Fáziskontraszt mikroszkóp alapú sejt szegmentáció és klasszifikáció

Önálló laboratórium dokumentáció

# Az adathalmaz bemutatása

Az adathalmazunk fáziskontraszt mikroszkóppal készített képeket tartalmaz sejttenyésztő lemekről. A sejttenyésztő lemezeink vagy nem kezeltek, vagy fibronektinnel vannak bevonva. A fibronektin elősegíti a sejtek tapadását, migrációját, növekedését és differenciálódását. Erre a lemezre kerülnek rákos sejtvonalból sejtek, több százas nagyságrendben. Aztán a fáziskontraszt mikroszkóppal elkészítjük a képeket, amelyek szürkeárnyalatúak és nagyjából 4000x4000-es méretűek lesznek.

A felhasznált sejtvonalak:

* HeLa
* LCLC
* MCF7
* MDAMB231
* H838
* HepG2

|  |  |
| --- | --- |
| C:\Users\Asrock\AppData\Local\Microsoft\Windows\INetCache\Content.Word\dataset0.png |  |

Képek az adathalmazból.

Az adatok úgy lettek struktúrálva, hogy a gyökérkönytárban van két mappa: **fibronectin** és **noncoated**. Itt kerül meghatározásra, hogy be lett-e vonva a sejttenyésztő lemez fibronektinnel vagy sem. Ezekben a mappákban további mappák vannak **dátum\_sejtvonal(ak)\_[fn vagy nonc]** formátumban. A dátum ebben az esetben a mérés dátumát jelöli, a sejtvonalak értelemszerűen azt, hogy a mérésben milyen rákos sejtvonalakat használtak, ezután csak a fibronektin használatának vagy ennek hiányának megismétlése van. Ezekben a mappákban számunkra az **azonosító\_seg.npz** formátumú fájlok relevánsak. Az azonosító ebben arra utal, hogy melyik sejttenyésztő lemezről készült a felvétel.

*Példa egy felvétel útvonalára: /fibronectin/20210526\_MCF7\_fn/A1\_seg.npz*

Ez az adatstruktúra egy NumPy-val elmentett Python szótár, ami a mérés eredményeit tartalmazza, ennek kulcsai, és azok tartalma:

|  |  |
| --- | --- |
| **Kulcs** | **Tartalom** |
| well | bioszenzoros kép (nem használjuk) |
| im\_markers | mikroszkópos annotáció |
| im\_mic | mikroszkópos kép |

(Az adott sejttenyésztő lemezről készült még idősoros felvétel bioszenzor használatával, ennek kiértékelése nem volt feladat.)

A mikroszkópos felvételek még fel lettek annotálva, amely azt jelenti, hogy be lettek jelölve a képen a sejtek. Ezt az információt egy olyan két dimenziós tömb tartalmazza (im\_markers), amelynek mérete megegyezik a kép dimenzióival. Ha a tömb eleme 0, az azt jelenti, hogy az adott pixel nem tartozik sejthez, ha ettől különböző, akkor a szám a sejt azonosítóját jelöli.

|  |  |
| --- | --- |
|  |  |

Az előző képekhez tartozó annotációk.

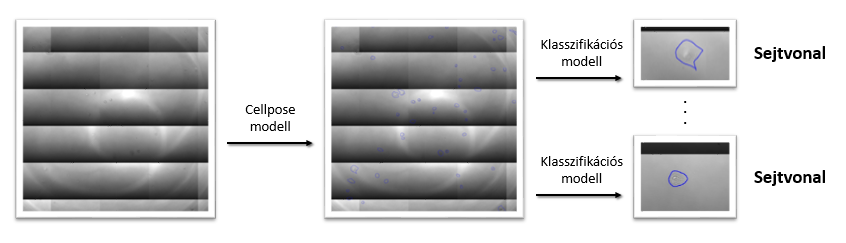
Fontos még megemlíteni, hogy a képek különböző méretűek is tudnak lenni, ezzel fontos foglalkozni a későbbiekben, mivel a hálózatok azonos méretű tenzorokat várnak bemenetként.

# A projekt célja

Ebben a dolgozatban a megvalósított munka célkitűzése az volt, hogy képesek legyünk ezzel az adathalmazzal egy globális szegmentációs modellt készíteni. Ez azt jelenti, hogy egy ilyen lemezen vannak sejtjeink, nem feltétlenül egy sejtvonalból (akár vér). Egy olyan modellt kell készíteni, amely bemenetként megkap egy ilyen mikroszkópos képet, és kimenetként elkészíti a fenti annotációt, minnél nagyobb pontossággal. A nehézség ebben, hogy a tanítóadataink egy lemezen csak egy sejtvonalból tartalmaznak sejteket, azonban a modellünknél ezzel a kezdeti feltételezéssel nem élünk, egy cellán több fajta sejt is lehet.

Miután elkészítettük ezt a szegmentációt, klasszifikálnunk kell az egyes sejteket. Ez azt jelenti, hogy ebből a nagyfelbontású képből az annotáció alapján kivágjuk az egyedi sejteket, ez lesz a klasszifikációs modellünk bemenete, a kimenet egy valószínűségi eloszlás, amely az egyes sejtvonalakat tartalmazza. Tehát a modell a sejt tulajdonságai alapján megpróbálja minnél nagyobb pontossággal besorolni egy sejtvonalba a sejtet.

Azonban a modelleinket külön is tudjuk használni, ennek főleg a szegmentációs modellnél lehet szerepe. Például lehet arra is használni a modellt, hogy egy lemezen megszámoljuk a sejteket, amely számos kísérletnél lehet fontos szerepe, a kimeneti jel normalizációjánál. Azonban a klasszifikációs modellnél is elképzelhetőek egyedi felhasználási lehetőségek a jövőben.



A kiértékelés menetének ábrázolása.

# Az adatok előkészítése

Az adatok előkészítésén azt értjük, hogy valahogy fel kell dolgozni úgy az adatokat, hogy a modell számára felhasználhatóak legyenek, és minnél jobban segítsék a modell pontos tanulását. Az első és legszembetűnőbb problémát a képek méretbeli eltérése jelentette. Erre többféle megoldás is létezik: lehet átméretezni (rescale) a képeket, vagy kiegészíteni (például fekete pixelekkel). Problémafüggő, hogy melyikkel érhetünk el jobb eredményeket, érdemes mindegyiket kipróbálni és összehasonlítani.

Általában egy jó irány szokott lenni az adatkiegészítés, amelynek az a lényege, hogy lehetővé teszi a maximális valószínűség becslést hiányos adatokból. Például forgatásokkal, tükrözésekkel transzformáljuk az adatokat. Ez a technika is felhasználásra került ebben a projektben.

Idáig eljutva már értjük az adathalmaz felépítését, képesek vagyunk úgy feldolgozni az adathalmazunkat, hogy már a neurális hálózatok számára is használhatóak legyenek és az adatkiegészítést is elvégeztük. Azonban az egyes hálózatok tanításánál még felmerülnek nem várt problémák, amelyekre majd megoldást kell találni (de erről majd a későbbiekben).

# A projekt struktúrája

A projekt kialakításánál elsősorban a szegmentáció volt meghatározó, ezzel kezdtem a munkát. Itt fontosnak tartottam, hogy több féle metodikát is ki lehessen értékelni az általam elkészített rendszerrel, többféle metrika alapján. Ez csak egy keretrendszer, nem tartalmazza a konkrét háló tanításokat, azokat Jupyter notebookokban csináltam Colabban. Ennek implementációja az **src** mappában található.

A **loader.py** fájlban található az adatbetöltés implementációja, ezt PyTorch segítségével készítettem el. Az adathalmaz szerkezetét felhasználva megkerestem az összes **\_seg.npz** végződésű fájlt, hiszen ezek tartalmazzák az összes szükséges adatot. Megírtam a **ClassificationDataset** osztályt, amely a PyTorch datasetből származik le. Ennek az a lényege, hogy felhasználásával lehetőségünk van batchenként betölteni az adatokat (tehát nem az összes képet töltjük be a memóriába, hiszen be se férne). Ennek paraméterei a **search\_dir** és a **same\_size\_transform**: a search\_dirrel adjuk meg a gyökérkönyvtárat, amiben megkeresi a \_seg.npz fájlokat az algoritmus, a same\_size\_transform pedig egy objektumot vár, amelynek apply metódusa egy egységes méretre transzformálja a mikroszkópos képet és annotációt (jelenleg egy ilyen módszer van implementálva, a **PaddingTransform**, amely fekete pixelekkel tölti ki a rendelkezésre álló helyet).

Ebből a PyTorch DataSetből célszerű csinálnunk egy DataLoadert, amelyet a tanításhoz használhatunk, ez már ténylegesen tartalmazza a batch méretet. Itt érdemes megkeverni az adatokat (shuffle).

Mivel az adathalmazunk többek között tartalmazta a bioszenzoros képeket is, amelyre nekünk nem volt szükségünk, ezért sokkal több memóriára van szükség a beolvasáshoz, mint amennyi amúgy ténylegesen kéne. Ennek megoldására kompresszáltam az adathalmazt, amelynél igazából csak annyi történik, hogy egy másik mappában újra létrehozza a seg fájlokat, de ezek már csak a szükséges adatokat tartalmazzák (im\_mic és im\_markers). Így nagyon lényeges javulást sikerült elérni az adathalmaz méretét illetően (negyedére csökkent).

A **metrics.py** fájlban vannak definiálva a metrikák, amelyek alapján kiértékelhetőek a modellek. Egyelőre a pontosság és a lefutási idő van meg. Ebben az esetben a pontosság az IoU-t jelenti (Intersection over Union), ennek az a lényege, hogy veszi a konkrét annotáció és a prediktált annotáció metszetét (ezeket találta el a modellünk), és ezt leosztja ezek úniójával. Így egy százalékos értéket kapunk.

Az **evaluation.py** fájlban van megírva a konkrét kiértékelési logika. Megadható egy adatbetöltő, amely a teszt adatainkat tölti be (ezen hajtjuk végre a kiértékelést). Továbbá megadhatók a metodikák, amelyeket kiszeretnénk értékelni, és az előbb definiált metrikák. Az összes modellünkön kiértékeli a tesztadatokat és kiszámolja a metrikákat (átlagolja az összes teszt adatunkra). A kimenetünk egy Python szótár lesz, amely így érhető el: *result[metodika neve][metrika neve]*.

A **visualization.py** fájlban vannak a megjelenítésért felelős függvények. Ezek matplotlib segítségével vannak megírva. Egyelőre egy input batch-et lehet megjeleníteni, továbbá ezek mellett még a predikciókat is meg lehet jeleníteni egy külön függvénnyel.

A **methods** mappában vannak definiálva a szegmentációs metodikák, itt egyelőre van egy WaterShed-es megoldás (a következő fejezetekben bővebben) és egy cellpose-os megoldás (csak az alap cyto3-as cellpose modellt használja).

Csináltam még egy **colab.ipynb**, amelynek az lett volna a lényege, hogy segít ezt a keretrendszert felvinni Colabba, hogy használhatóak legyenek az implementált szolgáltatások. Ezt sikerült is megcsinálni, azonban ez nem bizonyult elég robusztusnak, így a háló tanításokat már külön notebookokban csináltam. Véleményem szerint a keretrendszer inkább akkor használható ergonómikusan, ha saját környezetet használunk, a Colab eléggé fejfájóssá tette a dolgokat.

# Watershedes szegmentáció

Az **src/methods/watershed.py** fájlban implementáltam a WaterShed algoritmust használó szegmentációt. Ennek lényege, hogy különböző képfeldolgozó eljárásokkal megpróbáljuk különválasztani az előtér és háttér tartalmakat (sure\_fg, sure\_bg, unknown). Ezután Connected-component labeling és WaterShed algoritmusok segítségével meghatározható az annotáció. Ennek az a legfőbb problémája, hogy nem robusztus. Annyi féle bemeneti konfiguráció lehetséges, hogy nagyon nehéz egy egységes eljárást készíteni, amely jól szét tud választani, márpedig ettől függ a pontosság. Valamint itt nincs is tanítási fázis, nehéz iterálni egy adott eljárást, valamint eldönteni, hogy jó irányba haladunk-e. Próbálkoztam, és volt egy olyan eljárás, ami viszonylag jól szegmentálta a képeket, amiket teszteltem, viszont egy jópár bemenetnél rosszul teljesített. Viszont tanulási szempontból jó volt, sikerült megérteni jópár képfeldolgozási algoritmust és a WaterShed működését is elsajátítottam. A neurális hálózatok lényegesen jobban szerepeltek már szemlátomást is szinte minden képen. Viszont azt nem szabad elfelejteni, hogy a WaterShedes szegmentáció sokkal kevésbé erőforrásigényes. Erőforrásszegény környezetekben esetleg használható, viszont eléggé megbízhatatlan.

|  |  |
| --- | --- |
| https://docs.opencv.org/4.x/water_coins.jpg | https://docs.opencv.org/4.x/water_fgbg.jpg |

Érméket tartalmazó példában előtér és háttér szétválasztása.[[1]](#endnote-1)

# Cellpose szegmentációs modell

Mivel az adathalmazunk viszonylag kicsi, ezért az úgynevezett transfer learning technika mellett döntöttem a szegmentáció esetében. Ennek az a lényege, hogy kiindulunk egy nagyobb adathalmazon betanított modellből, ami illeszkedik a problémánkhoz. Ez a modell már a problémánkhoz illeszkedő jellemzőket szed ki az adatunkból (feature extractor). Ezeket a paramétereket lefagyasztjuk és ezekkel a jellemzőkkel tanítunk egy saját modellt.

Egy ilyen modell a Cellpose, annak is a cyto3-as modellje, amely kiválóan illeszkedik a problémánkhoz. Nagy, diverz adathalmazon tanították, hasonló képekkel. Tehát a globális szegmentáció probléma megoldásához tökéletes lesz.

Általában a szegmentációs modellek Unet-hálózat alapúak, nevéhez hűen egy U alakú architektúra, a lényege, hogy először konvoluciós és max-pooling rétegekkel csökkentjük a kép dimenzióját, aztán dekonvoluciós rétegekkel skálázzuk fel, hogy a végén egy maszkot kapjunk. A Cellpose is ilyen architektúrát követ, de először egy flow-mezővé alakítja a tanító-maszkokat és ezeket többek között ezt is prediktálja (még egy tartalmazási maszkot)[[2]](#endnote-2). Ezeket a kimeneteket felhasználva már előállítható a fent bemutatott formátumú maszk.

A tanítást a **cell\_segmentation/cellpose\_finetuning.ipynb** notebookban valósítottam meg. Először letöltjük a drive-ból az adatokat, ezeket kicsomagoljuk. Ezután implementáltam a saját PyTorch datasetet, hogy utána könnyen végigiterálhassak az adatokon (bár a cellpose a tanításhoz nem ezt használja).

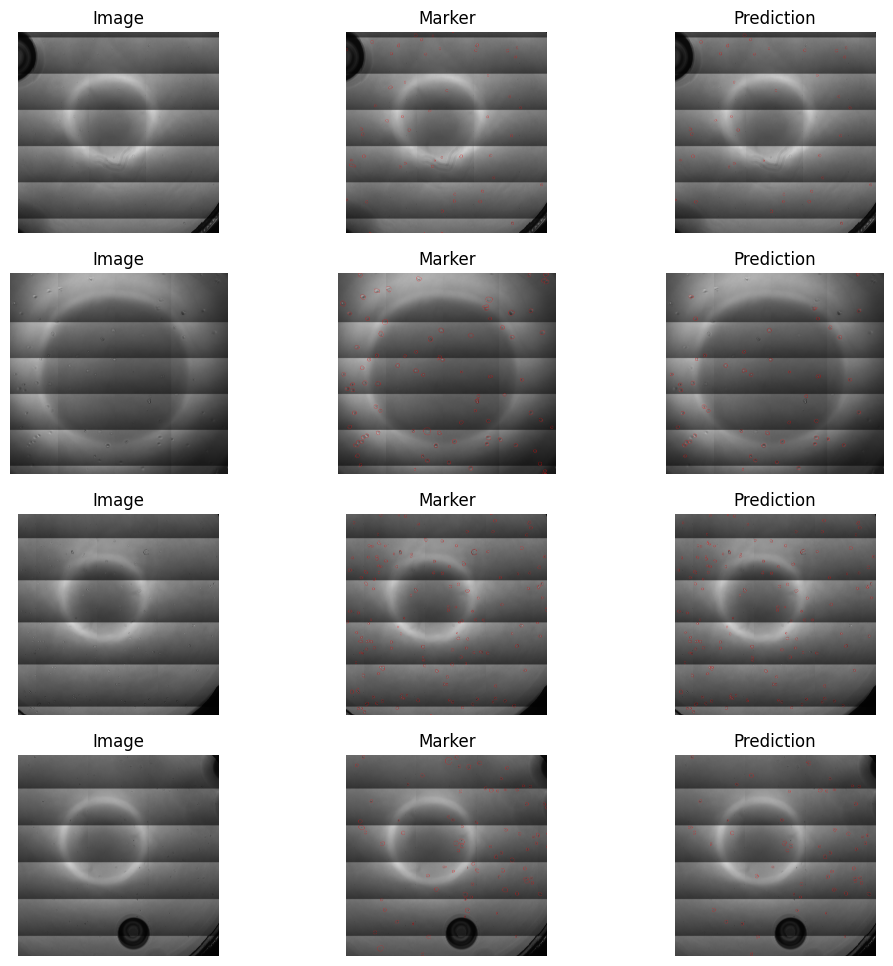
A tanítás nehézségét az adta, hogy a Cellpose nem dataloaderrel dolgozik, tehát nem batchenként tölti be a memóriába az adatokat, és azokat dolgozza fel. Hanem az egész tanító adathalmazt behúzza a memóriába, azonban ez még pár kép esetében jól működik, azonban a mi képméret-képmennyiség paraméterünknél már ez nem kivitelezhető. Erre azt a megoldást találtam ki, hogy iterálva tanítom a modellt. Ezt azért gondoltam működőképesnek, mert több forrásban is láttam, hogyha érkezik egy pár új kép, akkor rögtön betanítják velük a modellt.

Ezért 20 részre osztottam a tanító adathalmazomat, ezekkel kialakítottam egy a cellpose által elfogadott mappaszerkezetet, amelyből már magától el tudja végezni a tanítást. Lényegében ebből állt az előfeldolgozás, minden mást megold a cellpose (Rescalel, hogy azonos méretűek legyenek a képek, stb…)

|  |  |
| --- | --- |
| **Epochs** | 100 |
| **Learning rate** | 0.1 |
| **Weight decay** | 0.0001 |
| **Training iterations** | 20 |
| **Channels** | [0, 0] – szürkeárnyalatú képek |
| **Optimizer** | SGD |

A modell tanításához használt paraméterek.

Persze még a tanító paramétereket meg kellett adni, itt optimalizálást nem csináltam, az ajánlott értékekkel dolgoztam.



A már finomhangolt modell által prediktált maszkok, és az eredetiek.

Azonban ez a modell már jóval komplexebb, mint az előbbi egyszerű WaterSheden alapuló megoldás. Ez a futásidőben is látszik, nyersen CPU-val már lehetetlen volt akár egy kép maszkját is meghatározni megfelelő időn belül, ezért itt már mindenféleképpen GPU-val kell dolgozni.

|  |  |
| --- | --- |
| Dice érték | 0.66 |
| Lefutási idő | 10.58 másodperc |
| Deketálási ráta | 0.634 |

Átlagos metrikák a teszt adathalmaz kiértékelésénél.

A kiértékelési adatoknál látszik, hogy amelyik sejteket megtalálja a modell, azokat viszonylag nagy egyezéssel találja meg. Azonban a deketálási ráta nem kielégítő, ezen még lehet finomítani, azonban a tanító adatok minőségét és mennyiségét figyelembevéve ez sem tragikus.

# ResNet50 klasszifikációs modell

A metodika következő lépése a klasszifikációs modell betanítása volt. Itt már az adatok előkészítése nagyobb figyelmet igényelt. Az adathalmazunk mikroszkópos képeket és ezek annotációs maszkját tartalmazza. Ezekből kell előállítanunk az egyedi sejtes képeket és az ezekhez tartozó osztályokat.

Ehhez úgy dolgoztam fel a képeket, hogy vettem egy képet és meghatároztam az összes egyedi sejt középpontját. Ezt úgy tettem, hogy az adott azonosítójú sejtnek kiátlagoltam a koordinátáit. Azonban a sejtek méretei különbözőek lehetnek, ezt is számításba kell venni, lehetőleg úgy, hogy a sejt teljesen szerepeljen a képben. Ehhez meghatároztam a középponttól vett maximális eltérést x vagy y irányba. Ezek lesznek a tartalmazó téglalap féloldalai. Ezután elkészítettem a tartalmazó téglalaphoz tartozó maszkot (azaz 1-est tartalmaz, ha a kimeneti kép eleme, különben 0-t). Ez a maszk alapján kivágtam a képből a megfelelő egyedi sejtet és ezt elmentettem egy képként egy külön mappában. Miután ezt az összes maszkra megcsináltam, tömörítettem ezt az állományt, hogy később felhasználható legyen. A kép nevébe kódoltam azt, hogy melyik sejtvonalba tartozik a kép (azaz az osztályt). Ezt a **cell\_classification/data\_extract.ipynb** notebookban végeztem el.

|  |  |  |
| --- | --- | --- |
|  | C:\Users\Asrock\AppData\Local\Microsoft\Windows\INetCache\Content.Word\H838_A4_seg_2.png | C:\Users\Asrock\AppData\Local\Microsoft\Windows\INetCache\Content.Word\MCF7_C4_seg_222.png |

Pár kép a tanító adathalmazból.

Ezután ezeket a legenerált tanító adatokat felhasználva elkészítettem a PyTorch-os DataSetet. Itt nem kellett saját DataSet osztályt készítenem, hiszen az ImageFolder tökéletesen megfelet a célnak. Itt meg kellett adni a gyökérkönyvtárat, és hogy hogyan transzformálja a bemeneti képeket.

Itt végeztem el az egységes méretre hozást, amely csak egy egyszerű resize, az összes képet 400x400-as méretre hoztam. Ezután a fentebb említett adatkiegészítést hajtottam végre, véletlenszerűen elforgattam a képet, hiszen nem igazán számít, hogy a sejt milyen orientációban van. Továbbá véletlenszerűen tükröztem a képet. Ezután a normalizációt hajtottam végre, a pixelintenzitást 0.5 várható értékű és 0.5 szórású adathalmazzá formáltam. A normalizálás nagyon fontos a neurális hálózatok tanításánál, lehetővé teszi, hogy a hálók hatékonyan dolgozzák fel az adatokat.

Miután megvan a tanító adathalmaz, már az adatkiegészítéssel alapvetően kétféle fő irányt képzeltem el:

* saját CNN modellt tanítok
* vagy szintén a fentiekhez hasonlóan transfer learninget használok

A saját modell tanítása nagyon erőforrás- és időigényes, valamint a legtöbbször, hacsak nem valami nagyon rossz kiinduló modellt választunk, nem is ér el jobb eredményeket. Ezeket megfontolva szintén a transfer learning mellett döntöttem.

A ResNet architektúra az elenyésző gradiens problémára oldott megoldást, amely nagyon mély CNN hálóknál jelentkezett. Ez nyitotta meg a kaput a nagyon mély, nagyon komplex CNN hálózatok felé, amelyek előzőleg elképzelhetetlennek bizonyultak.

A ResNet50 egy 50 réteg mély ilyen konvolúciós hálózat, amelyet már előre betanítottak az ImageNet-1k adathalmazon 224x224-es képekkel. Esély se lenne saját magunknak betanítani ezt a modellt, hiszen nagyon sok paraméteres, de bátran használhatjuk az általa kinyerhető jellemzőket.

Így is tettem, a kinyert jellemzőkhöz hozzákötöttem egy lineáris réteget, amely 6 kimeneti neuront jelent és a kiindulási modell paraméteret lefagyasztottam (azaz ezeket nem tanulnak). A veszteség függvényhez a több-osztályos klasszifikációnál standard használt CrossEntropyLoss-t használtam az alap paraméterekkel.

|  |  |
| --- | --- |
| **Loss** | CrossEntropy |
| **Optimizer** | SGD |
| **Learning rate** | 0.001 |
| **Momentum** | 0.9 |
| **Epochs** | 20 |
| **Batch size** | 64 |

A tanításhoz használt paraméterek.

Elkészítettem a tanító, validációs és teszt adatokhoz az adatbetöltőket. Ez azért fontos, mert így a tanító és validációs adatok segítségével megállíthatjuk, hogy mikor tanul túl a modell, ezáltal hogy mikor állítsuk le. A teszt adatokkal pedig egy tanítástól független kiértékelést végezhetünk el. Az volt a feltételezésem, hogy egy adott méréshez tartozó egyedi sejteknek egyébként nincs sok közük egymáshoz, ezért ezeket a halmazokat véletlenszerű válogatással alakítottam ki.

A tanításhoz wandb-t is használtam, amely segít követni a tanítás menetét, segít struktrálni a tanításainkat és a hiperparaméter optimalizációban (itt nem végeztem el) is nagyon tud segíteni sweepekkel. Ennek inkább a jövőben lehet szerepe, amikor ehhez a problémához több modellt is kipróbálok.

|  |  |
| --- | --- |
|  |  |

A tanító és validációs pontosságok alakulása a tanítás folyamán.

Ezután elvégeztem a kiértékelést a tesztadatokon, ahol érdekes módon 1-es pontosságot kaptam. Ez az eredmény elsőre nagyon jónak tűnik, viszont aggodalomra adhat okot. Ilyenkor célszerű feltenni a kérdést, hogy pontosan mire is tanult rá a modell: tényleg általánosított? vagy a mérési metodika jellemzőire tanult rá?

Az utóbbira egy megtörtént eset: szintén orvosi képfeldolgozás, röntgenképeken tanítottak, de a képek alján szerepelt egy kis szöveg, erre tanult rá a hálózat. Itt is valami hasonló képzelhető el, csak nem feliratokkal, hanem valami mással. Ezért gyanús az 1-es pontosság, vagy még elképzelhető, hogy rossz volt az eredeti feltételezés, és a tesztadatok nem voltak teljesen függetlenek a tanító adatoktól. Az előbbi kizárására különböző vizualizációs módszerekkel lehet megvizsgálni a hálózat neuronjainak aktivitását és elemezni, hogy pontosan mire is tanult rá.

Ezt a tanítási ciklust nem csak a ResNet50-el lehet elképzelni, számos előre betanított modell is van, ezeket célszerű lehet összehasonlítani és kiválasztani a legjobban teljesítőt egy adott metrika alapján. Sőt, érdekes lehet még egy saját CNN modellt is tanítani és összehasonlítani azt a transfer learning-es modellekkel.

A tanítást és kiértékelést a **cell\_classification/train\_and\_eval.ipynb** notebookban implementáltam.

# További feladatok/lehetőségek

Már implementálva van a két neurális hálós modell, viszont ezek még nincsenek konkrétan összekötve, ezt célszerű lenne átgondoltan összekötni, hogy a bemeneti képből könnyen előállítható legyen a már klasszifikált kimeneti szegmentáció.

Ehhez célszerű lenne valamilyen megjelenítési felületet is készíteni, ahol akár az újonnan érkező felannotált képekkel tovább lehetne tanítani mindkét hálót. Elképzelhető, hogy ezekre már vannak megoldások, ennek is célszerű lenne utánanézni, nem újra feltalálni a kereket.

Érkezett egy kérés a kutatóintézettől, ahol ezt a projektet csinálom/ahol a szakmai gyakorlatomat fogom csinálni, hogy nem lehetne-e ezt a szegmentációs modellt arra használni, hogy egy adott képen megszámoljuk a sejteket. Ennek az egyik kísérletüknél van jelentős szerepe, hiszen nem lehet előre pontosan meghatározni az adott lemezre kerülő sejtek számát, ennek van valamekkora szórása. Azonban az eredmények kiértékelésénél ennek nagy szerepe lehet. Természetesen erre is használható a szegmentációs modell, azonban, akkor már elképzelhető, hogy nem egy globális, hanem inkább egy adott sejtvonalra optimalizált modellt lenne célszerű tanítani, hiszen itt minnél nagyobb pontosságot akarunk elérni. Ha nem tudjuk pontosan meghatározni a sejtek számát, akkor ez nem nagy segítség.

Azonban ebben a projektben nem csak ez adja a nehézséget, kéne egy felhasználói felületet is csinálni, amelyet egyszerűen használhatnak a kutatók. De hol fusson a projekt? Valószínűleg a kutatók által használt gépekben nem lesz erős GPU, ezért az nem biztos, hogy opció, hogy náluk végezzük el a számításokat, mert így nagyon lassú lesz a kiértékelés.

Vagy egy kisebb számításigényű megoldást kell kitalálni, amely akár GPU-mentes gépeken is elfut, vagy valamilyen felhő alapú megoldást kell kitalálni. Az előbbire az merült fel bennem, hogy esetleg ha a kísérlet körülményei nemigen változnak, akkor meg lehetne próbálni az átlagos pixel-intenzitásból megbecsülni a sejtek számát. Azonban ennek pontossága erősen megkérdőjelezhető, de akár ki is lehetne próbálni.

A klasszifikációs modellt mindenféleképpen ki akarom elemezni, hogy eldöntsem: tényleg általánosít a modell vagy csak a mérési metodika sajátosságait tanulta meg. Ezt nagyon fontos eldönteni, hiszen ezen múlik a modell jövőbeli felhasználhatósága.

A szegmentációs modellt illetően a konzulensem felhívta a figyelmemet, hogy ő már kijavította a Cellpose-nak azt a hiányosságát, miszerint nincsen adatbetöltő benne, az egész adathalmazt behúzza a memóriába. Ezt mindenféleképpen kiszeretném próbálni, nagyon robusztussá tenné a tanítást a mostanihoz képest.

Továbbá a konzulensem még egy olyan irányra is gondolt, hogy célszerű lenne valahogyan klaszterizálni az egyedi sejteket, és ezáltal eldönteni, hogy mennyire szétválaszthatóak, ez is egy érdekes irány lehet a későbbiekben.

Tartalom

[Az adathalmaz bemutatása 1](#_Toc167062876)

[A projekt célja 2](#_Toc167062877)

[Az adatok előkészítése 3](#_Toc167062878)

[A projekt struktúrája 3](#_Toc167062879)

[Watershedes szegmentáció 4](#_Toc167062880)

[Cellpose szegmentációs modell 5](#_Toc167062881)

[ResNet50 klasszifikációs modell 6](#_Toc167062882)

[További feladatok/lehetőségek 8](#_Toc167062883)

1. https://docs.opencv.org/4.x/d3/db4/tutorial\_py\_watershed.html [↑](#endnote-ref-1)
2. Stringer, Carsen & Wang, Tim & Michaelos, Michalis & Pachitariu, Marius. (2021). Cellpose: a generalist algorithm for cellular segmentation. Nature Methods. 18. 1-7. 10.1038/s41592-020-01018-x. [↑](#endnote-ref-2)