TDK-dolgozat

Pap Martin Felicián

Széchenyi István Egyetem Gépészmérnöki, Informatikai és Villamosmérnöki Kar
Informatika tanszék

TDK

Egymást átfedő kézzel írott karakterek felismerése neurális hálózatok segítségével

Pap Martin Felicián
[Mérnöktanár-informatika MA]

Konzulens:
Hollósi János
egyetemi tanársegéd

[2022]

Kivonat

Egymást átfedő kézzel írott karakterek felismerése neurális hálózatok segítségével

A dolgozat célja az elterjedt neurális hálózati architektúrák kiértékelése, részben vagy teljesen egymást átfedő számjegyeket tartalmazó adatsoron. Cél, hogy a hagyományos MNIST adatsor felismeréséhez képest láthatóvá váljon, hogy az átfedés növekedésével, hogyan változnak a különböző modellek hatékonysági mutatói. A dolgozatban az MNIST adathalmazon készülnek mérések, ami kézzel írott felcímkézett számjegyeket tartalmaz. Az adatsor kiválasztásánál fontos szempont volt, hogy a területen elterjedt és gyakran használt, szakmailag is elfogadott adatokon történjenek a mérések. A feladathoz az eredeti MNIST halmaz átalakításra került és így ált össze a részben vagy egészében egymást átfedő számjegyek halmaza. Napjainkban számos terület digitalizáción megy keresztül, azonban számos olyan folyamat van, ahol gyakorlati okok miatt az adatok nem digitális módon állnak elő. Ezekben az esetekben kell megoldást találni az adatok nagy hatékonyságú digitalizálására, hogy a modern eszközökkel ezeket fel lehessen használni és új hasznosíthatósági lehetőségek kerüljenek meghatározásra. A dolgozatban keletkező eredmények betekintést adnak arról, hogyan hat az átfedés mértékének növekedése a pontosságra. A dolgozat során keletkező eredmények várhatóan segítenek előkészíteni későbbi kutatásokat, ahol a különböző modellek hiperparaméterezéseire, finomhangolásukra lehet fókuszálni, valamint a gyakorlatba történő alkalmazásra.

Tartalomjegyzék

Kivonat 2

1. Bevezetés 5

1.1 A dolgozatban vizsgált probléma és a dolgozat céljának bemutatása 5

1.2 A dolgozat felépítése 6

2. A feladathoz kiválasztott adathalmaz bemutatása 7

2.1 A feladathoz létrehozott adatkészlet 8

3. Felhasznált neurális hálózat rétegek bemutatása 9

3.1 Input 9

3.2 Flatten 9

3.3 Dropout 9

3.4 Dense 9

3.5 Aktiváció 11

3.6 Conv2D 12

3.7 MaxPooling2D 13

3.8 BatchNormalization 14

3.9 GlobalAveragePooling2D 15

4. Alkalmazott neurális hálózat architektúrák 16

4.1 DenseNet 16

4.2 LeNet 16

4.3 ResNet50 17

4.4 ResNet50v2 20

4.5 VGG19 21

5. Hálózatok tanítása, eredmények 23

5.1 Tanítás 23

5.2 Tesztelés. 24

6. Konklúzió 33

7. Irodalom Jegyzék 34
8. Ábrajegyzék 36
9. Táblázatok 37

1. Bevezetés

A dolgozat célja az elterjedt neurális hálózati architektúrák kiértékelése, részben vagy teljesen egymást átfedő számjegyeket tartalmazó adatsoron. Cél, hogy a hagyományos MNIST[1] adatsor felismeréséhez képest láthatóvá váljon, hogy az átfedés növekedésével, hogyan változnak a különböző modellek hatékonysági mutatói.

1.1 A dolgozatban vizsgált probléma és a dolgozat céljának bemutatása

Napjainkban egyre nagyobb szerepet kap a különböző folyamatok digitalizációja. Egyre több területre jellemző a hagyományos működés átalakítása. Ezzel párhuzamosan az adatok is egyre gyakrabban digitálisan jönnek létre. A digitális adatok nagy előnye, hogy nagy adatmennyiség esetén is hatékonyan lehet őket rendezni és feldolgozni. A nagy mennyiségben előálló adatok segítségével mélyebben megismerhetővé válnak a folyamatokat. Számos területen azonban nagy mennyiségű nem digitális, többnyire kézzel írott adat áll rendelkezésre, amit a hasznosíthatóság érdekében digitalizálni szükséges. A kézzel írott adatok digitalizálása sajátos problémát vett fel. A kézírás felismeréséhez a bemeneti kép több jellemzőjét is figyelembe kell venni, ami a legtöbb esetben hagyományos informatikai megoldással alacsony hatékonysággal valósítható meg. Az ehhez hasonló képfeldolgozási feladatokhoz előszeretettel használt és ígéretes eredményeket mutatnak a különböző gépi tanulási azon belül is a Neurális hálózat alapú megoldások. A kézi karakterek felismerésének külön problémája, hogy nem elég elkülönített karaktereket felismerni hiszen a gyakorlatban a kézzel írott szövegek, számok esetén az egyes karakterek gyakran részleges átfedésben vannak. A Neurális hálózat alapú megoldásokat ezért nem elég a karakterek felismerésére betanítani, az átfedést is tudnia kell kezelni.

Számos területen a digitalizáció kiváltja a kézzel írott adatokat. Azonban több olyan folyamat is létezik, ahol a kézzel írott adatok sokkal praktikusabbak a mindennapi munkavégzés esetén. Ezeken a területeken aktívan használható rendszerekre van szükség, amik képesek az adatok nagy pontosságú digitalizálására, hogy mélyebb információk kinyerésére legyen lehetőség, illetve, hogy az alapvető feldolgozás is felgyorsuljon.

A gépi tanulás, mint terület egy gyorsan fejlődő ágazat. A gyors fejlődés egyik hátránya, hogy a létrejövő megoldások nem minden esetben kerülnek kellő képpen feltárásra a bennük

lévő hasznosíthatósági potenciál nincs minden esetben kiaknázva. A dolgozatban az MNIST adathalmazon készülnek mérések, ami kézzel írott felcímkézett számjegyeket tartalmaz. Az adatsor kiválasztásánál fontos szempont volt, hogy a területen elterjedt és gyakran használt, szakmailag is elfogadott adatokon történjenek a vizsgálatok.

1.2 A dolgozat felépítése

A dolgozat 6 fejezetből áll. Ezekben a fejezetekben kerülnek bemutatásra a használt megoldások és azok eredményei.

A második fejezetben kerül bemutatásra a kiválasztott adathalmaz és annak átalakítás.

A harmadik fejezetben a különböző neurális hálózati architektúrák által használt rétegek és azok működései kerülnek bemutatásra.

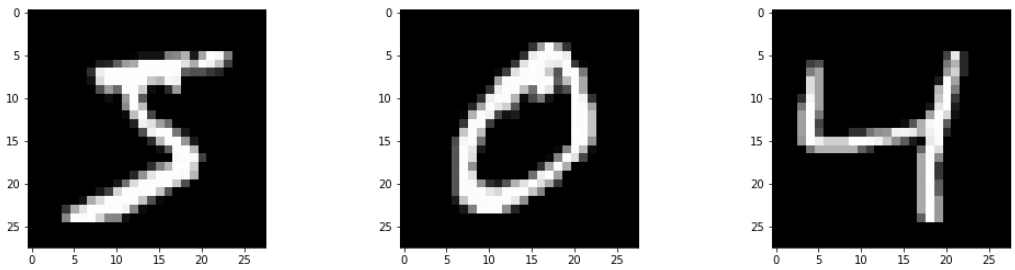
A negyedik fejezet a kiválasztott neurális hálózatok felépítését mutatja be. Ebben a fejezetben került kifejtésre mindegyik modell legfontosabb tulajdonsága, valamint a megvalósítás során használt paraméterek és azok értékei.

Az ötödik fejezet mutatja be a mérések eredményeit. Modellekre bontva és összesítve is megjelenítésre kerültek a különböző metrikákkal kapott eredmények.

A hatodik fejezet a dolgozat általános konklúzióját tartalmazza és további kutatási lehetőségeket vet fel.

2. A feladathoz kiválasztott adathalmaz bemutatása

Az MNIST egy nagyméretű adatbázis, amely kézzel írt számjegyeket tartalmaz. Legtöbbször képfeldolgozó rendszerek feltanítására használják. Az adatbázis mérete és megbízhatósága miatt népszerű tanítási és tesztelési feladatok elvégzésére a gépi tanulás területén. Az MNIST adatkészletet Yann LeCun, Corinna Cortes és Christopher J.C. Burges hozták létre az eredeti NIST adatkészletből származó minták újrakeverésével. Az MNIST egy mozaikszó, amely a Modified National Institute of Standards and Technology rövidítése. Az adatkészlet 60 000 szürkeárnyalatos képet tartalmaz. Mindegyik képen kézzel írt számjegyek találhatóak 0 és 9 között. A képeket méretükben és a rajtuk lévő számjegyek pozícióját tekintve standardizáltnak tekinthető. A számjegyek minden képen középre vannak igazítva egy 28×28 -as négyzeten belűl. Az adatkészlet a standardizálás mellett fel is van címkézve így minden képhez egyértelműen hozzápárosítható a saját címkéje, amin a valós számjegy szerepel. Az egyértelműen kialakított kép-címke pároknak köszönhetően az adatkészlet kiválóan használható supervised (felügyelt) klasszifikációs (osztályozó) feladatokhoz modellek feltanítására. A gépi tanulás egyik nagy problémája, hogy nagy mennyiségű és jó minőségű adatra van szükség. Ezeknek előállítása felcímkézése és standardizálása sok időt és erőforrást igényel. Az MNIST egy jó példa arra, hogy egy jó adathalmaz, hogyan tudja segíteni a gépi tanulás fejlődését, azzal, hogy stabil, megbízható és nagy méretű adathalmazt biztosít így az új ötleteknél elég az architektúrák kidolgozására fókuszálni. Az MNIST igazán hasznos azok számára, akik szeretnék megismerni a gépi tanulási technikákat, mintafelismerési módszereket és tesztelni akarják saját megoldásaikat. Az adathalmaznál egy tanítás esetén 60 000 magára a tanításra míg 10 000 kép a teszteléshez van használva. Az adathalmaz publikusan elérhető a http://yann.lecun.com/exdb/mnist/ weboldalon.

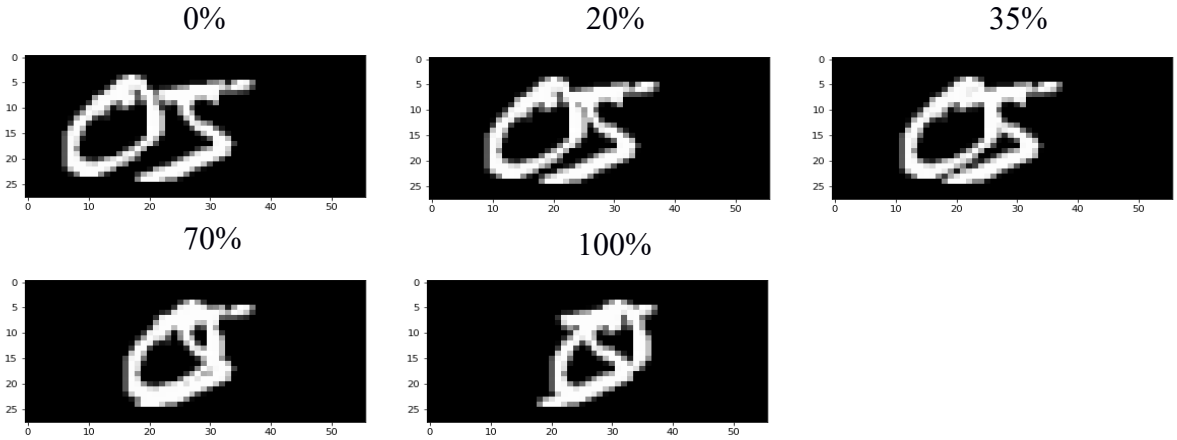


1. ábra Minták az MNIST adathalmazból.

2.1 A feladathoz létrehozott adatkészlet

A feladathoz az eredeti MNIST adathalmaz, amelynek minta elemei az 1. ábrán láthatóak átalakításra került annak érdekében, hogy az átfedés hatásait lehessen vizsgálni. Az adathalmaz a Keras könyvtárból került importálásra tanítási és tesztelési adatcsoportokra bontva. Mind a tanítási adatokhoz mind pedig a tesztelésre szolgáló adatokhoz betöltésre kerültek a címkék is. Ezt követően definiálásra kerültek az átfedés paraméterei. Fontos szempont volt, hogy a teljesen átfedő és a teljesen külön álló karakterek mellett több átfedési szinttel is generálásra kerüljenek minták. A követhetőség érdekében minden átfedési szinttel teljesen külön álló tanítási és tanulási adatsorok kerültek legenerálásra. Mivel egy MNIST kép 28×28-as méretű így a feladathoz 28×56-os képek kerültek kialakításra, amin minden esetben 2 karakter van adott szintű átfedéssel egymáshoz képest. Az átfedések generálásánál első lépésben az alap MNIST adathalmazból kiválasztásra kerül 2 kép. Az első kép középre igazításra kerül egy 28×56-os üres mátrixba. Ezt követően a másik bal szélre kerül elhelyezésre úgy, hogy az előre megadott átfedési mértéknek megfelelően a 28×56-os mátrix közepe felé van csúsztatva. A végső képen megjelenő szám a bemeneti 2 szám uniójaként áll elő. A vizsgálathoz a következő átfedési értékek kerültek meghatározásra: $0\%, 20\%, 35\%, 70\%, 100\%$ Ezzel a módszerrel 5 különböző tanító és teszt halmaz jött létre. Mindegyik tanító adatsor $60\,000$ elemet tartalmaz míg minden teszt adatsor $10\,000$ elemmel rendelkezik.

A 2. ábra jól szemlélteti, hogy a különböző átfedési értékek, hogyan befolyásolják az eredményt.



2. ábra átfedés generálása után kapott minta adatok. Az átfedés mértékét az adatok felett feltüntetett szám jelzi.

3. Felhasznált neurális hálózat rétegek bemutatása

3.1 Input

A modell ezzel az objektummal hozza létre a tensort. A feladatban a paraméterezés során a shape argumentum került meghatározásra. A bemeneti adatok alapján a mátrix $28 \times 56 \times 1$ formában került megadásra, ahol a 28 és az 56 a kép dimenziót jelölik az 1-es pedig a szín csatornákat határozza meg, mivel a bemeneti adatok szürkeárnyalatosak így 1 színcsatornával tud dolgozni a rendszer.

3.2 Flatten

A Faltten réteg kisimítja az Inputból érkező adatokat. A 2 dimenziós 28×56-os mátrixból 1 dimenziós 1568 méretű kimeneti mátrixot állít elő.

3.3 Dropout

A Dropout[2] a túltanulás (overfitting) jelenségét segíti minimalizálni. Túltanulás esetén a hálózat erősen csak a tanuló mintákra fókuszál és így csökken az általánosító képessége és a valódi mintákon sokkal rosszabb teljesítményt fog nyújtani A dropout segítségével egy előre meghatározott mérték szerint véletlenszerűen kiválasztott neuronokat nem vesz figyelembe a hálózat a tanulási folyamat során. Ezek úgymond kidobásra kerülnek. A dropout segítségével elkerülhető, hogy nagyon specifikus jellemzőkre tanuljon rá a rendszer. A dropout réteg hatása a tanítás során kerül érvényesítésre. A következtetés során ez a réteg nincs használatban. A feladat során a rate argumentum került meghatározásra $0{,}4$-es értékkel.

3.4 Dense

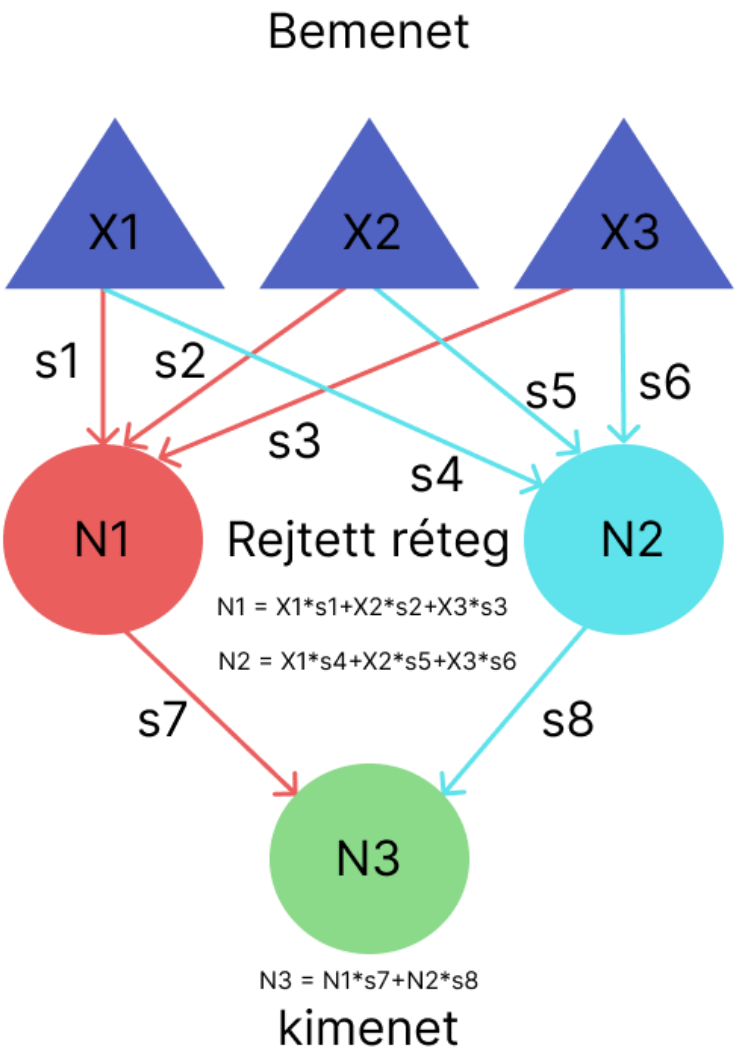
A Dense[3] réteg az egyik legalapvetőbb réteg a neurális hálózatoknál. A Dense Layer a következő működést valósítja meg:

$$o = \alpha(x \cdot k + \beta) \tag{1}$$

Ahol az o jelöli az eredményt az α az aktivációs függvényt, x a bemenetet, k a kernelt és β

a torzítást.

A Korábbi Flatten segítségével vektorrá alakítottuk a bejövő adatokat, ezzel egyszerűsítve a skaláris szorzás műveletét. Az aktivációs függvény argumentumként kerül megadásra, a bias értékek pedig csak akkor kerülnek a réteg által meghatározásra, ha az argumentumban True értékre van állítva. A bemenő adatokat a units argumentummal lehet megadni, a kívánt kimenet méretét.



3. ábra A Dense réteg működése, X a bemenet, N a neuron, s a súly

A Dense layer a teljesen összekapcsolt réteget valósítja meg. Ez azt jelenti, hogy bemenetként az előző réteg összes neuronjának kimenetét kapja. Az egyes neuronok értékei a bejövő értékek és a hozzájuk tartozó súlyok szorzatának összegeként értelmezhető. ahogyan az a 3. ábra is szemlélteti. A probléma megoldás során a Dense layer a következő

képpen került paraméterezésre:

Az Aktivációs argumentumot alapértelmezetten hagyjuk (NONE), az aktivációs a denst követő külön rétegben került megvalósításra. Argumentumként a units érték kerül megadásra amivel fokozatosan csökken a méret egészen addig amíg el nem éri a prediktálni kívánt osztályok számát.

3.5 Aktiváció

Az aktiváció nélküli neurális hálózat lényegében csak egy lineáris regressziós modell. Az aktivációs réteg elvégzi a bemenet nemlineáris átalakítását, ami képessé teszi a tanulásra és összetettebb feladatok elvégzésére. Az aktivációs[4] réteg számára argumentumként az aktivációs függvényt kell megadni, ami meghatározza milyen matematikai elv szerint történik az aktiválás.

A feladat során 2 aktivációs függvény került felhasználásra. A ReLU és a softmax. A ReLU a köztes aktivációs rétegekben került felhasználásra, míg a softmax az utolsó rétegnél a megfelelő osztály kiválasztásához vannak használva.

ReLU[5]

A ReLU (rectified linear unit) aktiváló argumentumának pozitív részével határozható meg:

$$f(x) = \begin{cases} x \; ha \; x > 0 \\ 0 \; ha \; x \; \leq 0 \end{cases} \tag{2}$$

ahol x egy neuron bemenete. Kimenete 0 ha az input kisebb vagy egyenlő 0-val, egyébként az inputot adja vissza.

A ReLU-nak több változata is van. Az egyes változatok között a használt függvény a különböző, aminek eredménye, hogy a negatív értékekere nem 0-ad vissza. Egyes esetekben ezzel további információkat tudunk kinyerni. A feladat során a hagyományos ReLU aktivációs függvény került beépítésre.

Softmax[6]

A softmax függvény egy K valós számból álló vektort K lehetséges kimenetelű valószínűségi eloszlássá alakít. Ez a logisztikus függvény több dimenzióra való általánosítása. A softmax függvényt gyakran használják neurális hálózatok utolsó aktiválási függvényeként, hogy a hálózat kimenetét a megjósolt kimeneti osztályok valószínűségi eloszlásra normalizálják.

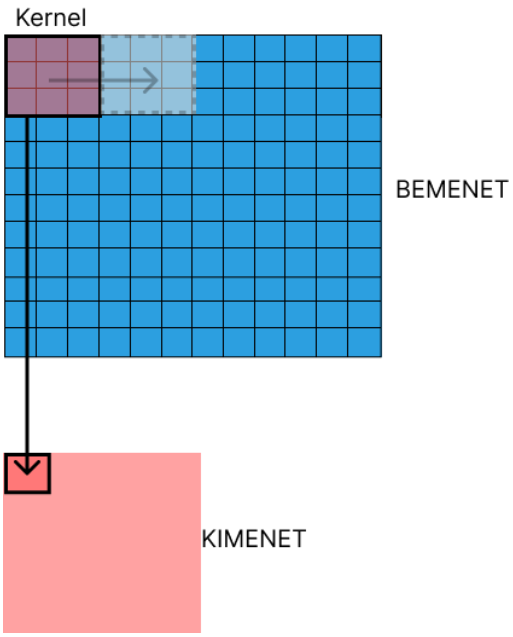
A V vektor x elemeinek softmax értékét a következőképpen számítjuk ki:

$$P_{vx} = exp(V_x) \,/\, \sum_{i=0}^{i} exp\,(V_i) \tag{3}$$

A vektor adott elemének exponenciális függvény értéke osztva a vektor összes elemének exponenciális függvény értékének összegével. A bemeneti vektor minden elemére elvégezve eredményül kapunk egy vektort, ami az egyes osztályokhoz tartozó valószínűségi értékeket tartalmazza. A kimeneti vektor maximuma értelmezhető a neurális hálózat által prediktált osztályként.

3.6 Conv2D

A konvolúciós[7] neurális hálózat központi eleme a konvolúciós réteg, amely a hálózat nevét adja. Ez a réteg egy "konvolúciónak" nevezett műveletet hajt végre. A konvolúció egy lineáris művelet, amely egy súlykészlet és a bemenet szorzatát jelenti. Tekintettel arra, hogy a művelet kétdimenziós bementi adatra hajtódik végre, a szorzás a bemeneti adatok egy tömbje és egy kétdimenziós súlytömb, az úgynevezett kernel között történik.



4. ábra Konvolúciós réteg vizualizálása

A 3. ábra a konvolúciós réteg működésének egyszerű szemléltetése. Az ábrán egy 12×12 bemenet és egy 3×3 -as kernel látszik. A ponttermék a bemenet és a Kernel közötti elemeként történő szorzás, amelyet aztán összegzésre kerül, és mindig egyetlen értéket ad vissza. Mivel egyetlen értéket eredményez, a műveletet gyakran "skaláris szorzatnak" nevezik. A bemenetnél kisebb kernel használata szándékos. Lehetővé teszi, hogy ugyanaz a kernel (súlykészlet) a bemenet különböző pontjain többször is szorzásra kerüljön a bemeneti

tömbbel. Pontosabban, a kernel szisztematikus alkalmazásra kerül a bemeneti adatok minden egyes átfedő részre, balról jobbra, fentről lefelé. A paraméterek, amik tanulás során kerülnek meghatározásra a kernel elemei. Azaz egy 3×3 -as kernel esetén 9 tanítható paraméter van.

$$G(x,y) = \omega \times F(x) = \sum_{\delta_x=-k_i}^{k_i} \sum_{\delta_y=-k_j}^{k_j} \omega(\delta_{x,}, \delta_y) \times F(x+\delta_x, y+\delta_y) \quad (4)$$

ahol ω a kernel és $-k_i \leq \delta_x \leq k_i, -k_j \leq \delta_y \leq k_j$

A 4. képlet adja a konvolúciós réteg matematikai alapját. Az eredmény mátrix adott eleme egyenlő a bemeneti mátrix adott elemeinek és a hozzájuk tartozó kernel elemek szorzatának összegével. A megbízhatóság növelésének érdekében érdemes a kernel torzítását is figyelembe venni, ami a tanítható paraméterek számát növel 1-el. Ekkor a képlet a következő képpen változik:

$$G(x,y) = \ \omega \times F(x) = \left(\sum_{\delta_x=-k_i}^{k_i} \sum_{\delta_y=-k_j}^{k_j} \omega(\delta_{x,}, \delta_y) \times F(x+\delta_x, y+\delta_y)\right) + \omega_d \quad (5)$$

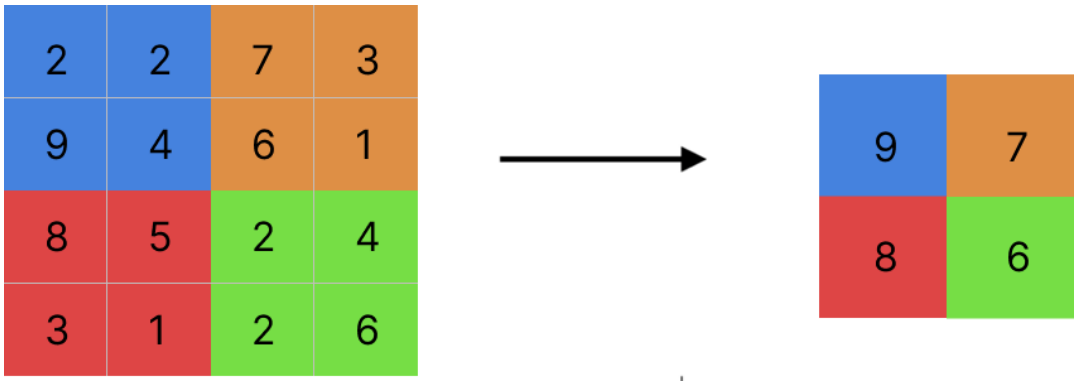
Ahol $\omega_d \in \mathbb{R}$ a Kernel torzítása

A réteg paraméterezése során több kernel is használható. Több kernel esetén a kimenet az egyes kernelek által meghatározott kimeneti értékek summája. Ugyan ez igaz akkor is ha több bemeneti csatorna van (pl: RGB). Ekkor minden csatornán minden kernellel meghatározásra kerül a kimeneti mátrixot és ennek a summája adja a végső eredményt.

3.7 MaxPooling2D

A bemeneten történő mintavételezést végzi el a dimenziók mentén. A mintavételezés a pool_size argumentum által meghatározott méretű. A MaxPooling2D[8] célja egy bemeneti reprezentáció (kép, rejtett réteg kimeneti mátrixa stb.) mintavételezése, csökkentve annak dimenzionalitását, és lehetővé téve a részterületekben található jellemzőkkel kapcsolatos feltételezések megtételét. Javítja a túltanulást és csökkenti a számítási költségeket azáltal, hogy csökkenti a megtanulandó paraméterek számát. A max pooling úgy történik, hogy a kezdeti reprezentáció nem átfedő alrégióira egy max szűrőt alkalmazunk. A max szűrő az adott régió maximumát használja és ezzel hozza létre a dimenziócsökkentett mintát. A pool_size a max pooling esetén meghatároz egy a kernelt, ami hasonlít a 3.6-os pontban leírt

Conv2D-ben is bemutatott kernelhez. A különbség, hogy ebben az esetben a kimenet meghatározható a bemeneti mátrix kernel által lefedett területéből kiválasztott maximum értékkel.



5. ábra Maxpooling2D

3.8 BatchNormalization

A BatchNormalization[9] egy olyan módszer, amelyet a mesterséges neurális hálózatok képzésének gyorsabbá és stabilabbá tételére használnak a rétegek bemeneteinek normalizálásával, a rétegek újbóli központosításával és újra méretezésével. Ezzel a kiegészítő művelettel a hálózat nagyobb tanulási sebességet tud használni. Továbbá úgy tűnik, hogy olyan regularizáló hatása van, hogy javítja a hálózat általánosítási tulajdonságait, és így szükségtelen a dropout alkalmazása a túlillesztés mérséklésére.

Fontos tulajdonsága a rétegnek, hogy tanulás és következtetés közben különböző a működési mechanizmusa.

A tanítás során a réteg normalizálja a kimenetét az aktuális bemenet átlagának és szórásának segítségével. A réteg adott batch-re kiszámolja a középértéket majd pedig a varianciát. Ezeket felhasználva történik a bemenet normalizálása és ezt követően az eltolás és a skálázás. Ez egy minta és egy batch esetén működik. Tetszőleges minta és batch szám esetén az általános képletek a következők

Közép érték

$$E_x = \frac{1}{m}\sum_{i=1}^{j}\mu_B^i \tag{6}$$

Ahol m a minta darabszáma j pedig a batch szám és μ a batch középérték

Variancia

$$Var_x = \left(\frac{m}{m-1}\right)\frac{1}{m}\sum_{i=1}^{j}\sigma_B^{2i} \tag{7}$$

Ahol az σ az adott batch varianciája

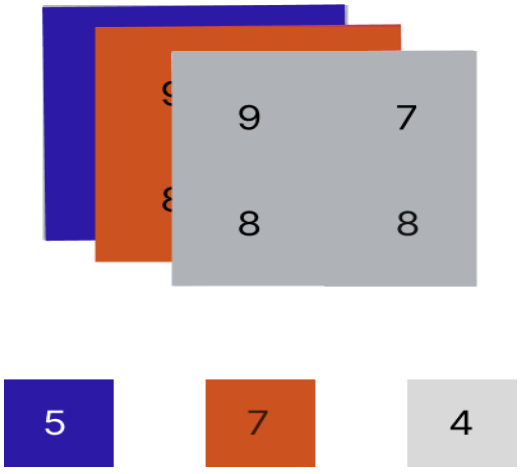
Eltolás és Skálázás

$$y = \frac{\gamma}{\sqrt{Var_x+\epsilon}}x + \left(\beta + \frac{\gamma E_x}{\sqrt{Var_x+\epsilon}}\right) \tag{8}$$

A tanítás során kerül meghatározásra a középérték és a variancia. Ezek a predikciós módban fix értékek.

3.9 GlobalAveragePooling2D

A globalAveragePooling2D[10] réteg segítségével a bemenő adat dimenzióinak méretét lehet csökkenteni a bemeneti adatok magassági és szélességi dimenzióinak átlagának kiszámításával történő mintavételezési eljárással. Működése hasonló a Flatten rétegéhez, azonban jobban megőrzi a pixelek egymáshoz való helyzetének viszonyát és az ebben rejlő információt. Sok esetben használatával csökkenthető a paraméterek száma, ami hatékonyabb és gyorsabban tanuló modellekhez vezethet.



6. ábra globalAveragePooling2D működése

Ahogy az a 6. ábrán is látható, a globalAveragePooling2D() réteg a kép minden csatornáján átlagolt és így képezte az eredményt.

4. Alkalmazott neurális hálózat architektúrák

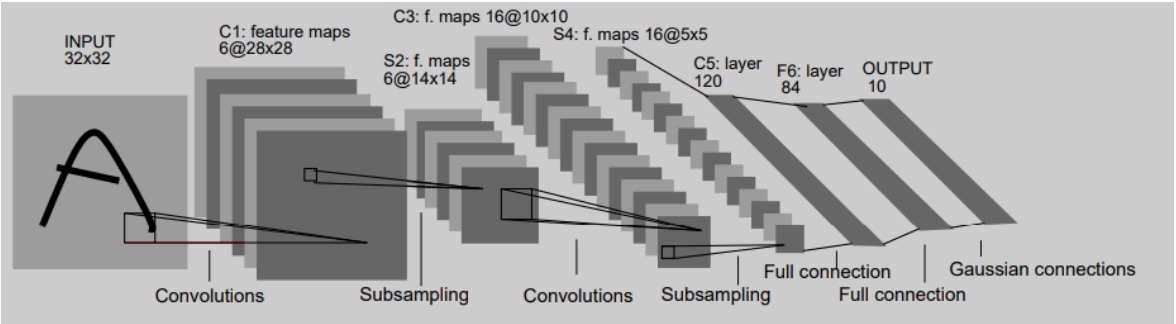
A probléma vizsgálata során 5 különböző modell került kiértékelésre. Dense[3], Lenet[1], Resnet50[11], Resnet50v2[11], VGG19[12]

4.1 DenseNet

Az első modell architektúra, ami tesztelésre került az adathalmazon egy egyszerű Dense Neural Network[3]. Ez a modell egyszerű működése és felépítése miatt ideális volt az alapszint meghatározásához. Ahogy azt a neve is mutatja a Dense egy ”sűrű” hálózat ami ebben az esetben azt jelenti, hogy teljesen összekapcsolt rétegek építik fel. A hálózat Dense rétegekből épül fel ezeknek a működését jól szemlélteti a 3.ábra. Az **Input** réteg a shape paraméterben megadott $28 \times 56 \times 1$ bemenetet várja amit tovább ad egy **Flatten** rétegnek ami a mátrixot vektorrá alakítja. Ezután 2 blokk következik. Mindegyik blokk egy **Dropout,** egy **Dense és** egy **Aktivációs** rétegből áll össze. Az első Dropout mértéke $\mathbf{0,4}$ a 2 blokkban lévő pedig $\mathbf{0,1.}$ A Dense réteg az első blokkban $\mathbf{256}$ **-os** a másodikban pedig $\mathbf{100}$ -as értékkel lett létrehozva. Az első blokkban lévő Aktivációs **ReLU** függvényt míg a második blokkban lévő **softmax**-ot használ.

4.2 LeNet

Második architektúrának a klasszikus LeNet[1] struktúra került kiválasztásra. A LeNet a kezdeti neurális hálózatok felépítését követi. Yan LeCun tervezte meg eredetileg, de azóta számos iteráción ment keresztűl. A LeNet egy előre csatolt konvolúciós hálózat aminek nagy előnye, hogy nem csak az egyes képpontokat, de azok egymáshoz való viszonyát is figyelembe veszi.



7. ábra A LeNet-5 felépítése

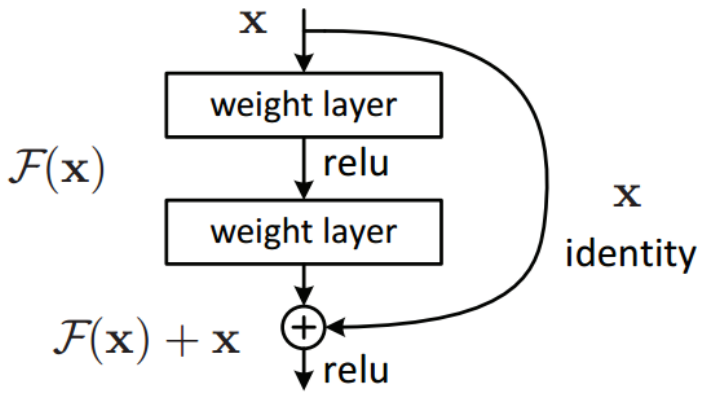
A Konvolúciós hálózatoknál egy $n \times m$-es kernel segítségével történik a mintavételezés. Ezt azt jelenti, hogy a képpontokból nem azonnal 1 dimenziós vektort jön létre, hanem a kernelben lévő képpontok értékei együtt érvényesülnek és így a képpontok egymáshoz való helyzete és az ebben rejlő információ is érvényesül a tanulás során. A kernel mozgatásával kinyert jellemzők a konvolált jellemzők. A konvolúciós hálózat második fele a teljesen összekapcsolt rétegek, ahol maga az osztályozás történik.

A LeNet-5 esetében a felépítés 2 konvolúciós blokkból áll. 1 konvolúciós blokkon belül található a konvolúciós réteg és egy pooling. Ezt követi egy **Flatten** réteg majd pedig 4 darab a 4.1-es pontban a DenseNet felépítésénél leírt teljesen összekapcsolt rétegekből álló blokk. A konvolúciós blokkon belül a konvolúciós réteget a **Conv2D** réteg valósítja meg az első blokkban a paraméterezése filters=6, kernel_size=(5, 5), padding="same", aktivációs="ReLU"), míg a másodikban: (filters=16, kernel_size=(5, 5), aktivációs="ReLU"). A pooling réteget a MaxPooling2D valósítja meg, mindkét blokkban 2×2 -es pool_size paraméterrel. A teljesen összekapcsolt rétegeknél a Dense rétegek 500-tól 100-as értékig folyamatossan csökkenek, a Dropout rétegek értéke minden esetben 0,3.

4.3 ResNet50

Az 4.1-es és a 4.2-es pontban bemutatott architekturát funkcionális modellezési módszerrel kerültek megvalósításra. A ResNet[11] komplexitása miatt a Keras könyvtárból elérhető már előre feltanított modellként került betöltésre, majd pedig a feladatnak megfelelő dense és aktivációs rétegekkel lett kiegészítve és újra tanítva az alapmodell rétegeinek súlyát megtartva.

A ResNet architektúrát a legnépszerűbb konvolúciós neurális hálózati architektúrák között tartják számon. A rétegek számának növelése növeli a teljes hálózat komplexitását és így a tanítás erőforrás igényét is. A különböző mérések azonban azt mutatják, hogy a rétegszám növelésével egy pont után nem minden esetben javítható a rendszer pontossága. A reziduális hálózatok egy új megközelítéssel olyan pontokat adnak a neurális struktúrához, ahol több réteg is átugrásra kerülhet. Ez azért fontos mert a mély neurális hálózatoknál lehetnek olyan rétegek, blokkok, amik rontják a hálózat hatékonyságát. A reziduális hálózatok lehetővé teszik ezeknek a blokkoknak az átugrását.



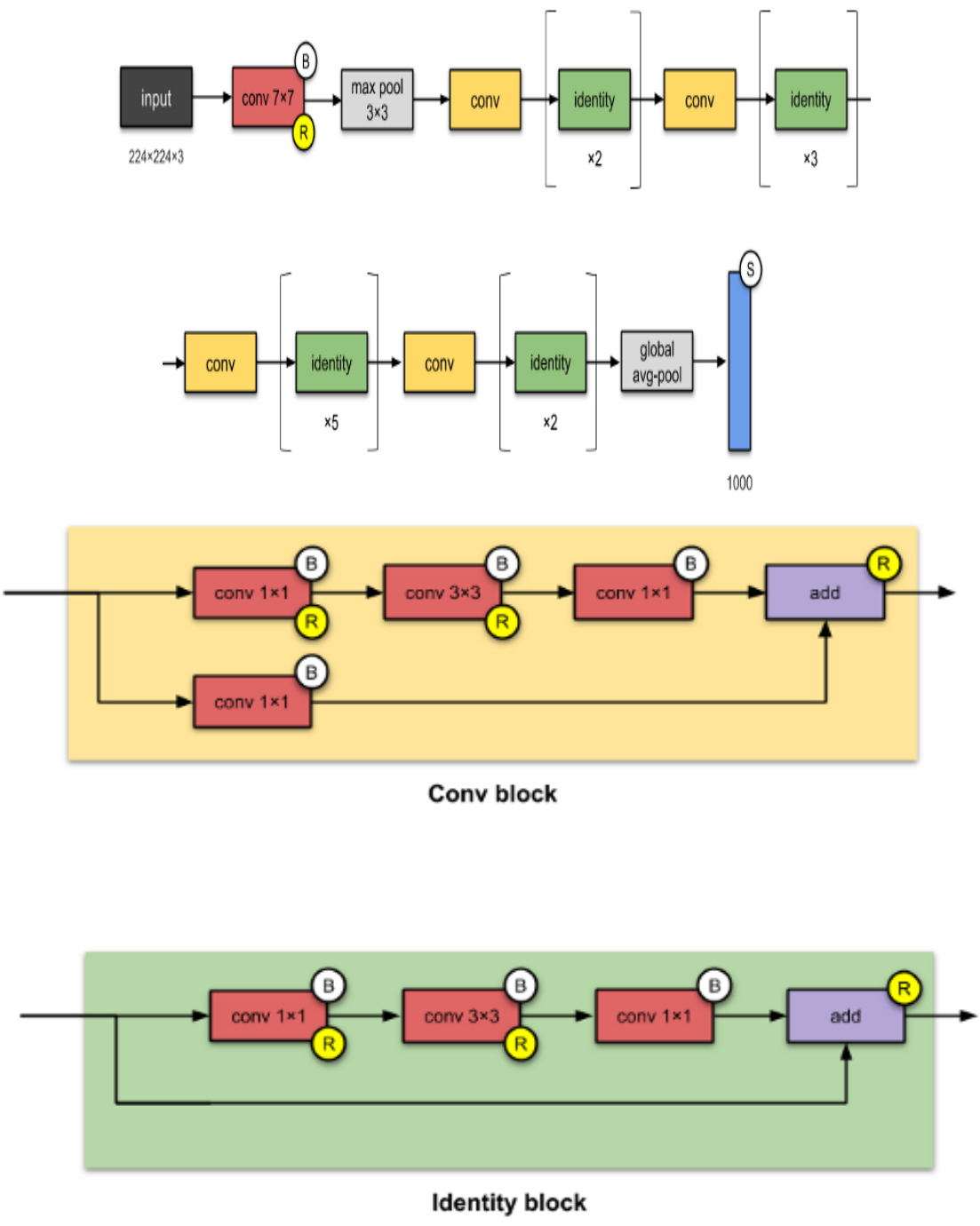
8. ábra reziduális alap blokk

Matematikai értelemben $y = x + F(x)$ ahol y a végső eredmény az $F(x)$ a residuális blokkban lévő rétegek kimenete és x a residuális blokk előtti érték.

1. táblázat reziduális hálózatok különböző rétegszámmal.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

A ResNet50 a reziduális hálózatok elvén épül fel. Az 50 a rétegek számát jelzi. Az 1. táblázat tartalmazza a különböző rétegszámú ResNet hálózatok felépítését.



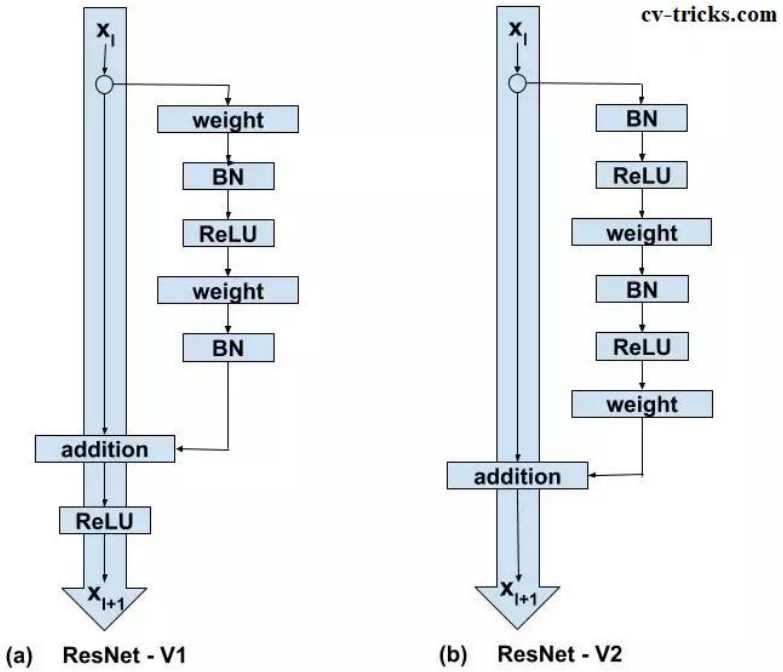
9. ábra Resnet50 szerkezete

Az előre feltanított ResNet50 betöltésre került. Python környezetben a Keras eszközeivel a modell a következőképpen épül fel:

A 15.ábra a ResNet50 felépítésének vizuális szemléltetése. Fontos kiemelni, hogy a sárga és zöld blokkon belüli piros konvolúciós blokkok **Conv2D, BatchNormalization és Aktivációs** rétegekből épülnek fel

4.4 ResNet50v2

A ResNet50 sikere után további fejlesztésekkel jött létre a ResNet50v2. Az alap elv és a fő struktúra ugyan az. A fő különbség a rétegek sorrendjében van. A ResNetv2 esetén a BatchNormalization réteg megelőzi a Conv2D rétegeket míg a hagyományos ResNet50 esetén ez pont fordítva van.



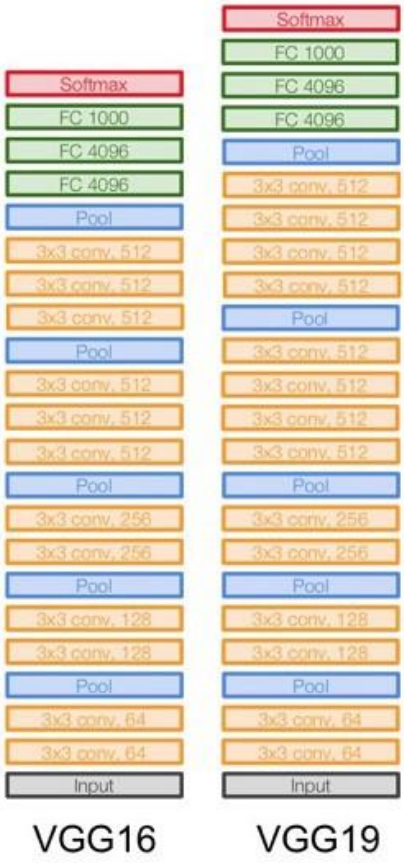
10. ábra A ResNet50 és a ResNet50v2 összehasonlítása

Ahogy az a 19. ábrán is jól látható a Resnet50 esetében post aktivációs míg a ResNet50V2 esetén pre-aktivációs figyelhető meg. Ez azt jelenti, hogy a 14.ábrán is szemléltetett $y = F(x) + x$ azaz az addition művelet után vagy előtt történik meg a az aktiváció. A ResNetv2 esetén az addition művelet után már nincs külön aktiválás ezzel az *x* rögtön bemenete lehet a következő blokknak köztes művelet nélkül.

A réteg sorrend megváltoztatásával, valamint a pre-aktivációs használatával a ResNet50v2 jobb eredményeket tud elérni mint a sima ResNet50.

4.5 VGG19

A VGGNet[12] egy konvolúciós neurális hálózati architektúra, amelyet Karen Simonyan és Andrew Zisserman javasolt az Oxfordi Egyetemen 2014-ben. A VGG-alapú convNet bemenete egy RGB kép. Az előfeldolgozó réteg a 0-255 közötti pixelértékeket tartalmazó RGB-képet veszi, és kivonja belőle a kép átlagos értékét. A bemeneti képek az előfeldolgozás után egy sor konvolúciós rétegen haladnak át. A VGGNet 19 rétegből áll, amelyek 16 konvolúciós rétegből, 3 teljesen összekapcsolt rétegből és 5 pooling rétegből tevődik össze. A VGGNet két, egyenként 4096 csatornával rendelkező, teljesen összefüggő rétegből áll, amelyet egy másik, 1000 csatornával rendelkező, teljesen összefüggő réteg követ 1000 címke prediktálásához. Az utolsó teljesen összekapcsolt réteg a softmax réteget használja az osztályba történő besoroláshoz. A VGG-nek akárcsak a ResNet-nek több verziója is van. A feladat során a VGG19 került implementálásra, azonban létezik VGG16 is aminek a felépítése a rétegek számában tér el. A VGG architektúrának fontos jellemzője a kis filter méret. A nagy méretű filter használat helyett kis méretű filterekből tesz többet egymásra ezzel egy mélyebb struktúrát kialakítva.



11. ábra VGG16 és 19 felépítése

A VGG19 Conv2D, illetve MaxPooling2D rétegből épül fel. A betöltött modell felépítését megnézve látszik, hogy tartalmaz egy globalAveragePooling2D réteget is. Esetén a konvolúciós rétegeket a **Conv2D** valósítja meg az poolingot a **MaxPooling2D és a GlobalAveragePooling2D.** A teljesen összekapcsolt blokkok pedig a DenseNet struktúrájára épülnek.

5. Hálózatok tanítása, eredmények

Az adatok a kiválasztott modellek által elvárt bemeneti formára lettek alakítva, ezt követően pedig megkezdődött a tanítási folyamat.

5.1 Tanítás

A tanítás során az előre elkészített 5 különböző mértékű átfedési szinttel ellátott tanító és teszt halmaz került felhasználásra. Minden esetben 60 000 elem került a tanításba és 10 000 a tesztelésbe. A tanítás lokális számítógépen GPU-n történt. Minden modellt esetén 10 epoch volt megadva a tanulásra. Minden modell feltanítása azonos környezetben azonos feltételek szerint történt. Minden modell mind az 5 adatsorral külön-külön feltanításra került. Így összesen 25 elérhető, kimentet modell keletkezett. A Resnet50, illetve a VGG19 megkívánta a 3 csatornás (RGB) bemenetet. A szürkeárnyalatos mintáink csak egy csatornával rendelkeznek, ez azonban ugyan annak a rétegnek az egymásra illesztésével áthidalható. Azoknál a modelleknél, ahol erre szükség volt külön meghívásra került egy a bemenetet átalakító függvény. Az adatokon érdemi változás nem történt.

Tanításnál használt metrika

A tanulás során a modell compile függvénye egységesen került paraméterezésre. A tanítás közben nyomon követett metrika az Accuracy volt. Ez a validációs adatok segítségével megmutatja, hogy adott állapotban hányszor találja el a modell a valós címkét.

$$ACC = \frac{y_g}{n} \tag{9}$$

Ahol n a prediktált osztályok darabszáma és y_g a helyesen prediktált osztályok száma.

Tanításnál használt Loss függvény

A modell által visszaadott eredményből minden esetben megoldást kell generálni. Erre szolgál a Loss függvény. Feladattól függően változik, hogy milyen függvényt érdemes használni. A dolgozat során körbejárt probléma egy osztályozási feladat, ahol több osztály közül egyet kell kiválasztani. Ezekre a problémákra a Categorical crossentropy azaz Kategorikus kereszt entrópia függvényt érdemes használni.

$$Loss = -\sum_{i=1}^{n} y_i \times \widehat{y_i} \tag{10}$$

Ahol $\widehat{y_i}$ az i. Skalár érték a modell kimenetében, y_i pedig az elvárt kimenet n pedig a kimenet méretét jelöli.

Tanításnál használt Optimizer

A megfelelő optimalizáló algoritmus kiválasztása különösen fontos. Sok esetben ez jelenti a különbséget a megbízható és a rosszul teljesítő modell között. A feladat során a népszerű Adam algoritmus került kiválasztásra a modellekhez. Az Adam (adaptive moment estimation) kiterjesztés a sztochasztikus gradiens csökkentő algoritmusnak. Az algoritmus jellemzője, hogy a tanulási ráta a tanítás alatt stabil, nem változik. Ötvözi más algoritmusok előnyeit, mint a AdaGrad illetve RMSProp. Az optimalizáló Adam algoritmus az alapértelmezett paraméterekkel került beépítésre. Mivel a megvalósítás Keras használatával történt így az alap paraméterek a következők:

$$lr = 0{,}001, \beta 1 = 0{,}9, \beta 2 = 0{,}999, \varepsilon = 1e - 08 \tag{11}$$

Ahol lr a tanulási ráta (LearningRate) $\beta 1$ Az exponenciális lecsengési arány az első pillanatra vonatkozó becsléseknél, $\beta 2$ A második pillanatra vonatkozó becslések exponenciális lecsengési rátája. Ezt az értéket 1,0 közelébe kell állítani a ritkás gradiensű problémáknál, Az ε egy nagyon kis szám, hogy megakadályozza a nullával való osztást a végrehajtásban.

5.2 Tesztelés.

A tanítási fázis után a modelleke kiértékelésre kerültek a teszt halmazon. A modellek kiértékeléséhez a teszthalmazok közül kiválasztásra került a megfelelő átfedési szinttel generált adathalmaz. Azaz a modellek elsősorban azokon az adatokon dolgoztak, amiknek az átfedési szintjük egyezik a tanuló halmaz átfedési értékeivel. A kiértékeléshez több metrika is beépítésre került.

MAE (Mean Absolute Error)

Az összes adatpont abszolút hiba eltérésének az átlag. A függvény a modell által prediktált és a valós címkék különbségének az abszolút eltérését vizsgálja.

$$MAE = \frac{\sum_{i=1}^{n} |(y_{(i)t} - y_{(i)p})|}{n} \tag{12}$$

Ahol n a prediktált elemek száma

MSE (Mean Squared Error)

Az összes adatpont négyzetes hiba eltérésének az átlag. A függvény a modell által prediktált és a valós címkék különbségének a négyzetes eltérését vizsgálja.

$$MSE = \frac{\sum_{i=1}^{n}(y_{(i)t} - y_{(i)p})^2}{n} \quad (13)$$

Ahol n a prediktált elemek száma

ACC

Az 5.1-es pontban leírt metrika. Visszaadja a modell pontosságát az alapján hány címkét prediktált helyesen és, hogy hány címke volt összesen.

A három metrika használatával az alábbi táblázatban látható eredményeket érték el az egyes modellek.

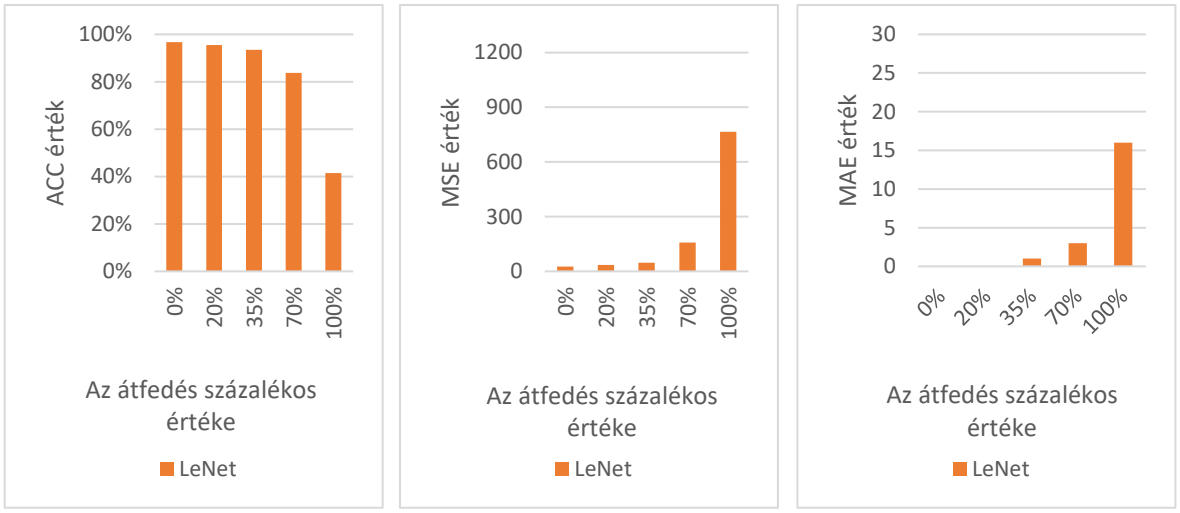
2. táblázat A modelleken mért eredmények

Model típus	Átfedés mértéke	MAE	MSE	ACC
DenseNet	0	1	76	90,90%
DenseNet	3	2	91	90,22%
DenseNet	5	2	128	87,36%
DenseNet	10	5	249	74,76%
DenseNet	14	19	906	36,10%
LeNet	0	0	27	96,73%
LeNet	3	0	35	95,54%
LeNet	5	1	48	93,45%
LeNet	10	3	158	83,70%
LeNet	14	16	765	41,47%
ResNet50	0	12	552	40,84%
ResNet50	3	12	552	41,53%
ResNet50	5	14	602	37,22%
ResNet50	10	17	820	30,59%
ResNet50	14	25	1227	14,99%
ResNet50v2	0	10	446	51,43%
ResNet50v2	3	9	435	51,90%
ResNet50v2	5	10	460	49,91%
ResNet50v2	10	13	635	40,85%
ResNet50v2	14	22	1108	21,38%
VGG19	0	4	219	76,16%
VGG19	3	5	240	72,93%
VGG19	5	6	264	69,80%
VGG19	10	9	401	59,60%
VGG19	14	19	864	31,30%

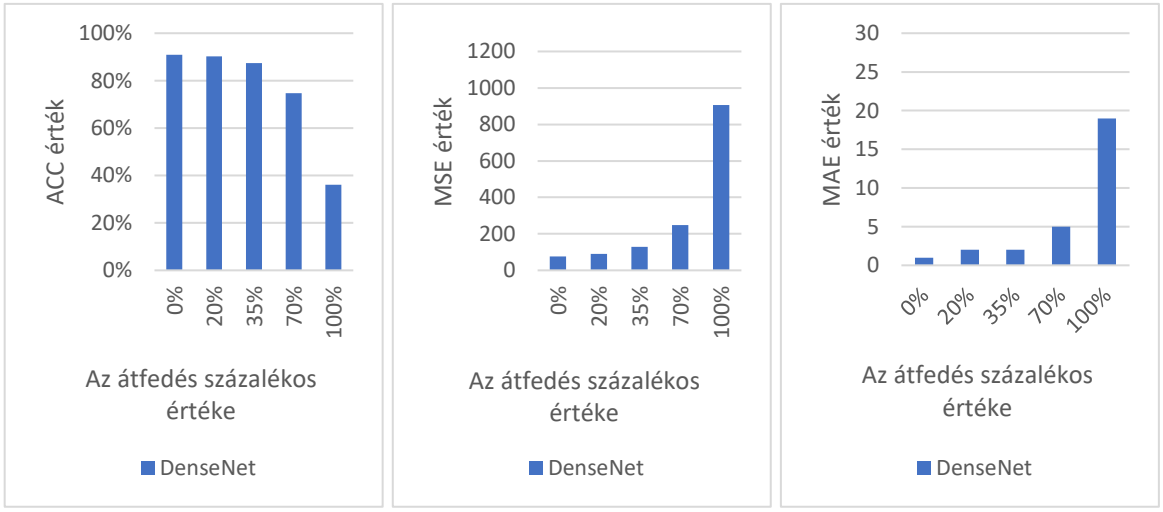
A 2. táblázat alapján elmondható, hogy minden modellre, igaz az, hogy az átfedés mértékének növekedésével folyamatosan romlik a teljesítmény. Minden modell esetén kiugróan nagy esés van a pontosságban a teljes átfedés esetén. A modellek közül a LeNet teljesített a legjobban, és egyértelműen látszik, hogy a problémát nem lehet összetettebb hálózatokkal jobban megoldani hiszen a ResNet és a VGG is alul teljesített.

Feltételezhető, hogy az egyes modellek további hipermaraméterezési finomhangolás után még jobb teljesítményre képesek azonban a teljes átfedés esetén feltételezhető, hogy

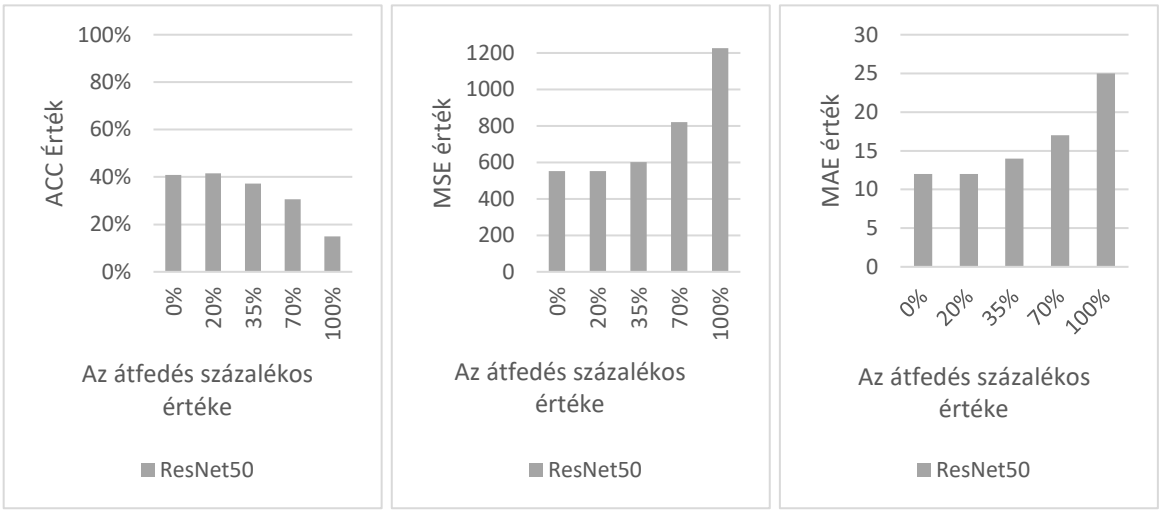
gyakorlatban történő hasznosításhoz még ezután sem érik el a kellő hatékonyságot. A 12. 13. 14. 15. és 16. ábrákon megjelenítésre kerültek az egyes modellek értékei a jobb érthetőség végett.



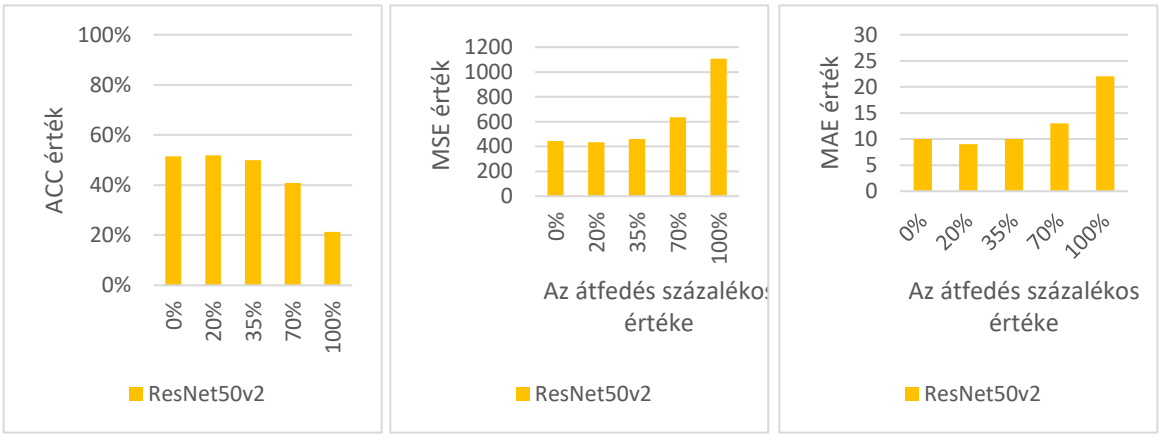
12. ábra A LeNet értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE



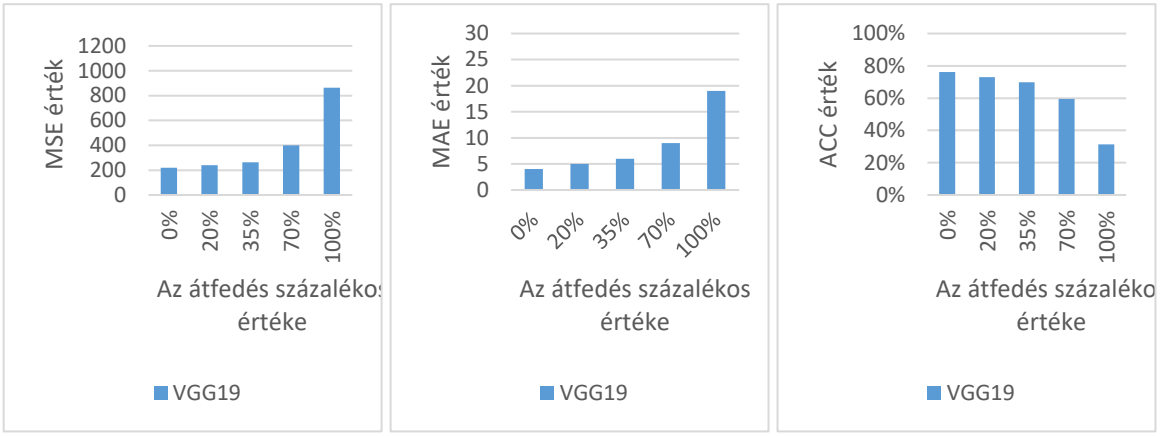
13. ábra A DenseNet értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE



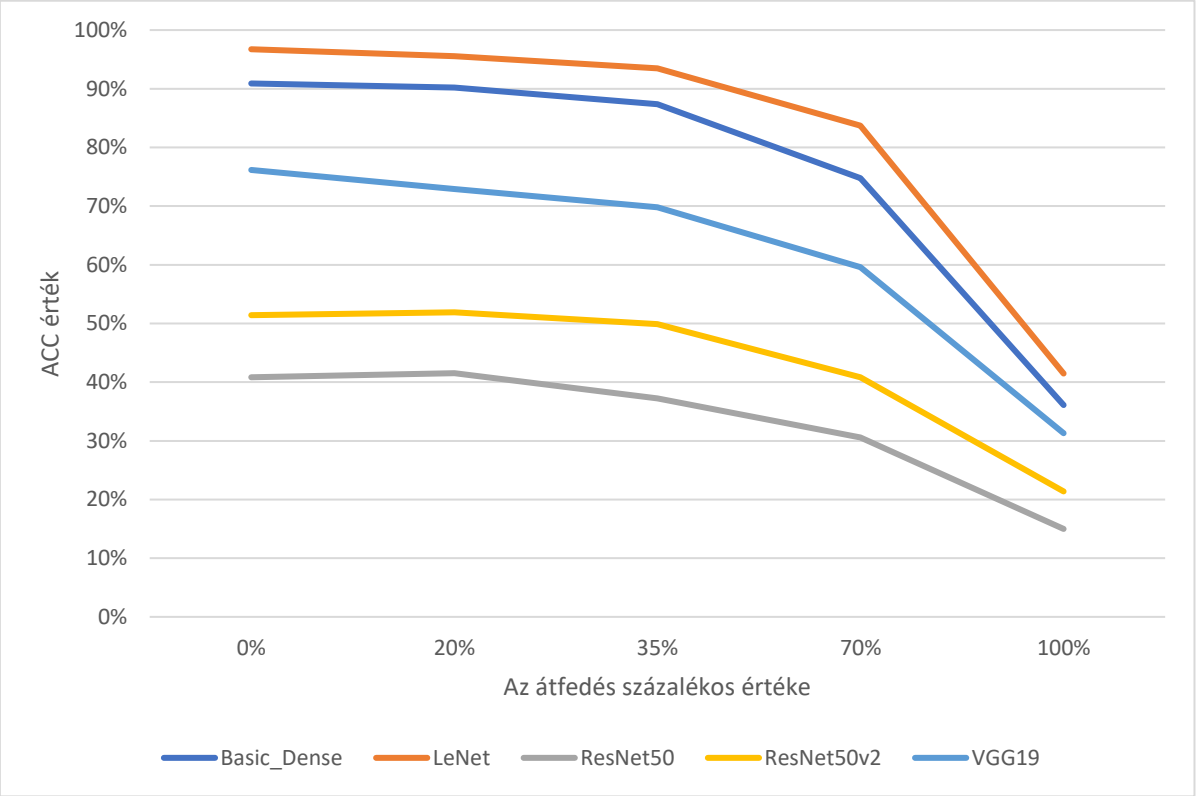
14. ábra A ResNet50 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE



15. ábra A ResNet50v2 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE

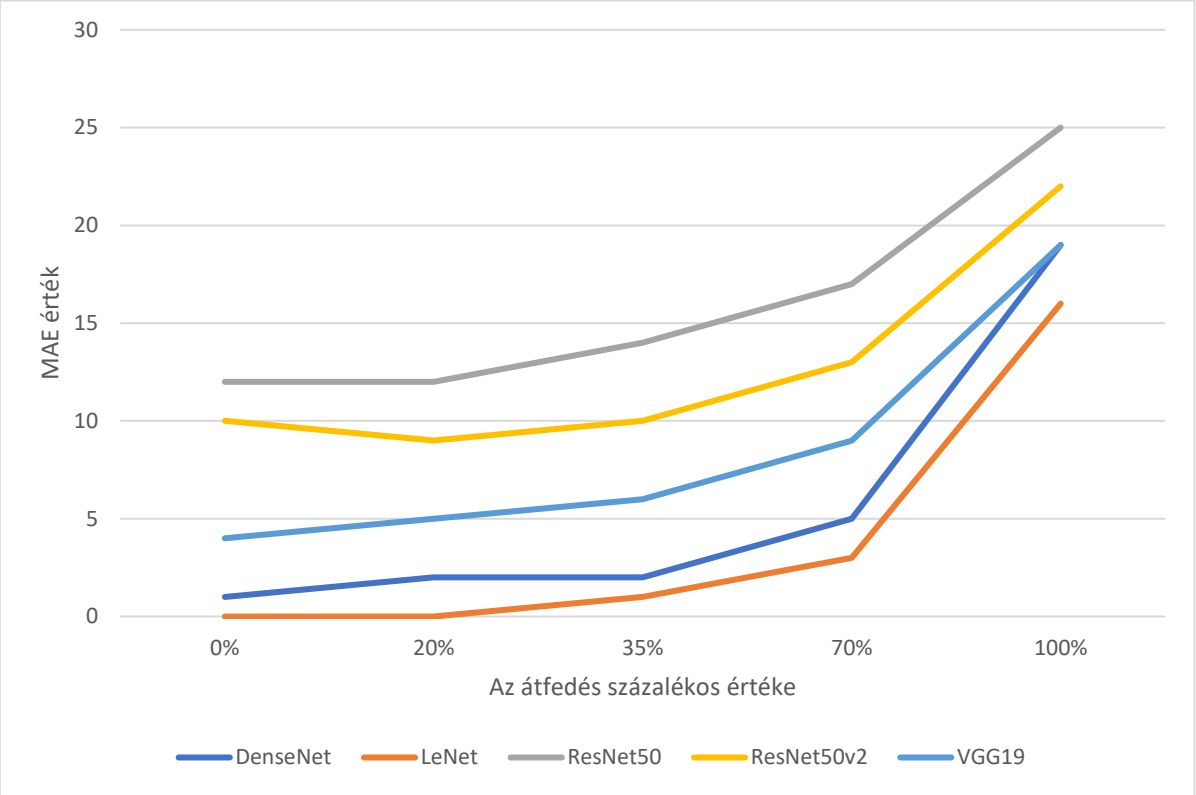


16. ábra A VGG19 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE



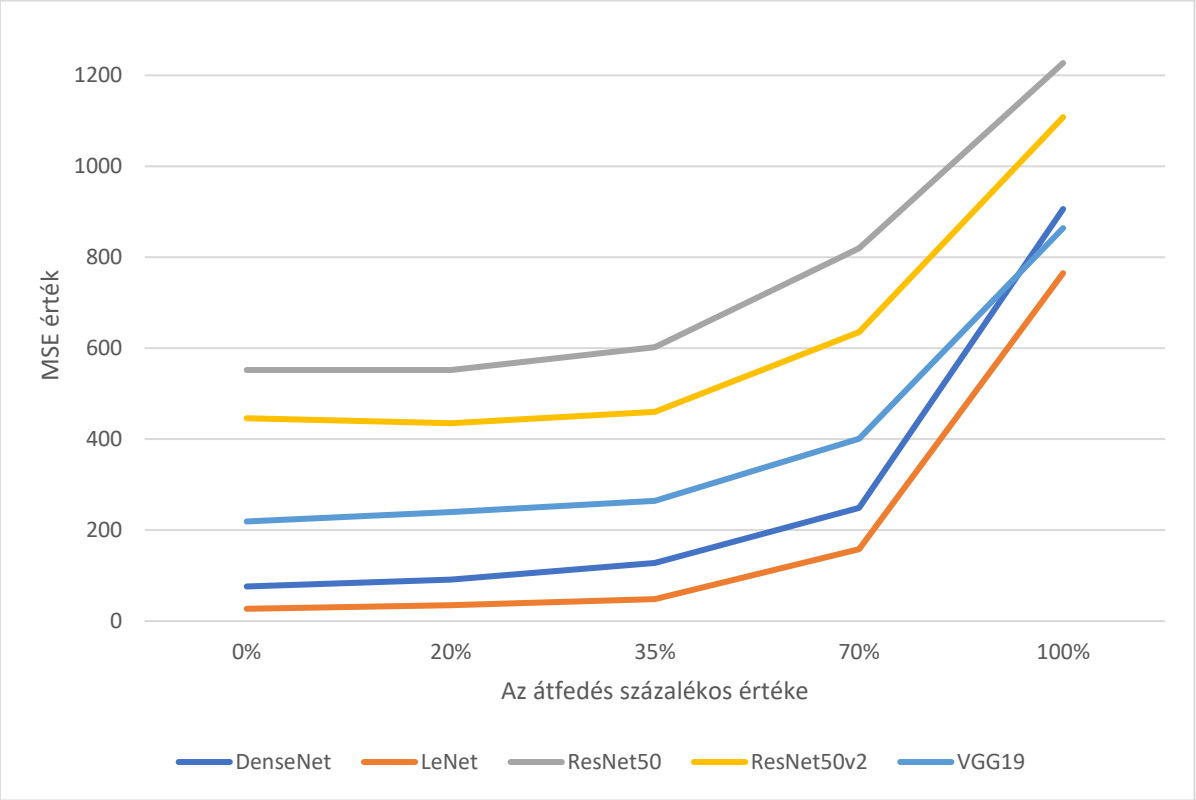
17. ábra Modellek ACC értékek egymáshoz való viszonya

A modellek egyes értékeit a kezdeti feltételezés vizsgálatához célszerű együtt megvizsgálni. A 17.ábra vizualizálja a modellek ACC értékének változását az átfedés mértékének növekedése szerint. Ami egyértelműen látszik az ábrán az 2 töréspont a meredekségben. Az 35%-os átfedésnél, illetve a 70%-os átfedésnél történik nagyobb meredekségű zuhanás a pontosság tekintetében. A megfelelő modellek segítségével 35%-os átfedési szint alatt a hagyományos megoldások is kellő pontossággal működnek.



18. ábra a modellek által elért MAE értékek egymáshoz való viszonya

A 18.ábra vizualizálja a modellek MAE értékének változását az átfedés mértékének növekedése szerint. Itt is a 17-es ábrához hasonlóan 2 töréspont figyelhető meg a meredekségben. Az 35%-os átfedésnél, illetve a 70%-os átfedésnél ahogy azt az ACC értékeknél is történik nagyobb meredekségű emelkedés a hibaérték tekintetében. A megfelelő modellek segítségével 35%-os átfedési szint alatt a hagyományos megoldások is kellő képpen alacsony hiba értékkel működnek.



19. ábra a modellek által elért MSE értékek egymáshoz való viszonya

A 19.ábra vizualizálja a modellek MSE értékének változását az átfedés mértékének növekedése szerint. Itt is a 17-es és 18-as ábrához hasonlóan 2 töréspont figyelhető meg a meredekségben. Az 35%-os átfedésnél, illetve a 70%-os átfedésnél ahogy azt az ACC és MAE értékeknél is történik nagyobb meredekségű emelkedés a hibaérték tekintetében. A megfelelő modellek segítségével 35%-os átfedési szint alatt a hagyományos megoldások is kellő képpen alacsony hiba értékkel működnek.

A 17. 18. 19.-es ábrák egymást alátámasztják és egyértelműen megmutatják, hogy a 35%-os átfedési szint felett jelentősen csökken a hagyományos karakterfelismerési megoldások hatákényossága.

6. Konklúzió

A probléma megismerése során több, a hagyományos kézzel írt karaktereken nagy hatékonyságot elérő neurális hálózati architektúrán is mérések történtek egymást átfedő kézzel írott karakterek felismerésével kapcsolatban. A vizsgálat célja az architektúrák magasabb szintű megismerése volt, valamint annak a hipotézisnek a vizsgálata miszerint a probléma hasonlósága miatt nincs feltétlen szükség új megoldásokra a részben vagy teljesen átfedő karakterek felismeréséhez.

A vizsgálatok során kapott eredmények egyértelműen megmutatják, hogy 35%-nál alacsonyabb átfedési szint esetén a hagyományos karakterfelismeréshez használt modellek kellő hatékonysággal tudnak működni. Az egymástól erősen különböző modell architektúrák pontosságának negatívváltozásának pontja erősen egybe vág, ahogyan azt a 17,18 és 19-es ábrán is láthatjuk. Ez az egybevágás egyértelműen megmutatja, hogy mi az a pont, ahol azt mondhatjuk, hogy a hagyományos kézzel írt karakterfelismerési problémához képest egy valóban új probléma áll fenn. A 35% feletti átfedés esetén látható csökkenés a pontosságban és növekedés a hibában jelzi, hogy egy új probléma körbe lépet át a feladat. Az ilyen szintű átfedés esetén már érdemes más megoldásokkal is próbálkozni.

Kérdésként felmerül, hogy a gyakorlatban mennyire gyakori az ilyen szintű átfedés és, hogy mik azok a területek ahol ez a probléma előkerül. A 35%-os átfedési szint feletti felismeréshez érdemes a területeket is megvizsgálnunk a megfelelő megoldás kiválasztásához, hogy az adott terület más jellemzőit is figyelembe tudjuk venni.

A dolgozat elérte célját és egy olyan alapvonalat állított fel amire későbbi munkákat érdemes alapozni. Érdemes megvizsgálni az egyes modellek hiperparaméterezéseinek finomhangolását és annak hatásait a pontosságra.

7. Irodalom Jegyzék

[1] Y. LeCun, L. B. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278-2324. Letöltés dátuma: 2022. 10 15, forrás: http://yann.lecun.com/exdb/mnist/

[2] Nitish Srivastava, G. H. (2014). Dropout: A Simple Way to Prevent Neural Networks from. *Journal of Machine Learning Research* , 1929-1958.

[3] Huang, G., Liu, Z., Maaten, L. V., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (old.: 2261-2269).

[4] Shiv Ram Dubey, S. K. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 92-108.

[5] Agarap, A. F. (2018). *Deep learning using rectified linear units (relu).* . Forrás: Arxiv.org: https://arxiv.org/abs/1803.08375

[6] B. Chen, W. D. (2017). Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (old.: 4021-4030).

[7] O'Shea, K. &. (2015). *An Introduction to Convolutional Neural Networks.* Forrás: ArXiv e-prints: https://arxiv.org/abs/1511.08458

[8] Nagi, J. &. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. *IEEE International Conference on Signal and Image Processing Applications, ICSIPA*, (old.: 342-347).

[9] Sergey, I., & Christian., S. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Forrás: arxiv.org: https://arxiv.org/pdf/1502.03167.pdf

[10] Min Lin, Q. C. (2013). *Network In Network.* Forrás: arxiv.org: https://arxiv.org/abs/1312.4400

[11] K. He, X. Z. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, (old.: 770-778).

[12] Karen Simonyan, A. Z. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition.* Forrás: arxiv.org: https://arxiv.org/abs/1409.1556

[13] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[14] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

8. Ábrajegyzék

1. ábra Minták az MNIST adathalmazból. 7
2. ábra átfedés generálása után kapott minta adatok. Az átfedés mértékét az adatok felett feltüntetett szám jelzi. 8
3. ábra A Dense réteg működése, X a bemenet, N a neuron, s a súly 10
4. ábra Konvolúciós réteg vizualizálása 12
5. ábra Maxpooling2D 14
6. ábra globalAveragePooling2D működése 15
7. ábra A LeNet-5 felépítése 16
8. ábra reziduális alap blokk 18
9. ábra Resnet50 szerkezete 19
10. ábra A ResNet50 és a ResNet50v2 összehasonlítása 20
11. ábra VGG16 és 19 felépítése 22
12. ábra A LeNet értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE 27
13. ábra A DenseNet értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE 27
14. ábra A ResNet50 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE 28
15. ábra A ResNet50v2 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE 28
16. ábra A VGG19 értékeinek vizualizációja. ballról jobbra ACC, MSE, MAE 29
17. ábra Modellek ACC értékek egymáshoz való viszonya 30
18. ábra a modellek által elért MAE értékek egymáshoz való viszonya 31
19. ábra a modellek által elért MSE értékek egymáshoz való viszonya 32

9. Táblázatok

1. táblázat reziduális hálózatok különböző rétegszámmal..... 18
2. táblázat A modelleken mért eredmények..... 26