Projektmunka 3

Szücs Klaudia

2022/23/1

Github link: https://github.com/kaluluh/oebachelorporject

1 Absztrakt

Jelen félévemnek munkája az, hogy az előzetesen kidolgozott tervet a kutatások alapján megalapozott kiválasztott technológiákkal megvalósítsam. Azaz egy Faster R-CNN modell betanítása, ami Resnet50-et használ klasszifikációra.

Először a felhasználni kívánt adathalmazt átalakítottam olyan formátumúvá, ami könnyebben értelmezhető és kezelhető. Ezután az egyes technológiák követelményeinek megfeleltettem az adatokat, majd előállítottam a tanításra alkalmas környezetet. Sikeres tanítást követően az így elkészült modellt elmentettem, majd a következtetések végrehajtásához szükséges kódot implementáltam és lefuttattam a teszt adathalmazon. Ezután az észlelt hibák alapján módosításokat hajtottam végre az adathalmazon.

2 Adathalmaz

A felhasznált adathalmaz az alábbiak szerint nézz ki, amit a szakdolgozatomban az Adathalmaz szekcióban részletesebben kifejtettem:

```
<case>
    <number>129</number>
    <age>49</age>
    <sex>F</sex>
    <composition>predominantly solid</composition>
    <echogenicity>isoechogenicity</echogenicity>
    <margins>well defined</margins>
    <calcifications>microcalcifications</calcifications>
    <tirads>4b</tirads>
    <reportbacaf></reportbacaf>
    <reporteco>Nodule 2</reporteco>
    <mark>
        <image>2</image>
        <svg>[{"points": [{"x": 214, "y": 92}, {"x": 212, "y": 82}, {"x": 208, "y": 77}, {"x": 202, "y": 73}, {"x": 186, "y": 69}, {"x": 176, "y": 65}, {"x":
            166, "y": 63}, {"x": 142, "y": 63}, {"x": 130, "y": 63}, {"x": 122, "y": 63}, {"x": 110, "y": 69}, {"x": 98, "y": 83}, {"x": 95, "y": 87}, {"x":
            91, "y": 94}, {"x": 91, "y": 101}, {"x": 109, "y": 141}, {"x": 111, "y": 143}, {"x": 117, "y": 154}, {"x": 123, "y": 156}, {"x": 147, "y": 157},
            {"x": 195, "y": 146}, {"x": 222, "y": 115}, {"x": 222, "y": 104}, {"x": 221, "y": 96}, {"x": 212, "y": 90}], "annotation": {}, "regionType":
            "freehand"}, {"points": [{"x": 407, "y": 88}, {"x": 406, "y": 82}, {"x": 398, "y": 76}, {"x": 395, "y": 69}, {"x": 390, "y": 67}, {"x": 384, "y":
            67}, {"x": 373, "y": 67}, {"x": 363, "y": 66}, {"x": 354, "y": 66}, {"x": 324, "y": 81}, {"x": 316, "y": 85}, {"x": 309, "y": 89}, {"x": 301, "y":
            97}, {"x": 298, "y": 102}, {"x": 310, "y": 129}, {"x": 318, "y": 146}, {"x": 331, "y": 149}, {"x": 336, "y": 149}, {"x": 356, "y": 157}, {"x": 361
            , "y": 159}, {"x": 387, "y": 161}, {"x": 389, "y": 161}, {"x": 396, "y": 159}, {"x": 405, "y": 150}, {"x": 413, "y": 138}, {"x": 415, "y": 131},
            {"x": 414, "y": 123}, {"x": 414, "y": 114}, {"x": 415, "y": 104}, {"x": 415, "y": 95}, {"x": 415, "y": 90}, {"x": 409, "y": 87}, {"x": 407, "y":
            86}], "annotation": {}, "regionType": "freehand"}]</svg>
    </mark>
</case>
```


Figure 2.1: Adathalmaz felépítése

A tanítás szempontjából releváns információ jelenleg a TI-RADS score, illetve a “pontok” tömb, hiszen az egyik maga a csomó státuszáról, míg a másik a csomó lokalizációjáról tartalmaz hasznos információt.

A későbbi, könnyebb feldolgozás érdekében az XML állományt átalakítottam JSON fájlokká. A JSON fájlokká alakítás közben az adathalmazt felosztottam a tanitásnak megfelelő arányban. (80:10:10) Így előállt a train, validation, illetve a test adathalmaz. Ezekben a JSON fájlokban már nem pont tömbjeim vannak, hanem bounding box-ok.


```
{
    "case_id": "162_1",
    "age": "40",
    "sex": "F",
    "composition": "solid",
    "echogenicity": "hypoechogenicity",
    "margins": "ill defined",
    "calcifications": "non",
    "tirads": "4c",
    "reportbacaf": "",
    "reporteco": "",
    "bboxes": [
        {
            "x": 367,
            "y": 78,
            "w": 84,
            "h": 95
        },
        {
            "x": 101,
            "y": 86,
            "w": 126,
            "h": 89
        }
    ]
},
```


Figure 2.2: JSON fájl felépítése

3 Tensorflow

A nagyobb elterjedtség miatt először a Tensorflow keretrendszerre esett a választásom.Illetve Tensorflow sokkal jobb vizualizációs eszközöket kínál, ami a hibakeresés folyamatát nagyban lerövidítheti, megkönnyítheti. Ezen jó tulajdonságokból kiindulva először ezzel a technológiával kezdtem el dolgozni.

3.1 Adatok előkészítése tanításra

Ahhoz, hogy a Tensorflow beépített API-t megtudjam hívni, az adatokat át kell alakítani úgynevezett TFRecord-okka. Ezek reprezentálják a Tensorflow saját bináris tárolási formátumát. Azonban véleményem szerint túl szigorú struktúrába kell előfeldolgozni az adatokat az API meghívásához. Az adathalmaz elemei háromfajta típusúak lehetnek: tf.train.BytesList, tf.rain.FloatList, tf.train.Int64List.

Az adatokat ezeknek a követelményeknek megfelelően előkészítettem a tanításra, azonban még nem jártam teljes sikerrel.A talált megoldások egyike sem oldotta fel maradéktalanul a problémákat, illetve a nyers adatok erőszakos becsomagolásából adódó utólagos hozzá-nem férhetőségből új technológia mellett döntöttem, amivel már jelentősebb sikereket értem el.

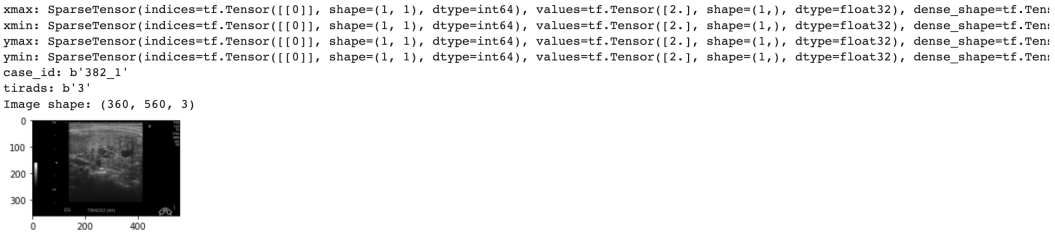

Figure 3.1: Egy TFRecord element felépítése

4 Pytorch

A Tensorflow-val való sikertelen kísérlet után, ezen területen tapasztaltabb emberek javaslatára, a Pytorch nevezetű keretrendszerrel kezdtem el dolgozni, amivel már nagyobb tempóban, szemmel látható sikereket tudtam elérni.Személyes véleményem szerint, ez a keretrendszer sokkal inkább fejlesztő barátibb meg közelítést kínál.

4.1 Első kísérlet

Az új tanítási- es teszt felállítása után kiderült, hogy a hatékony tanításhoz jóval több előfeldolgozás alá kell vetni a bemeneti adatokat.A predikciók magabiztossága alacsony volt, illetve nem csak a releváns objektumokat tekintette fontosnak

4.2 Második kísérlet

A képek átméretezésével és a fölösleges részek levágásával jobban kiemeltem a hasznos információkat. Az átméretezésből következően a bounding box-okat is arányosan transzformálnom kellett.

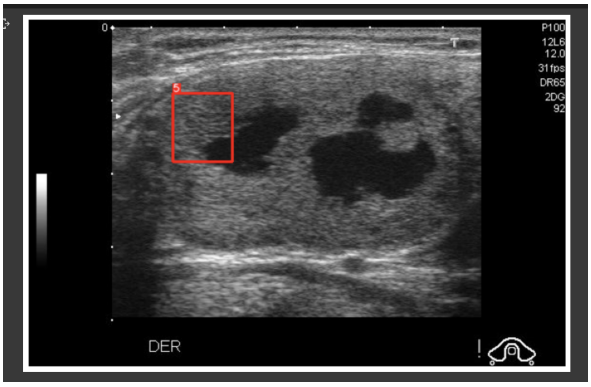

Figure 4.1: A kép az újraméretezés előtt

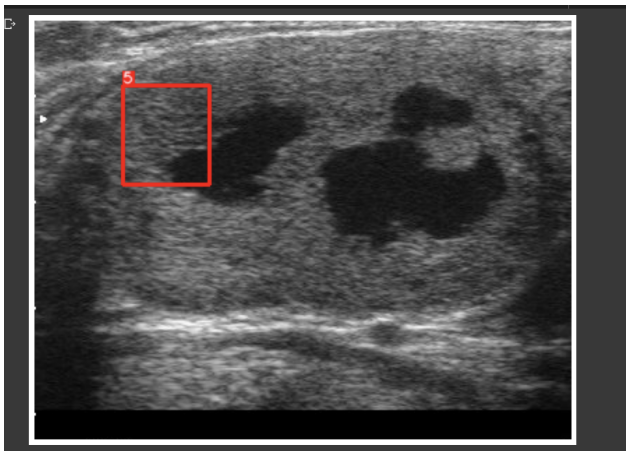

Figure 4.2: A kép az újra méretezés után

A képen alapján jól látható, hogy az újra méretezés sikeresen végre hajtódott úgy, hogy a bounding box ugyanúgy a megfelelő objektumot jelöli. Az így keletkezett újra méretezett adatok szintén JSON formátumban eltároltam.

```
{
    "case_id": "88_2",
    "bboxes": [
        [
            2.0,
            0.0,
            245.0,
            209.0
        ]
    ],
    "labels": [
        4
    ]
},
```


Figure 4.3: Az újra méretezett JSON fájl

A labelek osztályát is lecsökkentettem a következő felosztásban:

```
[ ] category_id_to_name = {2: 'benign', 3: 'benign',4: 'malignant',5: 'malignant' }
```

Figure 4.4: Új label felosztás

4.2.1 Konkluzió

Nagy mértékben sikerült kizárni a felesleges objektumok észlelését a képekről, azonban még nem tudható be sikeresnek a tanítás.

A tanítás az adatok előfeldolgozásából adódó hiba miatt nem volt sikeres.

A bounding box konvertálás során valamikor az eredeti szabadkézzel rajzolt annotációk nagyobbak voltak, mint az újonnan gyártott négyzetek. Továbbá a képek levágási mérete is túl kicsinek bizonyult, így ezt is finom hangolni kellett.

Ezen változtatások után a modell tanítása már sikeresnek mondható.

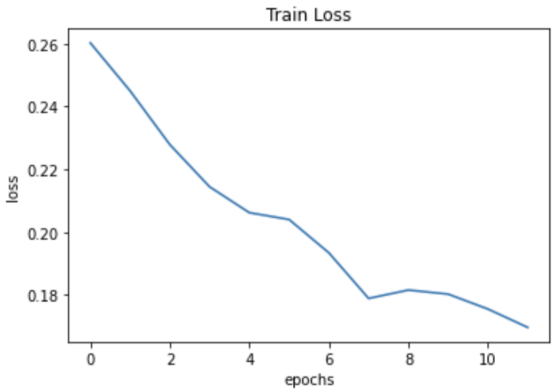

Figure 4.5: Train loss grafikon

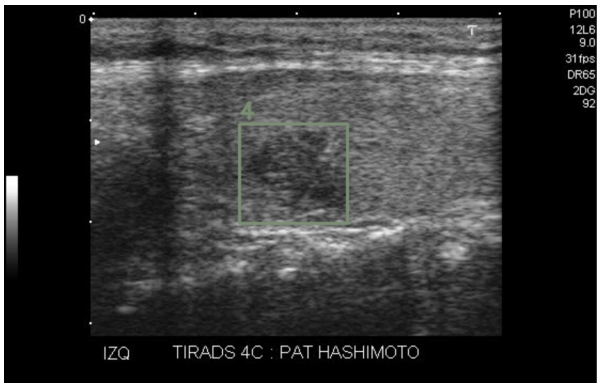

Figure 4.6: Második kísérlet eredménye

4.3 Adatbővítés

A fenti tanítási hiba javítására, az adathalmaz bővítését gondolom megoldásnak.