Dokumentáció

Téma: Áttétes nyirokcsomó daganatok meghatározása szövettani képek osztályozásával neurális hálókat alkalmazva

A probléma a következő: meg kell határozni, hogy a nyirokcsomó szövetmintáján lévő sejtek daganatosok-e. Ehhez szükséges tudni, hogy mik is azok a nyirokcsomók és hogy hogyan válhatnak ilyen formában daganatossá. A nyirokcsomók a nyirokerekkel együtt alkotják a nyirokrendszerünket, amely az immunrendszerünk működésében játszik fontos szerepet. A nyirokrendszer egyfajta tisztogató, szűrő funkciót lát el, amely a kórokozók elleni védelemhez szükséges. A nyirokrendszer nyirokérhálózata hasonló az érrendszerünkhez, összeköttetésben áll minden szervünkkel, ezáltal ha egy szervünk daganatossá válik, a nyirokrendszeren keresztül könnyen elterjedhet a daganat, azaz áttéteket képezhet a test különböző részein. Ekkor ez az áttétes daganat ugyanolyan sejtekből áll, mint a kiindulási, primer tumor, ezáltal a nyirokcsomókban megtalálható áttétes daganatok nagyon különféle sejttípusúak lehetnek (például, ha a primer tumor tüdő-, mell-, vagy vesedaganat akkor az áttét is olyan típusú). Ezért a szakképzett orvosoknak (patológusoknak) is különösen nehéz megállapítani, hogy vannak-e daganatos sejtek a szövettani képen.
Remélem, hogy a tudomány majd később odáig fejlődik, hogy a korábban megvizsgált minták alapján képes lesz egy olyan modellt alkotni, amely nagyon kicsi, szinte nulla hibázási valószínűséggel megmondaná az orvosoknak, hogy felfedezhető-e daganatos sejt a szövettani mintán, vagy akár azt is megmondaná, hogy milyen típusú a primer tumor.
Szerintem ez egy igen érdekes téma, ezért úgy gondoltam, hogy készítek egy egyszerűbb modellt, ami meghatározza egy szövettani mintáról készített képről, hogy tartalmaz-e daganatos sejteket vagy nem. Ehhez találtam megfelelő adatokat kaggle.com-on.

Korábbi megoldások, cikkek, tanulmányok áttekintése, tanulságok levonása

Az előbb említett oldalon található cikket átvizsgálva, úgy gondoltam, hogy az ott szereplő tervezetet jól tudnám alkalmazni a projektem elkészítése során. Azaz a probléma megoldását én is a következőképpen végeztem el:

1. az adatok és képek letöltése,
2. adathalmaz elkészítése, adatok megvizsgálása,
3. a modell elkészítése, tanítása 48pixeles képekkel,
4. a modell újra tanítása 96 pixeles képekkel,
5. az eredmények megvizsgálása, összehasonlítása.

Ezen az oldalon a modell elkészítéséhez a Python egy beépített csomagját használja, amelyet megpróbáltam alkalmazni, hogy jobban megértsem a működését, de sajnos nem sikerült valamiért megfelelően telepítenem a csomagot.
Ezért egy más módon próbáltam neurális hálós modellt készíteni a képek osztályozásához.

Ehhez több fellelhető hasonló modellt tanulmányoztam át, amelyek esetén a következőkre figyeltem oda:

- milyen formában találhatóak meg az adatok,
- hogyan történi az adatbeolvasás,
- milyen formában, adathalmazban tárolják az adatokat,
- milyen paraméter beállításokkal készítik el a modellt,
- hogyan történi a kiértékelés,
- milyen következtetések vonhatók le.

Az ehhez megvizsgált modellek (oldalak):

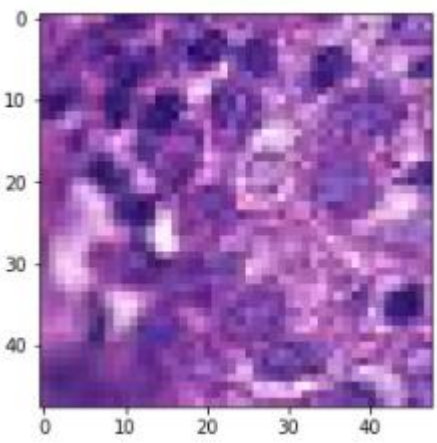
- Convolutional Neural Network For Image Processing
- Convolutional Neural Networks in PyTorch
- CNN Binary Image Classifier in TensorFlow

A választott adathalmaz jellemzése

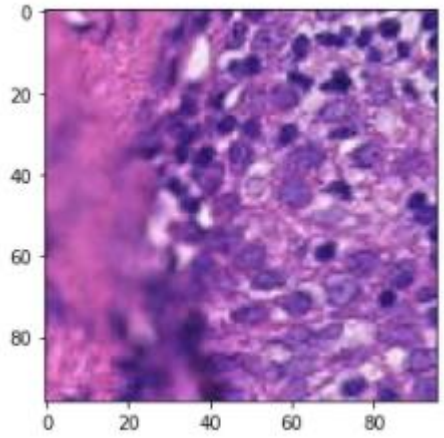
A kaggle.com-ról letöltött adatok két részből álltak: képekből és csv fájlokból. A letöltés után észrevettem, hogy van külön train és test mappa bennük képekkel, azonban a test mappában található képekhez nem volt csv fájl, amelyben a képek bináris címkéi találhatók, arról hogy daganatos sejtek találhatóak-e rajtuk vagy nem. Ezáltal úgy döntöttem, hogy csak a train adatokat fogom felhasználni a modell elkészítése és kiértékelése, tesztelése során, mivel így is rengeteg adat állt a rendelkezésemre.
Összesen 220025 darab 96x96 pixeles, színes képet töltöttem le, amelyet felhasználhattam volna. Ezekhez tartozott egy csv fájl, benne a címkéjükkel, amelyet megvizsgálva körülbelül az egyharmaduk lehetett daganatos típusú, feltéve, hogy ha az 1-es címke jelenti ezt.

Előfeldolgozási lépések részletezése

A csv fájlt dataframeként olvastam be a tanult módon.
A képek beolvasása és megfelelő formátumra hozása már nehezebb volt. Több beolvasási módszert is kipróbáltam mire megtaláltam a megfelelőt, amely könnyen olyan formátumúra alakíthatom őket, hogy a beolvasásukkal a modellnek ne legyen gondja. Ehhez a Python Image és OpenCV csomagjait használtam.
A modellt úgy akartam betanítani az adatokkal, hogy először kisebb méretű képekkel próbálkozok, megnézve, hogy így hogyan is teljesít. Tehát első körben 48x48 pixeles képeket használtam, amelyhez beolvastam a teljes képeket és körbe vágtam őket, hogy csak egy 48x48 pixelek kép maradjon. Ezután ezt RGB színeknek megfelelően 3 rétegre bontottam és a modellhez szükséges np.array típusúakká konvertáltam, az értékeket RGB színek szerint normálva.



48x48 pixeles kép



96x96 pixeles kép

A programban először a tesztelgetéshez így gondoltam a gyorsabb futási idő érdekében csak kisebb adathalmazt alkalmazok, ezért a képeknek csak egy részét használtam fel a csv fájl alapján kiválasztva őket.

A kipróbált és végül alkalmazott modellek, ill. algoritmusok, valamint ezek paraméterezésének dokumentálása

Az órán megismert modellt (CNN) szerettem volna alapjául venni a saját modellemnek, más, az adatokra jobban illeszkedő beállításokkal, paraméterezéssel.
A következő képen jól látható a modelltípusa, paraméterei, a CNN rétegei:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 16)        448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 21, 21, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 512)               33280
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 94,305
Trainable params: 94,305
Non-trainable params: 0
```

A képen jól látható, hogy 3 konvolúciós réteget (2D-ós szűrőt) és ezekhez szintén 3 pooling réteget (összevonót) alkalmaztam, hogy a modell minél jobban rátanuljon a train adatokra, egy flattern réteggel „kisimítottam” az eredményeket, majd kiszűrtem a dense-el a rejtett neuronokat és végül egy neuronom maradt, amely megadja, hogy a kép tartalmaz-e majd daganatos sejteket.

A modellek kiértékelése

A modellt több képmennyiséggel és képmérettel tanítottam és teszteltem. A következő eredményeket kaptam (a megemlített képmennyiségek 2/3-án tanítottam a modellt és a maradék 1/3-án teszteltem):

1000 darab 48x48 pixeles kép esetén:

```
11/11 [==============================] - 0s 23ms/step - loss: 0.5104 - accuracy: 0.7697

0.7696969509124756
```

1000 darab 96x96 pixeles kép esetén:

```
11/11 [==============================] - 1s 77ms/step - loss: 0.5035 - accuracy: 0.7818

0.7818182110786438
```

10000 darab 48x48 pixeles kép esetén:

```
104/104 [==============================] - 3s 26ms/step - loss: 0.6084 - accuracy: 0.6961

0.6960605978965759
```

10000 darab 96x96 pixeles kép esetén:

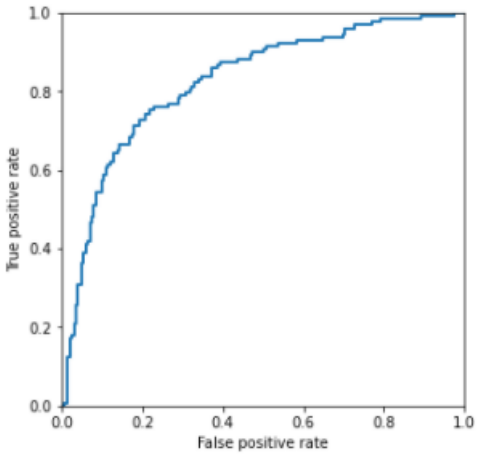
```
104/104 [==============================] - 10s 98ms/step - loss: 0.5163 - accuracy: 0.7461

0.7460606098175049
```

Látható, hogy ilyen mennyiségű képek esetén a modell szinte mind a 4 esetben hasonlóan teljesített.

A kapott eredmények vizualizálása

Ábrázoltam az előző négy esetben a ROC görbéket, amelyek szintén hasonló eredményeket produkáltak.

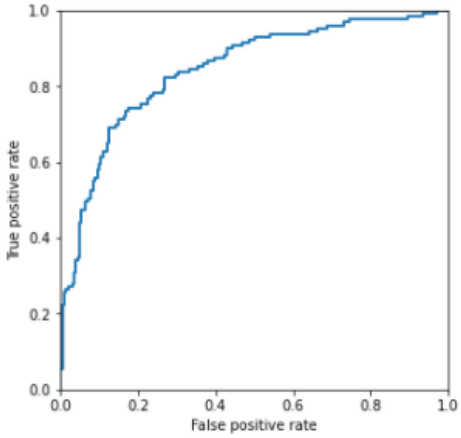
1000 darab 48x48 pixeles kép esetén:




```
metrics.roc_auc_score(test_y, pred)
```

```
0.8252150102202167
```

1000 darab 96x96 pixeles kép esetén:

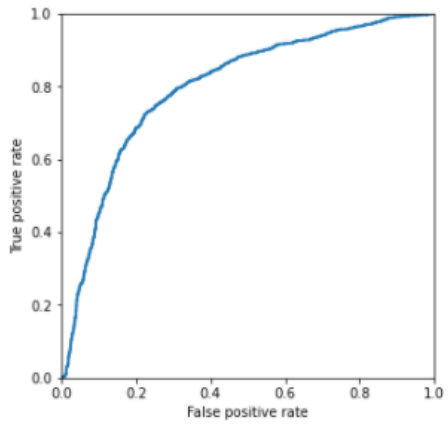



```
metrics.roc_auc_score(test_y, pred)
```

```
0.8440356357746153
```

10000 darab 48x48 pixeles kép esetén:

```
(0.0, 1.0)
```

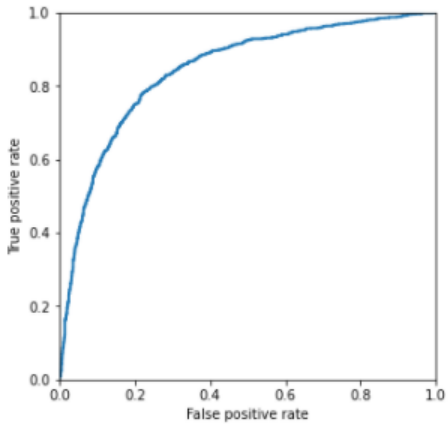



```
metrics.roc_auc_score(test_y, pred)
```

```
0.7987815809810236
```

10000 darab 96x96 pixeles kép esetén:

```
(0.0, 1.0)
```




```
metrics.roc_auc_score(test_y, pred)
```

```
0.843504868416658
```

Következtetések levonása, eredmények használhatóságának értékelése

A modellem 75%-os valószínűséggel osztályozza jól a képeket. Ez sajnos nem olyan jó, mint a kaggle.com-os cikk automatikusan generált modellje, de pusztán a paraméterek tesztelgetésével és állítgatásával szerintem ez egy elég jó eredmény. Valamint az eredményemet nagyban befolyásolhatta az is, hogy nem tudtam sajnos nagyobb mennyiségű adattal futtatni, mert sajnos ezt már nem bírta a számítógépem és mindig lefagyott.

Összességében még lenne mit optimalizálni, például valahogy csökkenteni kellene a futásidőt és a képek beolvasását gyorsítani, a tárolásukhoz szükséges adatmennyiséget pedig csökkenteni. Véleményem szerint, ha esetleg több képpel tudtam volna futtatni, akkor jobb eredményt érhetnék el, de ez most sajnos nem volt lehetséges.

Hivatkozások:

https://www.kaggle.com/omniscientist99/metastatic-cancer-in-lymph-nodes-fastai-resnet50

https://www.youtube.com/watch?v=p-Kw1C5KJog&ab_channel=GreatLearning

https://medium.com/analytics-vidhya/build-an-image-classification-model-using-convolutional-neural-networks-in-pytorch-45c904791c7e

https://towardsdatascience.com/10-minutes-to-building-a-cnn-binary-image-classifier-in-tensorflow-4e216b2034aa

2021. január 13.

Kubicza Gréta Andrea