

TDK-DOLGOZAT

Szerző: Ráczi Adrienn

Utolsó módosítás dátuma: 2022. december



Pázmány Péter Katolikus Egyetem

Információs Technológiai és Bionikai Kar

Félautomata annotációs program fejlesztése szegmentációs problémákra - Infarktusos szívszeletek értékelése mesterséges intelligenciával

Szerző: Ráczi Adrienn

mérnökinformatikus BSc, 4. évfolyam

Témavezetők: Dr. Horváth András (egyetemi docens), Kovácsné

Csenger (PhD hallgató)

Köszönetnyilvánítás

Szeretnék köszönetet mondani dr. Horváth András témavezetőmnek, aki végig felügyelte a munkámat és mindig válaszolt az esetlegesen felmerülő kérdéseimre, valamint rendelkezésemre bocsátotta a PPKE ITK laboratóriumát, ahol a neurális hálóim tanítását tudtam végezni.

Szeretném megköszönni Kovácsházi Csenger konzulensemnek, aki a hálók tanításához és teszteléséhez szükséges képeket szolgáltatta, mindig a rendelkezésemre állt, ha segítségre volt szükségem, és segített megérteni a munkám statisztikával kapcsolatos részét.

Tartalomjegyzék

Köszönetnyilvánítás	V
1. Bevezetés	1
2. A konvolúciós neurális hálózatok bemutatása	3
2.1. A neurális hálózatok általános jellemzői	3
2.2. A szegmentációs hálók	6
2.2.1. U-Net	6
2.2.2. U-Net++	7
2.2.3. DeepLabV3	8
2.2.4. A U-Net, U-Net++ és a DeepLabV3 viszonya	8
3. A neurális hálók tanítása és tesztelése	10
3.1. Adatok előfeldolgozása	10
3.2. A hálók tanítási paramétereinek és jellemzőinek beállítása	11
3.2.1. U-Net	11
3.2.2. U-Net++	13
3.2.3. DeepLabV3+	14
3.3. Hálók tanítása és előzetes tesztelése	14
4. A neurális hálók tesztelése nagy méretű tesztalmazon	17
4.1. Tesztelés különböző metrikákkal	17
4.2. A tanított hálók hatékonyságának összehasonlítása	20
4.3. Scoring mechanizmus a háló kiértékelésének minőségére	22
4.3.1. Átlagos különbség	23
4.3.2. Szórás és átlag hányadosa	24
4.3.3. Pixelék valószínűségének átlaga	26
4.3.4. AUC értékek	28
4.4. Tanításhoz használt képek számának hatása a hálók hatékonyságára . .	28

4.4.1. Igazságmátrix	30
5. Összefoglalás	33
5.1. Elvégzett munka	33
5.2. Eredmények	34
5.3. Jövőbeni tervek	34
Felhasznált irodalom	38

1. fejezet

Bevezetés

A pre-klinikai kardiovaszkuláris kutatások során bevett gyakorlat állatmodellekben - például egerekben, patkányokban - szívinfarktus kiváltása. Ezen állatok szívének felépítése megfelelő mértékben hasonlít az emberi szívre [1], így alkalmas különböző kezelések és beavatkozások hatékonyságának vizsgálatára és kiértékelésére.

A kísérlet végén a szívet felszeletelik, illetve Evans Blue és 2,3,5-Trifenil-tetrazolium-klorid festékekkel festik, a szeletekről képeket készítenek, majd azokat manuálisan annotálják az oxigénhiányt elszenvedett, illetve az elhalt szövet területeinek meghatározására. Az annotált területek elhelyezkedése hasznos információkat szolgáltat a különböző kardiovaszkuláris betegségek mechanizmusairól, valamint az annotálásból nyert információk elengedhetetlenek a betegségek kezelési módjának kidolgozásához [1].

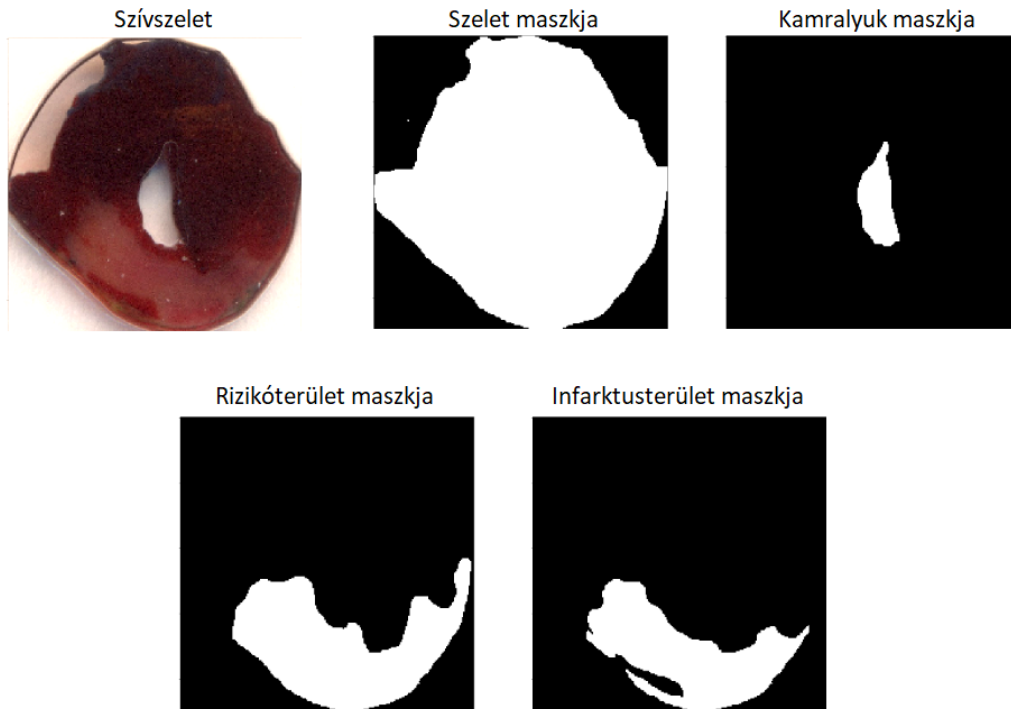
Az annotálás hosszadalmas munka, és a szeletek kiértékelése is szubjektív. Annak érdekében, hogy ezt a kiértékelést felgyorsítsuk, illetve objektivizáljuk és kvantitatívabbá tegyük, célul tűztük ki, hogy a szeletek kiértékelését szegmentációs hálók használatával automatizáljuk. Ezáltal a betegségek mechanizmusainak és a kezelési módoknak vizsgálata is meggyorsítható lenne.

Az annotálás során négy területet kell azonosítani: magát a szívszeletet, a kamrát, valamint a két legfontosabb területet, a rizikó és infarktus területét.

A rizikóterület a szívizom megcsappant vérrellátású és ezáltal oxigénhiányos része. Az infarktus során ennek egy része reverzibilis, másik része pedig irreverzibilis károsodást szenved, de ezen károsodás mértéke számos faktortól függ. Az infarktusos terület a myocardium elhalt része [2].

Az 1.1. ábrán látható, hogy az annotálás során több artefaktot is fel kell ismerni - például a szelet oldalán (jelen esetben a bal felső részen) megfigyelhető folyadékot, ami a szelet két üveglap közé szorítása közben gyülemlett fel. Ez nem tartozik a szívszelet-

hez. Ezeket az artefaktokat néha emberi szemmel is nehéz megkülönböztetni a tényleges szívzszelettől, és a hálónak is meg kell tanulniuk felismerni a különbséget.



1.1. ábra. Annotálandó szívzszelet és a hozzá tartozó területek maszkjai

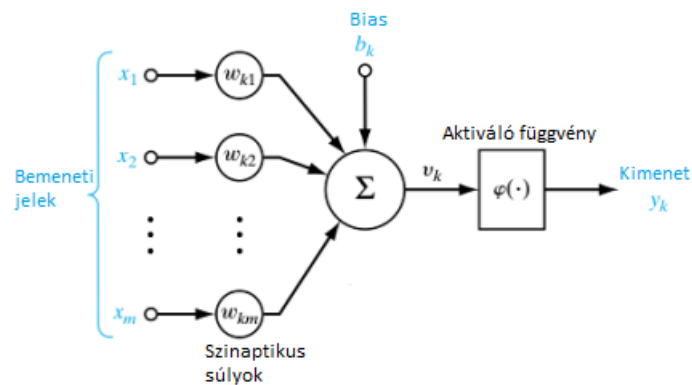
A cél eléréséhez több neurális hálót is tanítottam, melyek irodalmi háttere az alábbiakban bemutatásra kerül. Munkánk során beállítottam a tanítások megfelelő paramétereit, majd a tanított hálókat összehasonlítottam egymással, illetve a várt kimeneti értékekkel. Kiválasztva a leghatékosabb kimenetet, beállítottam egy értékelési rendszert is, mely előrejelzést ad az annotálás minőségéről. Végül teszteltem, hogy mekkora adatmennyiség szükséges a hálók tanításához, hogy kipróbáljam a rendszerünk robusztusságát.

2. fejezet

A konvolúciós neurális hálózatok bemutatása

2.1. A neurális hálózatok általános jellemzői

A neurális hálózat az emberi idegrendszer mintájára épülő, a mesterséges intelligencia egyik legnépszerűbb és legelterjedtebb változata. Egy gradiens alapú módszert alkalmazó tanuló algoritmról van szó, ami több különböző, egymással kapcsolatban álló rétegből és neuronokból (perceptronokból) áll.



2.1. ábra. Egy perceptron felépítése Zarándy Ákos "Neural Networks" előadása nyomán [3]

A perceptron a 2.1. ábrán látható módon a bemeneteket (ez lehet az eredeti bemenet vagy az előző réteg kimenete) súlyozza, ezeket összeadja egymással és egy úgynevezett bias értékkel, majd egy nemlineáris aktiváló függvény kimenete felhasználásával megalkotja a perceptron kimenetét.

A belső (rejtett, a bemeneti és kimeneti rétegek között elhelyezkedő) rétegek esetében

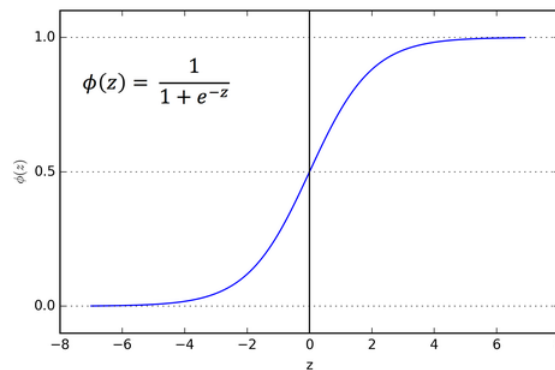
a legtöbbször használt aktivációs függvény a kiigazított lineáris egységfüggvény (Rectified Linear Unit, azaz ReLU). Ennek értéke [3]:

$$f(x) = \max(0, x) \quad (2.1)$$

A ReLU egy változata, a szivárgó (leaky) ReLU is használatos, amely $x = 0$ alatti értékek esetén sem nulla meredekségű, helyette az értékek valamilyen konstanssal szorzásra kerülnek. Ezáltal a függvény használata segíthet elkerülni az eltűnő gradiens problémát [4].

Az eltűnő gradiens azt jelenti, hogy sok réteg esetén a veszteség függvény gradiense nagyon kicsi, közel nulla, így a paraméterek lassan javulnak, és a tanítás sok időt és erőforrást vesz igénybe.

Például a 2.2. diagramon látható szigmoid aktivációs függvény is okozhat ilyen problémát, ez a függvény ugyanis a bemeneti adatok terét nulla és egy közé szorítja. Ez kis különbséget jelent nagy eredeti eltérések esetében, ezáltal a gradiens kicsi lesz [5].



2.2. ábra. A szigmoid aktivációs függvény [6]

Azt, hogy a háló tanul abból láthatjuk, hogy a veszteségfüggvény egyre csökken. Ez a függvény az elvárt kimenetet és a háló által adott kimenetet hasonlítja össze, és a veszteség értéke annál nagyobb lesz, minél nagyobb a kettő közti különbség [7].

A veszteség csökkentéséért a hálóparaméterek változtatásán keresztül az optimalizálók felelősek, amik ezt a feladatot hiba-visszaterjesztéssel végzik. A veszteségfüggvény gradiensvektora ennek során "visszaadódik" a kimenet irányából a bemeneti rétegek felé, és az optimalizálók a használt optimalizáló algoritmusnak megfelelő módon változtatják a paramétereket [8]. A változtatás mértéke a beállított tanulási rátától függ.

A ReLU rétegek előtt a tulajdonság-kinyerés (feature extraction) során konvolúciós rétegeket szoktak alkalmazni. Ezek szűrőként működnek [9]. A konvolúció egy adott bemeneti képen egy mag (kernel) használatával történik, eredménye pedig egy tulajdonság-

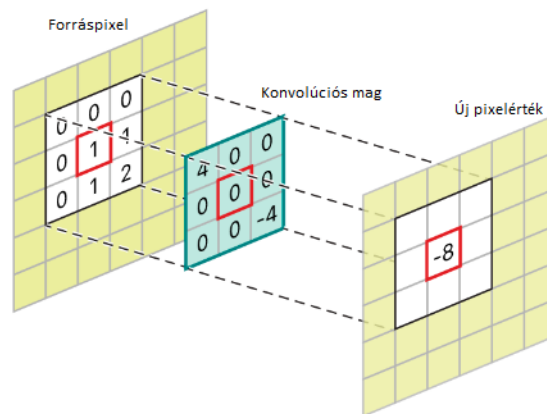
térkép (feature map) [10].

A (diszkrét) konvolúció művelete [9]:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \quad (2.2)$$

Ahogy az a 2.3. ábrán is megfigyelhető, a konvolúció során nem egyszerre történik a teljes bemeneti adat kiértékelése, hanem részleteiben.

Ezzel a művelettel az adathalmaz dimenziója csökkenthető, különböző jellemzők kiemelhetőek, és több mag alkalmazásával többféle jellemző kiemelése és ezáltal többféle feldolgozási csatorna létrehozása valósítható meg [10].



2.3. ábra. Példa a konvolúció műveletére az Apple Development dokumentációja nyomán [11]

A konvolúció során fontos paraméter a szűrő lépésköze és a kipárnázás - ez utóbbi azt jelenti, hogy a képhez értékeket toldunk (általában nullát) abból a célból, hogy a szűrő a szélső pixelértékekre is illeszthető legyen.

Egy másik fontos módszer az egyesítés (pooling), ami a felbontást azáltal csökkenti, hogy valamilyen módszerrel összegzi az adott helyről kinyert jellemzőket. Ez általában átlagoló vagy max egyesítés, mikor a pixelcsoport maximumát vesszük [9].

A tanítás során a bemeneti adatok normalizálása meggyorsíthatja és segítheti a tanulás folyamatát, ez történhet kötegenként. Egy köteg a tanító adathalmaz egy kisebb része, ami megadott számú képet tartalmaz. A tanítás során a háló kötegenként szokta feldolgozni a bemeneti adatokat.

Kötegelt normalizáció esetén az adott réteg bemeneteinek átlaga és variáciája kerül kiszámolásra, majd ezek felhasználásával történik a normalizáció, amit skálázás és eltolás követ [4].

2.2. A szegmentációs hálók

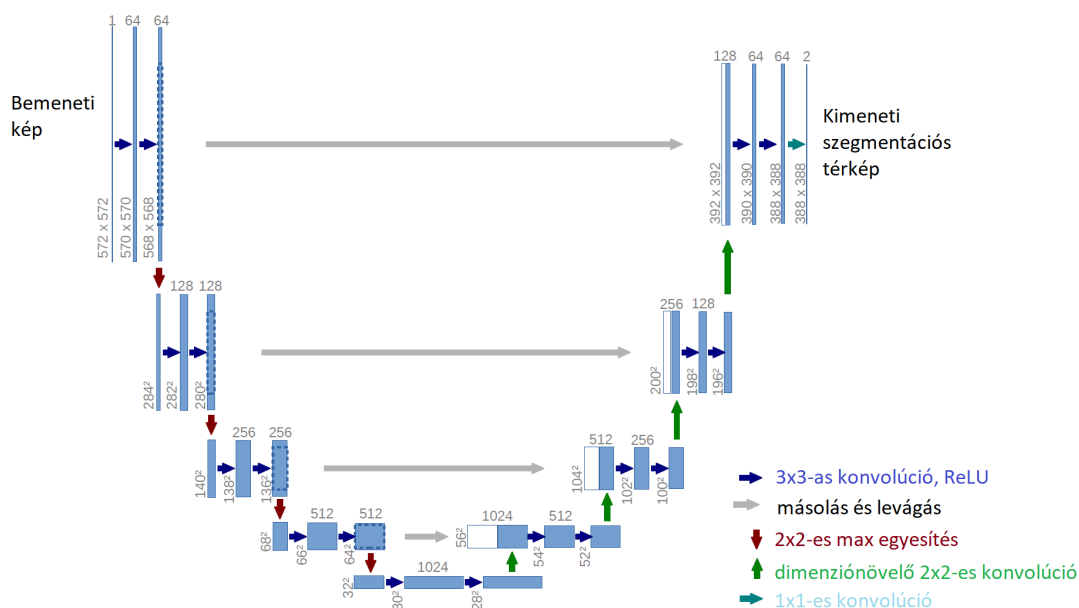
A neurális hálók többféle feladatra is felhasználhatóak. Klasszifikáció esetén a háló egy adott osztályt rendel a klasszifikálandó képhez, így a kimenete ennek az osztálynak a címkéje. Objektum detekció esetén a képen található különböző osztályba tartozó objektumok nem csak egy, az osztályhoz tartozó címkét kapnak, hanem egy úgynevezett határoló dobozzal (bounding box) is körül lesznek határolva. Ezáltal a pozíciójuk is jelzésre is kerül.

A harmadik fő típus a szegmentáció. Ekkor nem csak az objektumok osztályba sorolása történik meg, hanem egy úgynevezett maszk illesztése is azon pixelek fölé, amik az objektumot képezik. Szemantikus szegmentáció esetén az ugyanabba az osztályba tartozó objektumok megkülönböztetése nem történik meg, esetenkénti szegmentáció esetén azonban igen [12]. Az általam tanított hálók esetében ez utóbbira magától értetődő módon nincs szükség.

A hálók tanítása történhet felügyelt vagy felügyelet nélküli módon. Előbbi esetben a tanulás során a hálónak rendelkezésre állnak a megfelelő maszkok, amikkel összehasonlíthatja a saját kimenetét, utóbbi esetben azonban nem. Én felügyelt tanítást alkalmaztam.

2.2.1. U-Net

A U-Net az egyik legrégebbi és legismertebb szegmentációs háló, ami orvosi képek szegmentálására készült [13].



2.4. ábra. U-Net architektúra Derrick Mwiti és Katherine (Yi) Li cikke alapján [13]

Ahogy a 2.4. képen is látható, a kódoló részen a 2x2-es max egyesítés minden blokk végén megfelel a dimenziókat, a feldolgozási csatornák pedig megkétszereződnek. A dekódolónál ennek az ellenkezője történik [13].

Minden kódoló blokk két 3x3-as (ez a mag mérete) konvolúciót és egy ReLU aktivációs függvényt alkalmaz. Ez utóbbi nemlinearitása segíti az általánosítást a hálózatban.

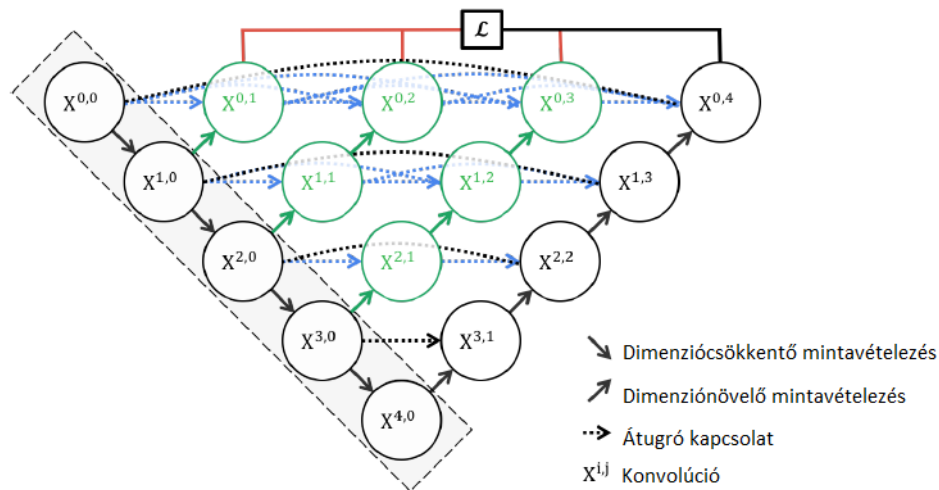
A kódoló és dekódoló részt egy 3x3-as konvolúcióból és ReLU függvényből álló híd köti össze [14].

A dekódoló minden szinten két bemenet kap, egyet az alatta lévő szinttől, egyet pedig a kódoló vele megegyező szintjétől [13]. 2x2-es transzponált konvolúcióval kezdődik, ami a dimenziókat növeli. Ezután történik a konkatenáció az ő szintjén lévő kódoló blokk kimenetével. Ezáltal olyan jellemzők is megmaradnak, amik alapvetően elvesztek volna. Ezt a kódoló részhez hasonlóan 3x3-as konvolúciók és ReLU függvények követik [14].

A dekódoló végén egy 1x1-es, szigmoid aktivációs függvényt megelőző konvolúció történik. A szigmoid aktivációs függvény kimenete a szegmentációs maszk [14].

2.2.2. U-Net++

A U-Net++ - ahogy az a nevéből is sejthető - a U-Net egy továbbfejlesztett változata. A U-Netben jelen lévő, kódoló és dekódoló közti pályákat átalakítja. Itt egy konvolúciós rétegekből álló konvolúciós blokk található ahelyett, hogy a dekódoló blokkjai egyszerűen csak a kódoló adott szintű blokkjának kimenetét kapnák meg [15].



2.5. ábra. U-Net++ neurális háló-architektúra sematikus bemutatása Liang-Chieh Chen és munkatársai cikke alapján [15]

A U-Net++ felépítését a 2.5. ábra szemlélteti. Minden konvolúciós réteget egy konkatenációs réteg előz meg, ami összeolvasztja a blokkban öt megelőző konvolúciós réteg

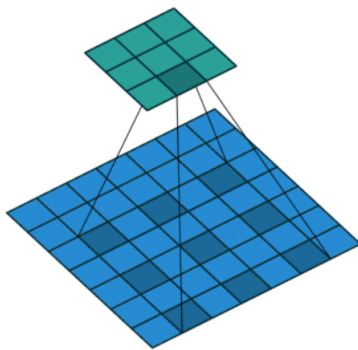
kimenetét az eggyel alatta lévő blokk megfelelő, nagyobb mintavételezésű (upsampled) réteg kimenetével. Ezáltal a kódoló tulajdonság-térképe szemantikusan közelebb kerül a dekódoló térképéhez, ami megkönnyíti az optimalizáló munkáját.

Mivel így több szemantikus szinten jön létre az eredetivel megegyező felbontású tulajdonság-térkép, a veszteségfüggvény értéke is mindezek felhasználásával kerül számolásra.

Amennyiben gyorsan akarjuk elvégezni a szegmentációt, csak egy adott ág szegmentációs térképét is figyelembe vehetjük, vagy átlagolhatjuk az ágak kimenetét, ezáltal pontosabb eredményt kapva [15].

2.2.3. DeepLabV3

A DeepLab egy olyan szegmentációs architektúra, ami a 2.6. képen látható, úgynevezett atrous konvolúciót valósítja meg. Ennek során a mag méretét úgy növelik, hogy az a valódi értékek között kipárnázásra kerül. Így a felbontás sem csökken, a látótér szélesebb lesz, mint párnázás nélkül, de a művelet elvégzéséhez szükséges erőforrások mennyisége nem nő [16].



2.6. ábra. Atrous konvolúció [16]

Az architektúra ezen kívül egy klasszifikációs, mély konvolúciós hálózaton (adott modelttől függően például VGG16 vagy ResNet101) alapul, ami átalakításra kerül úgy, hogy a teljesen kapcsolt rétegeket konvolúcióssá alakítják, ezáltal egy teljes mértékben konvolúciós hálót hozva létre [17].

2.2.4. A U-Net, U-Net++ és a DeepLabV3 viszonya

Az említett architektúrák már többféle szegmentációs feladat keretein belül is összehasonlításra kerültek, általában azzal az eredménnyel, hogy a U-Net nem jelentősen, de rosszabb teljesítményt nyújtott a többinél.

Például a sejtmag, bélpolip, tüdőcsomó és máj szegmentációjában a Jaccard indexet vizsgálva a U-Net mindig 2-5 %-kal alulmaradt a U-Net++-szal szemben. A sejtmag esetén utóbbi Jaccard indexe 92.63 %, míg előbbié 90.77 % volt, a májnál pedig 79.7 % és 76.62% voltak az értékek [15].

A DeepLabV3 és a U-Net felülnézetes képeken lévő emberek szegmentációjában kerültek összehasonlításra. Itt a DeepLabV3 Jaccard indexe 86 %, a U-Neté pedig 84 % volt [18].

Ezek alapján mi is arra számítottunk, hogy a hálók körülbelül ugyanolyan teljesítményt fognak nyújtani egy esetleges kis eltéréssel a DeepLabV3+ és a U-Net++ javára.

3. fejezet

A neurális hálók tanítása és tesztelése

3.1. Adatok előfeldolgozása

A tanításhoz használt szív szelet képek és a hozzájuk tartozó maszkok a Semmelweis Egyetem Farmakológiai és Farmakoterápiai Intézetének korábbi munkája során készültek. A képeket nyersen nem lehetett felhasználni a hálózatok tanítására, szükség volt ezek átalakítására.

A képeket tartalmazó numpy array-eket dr. Horváth András témavezetőm hozta létre korábbi munkája során, a már rendelkezésre álló képekből. Az array-ek 1182 darab, 224 x 224-es képet tartalmaztak, és mindegyikhez hét csatorna tartozott. Az első három csatorna a piros, a zöld és a kék színcsatornát, a maradék négy pedig a kívánt kimeneti maszkokat tartalmazta.

A tanításhoz tíz képet leszámítva minden adatot felhasználtam, így maximalizálva a tanítóhalmazt, azonban korlátozva a teszteléshez felhasználható képek számát.

Mivel bináris szegmentációról volt szó, a maszkhoz tartozó pixeleknél 1, a többi pixelnek pedig 0 értéket kellett felvennie, de mivel úgy vettem észre, néhol a képek tartalmaztak ezektől eltérő értékeket, írtam egy függvényt, amivel minden 0-nál nagyobb értéket 1-re váltottam át, és ezeken a maszkokon tanítottam a hálókat.

A bemeneti képek eredeti pixelértékeit normalizáltam, tehát leosztottam 127.5-tel és kivontam belőle 1-et, ezáltal -1 és 1 közé szorítva a pixelek értékeit.

Adataugmentációt is használtam, pontosabban véletlenszerű tükrözést és forgatást. Ezzel a hálók egy nehezebb tanulási feladatot kaptak.

Abból a célból, hogy a dataloaderbe tudjam tölteni a képeket és a maszkokat a ta-

nításhoz saját osztályt hoztam létre, ami a képeket és az elvárt kimeneti maszkokat tartalmazta. A tanítás során ennek a metódusaival történt meg a képek és maszkok lekérése, valamint a transzformációk alkalmazása. A teszt adathalmazhoz egy majdnem ugyanilyen osztályt hoztam létre, viszont abból kihagytam a transzformációkat.

3.2. A hálók tanítási paramétereinek és jellemzőinek beállítása

A hálók hatékonysága nem csak a használt architektúrától, hanem a tanítás során használt egyéb paramétereiktől is erősen függ. Ezért, hogy a legjobb teljesítményű hálókat alkalmazzam a feladat megoldása során, többféle optimalizálót, tanulási rátát és veszteségfüggvényt is kipróbáltam.

Munkámat elsősorban a Google Colaboratoryban [19] végeztem. Ez a Google egy felhő alapú szolgáltatása, ami lehetőséget nyújt GPU-hoz való csatlakozásra és ezáltal képek gyorsabb feldolgozásására. Később az egyetem Információs Technológiai és Bionikai Karjának laborját is igénybe vettem a tanításhoz.

Többféle architektúrával is kísérleteztem, végül három, már korábban részletezett felépítésű hálót - egy U-Net, egy U-Net++ és egy DeepLabV3+ architektúráját - tanítottam.

3.2.1. U-Net

A munkám kezdetekor témavezetőmtől, dr. Horváth Andrásztól kaptam meg a már annotációra használt U-Net kódot, ami a tradicionális U-Nettől eltérően ReLU helyett LeakyReLU függvényt alkalmazott.

Az eredeti háló az 512 feature map-et elérve már megkezdte a dekódolást, ezt én egy szintet hozzáadva 1024-re növeltem, valamint a dekóderben használt Upsample függvényt ConvTranspose2d függvényre (ahogy azt a neve is sejteti, transzponált konvolúciót valósít meg, míg az upsampling csak a dimenzió megkétszerezését végzi) cseréltem.

A kódoló részben a blokkok végén kétdimenziós konvolúció helyett max egyesítést alkalmaztam, ezáltal közelebb hozva a háló architektúráját a "hagyományos" U-Net architektúrához.

Kétféle veszteségfüggvényt is kipróbáltam, MSELost és BCEWithLogitsLoss.

Az MSE - vagyis négyzetes hiba - minden adatpontra (pixelre) kiszámolja az elvárt kimenet és a háló kimenete közti különbséget, majd négyzetre emeli. Ezek az értékek a

teljes képre vonatkozóan aztán átlagolásra kerülnek [20]:

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2 \quad (3.1)$$

A képletben n az adatpontok számát, x az elvárt, y pedig a háló kimenetének értékét jelöli.

A BCEWithLogitsLoss a BCE-t (bináris kereszt-entrópia) és a szigmoid aktivációs függvényt kombinálja. Általában ez előbbi szokás bináris szegmentáció esetén használni, ami itt is történik minden maszk esetén [21].

A 3.1. táblázatban láthatóak a tanítás és a minden epoch után, tíz képen történt tesztelés veszteség értékei.

3.1. táblázat. Veszteségértékek eltérő veszteségfüggvények esetén

	Tanítás (átlag)	Tanítás (min.)	Teszt (átlag)	Teszt (min.)
MSE	0.027	0.023	0.018	0.026
BCEWithLogits	0.086	0.075	0.093	0.059

Amint látható, az MSELoss esetében az értékek 0.02-0.03 környékére esnek, a BCEWithLogitsLoss esetén pedig 0.08-0.09 köré. Mivel ekkor még nem használtam a veszteségfüggvényen kívüli metrikákat, a kimeneti maszkokat megvizsgáltam.

Tekintve, hogy eltérő veszteségfüggvényekről volt szó, amik eltérő módon és ezáltal eltérő értékeket számolnak ki, csak azért, mert az egyik kisebb értékű, nem feltétlenül jelenti azt, hogy valóban jobban teljesít, mint a többi.



3.1. ábra. Teszthalmaz rizikóterületeinek maszkjai

A 3.1. ábrán láthatóak a különböző veszteségfüggvényt használó hálók kimeneti maszkjai a rizikóterület esetében. Mint látható, bár nem a BCEWithLogitsLoss rendelkezett kisebb értékekkel, a másik veszteségfüggvény esetében a háló rosszabb eredménnyel

szolgált. Ez jellemző volt az összes tesztalmazbeli képre (az MSELosst használó háló csak a szívseletek maszkját találta el), ezért a BCEWithLogitsLoss használata mellett döntöttem.

A kötegméretet 8-ra állítottam. Próbáltam ennél nagyobb is, de technikai limitációk miatt ez nem volt megvalósítható.

Az optimalizáló esetében kipróbáltam a sztohasztikus gradiens süllyedést (SGD) és Adam optimalizálót is. Utóbbi esetében a tanulási ráta adaptív, tehát az SGD-vel ellentétben változik a tanulás során [22] .

Elsőként 0.001-es és 0.0001-es tanulási rátával próbáltam ki mind a sztohasztikus gradiens süllyedést, mind az Adam optimalizálót. Előbbi esetén 0.9-es momentumot használtam, amely módszer abban nyújt segítséget, hogy az optimalizáló ne "ragadjon be" egy lokális minimumba.

Ahogy az a 3.2. táblázatban látható, az Adam optimalizáló a tanulás és a tesztelés során is jobb (kisebb) veszteségértékekkel rendelkezett mind átlagban, mind pedig a legkisebb elért veszteségértéket tekintve, mindkét tanulási ráta esetén.

3.2. táblázat. Veszteség értékek 0.0001-es és 0.001-es tanulási ráta esetén

Tanulási ráta	0.0001				0.001			
	Tanítás		Teszt		Tanítás		Teszt	
	Átlag	Min.	Átlag	Min.	Átlag	Min.	Átlag	Min.
Adam	0.158	0.075	0.146	0.068	0.092	0.075	0.093	0.061
SGD	0.320	0.210	0.299	0.198	0.151	0.105	0.135	0.095

Mivel a 0.001-es tanulási ráta esetén az eredmény jobbnak bizonyult, mint a 0.0001-es esetén, tovább növeltem az értékét 0.006-ra, azonban csak minimális javulás volt megfigyelhető (3.3. táblázat), így ezt a tanulási rátát használtam a végleges hálók tanítása során.

3.3. táblázat. Veszteségértékek 0.006-os tanulási ráta és Adam optimalizáló esetén

Tanítás		Teszt	
Átlag	Min.	Átlag	Min.
0.086	0.075	0.093	0.059

3.2.2. U-Net++

A háló tanításához a segmentation_models.pytorch könyvtárat [23] használtam, ami már betanított architektúrákat tartalmaz. A kódolóban a gerinc ResNet34 architektúra,

a paraméterek pedig az ImageNet adathalmazon tanított háló súlyai voltak.

Mivel ez a háló kevesebb helyet foglalt a memóriában, itt a Colaboratoryban való tanítás során tudtam 32-es kötegméretet alkalmazni, azonban az ITK laborban nem fért el nyolcnál több képet tartalmazó köteg a memóriában. Végül a 8-as kötegmérettel tanított hálót használtam a mélyrehatóbb tesztelésnél.

Veszteségfüggvényként a U-Nethez hasonlóan BCEWithLogitsLoss-t használtunk, valamint Adam optimalizálót 0.006-os tanulási rátával.

3.2.3. DeepLabV3+

DeepLabV3-mal kezdtem a tanítást, utána azonban átváltottam a jobb teljesítményű, DeepLabV3-ból továbbfejlesztett DeepLabV3+-ra. A kódolóban gerincként itt is ResNet34 architektúrát használtam az ImageNet adathalmazon tanított háló paramétereivel. Kötegméretként szintén 32-t, majd a laborban 8-at használtam. A háló veszteségfüggvénye, optimalizálója és tanulási rátája megegyezett a korábbi két hálóéval.

3.3. Hálók tanítása és előzetes tesztelése

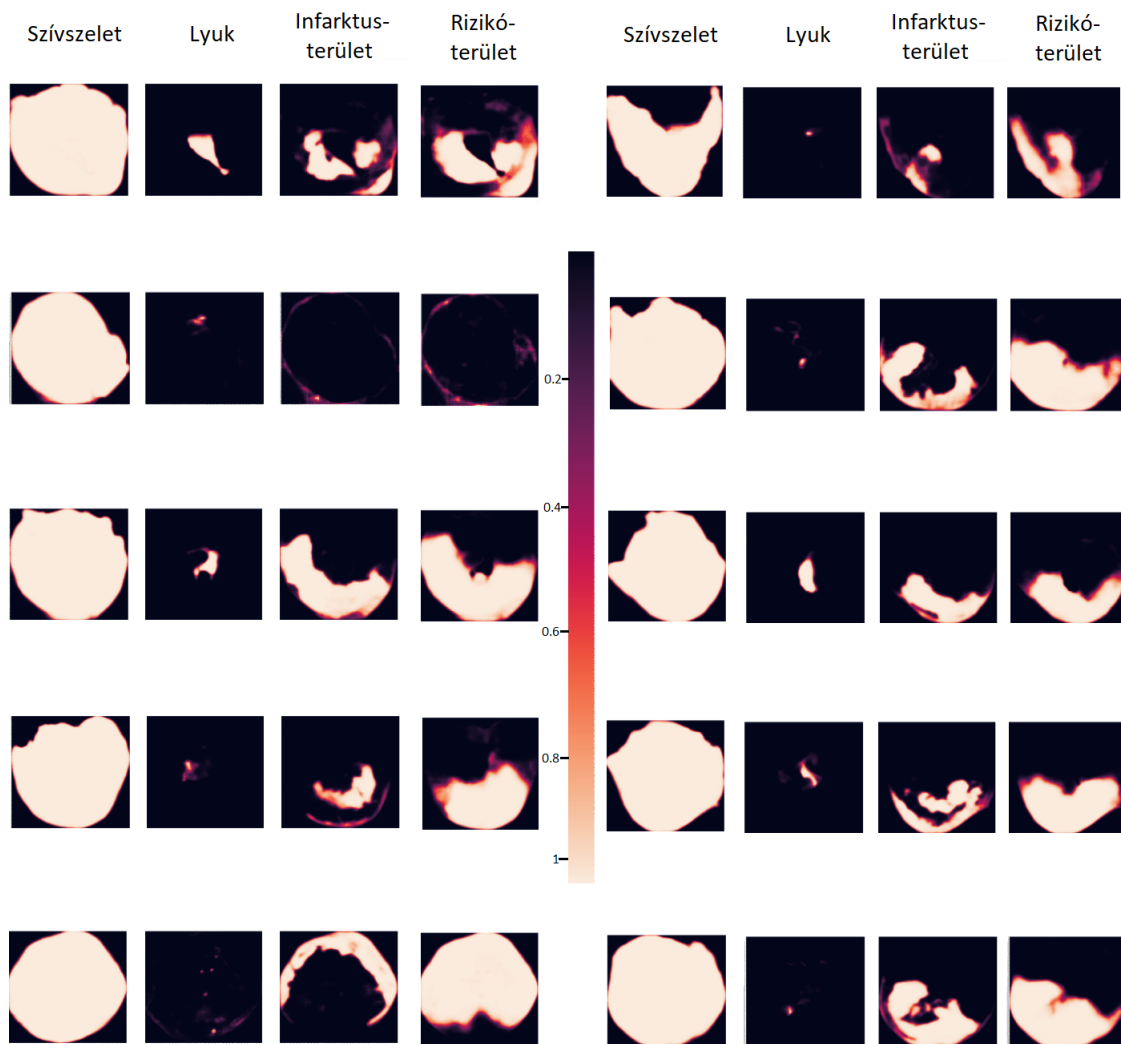
Elsőként harminc epochon keresztül tanítottam a hálókat, és amennyiben a veszteség csökkent, elmentettük a paramétereket. Amennyiben a veszteség öt epochon keresztül nem javult, befejeztem a tanítást, majd betöltöttem a háló legjobb paramétereit. Ezt követően teszteltem a hálókat a tíz félretett képen, majd mivel a kirajzolt maszkok alapján úgy gondoltuk, a tanítás eredményes volt, tovább tanítottam a hálókat még kétszer harminc epochon keresztül. Bár sikerült javulást elérnem, a kirajzolt maszkok alapján az nem volt jelentős.

A neurális hálók túltaníthatóak, ekkor nem a különböző területek általános tulajdonságait tanulják meg, hanem a tanulóhalmaz képeit, ezért láttuk szükségesnek a hálók előzetes tesztelését. A tíz, tesztelésre félretett képet egyenként, transzformálás nélkül értékeltem ki az összes hálóval, majd a hálók kimeneteit átlagoltam és ennek minőségét is teszteltem, melyet kombinált kimenetnek neveztünk el.

Mivel a hálók kimenetei nem csak 0 és 1 értékeket tartalmaztak, hanem negatívakat és pozitívakat is, egy küszöbérték-függvényt alkalmaztam, ami az előbbieket 0-ra, utóbbiakat pedig 1-re állította át. Mivel BCEWithLogitsLoss függvényről volt szó, minél negatívabb volt az eredeti érték, annál biztosabb volt a háló abban, hogy háttéri pixelről volt szó, és minél pozitívabb, annál valószínűbbnek tartotta, hogy a pixel a maszkhoz tartozott.

Később az ITK laborjában folytatott tanítás a U-Net esetében 180, a másik két háló esetében 380 epochon keresztül folyt - az előre tanított modellek esetében a tanítás jelentősen kevesebb időt vett igénybe. A hálók paramétereit ebben az esetben akkor mentettem el, ha javult a teszt-veszteség értéke, amit minden tanítási epoch után megvizsgáltam.

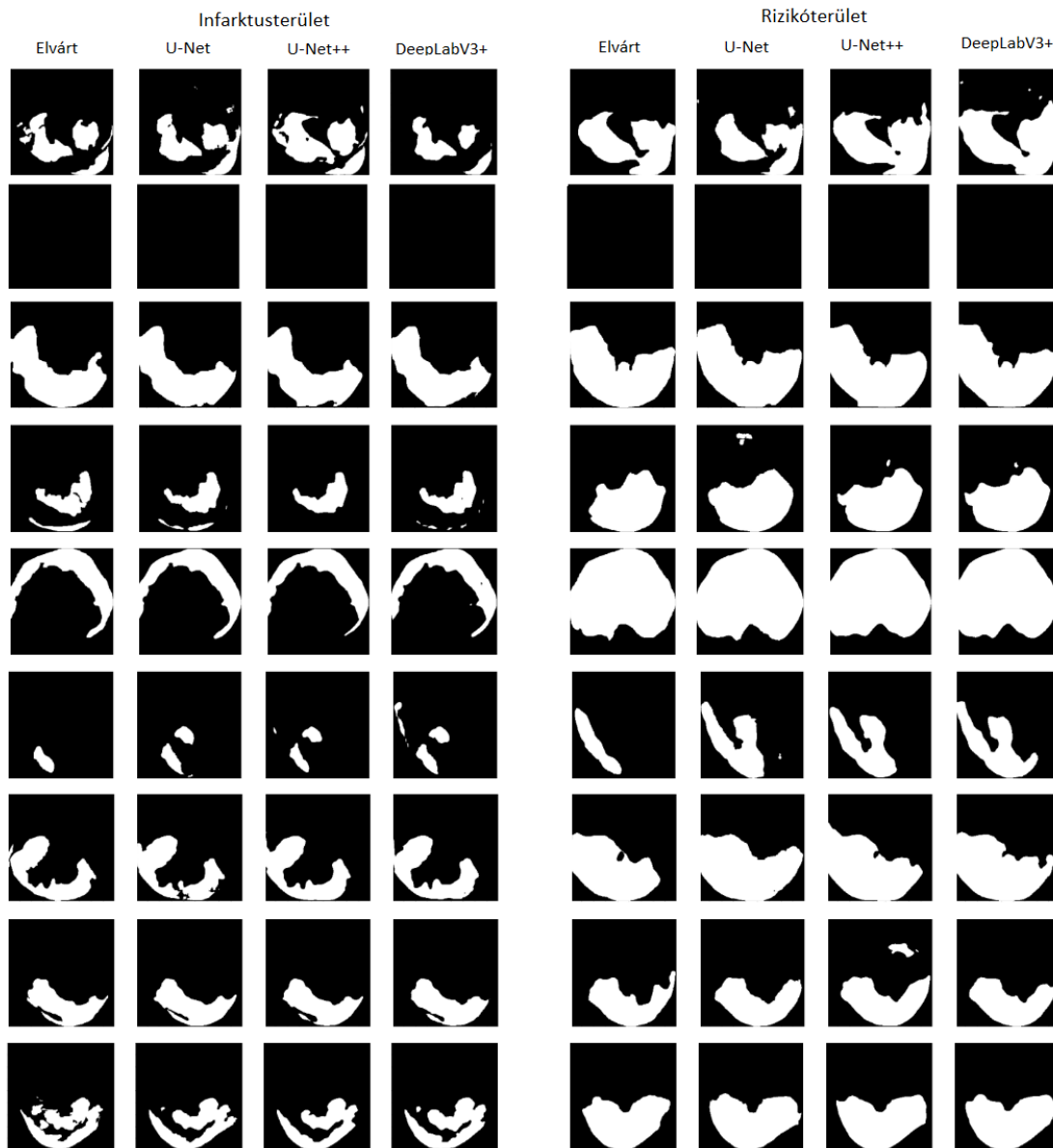
A teszhalmaz valószínűségi értékeit a 3.2. ábrán látható hő térképen is kirajzoltam, hogy megnézzük, esetleg érdemes-e változtatni a küszöb értékén. Úgy találtuk, hogy az esetek többségében a hálók egész biztosak voltak abban, ha valami a maszkhoz tartozott, így a nullás határnál maradtunk.



3.2. ábra. Teszhalmaz maszkjainak hő térképe a laborban tanított hálók kombinált ki-menete esetén, az értékek valószínűségben értendők

A 3.3. ábrán látható, hogy a hosszabban tanított hálók által majdnem az összes szívszelet helyesen szegmentálásra került, egynek az infarktus- és rizikóterületét azonban

mindhárom jelentősen eltévesztette.



3.3. ábra. Tízelemű teszhalmaz rizikó- és infarktusterületének maszkjai a laborban tanított hálók esetén

Mindezek alapján arra a konklúzióra jutottunk, hogy a hálók valószínűleg elfogadhatóan fognak teljesíteni új, nem ezen halmazba tartozó képeken is, így belekezdtem a hálózatok mélyrehatóbb tesztelésébe.

4. fejezet

A neurális hálók tesztelése nagy méretű tesztthalmazon

4.1. Tesztelés különböző metrikákkal

Hogy a hálók teljesítménye számszerűsíthető legyen, és ezeket a teljesítményeket képesek legyünk összehasonlítani, szükségünk volt egy metrikára. Többet is kipróbáltam, és azt is megvizsgáltam, milyen előnyökkel és hátrányokkal járna a használatuk.

Annak céljából, hogy a túl kevés kép használata ne torzítsa az eredményeket, a hálók hatékony teszteléshez újabb 82 tesztképet vontunk be a projektbe. Mivel a képeken általában hat-hét szívszület volt látható, a képeket egy, a témevezetőmtől kapott kód segítségével feldaraboltam és átméreteztem 224 x 224-es méretre, hogy a hálónk tudjanak velük dolgozni. Így végül 496 szeletképet kaptam.

Ezen képek esetében a szeleteket első lépésben a korábban tanított neurális hálónk elő-annotálta, majd ezeket a kiértékeléseket javította a kiértékelő személy. A végleges maszkokat .json formátumban exportáltuk, ami az eredeti, több szívszületet tartalmazó képenként tartalmazta a maszkok körvonalának x és y koordinátáit. Ezért a koordinátákat képenként kiolvastam a fájlból, majd egy, az eredeti képpel megegyező fekete képet hoztam létre, amin a koordinátáknak megfelelő részeket fehérre színeztem. Ezeket végül ugyanúgy feldaraboltam, mint az eredeti képeket.

Néhány kép kirajzolása és a hálókön való tesztelés után azt figyeltem meg, hogy a referenciamaszkok esetenként hibásnak tűntek, azonban ez elenyésző esetben fordult elő, így ennek hatását elhanyagolhatónak tekintettük.

A hálók összehasonlításához megvizsgáltam a Jaccard indexet (JI), mint felhasználható metrikát. Ehhez a tényleges kimenet és az elvárt kimenet metszetének méretét, majd

az uniójuk méretét kell kiszámolni, a JI pedig ezek hányadosa [24]:

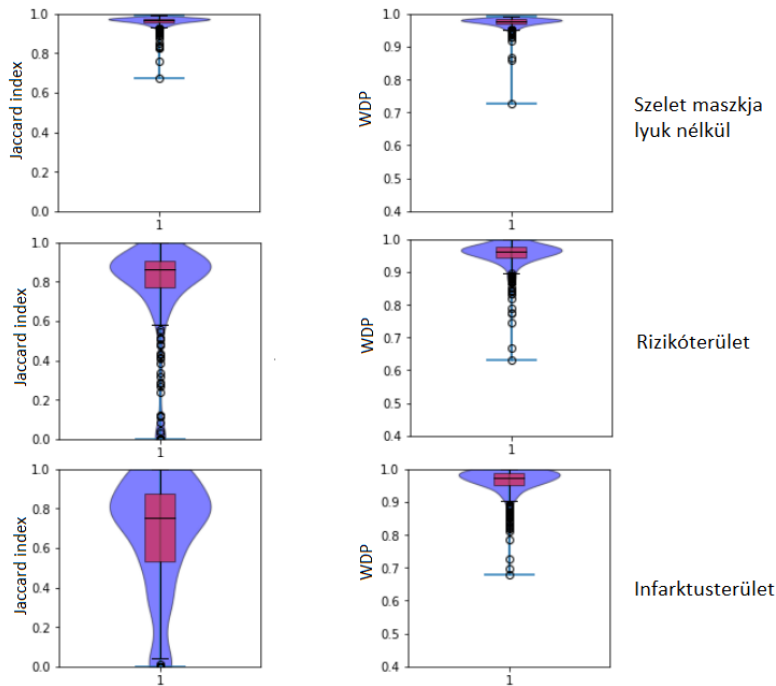
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (4.1)$$

Sajnos kisméretű maszkok esetén ez az index a hibát felnagyítja, hiszen ekkor már a gyakorlatban elhanyagolható eltérés mérete is a maszk méretének nagy hányadával egyezik meg, ezért egy másik metrika használatának gondolata is felmerült.

Kiszámoltam, a pixelek hány százalékát sikerült sikeresen eltalálnia a hálónak, beleértve a háttérpixeleket is. Ezen metrikára a továbbiakban WDP-ként, azaz well-defined pixelsként fogok hivatkozni. Mivel a szeletet és a kamrát lefedő maszkok külön-külön nem, csak utóbbit az előbbiből kivonva (vagyis azt a részt megtartva, ahol tényleges szív-szövet található a képen) hordoznak fontos információt, ezért ezen területek esetében ezt az értéket vizsgáltam.

Ezen metrika hátránya szintén a kisméretű maszkok esetében mutatkozik meg: még ha a hálók teljes mértékben el is vétették a maszk bejelölését vagy minden pixelt a háttérhez soroltak, a nagy háttérterület miatt a WDP értéke így is magas maradt.

A két metrika közötti eltérés időnként igen jelentős volt. Előfordult, hogy a JI mindössze 0.3 körüli értéket adott, míg a WDP 0.9 fölött volt. A két metrika eltérése főként a rizikó és infarktus maszkok esetén volt jelentős, hiszen ezek sokkal kisebb régiók, mint a teljes szív-szelet, és sokkal kevésbé különböznek a szövet többi részétől, mint a kamra lyuka.



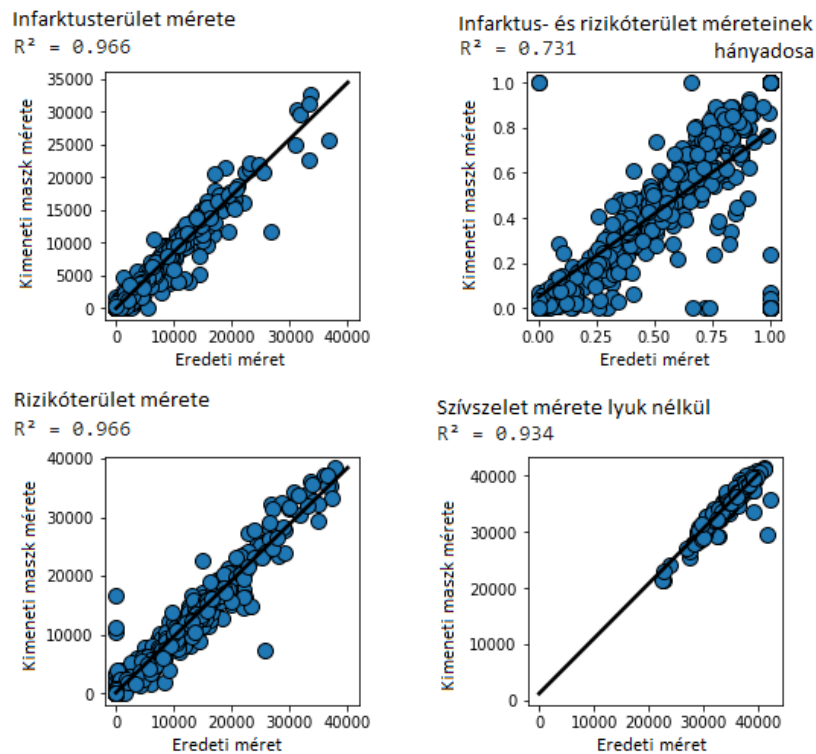
4.1. ábra. JI és WDP eloszlása különböző maszkok esetében a kombinált kimenetnél

Az értékek eltérését szemlélteti a 4.1. ábra, melyen violin és box plotok segítségével vizualizáltam a kombinált kimenetnél kapott értékeket.

A violinpot esetében minél szélesebb a tengely egy adott pontjánál a plot, annál gyakrabban van jelen az adott érték, így megmutatja az adatok eloszlását. A box plot a minimum, maximum, medián, első és harmadik kvartiliseket tünteti fel, valamint az outlier értékeket [25].

Mivel a szívszeletek kiértékelése során a végső eredményeket a területek méretéből számoljuk, így kiemelkedően fontos, hogy a háló milyen mértékű eltérést mutat ebben a metrikában a humán kiértékelőhöz képest.

Ennek vizsgálatához grafikusán ábrázoltam az elvárt és a kimeneti maszk méretét egy koordinátatengelyen, majd lineáris regresszióval egyenest illesztettem az adatokra. Ennek során a 224x224-re átméretezett képekre illesztett maszkok méretét vettem figyelembe.

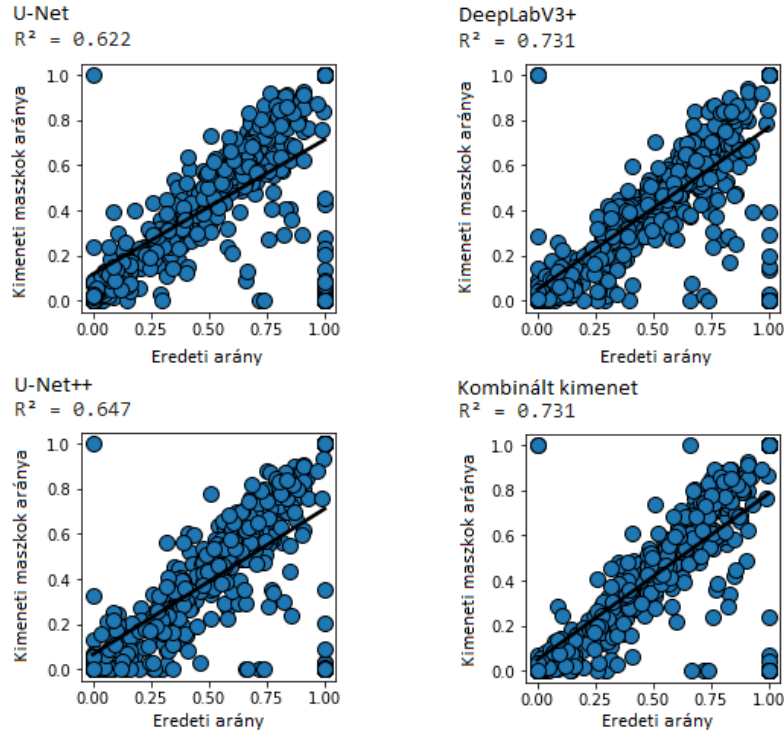


4.2. ábra. Lineáris regresszió a területek maszkjainak méretére a kombinált kimenet esetén

A 4.2. ábrán látható R^2 érték a korrelációs együttható, és amint látszik, a legtöbb esetben a tényleges és az elvárt méretek között nagymértékű korreláció áll fenn. Az együttható maximum +1-es értéket vehet fel, ez utóbbi esetben teljes korreláció van a két érték között.

A nagy tesztalmaz miatt voltak kilógó értékek. Ez főleg az infarktus- és rizikóterü-

letek hányadosának esetében volt jelentős, mely érték a vizsgálat elsődleges végpontja. Ebben az esetben, ahogy az a 4.3. ábrán is látszik, minden hálónál az R^2 érték 0.6-0.7 körüli volt a többi maszktípus 4.2. ábrán látható 0.9 körüli értékeivel ellentétben.



4.3. ábra. Lineáris regresszió az infarktus- és rizikóterületének hányadosára a különböző hálók esetén

Ez valószínűleg annak volt köszönhető, hogy csupán az egyik érték nagy eltérése a várttól is elegendő ahhoz, hogy az arány értéke is torzuljon. Az együtttható értéke, mint az ábrán is látható, kissé javult a kimenetek kombinálása esetén, de még mindig nem közelítette meg a többi terület R^2 értékét.

Mindent összevetve végül arra a döntésre jutottunk, hogy mind a JI-t, mind a WDP-t használni fogjuk a további mérések során, remélve, hogy kompenzálni fogják egymás hibáit.

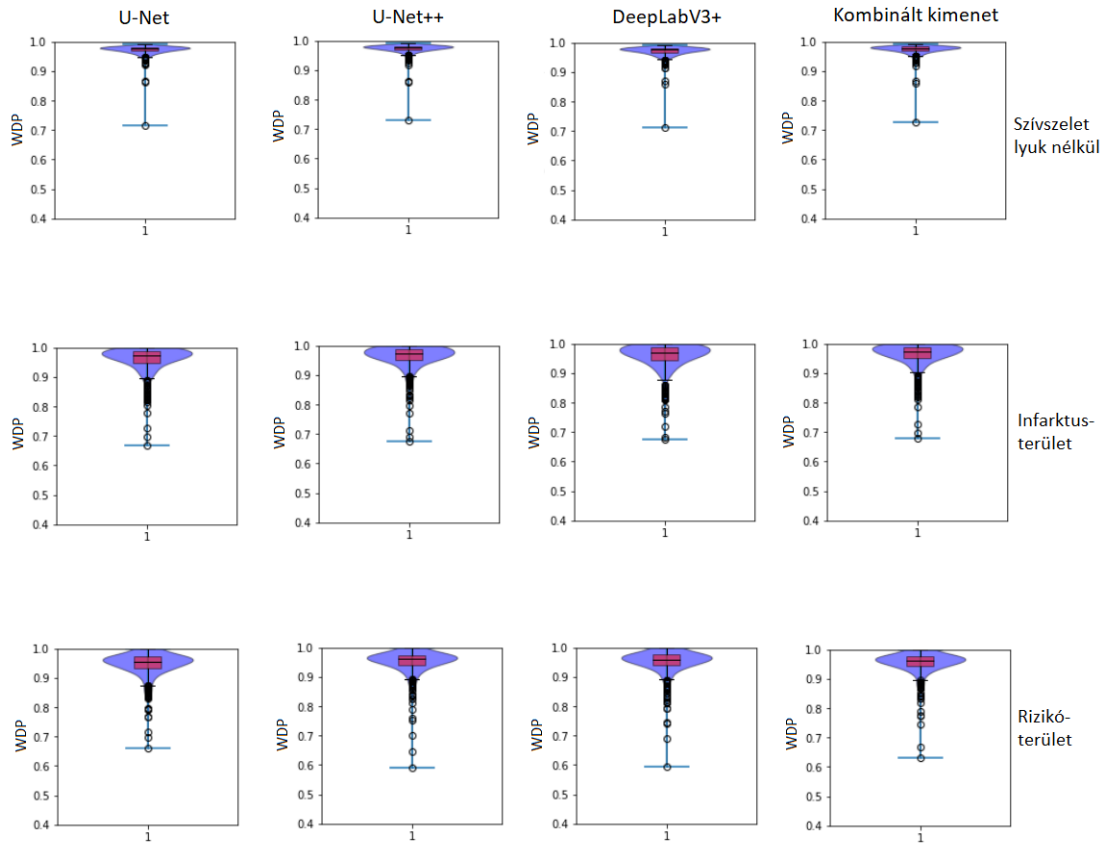
4.2. A tanított hálók hatékonyságának összehasonlítása

Mivel több háló is a rendelkezésünkre állt, a leghatékonyabbat kívántuk kiválasztani és felhasználni mind a jövőbeni mérések, mind az esetleges új szeletek kiértékelése során.

A három hálót és ezek kombinált kimenetének hatékonyságát összehasonlítottam a már említett violin és box plotok segítségével. Az összehasonlításhoz először a WDP-t

választottuk, mert úgy gondoltuk, hasznos, hogy ez nem csak a maszkhoz tartozó, hanem a háttérpixeleket is figyelembe veszi, valamint a JI-vel ellentétben nem nagyítja fel a kisebb hibákat.

A hálók a 4.4. ábrán is látható módon nagyon hasonlóan teljesítettek minden területen, ahogy azt vártuk.



4.4. ábra. A tanított hálók hatékonyságának összehasonlítása

Később Kovácsházi Csenger témavezetőm a Jaccard indexet is megvizsgálta. A kombinált kimenet átlaga enyhén magasabb volt, mint az egyedi hálóké. Emellett a minimum értékek is a legtöbb esetben nagyobbak voltak a kombinált kimenet esetében (4.1. táblázat).

Emiatt úgy döntöttünk, hogy a továbbiakban a kombinált kimenetet használjuk, de a tény, hogy mind a három háló elég hasonló eredményeket szolgáltatott, azt mutatja, hogy a szegmentációs feladatot megoldó hálók tanítása jól reprodukálható. Ebből feltételezzük, hogy nagy eséllyel a hálók paramétereinek értéke nem csak egyszeri véletlen miatt állt be a hatékony szegmentáláshoz szükséges értékre, hanem a feladat valóban jól megoldható szegmentációs neurális hálók használatával.

4.1. táblázat. Hálók Jaccard indexeinek átlaga és minimuma a különböző maszktípusokra vonatkozóan

Maszktípus	Architektúra	Átlag $\pm$ Szórás	Min.
Rizikóterület	Kombinált	0.955 $\pm$ 0.040	0.630
	DeepLabV3+	0.952 $\pm$ 0.042	0.593
	U-Net++	0.951 $\pm$ 0.043	0.590
	U-Net	0.946 $\pm$ 0.043	0.659
Infarktusterület	Kombinált	0.963 $\pm$ 0.041	0.679
	U-Net++	0.962 $\pm$ 0.041	0.675
	U-Net	0.961 $\pm$ 0.043	0.670
	DeepLabV3+	0.956 $\pm$ 0.047	0.677
Szelet lyuk nélkül	Kombinált	0.974 $\pm$ 0.017	0.726
	U-Net++	0.974 $\pm$ 0.017	0.731
	U-Net	0.972 $\pm$ 0.017	0.717
	DeepLabV3+	0.972 $\pm$ 0.018	0.711

4.3. Scoring mechanizmus a háló kiértékelésének minőségére

Annak érdekében, hogy a humán kiértékelő könnyedén megállapíthassa, hogy mely szeletek esetében szükséges manuális korrekció, a kiértékelésre egy scoring mechanizmust is beállítottunk, mely egyfajta minőségi értékként megmutatja, mely maszkok tartalmazhatnak jelentős hibát.

A score értéket szeletekre, területenként számoltam ki. Használhatóságának tesztelése ROC görbe illesztéssel, lineáris regresszióval és a korrelációs együttható vizsgálatával történt. Mivel az értékre azért van szükség, hogy a programot használó kutató visszajelzést kapjon arról, hogy az automatikusan annotált szeletek közül melyek esetén van szükség nagymértékű manuális korrekcióra, a score értéknek a kombinált kimenet JI-jével való korrelációját határoztam meg.

A ROC (Receiver Operating Characteristic) görbe alapvetően arra használatos, hogy a klinikai szenzitivitás és specificitás közti kapcsolatot bemutassa, ezáltal egy adott diagnosztikai teszt használhatóságáról szolgáltat információt. A görbén az 1-specificitás függvényében kell ábrázolni a szenzitivitást. Előbbi a fals pozitív (olyan egészséges egyének, akiknél a teszt pozitív lett) és az egészséges (beleértve a fals pozitív tesztek is) egyének számának hányadosa. Utóbbi pedig a valódi pozitív (olyan betegek, akiknél a

teszt ki is mutatta a betegséget) és beteg (beleértve a fals negatív tesztek is) egyének számának hányadosa [26]:

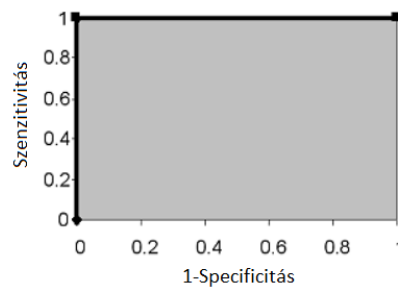
$$1 - spec. = \frac{FP}{FP + TN} \quad (4.2)$$

$$szenz. = \frac{TP}{TP + FN} \quad (4.3)$$

FP a fals pozitív, TP a valódi pozitív, TN a valódi negatív, FN pedig a fals negatív tesztek száma.

A görbe elkészítése során különböző határértékekre kell kiszámolni az 1- specificitást és a szenzitivitást (a határérték lehet például, hogy valakit akkor tekintünk betegnek, ha az adott betegséget jelző enzim koncentrációja egy adott érték alatt van), és a kiszámolt értékeket kell ábrázolni a koordináta-rendszerben [26].

A teszt annál jobb, minél nagyobb a görbe alatti terület, tehát akkor ideális, ha mind a specificitás, mint a szenzitivitás 1-es értéket vesz fel. A 4.5. ábrán látható, hogy ekkor minden betegnek pozitív a tesztje, és minden egészségesnek negatív, tehát nincsenek tévesen negatív vagy pozitív tesztek.



4.5. ábra. Ideális ROC görbe [26]

A teszt teljesen használhatatlan, ha a görbe a (0,0) és (1,0) pontokat összekötő egyenesre esik [26].

A scoring kiértékelése során a pozitív jelzőt azon annotációk kapták, amiknél a JI kisebb volt, mint 0.75. A teszt pedig az volt, hogy a vizsgált score értéke hogyan tudja megfelelően besorolni a vizsgált szeleteket.

Munkám során a kombinált kimenetekhez tartozó értékeket használtam.

4.3.1. Átlagos különbség

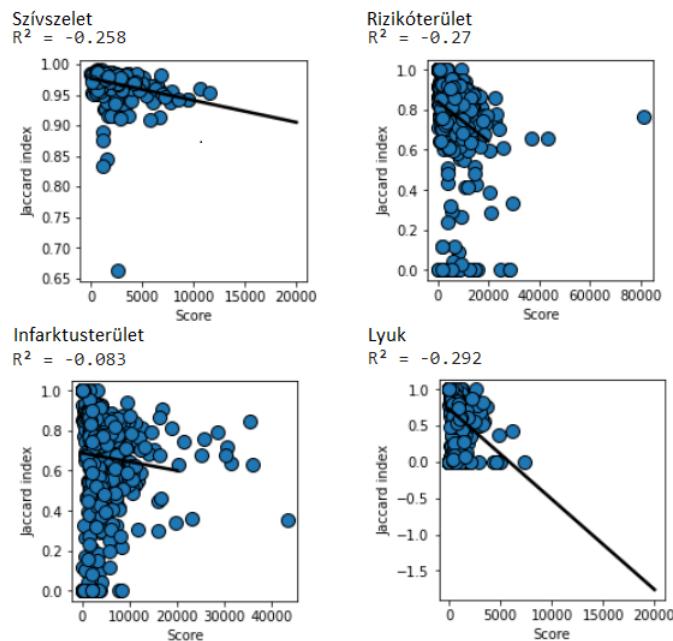
A score értéket először a három háló átlagos különbségeként számoltam ki. Ez azt jelenti, hogy minden szelet esetében minden területre megnéztem páronként a hálók által

szegmentált területek méretének különbségét, majd az így kapott három értéket átlagoltam.

Feltételeztük, hogy minél kisebb a score érték - és ezáltal az eltérés -, annál jobban teljesít a háló. Tehát amennyiben s a score, n_1 , n_2 és n_3 pedig rendre a U-Net, a U-Net++ és a DeepLabV3+ hálók ugyanahhoz a szelethez tartozó ugyanazon terület maszkjának területei:

$$s = \frac{|n_1 - n_2| + |n_1 - n_2| + |n_1 - n_2|}{3} \quad (4.4)$$

Már a lineáris regresszió is mutatja, hogy ez a score érték nem megfelelő a maszkok pontosságának mérésére (4.6. ábra). Mivel itt a nagyobb Jaccard indexhez kisebb score értéknek kellett tartoznia a korreláció meglétéhez, a negatív korrelációs együttható nem a korreláció hiányát, hanem ezt a fordított arányosságot jelzi. Ez alapján a korrelációs együttható az infarktusterület esetén volt a legrosszabb, a legjobb érték pedig csupán -0.292 volt, tehát semmiféle korreláció nem állt fenn a két érték között. Habár a score érték normálása, azaz nulla és egy közötti értéké szorítása javíthatott volna a hatékonyságon, az eredeti gyenge korreláció miatt erre nem került sor.



4.6. ábra. Átlagos különbségek score és JI lineáris regressziója

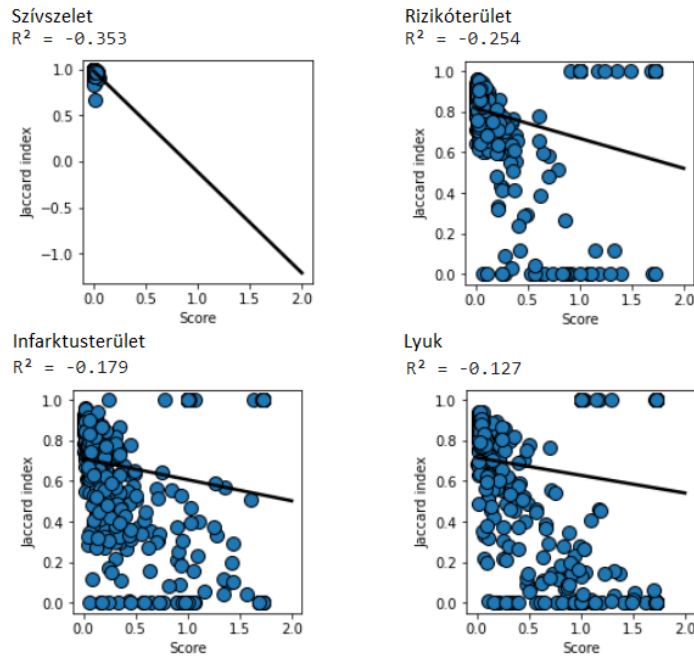
4.3.2. Szórás és átlag hányadosa

Minden szelet minden területéhez szórást számoltam az egyes hálók kimenetének maszkterületének méretéből, majd ezt az értéket elosztottam a területméretek átlagá-

val. Ezt az értéket variációs együtthatónak nevezzük, ahol δ a szórás, μ pedig az átlag:

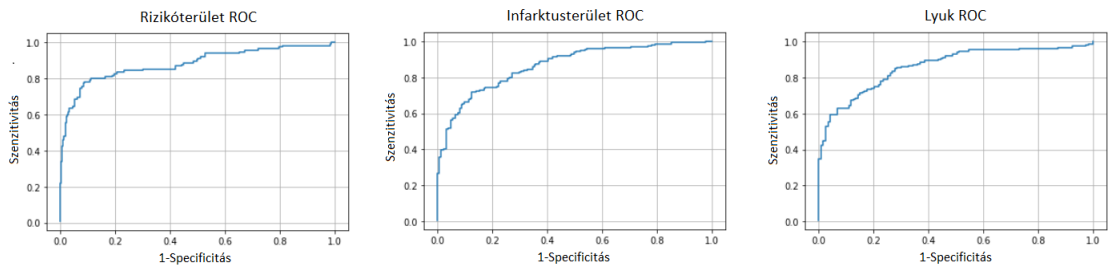
$$CV = \frac{\delta}{\mu} \quad (4.5)$$

Az értéket korreláltattam a JI-vel. Eredményeink gyenge korrelációt mutattak (4.7. ábra). Mivel itt is fordított arányosságnak kellett fent állnia a JI és a score között, a legrosszabb érték -0.127 volt a lyuk esetében, a legjobb együttható pedig -0.353 a szívzszeletnél.



4.7. ábra. Variációs együtthatók score és JI lineáris regressziója

A ROC görbék meg sem közelítették az ideális görbét, ezért ezt követően úgy is megvizsgáltam őket, hogy amennyiben a hálók kombinált kimenete egy üres maszk volt, tehát a szoftver nem talált adott területet a képen, annak az értékét nem vettem figyelembe. Így a görbék már jelentősen jobban megközelítették az ideális ROC görbe alakját, ahogy azt a 4.8. ábra is szemlélteti.

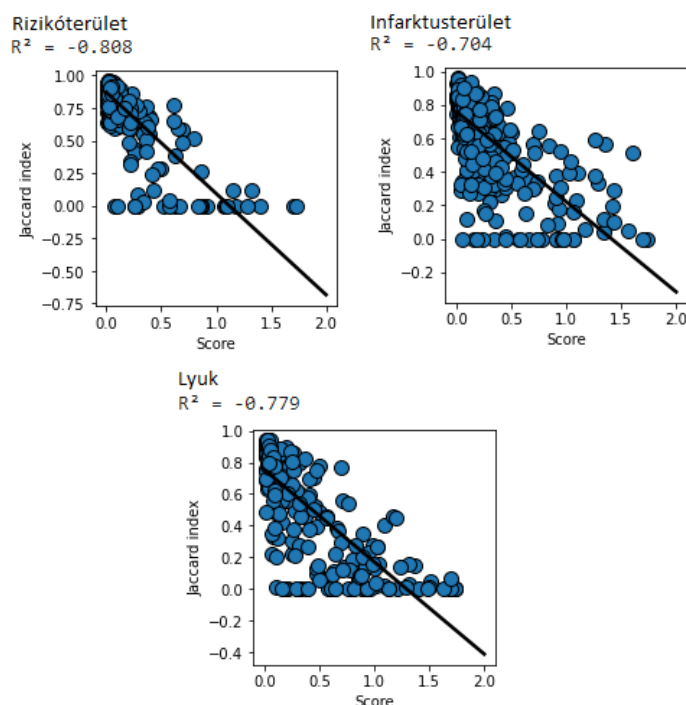


4.8. ábra. ROC görbék üres maszkok kihagyása és variációs index esetén

Figyelembe véve, hogy a szívzszeletek esetében minden JI a 0.75-ös határérték feletti

volt, így nem rendelkezünk "negatív tesztekkel", ezért úgy döntöttünk, ezt a ROC görbét végül nem vizsgáljuk.

Mivel a ROC görbék megfelelőnek tűntek, a lineáris regressziót is végrehajtottam (4.9. ábra). A korrelációs együtthatók értéke itt már sokkal magasabb volt, mint a korábban kipróbált scoring mechanizmusok esetében.



4.9. ábra. Variációs együtthatók score és JI lineáris regressziója üres maszkok kihagyásával

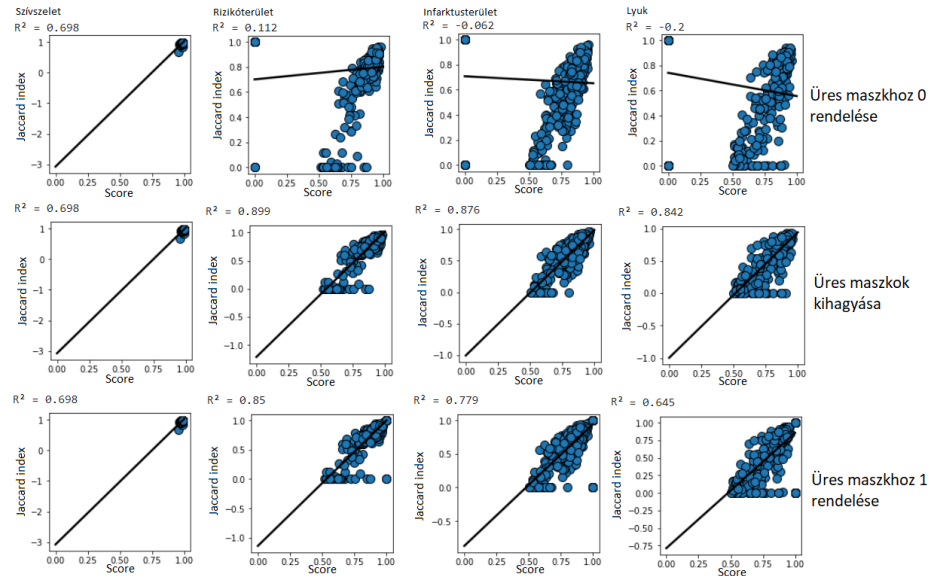
4.3.3. Pixelek valószínűségének átlaga

Mivel minden pixelhez tartozik egy valószínűségi érték, ami megadja, a háló szerint az adott pixel a maszkhoz tartozik-e, ezek átlagának és a JI-nek a korrelációját is megvizsgáltam.

A maszkhoz tartozó pixelek valószínűségi értékeit átlagoltam (tehát a 0.5-tel egyenlő vagy annál nagyobb értékeket), és ezt használtam score értéknek. Abban az esetben, amennyiben a háló nem talált maszkot, nem számoltam score értéket, helyette vagy egy előre meghatározott értéket rendeltem hozzá, vagy teljes mértékben kihagytam a lineáris regresszióból. Itt már a nagyobb JI-hez nagyobb score értéknek kellett tartoznia, ezért pozitív korrelációs együtthatóra számítottunk nagymértékű korreláció esetén.

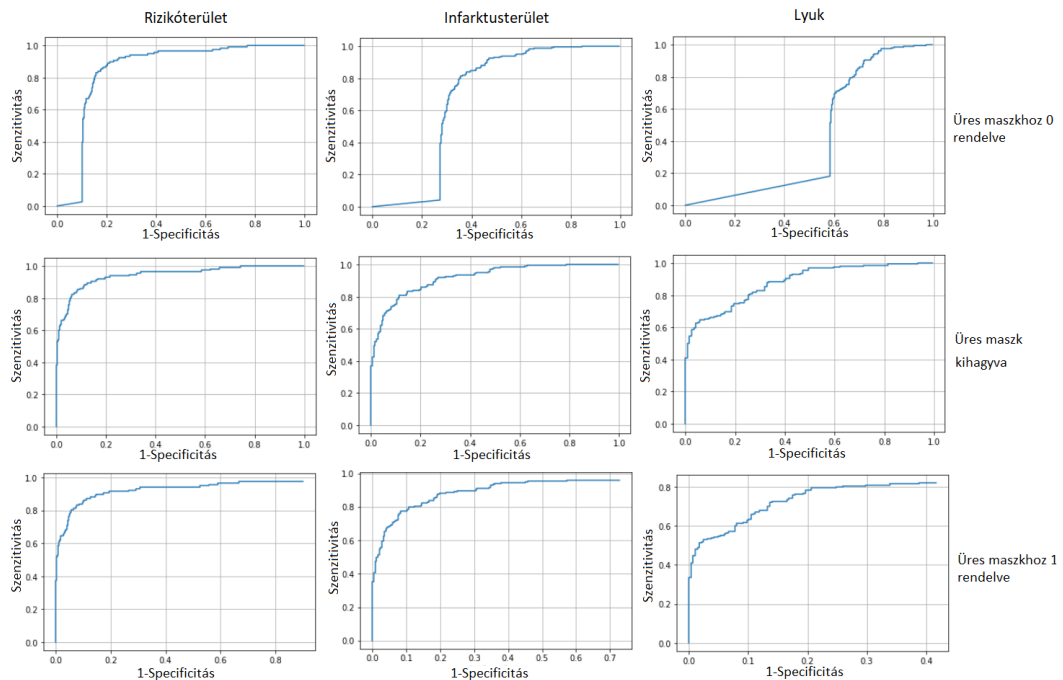
A korrelációs együttható akkor vette fel a legnagyobb értéket, amikor az üres maszkok score-ját kihagytam a lineáris regresszióból (4.10. ábra). Például a legfontosabb terület

- az infarktusterület - esetén a maszkok kihagyásánál 0.876 volt a korrelációs együttható értéke, nullás érték esetén - 0.062, egyes érték esetén pedig 0.779.



4.10. ábra. Score és JI lineáris regresszió az üres maszkokhoz rendelt különböző score értékek esetén

A szelet ROC görbéje az üres maszkok kihagyása és egyes érték esetén is közel ideális volt (4.11. ábra), a többi területnél azonban már ránézésre is sokkal nagyobb volt a görbe alatti terület, amennyiben az üres maszkok nem lettek figyelembe véve.



4.11. ábra. ROC görbék az üres maszkokhoz rendelt különböző score értékek esetén

4.3.4. AUC értékek

Annak céljából, hogy a scoring kiértékelését számszerűsítsük, kiszámoltam az egyes scoring módszerekhez tartozó AUC, azaz Area Under the Curve értéket is. Ez a görbe alatti terület nagyságát adja meg, és ideális esetben 1-es értéket vesz fel [26].

A valószínűségek átlagolása esetén egyértelműen az üres maszkok kihagyása szolgáltatja a legjobb eredményt. Például az infarktusterület esetén az AUC érték ezen score-ok elhagyása esetén 0.914, nullás érték esetén 0.673, egyes érték esetén pedig 0.641 volt.

A főbb értékek a 4.2. táblázatban találhatóak.

4.2. táblázat. AUC értékek különböző scoring módszerek esetén

	Szívszelet	Lyuk	Rizikó	Infarktus
Átlagos különbség	0.875	0.757	0.711	0.683
Szórás és átlag hányadosa (üres maszk nélkül)	0.911	0.863	0.882	0.868
Valószínűségek átlaga (üres maszk nélkül)	0.998	0.872	0.942	0.914

Ahogy már a ROC görbéből is látható volt, az üres maszk nélküli valószínűség-átlagolás szolgáltatja a legjobb eredményt, ezért ezen scoring mechanizmus használata mellett döntöttünk.

4.4. Tanításhoz használt képek számának hatása a hálók hatékonyságára

Végül azt kívántuk megvizsgálni, hogy a hálók hogyan teljesítenek, amennyiben kevesebb képen kerülnek tanításra. Ezt azért találtuk fontos információnak, mert később elképzelhető, hogy minden egyes felhasználónak magának kell tanítania az általunk optimalizált hálót, azonban nem biztos, hogy mindenkinek rendelkezésére áll többszáz vagy több ezer kép a tanításhoz.

Mind a három típusú hálót betanítottam kevesebb képpel is, majd elvégeztem velük az átfogóbb tesztelést.

Mivel az infarktusterület a legfontosabb és egyben legbonyolultabb struktúra, az ehhez kapcsolódó értékekkel dolgoztam. Kiszámoltam a JI-t és a WDP-t. Ezek mediánját ábrázoltam a képek számának függvényében minden egyes háló esetében. Azért mediánt használtam, mert a hálók hibája nem normál eloszlást követett, ahogy azt korábban láthattuk.

Először kettőtől tíz képig végeztem a tanítást, egyesével növelve a képek számát, majd tízesével növeltem százig, onnan pedig száztól ezerig. A tanulóhalmaz méretének növelésekor minden alkalommal az előző tanításhoz használt halmazt bővítettem ki új képekkel, mert úgy gondoltam, realisabb lesz az eredmény, ha minél jobban törekszünk arra, hogy a tanítások körülményei megegyezzenek.

Mindegyik hálót harminc epochon keresztül tanítottam minden tanulóhalmaz esetén. Azért választottam ezt a számot, mert tíz kép esetén körülbelül ennyi után már nem javult a veszteségérték egyik háló esetében sem, csak egyre rosszabb lett, a pár száz számú képnél pedig már belassul a javulás.

A 4.12. ábra mutatja, hogy a JI láthatóan az összes kép esetében is kisebb értékű, mint az eredetileg tanított hálóké, azokat viszont pár száz epochon keresztül tanítottam. Emellett megfigyelhető egy nagyobb ugrás tíz kép esetén mind a WDP, mind a JI esetén.



4.12. ábra. Háló-kimenetek infarktusterületeinek metrikáinak változása eltérő képszámok esetén (medián)

Meglepő módon a U-Net maszkjainak JI-je már két kép esetén is messze felülmúlta a másik két hálót. Ennek az lehet az oka, hogy a másik két háló előre tanított volt, és az ImageNet paraméterek talán nem voltak megfelelőek a szívselet területeinek szegmentálásához. Ezen kívül az is lehetséges, hogy mivel ez a két háló bonyolultabb architektúrával és emiatt több paraméterrel rendelkezik, nekik több információra van szükségük ahhoz, hogy minden paraméter helyesen beállításra kerüljön.

Megvizsgáltam a hálók kimeneti maszkjait kis méretű tanítóhalmaz esetén. Azt találtam, hogy a maszkok általában teljesen feketék vagy fehérek voltak (esetenként néhány

véletlenszerűnek tűnő ellenkező színű pixellel). Mivel az infarktusterület mérete általában a kép méretéhez képest kicsi, amennyiben minden pixel fekete, a WDP a már korábban leírt okokból viszonylag jó értéket fog felvenni.

Mindemellett a prezentált eredmények azt mutatják, hogy meglepően kevés képre és epochra van szükség ahhoz, hogy a hálók jól teljesítsenek, de a szegmentálás tíz-húsz kép esetén még messze nem tökéletes.

Bár kis tanulóhalmaz esetén a szívszelet, az infarktus és a rizikó területe viszonylag könnyen megtalálásra kerülnek, a kamrallyuk maszkja általában még ekkor is teljesen fekete, és csak pár képen találhatóak egyáltalán olyan pixelek, amik a hálók szerint a lyukhoz tartoznak. Ennek az lehet az oka, hogy a lyuk általában nagyon hasonló színű a háttérhez vagy a szívszelet előkészítése során keletkező artefaktokhoz (az árnyékok miatt esetleg valamivel sötétebb), ezért szegmentálása során azt is figyelembe kell venni, található-e körülötte szívszövet, míg a többi terület esetén már a pixelek színéből is lehet következtetni arra, milyen területhez tartozik az adott pixel.

Összességében levonható a következtetés, hogy az eredmények alapján, a kiértékelés hibái mellett, meglepően kevés kép elengedő a hálók tanításához, ami arra enged következtetni, hogy egy de novo tanítás esetén is elég egy kis tanítóhalmaz egy elfogadható eredményhez.

4.4.1. Igazságmátrix

Annak céljából, hogy megvizsgáljuk, a különböző méretű tanítóhalmazok esetében a pixelek mekkora része került helyesen vagy helytelenül klasszifikálva, igazságmátrixot (confusion matrix) készítettem.

Mivel csak különböző módon klasszifikált pixelek eloszlásnak tendenciájára voltunk kíváncsiak, úgy döntöttem, csak a 2, 5, 10, 50, 100, 500, 1000 és 1172 képből álló tanulóhalmaz esetében végzem el a vizsgálatot.

Az igazságmátrixok a ROC görbénél már említett négy értéket, a valódi negatív, valódi pozitív, fals negatív és fals pozitív értékeket tartalmazzák [27]. Mivel esetemben a pixelek klasszifikációjáról volt szó, minden egyes teszhalmazbeli képhez készíthettem volna egy mátrixot, de mivel majdnem ötszáz képről volt szó, ez a módszer az átláthatatlansága miatt nem szolgált volna használható eredményekkel.

Ennél fogva inkább az összes tesztképhez tartozó maszk pixeleit megvizsgáltam, és egyetlen, a teljes teszhalmazra vonatkozó mátrixot készítettem minden, ugyanazon a tanulóhalmazon tanított háló és a kombinált kimenetük esetében. Mivel a korábbi mérés-

seket az infarktusterület maszkjain végeztem, itt is így tettem.

Hogy a kiértékelés könnyebb legyen, a pixelek számát elosztottam az összes pixel számával, így százalékos értékeket kaptam.

4.3. táblázat. Igazságmátrix kételemű tanulóhalmaz (kombinált kimenet infarktusterület) esetén

	Maszk	Háttér	Összesen
Maszkként klasszifikált	0.03 %	0.04 %	0.07 %
Háttérként klasszifikált	12.77 %	87.16 %	99.93 %
Összesen	12.8 %	87.2 %	

Ahogy azt a 4.3. táblázat mutatja és ahogy már fentebb említettem, kis tanulóhalmaz esetén a legtöbb pixel háttérként kerül klasszifikálásra, ezáltal a kép fekete lesz. Bár csak a pixelek kicsit több, mint 87 %-a háttér, 99.93 %-uk kapja ezt az osztályt. A pixelek maszkként történő besorolása pedig teljesen véletlenszerűnek tűnik a mindössze 0.01 %-os különbség alapján, ami a valóban maszkhoz és valójában háttérhez tartozó maszkhoz besorolt pixeleket illeti.

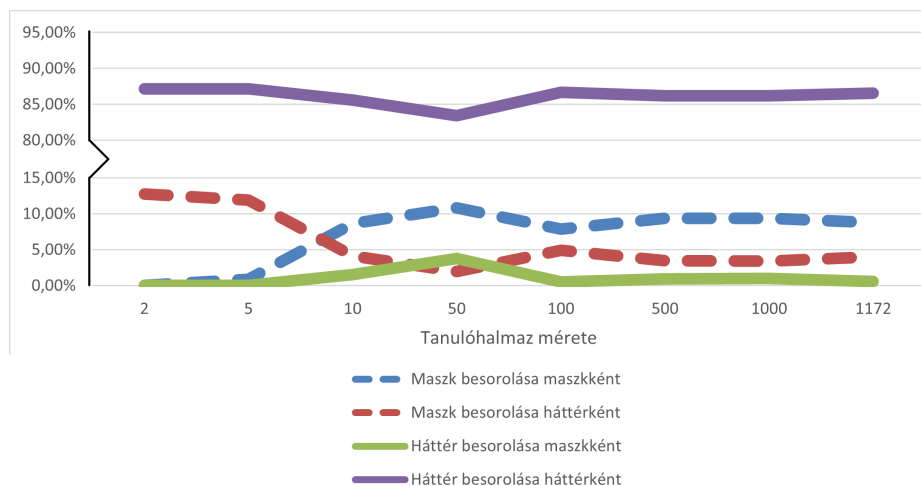
Megemlítendő, hogy a U-Net és U-Net++ hálók a pixelek sokkal nagyobb százalékát (U-Net++ esetében összesen a pixelek 34.68 %-a került maszkként klasszifikálásra) rendelték a maszkhoz, míg a DeepLabV3+ egy maszkpixelt sem volt képes helyesen a maszkhoz sorolni (4.13. ábra).

Ezek alapján úgy gondolom, hogy a DeepLabV3+ pixelvalószínűségi értékei nagyok lehetnek, a másik két háló pedig valószínűleg elég bizonytalan lehetett a pixelek besorolásában, és csak véletlenül kerültek a maszkhoz való tartozás valószínűségének értékei kicsivel 0.5 fölé.

U-Net				U-Net++				DeepLabV3+			
	Maszk	Háttér	Összesen		Maszk	Háttér	Összesen		Maszk	Háttér	Összesen
Maszkként	12.38 %	22.3 %	34.68 %	Maszkként	1.07 %	10.47 %	11.54 %	Maszkként	0.0 %	0.00001 %	0.00001 %
Háttérként	0.43 %	64.89%	65.32 %	Háttérként	11.73 %	76.73%	88.46 %	Háttérként	12.8 %	87.2 %	100 %
Összesen	12.8 %	87.2 %		Összesen	12.8 %	87.2 %		Összesen	12.8 %	87.2 %	

4.13. ábra. Hálók igazságmátrixai kételemű tanulóhalmaz esetén

A különböző értékek változását a 4.14. diagram szemlélteti. Öt kép esetén a kombinált kimenetnél az értékek nagyon hasonlóak voltak, bár a maszkként klasszifikált és valóban maszkhoz tartozó pixelek százaléka felemelkedett 0.87 %-ra. Tíz kép esetén ez az érték 8.69 %-ra nőtt, és a nagyobb tanulóhalmazok esetén ezen érték körül maradt.



4.14. ábra. Igazságmátrix értékeinek változása kombinált kimenet infarktusterületének esetében a tanulóhalmaz méretének függvényében

A maszkként klasszifikált, de háttérhez tartozó pixelek száma is nőni kezdett a tanulóhalmaz méretének növelésével, mielőtt újra csökkenni kezdett volna. Véleményem szerint ez az átmeneti növekedés annak köszönhető, hogy a hálók addigra kaptak elegendő maszkként klasszifikált pixelt ahhoz, hogy rendesen "érzékeljék" ennek a csoportnak a létezését (és ezért nagyobb bizonyossággal sorolták ide a pixeleket), de még nem rendelkeztek elég információval ahhoz, hogy a klasszifikációt helyesen végezzék.

4.4. táblázat. Igazságmátrix 1172 elemű tanulóhalmaz (kombinált kimenet infarktusterülete) esetén

	Maszk	Háttér	Összesen
Maszkként klasszifikált	8.77 %	0.61 %	9.38 %
Háttérként klasszifikált	4.03 %	86.59 %	90.62 %
Összesen	12.8 %	87.2 %	

Végül a helytelenül klasszifikált háttérpixelek százaléka a 1172 képből álló tanulóhalmaz esetén a 4.4. táblázatban látható 0.61 %-ra állt be. Ennyi kép esetén már mind a három háló nagyon hasonló eredményeket hozott. Mindegyik esetén kevesebb, mint 1 %-os eltérés volt a négy, mátrixban feltüntetett értékben a kombinált kimenethez képest.

Eredményeink alátámasztják a fentebb kifejtett feltételezést, hogy nincs szükség kifejezetten sok képre a hálók betanításához, azonban a kiértékelés minősége pár száz képig jelentősen javul.

5. fejezet

Összefoglalás

5.1. Elvégzett munka

A munkánk során különféle architektúrájú - U-Net, U-Net++ és DeepLabV3+ - neurális hálókat tanítottam szívszelvények különböző területeinek felismerésére és szegmentálására, hogy ezek kiértékelését automatizáljuk. A három hálóból kettő előre tanított háló volt ImageNet adathalmazon tanított paraméterekkel inicializálva, egy pedig egy általam felépített U-Net architektúra.

A munkám során különféle veszteségfüggvényekkel és optimalizálókkal, különböző ideig tanítottam a három, fent említett architektúrájú hálót, hogy megtaláljam a legoptimálisabb módját a tanításnak. A hálókat összehasonlítottam egymással a JI és WDP metrikákkal.

A hálók teljesítményét lineáris regresszióval és korrelációs együttható számításával, valamint a választott metrikák violin és box plotokon való ábrázolásával vizsgáltam.

Hogy a gyakorlatban megvalósított automatizált kiértékelést hatékonyabbá tegyük, egy scoring rendszert is kidolgoztunk, ami arra hivatott, hogy jelezze, egy adott szelet adott maszkja esetében szükség lehet-e manuális korrekcióra.

Ennek során többféle scoring mechanizmust is kipróbáltam, a területméretek átlagos eltérését, a variációs együtthatót és a maszkhoz tartozó pixelek valószínűségét. A mechanizmusok hatékonyságát korrelációval és ROC görbékkel vizsgáltam.

A hálókat eltérő nagyságú tanulóhalmazon is betanítottam, annak céljából, hogy lássuk, hogyan hat a tanítási képek száma a hálók hatékonyságára és pontosságára.

5.2. Eredmények

Az általam tanított hálók esetében a BCEWithLogitsLoss függvény és a 0.006-os tanulási rátás Adam optimalizáló használata bizonyult a leghatékonyabbnak. A hálókat pár száz epochon keresztül kellett tanítani és a paramétereket csupán akkor elmenteni, amennyiben a tesztelés veszteségértéke csökkent ahhoz, hogy a legjobb eredményt érjem el.

Az összes háló a violin és box plotok, ROC görbék és a használt metrikák alapján közel azonos teljesítményt nyújtott. Bár a JI és a WDP használata is jár hátrányokkal, úgy ítéltük meg, együttes használatukkal nagyjából sikerült kompenzálni a metrikák jellegéből eredő torzításokat.

A különböző struktúrájú hálók hasonló pontossága véleményünk szerint azt mutatja, hogy a szívzszeletek automatikus annotálása egy szegmentációs neurális hálózatokhoz illő, azokkal megoldható feladat. Az itt leírt eredmények tehát (a véletlenszerűen inicializált paraméterek által szolgáltatott apró változtatások keretein belül) reprodukálhatóak, nem csak egy véletlen szerencsés paraméterinicializációból fakadnak.

A kimeneti maszkok minőségének számszerűsítésére a legjobb score értéknek a maszkpixelek valószínűségének átlaga bizonyult. Ennek a JI-vel való korrelációja volt a legnagyobb, és a ROC görbe is megközelítette az ideálisat mind a négy maszktípus esetében. Ehhez azonban arra volt szükség, hogy az üres maszkokhoz ne rendeljek score értéket.

A tény, hogy a hálók már tíz kép esetén is képesek megtanulni a maszkok főbb jellemzőit, és nem csak véletlenszerűen rendelnek valószínűségi értékeket az egyes pixelekhez - még ha nem is szolgáltatnak tökéletes eredményeket - azt mutatja, hogy a kapott szegmentációs feladat nem jelent nagy kihívást a hálók számára, így a jövőben kisebb bemeneti adathalmaz is elegendő a tanításukra.

5.3. Jövőbeni tervek

Mivel az eddigi munkánk bizonyította, hogy a kitűzött feladat megoldható neurális hálóval, és az eredmények reprodukálhatóak, reálisnak gondoljuk a hálók egy kiértékelő szoftverben való használatát.

A széleskörű felhasználhatóság érdekében azonban szükség van arra, hogy a hálók ne csak a Semmelweis Egyetem Farmakológiai és Farmakoterápiai Intézetének laborjában kezelt szívzszeleteket ismerjék fel. Különböző laborok kissé eltérő módon kezelhetik a

szeleteket, így azok kinézete eltérő lehet.

Emiatt domain adaptációra van szükség, hogy a hálók különböző laborok képeit is képes legyen helyesen annotálni. Jelenleg már rendelkezésünkre állnak képek öt laborból, amik segítségével az adaptációt elvégezni tervezzük, és bízunk abban hogy a hálók eddigi hatékonyságát tekintve jól fognak teljesíteni a domain adaptáció terén is. Így akár külön tanítás nélkül széleskörűen, több labor között is alkalmazható neurális hálózatokat tudunk alkotni.

Felhasznált irodalom

- [1] B. Redfors, S. Yangzhen és E. Omerovic, „Myocardial infarct size and area at risk assessment in mice“, *National Institutes of Health's National Library of Medicine*, 2012. cím: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3627291/>.
- [2] A. H. Aletras, G. S. Tilak, A. Natanzon és tsai., „Retrospective Determination of the Area at Risk for Reperfused Acute Myocardial Infarction With T2-Weighted Cardiac Magnetic Resonance Imaging“, *American Heart Association Journals*, 2006. cím: <https://www.ahajournals.org/doi/10.1161/circulationaha.105.576025>.
- [3] Á. Zarándy, „Perceptrons“, *PPKE ITK, Neural Networks előadás*, 2021.
- [4] Á. Zarándy, „Components and methods of deep neural networks“, *PPKE ITK, Neural Networks előadás*, 2021.
- [5] C.-F. Wang, „The Vanishing Gradient Problem“, *Towards Data Science*, 2019. cím: <https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484>.
- [6] S. Sharma, „Activation Functions in Neural Networks“, *Towards Data Science*, 2017. cím: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>.
- [7] J. Brownlee, „Loss and Loss Functions for Training Deep Learning Neural Networks“, *Machine Learning Mastery*, 2019. cím: <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/>.
- [8] S. Doshi, „Various Optimization Algorithms For Training Neural Network“, *Towards Data Science*, 2019. cím: <https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6>.

- [9] Á. Zarándy, „Components and methods of deep neural networks II“, *PPKE ITK, Neural Networks előadás*, 2021.
- [10] B. Benyó, „Konvolúciós hálózatok“, cím: https://semmelweis.hu/kepalkotas/files/Convolut_network_Benyo.pdf.
- [11] „Blurring an image“, *Apple Developer Documentation*, cím: https://developer.apple.com/documentation/accelerate/blurring_an_image.
- [12] „Object Detection vs Object Recognition vs Image Segmentation“, 2022. cím: <https://www.geeksforgeeks.org/object-detection-vs-object-recognition-vs-image-segmentation/>.
- [13] D. Mwití és K. (Yi) Li, „Image Segmentation in 2022: Architectures, Losses, Datasets, and Frameworks“, *Neptune Labs*, 2022. cím: <https://neptune.ai/blog/image-segmentation>.
- [14] N. Tomar, „What is UNET?“, 2021. cím: <https://medium.com/analytics-vidhya/what-is-unet-157314c87634>.
- [15] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh és J. Liang, „UNet++: A Nested U-Net Architecture for Medical Image Segmentation“, *Arizona State University*, 2018. cím: <https://arxiv.org/pdf/1807.10165v1.pdf>.
- [16] P.-L. Pröve, „An Introduction to different Types of Convolutions in Deep Learning“, *Towards Data Science*, 2017. cím: <https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d>.
- [17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy és A. L. Yuille, „DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs“, 2017. cím: <https://arxiv.org/pdf/1606.00915.pdf>.
- [18] I. Ahmed, M. Ahmad, F. A. Khan és M. Asif, „Comparison of Deep-Learning-Based Segmentation Models: Using Top View Person Images“, cím: https://developer.apple.com/documentation/accelerate/blurring_an_image.
- [19] „Google Colaboratory“, cím: <https://colab.research.google.com/>.
- [20] „Understand torch.nn.MSELoss(): Compute the Mean Squared Error (Squared L2 Norm) – PyTorch Tutorial“, 2022. cím: <https://www.tutorialexample.com/understand-torch-nn-mse-loss-compute-the-mean-squared-error-squared-l2-norm-pytorch-tutorial/>.

- [21] D. J. Huang, „Learning Day 57/Practical 5: Loss function — CrossEntropyLoss vs BCELoss in Pytorch; Softmax vs sigmoid; Loss calculation“, *Medium*, 2021. cím: <https://medium.com/dejunhuang/learning-day-57-practical-5-loss-function-crossentropyloss-vs-bceloss-in-pytorch-softmax-vs-bd866c8a0d23>.
- [22] J. Brownlee, „Gentle Introduction to the Adam Optimization Algorithm for Deep Learning“, *Machine Learning Mastery*, 2017. cím: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>.
- [23] „Python library with Neural Networks for Image Segmentation based on PyTorch.“, cím: https://github.com/qubvel/segmentation_models.pytorch.
- [24] I. E. Vorontsov, I. V. Kulakovskiy és V. J. Makeev, „Jaccard index based similarity measure to compare transcription factor binding site models“, *Algorithms for Molecular Biology*, 2013. cím: <https://almob.biomedcentral.com/articles/10.1186/1748-7188-8-23>.
- [25] M. Yi, „A Complete Guide to Violin Plots“, cím: <https://chartio.com/learn/charts/violin-plot-complete-guide/>.
- [26] S. Ekelund, „ROC curves – what are they and how are they used?“, *Acute Care Testing*, 2011. cím: <https://acutecaretesting.org/en/articles/roc-curves-what-are-they-and-how-are-they-used>.
- [27] S. Narkhede, „Understanding Confusion Matrix“, *Towards Data Science*, 2018. cím: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>.