Önálló laboratórium beszámoló

Tüdő szegmentációja mellkas CT felvételeken neurális hálókkal

Tumay Ádám

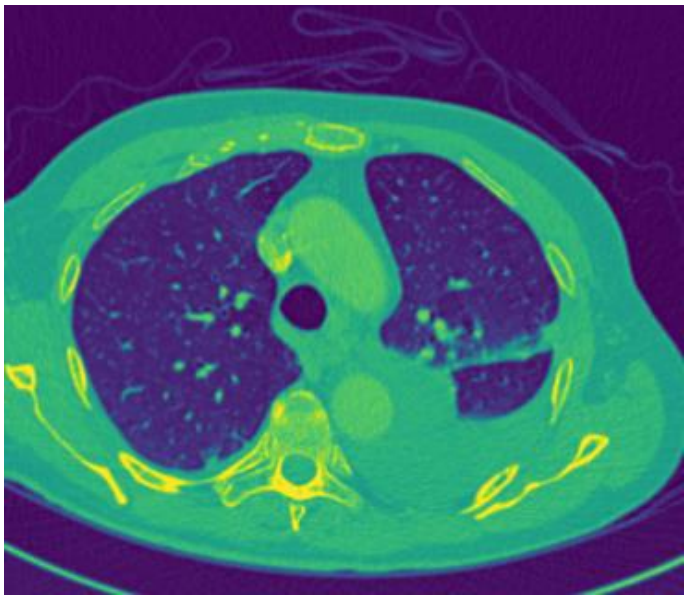
Konzulens: Hadházi Dániel

Bevezetés

Az önálló laboratórium során neurális hálók konstruálásával és tanításával foglalkoztam. Ennek keretében mellkas CT felvételeken tüdő szegmentációját végeztem el. A feladatot irodalomkutatással kezdtem, ahol a megfelelő architektúrák, és módszerek kiválasztása történt, illetve egy alkalmas adathalmaz beszerzése. (Kezdetben a feladat erek detekciója lett volna, azonban miután nem állt rendelkezésre megfelelő adathalmaz, így jött a körvonal szegmentálás. Ennek esetleges az eredményre vonatkozó következményeire ki fogok térni ahol szükséges.) Ezek után implementáltam a kiválasztott architektúrát, és nekiláttam a tanításnak. Az eredmények függvényében iteratívan változtattam a háló felépítésén, és úgy folytattam munkát. Miután látszólag az alap architektúra képességeit maximálisan kihasználó eredményre jutottam, IMSC feladatként a fentieket végrehajtottam egy másik, egyszerűbb hálóra is.

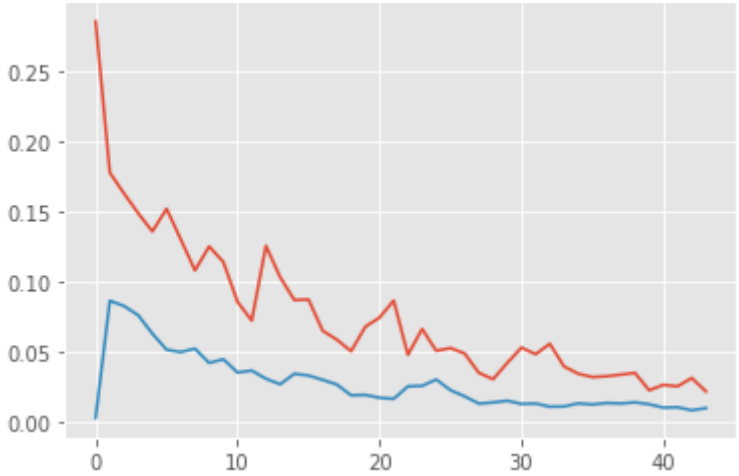
Dataset

Bemenetként a Vessel12Dataset-et használtam[1]. Az adathalmaz eredetileg erek szegmentációjára volt hivatott, azonban annak teljesen annotált részéhez nem sikerült hozzáférést szerezni. Rendelkezésre álltak azonban a tüdő körvonal maszkok, amelyek már megfelelő bemenetei lehettek a hálónak. Az adathalmaz 20 kontraszttal ellátott felvételt tartalmaz, ebből egy később részletezett feldolgozási lépés után 19 bizonyult használhatónak. Itt a felvételek felbontása a tanító, tesztelő, és kiértékelésre használt adathalmazok között a 11-5-3 arány lett. Külön kihívást jelentett, hogy beteg páciensek felvételeit is tartalmazta, ezáltal a detekciót gyulladásos képletek, daganatok, és egyéb deformációk nehezítették.



ábra 1: Egy beteg páciens felvétele, jól kivehetőkép jobb oldalán az egyik lebeny hiánya, valamint egy nagyobb kiterjedésű gyulladás

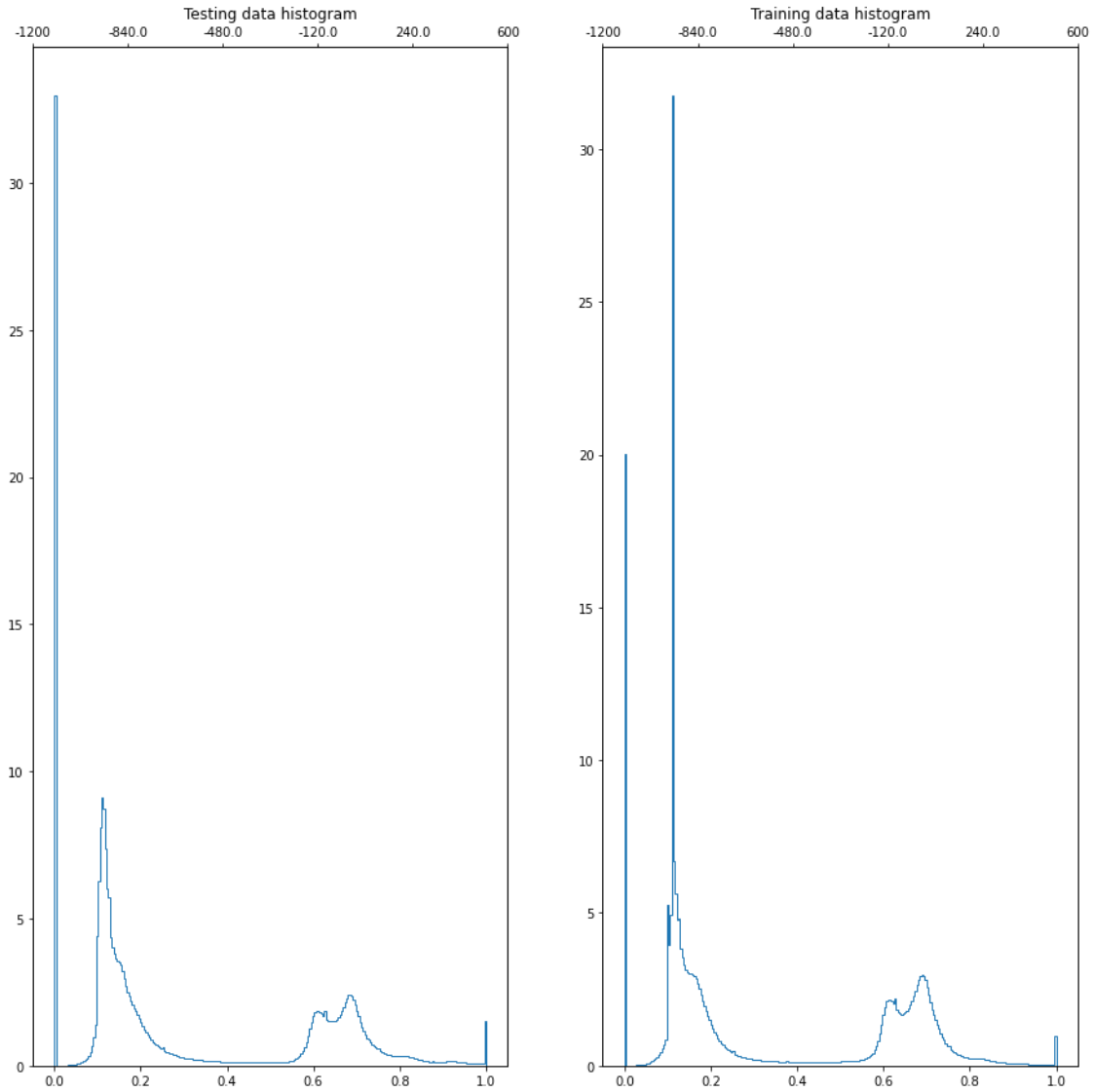
Egy másik probléma volt, hogy megfelelő mennyiségű memória hiányában nem lehetett egyszerre az összes felvételt betölteni, így eleinte a tanítás a következőképpen történt: CT betöltése -> tanítás a felvétel szeletein -> másik felvétel betöltése. A megoldás, bár memória, és IO szempontjából hatékony, miután egyes felvételek jelentősen más eloszlást mutathatnak a tanítás előrehaladottabb szakaszaiban problémát okozhat, hiszen mindig túltanít az éppen betöltött felvétel eloszlására.



ábra 2: Mind a tanító, mind a kiértékelő halmaz esetén a hibafüggvényen jól látható ugrásként jelennek meg a felvétel váltások

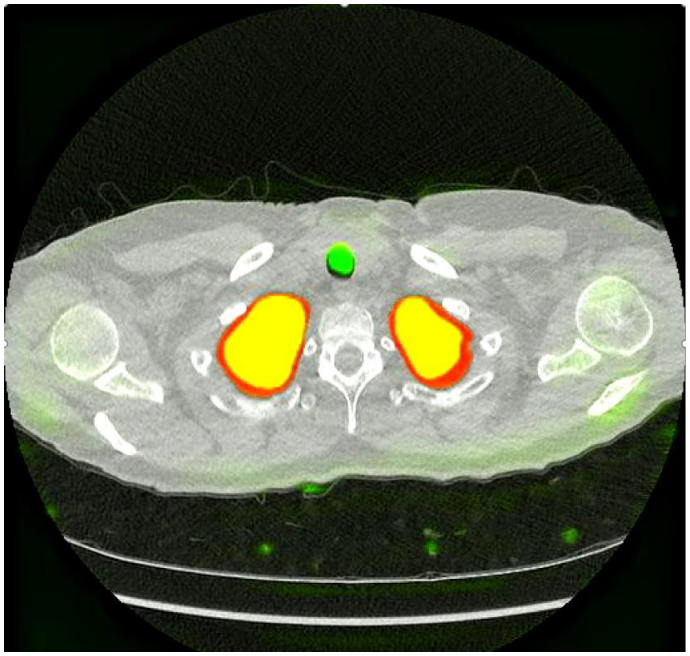
[1] https://vessel12.grand-challenge.org/

Ennek feloldására a következő módszert alkalmaztam: Egy adathalmaz minden felvételét egyesével betöltöm, majd adott, véletlenszerűen kiválasztott kezdőindextől folytonosan beolvasok adott számú szeletet (esetünkben ez felvételenként 100), és elmentem. Az így kapott részhalmazok konkatenáltja fogja egy epoch-ban a tanítást végezni, remélve, hogy az így kapott eloszlás már reprezentatív lesz az összes felvételre. Az elméletet az intenzitások eloszlásának vizsgálatával ellenőriztem, és arra jutottam, hogy adott adathalmazon belül valóban reprezentatív képet kapunk, azonban a tanító és a tesztfelvételek halmaza nem elég nagy, nem teljesen azonos az eloszlásuk, de a különbség remélhetőleg már nem fog komolyabb problémát okozni.



ábra 3: A tanító és teszt adathalmazok eloszlása a mintavételezés után. Jól látható különbségek alacsonyabb intenzitások mellett.

Egy másik így keletkezett probléma, hogy miután a halmaz kezdőindexe egyenletesen véletlenszerűen választódik a [0-„felvétel_vége – kiválasztandó_szakasz_hossza”] tartományból így a végeken kiválasztandó_szakasz_hossza/2 számú szeletnél a szélek felé haladva egyenletesen csökken annak a valószínűsége, hogy benne lesznek-e egy adathalmazban. Ezt a problémát azonban elhanyagoltam abból az okból kifolyólag, hogy széleken amúgy sem vagyunk kifejezetten kíváncsiak a mérés eredményére, hiszen általában ott már csak a hasi/nyaki tájékok találhatóak. Kiértékelésnél azonban a tanítás hiánya miatt feltűnő ezeken a területeken a jóslás pontatlansága.



ábra 4: Pontatlan detekció a felvétel szélein

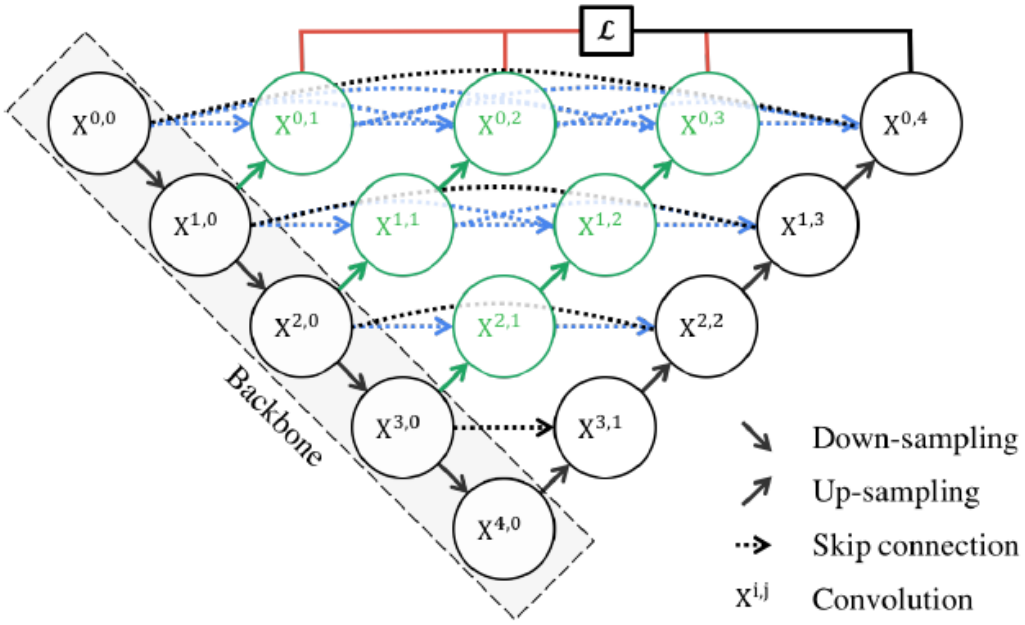
A fenti módon betöltött adatokat előfeldolgozásként még normáltam, mely intenzitás szerinti vágást jelent a [-1200, 600] tartományra, majd normalizációt. A vágás az ilyen felvételek tipikus szélsőértékeire történik, így elkerülve az outlier adatok miatti min-max esetén egyébként bekövetkező eloszlásshiftet a különböző felvételek értékei között.

A feladathoz szükség volt még automatikus közelítő szegmentációkra is, melyek a tanítás-, valamint a háló predikciójának segítésére voltak felhasználva. Ehhez egy a felvétellel azonos méretű tömbben, a voxelek értékei az eredeti felvételben -500 alatti értékek esetén 1-nek, az a fölöttiek esetén pedig 0-nak lettek véve, majd rekurzívan egy, a felvétel szélén található ponttól a háttér vissza lett feketítve.

ábra 5: Egy automatikus szegmentációból származó szelet

Architektúra:

Szegmentációs feladatokra az egyik legnépszerűbb alkalmazott architektúra a UNET. Itt a feldolgozás alapját blokkok alkotják, melyek több réteg konvolúciót, nemlinearitást, illetve esetlegesen extra regularizációs komponenseket tartalmaznak. Ezekből épül fel egy többszintű enkóder és egy dekóder hálózat, ahol az azonos szinteket skip connection-ök is összekötik. Én a fent vázolt hálózat egy továbbfejlesztett, Nested UNET-nek (vagy UNET++-nak) nevezett változatát használtam (Zhou Z., Rahman Siddiquee M.M., Tajbakhsh N., Liang J., 2018), mely korábbi kutatások során már jól teljesített erek szegmentációjában (Cui, H., Liu, X., Huang, N., 2019). A fő változtatás a sima UNET-hez képest, hogy a magasabb absztrakciós szinten dolgozó feature map-ek felmintavételezés után skip connection-nel ugyancsak be vannak kötve a dekóderbe, így egy-egy blokk jelentős extra információmennyiséghez juthat, de a háló komplexitása ugyancsak erősen nő, így a túltanulás ellen jelentős regularizációra lesz szükség.



ábra 6: UNET, és Nested UNET: a feketével kijeölt blokkok alkotják a UNET-et, zölddel jelölve a további addíciók láthatóak a UNET++-hoz. [2]

Esetünkben egy blokk felépítése a következő volt, az utolsó blokkot leszámítva:
2D konvolúció -> Regularizáció* -> ReLU -> 2D konvolúció -> Regularizáció* -> ReLU

Regularizációnak a következő technikák voltak alkalmazva:

- Nincs regularizáció,
- Dropout 0.1-0.2 közötti valószínűséggel,
- Switchable Normalization (a továbbiakban switchnorm),
- Switchnorm és Dropout együttesen.

Az utolsó blokk felépítése pedig a következő volt:
2D konvolúció -> Regularizáció* -> ReLU -> 2D konvolúció -> Sigmoid

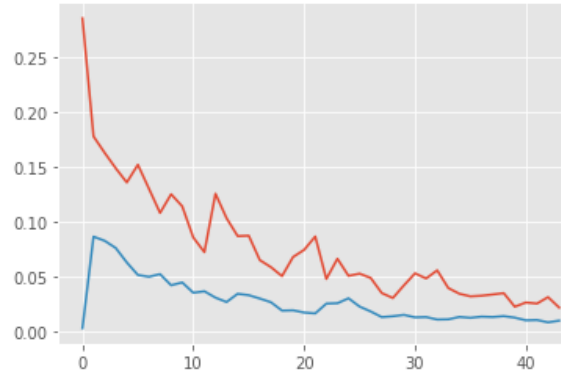
Itt a szigmoid függvény alapú nemlinearitás megválasztásának oka, hogy a háló kimenete 1, és 0 közé essen, mely lehetővé teszi a bináris keresztentrópia alapú hibafüggvény alkalmazását.

[2] Forrás: Forrás: https://link.springer.com/chapter/10.1007/978-3-030-00889-5_1

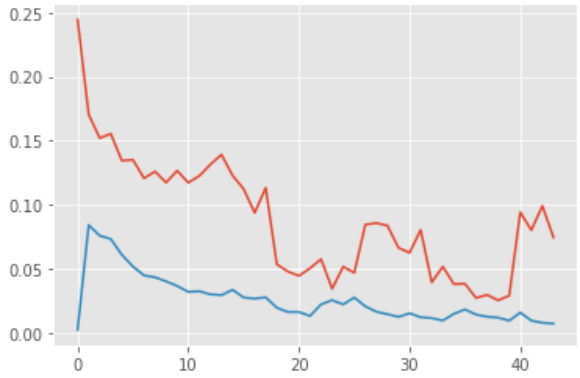
Magyarázatra szorul még a 2D konvolúciók alkalmazása, hiszen alapvetően a CT felvételek 3D mátrixok, így a feldolgozáshoz is logikusnak hangozhat a 3D konvolúció, azonban ezek használatának memória, és számításigénye rendkívül magas, így a felvételek szeletenkénti feldolgozását részesítjük előnyben. Ennek következménye viszont, hogy egy-egy szelet feldolgozásánál elveszik a mélységinformáció, azaz, hogy mi volt a szelet fölött, illetve alatt található, mely fontos információ lehet. Ennek kiküszöbölésére lett bevezetve egy 2,5D-s megoldás, melyben egyszerre nem egy, hanem 9 szeletet adunk több csatorna alkalmazásával, a kiértékelést, illetve a hiba számítását pedig csak a középső szeletre végezzük el, ekkor lokálisan rendelkezésre áll mélységinformáció, melyet a 2D konvolúciók is képesek figyelembe venni. Bizonyos esetekben egy további csatornára lett helyezve bemenetként az automatikus szegmentáció, mely a Dataset fejezetben volt részletezve. A háló kimenete mindig egy csatorna, ahol az intenzitás a háló szerinti jóslás arra, hogy az adott voxel része-e a tüdőnek.

Tanítás:

Kezdetben a megfelelő hibafüggvény kikísérletezésén dolgoztam, itt kétféle módszert teszteltem: átlagos négyzetes hibát néztem, illetve bináris keresztentrópiát. Azt tapasztaltam, hogy utóbbi esetben jobban tanul, gyorsabban konvergál, így a továbbiakban ezt alkalmaztam. Optimizernek Adam-ot használtam, 10^{-4}-es tanulási tényezővel, illetve 10^{-7}-es epszilonnal a nagyobb stabilitás érdekében. Eleinte próbálkoztam egyszerű SGD-vel is, azonban az eleinte lassú volt, később pedig túl magabiztos volt, és túltanult, így ezt elvetettem. A kezdeti architektúrában a felhasznált irodalom szerinti switchnorm volt használva regularizációnak. Első körben azt vizsgáltam, hogy hogyan tanul háló segédszeletekkel, illetve anélkül, ezzel a következő eredményekre jutottam:

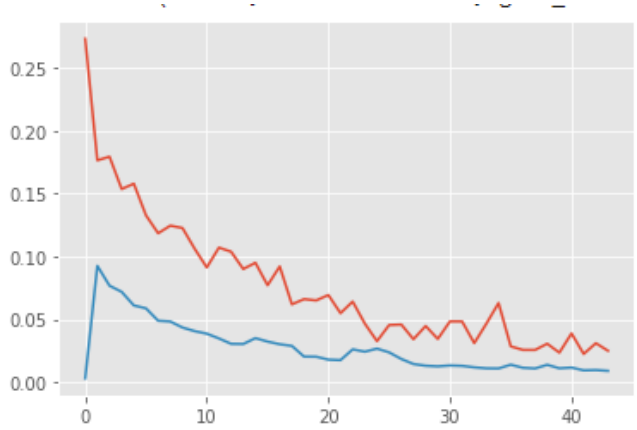


ábra 8: Tanítás segítséggel



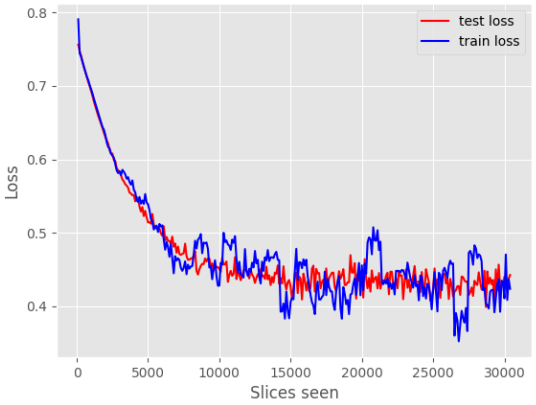
ábra 7: Tanítás segítség nélkül

Az ábrákon pirossal a teszt halmaz, kékkel a tanító halmaz hibája látható. Ami megfigyelhető, hogy bár mindkét esetben a tanító halmazon a hiba kb. azonosan konvergál a teszt halmaz esetén ez már nem igaz, erős túltanulás jellemzi a modellt a második esetben. (Itt még hibafüggvénynek MSE volt használva, illetve a tanítás minden epochban egy felvétellel történt.) Látható, hogy segédszelettel a nyereség van akkora, hogy az megérje az extra erőforrásigényt, így a következőkben ezzel dolgoztam tovább. Miután a háló meglehetősen nagy mennyiségű memóriát alkalmazott a következő egyszerűsítéseket tettem: Eleinte a sima bemenettel megegyező, 9 layernyi automatikus szegmentációt csatoltam segítségként, ezt 1-re csökkentettem, valamint feldolgozás előtt felére csökkentettem a bemenet felbontását, majd utána visszaskáláztam. Az így kapott memórianyereséggel a batch méretet tudtam növelni 5-re. Hogy szemléltethető legyen a veszteség, amit az egyszerűsítésékkel kapunk a következő mérést még 1-es batch méret szerint végeztem. Amint látható, a kimeneten mért hiba különösebben nem változott meg ezek hatására:



ábra 11: Tanítás egyszerűsítésekkel

Most, hogy adott az architektúra, a bemenet struktúrája, az elvárt kimenet, valamint a hibafüggvényt is lecseréltem bináris keresztentrópiára, továbbá már a reprezentatívan mintavételező tanító, és kiértékelő adathalmazra váltottam, a különféle regularizációk hatását kezdtem el vizsgálni. Kezdetben kivettem minden regularizációt és így figyeltem meg a hibafüggvényeket. Amint látható a konvergencia egy viszonylag magas értékre történik (az automatikus szegmentáció átlagos hibája 0.6-0.7 körül alakul), nem kifejezetten stabil az eredmény, hasonlót tapasztaltam dropout normalizáció mellett is 0.2-es értéknél. A hiba nem csökkent lényeges mértékben, és továbbra sem konvergál túl szépen.



ábra 10: Tanítás regularizáció nélkül

ábra 9: Tanítás dropout regularizációval

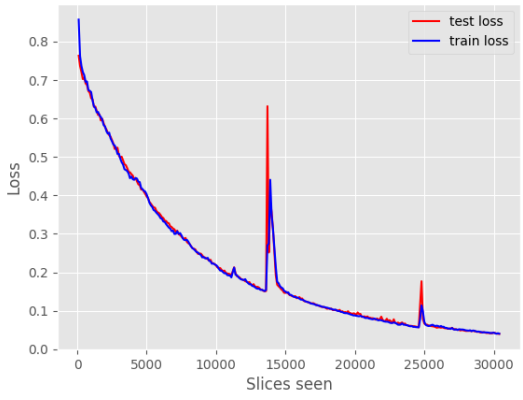
A következő kipróbált módszer a már ajánlott switchnorm volt. Ennek lényege, hogy layer, batch, és instance szinten számol átlagot és szórást a bemenetről, majd ezek alapján a következőképp normalizálja egy voxel értékét: (Ping Luo, Ruimao Zhang, Jiamin Ren, Zhanglin Peng, Jingyu Li, 2019)

$$\hat{h}_{ncij} = \gamma \frac{\mathrm{h_{ncij}} - \Sigma_{\mathrm{k}\in\Omega}\, \mathrm{w_k}\mu_\mathrm{k}}{\sqrt{\Sigma_{\mathrm{k}\in\Omega}\, \mathrm{w'_k}\sigma^2_\mathrm{k} + \epsilon}} + \beta$$

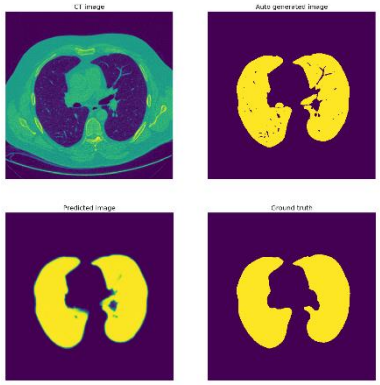
$$w_k = \frac{e^{\lambda_k}}{\Sigma_{\mathrm{z}\in\{\mathrm{in,ln,bn}\}} e^{\lambda_z}} \text{ és } k \in \{in, bn, ln\}$$

Itt h az adott voxel értékét jelöli az n-edik batchben, c-edik csatornában, i, j koordinátákon. ϵ a numerikus stabilitásért felelős tag, γ és β skálázási és eltolási paraméterek, μ, σ az átlagot és szórást jelölő tagok, valamint $\Omega = \{\mathrm{in}, \mathrm{bn}, \mathrm{ln}\}$. w, illetve w' a különböző statisztikákhoz tartozó súlyok, számításuk a fenti módon történik. Látszik, hogy az összegük minden esetben 1. λ-k a normalizáció tanulható paraméterei.

A módszer előnye, hogy kis batch méret esetén is eredményesen működik, amely esetünkben is fennáll. Ekkor a hibafüggvény és az eredmények az alábbiak szerint alakulnak:

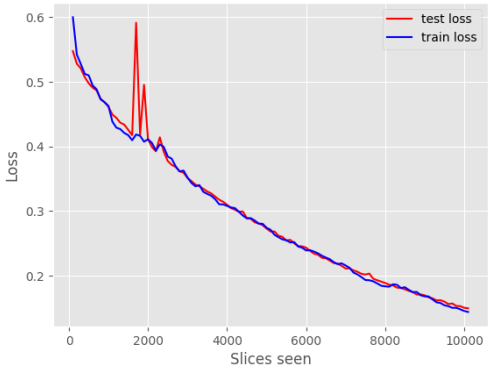


ábra 13: Tanítás switchnorm regularizációval

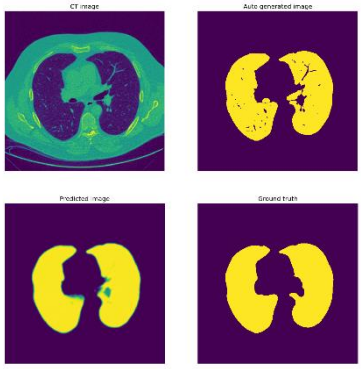


ábra 12:Eredmények switchnorm regularizáció mellett

Ekkor már látható, hogy a hiba nagyon szépen, kis értékre konvergál. Pár tüske adódik tanítás során, ennek oka vélhetően az, hogy az optimalizáló túl magabiztos lesz, és túlságosan erősen módosítja a súlyokat. Végül azt vizsgáltam, hogy mi történik, ha emellé az utolsó blokk kivételével gyenge dropout rétegeket is teszek, ekkor az alábbi eredményekre jutottam:

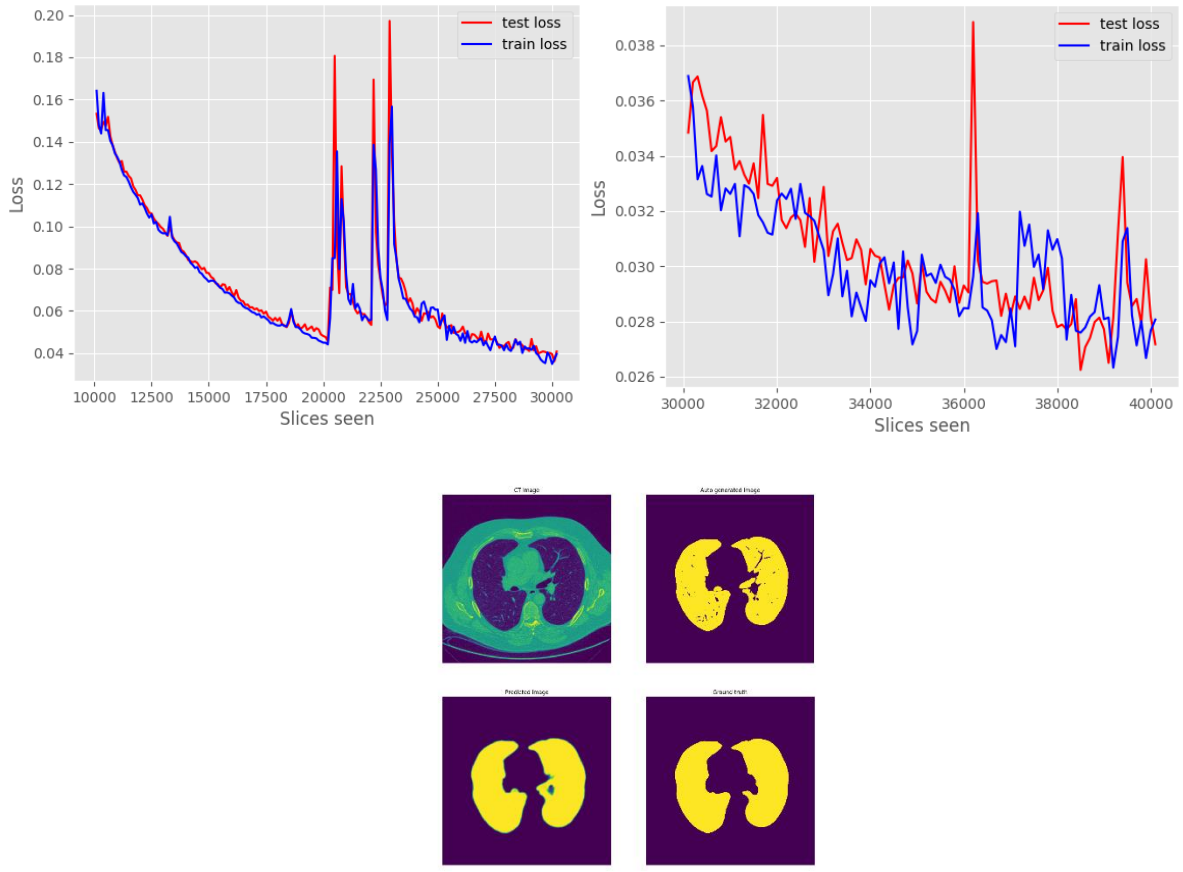


ábra 14: Tanítás switchnorm és dropout regularizációval



ábra 15: Eredmények switchnorm és dropout regularizációval

Megfigyelhető, hogy így tanítás még gyorsabban konvergál, itt már 10000 látott szelet után elérte azt a hibát, amihez sima switchnorm esetén 30000-re volt szükség. Ezt választottam hát a végső architektúrának, és tanítottam tovább, amíg el nem érte a minimumot. A tanítás leállítására külön algoritmust nem alkalmaztam, hiszen amint látható, közben olyan instabilitások vannak a hibafüggvényekben, amire vélhetően bármely ilyen algoritmus túlságosan hamar leállna. Helyette a tanulási folyamat közben szemrevételezéssel figyeltem a hibafüggvények alakulását, és amikor már főként csak a zaj komponens módosította őket (ez kb. 0.03-as értéknél alakult ki) leállítottam a tanítást. Az eredmények alább figyelhetőek meg. Mint látható, a háló kimenete oly módon módosította az automatikus szegmentációt, hogy az abban található „lyukakat” (világos, de a tüdőhöz tartozó képleteket) betömte. Azonban, az automatikus szegmentáció szerinti false positive-ok kitakarásában már nem járt ekkora sikerrel, a hörgőfa detektálva maradt.



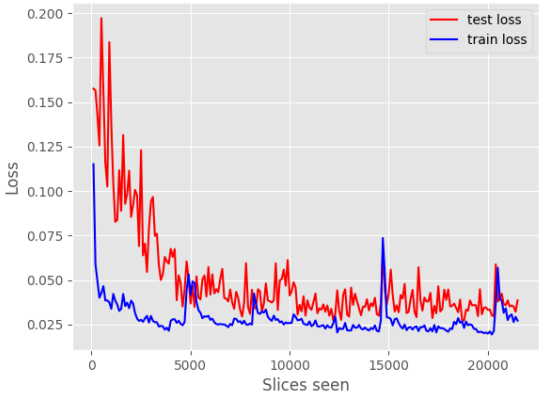
ábra 16: Tanítás switchnorm és dropout regularizációval konvergenciáig

A következőkben tehát a hörgőfa a szegmentációból való eltüntetésén dolgoztam. A feladat két szempontból bizonyult nehéznek. Az adathalmaz nem teljesen konzisztens, hogy hányadik elágazástól, mennyire veszi be az eredményhalmazba a hörgőket, továbbá vélhetően a háló túl erősen veszi figyelembe az automatikusan szegmentált bemenetet, ezáltal nem tudja azt felülbírálni. Ennek orvoslására fokozatosan zajt kevertem az automatikus bementhez remélve, hogy az így generált megnövekedett false positive arány által a háló kevésbé fog rá hagyatkozni. (Egy másik megoldás lehetett volna a bemenet jelentőségének csökkentésére annak fokozatos elhalványítása, azonban abban az esetben esélyes, hogy a háló csak egyre nagyobb súlyokkal vette volna be azt, egészen addig amíg a bemenet teljesen sötét nem lesz, amikor várhatóan nagyon pontatlan kimeneteket kaptunk volna. Weight decay, illetve L1, L2 regularizáció segítségével a súlyok növekedése valamelyest kiküszöbölhető lett volna, azonban így is egyszerűbbnek ígérkezett a bemenet információtartalmának támadása.) A zaj keverésének menete a következő volt:

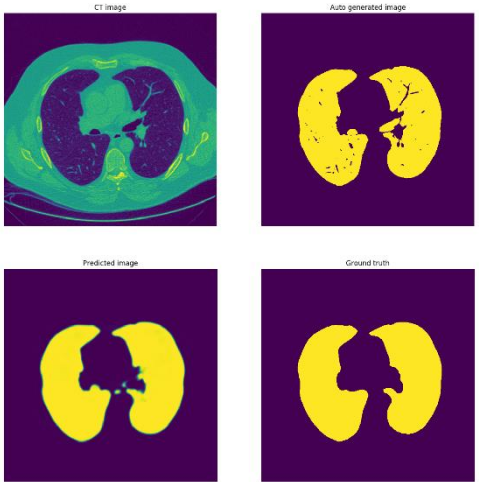
$$Input = \alpha * Randomnoise + (1 - \alpha) * Guide$$

Ahol Input a háló végső bemenete a guide szeletnek megfelelő layeren, α pedig a zaj keverésének bátorsági tényezője, mely fokozatosan növelve volt a zajhoz szoktatás során 0-tól 1-ig. Ezután a hálóból kivágtam az automatikusan szegmentált bemenetet, és tovább tanítottam az eredeti felvételből származó szeletekkel. Meglepetésemre ez a transzfer tanítás rendkívül sikeres volt, az automatikus bemenet eliminálása után a hiba szintje visszatért a korábbi mértékre. Most úgy kapunk 0.03 körüli hibát segítő szelet nélkül, hogy eredetileg a háló anélkül körülbelül használhatatlan volt, továbbá a várt módon a hörgők kiszűrése is sikeresebb volt, bár ez még további munkát igényelt.

ábra 21: Tanítás a zaj növelésével a bemeneten, láthatóan, csak nagyobb tagokra kezdett igazán megugrani a hiba. A kiértékelő halmaz továbbra is a zaj nélküli bemenetet kapta.

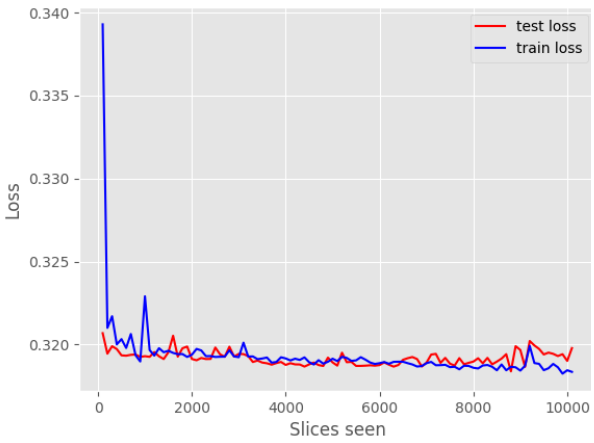


ábra 19: Tanítás az automatikus szegmentáció kivágása után

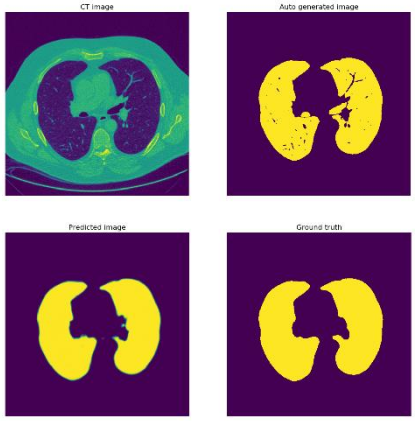


ábra 20: Eredmények az automatikus szegmentáció kivágása után

A false positive-ok kiszűrésére végezetül még büntetősúlyok alkalmazásával próbálkoztam. Ennek lényege, hogy a false positive hibák 10-szer akkora súllyal számítottak a végső hibába, mint a false negative-ok. Ennek alkalmazásával sikerült valamivel jobban visszaszorítanom a hörgők szegmentálását, azonban az eredmény így sem lett tökéletes, valamint hatására a tüdő szélek detekciója is bizonytalanabb lett.



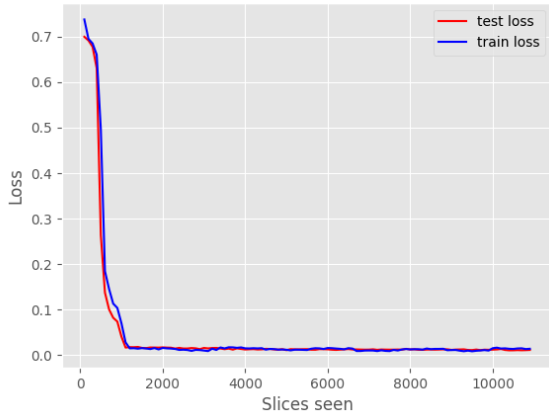
ábra 18: Tanítás büntetősúlyokkal. A háló gyorsan konvergál, azonban a hiba mértéke nem csökken jelentősen. A növekményből láthatóan a hiba nagy része a 10-szeres false positive-ekből áll össze. A tanítás végén némi túltanulás figyelhető meg

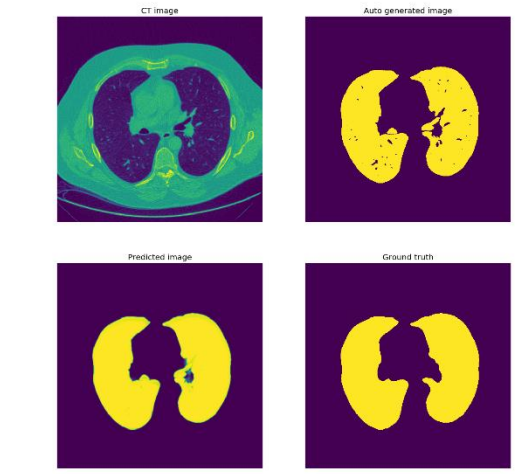


ábra 17: Eredmény büntetősúlyok alkalmazása után

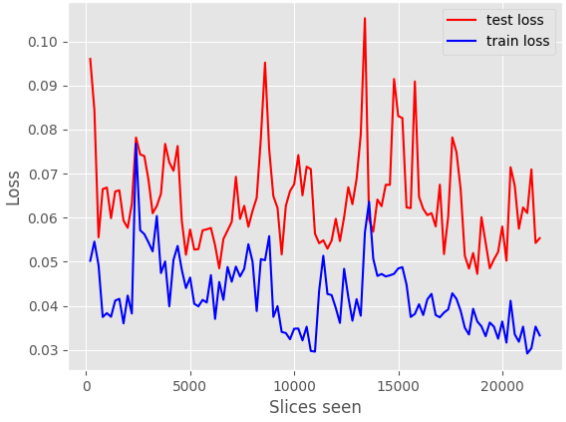
IMSC feladat

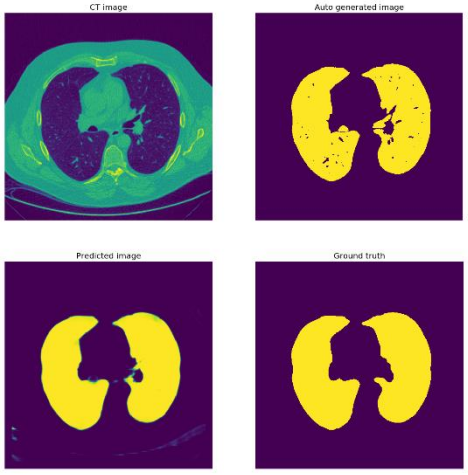
Az imsc feladat során a fenti tanítást elvégeztem egyszerű UNET-re is, ennek felépítése már részletezve lett az Architektúra fejezet ábráján. Mivel ez a háló jóval kevesebb blokkból áll, kisebb memóriát is foglal, ezáltal jelentősen lehetett növelni a batchméretet. Ennek következtében itt regularizációként batchnormmal, switchnormmal, és dropouttal próbálkoztam. Első körben azonban itt is azt figyeltem meg, hogy hogyan teljesít a háló segédszeletekkel, illetve anélkül:





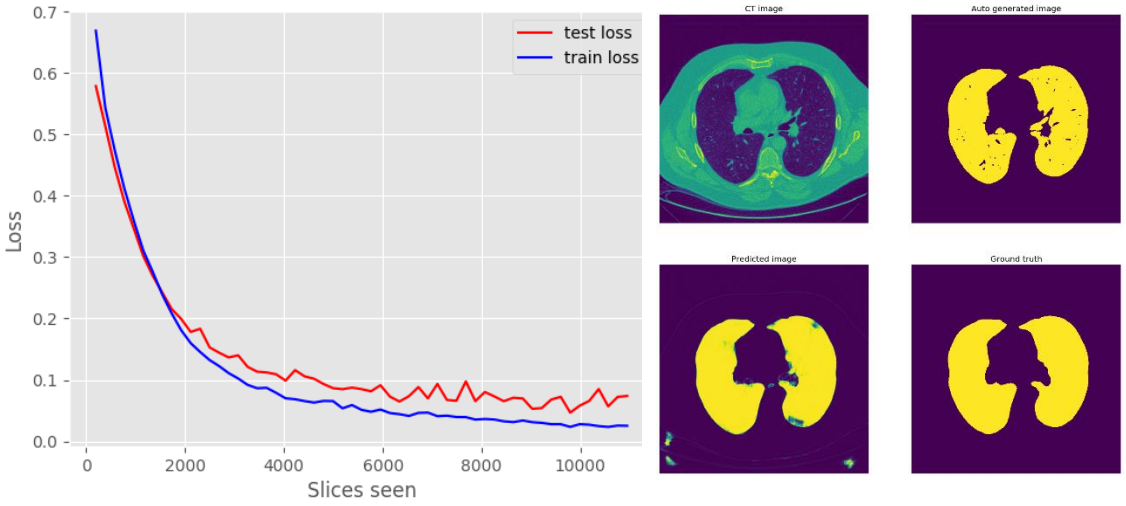
ábra 23: Tanítás guide szeletekkel



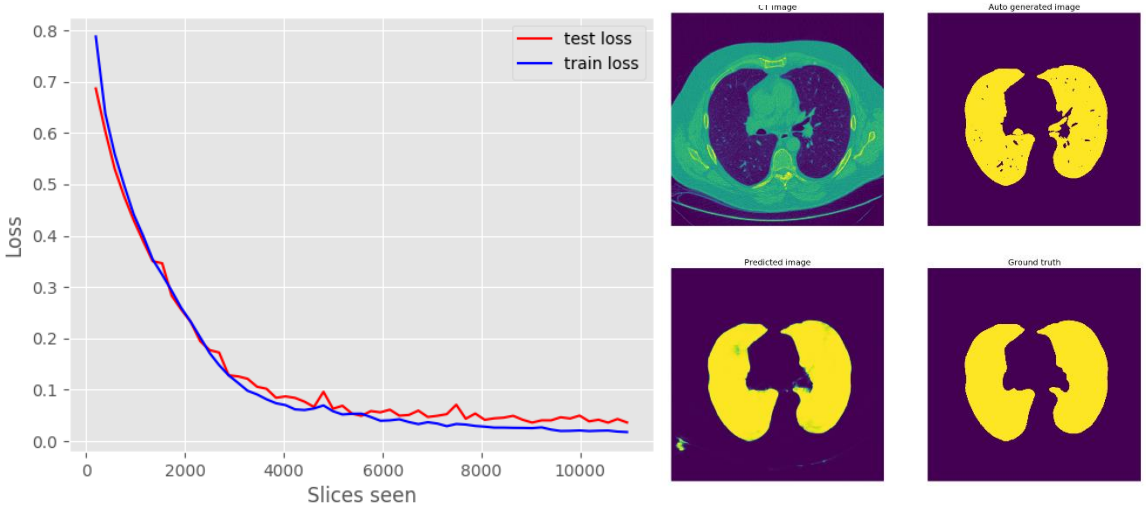


ábra 22: Tanítás guide szelet nélkül

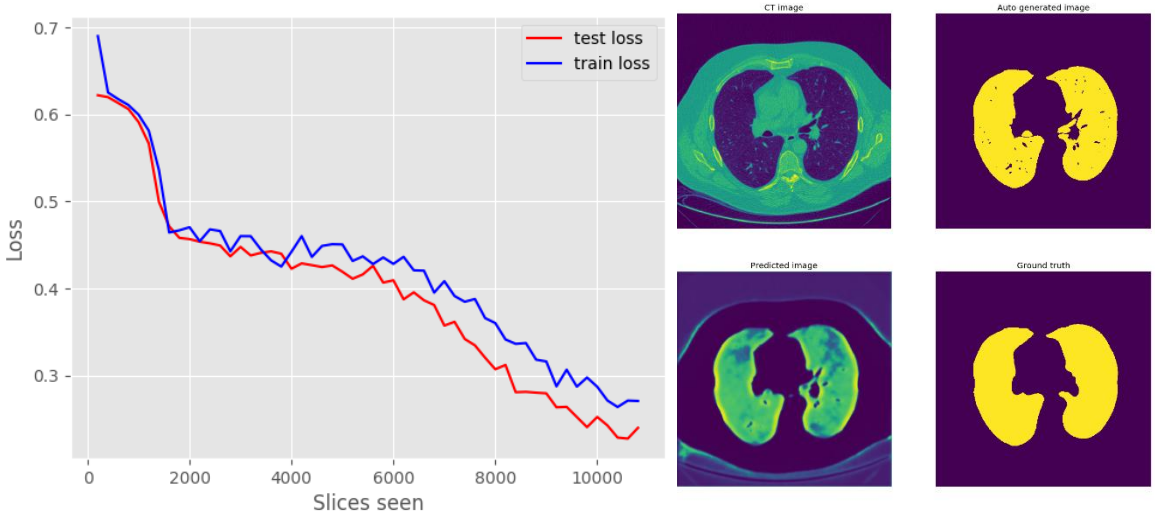
A hibafüggvények grafikonja csalóka, ugyanis azt láthatjuk, hogy az első esetben gyorsan konvergált egy igen alacsony szintre, a második esetben viszont az látható, hogy már az első kiértékelések alkalmával olyan alacsony szintet ért el a hiba, hogy azon később sem volt megfigyelhető jelentős javulás, csak oszcilláció. Ezen gyors konvergencia tulajdonság, illetve amiatt, hogy az eredményképeken megfigyelhető módon sokkal sikeresebb volt a hörgőfa kiszűrése, a guide szelet nélküli verzióval dolgoztam tovább, még úgy is, hogy itt a hiba némileg magasabbnak adódott, főként a háttérben fellelhető fals detekciók miatt. Ezek után a már fent ismertetett regularizációkkal próbáltam javítani a háló teljesítményén. Itt arra jutottam, hogy a dropout esetében a háló túlregularizált lesz, és egy jóval magasabb hibaszintre konvergál, switchnorm, és batchnorm esetén pedig a konvergencia jóval lassabb, de azonos hiba szintre következik be, miközben vizuálisan az eredmények rosszabbnak tűntek a regularizáció nélküli esetnél. Ezek alapján arra a következtetésre jutottam, hogy sima UNET esetén az architektúra approximációs képessége körülbelül megegyezik a probléma nehézségével, így a regularizáció nélküli változatot véglegesítettem.



ábra 26:Tanítás és eredmények batchnorm mellett



ábra 25: Tanítás és eredmények switchnorm mellett

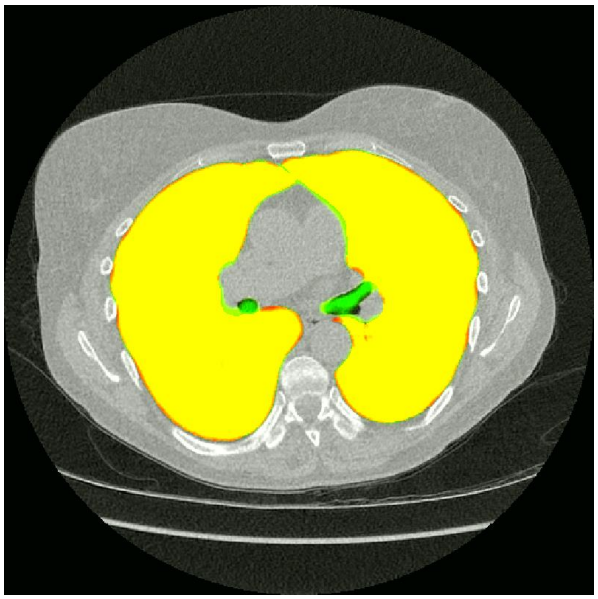


ábra 24: Tanítás és eredmények dropout mellett

Összefoglalás

Eredmények tekintetében mind az alap, mind az IMSC feladat során alkalmazott architektúra jól felismerte a tüdő külső körvonalait, valamint az automatikus szegmentációhoz képest helyesen bevette a világosabb képleteket is az eredményhalmazba. A külső körvonal szegmentációban a sima UNET egy fokkal jobban teljesít vélhetően amiatt, hogy ennek tanítása során nem voltak büntetősúlyok alkalmazva a false positive detekciókra. Meglepő módon azonban a hörgők kiszűrésében is jobbnak mutatkozott összetetteb társánál. Amiben a 2,5D-s módszer jobb volt, az a tüdőn kívüli háttér kiszűrése, valamint a nagyobb térbeli kiterjedésű világosabb képletek helyes szegmentálása. A jövőben egy lehetséges fejlesztési irány lehetne a két háló kimenetére egy új háló tanítása, mely célja, hogy eldöntse, hogy az adott helyzetben melyik háló becslése pontosabb. Várható, hogy ez jól működne, hiszen külön-külön mindkét architektúrának más volt az erőssége.

Az alábbi ábrákon a két háló teljesítménye figyelhető meg a ground truth-hoz viszonyítva. A detektált tüdő zölddel, a ground truth pirossal van jelölve, a kettő metszete sárga színnel látható.



ábra 27: Kiértékelés UNET++-al

ábra 28: Kiértékelés UNET-el

Hivatkozások

Cui, H., Liu, X., Huang, N. (2019). Pulmonary Vessel Segmentation Based on Orthogonal Fused U-Net++ of Chest CT Images. *MICCAI 2019.* Springer, Cham.

Ian Goodfellow and Yoshua Bengio and Aaron Courville. (2016). *Deep Learning.* MIT Press.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 1929-1958.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI 2015.* Springer, Cham.

Ping Luo, Ruimao Zhang, Jiamin Ren, Zhanglin Peng, Jingyu Li. (2019). Switchable Normalization for Learning-to-Normalize Deep Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Sergey Ioffe, Christian Szegedy. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning* (old.: 448-456). PMLR.

Zhou Z., Rahman Siddiquee M.M., Tajbakhsh N., Liang J. (2018). UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *DLMIA ML-CDS 2018.* Springer, Cham.