Szegmentációs technikák

Önálló projekt III.
2021/22, 1. félév

Bakos Bence

Témavezető:
Lukács András

Bevezetés

Az utóbbi években rengeteg eredmény született a gépi látás területén. Ennek a területnek a fókuszában a gépi tanulással megtámogatott képfeldolgozási módszerek állnak. A témakör egy jelenleg népszerű feladata a szemantikus szegmentáció. Többek között az utóbbi években megjelenő új modelleknek köszönhetően jelentős eredmények születtek itt, például orvosi adathalmazok esetén. A projektmunkám keretein belül a félévben a szemantikus szegmentáció feladatával, az erre kifejlesztett új modellek megismerésével és azok orvosi adathalmazokra való alkalmazásával foglalkoztam.

Gépi látás feladatok

A gépi látáson belül többféle feladatot különböztethetünk meg, például a kimenet komplexitása szempontjából. Ezek gyakran egymás általánosításai, és a rájuk kitalált modellek magja is sokszor hasonló, mégis érdemes őket külön vizsgálni.

Classification

A klasszikus gépi látási feladat a klasszifikáció. Ebben a kérdéskörben az inputhoz egy vagy több osztály címkéje tartozik annotációként. A feladat ezeknek a címkéknek a meghatározása.

Object detection

Ebben a feladatban a képen található objektumoknak a pozíciójára is kíváncsiak vagyunk amellett, hogy milyen osztályba tartozik az. Az inputhoz annotációként egy vagy több téglalap alakú „bounding box” van rendelve, melyek körülhatárolnak egy-egy objektumot. A bounding boxokhoz osztálycímke is tartozik.

Semantic segmentation

Gyakori, hogy a képen szereplő objektumok pozíciójának nem csak egy reprezentációjára vagyunk kíváncsiak, hanem pontosabb (pixel-szintű) információra van szükségünk. A szemantikus szegmentációs feladatban az inputhoz egy ún. maszk tartozik annotációként. Ez a maszk az input kép minden egyes pixeléhez hozzárendeli annak osztályát. Tehát

minden pixelre egy klasszifikációs problémát kell megoldanunk, ezért hívják az irodalomban a pixelenkénti annotációval megadott feladatokat gyakran sűrű klasszifikációnak is.

Instance szegmentation

Míg a semantic segmentation nem tesz különbséget egy osztály különböző egyedei között (például, ha egy képen több ember is van), ám gyakran erre is szükségünk van. Amikor egy sűrű klasszifikációs feladatban az annotációban egy pixelhez az osztálya mellett a konkrét példányt is meg van jelölve, amihez tartozik, akkor instance segmentation feladatról beszélünk.

A skálát lehetne tovább finomítani és megkülönböztetni további feladattípusokat ezeken belül, de a legtöbb helyen az irodalomban ezt a csoportosítást veszik alapul.

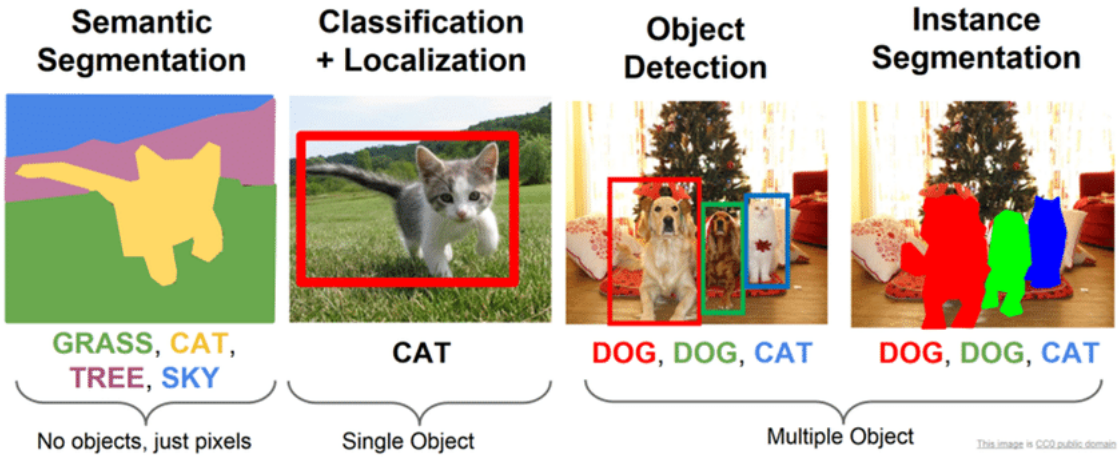

Figure 1: Gépi látási feladatok osztályozása [11]

Semantic segmentation

Az alábbiakban a semantic segmentation problémát mutatom be részletesebben. Adott tehát egy N elemből álló képi adathalmaz: $\{X_i\}_{i=1}^N \subset \mathbb{R}^{C\times H\times W}$. A hozzá tartozó maszkok: $\{M_i\}_{i=1}^N \subset \mathbb{N}^{H\times W}$, ahol M_i értékei az input megfelelő pixelének osztályát adják meg. (Itt C a csatornaszámot, H és W pedig a térbeli méretet jelöli.) Olyan f modellt szeretnénk, amelyre $f(X_i)$ valamilyen értelemben jól közelíti M_i-t.

Metrikák, loss-ok

Egy modell teljesítményét mérő függvény (metrika) és a veszteségfüggvény kiválasztása minden gépi tanulási feladatban fontos szerepet játszik. Most csak azzal az esettel foglalkozok, amikor csupán egy lehetséges osztály van (one-class) és a pixelekről azt kell eldönteni,

hogy beletartoznak-e ebbe vagy pedig a háttérhez (background) tartoznak. Több lehetséges osztály (multiclass) esetén a feladatot úgy oldjuk meg, hogy a maszk és a modell output is $C \times H \times W$ méretű. Az előbbiben egy pixelpozícióhoz tartozó vektor ($M_i^{h,w} \in \mathbb{N}^C$) a (h, w) pozícióban lévő pixel osztályának karakterisztikus vektora. Ekkor a maszk minden csatornájára alkalmazhatjuk a one-class esetet.

Legyen az input $X \in \mathbb{R}^{C \times H \times W}$, a hozzá tartozó maszk $M \in \{0, 1\}^{H \times W}$. Tegyük fel, hogy a modell kimenete $P \in \mathbb{R}^{H \times W}$ alakú és minden $P_{i,j} \in [0, 1]$. Ezt garantálja a modellek végén gyakran alkalmazott pontoknkéni szigmoid függvény.

A klasszifikációs feladatoknál gyakran használt cross-entropy veszteségfüggvényt egyszerűen általánosíthatjuk szegmentációs feladatra, ha pixelenként alkalmazzuk a cross-entropy losst, majd átlagolunk:

$$\text{BCELoss}(P, M) = \frac{1}{H \cdot W} \sum_{i=(h,w)} \alpha M_i \log P_i + (1 - M_i) \log(1 - P_i)$$

Az α paraméterrel segítségével kiegyensúlyozatlan adathalmaz esetén jobban tudjuk súlyozni az alulreprezentált osztályba tartozó pixelek klasszifikációjából származó veszteséget.

Egy érdekes módosítása a klasszikus cross-entropynak a focal-loss [12]:

$$\text{BinaryFocalLoss}(P, M) = \frac{1}{H \cdot W} \sum_{i=(h,w)} \alpha(1 - P_i)^\gamma M_{h,w} \log P_i + P_i^\gamma (1 - M_i) \log(1 - P_i).$$

Az extra tényezők a nehezen eldönthető pixelek veszteségjelét erősítik fel.

A klasszifikációs feladatokat kiértékelő metrikák közül sok a tévesztési mátrix elemeit használja, például a Jaccard-loss vagy a Dice-loss. Itt a bevett módszer az, hogy a $[0, 1]$-beli outputra alkalmazunk egy küszöbfüggvényt és a küszöbparaméter függvényében értelmezzük a tévesztési mátrix elemeit.

Ugyanezzel a módszerrel sűrű klaszifikációs feladatokra is jó metrikákhoz jutunk. Azonban ezek a mérőszámok a küszöbfüggvény miatt nem alkalmazhatóak veszteségfüggvényként, mert nem differenciálhatóak az output szerint. Ugyanakkor a tévesztési mátrix elemei értelmezhetők (általánosíthatók) a küszöbfüggvény nélküli outputra is:

$$TP = M * P,\ FP = (J - M) * P,\ FN = M * (J - P),\ TN = (J - M) * (J - P),$$

ahol $J \in \mathbb{R}^{H \times W}$ a csupa 1 mátrix, a $*$ szimbólum pedig a mátrixok elemenkénti szorzása. Ezeket az értékeket használja például a Tversky loss:

$$\text{TverskyLoss} = \frac{TP + \varepsilon}{TP + \alpha FN + \beta FP + \varepsilon}.$$

Az ε tagra a szélsőséges esetek elkerülése végett van szükség. A Tversky-loss speciális esete megfelelő α és β esetén például a Dice-loss vagy az IoU-loss.

Modellek

Ebben a fejezetben bemutatok pár fontos deep learning modellt, melyeket mind az elmúlt pár évben dolgoztak ki szegmentációra.

Mask R-CNN:

Az object detection vagy segmentation feladatot megoldó korai modellek nagy része ún. region proposal algoritmusokat használ. Ezek lényege, hogy mielőtt klasszifikációt vagy lokalizációt végeznénk, kiválasztunk olyan régiókat a képen, melyek potenciálisan objektumokat tartalmazhatnak, majd ezeket egyesével kielemezzük valamilyen módszerrel. A sikeres R-CNN, Fast R-CNN és Faster R-CNN modellek is ezt a technikát alkalmazzák, miközben egyre nagyobb szerepet játszanak a region proposal részben a konvolúciós rétegek. Utóbbit módosították, hogy kimenetként ne csak bounding box paramétereket, hanem pixelenkénti maszkokat kapjunk a RoI-ken (region of interest) a Mask R-CNN modellben [10].

A modell vázlatosan a következő: az input képre alkalmazunk egy alkalmas feature extraction hálót (pl. Resnet vagy VGG). Az így kapott feature mapeken egy 3x3-as ablakot mozgatunk és minden pozícióhoz hozzárendelünk k ún. anchor régiót, melyek az ablakhoz tartozó részek a képen, transzformálva méret és képarány szerint. Az ablakot egy háló először egy vektorrá transzformálja, majd egy $2k$ hosszú *cls* és egy $4k$ hosszú *bbx* vektorrá. Ezek egy-egy anchorhoz tartozó értékek, és azt fejezik ki, hogy mennyire valószínű, hogy az adott anchor régióban van objektum és pontosan hol helyezkedik el.

A modell ezen részét úgy tanítjuk, hogy minden anchorhoz hozzárendelünk egy olyan bounding boxot az annotációból, amire kellően nagy az átfedés a kettő között. Ha van ilyen, akkor a target cls score 1 lesz és a target bounding box értékek az illeszkedő bounding box paraméterei. Ha nincs ilyen bounding box egy anchorhoz, akkor a target cls score 0.

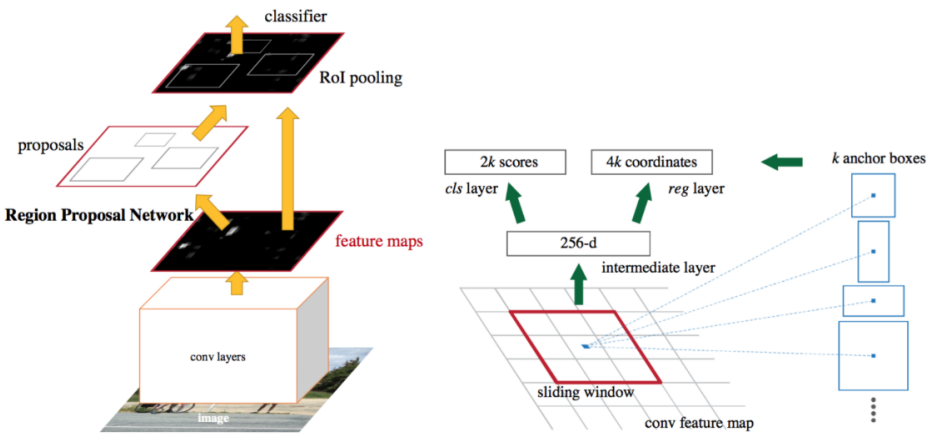

Figure 2: Mask R-CNN region proposal unit [2]

A kapott cls score értékekből N legnagyobbat választva kapjuk a javasolt régiókat. Ezekre leszűkítve a feature mapet alkalmazzuk a modell klasszifikáló/lokalizáló/szegmentáló fejét. A klasszifikáló rész fontos a szegmentációban is. A szegmentációs rész konvolúciós rétegekkel készít maszkot minden osztályhoz, de a veszteségfüggvényben csak a klasszifikáló részt által meghatározott osztályé fog szerepet játszani.

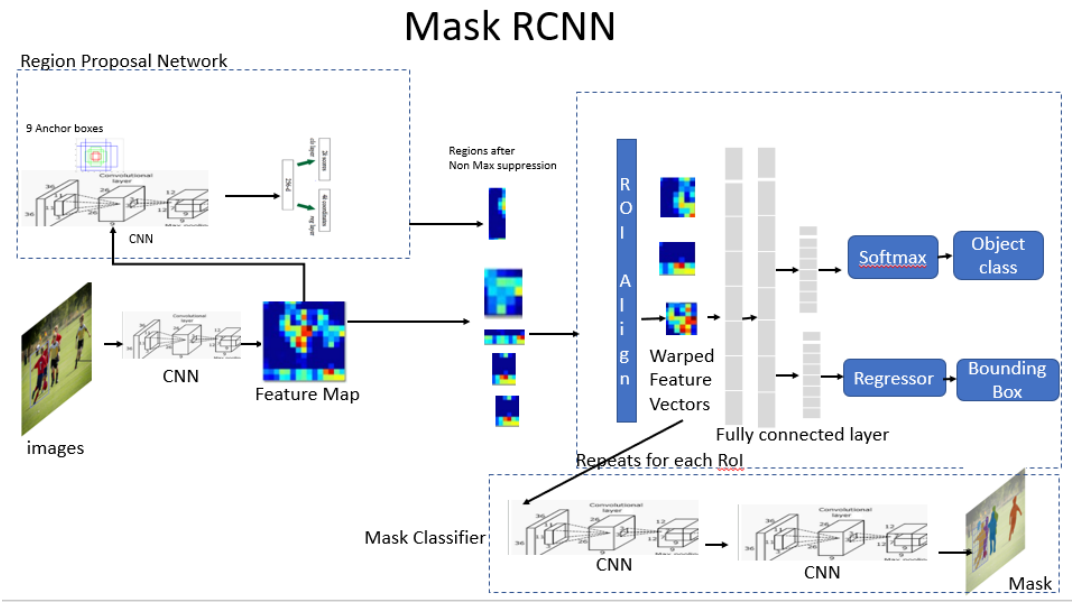

Figure 3: Mask R-CNN modell [1]

FCN

A klasszikus klasszifikációs modellek (VGG, ResNet architektúrák) általában az i-edik rétegben (vagy blokkban) $C_i \times H_i \times W_i$ méretű képekkel dolgoznak, ahol C_i monoton növő, H_i és W_i pedig monoton csökkenő értékek. Egy ponton ezeket a feature-mapaket kilapítják és sűrű rétegeknek feed-elik őket. Ezen sűrű rétegek kimenete adja a predikciót.

Az első probléma egy ilyen háló szegmentációra való alkalmazásakor az, hogy szegmentációs (sűrű klasszifikációs) modellek esetén a kimenetnek is ugyanakkora felbontásúnak kell lenniük, mint az input képeknek. Ezért az első ötlet a H_i és W_i értékek fixen tartása a konvolúciós rétegek során és a sűrű rétegek elhagyása a modell végéről. Egy ilyen modell azonban igen költséges számításokhoz vezet nagy csatornaszámok esetén. Ezt küszöböli ki a FCN modell [13], ami csak konvolúciós rétegeket használ, azonban a modell első felében a H_i és W_i értékek csökkennek, majd a második felében (transpose convolution-öket alkalmazva) ismét nőnek. Hogy kevesebb térbeli információt veszítsünk, a modell első feléből időnként eltároljuk az aktuális réteg kimenetét és hozzáadjuk a modell második felében azon réteg inputjához, amely megfelelő méretű.

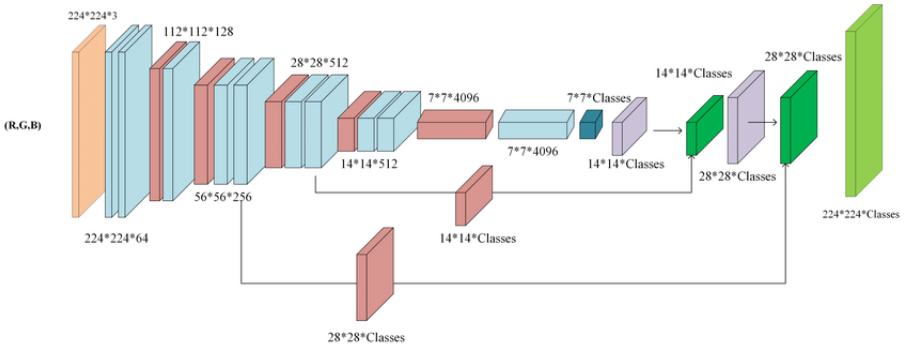

Figure 4: FCN-8 modell [15]

U-Net

Az U-Net modell [16] a korábbi szegmentációs megoldások közül az FCN-re hasonlít a legjobban. A modellt két ágra bonthatjuk, egy encoder és egy decoder ágra, melyek szintekre tagozódnak. Az encoder minden szintjén egy konvolúciós blokk van, aminek outputja egy max-pooling után megy a következő szinten lévő konvolúciós blokkba. A legalsó szinten még egy konvolúciós blokkot találunk, majd felváltva upsampling vagy upconvolution rétegeken és konvolúciós blokkokon át jutunk el a szinteken visszafelé az outputig. Az egyre alacsonyabb szinteken egyre több csatornával, de egyre kisebb felbontású képeket kapunk, ami miatt a pontos térbeli pozíciókról információt veszítünk. Ezt a jelenséget hivatottak ellensúlyozni a modellben az úgynevezett skip connection-ök. Ezek révén a decoder ág konvolúciós blokkjai nem csak az alacsonyabb szintekről érkező, kevésbé részletes térbeli információt tartalmazó adatot kapja meg inputként, hanem az encoder megfelelő szintjén lévő konvolúciós blokk kimenetét is. Az elképzelés az, hogy egyrészt az alacsony szintekről érkező információ pontos képet ad a kép tartalmáról az alsó szinting kinyert feature-ök alapján. Másrészt a skip connection-ön keresztül érkező információ nem ment keresztül ún. spatial bottleneck-en vagyis a pontos térbeli részletek megmaradnak. Ezt a két inputot a csatornák dimenziójában egymás után fűzve a konvolúciós blokk a két inputból képes helyes és pontos információt kinyerni a kép tartalmáról és annak pozíciójáról.

Megjelenése után az alap U-Net modell olyan jól teljesített benchmark szegmentációs feladatokon, hogy sokan kezdtek foglalkozni azzal, hogy miként lehetne javítani ezt az architektúrát. Erre egy példa az Attention U-Net.

Attention U-Net

Az attention mechanizmust először NLP feladatokra alkalmazták. Itt gyakran szekvenciális inputokkal és outputokkal találkozunk. Az output egyes elemeinek meghatározásakor figyelembe kell venni az inputot és az addig előállított outputot is. Előbbiből figyelhetünk egyszerre csak egy elemre a szekvenciából, de általában érdemes többet is számításba venni, vagyis számít a kontextus. Az attention mechanizmus abban segít, hogy a modell milyen

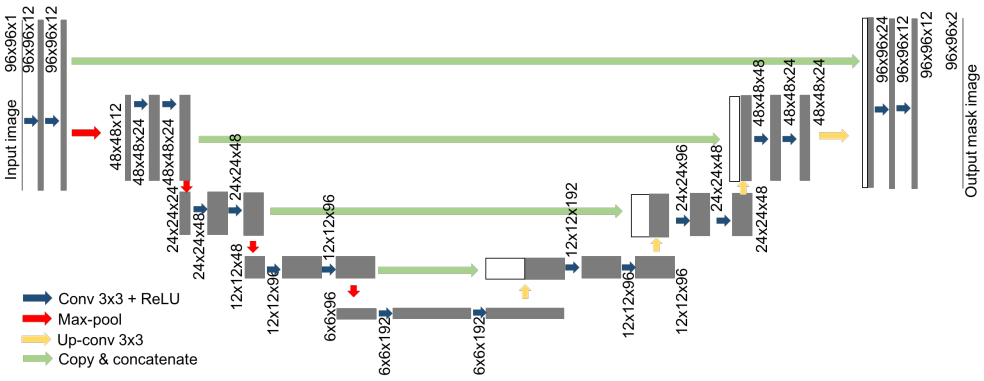

Figure 5: U-Net modell [3]

súllyal vegye figyelembe az input szekvencia egyes elemeit. Mindegyik input elemhez (tanulható módón) hozzárendel az eddigi outputtól is függő score-t, ami alapján egyes részeire az inputnak jobban figyel a modell az aktuális output elem kiszámolásánál.

Azt az ötletet, hogy a predikció során egy modell az inputnak bizonyos területeire több hangsúlyt fektessen azóta sok más területen is sikeresen alkalmazták. Egyik ilyen eredmény az Attention U-Net [14], mely ún. attention modulok segítségével javítja a skip-connection-ök hatékonyságát, miközben nem növeli számottevően a modell méretét a sima U-Nethez képest.

Az attention modulnak (vagy attention gate) két inputja van: az encoder ág megfelelő szintjén lévő konvolúció blokk kimenete (x), valaminet a decoder ág előző szintjén lévő konvolúció blokk kimenete (g). A két inputotra alkalmazunk 1-1 konvolúciós rétegeket (bár eredetileg különbözőek voltak, de ezután a méreteik megegyeznek) melyek eredményét pozíciónként összeadjuk. Erre egy ReLU függvény után egy konvolúciós réteget alkalmazunk, melyet egy sigmoid aktiváció követ. Ezt követően a térbeli dimenzóiók mentén egy upsampling-et hajtunk végre, hogy az így kapott α tensor mérete megegyezzen x méretével. Ezt az α tensort használjuk attention map-ként x-en, vagyis pozíciónként összeszorozzuk a kettőt, x értékeit súlyozzuk α értékeivel. Az így kapott $\hat{x}$ játsza ezután a klasszikus U-Netben a skip-connection-ön keresztül érkező input szerepét a decoder konvolúciós blokkjában.

További U-Net variánsok

További U-Net variánsokat kapunk, ha az encoder és a decoder ágakban a konvolúciós blokkokat módosítjuk. Például a Residual U-Net modell [7] a ResNet hálóarchitektúra alapján residuális blokkokkat használ. Ehhez hasonlóan a rekurrens blokkokat is alkalmazhatunk (R2-UNet [4]).

Ezek a módosítások természetesen együtt is alkalmazhatóak, pl. attention modul és residuális konvolúciós blokk alkalmazása adja az Attention ResU-Net-et. Ezeket és még további U-Net architektúrákat gyűjtöttek össze [17]-ban.

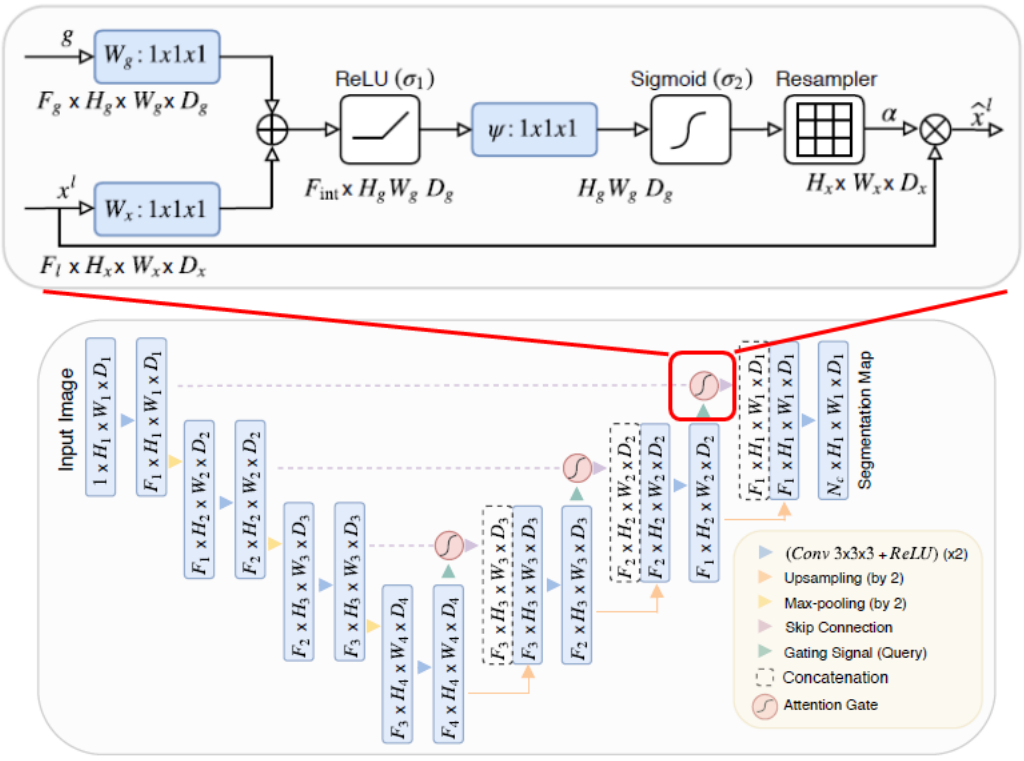

Figure 6: Attention U-Net modell és az attention modul [14]

Transzformerek

Az első Transformer modell 2017-ben jelent meg [18] és nagyon hamar komoly áttöréseket hozott NLP területen. A modell ún. multi-head self-attention blokkokból épül fel. Ezek a blokkok a szöveg kisebb egységeit, például szavakat, reprezentáló vektorok (tokenek) egymáshoz való viszonyát határozzák meg, így a modell automatikusan figyel a globális kontextusra.

A nyelvfeldologzásban mutatott figyelemre méltó eredmények miatt sokat próbálták transzformer modellt átültetni képi domainre, ez azonban nem könnyű feladat. A transzformer multi-head attention blokkjai a bemenetként kapott tokenek közötti viszonyt modellezi és emiatt négyzetes műveletet végez az input tokekek számában. Ez NLP területen elfogadható, de ha egy kép minden pixelét tekintjük egy tokennek, akkor hatalmas számításigényt kapunk. Ezt küszöbölték ki a szerzők a híres Vision Transformer cikkben (ViT [8]) azzal, hogy a bemeneti képet kisebb patch-ekre bontották és ezeket a patcheket adták inputként egy transzformernek egy lineáris elkódoló réteg után.

A ViT modellt klasszifikációs feladatra fejlesztették, de részben ennek a hatására sorra jelentek meg újabb transzformereket alkalmazó modellek más computer vision feladatra is,

pl: [5, 6, 19]). Ezek között akad számos szegmentációra fejlesztett modell is. Látva ezek teljesítményét különböző benchmark adathalmazokon midenképpen érdemes transzformer alapú szegmentációs modellekben is gondolkodni.

Rákszegmentáció hisztopatológiai tüdőfelvételeken

A megismert modelleket, módszereket és technikákat egy valós orvosi adathalmazon próbáltuk ki. Az adathalmaz a Korányi Pulmonológiai Központból származó mikroszkópos felvételeket tartalmaz. A felvételeken egészséges és rákos tüdőszövetek láthatóak hematoxylin és eosin festékanyaggal (H&E) megszínezve. A felvételekhez egy szakorvos annotációkat készített, melyeken megjelölte az adott felvételen mely területeket érint a rákos megbetegedés (Figure 7).

A felvételeken elméletileg három különböző ráktípus fordulhat elő, és ezek az annotációban is meg vannak különböztetve. Azonban mivel egyelőre csak az egyik típusból áll rendelkezésre megfelelő mennyiség, így egyelőre nem foglalkoztunk a többosztályos problémával. A feladatunk tehát első körben az, hogy meghatározzuk azt a területet a szöveten, melyet ez a ráktípus érint.

A teljes slide-ok túl nagyok (4096x4096 pixel), hogy azokon tanítani tudjunk, így először kisebb patcheket (512x512) vágtunk ki. Véletlen szöggel történő elforgatás mint augmentáció után ezekből vágtunk ki 256x256 méretű tanító adatot. A validációs halmazon nem augmentálunk.

Az első mérések során klasszikus U-Net modellt teszteltünk súlyozott binary cross-entropy veszteségfüggvénnyel, osztálykiegyensúlyozás nélkül az adatban. Az eredmények azt mutatják, hogy a mivel az egészséges pixelek nagyon felülreprezentáltak (Figure 8), így a modell rátanul a csupa 0 kimenetre. Ez még nagyon nagy súlyozással módosított veszteségfüggvénnyel is megtörténik.

A mérések folytatásaként először az adathalmaz kiegyensúlyozásával foglalkozunk oversamplinggel (pozitív adatpontok többszörözése) és undersamplinggel (negatív adatpontok számának csökkentése). Ezután áttérnénk összetettebb modellek és más veszteségfüggvények tesztelésére.

Továbbiak

A munkát szakdolgozat formájában szeretném tovább folytatni. A legjobb eredmény elérése érdekében a bemutatott adathalmazon valószínűleg új, módosított modellarchitektúrákat is érdemes kidolgozni. Mivel a transzformer alapú szegmentációs modellek talán a legújabb és legígéretesebb irány, így erre a területre szeretnék nagyobb hangsúlyt helyezni. A munka során kialakuló esetleges új modelleket és technikákat ki szeretném próbálni ismertebb benchmark adathalmazokon is, hogy képet kapjak azok széleseb körben való felhasználhatóságáról.

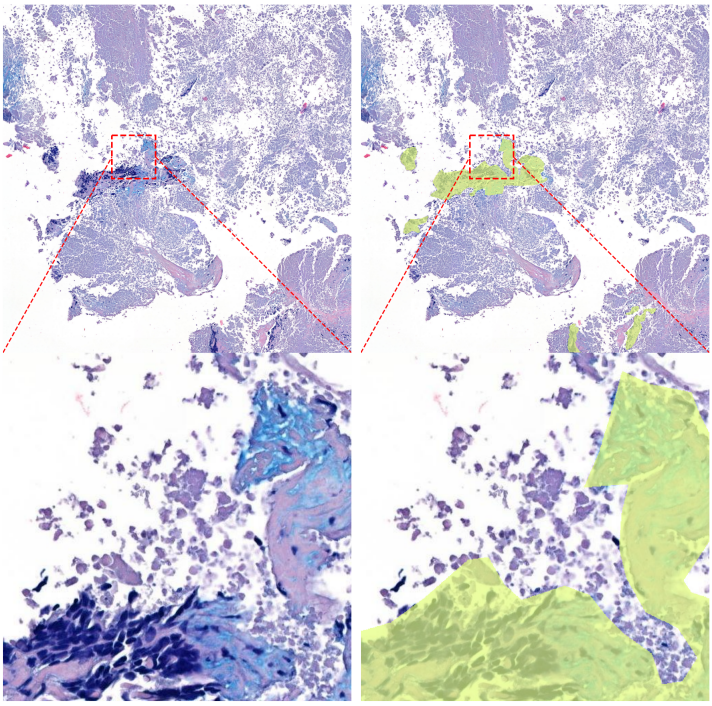

Figure 7: Fent: Egész slide-ok (4096x4096), lent: kivágott patchek (512x512), bal: eredeti kép, jobb: annotáció

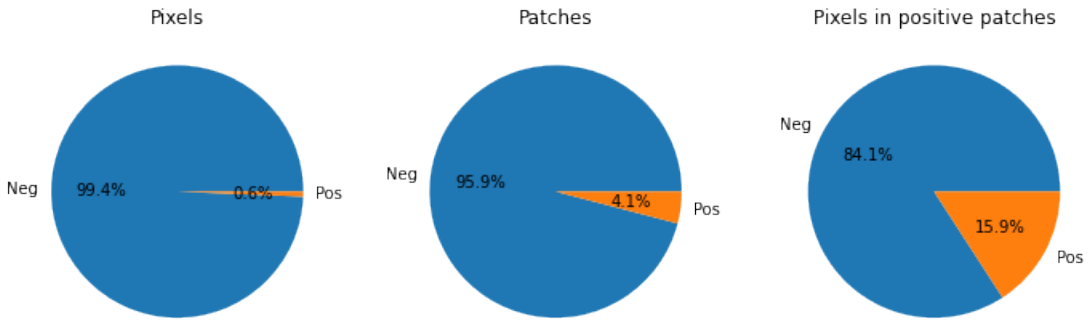

Figure 8: Beteg és egészséges pixelek és patchek arányai

Bibliography

[1] Computer vision: Instance segmentation with mask R-CNN | by renu khandelwal | towards data science. https://towardsdatascience.com/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1. (Accessed on 12/06/2021).

[2] Faster R-CNN explained for object detection tasks | paperspace blog. https://blog.paperspace.com/faster-r-cnn-explained-object-detection/. (Accessed on 12/06/2021).

[3] Using U-Net deep convolutional neural networks to segment ventricle from MR image (tensorflow). http://jidan.sinkpoint.com/MRI_ventricle_UnetCNN/. (Accessed on 12/06/2021).

[4] Md Zahangir Alom, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. Nuclei segmentation with recurrent residual convolutional neural networks based U-Net (R2U-Net). In *NAECON 2018-IEEE National Aerospace and Electronics Conference*, pages 228–233. IEEE, 2018.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[6] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. TransUNet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[7] Foivos I Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114, 2020.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[11] Fei-Fei Li, Justin Johnson, and Serena Yeung. Detection and segmentation. `cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf`.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[14] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention U-Net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[15] Sankaranarayanan Piramanayagam, Eli Saber, Wade Schwartzkopf, and Frederick W Koehler. Supervised classification of multisensor remotely sensed images using a deep learning framework. *Remote Sensing*, 10(9):1429, 2018.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[17] Nahian Siddique, Paheding Sidike, Colin Elkin, and Vijay Devabhaktuni. U-Net and its variants for medical image segmentation: theory and applications. *arXiv preprint arXiv:2011.01118*, 2020.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021.