

TDK-dolgozat

Fehér Gergő

Fehér Gergő

MESTERSÉGES NEURÁLIS HÁLÓZAT ALAPÚ ALGORITMUS HASZNÁLATA A BINÁRIS TOMOGRÁFIÁBAN

Témavezető: Dr. Szűcs Judit, adjunktus

Műszaki Informatikai Kar, Informatikai Rendszerek és
Alkalmazásai Tanszék

Pannon Egyetem

Intézményi Tudományos Diákköri Konferencia

2022, november



Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani témavezetőmnek, Dr. Szűcs Juditnak a TDK dolgozat megvalósításához nyújtott szakmai és technikai segítségéért, hasznos észrevételeiért, építő jellegű kritikáiért és az emberi támogatásáért. A dolgozat nem jöhetett volna létre a konzultációk során nyújtott odafigyelés, segítség és iránymutatás nélkül.

Szintén köszönöm Pap Gergelynek a hasznos észrevételeit, továbbá szakmai segítségét.

Tartalomjegyzék

Ábrajegyzék.....	5
1. Bevezetés	6
2. Szakirodalmi áttekintés, felhasznált ismeretek.....	8
2.1 Bináris Tomográfia.....	8
2.1.1 Ryser algoritmus.....	9
2.1.2 Shift Local Binary Patterns	10
2.1.3. Relatív Átlagos Hiba	11
2.1.4. Pixelhiba	12
2.2. Morfológia.....	13
2.3 Neurális hálózat	16
2.3.1. Aktivációs függvények.....	17
2.3.2. Optimalizációs algoritmusok.....	19
2.3.3. Regularizációs technikák.....	20
3. Módszertan	21
4. Elért eredmények ismertetése.....	24
4.1. Eredmények vetületi adatok felhasználásával	25
4.2 Eredmények vetületi és SLBP adatok felhasználásával	26
4.3 Eredmények a Ryser algoritmussal előállított kép kiegészítésével	27
5. Összegzés	30
6. Irodalomjegyzék	31

Ábrajegyzék

1. ábra Példa a horizontális és vertikális vetületekre	8
2. ábra Példa a Ryser algoritmus lépéseire	9
3. ábra Példa kapcsoló-komponensekre	10
4. ábra Példa RME értékekre 1.	11
5. ábra Példa RME értékekre 2.	12
6. ábra Példa pixelhibára 1.....	13
7. ábra Példa pixelhibára 2.....	13
8. ábra Példa dilatacióra	14
9. ábra Példa erózióra.....	14
10. ábra Példa nyitásra	14
11. ábra Példa zárásra	15
12. ábra Példák morfológiai szűrésre	15
13. ábra Egy neuron vázlatos képe	16
14. ábra Neurális hálózat matematikai modellje.....	17
15. ábra Lineáris aktivációs függvény	18
16. ábra ReLu függvény.....	18
17. ábra Sigmoid aktivációs függvény	19
18. ábra Képosztályok alapképei	24
19. ábra Példa eredmény a Szoftveres fantom képosztályra (18. ábra (b)) futtatott tesztek- ből	26
20. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztek- ből	26
21. ábra Példa eredmény a Bentheimer hálózat képosztályra (18. ábra (c)) futtatott tesztek- ből	27
22. ábra Példa eredmény a Szoftveres fantom képosztályra (18. ábra (b)) futtatott tesztek- ből	27
23. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztek- ből	27
24. ábra Példa eredmény a Bentheimer hálózat képosztályra (18. ábra (c)) futtatott tesztek- ből	28
25. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztek- ből	28
26. ábra Példa eredmény a Bentheimer hálózat képosztályra (18. ábra (c)) futtatott tesztek- ből	29

1. Bevezetés

A számítógépes képfeldolgozás mára a mindennapi életünk jelentős részévé vált, bár ezt sok esetben még csak észre sem vesszük. Gondoljunk csak a modern autóknál a beépített parkolást segítő, kamerával ellátott rendszerre, mely akár hangjelzéssel figyelmeztet a közeli objektumokra [1], [2]; vagy például az információcserét megkönnyítő QR kódok generálására és olvasására [3], [4], [5]; vagy például beltéri kamerán történő gyanús személyek követésére [6]; de számtalan egyéb példát sorolhatnánk ide.

A számítógépes tomográfia (Computed Tomography, CT) területén sincs ez másképp. Az 1970-es években Allan M. Cormack és Godfrey N. Hounsfield fejlesztették ki, amiért 1979-ben orvosi Nobel-díjat is kaptak. A feltalálás nagyszerűségét az is mutatja, hogy az odáig ismert, hagyományos röntgen-átvilágítási technikát oly módon módosították, hogy vékony, közel síkszerű sugár-nyalábokat használtak. Egy detektor segítségével érzékelték, hogy mekkora rész nyelődött el, és így következtetni tudtak az anyag sűrűségére melyen áthaladtak a sugarak. Amennyiben egy-egy síkot ilyen sugárnyalábokkal több irányból is megvizsgáltak, úgy elég információhoz jutottak az eredeti 128×128 pixeles képrekonstrukció megvalósításához.

A tomográfia alapfeladata egy objektum belső szerkezetének rekonstruálása annak szeleteiből. A tomográfiában szeletnek nevezzük azokat a kétdimenziós keresztmetszeti képeket, melyeket háromdimenziós objektumok kétdimenziós síkkal történő metszésének eredményeként kapunk. Természetesen, ebben az értelemben a “metszésen” nem a hagyományos értelemben vett szeletelést értjük, hiszen ebben az esetben nemroncsoló tesztelésről beszélünk, tehát a szeletekhez nem férünk közvetlenül hozzá, csak valamely másodlagos információ segítségével (általában ez az anyagsűrűség összege bizonyos irányok mentén).

A rekonstrukció végrehajtására számtalan algoritmus létezik (pl. visszavetítés, szűrt visszavetítés, SIRT, SART [7]; ART [8], stb.), ám ezek jellemzően többszáz vagy akár több ezer vetülettel működnek jól. A gyakorlatban ez sokszor nem elérhető. Diszkrét képek esetén azonban a szükséges vetületek száma jelentősen csökkenthető, hiszen a megoldás keresési tere szűkíthető azon előzetes információ alapján, mely szerint a diszkrét képek csak bizonyos értékeket vehetnek fel. Ezt a tulajdonságot használja ki a Diszkrét Tomográfia [9] [10] [11] is. Amennyiben még azt is tudjuk, hogy a rekonstruálandó objektum egyetlen anyagból áll, azaz a hozzá tartozó kép csak 0 és 1 értékeket tartalmazhat, akkor tovább szűkíthető a keresési tér és

ezt a tulajdonságot fel lehet használni a későbbi metódusokban is. Ezzel elérkeztünk a CT egy nagyon speciális esetéhez, a Bináris Tomográfiához (BT), mellyel a kutatómunkámban is foglalkoztam.

Természetesen minél több vetület áll rendelkezésre, annál könnyebb és pontosabb a rekonstrukció, de a vizsgálat annál magasabb költségekkel jár, illetve a vizsgált anyag roncsolódása is előfordulhat nagymértékű sugárterhelés esetén [12]. Ez a későbbiekben problémát jelenthet.

A dolgozatomban azt a célt tűztem ki magam elé, hogy a rekonstrukció minőségét extrém kevés vetület esetén mesterséges neurális hálózattal javítsam.

2. Szakirodalmi áttekintés, felhasznált ismeretek

Ebben a fejezetben a dolgozatom megértéséhez szükséges alapvető definíciókat ismertettem.

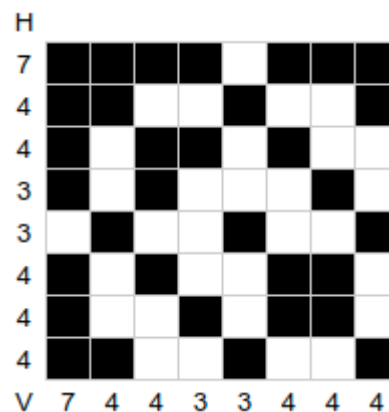
2.1 Bináris Tomográfia

A Bináris Tomográfia (BT) során a rekonstruálandó kép csak 0 és 1 értékeket tartalmazhat, hiszen az eredeti objektum csupán egyetlen anyagból áll. A bináris rekonstrukciós probléma visszavezethető az alábbi lineáris egyenletrendszerre

$$\mathbf{Ax} = \mathbf{b},$$

ahol az $\mathbf{A} \in \mathbb{R}^{k \times mn}$ mátrix leírja a k vetítősugár és a pixelek közötti kapcsolatot, a $\mathbf{b} \in \mathbb{R}^k$ a mért vetületi értékeket tartalmazó vektor, $\mathbf{x} \in \{0,1\}^{mn}$ az $m \times n$ méretű ismeretlen kép (sorvektorként megadva).

Dolgozatomban azt a szélsőséges esetet vizsgálom, melynek során összesen két vetület, a horizontális és vertikális áll rendelkezésünkre. A bináris tomográfiában ezek kiszámítása egyszerű, a horizontális vetületek a sorösszegeket, míg a vertikális vetületek az oszlopösszegeket jelentik. Az 1. ábra alapján egy példát láthatunk, H-val jelölve a horizontális, V-val pedig a vertikális vetületeket.

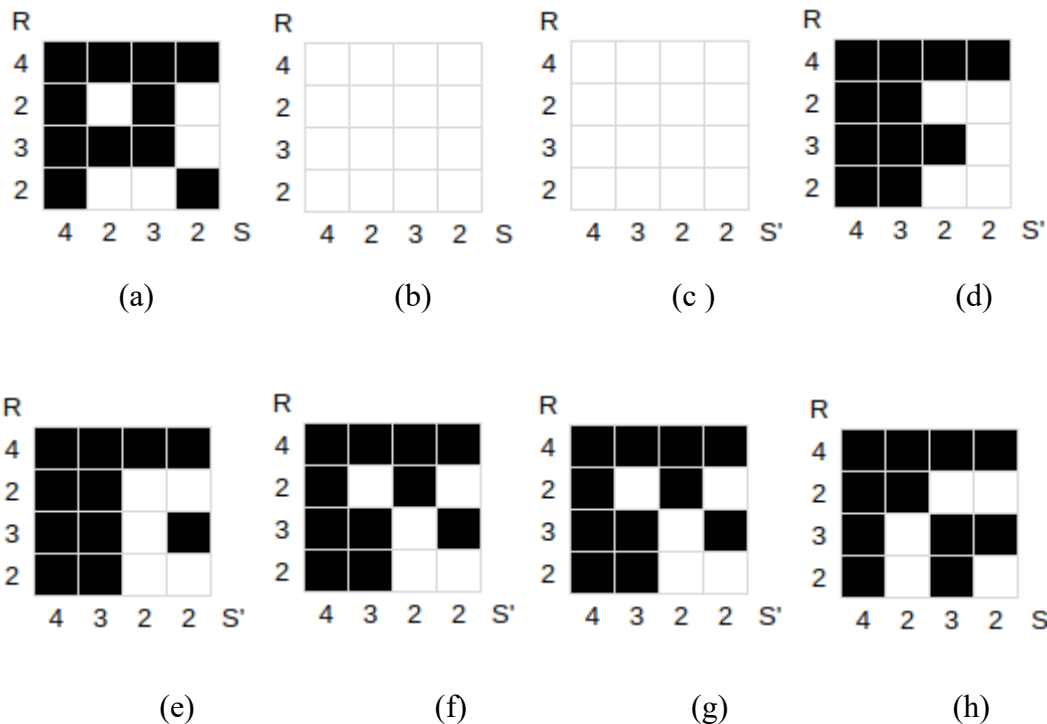


1. ábra Példa a horizontális és vertikális vetületekre

2.1.1 Ryser algoritmus

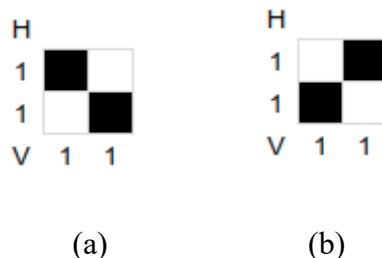
Az egyik legrégebbi algoritmus, mely megoldást ad a bináris tomográfia problémára, a Ryser algoritmus [13]. A módszer képes polinom időben rekonstruálni egy mátrixot a sor- és oszlopösszegek alapján, a legtöbb esetben az eredmény azonban nem egyértelmű a kapcsoló komponensek miatt. Dolgozatomban a Ryser algoritmust referenciaként használtam.

Az algoritmus pontos működését az alábbiakban, a leírás mellett képekkel is szemléltetem és feltárom a kapcsoló komponensek problémáját is. Legyen a kiindulási képünk a 2. ábra (a), amelyen feltüntettem a vetületeket is (R jelöli a sor- és S az oszlopösszegeket). Ezt követően generáltam egy üres képet, amelyiken csak a vetületi értékeket tüntettem fel (2. ábra (b)). Az oszlopösszegeknek vettem egy permutációval való nemnövekvő átrendezését (2. ábra (c)). A sorösszegeknek megfelelően minden sort balról jobbra feltöltöttem a megfelelő számú pixellel kép (2. ábra (d)). Ezt követően az oszlopokon jobbról balra haladva minden oszlopba az ő vetületének megfelelő számú pixel eléréséig, az oszlopon fentről lefelé haladva toltam el a tőle balra elhelyezkedő legjobboldalibb oszlopból pixelt amennyiben volt (2. ábra (e), (f) és (g)). Utolsó lépésként pedig az eredeti permutáció inverzével visszarendeztem az eredeti sorrendbe a kép oszlopait. A kapott eredmény (2. ábra (h)) csak hasonlít, de nem egyezik meg az eredeti képpel (2. ábra (a)) ábra.



2. ábra Példa a Ryser algoritmus lépéseire

A problémát a kapcsoló komponensek jelentik, amire egy szemléletes példa az alábbi két kép (3. ábra), melyek mindegyike kielégíti ugyanazon sor- és oszlopösszegeket úgy, hogy vizuálisan nem egyeznek meg.



3. ábra Példa kapcsoló-komponensekre

A kapcsoló komponenseket úgy is megfogalmazhatjuk, mint egy bináris részmátrix, melynek szemközti sarkaiban 1, míg a másik szemközti sarkaiban 0 értékek szerepelnek. Amennyiben végrehajtunk egy kapcsolást (azaz felcseréljük a 0 és 1 értékeket), abban az esetben a sor-és oszlopösszegek nem változnak, ám az eredményül kapott kép igen. Minél nagyobb egy-egy bináris kép, annál több kapcsoló komponens fedezhetünk fel rajta, melynek következtében számtalan vizuálisan eltérő, ám azonos sor-és oszlopösszegekkel rendelkező megoldása lehet egy adott feladatnak. Ez okozza mind a Ryser, mind pedig más algoritmusok esetén, hogy a megoldás nem egyértelmű.

2.1.2 Shift Local Binary Patterns

Az általam vizsgált esetben csupán két vetület áll rendelkezésre, és ilyenkor a rekonstrukciós feladat rendkívül alulhatározott. A keresési tér csökkentése érdekében azonban különféle előzetes információ építhető be az algoritmusba azon kívül, hogy csupán fekete és fehér pixelekből állhat a rekonstruálandó kép. Számos ilyen prior információt vizsgáltak az évek során, melyek közül néhány példa: az alakzat orientációjára kerületére fókuszálva [14], konvexitás felhasználásával [15], lokális medián alkalmazásával [16] az alakzat középpontjának figyelembevételével [17], textúra alapú prior információval [18].

Ha a bináris rekonstrukció során textúra alapú prior információt kívánunk felhasználni, akkor egy jó választás lehet a Lokális Bináris Minták (Local Binary Patterns, LBP [19]), és variánsainak alkalmazása. A [18] irodalomban bizonyosságot nyert, hogy a variánsok közül az egyik legmegfelelőbb változat a Shift Local Binary Patterns, röviden SLBP [20].

Az SLBP egy osztályozáshoz gyakran használt eljárás, mely a képeken előforduló ismétlődéseket méri, és azok alapján segíti az osztályozás pontosságát. Az eljárás információt

biztosít számunkra az egyes pixelek és a szomszédaik közötti viszonyról, mely segítségével információt kaphatunk a kép mintázatáról. Ezt a textúra leíró egy 256-méretű hisztogramban tárolja, melyben összeadódik az összes pixelhez tartozó minta-leíró.

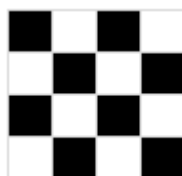
2.1.3. Relatív Átlagos Hiba

Ahhoz, hogy a rekonstrukció minőségét mérni tudjuk, szükségünk van valamilyen mértékre, mérőszámra. A Relatív Átlagos Hiba [21] a bináris tomográfiában széleskörűen használt hibabecslési módszer, mely az eredeti és a rekonstruált képek közti pixelenkénti eltérést adja meg százalékosan. Az angol elnevezésből (Relative Mean Error) az elterjedt rövidítését használjuk: RME.

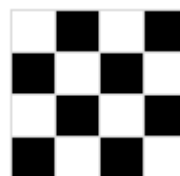
$$RME = \frac{\sum_i |x_i^o - x_i^r|}{\sum_i x_i^o} \cdot 100\%,$$

ahol az o felső index az eredeti képet jelenti, míg az r felső index a rekonstruált képet. Értelemszerűen, minél kisebb az érték, annál jobb a rekonstrukció minősége. Vigyázzunk azonban, ugyanis textúra-alapú rekonstrukció során tévútra vezethet az eredmény.

Vizsgáljuk meg az alábbi két példát! Az első esetben (4. ábra) az eltérés elhanyagolható a textúra szempontjából, mivel egy pixelnnyi eltolást (vagy invertálást) láthatunk egy tetszőleges irány mentén, ami texturális szempontból nem jelent szignifikáns különbséget. Ezzel szemben az RME érték a következő: $RME = \frac{16}{8} \cdot 100\% = 200\%$. Némi számolás után megérthetjük a félrevezető számértéket. A kép 44 méretű, azaz 16 pixel áll rendelkezésre, viszont csak a pixelek fele rendelkezik fekete színnel. Ezáltal az eredeti kép alapján a nevezőbe 8 kerül. Viszont minden pixel inverze az eredetinek, így az eltérés minden pixel esetén 1, azaz összesen 16. Ezáltal azt kapjuk, hogy az $RME = 200\%$, ami nem felel meg a valóságnak.



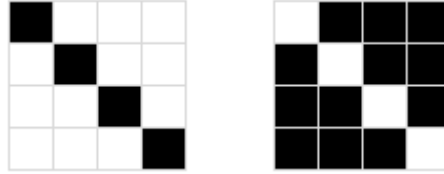
(a) eredeti kép



(b) rekonstruált kép

4. ábra Példa RME értékekre 1.

A második esetben (5. ábra) a két kép egymásnak szintén pixelenkénti inverze, viszont az eredeti képen csak 4 fekete pixel található, míg a pixelenkénti eltérés ugyanúgy 16. Ez alapján azt kapjuk, hogy $RME = \frac{16}{4} \cdot 100\% = 400\%$, ami duplája az első esetnek, viszont itt texturális szempontból teljesen rossz eredményt kaptunk.



(a) eredeti kép

(b) rekonstruált kép

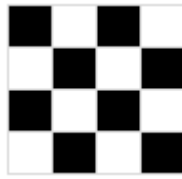
5. ábra Példa RME értékekre 2.

2.1.4. Pixelhiba

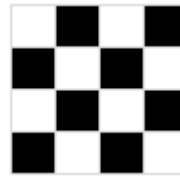
Egy másik alkalmazható hibabecslési módszer, mely százalékos értékben fejezi ki az eltérést két kép között, a pixelhiba (Pixel Error, PE). Nagyon hasonlít az RME-hez de vele ellentétben a kép méretéhez viszonyított eredményt ad.

$$PE = \frac{\sum_i |x_i^o - x_i^r|}{m \times n} \cdot 100\%,$$

ahol az o és az r felső indexek megegyeznek a relatív átlagos hibában taglaltakkal, viszont a pixelhibát nem az eredeti kép fekete pixeleinek a függvényében nézzük, hanem a képarány vonatkozásában, ezért a számlálóban az eredeti kép pixeleinek száma található. A pixelhiba értékelésével is csak óvatosan szabad bánnunk, az RME-hez hasonló okok miatt. Az előzőekben, a Relatív Átlagos Hiba példáján használt képeken ismertetem a pixelhibát is (6. ábra és 7. ábra). A képek megegyeznek, azonban a pixelhiba számításakor a hibát a vizsgált kép arányában nézzük, ezért a nevezőbe jelen esetben 16 kerül. Ennek következtében a két eltérő mintánál ugyanúgy azt kapjuk, hogy $PE = \frac{16}{16} \cdot 100\% = 100\%$. Ez texturális szempontból szintén félrevezető, hiszen az első eset a textúra szempontjából szinte hibátlan, míg a második eset teljes mértékben rossz.

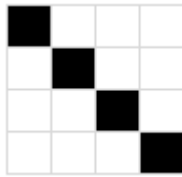


(a) eredeti

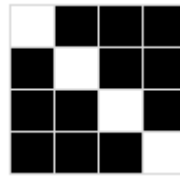


(b) rekonstruált

6. ábra Példa pixelhibára 1.



(a) eredeti



(b) rekonstruált

7. ábra Példa pixelhibára 2.

Összefoglalva tehát, elmondhatjuk, hogy a minőségmérésnél egyszerre kell figyelembe vennünk az eltérő hibabecslési módszerek által kapott értékeket, valamint a vizuális eredményeket is.

2.2. Morfológia

Ahhoz, hogy az eredményként kapott képet még inkább javítani tudjuk, használhatjuk az úgynevezett matematikai morfológiai műveleteket [22]. A morfológia alaktant, alakelemzést jelent, a morfológiai képfeldolgozás pedig a képen látható tárgyak alakjainak elemzésével foglalkozik. A két alapl művelete a dilatáció és az erózió. Mindkét művelet egy strukturáló elem – amely jellemzően egyszerű és kicsi mintát jelent – segítségével dolgozik. Ha nagyon egyszerűen akarjuk megfogalmazni, akkor azt mondhatjuk, hogy a dilatáció segítségével a képen található objektumok méretét tudjuk megnövelni, így a kisebb lyukakat „fel tudjuk tölteni” (8. ábra). A dilatációval szemben az erózió esetében az objektumok méretét csökkenthetjük, fogyaszthatjuk így az elenyésző kapcsolatok eltűnnek. A további morfológiai műveletet ezen két alapvető műveletből vezethetjük le (9. ábra).



(a) eredeti kép



(b) dilatált kép

8. ábra Példa dilatacióra



(a) eredeti kép



(b) erodált kép

9. ábra Példa erózióra

A nyitás egy erózió és egy dilatació egymás utáni végrehajtását jelenti. Használható az objektumokon kívüli zaj csökkentésére és az objektumok egymástól való szeparálására. Ezen műveletek végrehajtásával az objektumok csak csökkenhetnek, növekedni nem tudnak (10. ábra).



(a) eredeti kép



(b) nyitás utáni kép

10. ábra Példa nyitásra

A zárás a nyitás ellentetje, egy dilatació és egy erózió egymás utáni végrehajtásából áll. A nyitással szemben az objektumokon belüli zaj csökkentésére szolgál, illetve a művelet végrehajtása után az objektum csak növekedni tud, a csökkenés kizárt (11. ábra).



(a) eredeti kép (b) zárás utáni kép

11. ábra Példa zárásra

Egy magasabb fokú művelet a morfológiai szűrés, mely egy nyitásból és egy zárásból áll, tehát összesen 4 alpművelet egymás utáni végrehajtásából, melyek sorrendben: erózió és dilatació (a nyitás megvalósítására), majd dilatació és erózió (a zárás megvalósítására). Valójában a nyitás fázisban az erózió segítségével eltüntetjük a kisméretű zajokat, majd a dilatacióval visszaállítjuk az eredeti méretet, míg a zárás fázisban a dilatació segítségével kitöltjük a kisméretű lyukakat. Végül az erózió segítségével visszaállítjuk az objektumok eredeti méretét (12. ábra).

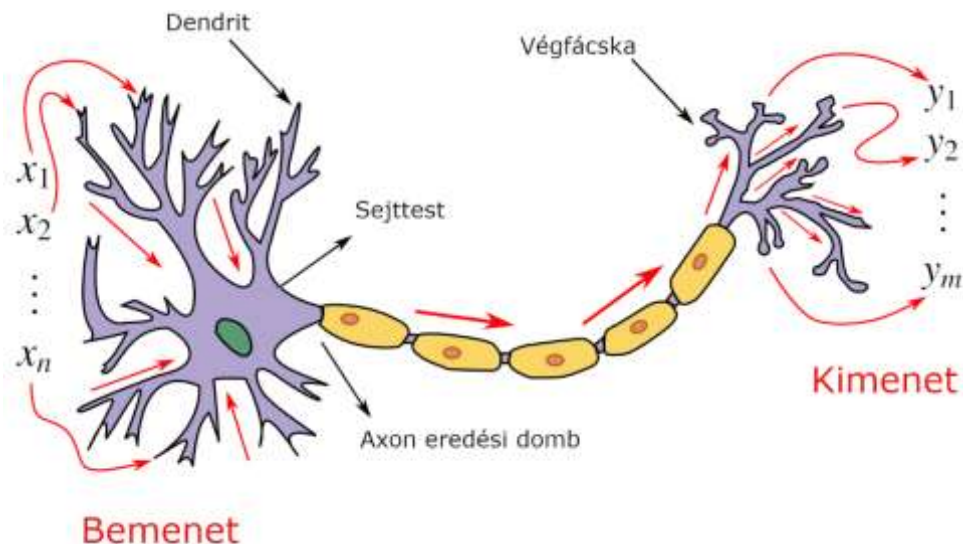


(a) eredeti kép (b) eredménykép (c) eredeti kép (d) eredménykép

12. ábra Példák morfológiai szűrésre

2.3 Neurális hálózat

Mint a legtöbb informatikai és matematikai innovációt, így a neurális hálózat eredetét is a természetben kereshetjük. Az emberi agyat becslések alapján 50-100 milliárd idegsejt alkotja, melyek a nyúlványaikon keresztül állnak egymással kapcsolatban [23]. Egy idegsejtet a nyúlványaival együtt neuronnak hívunk. Egy kimeneti (axon) és több bemeneti (dendritek) nyúlvánnyal rendelkeznek (13. ábra).

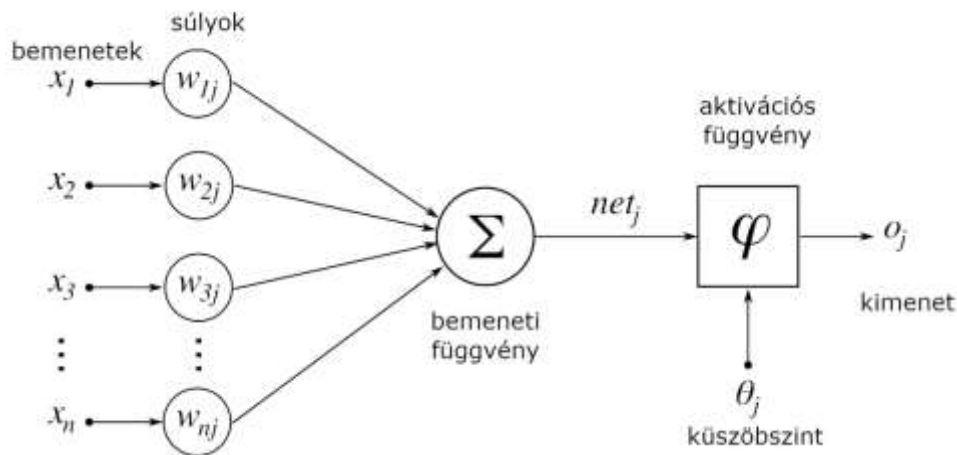


13. ábra Egy neuron vázlatos képe

Ezen nyúlványok közötti kommunikáció (elektromos aktivitás) felel egy-egy gondolatunkért, döntésünkért, vagy tettünkért. Az emberi agy analógiájára született meg a mesterséges neurális hálózat, amely működését, döntését és tanulását tekintve hasonlít az emberi agyra. A neurális hálózat rétegekből, a rétegeket felépítő neuronokból és a neuronok közötti kapcsolatból állnak. A rétegek közül a bemeneti és a kimeneti réteg kitüntetett, a kettő között elhelyezkedő réteget (vagy rétegeket) együttesen rejtett rétegeknek nevezzük. Egy réteg egy neuronja összeköttetésben áll a következő réteg összes neuronjával, a valós kapcsolatokat pedig ezen összeköttetéseknek a súlyai fejezik ki. Ezen kívül minden neuronhoz tartozik egy plusz érték, amivel az adott neuron elfogultsága állítható (ezt bias értéknek hívjuk). A neurális hálózatok a tanulást a környezetben tapasztalt minták értelmezése révén valósítja meg. A tanulás folyamata nem jelent mást, mint a súlyoknak és elfogultsági mértékeknek a finomhangolása, amíg a kívánt eredményt el nem érjük [24]. Az, hogy egy neuron tüzel-e, illetve ennek a mértékét – tehát a kimeneti kapcsolata aktiválódását és nagyságát –, a súlyozott bemeneti kapcsolatok összértéke és a neuronhoz tartozó aktivációs függvény dönti el.

Az előző fejezetben részletezett bináris tomográfia problémára számos megoldás létezik, közöttük például sztochasztikus módszerek (szimulált hűtés), vagy gradiens alapú determinisztikus módszerek. Az általam vizsgált extrém kevés vetület esetén azonban a probléma megoldása nem egyszerű. Gépi tanuló algoritmusok tervezése során sok esetben nem szükséges az adott folyamat teljes körű ismerete, így esetemben is hasznos lehet a megvalósításban. Éppen ezért esett választásom a neurális hálózat használatára, mellyel feltevéseim szerint jelentősen tudom növelni a két vetületből történő rekonstrukció minőségét. A hálózat azonban nem működik automatikusan, megfelelően kell paraméterekkel ellátni, azaz biztosítani kell a korrekt tanulást, melyet az eltérő aktivációs függvények, optimalizálók, regularizálás helyes megválasztásával, és egyéb módszerekkel tehetünk meg. Egy neurális hálózat kialakításában ezek a feladatok jelentik a legnagyobb kihívást a hiperparaméterek megfelelő finomhangolása mellett.

A neurális hálózatok típusai közül az egyik fő ág az előrecsatolt hálózatok (feedforward network), melyek közül az egyik legelterjedtebb alkategória a többrétegű perceptron (Multi-Layer-Perceptron, MLP) [25]. A neurális hálózat könnyen leírható matematikai modellel is az alábbi módon (14. ábra):



14. ábra Neurális hálózat matematikai modellje

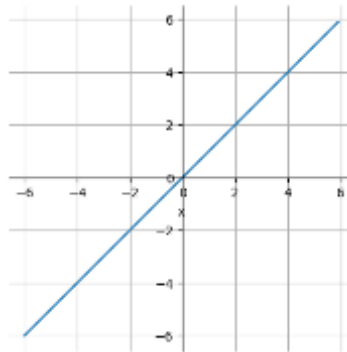
2.3.1. Aktivációs függvények

A neurális hálózatban sokféle aktivációs függvény közül választhatunk, amely nagyban befolyásolja a hálózat tanulási képességét, taníthatóságát és az elérhető eredmény minőségét is. Az alábbiakban a teljesség igénye nélkül mutatok be néhányat közülük.

2.3.1.1. Lineáris aktivációs függvény

A legegyszerűbb aktivációs függvény, mely változatlanul hagyja a bemenetet, az a lineáris aktivációs függvény. A legtöbb esetben nem használható felügyelet nélküli tanításra, ugyanis megakadályozza a hibavisszaterjesztést (backpropagation). A lineáris függvény gradiense 0, így a gradiens süllyedést nem lehet felhasználni a hibavisszaterjesztésre.

$$\text{linear}(x) = x$$

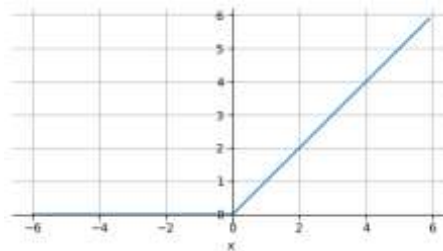


15. ábra Lineáris aktivációs függvény

2.3.1.2. Egyenirányított lineáris egység aktivációs függvény

Az egyenirányított lineáris egység aktivációs függvény (Rectified Linear Unit - ReLu) [26] a negatív bemeneteket 0-ra állítja, míg a pozitív bemeneteket változatlanul hagyja, ezáltal alacsony számítási igénnyel rendelkezik. A rejtett rétegekben az egyik leggyakrabban használt aktivációs függvény, a szakirodalom alapján gyakorlati problémák esetén nagyobb pontosságot eredményezhet, így használatát komolyan fontolóra vettem.

$$\text{relu}(x) = \max(0, x)$$

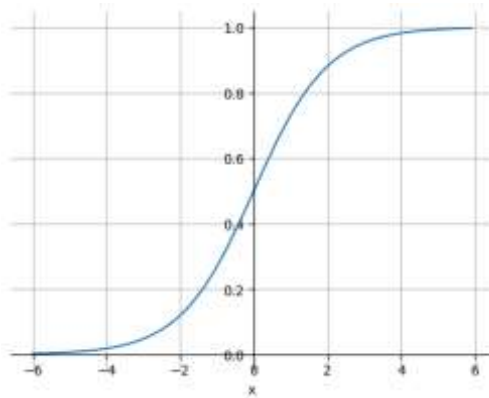


16. ábra ReLu függvény

2.3.1.3. Sigmoid aktivációs függvény

A sigmoid aktivációs függvény alatt egy úgynevezett ‘S’ alakú függvényt értünk. Segítségével garantálhatjuk, hogy bármilyen bemenet esetén a kimenet a $]0; 1[$ intervallumra képződik le.

$$\text{sigmoid}(x) = 1 / (1 + \exp(-x))$$



17. ábra Sigmoid aktivációs függvény

2.3.2. Optimalizációs algoritmusok

A különféle optimalizációs algoritmusok segítségével tanítható a neurális hálózat, amely tulajdonképpen egy szélsőérték-keresési feladat. Az alábbiakban a lehetőségek közül említek meg néhányat, rövid jellemző ismertetéssel. Az algoritmusok részletesebb, matematikai ismertetése jelen dolgozatom terjedelmén kívül esik.

2.3.2.1. Sztochasztikus gradiens süllyedés

A sztochasztikus gradiens süllyedés (Stochastic Gradient Descent, SGD) iteratív módszer olyan célfüggvény-érték minimalizálására, amely (szub)differenciálható függvények összegeként írható fel. [27]

2.3.2.2. Adaptív gradiens módszer

Az adaptív gradiens módszer (Adaptive Gradient, AdaGrad) az SGD egy módosított változata, mely az egyes paraméterek tanulási sebességét automatikusan határozza meg. [28]

2.3.2.3. Adaptív lendület módszer

Az adaptív lendület módszer (Adaptive Moment Estimation, Adam) a mély neurális hálók tanításának alapértelmezett módszere, mely adaptálódik a hibafelülethez, a gradiens és a gradiens második momentumainak futóátlagait használja. [29]

2.3.3. Regularizációs technikák

Regularizációs technikák segítségével javítható egy neurális hálózat modellje, mégpedig oly módon, hogy csökkenti az ún. túltanulás, illetve a hibás tanulás mértékét. Az alábbiakban erre két megoldást vetek fel.

2.3.3.1. Kiejtéses regularizáció

A kiejtéses regularizáció (angolul dropout), egy speciális réteg, mely a beállított mértékben véletlenszerűen korlátozza két réteg között a neuronok kapcsolatait úgy, hogy az előtte levő réteg neuronjainak konfigurációban kapott százalékát 0-ás súllyal látja el. Ezzel megakadályozza az általuk hordozott információ továbbítását a következő réteg számára. Egyszerűen kifejezve az előtte lévő réteg neuronjainak adott százalékát figyelemen kívül hagyja, így csökkentve a túltanulást és emelve a hálózat általánosító képességét.

2.3.3.2. Köteg-normalizáció

A köteg-normalizáció (batch normalization) olyan transzformációt alkalmaz, amely a kimenetek átlagát 0 közelében, míg a kimenetek szórását 1 közelében tartja, de ezt mindig csak az aktuális kötegre teszi meg. Gyorsabb és stabilabb tanítást eredményezhet.

3. Módszertan

Elsőnek a megfelelő programozási nyelv kiválasztásával foglalkoztam, ahol az általam jobban ismert nyelveket értékeltem ki (C++, NodeJS, Java, Python). A kiértékelés során figyelembe vettem az alapvető képfeldolgozás és mesterséges intelligencia lehetőségeket is. Fontos szempont volt számomra a nyelv egyszerű használhatóságán túl az adott könyvtár, vagy keretrendszer mögött álló közösség nagysága is. Ezen szempontok figyelembevételével a választásom a Python nyelvre esett a TensorFlow, illetve az e fölötti magas szintű API, a Keras felhasználásával.

A nyelv kiválasztása után számításba vettem azon lehetőségeket, amelyek segíthetik és gyorsíthatják a munkámat. Ezek alapján figyelembe vettem a grafikus feldolgozó egység (Graphics Processing Unit, GPU) felhasználását és az elkülöníthető részek – mint például adatelőkészítés; neurális hálózat tanítása, kiértékelése; eredmény megjelenítése numerikusan és vizuálisan egyaránt – újrafuttathatóságát is. Ebben nagy segítséget jelentett kezdetben a Jupyter Notebook, mivel a különálló szekciókban írt programok egymástól függetlenül újrafuttathatóak. Ennek eredményeként nem volt szükséges újra és újra elvégezni az adatelőkészítést egyes ötletek kipróbálására, finomhangolására. A Jupyter Notebook-ot nagyszerűen kiegészítette a Google Colaboratory. Ennek segítségével felhő alapú környezetben tudtam Jupyter Notebook-okat futtatni, melyeket Google Drive-on tároltam. Ez utóbbinál két fő korlátozásba futottam bele: nagyobb feladatoknál (hosszabb futtatásoknál) lejárhat az aktuális munkamenet, illetve elfogyhat az ingyenesen biztosított memória.

A kezdeti próbálgatások után végül egy offline környezetet alakítottam ki, melyet a saját számítógépemem tudtam futtatni, amely egy 8 magos, magonként 3.4 GHz-es Intel Core i7-6700-as processzort, egy NVIDIA GeForce GTX 1060 3GB-os grafikus kártyát és 16 GB Kingston Fury DDR4-es memóriát foglal magába amit egy Linux operációs rendszerrel használtam. Ehhez a főbb felhasznált könyvtárak:

- a képek alapvető feldolgozására: Pillow
- a neurális hálózatokhoz: TensorFlow és Keras
- eredmények vizuális megjelenítéséhez: matplotlib a pyplot támogatásával
- a nagyobb mátrixműveletek (mint például egy bináris kép betöltése mátrixként) támogatására: numpy.

A program felépítése a következő:

1. képek betöltése
2. szükséges tulajdonságok kinyerése
3. modell megalkotása a neurális hálózathoz
4. modell tanítása
5. az elkészült eredmények kiértékelése

A neurális hálózat modelljében hiperparaméternek tekintetem az optimalizációs függvényt és a tanítási rátát, az aktivációs függvényt, a regularizációt, a rejtett rétegek számát és a tartalmazott neuronokat. A tanítási, validációs és tesztelési halmazokhoz az elérhető adathalmaz 60% - 20% - 20% illetve a 80 % - 10% - 10%-os bontását vettem alapul.

Optimalizációs függvények közül az SGD és az Adam algoritmust vizsgáltam, melyek mindegyike sztochasztikus gradiens süllyedés alapú optimalizációt végez, a hozzájuk tartozó tanítási rátát pedig a $[0.1, 0.0001]$ intervallumon mozgattam.

Az aktivációs függvények közül a kimeneti rétegre a szigmoid függvényt választottam a rekonstruálandó kép bináris mivolta miatt, mivel így az egyes kimeneti neuronok a $[0, 1]$ intervallumról vehetnek fel értékeket, melyeket egészen kerekítve eredményül kapjuk a kívánt bináris értékeket. Rejtett rétegek esetén pedig a lineáris, a szigmoid és a ReLU aktivációs függvényeket teszteltem.

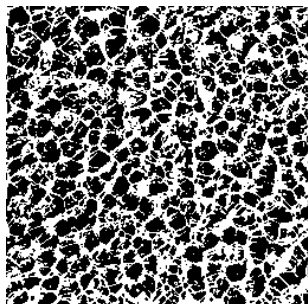
Regularizációs rétegek közül tesztelésre kerültek a kiejtéses és a köteg-normalizációs rétegek is, melyek közül a kiejtéses réteget teszteltem 5%, 10%, 15% és 20%-os rátával is.

Az eredeti képből nyert vetületi adatok és a hozzá kiszámított textúra-leíró képezi a neurális hálózat bemenetét, míg a kimenetén a rekonstruált kép áll elő. Ez alapján a bemeneten 256 neuronra van szükség a textúra-leíró miatt, illetve $n \times m$ méretű kép esetén további $n + m$ mennyiségű neuron a vetületek miatt. A kimeneten pedig $n \times m$ mennyiségű neuron szükséges, ahol minden neuron egy pixelt reprezentál a rekonstruált képből. A rejtett rétegek számát 1 és 3 között vizsgáltam, a neuronok számát – néhány eset kivételével – állandónak tekintve. Az egyes rejtett rétegekben elhelyezkedő neuronok számát a bemeneti réteg és a kimeneti réteg határozta meg a következő módon. A neuronok száma vagy megegyezett a bemeneti réteg méretének egész számú szorzatával – azaz a bemeneti réteg méretének 1, 2 vagy

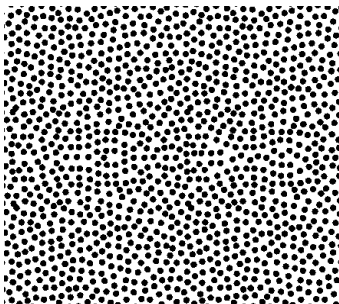
3-szorosával –, vagy megegyezett a kimeneti réteg egész számú szorzatával – azaz a kimeneti réteg méretének 1, 2 vagy 3-szorosával – vagy pedig a bemeneti és kimeneti réteg átlagának egész számú többszörösével – azaz 1, 2 vagy 3-szorosával –. Ez utóbbira példa 144 bemeneti és 256 kimeneti neuron esetén a neuronok száma: $(144 + 256)/2 = 200$.

4. Elért eredmények ismertetése

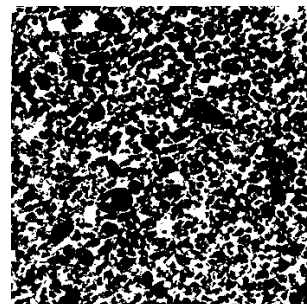
A neurális hálózat tanításához, valamint az algoritmus minőségének teszteléséhez 3 eltérő alapképet használtam, melyből képosztályokat alakítottam (18. ábra).



(a) Példa alapkép a Brodatz textúra adatbázisból [30]



(b) Szoftveres fantom alapkép



(c) Bentheimer hálózat minta alapkép [31]

18. ábra Képosztályok alapképei

Kezdetben a 3 képosztályra külön-külön hálózatokat hoztam létre és tanítottam, viszont így a szoftveres fantom képosztály (18. ábra (b)) esetében túlillesztés történt, melyből adódóan sajnos elvesztette az általánosító képességét a modell. Ennek oka, hogy könnyen felismerhető textúrára tanult rá. Ellenben a másik két képosztály (18. ábra (a) és (c)) esetén nem kielégítő eredmények születtek a jól felismerhető textúra hiánya miatt. Ezen okból kifolyólag a 3 képosztályból ugyanannyi mintavételezést követően az így létrehozott vegyes halmazzal folytattam a vizsgálatokat.

Az SGD optimalizáló használatával sajnos nem sikerült érdemleges eredményt elérnem egyik általam próbált aktivációs függvény segítségével sem. Ezzel szemben, az Adam optimalizáló használatával már szembetűnő eredményhez jutottam. A választott kötegméret 64-es és 128-as érték esetén bizonyult a legjobbnak, így ezen kötegmértékek mentén végeztem a további tesztelést. Jellemzően 0.001 és 0.0001 közötti tanulási rátával sikerült elérni jobb eredményeket míg 0.01 felett már kifejezetten romlott a tanulás minősége. A tanítás késői fázisaiban a neurális hálózatok esetében elterjedt korai megállás lehetőségét használtam. Ez segít a túltanulás megelőzésében oly módon, hogy bizonyos türelmi mutató mellett amennyiben több iteráción keresztül folyamatosan romlik a vizsgált metrika, akkor leállítja a tanítást az adott tanítási fázis (epoch) végén. A korai megállás segítségével több, összetettebb tanítást tudtam elindítani akár több százas nagyságrendű tanítási fázis beállítása mellett is.

A rekonstrukció hatékonyságára tesztelést végeztem pusztán a vetületi adatokból, illetve kiegészítve azokat az SLBP textúraleíróból nyert előzetes információval. Tapasztalataim alapján, amennyiben a tanítási és tesztelési adatok azonos képosztályból származnak, úgy az előzetes információ használata nem javít szignifikánsan a rekonstrukció minőségén.

A továbbiakban 3 különböző eredményt mutatok be, melyek a kiválasztott bemeneti tulajdonságok miatt felépítésükben is eltérnek. Az első esetben csupán vetületi adatokat használtam fel, a második esetben a vetületi információkat kiegészítettem az SLBP tulajdonsággal, míg az utolsó esetben kísérleti jelleggel az alap vetületi adatokat kiegészítettem a vetületek alapján előállított, Ryser algoritmus által generált képpel.

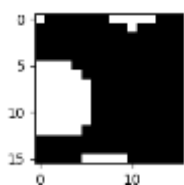
4.1. Eredmények vetületi adatok felhasználásával

Képcsoportonként 10 000 darab, azaz összesen 30 000 darab, egyenként 16×16 méretű mintával csupán vetületi bemeneteket használtam fel. A kimeneti réteg 256 méretű az előállítani kívánt kép méretei miatt, míg a bemenet ezen okból adódóan 32 méretű. Ebben az esetben 3 köztes réteget használtam. Közvetlenül a bemeneti réteg után egy 144 méretű, majd pedig egy 256 méretű réteget alkalmaztam, és mindegyiket ReLU aktivációs függvénnyel konfiguráltam. Ebben az esetben kísérleteztem mind köteg-normalizációval, mind kiejtéses regularizációval, de a minimálisan hosszabb tanítási folyamat ellenére nem sikerült érdemben jobb eredményt elérnem, mint a regularizációs technika használata nélkül. Az egyes tanítások után a teszhalmaz minden eleméhez kiszámoltam a neurális hálózat modell (NN) és a Ryser algoritmus által is rekonstruált kép pixelhibáit az eredeti képhez véve. Ezen értékeket összegyűjtöttem, majd az így kapott adatsorból meghatároztam a rá jellemző minimális, maximális, átlagos és medián értékeket, illetve az adathalmaz szórását (1. táblázat). A táblázat első sora tartalmazza a Ryser algoritmusra vonatkozó értékeket, a második pedig a modell által elért eredményeket a táblázat fejlécében feltüntetett értékekre vonatkozóan.

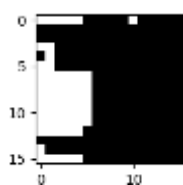
	min (%)	max (%)	átlag (%)	szórás (%)	medián (%)
Ryser	0.00	52.34	25.38	10.21	25.78
NN	0.00	42.58	20.04	7.76	20.31

1. táblázat PE értékek Ryser algoritmus és NN által kapott eredmények esetén

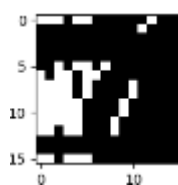
Szemléltetés céljából feltüntettem néhány ilyen képet a teszt adathalmazból véve, melyek bal oldalán az eredeti kép helyezkedik el, középen az aktuális modell segítségével előállított kép, illetve jobb oldalon a Ryser algoritmussal rekonstruált kép (19. ábra és 20. ábra).



(a) Eredeti kép

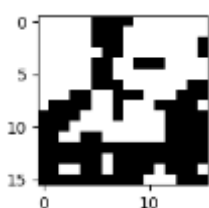


(b) NN eredménye

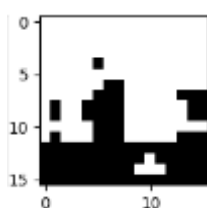


(c) Ryser algoritmus eredménye

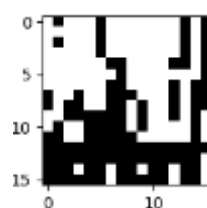
19. ábra Példa eredmény a Szoftveres fantom képosztályra (18. ábra (b)) futtatott tesztek



(a) Eredeti kép



(b) NN eredménye



(c) Ryser algoritmus eredménye

20. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztek

4.2 Eredmények vetületi és SLBP adatok felhasználásával

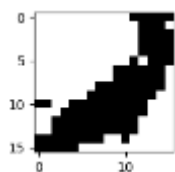
Képcsoportonként 10 000 darab, azaz összesen 10 000 darab, egyenként 16×16 méretű minta esetén, a vetületi információ mellett SLBP hozzáadásával szintén jó eredményeket sikerült elérnem. Az előző példához hasonlóan 2 rejtett réteget használtam, viszont a bemeneti réteg itt már $32 + 256 = 288$ méretű, míg a kimeneti csupán 256. Ezen okból kifolyólag az első rejtett réteg 272, míg a második 256 méretű, mindkettő ReLU aktivációs függvényrel konfigurált. A 2. táblázat tartalmazza a vonatkozó tesztelések minimális, maximális, átlagos értékeit, továbbá a szórás és medián értékeket a korábbiakhoz hasonló módon.

	min (%)	max (%)	átlag (%)	szórás (%)	medián (%)
Ryser	0.00	52.34	25.20	9.97	25.78
NN	0.00	50.39	18.54	7.54	18.75

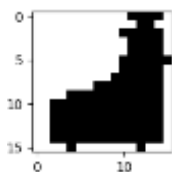
2. táblázat PE értékek Ryser algoritmus és NN által kapott eredmények esetén

Szemléltetés céljából itt is feltüntettem néhány ilyen képet a teszt adathalmazból véve, melyek bal oldalán az eredeti kép helyezkedik el, középen az aktuális modell segítségével

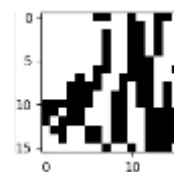
előállított kép, illetve jobb oldalon a Ryser algoritmussal rekonstruált kép (21. ábra és 22. ábra és 23. ábra).



(a) Eredeti kép

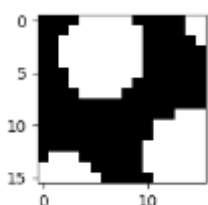


(b) NN eredménye

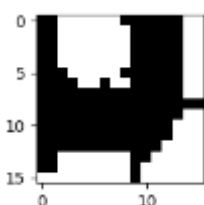


(c) Ryser algoritmus eredménye

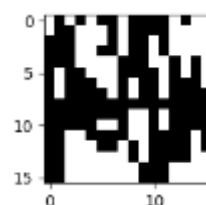
21. ábra Példa eredmény a Bentheimer hálózat képosztályra (18. ábra (c)) futtatott tesztek



(a) Eredeti kép

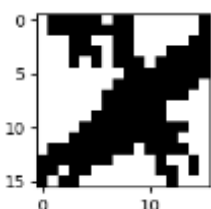


(b) NN eredménye

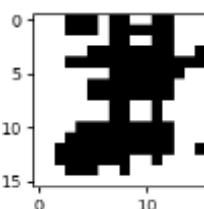


(c) Ryser algoritmus eredménye

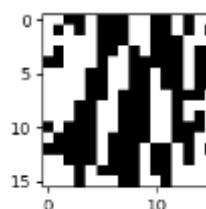
22. ábra Példa eredmény a Szofiveres fantom képosztályra (18. ábra (b)) futtatott tesztek



(a) Eredeti kép



(b) NN eredménye



(c) Ryser algoritmus eredménye

23. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztek

4.3 Eredmények a Ryser algoritmussal előállított kép kiegészítésével

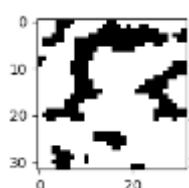
Ennél a tesztetnél képcsoportonként 2 000 darab, azaz összesen 6 000 darab, egyenként 32×32 méretű mintával dolgoztam. A bemenetet az alap vetületi adatokon felül kiegészítettem a vetületek alapján előállított, Ryser algoritmus által generált képpel. Ennél a vizsgálatnál is sikerült jó eredményeket produkálni. A modellben 1024-es méretű bemeneti és

szintén 1024-es méretű kimeneti réteg között egy darab, szintén 1024-es méretű rejtett réteget alkalmaztam regularizáció nélkül. A rejtett rétegben ReLU típusú, míg az output rétegben szigmoid típusú aktivációs függvényt használtam. A 3. táblázat tartalmazza a vonatkozó tesztelések minimális, maximális, átlagos értékeit, továbbá a szórás és medián értékeket a korábbiakhoz hasonló módon.

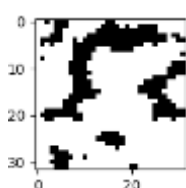
	min (%)	max (%)	átlag (%)	szórás (%)	medián (%)
Ryser	13.09	45.44	35.30	5.72	35.74
NN	0.00	46.78	14.78	9.57	13.96

3. táblázat PE értékek Ryser algoritmus és NN által kapott eredmények esetén

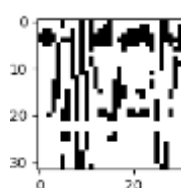
A korábbiakhoz hasonlóan, szemléltetés céljából itt is feltüntettem néhány ilyen képet a teszt adathalmazból véve, melyek bal oldalán az eredeti kép helyezkedik el, középen az aktuális modell segítségével előállított kép, illetve jobb oldalon a Ryser algoritmussal rekonstruált kép (24. ábra és 25. ábra és 26. ábra).



(a) Eredeti kép

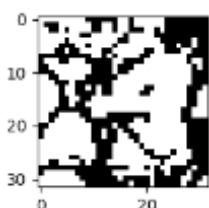


(b) NN eredménye

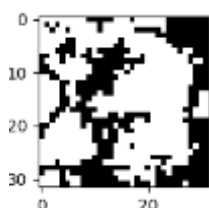


(c) Ryser algoritmus eredménye

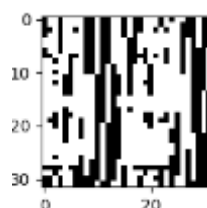
24. ábra Példa eredmény a Bentheimer hálózatra képosztályra (18. ábra (c)) futtatott tesztekkel



(a) Eredeti kép

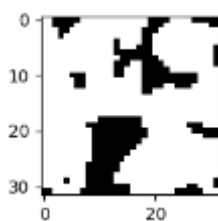


(b) NN eredménye

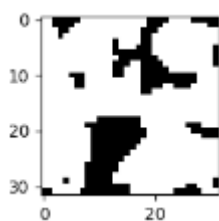


(c) Ryser algoritmus eredménye

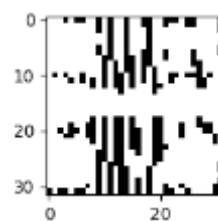
25. ábra Példa eredmény a Brodatz képosztályra (18. ábra (a)) futtatott tesztekkel



(a) Eredeti kép



(b) NN eredménye



(c) Ryser algoritmus eredménye

26. ábra Példa eredmény a Bentheimer hálózat képosztályra (18. ábra (c)) futtatott tesztek

A fejezetben ismertetett eredmények alapján elmondhatjuk, hogy általában ugyan numerikusan minimális javulás fedezhető fel az általam előállított neurális hálózati modell eredményeiben a Ryser algoritmusához képest, ámbar vizuálisan jelentős minőségjavítás érhető el. Mivel a két vetület rettentően kevés információt hordoz magában, így ez a javulás szignifikánsnak mondható. Az általam vizsgált extrém eset a jelenlegi szakirodalomban kevésbé kutatott terület, így más módszerek eredményeivel történő összehasonlítás nehezen kivitelezhető.

5. Összegzés

Dolgozatomban azzal foglalkoztam, hogy a bináris képrekonstrukció extrém kevés vetületből történő megvalósítása esetén hogyan tudnám javítani a rekonstrukció minőségét. Ehhez neurális hálózat alkalmazása mellett döntöttem, melynek paraméterezését a szakirodalmi áttekintés mellett empirikus teszteléssel oldottam meg.

A bináris tomográfia problémájában széleskörűen alkalmazott, Ryser algoritmusához képest jelentős javulást értem el a felhasznált hibabecslési módszerek alapján. Felismerhető textúrával rendelkező képek esetén, vizuálisan szignifikánsan jobb eredményt produkálható, amely bizonyította a kezdeti intuíciót. Bináris tomográfia és neurális hálózat ilyen jellegű kapcsolatáról és együttes felhasználásáról nem találtam hasonlót, így nem tudtam más hasonló eredményekkel összevetni az elért eredményeimet.

Az eredmények nagyon biztatóak, melyek alapján a későbbiekben érdemes további mélyreható vizsgálatokat végezni a területen, mellyel kapcsolatban már most számtalan ötletem van. A későbbiekben az alábbi kérdésekre is keresem majd a választ:

- Vetületek érzékenysége: több, de még mindig kevés vetület esetén szignifikáns javulás érhető-e el?
- Hibatűrő képesség: mennyire érzékeny a mesterséges neurális hálózattal támogatott rekonstrukció a mintavételezés közben felmerülő hibákra?

Ezeneken felül, a jövőben a kutatásomat szeretném kiterjeszteni a szakirodalmi áttekintésben részletezett morfológiai műveletekre is, vizsgálva, hogy a mostani eredmények tovább javíthatóak-e ezen műveletek mesterséges hálózatba való építésével.

6. Irodalomjegyzék

- [1] Omkar Dokur, Srinivas Katkoori, and Nouredddine Elmehraz. “Embedded system design of a real-time parking guidance system”. In: 2016 Annual IEEE Systems Conference (SysCon). 2016, pp. 1–8.
- [2] Juan Guerrero-Ibáñez, Sherali Zeadally, and Juan Contreras-Castillo. “Sensor technologies for intelligent transportation systems”. In: *Sensors* 18.4(2018), p. 1212.
- [3] Tamás Grósz et al. “QR code localization using deep neural networks”. In: 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP). 2014, pp. 1–6.
- [4] Yue Liu, Ju Yang, and Mingjun Liu. “Recognition of QR Code with mobile phones”. In: 2008 Chinese Control and Decision Conference. 2008, pp. 203–206.
- [5] Sumit Tiwari. “An Introduction to QR Code Technology”. In: 2016 International Conference on Information Technology (ICIT). 2016, pp. 39–44.
- [6] Ruey-Kai Sheu et al. “STAM-CCF: suspicious tracking across multiple camera based on correlation filters”. In: *Sensors* 19.13 (2019), p. 3016.
- [7] Avinash C. Kak and Malcolm Slaney. “Principles of computerized tomographic imaging. 1988”. In: IEEE, New York (1988).
- [8] Richard Gordon, Robert Bender, and Gabor T. Herman. “Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography”. In: *Journal of theoretical Biology* 29.3 (1970), pp. 471–481.
- [9] Gabor T. Herman and Attila Kuba. *Discrete Tomography: Foundations, Algorithms, and Applications*. Springer Science & Business Media, 1999.
- [10] Gabor T. Herman and Attila Kuba. *Advances in discrete tomography and its applications*. Springer Science & Business Media, 2008.
- [11] Gabor T. Herman. *Fundamentals of computerized tomography: image reconstruction from projections*. Springer Science & Business Media, 2009.

- [12] Cécile R L P N Jeukens et al. “Probability of receiving a high cumulative radiation dose and primary clinical indication of CT examinations: a 5-year observational cohort study”. In: *BMJ Open* 11.1 (2021)
- [13] Herbert J. Ryser. “Combinatorial properties of matrices of zeros and ones”. In: *Canadian Journal of Mathematics* 9 (1957), pp. 371–377.
- [14] Tibor Lukic and Péter Balázs. “Binary tomography reconstruction based on shape orientation”. In: *Pattern Recognition Letters* 79 (2016), pp. 18–24.
- [15] Sara Brunetti et al. “Reconstruction of 4-and 8-connected convex discrete sets from row and column projections”. In: *Linear Algebra and its Applications* 339.1-3 (2001), pp. 37–57.
- [16] Sakari Alenius, Ulla Ruotsalainen, and Jaakko Astola. “Using local median as the location of the prior distribution in iterative emission tomography image reconstruction”. In: *IEEE Transactions on Nuclear Science* 45.6 (1998), pp. 3097–3104.
- [17] Tibor Lukić and Péter Balázs. “Limited-view binary tomography reconstruction assisted by shape centroid”. In: *The Visual Computer* 38.2 (2022), pp. 695–705
- [18] Judit Szűcs and Péter Balázs. “Binary Tomography Using Variants of Local Binary Patterns as Texture Priors”. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2019, pp. 141–154.
- [19] Timo Ojala, Matti Pietikäinen, and David Harwood. “A comparative study of texture measures with classification based on featured distributions”. In: *Pattern Recognition* 29.1 (1996), pp. 51–59.
- [20] G. Kylberg és I.-M. Sintorn. “Evaluation of noise robustness for local binary pattern descriptors in texture classification”. *EURASIP Journal on Image and Video Processing* 2013.1:17 (2013).
- [21] Attila Kuba et al. “Medical applications of discrete tomography”. In: *Discrete Mathematical Problems with Medical Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, AMS 55(2000), pp. 195–208.
- [22] G Matheron. “Random Sets and Integral Geometry, Jonn Wiley & Sons”. In: Inc., New York (1975).], [Jean Serra. “Image analysis and mathematical morphology”. (1982).

- [23] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: Psychological review 65.6 (1958), p. 386.
- [24] Horváth Gábor. “Neurális hálózatok, Hungarian Edition Panem Könyvkiadó Kft”. In: Budapest 447 (2006)
- [25] M.W Gardner and S.R Dorling. “Artificial neural networks (the multi-layer perceptron)—a review of applications in the atmospheric sciences”. In: Atmospheric Environment 32.14 (1998), pp. 2627–2636.
- [26] Matthew D Zeiler et al. “On rectified linear units for speech processing”. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE. 2013, pp. 3517–3521.
- [27] Léon Bottou et al. “Online learning and stochastic approximations”. In: On-line learning in neural networks 17.9 (1998), p. 142
- [28] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: Journal of machine learning research 12.7 (2011).
- [29] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: arXiv preprint arXiv:1412.6980 (2014).
- [30] Phil Brodatz. Textures: a photographic album for artists and designers. Dover Pubns, 1966.
- [31] Chul Moon and Matthew Andrew. Bentheimer networks.
<http://www.digitalrockportal.org/projects/223>. (2019)