciós neurális hálózati architektúránk rétegein haladunk és egyre több réteg kapcsolódik egymáshoz.

Kernel vagyis a szűrő méretének meghatározása számos lehetőséget kínál. A kisebb szűrők a lehető legtöbb helyi információt gyűjtik össze, a nagyobb szűrők globálisabb, magasabb szintű és reprezentatív információkat képviselnek. Nagy mennyiségű pixel begyűjtéséhez nagyobb szűrő használata célszerű (9x9, 11x11). Ellenben, ha az objektumokat néhány kicsi és helyi jellemző különbözteti meg, akkor kis szűrőket (3x3, 5x5) ajánlatos használni. Érdemes elsőként kisebb szűrők használatával a lehető legtöbb helyi információt összegyűjteni, majd fokozatosan növelni a szűrő méretét, hogy globálisabb, magasabb szintű és reprezentatívabb információk is rendelkezésre álljanak.

A rétegek aktiváló függvénye egy egyszerű matematikai függvény, amely az adott bemenetet a kívánt kimenetre alakítja. A nevükből adódóan aktiválják a neuront abban az esetben, ha a kimenet eléri a funkció beállított küszöbértékét. Alapvetően ez felelős a neuron BE/KI kapcsolásáért. A ReLu függvény jelenleg a leggyakrabban használt aktiváló, a legtöbb konvolúciós neurális hálózatban vagy mélytanulási rendszerben fellelhető. [21]

Mindezen ismereteket felhasználva a kialakított hálózat következetesen egy szekvenciális Keras modellre épült. Az első rétegnek szükséges előírni a bemeneti méreteket, amit a későbbiekben a belső rétegek megkapnak az előző rétegektől. Ebből kifolyólag attribútumként átadásra került a lehetséges input adatok paraméterei. Az első rejtett réteg egy 32 szűrőből álló Conv2D konvolúciós réteg. Felépítését tekintve egy 3x3 nagyságú kernel felhasználásával jött létre. Ezt követően egy másik konvolúciós réteg áll 64 szűrővel, egy 5x5 nagyságú kernellel, aktivációs függvénye pedig a ReLu függvényt alkalmazza.

Ezeket egy pooling réteg követ, amely a kinyert jellemzők számának csökkentését végzi, mindazonáltal a réteg redukálja a megtanulandó paraméterek számát és a hálózatban végzett számítások mennyiségét. A pooling réteg a szűrő által lefedett régiókat összefoglalja, az ott lévő konvolúciós réteg generálta jellemzőtérkép adatai alapján. Majd ezeket a konvolúciós réteg által generált precízen elhelyezett jellemzőket helyettesíti egy összesített jellemzőtérképpel, ami robusztusabbá teszi a modellt. A MaxPooling egy olyan összevonási művelet, amely a szűrő által lefedett jellemzőtérkép régiójából kiválasztja a maximális elemet. Vagyis a MaxPooling réteg utáni kimenet az előző jellemzőtérkép legszembetűnőbb attribútumait tartalmazó térképként fogható fel. A MaxPooling esetünkben 2x2 szűrőmérettel van konfigurálva. [22]

Ezután alkalmazásra került egy szabályosító réteg, a Dropout. A Dropout egy olyan technika, amely használatával a tanítás során kiesik némely véletlenszerűen kiválasztott neuron. Ez azt jelenti, hogy a kiválasztott neuronok aktiválásához való jogosultság ideiglenesen megszűnik. A Dropout egy olyan technika, amellyel megakadályozhatjuk a modell túlillesztését. A hálózat kevésbé lesz ezáltal érzékeny a neuronok fajsúlyára, ami egy olyan modellt eredményez, amely jobban képes általánosítani.

Az korábbi lépések befejezése után rendelkezünk egy jól használható jellemzőtérképpel. Magának a konvolúciós rétegnek a kimenetét nem tudjuk közvetlenül átadni a teljesen összekapcsolt Dense rétegnek, mivel ez a kimenet többdimenziós alakú. A teljesen összekapcsolt réteghez egydimenziós formában, azaz egydimenziós tömbben kell megadni az adatokat. Minden neurális hálózatban a Dense réteg olyan réteg, amely mélyen kapcsolódik az előző rétegéhez, vagyis a réteg neuronjai az előző réteg minden neuronjához kapcsolódnak.

A Flatten segítségével a kétdimenziós mátrixunkat vektorrá alakíthatjuk, ami lehetővé teszi, hogy a kimenetünket teljesen összekapcsolt rétegek dolgozzák fel a későbbiekben. Ahogy ennek a lépésnek a neve is sugallja, az egyesített jellemzőtérképünket egy oszlopba rendezzük. A kiegyenlítés az adatok egydimenziós tömbbé konvertálása a következő rétegbe történő bevitelhez, vagyis kisimítjuk a konvolúciós rétegek kimenetét, hogy egyetlen hosszú jellemzővektort hozzunk létre. Ez a kimenet kapcsolódik a végső osztályozási modellhez, amelyet teljesen összekapcsolt rétegnek neveznek. Más szóval, az összes pixeladatot egy sorba helyezzük és kapcsolatot hozunk létre a végső réteggel.

Ezután hozzáadtunk egy teljesen összekapcsolt réteget, amely 128 neuronnal és egy ReLu aktiváló függvénnyel rendelkezik. A modellben a Dense réteg neuronja a megelőző réteg minden neuronjától kap kimenetet, a réteg a kép osztályozására szolgál a konvolúciós rétegek kimenete alapján.

Ezt követően alkalmazásra került egy újabb szabályosító réteg a túlillesztés csökkentése érdekében, ezúttal véletlenszerűen kizárjuk a neuronok 50%-át.

A neurális hálózatot egy 11 neuronból álló kimeneti réteggel fejeztük be, vagyis a felismerhető szavak számával megegyező neuron számmal. Illetve egy Softmax aktiváló függvénnyel, amely a válasz kiértékelésében segít.

A megalkotott modell végső felépítését az 5.7. ábra szemlélteti, itt történik az előzőekben leírt lépések megjelenítésé a Keras segítségével. Leolvasható a rétegek típusa mellett a beállított paraméterek is. Elmondható, hogy a végsőként kiválasztott, implementált, majd bemutatott háló a benne lévő rétegekkel és attribútumokkal együtt, egy hosszabb folyamat eredménye. Számos verzió készült a lehetséges modellekből, melyek a tesztelés folyamán olykor jobbnak, illetve rosszabbnak bizonyultak. Azonban a tesztelés szakaszát megelőzi a betanítási modul. A tanítás során is számos beállítási lépés történt meg, melyeket mélyebb kutatói munka előzött meg.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 178, 98, 32)       320

 conv2d_1 (Conv2D)           (None, 174, 94, 64)       51264

 max_pooling2d (MaxPooling2D  (None, 87, 47, 64)       0
 )

 dropout (Dropout)           (None, 87, 47, 64)        0

 flatten (Flatten)           (None, 261696)            0

 dense (Dense)               (None, 128)               33497216

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 11)                1419
```

5.7. ábra *A kiválasztott modell felépítése*

Ahhoz, hogy a modellünk betanítása kezdetét vehesse szükségünk van néhány paraméter meghatározására, hiszen konfigurálni kell a tanító metódust. Az első attribútum az optimalizáló függvény, utána a veszteség függvényt kell megadni, az utolsó paraméterrel pedig a mérés módját kell meghatározni.

A modell betanítása során a paramétereket és a súlyokat hangoljuk, hogy minimalizáljuk a veszteséget, illetve igyekszünk optimálisan beállítani a modell pontosságát. A hiperparaméterek kalibrálásához az optimalizáló vált megkerülhetetlenné, ez a modell paramétereit a veszteségfüggvénnyel köti össze úgy, hogy a veszteségfüggvény kimenetére reagálva frissíti a modellt. A veszteségfüggvény csak azt jelzi az optimalizálónak, hogy az jó vagy rossz irányba halad.

A gradiens süllyedés egy olyan optimalizálási algoritmus, amely a célfüggvény gradiensét használja a keresési térben való navigáláshoz. Az Adam optimalizálás egy sztochasztikus gradiens süllyedési módszer, amely az első és másodrendű momentumok adaptív becslésén alapul. Ezáltal rövidebb idő alatt, hatékonyan képesek vagyunk betanítani a neurális hálózatot. Az Adam optimalizáló az egyik legtöbbet használt függvény a neurális hálók betanítása esetén, így a függvény használata számunkra is megfelelő.

A veszteségfüggvényeket alapvetően két különböző kategóriába sorolhatjuk, az osztályozási és a regressziós típusba. Az osztályozási veszteség esetén a cél a különböző kategorikus értékekből származó kimenet megjóslása. A Categorical cross-entropy egy veszteségfüggvény, amelyet több osztályos osztályozási feladatokban használhatunk. Használata során egy adott elem csak egy kategóriába tartozhat, ezen tulajdonságok ismeretében a használata megfelelő számunkra.

Az optimálisnak vélt mérési mód az osztályozó műveletek esetén a pontosság mérése. A pontosság egy mérőszám, amely a modell teljesítményét írja le, értéke a helyes előrejelzések száma és az összes jóslat közötti arány. Akkor hasznos, ha minden osztály egyforma fontosságú.

Ezen attribútumok kijelölését követően a következő lépés a tanítás megkezdése volt, ezt a műveletet a fit() metódus meghívásával érhetjük el. A korábban létrehozott x_train, illetve az y_train paramétereket most vesszük használatba. Szükséges meghatároznunk itt a hálózat számára az epochok számát és a batch méretét.

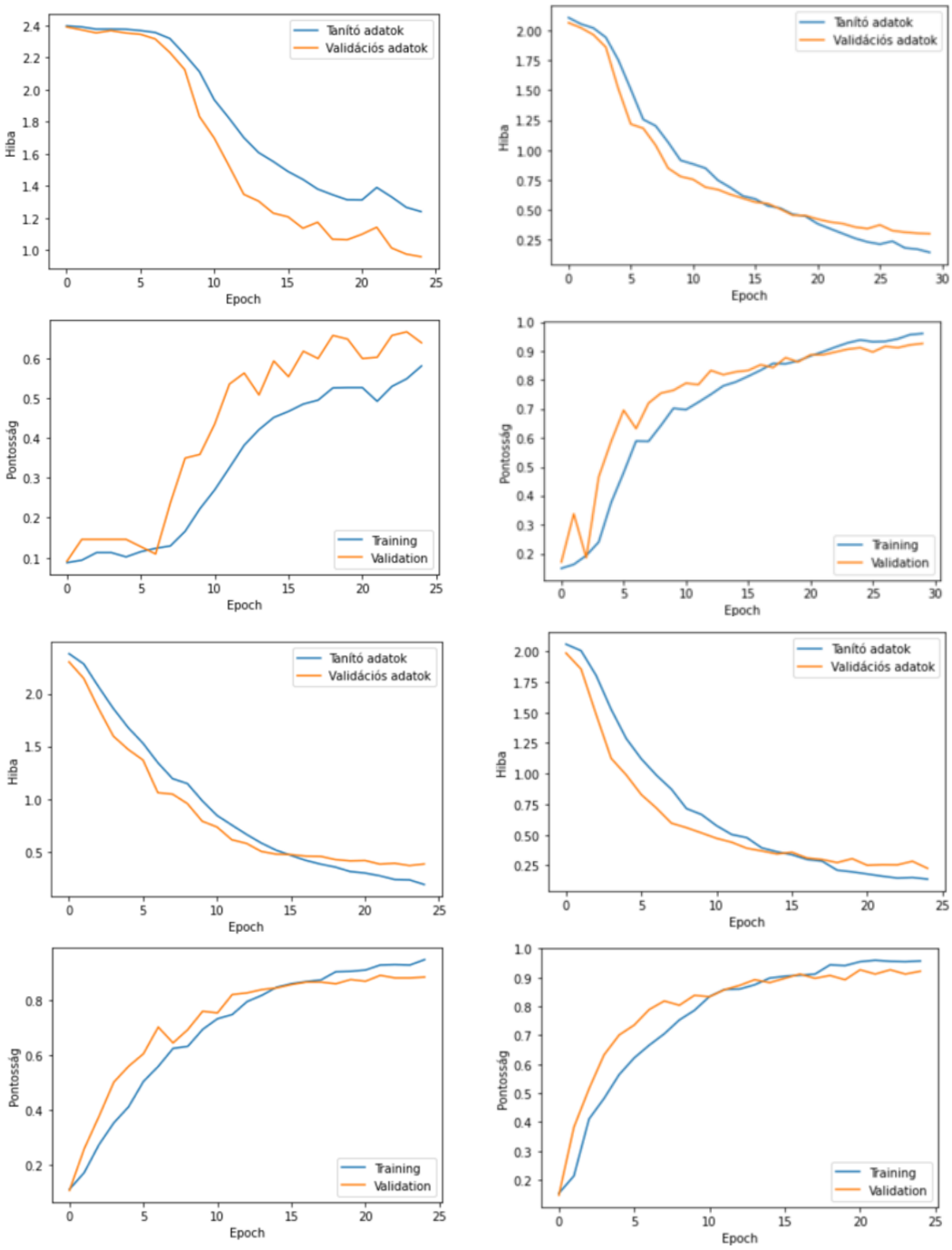
Az epochok száma egy hiperparaméter, amely meghatározza, hogy a tanulási algoritmus hányszor fog a teljes betanítási adathalmazon végigmenni, más szóval a betanítási adatkészleten való teljes áthaladások száma. Egy epoch lefutása esetén a betanítási adatkészlet minden mintája egyszer képes frissíteni a belső paramétereit. Az epochok száma hagyományosan nagy, gyakran több száz, ami lehetővé teszi a tanulási algoritmus futtatását mindaddig, amíg a modell hibáját kellően minimalizálni tudjuk. A teljes adatkészletet többször kell átadnunk ugyanannak a neurális hálózatnak, hiszen a súlyok frissítése egyetlen lépéssel nem megoldható. Minden lefutás esetén a neurális hálózat súlyának többszöri módosítása történhet meg.

A neurális hálónak nem lehet egyszerre a teljes adatkészletet átadni, ennél fogva szükségszerű az adatkészlet felosztása. Erre szolgál a batch, egy epoch egy vagy több batch-ből áll, a batchek száma megegyezik egy epoch iterációinak számával. A batch méretének kisebbnek vagy egyenlőnek kell lennie a betanítási adatkészletben lévő minták számával. [23]

Nincsenek szabályok ezen hiperparaméterek konfigurálására, így a különböző értékeket magunknak kell kipróbálnia, tesztelni és megtalálnia a lehető legjobbnak vélt megoldást. A mi esetünkben ez az epochok számát illetően 25 lett meghatározva, a batch mérete pedig 128.

A mélytanulás esetén a korai leállítás az egyik legszélesebb körben használt technika a túltanulási probléma leküzdésére. A Keras kínálta függvény, az EarlyStopping figyeli a modell teljesítményét minden epochban és leállítja a képzést bizonyos adatok függvényében. Az epoch múlásával kezdetben a tanító és a validációs halmaz hibája is csökken, azonban egy idő után a validációs hiba csökkenése leáll és újra növekedésnek indul. Ez azt jelzi, hogy a modell elkezd a tanító adatokra túlilleszkedni. Az EarlyStopping segítségével egyszerűen leállíthatjuk a tanítást, amint a validációs hiba eléri a minimumot. Ennek használata a programunkban is célszerűnek bizonyult, itt egy küszöbérték lett meghatározva. Ez az érték esetünkben 5, vagyis ha 5 epochon keresztül növekedik a validációs hiba, akkor a tanítás leáll. A mindenkori legjobb megoldást menti el a program .hdf5 formátumban, végül ez a modell kerül használatra is.

A vonaldiagramokat, amelyek esetén az x tengelyen az epochok száma van megadva, az y tengely mentén pedig a modell hibája vagy pontossága van értelmezve, tanulási görbéknek nevezzük. Ezek a diagramok segítenek a diagnosztizálásában, hogy a modell túltanult, alultanult, vagy megfelelően illeszkedik-e a betanítási adatkészlethez. Az 5.8. ábrán különböző tanulási görbék vannak kirajzolva, melyek a modell megalkotása során keletkeztek. Ez csak pár szemléltető példa, az amúgy tágabb halmazból.



5.8. ábra *Tanulási görbék*

6. TESZTELÉS

Egy neurális hálót betanítottnak nevezhetünk, ha a tanító adatokra ideális kimenettel reagál. Ezt az állapot elérve következik a tesztelés fázisa, amikor olyan adatokkal vizsgáljuk a rendszer működését, amelyek a tanítás során nem kerültek alkalmazásra. A fejezetben a modellünk tesztelése, az eredmények kiértékelése került leírásra.

A tesztelés során a hiba vagy a pontossági mutatók felhasználásával mérlegelni tudjuk a hálót és annak eredményeit. A tesztelés elvégzésre került az elvárt és a tényleges kimeneti értékek összevetésével, illetve egyéb hasonló felépítésű rendszerek által produkált végeredményekhez viszonyítva.

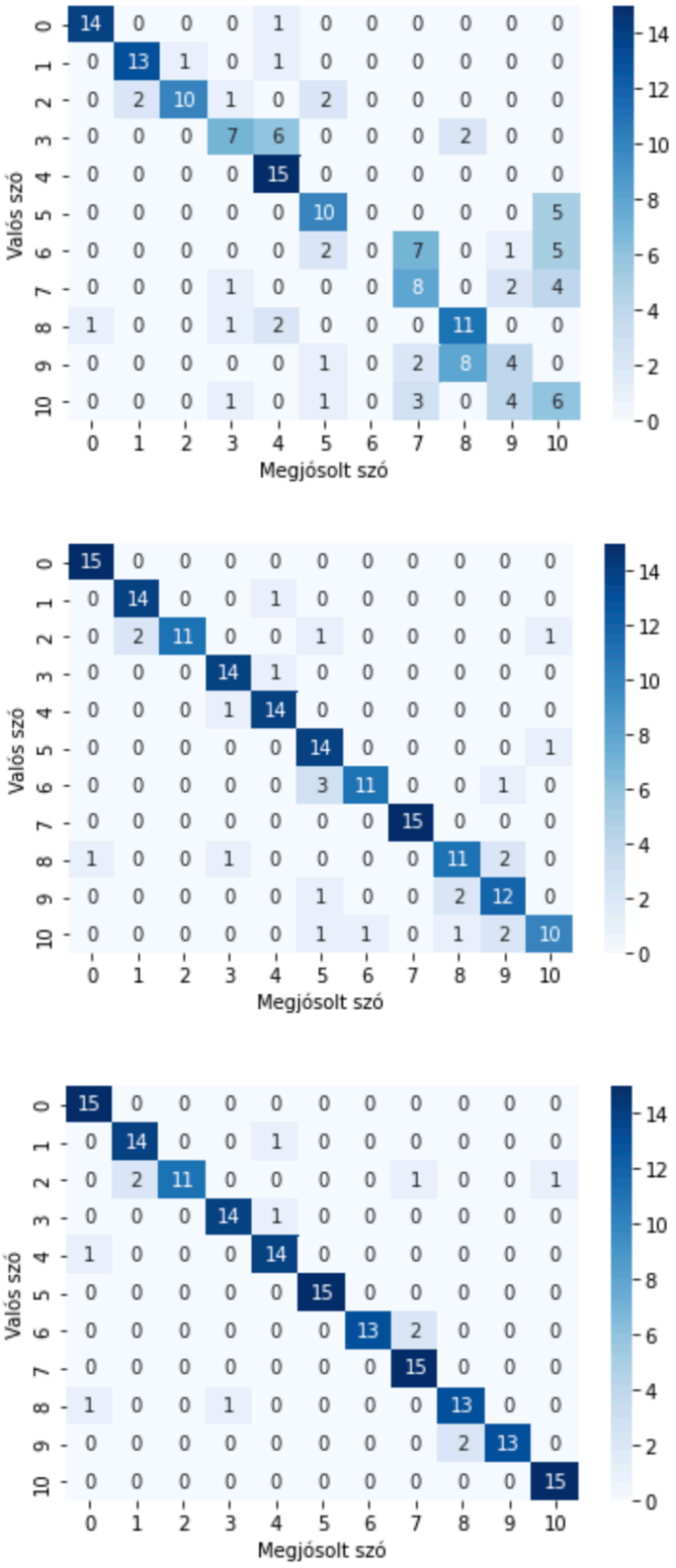
6.1. Tesztadatok használata

A Keras egy függvényt biztosít a számunkra, amely vezéreli a modell kiértékelését. Az evaluate() metódust segítségével képesek vagyunk a modell minősítésére, a függvény megadja nekünk a hiba és a pontosság mértékét. Az értékelés egy folyamat a modell fejlesztése során annak ellenőrzésére, hogy a modell mennyire illeszkedik az adott problémához és az adatokhoz. Ezáltal lehetőségünk van minden struktúra váltás és tanítás után az adott háló kiértékelésére és ezáltal a következtetések levonására.

A saját adatok létrehozása során elkészültek a tesztelésre szánt elemek gyűjteménye, amelyek a jelenlegi fázis során lettek használatba véve. A tanítás végeztével léptek ezek az elemek és a modell elsőkét kapcsolatba, az eredményekből pedig több információt is sikerült kinyerni. A teszteredmények egy .xlsx kiterjesztésű állományban lettek elmentve egy Drive mappában. Ezen eredmények vizsgálata többféleképp is elvégzésre kerültek, illetve különböző módszerek segítségével elemzésre kerültek.

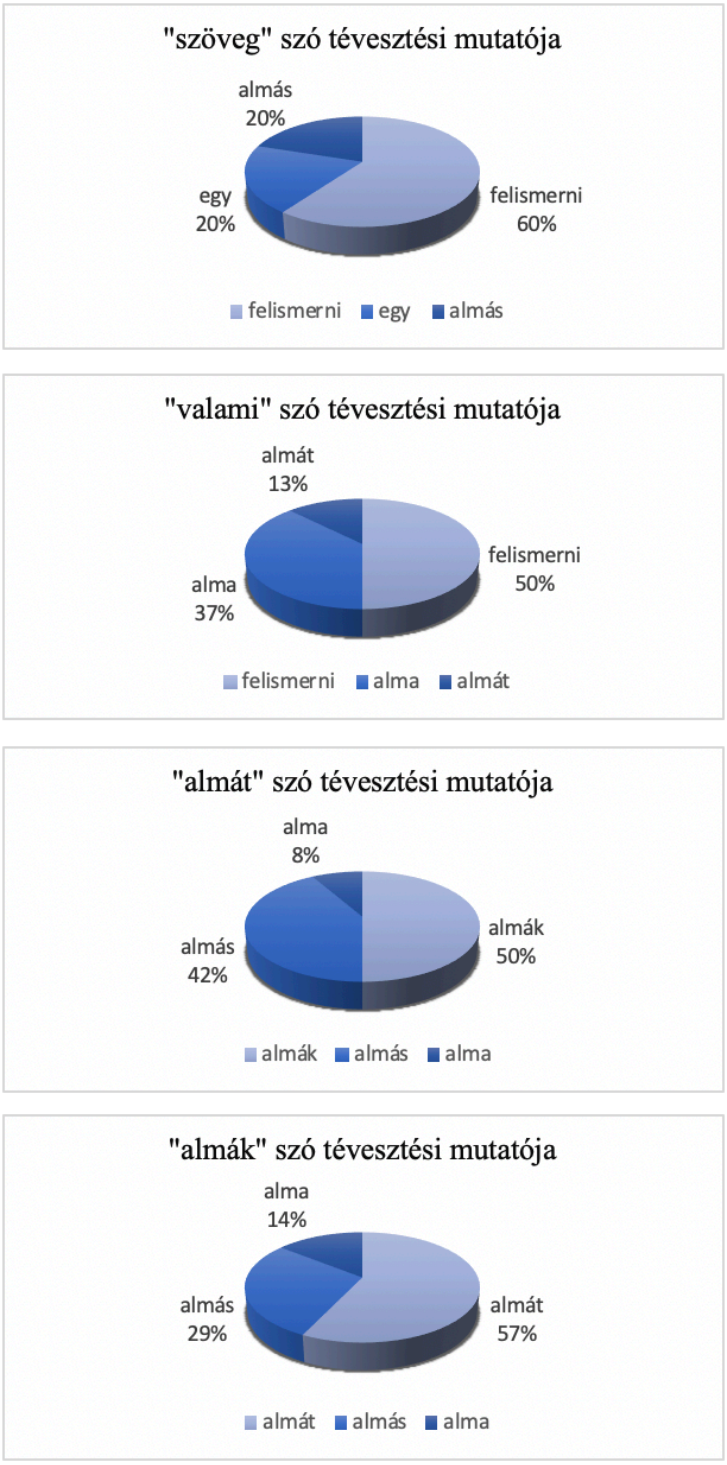
Ebben nyújtott segítséget a konfúziós mátrix használta. A konfúziós mátrix egy olyan táblázat, amelyet gyakran használnak az osztályozási modellek teljesítményeinek a leírására, olyan tesztadathalmazokon, amelyeknek a valódi értékeik ismertek. A konfúziós mátrix kicsi, de valójában sok információt hordoz magában. Segítségével egyszerűen meghatározható számos az osztályozást jellemző jósági paraméter. A 6.1. ábra szemlélteti a projekt megalkotása során lementett valamely konfúziós mátrixok képeit.

A konfúziós mátrix ezen lekepézésében a két tengelyen a megjósolt szavak és a valódi szavak indexei találhatók. A konfúziós mátrix diagonálisában találhatóak az úgynevezett true postive értékek. Ezek azok a kimenetek, amelyeket a modell helyesen klasszifikált, vagyis ahol a megjósolt szó és a valódi szó értéke megegyezik. A végső modell esetén a jósága százalékos értékben kifejezve, vagyis hogy az összes teszt adatból mennyit sikerült a hálónak helyesen detektálni, ez a szám 85,45%.



6.1. ábra *Konfúziós mátrixok*

A különböző kimeneteket összevetve lehetőségünk volt maguknak a tévesztéseknek az összetételeit is vizsgálat alá venni. Valamennyi modell eredményeit összemérve azt látszott, hogy a legtöbbet tévesztett elemek a „szöveg”, „valami”, „almát” és az „almák” szavak voltak. A 6.2. ábra az ezen szavak kimeneteinek a feldolgozásából készült grafikonokat mutatja be.



6.2. ábra *Téves kimenetek összetétele valamely szavak esetén*

A diagramokon az látszik, hogy a különböző modellek téves kimeneti milyen szavakból épült fel és azok milyen arányban oszlanak el. Itt érdemes megvizsgálni a hasonló alakú szavak tévesztési mutatóit, amiből jól látszik, hogy a hibás detektálás során a valós szóhoz merőben egyező szavak kerültek ki a hálóból.

Igaz a rendszer egy mesterséges környezetben lett felépítve, ahol az emberi folyóírást kézírásos betűtípusok használatával helyettesítjük. Azonban az előfeldolgozás használatával képesek vagyunk a valós kézírásról készült képeket is a rendszer bemenetének megadni, így lehetőség nyílik ezen elemek tesztelésére is. A 6.3. ábra képein néhány összegyűjtött kézírásos szó látható, az eredeti kép alatt a feldolgozott kép is megjelenik.

6.3. ábra *Kézírásos képek az előfeldolgozás előtt és után*

Egy kisebb gyűjtemény készült a különböző kézírásos képek eltárolására, a valós összegyűjtött elemek mindegyike az „alma” szót tartalmazza. Összesen 40 darab kép került elemzésre a tesztelés ezen fázisában, az összesített eredmény pedig a 6.4. ábrán látható. A háló pontossága a detektálás ezen szakaszában 48%-os értéket képvisel, hiszen a bemenő képek ilyen aránnyal kerültek sikeresen felismerésre. Érdemes megfigyelni a hasonló alakú szavak megoszlását a téves kimeneti adatokban, hiszen jelen esetben is azok előfordulása számottevően nagyobb, mint az egyéb fellelhető szavak aránya.

6.4. ábra "alma" szó felismerésének eredményei

Itt érdemes megemlíteni a tényt miszerint, ha a fényképezett elemek hangsúlyosabban lennének jelen a tanító adatok között, úgy ezeknek a felismerési rátája is javulna. Az előfeldolgozást érdemes lenne módosítani olyan irányba, hogy az árnyékok, illetve a megvilágításbeli differenciák jobban kiegyenlíthetők legyenek. Ekkor az adaptív binarizálás kerülne előtérbe, hiszen a képeken megjelenő kiugró értékeket és lokális különbségeket képesek lennénk kezelni. A másik felmerülő kiegészítés a képen lévő üres részek levágása, vagyis az objektum széleinek megkeresése, majd a kép megfelelő vágása.

6.2. Hasonló rendszerek

A tesztadatok felhasználása, illetve az összegyűjtött kézírásos szövegek elemzése után a soron következő feladat az egyéb hasonló felépítésű rendszerek által produkált eredmények elemzése volt. Sikerült megtalálni azokat az API-kat, amik képesek a folyóírás felismerésére. Az aktuális eredmények használatára fókuszálva, illetve a projekt igényes megvalósítása érdekében kettő valóban naprakész rendszer került elemzésre ebben a fázisban.

Az Azure Computer Vision és a Google Vision AI szolgáltatások azok, amik támogatják a magyar nyelvű írásfelismerést, következésképp ezen egységek kimenetei kerültek összehasonlításra a saját hálónk eredményeivel. Egy rövidebb rendszer bemutatást követően leírásra kerültek a tesztelések eredményei, majd megtörtént az összehasonlítás a saját modellünkkel, végül pedig az értékelés következett.

Mindkét rendszer alapesetben egy fizetős szolgáltatásnak tekinthető, ugyanis bizonyos összeg ráordítása esetén érhetők el a funkciók, amivel a szövegfelismerés elvégezhető. A stabilabb és hosszabbtávú működéshez elengedhetetlen egy előfizetés megléte. Azonban létezik ingyenesen használható demo verzió is, amivel képesek vagyunk tesztelni a szolgáltatásokat. Ezek azonban csak meghatározott ideig érhetők el és megszabott API hívási kerettel rendelkeznek.

6.2.1. Azure Computer Vision

Az Azure Cognitive Services dokumentációjának [24] a tanulmányozása során a következő ismereteket szerezhetjük meg a szolgáltatásról. A Cognitive Services minden fejlesztő számára elérhetővé teszi a mesterséges intelligencia használatát, ami mindössze egy API hívásba kerül. Az Azure Cognitive Services részeként elérhető Computer Vision használatával felhőbeli látástechnológiai képességeket ágyazhatunk be az alkalmazásokba, amivel automatizálhatjuk a szövegek kinyerését. Többféle nyelven és stílusban írt nyomtatott és kézírásos szöveget nyerhetünk ki képekből és dokumentumokból.

Az Optikai karakterfelismerés szolgáltatás szöveget nyer ki a képekből. A Read API-val nyomtatott és kézzel írt szöveget nyerhetünk ki fényképekből és dokumentumokból. Mélytanuláson alapuló modelleket használ, és különféle felületeken és háttereken dolgozik a szöveggel. Microsoft Read OCR motorja több fejlett gépi tanulási alapú modellből áll. [24]

Az API hívás a Google Colab platformján keresztül könnyen véghezvihető művelet, hiszen itt egyszerűen hozzá tudunk férni a tesztadatokhoz, illetve a kimenetek elmentése is elvégezhető. Ezáltal egy függvény híváson keresztül meg tudjuk kapni a szükséges eredményeket.

A Computer Vision használatba lett véve mind az általunk létrehozott tesztadatokon, mind a kézzel írt szavak képein. A rendszer jósága a teszt állomány felismerésének tekintetében 77,58%, míg a kézírásos elemek esetében 82,2%. Ezeknek a jelentősége, hogy a tesztadatok gyűjteménye 11 különféle szót tartalmaz, amik magas aránnyal felismerésre kerültek, az ékezetes betűket tartalmazó szavakat is beleértve. A detektálás sikerességének a szórása 8,6% amiből arra következtethetünk, hogy a rendszer valóban eredményesen ismeri fel a különböző szavakat. A kézírásos képek esetén azonban csupán egy szó felismerésének a hatékonyságát mértük, ennél leginkább az

előfeldolgozási lépések elemzése volt a cél. Mindazonáltal az jól látszik, hogy a szolgáltatás valóban sikeresnek mondható az írott szöveg felismerésének tekintetében.

6.2.2. Google Vision AI

A Google Vision AI előre betanított gépi tanulási modelleket kínál API hívásokon keresztül. A Cloud Vision segítségével könnyedén integrálhatók a látásérzékelő funkciók az alkalmazásokba, beleértve az optikai karakterfelismerést. Számos nyelven írt nyomtatott és kézírásos szöveget nyerhetünk ki képekből. Felismeri és kivonja a szöveget a képen belül, számos nyelv támogatásával. Automatikus nyelvazonosítóval is rendelkezik. [25] Az API hívás szintén a Colab platformján keresztül zajlik.

A Cloud Vision esetében ugyan azok a lépések lettek elvégezve, vagyis az általunk létrehozott tesztadatok, illetve a kézzel írt szavak képei is elemzésre kerültek. A rendszer jósága a teszt állomány felismerésének tekintetében 71,52%, míg a kézírásos elemek esetében 80%. A tesztadatok sikeres detektálásának a szórása 17,9%. A Cloud Vision szolgáltatás is sikeresnek tekinthető az írott szöveg felismerésében.

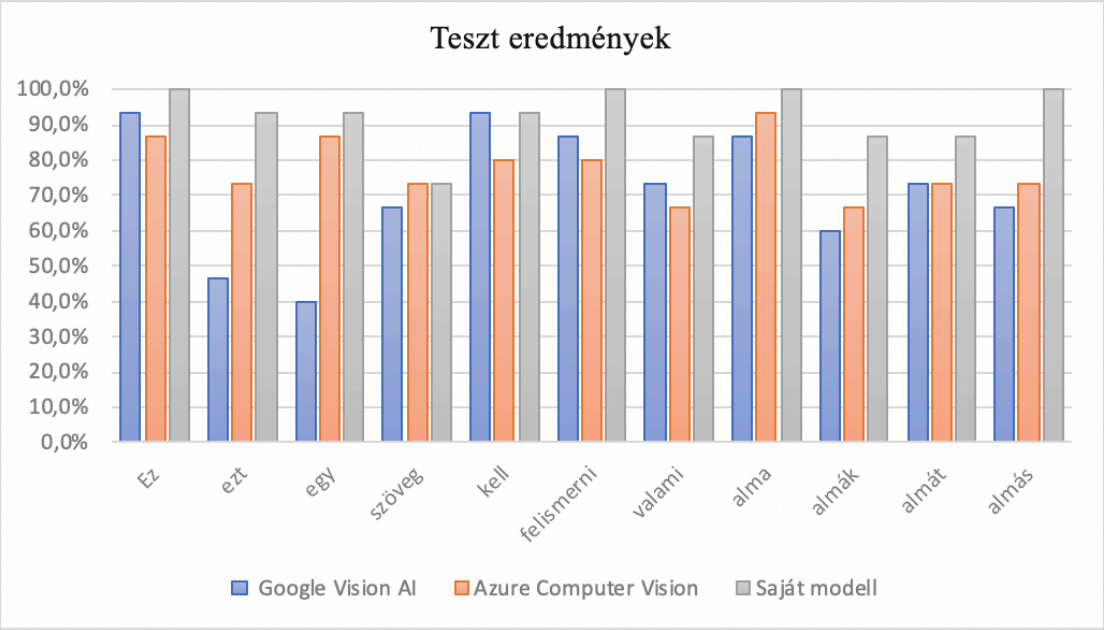
6.2.3. Eredmények értékelése

A felkutatott programok kipróbálását követően az alábbi megállapításokat tehetjük. Mindkét rendszer eredményesnek mondható a megvizsgált szempontok alapján, vagyis magas aránnyal képesek voltak a kézzel írt szövegről készült képek elemzésére. Mindkét esetben mélytanuláson alapuló modellek állnak a háttérben, amik képfeldolgozási algoritmusokkal vannak kiegészítve. Az eredményeken látszott, hogy a rendszerek nem mindig adnak nyelv specifikus megoldást, illetve nem törekszenek egy valós szó megadására, a karaktereket külön próbálják meg felismerni majd összeilleszteni.

A 6.5. ábrán a tesztadatok sikeres detektálásának az arányszámai láthatók a különböző rendszerek esetében. Igaz a saját modellünk által elért eredmények valóban eredményesnek mondható, de fontos szem elött tartani, hogy a teszt adatok értékei egyértelműen a rendszerünk kimeneti értékkészletének az elemeit tartalmazzák. Következésképp sokkal egyszerűbben képesek vagyunk jó eredményeket eléri a tesztelés során.

Mindazonáltal a lényeg jól látszik, vagyis egy jól működő mély neurális hálózat létrehozása a jelenlegi eszközökkel gond nélkül kivitelezhető. Amit a problémát körül jártuk és megalkotásra került a megfelelő struktúra, úgy a háló felépítését és tesztelését már számos optimális algoritmus segítségével elvégezhetjük. Egy megfelelő rendszer létrehozása rengeteg időt és erőforrást igényel, azonban a jelenlegi technológiai háttérrel már gördülékenyen el lehet indulni.

Elmondható, hogy a szakszerű rendszerek ugyan egy magas színvonalú megvalósítást képviselnek, azonban a hétköznapi használatra még nem állnak készen, nehezen lehet ezeket beszerezni, elérni. Eredményeik emberi ellenőrzést igényelnek, azonban sok művelet során segítséget nyújtanak.



6.5. ábra *A tesztadatok sikeres detektálásának arányai*

7. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A folyóírás felismerése számos lehetőséget kínál mind a tervezés, mind a megvalósítás, mind pedig a felhasználás terén. Egy igazán nagyméretű témakörről beszélhetünk, ami rengeteg oldalról megközelíthető. A szakdolgozat is egyfajta nézőpontot képvisel, amit többféleképp is tovább lehet még gondolni.

A projekt során élért eredményeket nézve nagyszámú megfontolandó szempont került elénk. A jelenlegi rendszert tovább lehetne fejleszteni egyrészt a szegmentálás irányában is. Sorok, szavak, illetve karakterek felbontása lenne ezáltal a cél, így az oldalnyi szöveg elemzése is elérhetővé válhat. A neurális hálók használata a detektálás során minden bizonnyal egy stabil alapot adhat, így ajánlott ezzel a technológiával tovább haladni. Azonban a neurális hálók témaköre is számtalan alternatívát kínál.

Egy hosszútávú terv is kirajzolódott a projekt megalkotása során, a diszgráfia felismerése mesterséges intelligencia segítségével. Ehhez azonban egy magas hatékonyságú írásfelismerő rendszer az előfeltétel. Ennek megléte esetén az íráselemzés során egy modell képes lenne felismerni az írásban fellépő gyakori hibákat, jellegzetességeket, így képesek lennénk a diszgráfia azonosítására. A diszgráfiához, mint tanulási zavarhoz hozzárendelhetők bizonyos „sémák”, vagyis azok a gyakori elemek, amiket a diszgráfiás egyének gyakran eltévesztenek. Ezeket a szoftver képes lenne felismerni, kijavítani, majd erről tájékoztatást adni.

8. Összefoglalás

A szakdolgozatban a fő feladat a folyóírás-felismerés aktuális eredményeinek feltárása, illetve bemutatása volt, majd egy lehetséges rendszer struktúra kiépítése amivel a folyóírás-felismerés megfelelően elvégezhető. Első lépésként megvizsgálásra kerültek az elérhető módszereket, majd kirajzolódtak a fő komponensnek az elmei, vagyis a mély neurális hálók használatának az alapjai.

Elsőként a dolgozat alapterületei kerültek elemzésre, az írásdigitalizálás, karakterfelismerés, a neurális hálózatok, illetve a mélytanulás. Majd a fellelhető technológiák elemzése után a saját rendszerterv összeállítása történt.

A szerzett ismeretekből kiindulva kiválasztásra került a konvolúciós neurális hálózati architektúra, később több konvolúciós struktúra is kiépítésre, illetve tesztelésre került. A tanítás elvégzését és az eredmények értékelését követően, a hiperparamétereket finomítva elkészült a végleges modell. Cél volt, hogy kisebb méretű neurális hálózattal oldjuk meg a feladatot, hiszen nagyobb háló méretével drasztikusan megnőne a tanítási idő is. A megalkotott rendszerek mindegyikénél megfigyelhető volt, hogy a rejtett rétegek mérete általában próbálgatásos módszerrel lett meghatározva. Végezetül számos hálózat sikeresnek bizonyult a felismerési feladat modellezésére.

Elmondható, hogy a folyóírás felismerésével egyre több technológia foglalkozik napjainkban és számos olyan rendszer elérhető, amivel már eredményesnek mondható kimenetekhez juthatunk. A szakdolgozat készítése során kipróbálásra került többféle szolgáltatás is amivel a szöveg detektálást elvégezhetjük, azonban a valós megoldásokat a nehezebben hozzáférhető robusztusabb modellek adták. Mindazonáltal ezek a rendszerek egy stabil alapot biztosítanak a különböző továbbfejlesztési lehetőségekhez.

A mély neurális hálózatok rendszere jelenleg egy hatalmas mozgatóerő a modern technológiák terén. Számos könnyen elérhető, egyszerűen megérthető és használható eszköz áll a rendelkezésünkre, hogy ezeket használatba véve saját modelleket alkossunk. A projekt során kiderült, hogy ezek a megalkotott hálózatok igen nagy hatásfokkal képesek működni.

Összeségében a magyar folyóírás felismerése még nem tekinthető egy hétköznapi folyamatnak, de számos előrehaladás van a témában. A terület fejlesztése számtalan lehetőséget nyújthat a kihasználását illetően, érdemes ebben a témában további kutatásokat végezni, időt szánni a megfelelő koncepció megtalálására.

9. ABSTRACT

In the thesis, the main task was to explore and present the current results of journal recognition, and then to build a possible system structure with which journal recognition can be performed properly. As a first step, the available methods were examined, and then the minds of the main component, i.e. the basics of using deep neural networks, were outlined.

First, the basic areas of the thesis were analyzed, such as digitization of writing, character recognition, neural networks, and deep learning. Then, after analyzing the available technologies, the own system plan was compiled.

Based on the acquired knowledge, the convolutional neural network architecture was selected, and later several convolutional structures were built and tested. After completing the teaching and evaluating the results, the final model was completed by refining the hyperparameters. The goal was to solve the task with a smaller neural network, since the teaching time would increase drastically with a larger network size. In all of the created systems, it was observed that the size of the hidden layers was usually determined by trial and error. Finally, several networks have proven successful for modeling the recognition task.

It can be said that more and more technologies are dealing with journal recognition these days and many systems are available that can be used to achieve results that can be said to be effective. During the preparation of the thesis, various services with which text detection can be performed were tested, however, the real solutions were provided by the more difficult-to-access more robust models. However, these systems provide a stable basis for various further development options.

The system of deep neural networks is currently a huge driving force in modern technologies. We have many tools that are easily available, easy to understand and use, so that we can use them to create our own models. During the project, it became clear that these created networks can operate with a very high degree of efficiency.

Overall, the recognition of Hungarian handwriting cannot yet be considered an ordinary process, but there is a lot of progress on the subject. The development of the area can provide countless opportunities for its utilization, it is worthwhile to carry out further research on this topic and take the time to find the right concept.

IRODALOMJEGYZÉK

[1] I. Fazekas, *Neurális hálózatok,* Debrecen: Debreceni Egyetem Informatikai Kar, 2013.

[2] „Build a Handwritten Text Recognition System using TensorFlow," [Online]. Available: https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5. [Hozzáférés dátuma: 2021 04 10].

[3] Nusratov és Geydarov, *Position-Width-Pulse Algorithm for Recognition of Handwritten Characters,* 2007.

[4] D. B. d. K. Piroska, *Ismerkedés a TensorFlow rendszerrel.*

[5] Rupsa Chakraborty és Jaya Sil, *Handwritten Character Recognition Systems Using Image-Fusion and Fuzzy Logic,* 2005.

[6] G. Koncz, „Kézzel írt szavak digitalizálása neuronhálók segítségével," Budapest, 2018.

[7] „Mi az OCR technológia?," [Online]. Available: https://ocrszoftver.hu/mi-az-ocr-technologia. [Hozzáférés dátuma: 2020 10 10].

[8] „Intelligent character recognition," [Online]. Available: https://en.wikipedia.org/wiki/Intelligent_character_recognition. [Hozzáférés dátuma: 2020 10 12].

[9] „What is IWR?," [Online]. Available: https://www.efilecabinet.com/what-is-iwr-intelligent-word-recognition-how-is-it-related-to-document-management/. [Hozzáférés dátuma: 2021 03 10].

[10] C. Szabó, *Kézzel írt szövegek feldolgozása tanuló algoritmusokkal,* Debrecen, 2007.

[11] „Neurális hálózatok elrendezési és funkcionális típusai," [Online]. Available: https://mesterin.hu/neuralis-halozatok-elrendezesi-es-funkcionalis-tipusai/. [Hozzáférés dátuma: 2021 03 14].

[12] G.-T. BÁLINT, A mélytanulás múltja, jelene és jövője, Budapest, 2019.

[13] M. Altrichter, G. Horváth, B. Pataki, G. Strausz, G. Takács és J. Valyon, *Neurális hálózatok,* Budapest: Hungarian Edition Panem Könyvkiadó Kft., 2006.

[14] „Mélyreható tanulás és gépi tanulás a Azure Machine Learning," [Online]. Available: https://docs.microsoft.com/hu-hu/azure/machine-learning/concept-deep-learning-vs-machine-learning. [Hozzáférés dátuma: 2021 03 12].

[15] G. Swinnen, *Tanuljunk meg programozni Python nyelven,* 2005.

[16] „Python,” [Online]. Available: https://mesterin.hu/6-ok-amiert-a-python-a-jovo-programozasi-nyelve/. [Hozzáférés dátuma: 2021 4 12].

[17] „TensorFlow,” [Online]. Available: https://www.tensorflow.org. [Hozzáférés dátuma: 2022 10 10].

[18] „GoogleColab,” [Online]. Available: https://research.google.com/colaboratory/faq.html. [Hozzáférés dátuma: 2022 10 12].

[19] K. Zoltán, *Szűrés képtérben,* Képfeldolgozás és Számítógépes Grafika tanszék.

[20] „OpenCV,” [Online]. Available: https://hu.wikipedia.org/wiki/OpenCV. [Hozzáférés dátuma: 2022 10 10].

[21] „A guide to an efficient way to build neural network architectures,” [Online]. Available: https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7. [Hozzáférés dátuma: 2022 11 10].

[22] „Activation Functions in Neural Networks,” [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Hozzáférés dátuma: 2022 11 12].

[23] „Epoch vs Batch Size vs Iterations,” [Online]. Available: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9. [Hozzáférés dátuma: 2022 11 22].

[24] „Az Azure Cognitive Services dokumentációja,” [Online]. Available: https://learn.microsoft.com/hu-hu/azure/cognitive-services/. [Hozzáférés dátuma: 2022 10 12].

[25] „Vision AI,” [Online]. Available: https://cloud.google.com/vision. [Hozzáférés dátuma: 2022 12 02].

[26] „Convolutional Neural Networks for Visual Recognition,” [Online]. Available: https://cs231n.github.io/neural-networks-1/.. [Hozzáférés dátuma: 2021 03 10].

[27] „Online ICR Services,” [Online]. Available: https://www.indiamart.com/proddetail/icr-services-8865474933.html.. [Hozzáférés dátuma: 2020 11 20].

ÁBRAJEGYZÉK