EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

Pontszerű jelek automatikus felismerése archív térképeken konvolúciós neurális hálózat használatával

DIPLOMAMUNKA
TÉRKÉPÉSZ MESTERSZAK

Készítette:
Vassányi Gergely

Témavezető:
Dr. Gede Mátyás
egyetemi docens, intézetigazgató-helyettes
ELTE Térképtudományi és Geoinformatikai Intézet

Budapest, 2022

Tartalomjegyzék

Abstract

Archive topographical maps are a key source of geographical information from past ages, which can be valuable for several science fields. To perform modern geospatial analysis, it is critical that the information extracted from the maps is in vector format. Most archive maps are only in paper or scanned raster format; therefore, vectorization is needed. The vectorization process consists of recognizing map objects, classifying them, and determining the coordinates of their representing vertices. Since manual digitization is usually slow and takes many human resources, automatic methods are preferred. Deep learning algorithms could be an alternative for this problem, as they have been used widely for image-based object recognition tasks. Although automatic vectorization is a common cartographic problem, there have been few approaches regarding point symbols. In my thesis, I propose a point symbol vectorization method which was tested on map sheets of the Third Military Survey of Austria-Hungary. This method performs map object detection by a Mask Regional Convolutional Neural Network (MRCNN). The MRCNN implementation uses the ResNet101 network improved with the Feature Pyramid Network architecture and is developed in a Google Colab virtual environment. Detection is achieved by training a pretrained model with own data consisting of manually annotated images of the chosen map symbols. The final trained model is capable of detecting four point symbol categories simultaneously. Results show 90% accuracy and 94% of symbols detected for some categories on the complete test sheet, but in other cases, results are far from perfect. Applying this methodology on other map sheets could save a lot of effort, but for better results further improvement is needed.

1. Bevezetés

Az archív térképek hatalmas mennyiségű földrajzi adatot tartalmaznak, amelyek kulcsfontosságúak lehetnek a körülöttünk bekövetkezett környezeti és fizikai változások vizsgálatában. A legtöbb archív térkép azonban csak papír, vagy beszkennelt raszteres formátumban érhető el, így a számtógépes feldolgozási és térbeli elemzési eljárások nem alkalmazhatóak rajtuk. Ehhez vektorizációra van szükség, melynek során a raszteres képen felismert térképi elemeket azonosítjuk, osztályozzuk, illetve meghatározzuk az objektumot felépítő pontok koordinátáit.

A manuálisan végzett vektorizálás számottevő időt és emberi munkát vesz igénybe, ezért kifizetődő az automatikus eljárások használata. Erre kínálnak lehetőséget a mesterséges neurális hálózatok, azon belül is a Konvolúciós Neurális Hálózatok (Convolutional Neural Networks - CNN). Ezt a technológiát széles körben használják objektumfelismerésre, így térképjelek detektálására is alkalmazható. Ennek számos példája akad, azonban viszonylag kevesen foglalkoztak idáig pontszerű jelek felismerésével.

Dolgozatomban a Harmadik Katonai Felmérés szelvényein található pontszerű jelek detektálásán keresztül kívánok bemutatni egy ilyen vektorizációs eljárást, melyhez a konvolúciós neurális hálók egyik típusát (Mask Regional Convolutional Neural Network - MRCNN) használtam. A kiválasztott térképi szimbólumokkal újratanított neurális hálózat felismeri a térképjeleket a bemeneti képeken, majd kimenetként elmenti az objektumok középpontjának koordinátáit, illetve egy képen megjeleníti a detektálás eredményét. A módszer elsősorban a kézzel történő digitalizálás felgyorsítását segíti elő, amellyel jelentős emberi erőforrások szabadulhatnak fel. Előnye, hogy nyílt forráskódú programra épül, emellett a megvalósításhoz használt felületek is ingyenesen hozzáférhetőek.

A probléma bemutatásához először ismertetem az archív térképek jelentőségét, majd – irodalmi adatok alapján – a vektorizációs eljárások főbb lépéseit tekintem át röviden. A következő fejezetben a neurális hálók módszertanát foglalom össze, melyben felvázolom a konvolúciós neurális hálók működését – kitérve az általam használt eljárásra, majd bemutatok néhány korábbi munkát a technológia térképészeti célú alkalmazásáról. Ezután rátérek a megvalósítás konkrét részleteire, illetve prezentálom a kapott eredményeket. Végül ezeket értékelve összefoglalom a dolgozat következtetéseit.

2. Az archív térképek vektorizálásáról

2. 1. Az archív térképek jelentősége

A térbeli változások nyomon követésének hatékony, és sokszor egyetlen módja az időben különböző, pillanatnyi állapotot rögzítő térképek összehasonlítása. Amennyiben részletes, régi korokból származó térkép áll rendelkezésünkre, átfogó képet kaphatunk a hosszú idő alatt lezajlódó átalakulásokról is. Ebből kifolyólag az archív térképek számos tudományterület számára kiemelt fontosságúak lehetnek, ahol ilyen változások vizsgálatára kerül sor.

A természet- és társadalomföldrajzi átalakulások nyomon követésére a 20. század elejétől alkalmaznak történeti térképeket (Bíró 2006), ezeken leginkább a tájhasználati típusokat, a felszínborítást és a növényzet összetételét vizsgálják. A környezeti földtudományon belül a folyószabályozások előtti vízrajz rekonstrukciójához, illetve a bekövetkezett geológiai változások tanulmányozásához is segítséget nyújthatnak archív térképek (Petrovszki 2009). E mellett a régészet számára is felbecsülhetetlen információkat kínálhatnak az emberi települések fejlődéséről (Laycock et al. 2011).

2. 2. A vektorizációs folyamat

Az vektorizációs folyamatok jellemzőit ebben a fejezetben Peller (Peller 2018), Gede és munkatársai (Gede et al. 2020), Iosifescu és munkatársai (Iosifescu et al. 2016), illetve néhány más munka alapján foglalom össze.

2. 2. 1. Általános tulajdonságok

A térképek számtalan változó megjelenési formája miatt nem létezik egységes vektorizációs eljárás, amely teljesen automatikusan működne minden esetben. A térképjelek hasonlóak lehetnek, átfedhetnek egymással, vagy akár a névrajzzal; sok esetben színük is megegyezhet. Archív térképek esetében számolnunk kell továbbá az alacsony felbontással, illetve az idő múlásából adódó károsodással is. Léteznek azonban módszerek, amelyekkel a folyamat bizonyos elemei automatikussá tehetők. A legtöbb vektorizációs eljárás a következő lépésekből áll: szkennelés, georeferálás, kép előfeldolgozás, szegmentálás, raszter szerkesztés, raszter-vektor konverzió, illetve vektoros utófeldolgozás (Peller 2018). Ezek a lépések elsősorban vonalas, illetve felületi elemek vektorizálása esetén érvényesek, és nem mindegyik alkalmazott a neurális hálóval történő detektálás során, ismeretük ennek ellenére lényeges a problémakör bemutatása érdekében.

2. 2. 2. Szkennelés

Ezzel a lépéssel kezdődik az összes vektorizációs eljárás. Szkennelés során egy szkenner, vagy digitális kamerarendszer segítségével raszteres képet készítünk a vizsgált térképről. Körültekintően kell megválasztanunk a felbontást, ez ugyanis alapvetően meghatározza a térkép információtartalmát. Alacsony felbontás esetén értékes részletek veszhetnek el, ugyanakkor túl magas felbontás használata során a térkép rajzi hibái zajként jelentkezhetnek, és ronthatják a vektorizáció eredményét (Pearson et al. 2013). A felbontás megválasztásához figyelembe kell vennünk a keresett térképjelek méretét, illetve a vonalak vastagságát is. Általánosan elmondható, hogy a 200-600 DPI közötti felbontás megfelelő lehet a térkép tulajdonságaitól függően (Peller 2018).

Fontos továbbá a helyes színmélység és a fájlformátum megválasztása. Színes térképek esetén ideális a 24 bites színmélység használata, ennél nagyobb érték esetén ugyanis a fájlméret jelentősen megnő. Célszerű a veszteségmentes tömörítést alkalmazó fájlformátumok alkalmazása (Gede et al. 2020), melyek közül a TIFF a legelterjedtebb.

2. 2. 3. Georeferálás

Georeferálás alatt a raszteres kép vetületi rendszerbe történő illesztését értjük, amelyet különböző GIS szoftverekkel végezhetünk (pl.: ArcGIS, QGIS, Global Mapper). A folyamat során illesztőpontokat adunk meg a raszteres képen, melyeknek meghatározzuk a térképi koordinátáit a kiválasztott vetületben. Célszerű azt a vetületet választani, amiben a térkép készült, azonban előfordulhat, hogy a térinformatikai szoftverek ezt nem támogatják, így egy másik, hasonló tulajdonságú vetületre van szükség (Gede et al. 2020). A vetületi koordinátarendszer illesztésére többféle módszer létezik, ezek közül a leggyakrabban használt a polinomiális illesztés lineáris változata (Tímár 2008).

2. 2. 4. Kép előfeldolgozás és szegmentálás

Ennek a lépésnek a célja a vektorizálni kívánt elemek egységes elkülönítése a térkép többi részétől. Az elemek típusától függően különböző lépésekből állhat, és általában valamilyen szűrő alkalmazását foglalja magában. Ilyen lehet például a vonalas felületkitöltés egységes színfelületté alakítása simítószűrő használatával. Ezután következik a szegmentálás, amely a keresett elemeket alkotó pixelek kiszűrését takarja, leggyakrabban színértékek alapján történő leválogatással. Ebből egy bináris kép készül, amely ideális esetben csak a kívánt jeleket tartalmazza (Iosifescu et al. 2016).

2. 2. 5. Raszter szerkesztés, vektor konverzió és utófeldolgozás

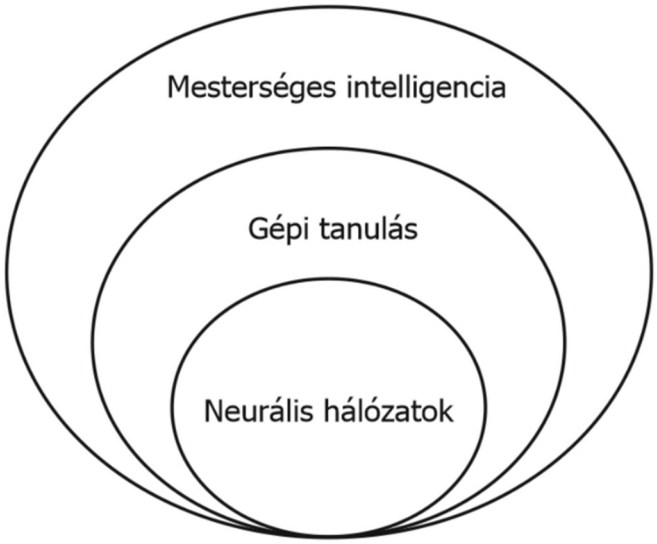
A vektoros konverzió előtt szükséges a bináris kép megtisztítása a különböző zajoktól, ezt foglalja magában a raszter szerkesztés. A hézagok és összemosódó vonalak formájában jelentkező zajokat átfedő elemek és feliratok, a pixelek színének esetlegessége, illetve a vonalak szélességének váltakozásai okozhatják. Ezek kiküszöbölésére morfológiai operátorok alkalmazhatóak, amelyek többek között betölthetnek lyukakat és szakadásokat, illetve eltüntethetik a nem kívánt szövegeket, valamint kis méretű, felesleges pixelcsoportokat (Chiang el al. 2014).

A tisztított bináris képen elvégezhető a vektoros konverzió, amely a keresett elemek típusától függően különböző algoritmusokkal történik. Számos GIS szoftver kínál erre lehetőséget: a kereskedelmi szoftverek között leggyakrabban használt az ArcGIS ArcScan bővítménye, illetve az R2V (Peller 2018). A nyílt forráskódúak közül ilyen a QGIS GRASS modulja, valamint a GDAL könyvtár (Gede et al. 2020). A folyamat utolsó lépéseként a kapott vektoros adatok utólagos feldolgozása, illetve javítása zajlik, amely több műveletből is állhat. Ezek közé tartozik a hibás adatok eltávolítása, a folytonos vonalban keletkezett szakadások megszüntetése, valamint a vonalak simítása.

3. A neurális hálózatkról

3. 1. Alapfogalmak, felépítés és működés

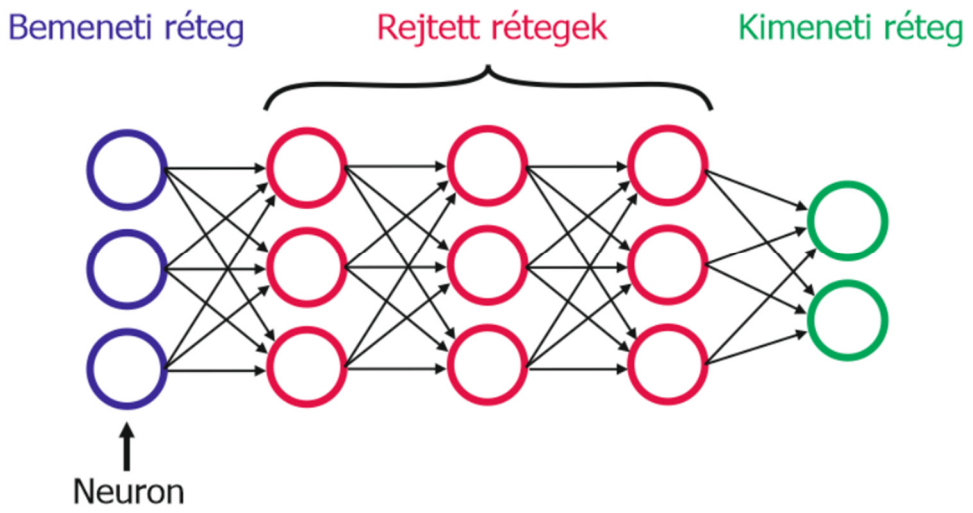
A téma bemutatásához szükséges definiálnunk néhány alapfogalmat. Mesterséges intelligenciának nevezünk minden olyan technológiát, amellyel a számítógép az emberi gondolkodást utánozva végez el összetett feladatokat. Ezek közé tartozik (többek között) a kép- és szövegfelismerés, illetve a nyelvek közötti fordítás. A mesterséges intelligencia egyik eleme a gépi tanulás, amely különböző technikákkal lehetővé teszi a számítógép számára a tapasztalatokra épülő tanulást, illetve fejlődést a feladatok elvégzésében. A gépi tanulási algoritmusok egyik osztályát képviselik a mesterséges neurális hálózatok (1. ábra), melyek az emberi agy működésén alapulva képesek tanulni az adatokból, valamint hatékonyan ismernek fel különböző mintázatokat (Microsoft 2021).



1. ábra A neurális hálózatok fogalmának elhelyezése.

A neurális hálózatok olyan információfeldolgozó eljárások, amelyek azonos, vagy hasonló típusú lokális feldolgozást végző műveleti elem – neuron – összekapcsolt rendszeréből állnak (Altrichter et al. 2006). Egy neuronnak több bemeneti értéke is lehet, amelyek skalármennyiségek, például egy kép pixeleinek az intenzitásértékei. A neuron a bemeneti értékeket súlyokkal látja el, majd ezeket összeadja, és egy nemlineáris, úgynevezett aktivációs függvény alkalmazásával állítja elő a kimeneti értéket. A legnépszerűbb aktivációs függvény a ReLU – Rectified Linear Unit függvény, amely negatív bemeneti érték esetén nullát ad eredményül, egyéb esetben pedig változatlanul hagyja azt (Agarap 2018). A neuron ezután továbbítja a kapott értéket a következő neuronnak.

A neuronok rétegekbe szerveződnek, melyeknek három típusát különböztetjük meg, ezek: bemeneti réteg, rejtett réteg és kimeneti réteg (IBM Cloud Education 2020). A bemeneti réteg szerepe kizárólag a bemeneti jelek továbbítása az első rejtett réteg felé. Az egymás után következő rejtett rétegeken történnek a fent említett számítások, melyekből a kimeneti réteg továbbítja a kapott eredményt a külvilág irányába (2. ábra). A neurális hálók architektúráját a rétegek és a rajtuk található neuronok száma, valamint az összekapcsolás módja határozza meg. Abban az esetben, ha minden neuron kapcsolatban áll egymással, teljesen összekapcsolt hálózatról beszélünk. Az architektúra mellett használatos még a modell kifejezés is. Amennyiben a rejtett rétegek száma meghaladja a kettőt, a hálózat a mély tanulás kategóriájába esik (Rosebrock 2017).



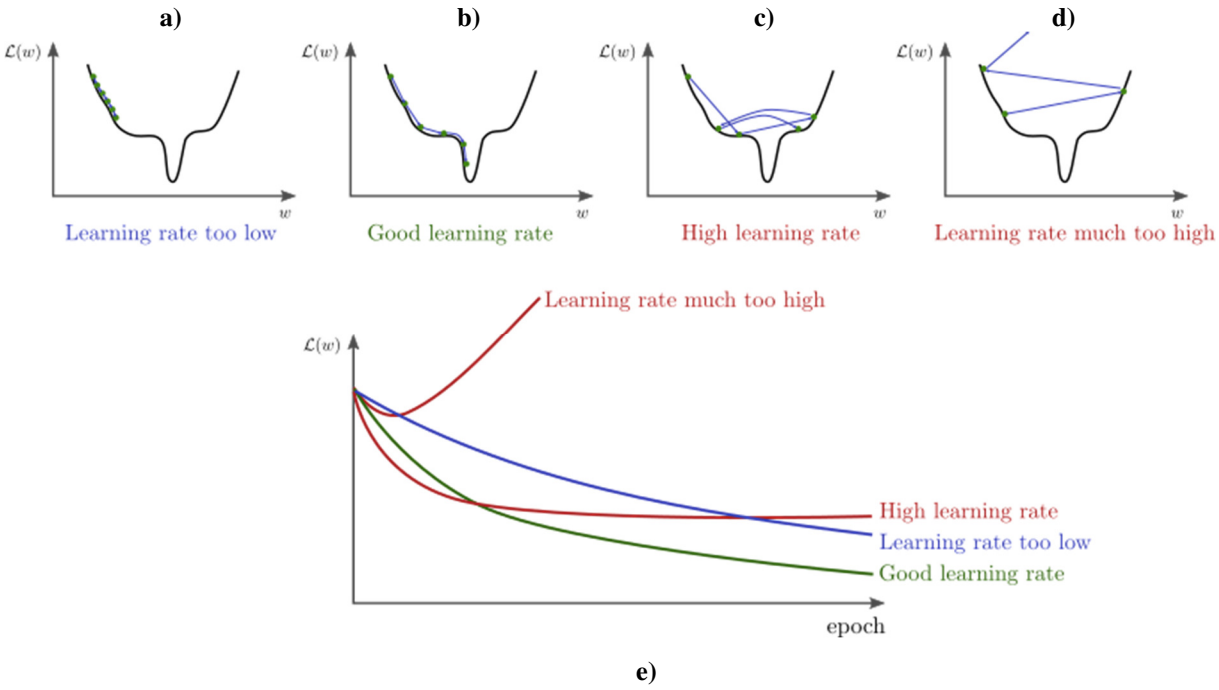
2. ábra A neurális hálók általános felépítése. A rétegek és a neuronok száma változtatható.

A tanulási fázis alatt a neurális háló a bemeneti adatokból hoz létre szabályokat, mintázatokat, amelyek alapján előrejelzést tud készíteni (Khandelwal 2020). Egy betanított hálózatnak csak bemeneti adatra van szüksége az előrejelzés végrehajtásához, ezért a tanítás szerepe kulcsfontosságú.

Tanításakor véletlenszerűen megválasztott kezdeti súlyokat használunk, amelyekkel eleinte nem fogunk jó eredményt kapni (Bushaev 2017). A tanítási folyamat alatt iteratív lépésekben a súlyok állításával, finomhangolásával érjük el azt, hogy a hálózat teljesítménye feljavuljon. Ehhez fontos megemlítenünk a veszteségfüggvény fogalmát, amely egy iteráció végén megadja, hogy mennyire pontosan sikerült a hálózatnak produkálnia a kívánt eredményt. Erre a célra többféle függvény használható a probléma típusától függően, ilyen az átlagos négyzetes

eltérés, a bináris-, illetve a többkategóriás kereszt-entrópia (Verma 2019). A veszteségfüggvény optimalizálásával érhető el a legnagyobb hatékonyság, ehhez a függvény súlyok tekintetében vett gradiensét használják fel (Goodfellow et al. 2016).

A legegyszerűbb gradiens alapú optimalizáló eljárás a gradienssereszkedés, melynek során kivonjuk a súlyokból a kiszámított gradiens értékét, megszorozva egy tanulási rátának nevezett paraméterrel (Durán 2019). A tanulási ráta meghatározza a súlyok állításának mértékét, ezért körültekintően kell megválasztanunk. Amennyiben túl kicsi értéket adunk meg, a tanítás lassúvá válik, több iterációra lesz szükség. Ugyanakkor, ha magasabb a tanulási ráta a kelleténél, nem tudunk elég közel kerülni a veszteségfüggvény minimumához, sőt, el is távolodhatunk tőle (3. ábra).



3. ábra A veszteségfüggvény optimalizálása különböző tanulási ráták esetében. A felső grafikonokon a fekete görbe a veszteségfüggvény; A függőleges tengely a függvény felvett értékét, a vízszintes tengely a súlyokat jelöli. Kék vonallal és csomópontokkal látható az iterációs lépésekben történt közelítés a függvény minimuma felé, balról jobbra haladva: alacsony (a), megfelelő (b), magas (c) és nagyon magas (d) tanulási ráta esetén. Az (e) grafikonon a veszteségfüggvény értéke látható az iterációk függvényében, fentről lefelé haladva nagyon magas, magas, alacsony és megfelelő tanulási ráta esetén (Durán 2019).

Egy neurális háló betanításához nagy mennyiségű annotált tanítóadatra van szükség, amely komoly korlátot szab a mély tanulási módszerek használatának. Ilyen adatok előállítása legtöbbször sok időt és fáradtságot vesz igénybe, illetve komoly szakértelem is szükséges lehet bizonyos esetekben, mint például egy természetes nyelv szavaiból álló adatkészlet

létrehozásánál (Donges 2019). Annak érdekében, hogy kevés adattal is lehessen neurális hálót hatékonyan tanítani, a transzfer tanulásnak nevezett eljárást (Transfer Learning) használják (Tan et al. 2018). A módszer lényege, hogy egy már betanított neurális háló tudását használjuk fel kiindulópontként egy hasonló probléma megoldásához. A gyakorlatban ez a betanított háló súlyainak felhasználását jelenti véletlenszerű kezdeti súlyok helyett az új modell tanítása során.

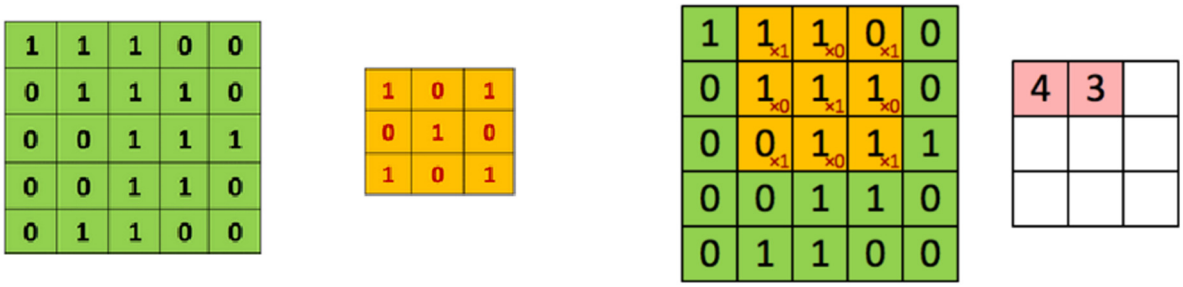
Neurális hálókat leginkább regressziós és osztályozási problémák esetén alkalmaznak. Ezek közül kiemelném a többkategóriás osztályozást, amibe a képek osztályozása is beletartozik. Ebben az esetben a kimeneti rétegen az osztályok számával megegyező számú neuron van, melyek mindegyike egy 0 és 1 közötti számot ad kimenetként. Ez jelzi az osztályba tartozás valószínűségét vagy az osztályozás bizonyosságát, melyek közül a legmagasabb érték határozza meg az osztályozás eredményét (Verma 2019).

3. 2. Konvolúciós neurális hálózatok, R-CNN és fajtái

A konvolúciós neurális hálózatokat (Convolutional Neural Network – CNN, ConvNet) leggyakrabban képfelismerési és osztályozási problémák megoldására használják, például arcok, közlekedési táblák (Dusek 2020), vagy egyéb tárgyak azonosítása során (Karn 2016). A konvolúciós neurális hálók három féle rétegből épülnek fel: konvolúciós, összevonó és teljesen összekapcsolt rétegekből (O'Shea és Nash 2015). Ezek közül a konvolúciós és összevonó rétegek felelnek a tulajdonságok kinyeréséért, míg a teljesen összekapcsolt rétegek feladata az osztályozás.

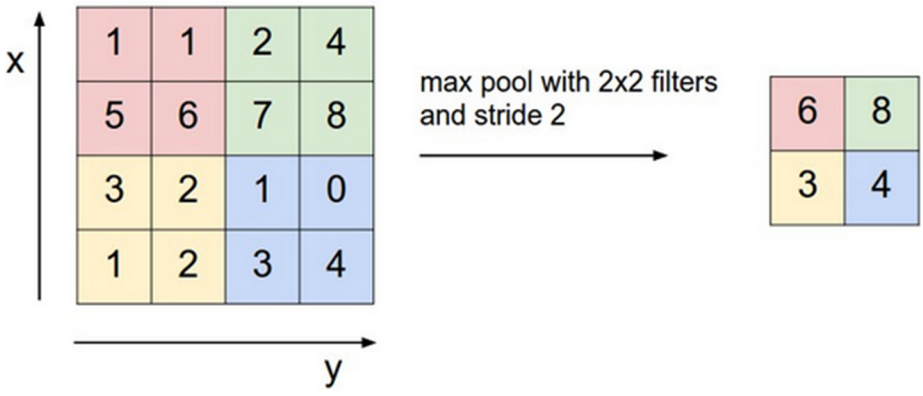
A konvolúciós rétegen egy szűrőt/kernelt vezetünk végig a bemeneti képen, amely egy mátrixszorzást hajt végre a képmátrix és a szűrő mátrixa között (4. ábra). Ez a lépés megfeleltethető a konvolúció műveletének (Khandelwal 2020). Az eredményből egy kimeneti mátrix készül, amelyen ezután egy nemlineáris függvény is lefut; itt általában ReLU-t szokás használni a negatív értékek eltüntetése végett. Az így kapott mátrixot nevezzük aktivációs térképnek, vagy jellemzőtérképnek – Feature Map (Karn 2016). Az aktivációs térkép tárolja el azt, hogy egy adott tulajdonság a kép mely részein erős, például hol találhatóak élek, vízszintes vagy függőleges vonalak, görbék és így tovább. A szűrő határozza meg a keresett tulajdonságot, ezért érdemes több különböző típusú szűrőt is egyszerre alkalmazni a jobb eredmény érdekében. Minden szűrő más tulajdonságot detektál hatékonyan, ezért minden esetben különböző lesz az aktivációs térkép, melyeket a konvolúciós réteg egymásra helyez; Ez adja a réteg mélységét. A gyakorlatban a szűrők értékeit is a tanulási folyamat során állítja be a neurális háló, azonban bizonyos paramétereket nekünk kell meghatároznunk, mint például a

szűrők számát, a kernelablak méretét, illetve, hogy hány pixellel menjen arrébb az ablak két lépés között (O'Shea és Nash 2015).

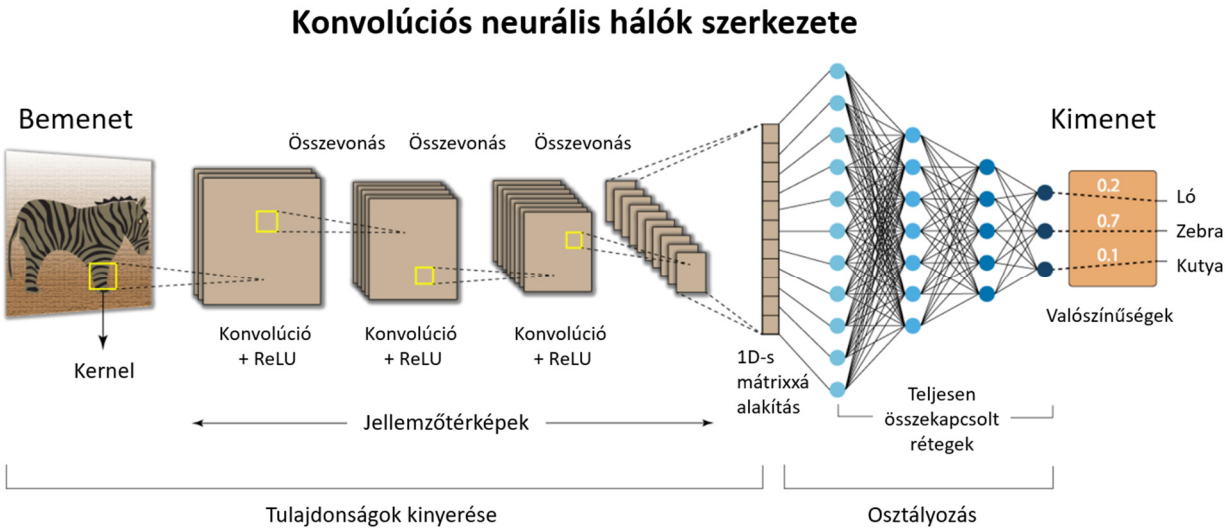


4. ábra A konvolúció folyamata. Balról jobbra a kép mátrixa, a szűrőmátrix, ezek egy adott lépésben vett konvolúciója, valamint az eredmény – a készülő jellemzőtértkép látható (Karn 2016).

Az összevonó réteg (Pooling layer) szerepe az aktivációs térképek dimenzióinak csökkentése a legfontosabb paraméterek megtartásával (Saha 2018). Ezzel a modell egyszerűbbé válik és kevesebb számítási kapacitást vesz igénybe a további műveletek elvégzéséhez. Több változata létezik, ilyen például a Max Pooling, melynek során egy (általában) 2×2-es ablak fut végig a bemeneten, és minden lépésnél a legnagyobb értéket tartja meg (5. ábra). Az aktivációs térkép mérete így ¼-ére csökken, miközben a mélysége változatlan marad. A CNN architektúrákban a konvolúciós és összevonó rétegek többször ismétlődnek egymás után, így egyre komplexebb mintázatokat képes megtanulni a modell. A folyamat végén teljesen összekapcsolt rétegek találhatóak, amelyek az előző fejezetben ismertetett módon működve többkategóriás osztályozást hajtanak végre. Egy tipikus CNN szerkezetet láthatunk a hatodik ábrán (6. ábra).

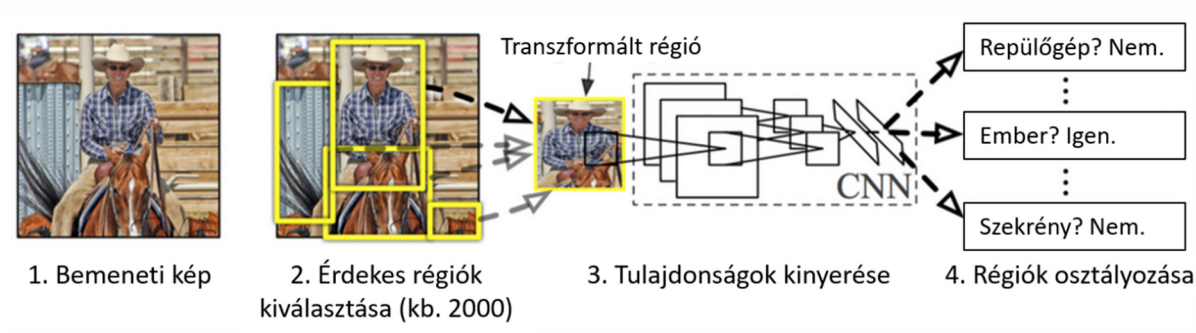


5. ábra A Max Pooling folyamata 2×2-es ablakkal és 2 pixelnyi lépésközzel. Bal oldalt a bemeneti mátrix, jobb oldalt pedig az összevonás eredménye látható (Stanford University 2021).



6. ábra A konvolúciós neurális hálók szerkezete. Az 1D-s mátrixxá alakításra a teljesen összekapcsolt rétegek bemeneti kritériumai miatt van szükség (Developers Breach 2020).

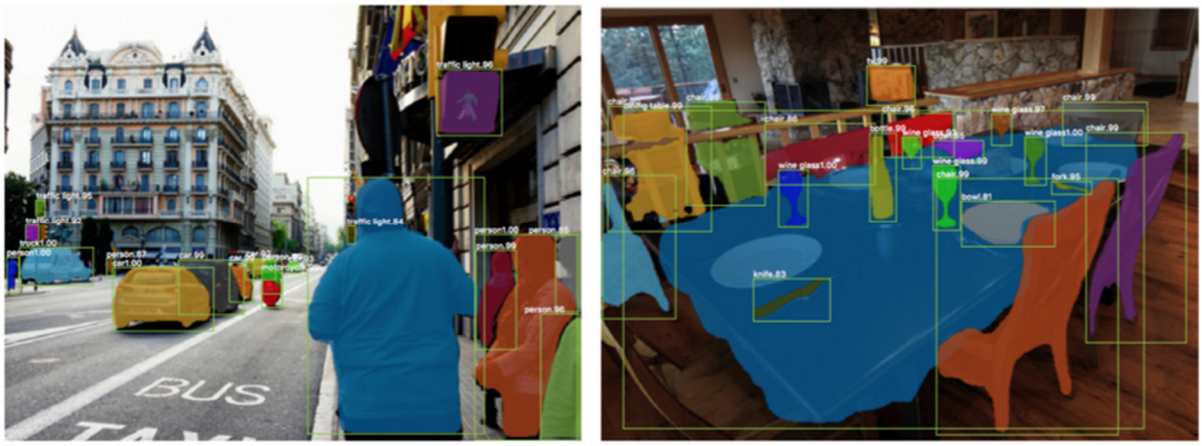
A konvolúciós neurális hálózatoknak számos fajtája létezik, azonban nem mindegyik alkalmazható hatékonyan objektumok detektálásra. Ennél az eljárásnál a célunk a képen található összes objektum azonosítása, melyek köré egy befoglaló téglalap rajzolható. Előre nem tudhatjuk, hogy mennyi objektumot fogunk találni, sem azt, hogy ezek hol helyezkednek el, valamint milyen kiterjedésűek (Gandhi 2018). Hagyományos CNN-ekkel ez a probléma nem megoldható, ezért hozták létre a régió alapú konvolúciós neurális hálókat (Region Based Convolutional Neural Networks – R-CNN). Az R-CNN szelektív keresés használatával kétezer régiót választ ki a képen, amelyek objektumokat tartalmazhatnak; Ezt „Érdekes régió”-nak (Region of Interest - RoI) is nevezzük (Girshick et al. 2013). Ezután az összes RoI-t azonos méretűre transzformáljuk, majd egy CNN-en keresztülküldve elvégezzük a régiók osztályozását (7. ábra).



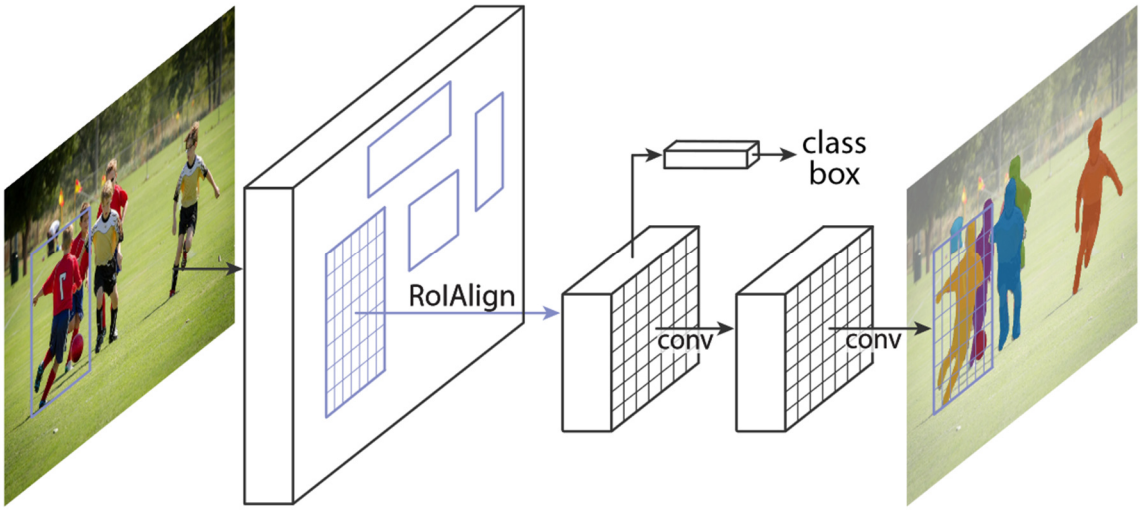
7. ábra Az R-CNN működése a fontosabb lépésekkel (Girshick et al. 2013).

Az R-CNN használatával minden képen kétezer régióra kell lefuttatni a CNN jellemzőkinyerést, ezért a folyamat lassú. E mellett a RoI-k kiszámítását, a konvolúciós lépéseket, valamint az osztályozást három különböző modell végzi, amely megnehezíti a számítások integrálását (Weng 2017). Ennek javítására jött létre a gyors R-CNN (Fast R-CNN), amely összevonja egyetlen modellbe az előbbi elemeket, valamint a vizsgálandó régiókat is a konvolúciós jellemzők alapján állapítja meg a RoI Pooling réteg segítségével (Girshick 2015). Ez a megoldás jelentősen lerövidíti a tanítási és a tesztelési időt a sima R-CNN-hez képest, azonban a folyamat továbbra is lassabb a kívántnál. Továbbfejlesztett változata a gyorsabb R-CNN (Faster-R-CNN), melynek során a régiók ajánlását is egy neurális háló végzi (Region Proposal Network - RPN) (Ren et al. 2015). A gyorsabb R-CNN szinte valós időben történő detektálást tesz lehetővé, így jelenleg a leghatékonyabb eljárások közé tartozik.

A gyorsabb R-CNN-nek több kiegészítése ismert, ezek közül a maszk R-CNN (Mask-R-CNN vagy MRCNN) az egyik legnépszerűbb változat, amely példány-szegmentációt (instance segmentation) tud végezni (He et al. 2020). Ez az objektumok detektálásán túl az objektumpéldányok konkrét, pixel-szintű kiterjedésének megadását takarja, amelyet egy maszk formájában hoz létre az eljárás (8. ábra). A maszkok elkészítését egy teljesen összekapcsolt neurális háló végzi, amely a RoI pixelein bináris osztályozással létrehozza magát a maszkot, majd hozzárendeli az osztályozó algoritmus által javasolt osztályt. A pixel-szintű szegmentáció megköveteli a magasabb pontosságot a RoI-k esetében, ezért a modell tartalmaz egy RoIAlign-nak nevezett réteget is, amellyel kiküszöbölhető a befoglaló téglalapok pozíciójának elcsúszása (9. ábra).



8. ábra Példány-szegmentáció színes maszkokkal és ezeket körülvevő befoglaló téglalapokkal két fényképen (He et al. 2020).



9. ábra Az MRCNN működésének vázlata. A befoglaló téglalap számítása és az osztályozás, illetve a maszk számítása külön történik a RoI-k meghatározása után (felső, illetve alsó ág a képen) (He et al. 2020).

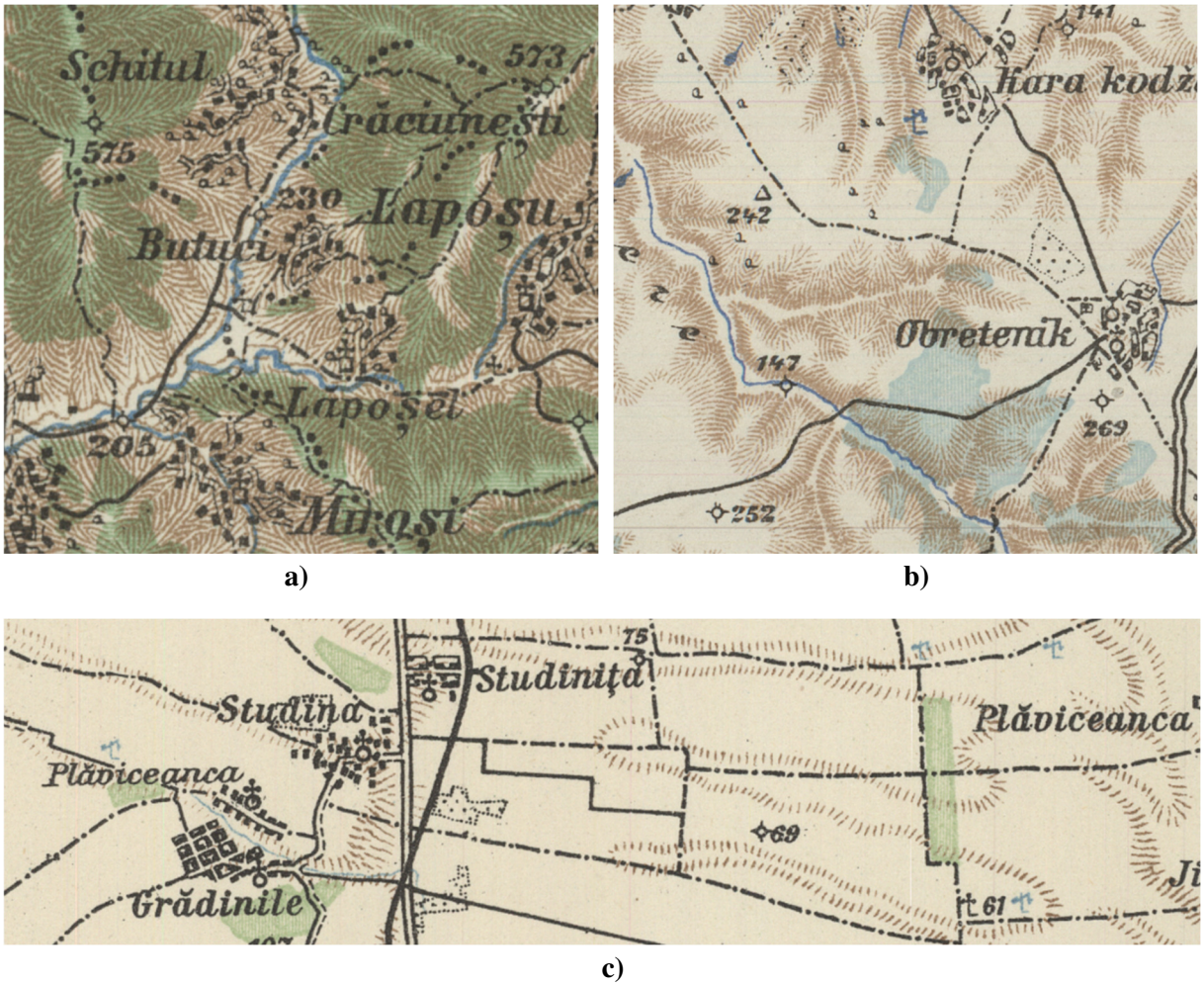
3. 3. Térképészeti alkalmazások

Mivel a vektorizálás folyamata megfeleltethető egy raszteres képen vett objektumfelismerési feladatnak, a konvolúciós neurális hálók alkalmazása kiváló lehetőséget nyújt a térképészek számára. Az utóbbi időben előszeretettel használnak ilyen eljárásokat különböző vektorizációs problémák esetében, legyen az térképi szövegek, vonalas, felületi, vagy éppen pontszerű jelek detektálása. Laumer és munkatársai CNN-t használtak térképi szövegek felismerésre, amellyel magasabb hatékonyságot értek el, mint kézi digitalizálással, és a szelvényenkénti átlagos emberi munka idejét az ötödére csökkentették (Laumer et al. 2020). Jiao és munkatársai mocsaras felületek vektorizációjára dolgoztak ki CNN-re épülő eljárást (Jiao et al. 2020), míg Groom és munkatársai felületi jelek detektálására használtak CNN-t, amivel a pixel alapú szegmentációs eljárás hatékonyságát tudták megnövelni (Groom et al. 2020). Saeedimoghaddam és Stepinski foglalkozott utak metszéspontjainak CNN-nel történő felismerésével, és 90%-os pontossággal sikerült a pontok 82%-át detektálniuk (Saeedimoghaddam és Stepinski 2020). Quan és munkatársai a szegmentálásos eljárás és a CNN kombinációjával 98,97%-os pontosságot értek el topográfiai térképek pontszerű jelei esetében (Quan et al. 2018).

4. Pontszerű jelek detektálása

4. 1. Kiindulási térképek

Munkám során a Harmadik Katonai Felmérés 1 : 200 000 méretarányú térképeit használtam fel, melyek fizikailag elérhetőek az ELTE Térképtudományi és Geoinformatikai Intézet térképtárában (42°44° Slatina, 43°44° Svistov, 44°44° Bukarest és 44°45° Ploiesti). A négy darab, XIX. század végén készült színes térképszelvény egyenként 1° × 1° nagyságú területet fed le a Földfelszínen, amely a mai Bulgária, illetve Románia Duna menti régióit takarja. A változatos domborzattal rendelkező területen hegyvidék, dombság és síkság is előfordul, ezért kifejezetten alkalmas a szimbólumok különböző háttér előtt történő detektálásának vizsgálatára (10. ábra). A térképek szkennelése 600 dpi-vel történt 24 bites színmélység és veszteségmentes tömörítés használata mellett.



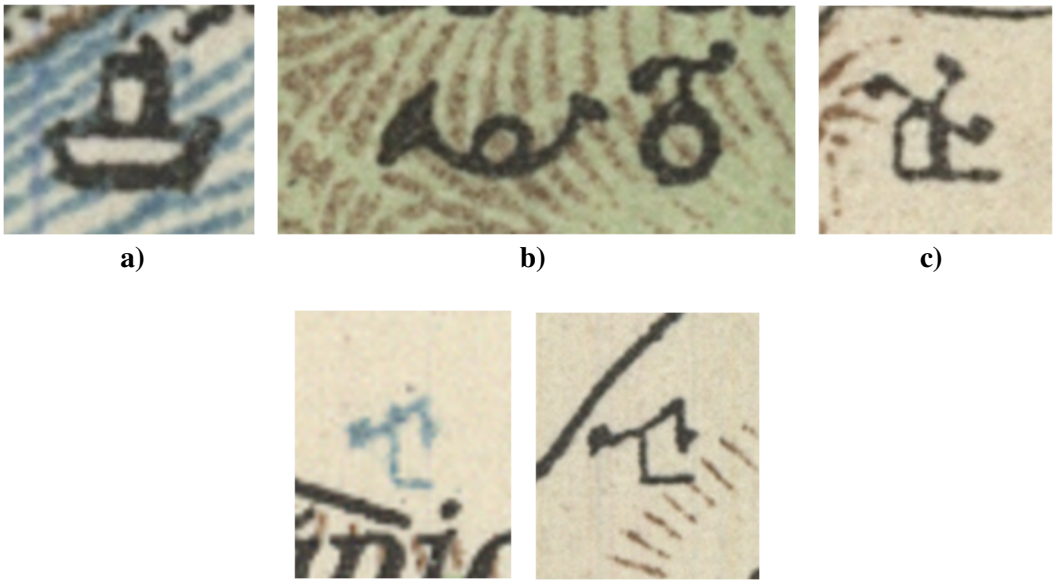
a) b)

c)

10. ábra A térképeken előforduló domborzati típusok: hegység (a), dombság (b) és síkság (c).

4. 2. A térképjelek kiválasztása

A négy szelvényen többféle pontszerű jel található, azonban nem mindegyik volt alkalmas arra, hogy a neurális hálózatot betanítsam a detektálására. A felismerni kívánt jeleknek ugyanis meg kell felelniük néhány kritériumnak, melyek közül a legfontosabb a szimbólum gyakori előfordulása a térképeken. Erre azért van szükség, hogy megfelelő számú tanító és validáló képet lehessen készíteni; Ezeket a szelvényekből kézi kivágással hoztam létre. Voltak olyan szimbólumok, amelyek csak egyetlen szelvényen szerepeltek, és olyanok is, amelyek, bár sokszor előfordultak, különbözőképpen voltak ábrázolva a szelvényeken (11. ábra). Végül azokat válogattam ki, amelyek minden térképszelvényen megtalálhatóak egységes ábrázolással. Ezek a következők: kör alapú templomjel, négyzet alapú templomjel, magassági pont, vízimalom (12. ábra).



11. ábra Ritka jelek (a, b és c) illetve ugyanaz a jel két szelvényen különböző ábrázolással (e és f).

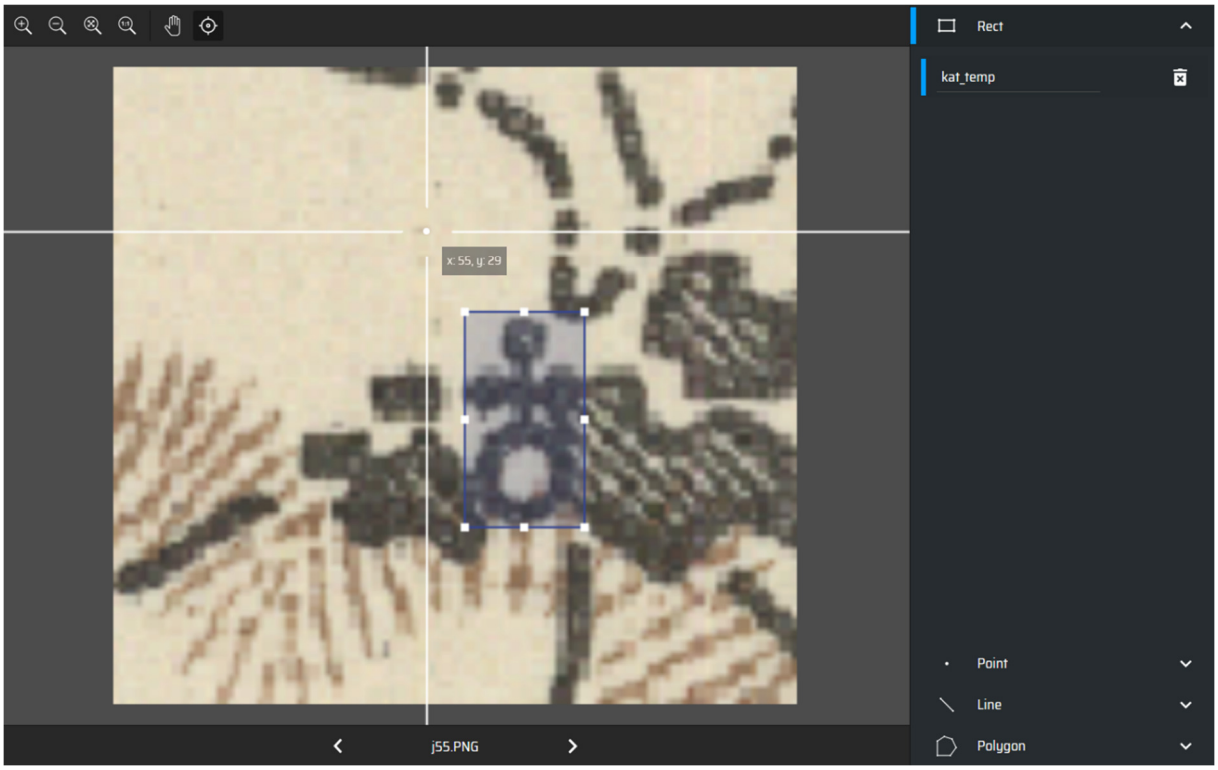
12. ábra A kiválasztott jelek. Balról jobbra: kör alapú templom, magassági pont, négyzet alapú templom, vízimalom.

A felmérés eredeti jelkulcsában a kör alapú templomjel esetében egyszerűen templom, a négyzet alapú templomjelnél pedig kolostor szerepel meghatározásként. Ennek ellentmond azonban, hogy a Bukarest és Ploiesti szelvényeken szinte az összes esetben a négyzetes templomjel szerepel, még a legnagyobb városokban is, tehát nem valószínű, hogy minden esetben kolostort jelentene. Mivel a szelvények nem teljesen egy időben és egyforma jelkulcs alapján készültek, a szimbólumok jelentése szelvényenként eltérhet egymástól (Jankó 2007), így azokat külön-külön kell értelmezni . Detektálás szempontjából azonban sokkal lényegesebb a jelek rajzolata azok jelentésénél.

4. 3. Tanítóképek létrehozása

A modell tanításához szükséges annotált képeket a térképszelvényekről kézi kivágással készítettem, melyhez különböző nagyításokat használtam. Úgy hoztam létre ezeket a képeket, hogy a rajzuk szereplő objektumok sokféle háttérrel szerepeljenek, ezért a térképek különböző pontjairól válogattam össze őket. A képek annotálását egy Make Sense nevű nyílt forráskódú webes eszközzel végeztem (Skalski 2019), melynek során megrajzoltam a jelek befoglaló téglalapját, és azt a jelnek megfelelő osztály típusával láttam el (13. ábra). A program az annotációkat JSON formátumban mentette el. Összesen 100 képet annotáltam minden jel esetén, azonban előfordult, hogy egy képen egynél több azonos típusú jel is szerepelt. Az

annotált képeket 70 : 30 arányban osztottam fel tanító és validáló képekre, a viszonylag kevés adat miatt.



13. ábra Az annotálás folyamata Make Sense segítségével.

4. 4. Megvalósítás Google Colaboratory környezetben

4. 4. 1. MRCNN implementáció és Python kód

Munkám során a 3. 2. fejezetben ismertetett Mask R-CNN-t használtam neurális hálóként, amely egyike az elérhető leghatékonyabb objektum-detektálási eljárásoknak. Ennek egy nyílt forráskódú megvalósításából indultam ki, amely mindenki számára elérhető a GitHub-on (Abdulla 2017). A rendszer gerincét a ResNet101 konvolúciós neurális háló képezi (He et al. 2016), amely egy általánosan alkalmazott, tulajdonságok kinyerésére szolgáló hálózat. Továbbá, a rendszer ki van egészítve egy Feature Pyramid Network-nek nevezett eljárással (Lin et al. 2017), amely az elemtérképeket (Feature Map) több méretben, piramisszerűen készíti el, így megnöveli a változó méretű objektumok detektálásának hatékonyságát. A GitHub könyvtár tartalmazza a hálózat előtanított súlyait az MS COCO adatkészleten (Lin et al. 2014), így ezek a súlyok felhasználhatóak a tanítás során a transzfer tanulás elveit követve. A forráskód Python 3 nyelven íródott a Keras és TensorFlow könyvtárak alkalmazásával. A neurális háló

betanítására, valamint a detektálásra, vizualizálásra és hibakeresésre ugyanitt elérhetőek Jupyter Notebook formátumú programok, melyeknek módosított változatait használtam.

4. 4. 2. Google Colaboratory

A detektáló programot egy virtuális környezetben futtattam, amelyhez a Google Colaboratory (röviden Colab) szolgáltatását vettem igénybe (Nelson és Hoover 2020). A szolgáltatás egy Linux alapú virtuális gépet tesz elérhetővé, amely az erőforrásoktól függően 8-12 GB RAM-mal és 50-70 GB tárhellyel rendelkezik. E mellett grafikus kártyát (GPU) is a felhasználó rendelkezésére bocsájt, amely nagymértékben meg tudja növelni a számítási kapacitást, ezáltal csökkentve a tanítás és a detektálás idejét. A programkód Jupyter Notebook formátumban érhető el, amely lehetővé teszi az interaktivitást és az eredmények vizualizálását a program futása közben. Google szolgáltatás révén a Colab összekapcsolható a Google Drive-val, amely kézenfekvő lehetőséget nyújt a program működéséhez szükséges fájlok importálására, illetve a betanított modellek és a kapott eredmények exportálására is. A szolgáltatás legfeljebb 12 órán keresztül használható ingyenesen, ezért a szükséges modulok telepítését és a Google Drive kapcsolat létesítését minden futtatáskor el kell végezni.

4. 4. 3. A modell tanítása

Tanítás során az előre betanított súlyokat tartalmazó fájlt és az annotált képeket elérhetővé kell tenni a programnak, ezt Google Drive használatával értem el. A tanítást először egyedül a kör alapú templomjelre végeztem el, majd ezt kombináltam a négyzetes templomjellel, míg végül mind a négy jelet egyszerre tanítottam be a hálózatnak. Ennek megfelelően több változat is készült, melyek közül az utolsót fogom ismertetni. A kiindulási kódot úgy módosítottam, hogy négy osztályt tartalmazzon az annotálás során használt azonosítókkal, valamint egyesítettem a tanító és a validáló képeket és ezek annotációit. A tanítás során használt legfontosabb paramétereket kísérletezés útján határoztam meg, a többit pedig változatlanul hagytam. A végső változatban négy iterációt és 1500 iterációnkénti lépést adtam meg, valamint minimum 80%-os bizonyosságot állítottam be. A tanítási folyamat eredményei a Google Drivera kerültek, így a detektáló program számára is könnyen elérhetőek maradtak.

4. 4. 4. Detektálás és export

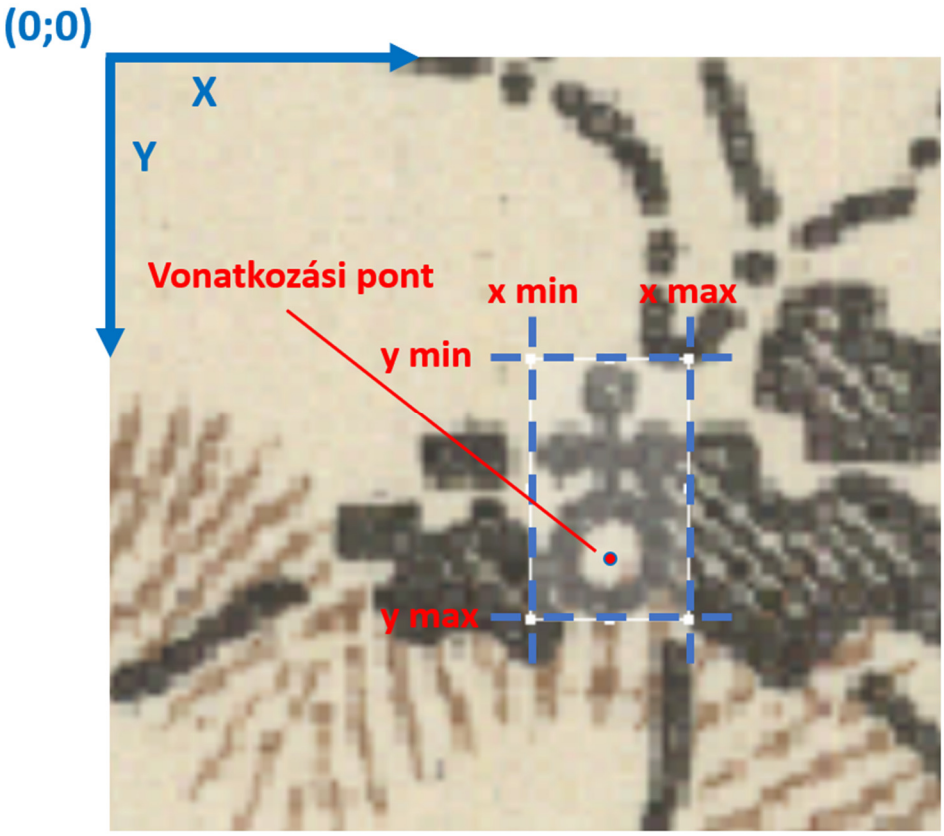
A detektáló program futtatása előtt szükséges a betanított modell és a detektálás tárgyát képező képek elérési útjának megadása. A konfigurációs beállításokat szintén meg kell adni, amelyek megegyeznek a tanítás során használtakkal. A program futásakor egy adott mappában

lévő összes képre lefut, és a detektálás eredményét igény szerint vizualizálja, melyhez egy új képet készít a detektált objektumok maszkjának kirajzolásával (14. ábra). Ebből már képet kaphatunk a detektálás hatékonyságáról, azonban a kimenet továbbra is egy raszteres kép.

```
Processing 1 images
image                    shape: (528, 650, 3)         min:   42.00000  max:  235.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  111.20000  float64
image_metas              shape: (1, 17)               min:    0.00000  max: 1024.00000  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
```

14. ábra A detektálás eredménye a program által megjelenítve. A képen színes maszkokkal négy detektált négyzet alapú templom látható.

Ahhoz, hogy vektoros formátumban nyerjünk ki információt a detektált objektumokról, szükséges meghatároznunk a jelek vonatkozási pontjainak koordinátáit. Ez a pont a malomjel és a magassági pont esetében a jel középpontjában van, azonban a templomjeleknél az alsó kör, illetve négyzet középpontjába esik. A keresett koordináták kiszámítására a befoglaló téglalapot alkalmaztam, amely pixelkoordináták formájában tartalmazza az X és Y tengelyeken vett minimum és maximum értékeket (15. ábra). Az első két esetben az X és Y koordináták átlagát vettem, míg a templomjeleknél az Y koordináták esetében a minimum és a maximum értékek $\frac{3}{2} : \frac{1}{2}$ arányban súlyozott átlagát használtam fel. A program a kapott eredményeket egy CSV fájlba exportálja, amelyben minden detektált ponthoz tartozik X és Y koordináta, rögzítésre kerül a jel típusa és a bizonyosság értéke (16. ábra).



15. ábra A vonatkozási pont elhelyezkedése a templomjelek esetén.

```
1  201,229,kat_temp,0.9974184
2  172,78,malom,0.995605
3  368,446,magassag,0.9954347
4  166,342,malom,0.9356459
```

16. ábra Az exportált CSV fájl. Az oszlopok balról jobbra: X, Y, típus, bizonyosság.

4. 4. 5. Detektálás nagy méretű képeken

Munkám során a teljes térképszelvényen történő futtatást is vizsgáltam, melyhez a Svistov szelvényt használtam fel. A beszkennelt térkép 10 215 pixel széles és 14 352 pixel magas, ebből adódóan viszont volt nem alkalmas arra, hogy lefuttassam rajta a detektálást. Átméretezés nélkül a kép dimenziói olyan számítási kapacitást igényelnének, amelyet a futtatókörnyezet nem tud megközelíteni, ezért a képet fel kellett darabolni. Erre a célra létrehoztam egy Python programot, amely állítható átfedéssel négyzetes csempéket hoz létre a felosztani kívánt képből. A változó méretű képekkel végzett detektálási kísérleteim alapján egy 1024×1024 méretű képet átlagosan 2,5 másodperc, egy 1984×1984-es képet 30 másodperc, míg egy 5120×5120-as képet 5 perc alatt tud feldolgozni a detektáló algoritmus, ennél nagyobb méretek esetében pedig leáll.

Emiatt végül az 1024×1024-es méret mellett döntöttem, az átfedés mértékét pedig 8%-ra állítottam be, ami 82 pixelnek felel meg. Átfedésre azért volt szükség, hogy a feldarabolásból adódóan ne vesszenek el jelek, és 82 pixel elégnek bizonyult ahhoz, hogy minden jel legalább egyszer szerepeljen valamelyik képen. Ha az algoritmus a kép széléhez ér, nagyobb átfedést hagy, majd a következő sorra ugrik. A keletkezett képek fájlnevében szerepel a sor és az oszlop száma, illetve az utolsó sorok és oszlopok esetében egy pixel érték is, amely a többlet átfedést jelzi.

Az így létrehozott képeken a program egyesével elvégzi a detektálást, és mindegyikhez külön CSV fájlt készít, amelyek neve megegyezik a képfájlok nevével. Ezeket egy másik Python program összeilleszti egyetlen CSV-be, és a fájlnévben lévő adatok alapján kiszámolja az összes pont eredeti képhez vett pixelkoordinátáit. Mivel az átfedések miatt előfordulhat, hogy a detektáló algoritmus ugyanazt a térképjelet két különböző képen is felismeri, a program egy adott küszöbérték alatt kiszűri az egymáshoz túl közel eső pontokat.

4. 4. 6. Georeferált koordináták számítása

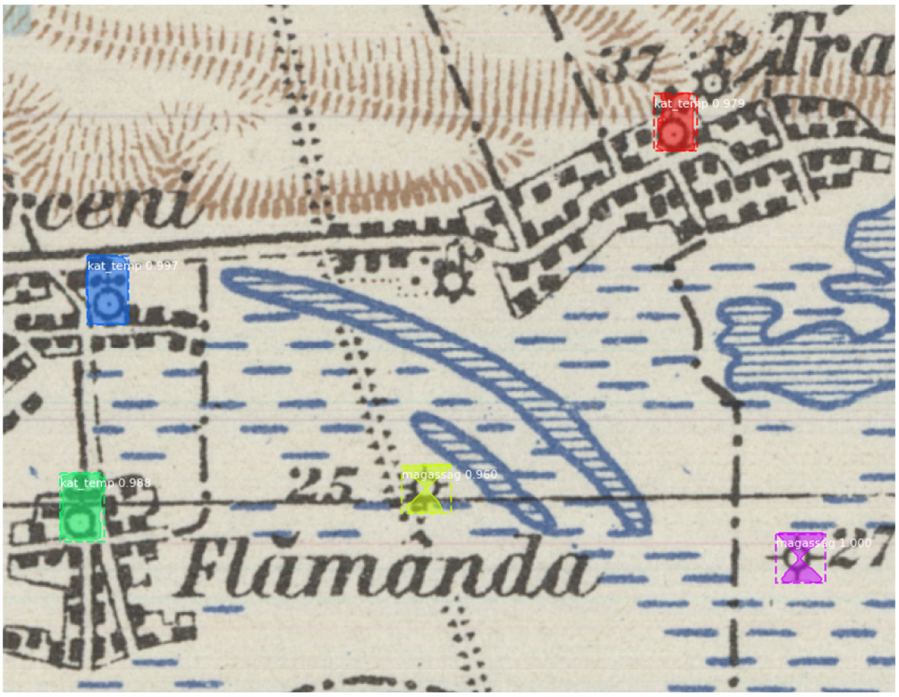
Teljes szelvény esetén, amennyiben az georeferált, a pixelkoordináták helyett térképi koordinátákra van szükség. Georeferálás során készíthető World fájl (.TWF), amely tartalmazza a térkép bal felső sarkának koordinátáját, a pixelek közötti távolságokat térképi egységekben, illetve az elforgatás X és Y irányú komponenseit. Ezt a fájlt felhasználva a CSV összeillesztő program transzformálni tudja a pixelkoordinátákat térképi koordinátákká, az eredmény pedig GIS szoftver segítségével megjeleníthető és vektoros formátumban (pl.: Shapefile-ként) elmenthető.

4. 5. Eredmények

4. 5. 1. Tesztképek esetén

Az eljárást először a szelvényekből kivágott, különböző méretű és nagyítású tesztképeken futtattam. Ez főleg a tanítási paraméterek beállítását szolgálta, e mellett visszajelzést adott a módszer hatékonyságáról. A detektálási eredményeket a program segítségével vizualizáltam, melyek alapján módosítottam a tanítás felétételeit. A végleges, négy szimbólumot felismerni képes változat eljutott odáig, hogy egy jelet sem azonosított tévesen, illetve az összes detektált objektumhoz valóban a neki megfelelő térképjel típusát rendelte (17. és 18. ábra). Előfordultak kimaradt jelek (19. ábra), ám ezek száma viszonylag alacsony volt, így arra a következtetésre jutottam, hogy a tanítás megfelelően sikerült. A többféle nagyításban készített tesztképek

vizsgálata során arra az is világossá vált, hogy a program könnyebben találja meg a nagyobb méretű jeleket.



17. ábra Detektálási eredmény helyes azonosítással. A két, vízimalomhoz hasonló jel fűrészmalom, ezeket a program helyesen nem detektálta.

18. ábra Helyes detektálás két malomjel, egy kör alapú templom és egy magassági pont azonosításával.

19. ábra Kimaradt jelek a kép jobb oldalán (négyzet alapú templom, vízimalom).

4. 5. 2. Teljes szelvényen

Mivel a Svistov szelvény georeferálásakor World fájlt készítettem, az eredményeket vetületi koordinátákkal kaptam meg (a 4. 4. 6. fejezet alapján), majd ezeket QGIS szoftver segítségével vizualizáltam (20. ábra). Elmondható, hogy a teljes szelvényen kapott eredmények vegyes összképet mutatnak. A kör alapú templomjel esetében a program 293 szimbólumot azonosított helyesen a térképen található 309 közül, ami a jelek 94.82%-át jelenti (1. táblázat). Tizenhat esetben nem történt detektálás, továbbá 33 téves pozitív eredmény született, így a pontosság 89.88%-os. A magassági pont esetében már kevésbé volt hatékony az eljárás, a 336 jelből összesen 157-et tudott helyesen azonosítani, négy téves pozitív és 179 kihagyott jel mellett. Ez 97.52%-os pontosságot jelent, azonban a magassági pontok több mint a felét nem detektálta a program. A szelvényen lévő 99 vízimalom és 2 db négyzetes templom esetében egyetlen detektálás sem történt. Előfordultak továbbá olyan téves pozitív azonosítások, ahol egy olyan, a keresetthez hasonló térképjelet detektált az algoritmus tévesen, amellyel a neurális háló nem volt betanítva (21. ábra).

20. ábra A detektálás eredménye a Svistov szelvényen QGIS-ben megjelenítve (részlet). A piros jel a kör alapú templomot, a kék kör a magassági pontot jelöli, mellettük a bizonyosság számértéke látható.

Jel	Kör alapú templom	Magassági pont	Vízimalom	Négyzet alapú templom
Jelek száma a térképen	309	336	99	2
Összes detektálás	326	161	Nem történt detektálás	
Téves pozitív	33	4		
Kihagyott	16	179		
Helyes detektálás	293	157		
Pontosság	89.88%	97.52%		
Detektált jelek aránya	94.82%	46.73%		

1. táblázat A detektálás eredménye a Svistov szelvényen szimbólumokra lebontva.

21. ábra Téves pozitív azonosítás kör alapú templomként, kéményen (bal fent). A piros kör a jel típusát jelöli (kör alapú templom), a szám pedig a bizonyosságot mutatja.

5. Diszkusszió

Az eredmények tükrében kijelenthető, hogy a betanított hálózat képes tesztképeken mind a négy térképjel felismerésére azok összekeverése vagy téves azonosítása nélkül. Ugyanakkor a módszer hatékonysága jelentősen visszaesett a teljes szelvényen történt futtatás során, és csak a kör alapú templom detektálása maradt az elvárt szinten. A magassági pont esetében az algoritmus jelentős részét felismerte a szimbólumoknak, azonban így is kevesebb, mint a felét tudta csak azonosítani, nem is beszélve a vízimalomról és a négyzetes templomjelről.

A teljes szelvényen végzett detektálás során kialakult, szimbólumok közti rendkívül nagy hatékonyságbeli eltérést több tényező is okozhatja, ám ezek azonosítása nem egyértelmű. Ilyen lehet a tanítási paraméterek nem megfelelő beállítása, amelyeket kísérleti úton határoztam meg, illetve változatlanul hagytam az eredeti kódhoz képest. Az ezzel kapcsolatos kísérletek során azonban hasonló eredményt értem el más értékek használatával is, ezért a rossz paraméterek nem tehetőek egyértelműen felelőssé. Ettől függetlenül a hatékonyság tovább növelhető ezek finomhangolásával.

Az ingadozó teljesítmény oka valószínűleg a tanítóadatokban keresendő. Ezek különböző nagyítású, kézzel készített és annotált képekből állnak, a térképjelek mérete ezért minden képen más és más. Elképzelhető, hogy a gyengébben teljesítő jeleknél kevés olyan tanítókép állt rendelkezésre, amelyen a szimbólumok mérete ugyanakkora volt, mint a detektálás során használt szelvényen. Ezáltal a hálózat nem a keresett méretű objektumokra lett kifejezetten érzékeny, amely leginkább a kör alapú templom esetén okozott látványos különbséget a többi jelhez képest. Alátámasztja ezt a feltételezést az az észrevétel, hogy a nagyobb méretű jeleket könnyebben detektálta a program a tesztképeken.

Ennek kiküszöbölésére a tanítóadatok felülvizsgálata és a kívánt méretben történt újra elkészítése nyújthatna megoldást, melyekkel újra lehetne tanítani a neurális hálót. Elképzelhető az is, hogy az egységesen használt 100 tanítókép nem elegendő minden térképjelnél, hanem többre van szükség a kevésbé jól teljesítő szimbólumok esetén. Mindez egyszerűen elvégezhető, a tanítóadatok számának csak a térképi előfordulás szab határt. E mellett a téves azonosítások elkerülése végett célszerű lenne minél több térképjelre kiterjeszteni az eljárást, így a modell különbséget tudna tenni a hasonlónak tűnő szimbólumok között.

6. Összefoglalás

Dolgozatomban igyekeztem rávilágítani az archív térképek jelentőségére, illetve a neurális hálózatok nagyfokú alkalmazhatóságára az ezzel kapcsolatos vektorizációs kutatásokban. Mivel hasonló témával idáig viszonylag kevesen foglalkoztak, nincs egy általánosan alkalmazott eljárás a pontszerű térképi jelek detektálására. Emellett rengeteg olyan feldolgozatlan archív térkép van, amelyből információt lehetne kinyerni a környezet állapotáról. Ezeket összevetve hosszútávú elemzések készíthetőek az adott területen bekövetkezett természeti és társadalmi változásokról.

A bemutatott módszer nyílt forráskódú elemekre épül, amely széles körű alkalmazhatóságot tesz lehetővé. A tanítás és a detektálás bármilyen beszkennelt térképen elvégezhető, amennyiben elég tanítóadatot tudunk hozzá készíteni. Az eljárás használatával nagymértékben csökkenthető a monoton, időigényes emberi munka mennyisége, amely az archív térképekkel kapcsolatos kutatásokra fordított energia hasznosabb beosztását eredményezheti. További felhasználását a részprogramok – feldaraboló, detektáló és összeillesztő program – egyetlen alkalmazásba történő integrálása segítené elő.

Felhasznált irodalom

A webes hivatkozások utolsó elérésének dátuma: 2020.12.10.

Abdulla, Waleed. 2017. “Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow.” *GitHub Repository*. Github. https://github.com/matterport/Mask_RCNN.

Agarap, Abien Fred. 2018. “Deep Learning Using Rectified Linear Units (ReLU),” https://arxiv.org/abs/1803.08375

Altrichter, M., G. Horváth, B. Pataki, Gy. Strausz, G. Takács, és J. Valyon. 2006. *Neurális Hálózatok*. http://project.mit.bme.hu/mi_almanach/books/neuralis/ch01s01

Bíró, Marianna. 2006. “Történeti Vegetációrekonstrukciók a Térképek Botanikai Tartalmának Foltonkénti Gazdagításával.” *Tájökológiai Lapok* 4 (2): 357–384.

Bushaev, Vitaly. 2017. “How Do We ‘Train’ Neural Networks ?” https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73.

Chiang, Yao-Yi, Stefan Leyk, és Craig A. Knoblock. 2014. “A Survey of Digital Map Processing Techniques.” *ACM Computing Surveys* 47 (1): 1–44. doi:10.1145/2557423.

Developers Breach. 2020. “Convolutional Neural Network | Deep Learning.” *Developersbreach.Com*. https://developersbreach.com/convolution-neural-network-deep-learning/.

Donges, Niklas. 2019. “What Is Transfer Learning? Exploring the Popular Deep Learning Approach.” *Built In*. https://builtin.com/data-science/transfer-learning.

Durán, Jaime. 2019. “Everything You Need to Know about Gradient Descent Applied to Neural Networks.” https://medium.com/yottabytes/everything-you-need-to-know-about-gradient-descent-applied-to-neural-networks-d70f85e0cc14.

Dusek, Bence. 2020. “Közlekedési Táblák Automatikus Térképezése.” ELTE Térképtudományi és Geoinformatikai Intézet.

Gandhi, Rohith. 2018. “R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.” *Towardsdatascience.Com*. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

Gede, Mátyás, Árvai Valentin, Vassányi Gergely, Supka Zsófia, Szabó Enikő, Bordács Anna, Varga Csaba Gergely, és Irás Krisztina. 2020. “Automatic Vectorisation of Old Maps Using QGIS –Tools, Possibilities and Challenges.” In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 - ELTE, Budapest*. doi:10.21862/avhm2020.04.

Girshick, Ross. 2015. “Fast R-CNN,” In *Proceedings of the IEEE international conference on computer vision - Santiago, Chile*. pp. 1440 - 1448.

Girshick, Ross, Jeff Donahue, Trevor Darrell, és Jitendra Malik. 2013. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 10.1109/CVPR.2014.81.

Goodfellow, Ian, Bengio Yoshua, és Courville Aaron. 2016. *Deep Learning: Adaptive Computation and Machine Learning*. The MIT Press. pp. 82 - 86.

Groom, Geoff, Gregor Levin, Stig Svenningsen, és Mads Linnet Perner. 2020. "Historical Maps – Machine Learning Helps Us over the Map Vectorisation Crux." In . doi:10.21862/avhm2020.11.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, és Ross Girshick. 2020. "Mask R-CNN." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2). IEEE Computer Society: 386–397. doi:10.1109/TPAMI.2018.2844175.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, és Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-December. doi:10.1109/CVPR.2016.90.

IBM Cloud Education. 2020. "Neural Networks." https://www.ibm.com/cloud/learn/neural-networks.

Iosifescu, Ionut, Angeliki Tsorlini, és Lorenz Hurni. 2016. "Towards a Comprehensive Methodology for Automatic Vectorization of Raster Historical Maps." *E-Perimetron* 11 (2).

Jankó, Annamária. 2007. "A III. Katonai Felmérés Jelmagyarázata." In *Magyarország Katonai Felmérései*. Arcanum Adatbázis Kiadó. https://www.arcanum.com/hu/online-kiadvanyok/Janko-janko-annamaria-magyarorszag-katonai-felmeresei-1/iii-a-harmadik-katonai-felmeres-18691887-1F6/iii-6-a-iii-katonai-felmeres-jelmagyarazata-245/.

Jiao, Chenjing, Magnus Heitzler, és Lorenz Hurni. 2020. "Extracting Wetlands from Swiss Historical Maps with ConvolutionalNeural Networks." In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 -ELTE, Budapest*. doi:10.21862/avhm2020.03.

Karn, Ujjwal. 2016. "An Intuitive Explanation of Convolutional Neural Networks." *The Data Science Blog*. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

Khandelwal, Renu. 2020. "Convolutional Neural Network: Feature Map and Filter Visualization." *Towardsdatascience.Com*. https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c.

Laumer, Daniel, Hasret Gümgümcü, Magnus Heitzler, és Lorenz Hurni. 2020. "A Semi-Automatic Label Digitization Workflow for the Siegfried Map." In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 -ELTE, Budapest*. doi:10.21862/avhm2020.07.

Laycock, S.D., P.G. Brown, R.G. Laycock, és A.M. Day. 2011. "Aligning Archive Maps and Extracting Footprints for Analysis of Historic Urban Environments." *Computers & Graphics* 35 (2): 242–249. doi:10.1016/j.cag.2011.01.002.

Lin, Tsung Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, és Serge Belongie. 2017. "Feature Pyramid Networks for Object Detection." In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-January. doi:10.1109/CVPR.2017.106.

Lin, Tsung Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, és C. Lawrence Zitnick. 2014. "Microsoft COCO: Common Objects in Context." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8693 LNCS. doi:10.1007/978-3-319-10602-1_48.

Microsoft. 2021. "Deep Learning vs. Machine Learning in Azure Machine Learning." https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning.

Nelson, Mark J., és Amy K. Hoover. 2020. "Notes on Using Google Colaboratory in AI Education." In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. doi:10.1145/3341525.3393997.

O'Shea, Keiron, és Ryan Nash. 2015. "An Introduction to Convolutional Neural Networks," https://arxiv.org/abs/1511.08458

Pearson, Matt, G. Salim Mohammed, Renzo Sanchez-Silva, és Patricia Carbajales. 2013. "Stanford University Libraries Study: Topographical Map Vectorization and the Impact of Bayer Moiré Defect." *Journal of Map & Geography Libraries* 9 (3): 313–334. doi:10.1080/15420353.2013.820677.

Peller, Peter. 2018. "From Paper Map to Geospatial Vector Layer." *IASSIST Quarterly* 42 (3): 1–24. doi:10.29173/iq914.

Petrovszki, Judit. 2009. "Archív Térképek Használata a Környezeti Földtudományban: Esettanulmány a Körösök Vidékéről." *Geodézia És Kartográfia* LXI (2): 28–31.

Quan, Yining, Yuanyuan Shi, Qiguang Miao, és Yutao Qi. 2018. "A Combinatorial Solution to Point Symbol Recognition." *Sensors (Switzerland)* 18 (10). doi:10.3390/s18103403.

Ren, Shaoqing, Kaiming He, Ross Girshick, és Jian Sun. 2015. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *Advances in neural information processing systems* 28, pp 91-99.

Rosebrock, Adrian. 2017. "Deep Learning for Computer Vision with Python." *PyImageSearch* 53.

Saeedimoghaddam, Mahmoud, és T. F. Stepinski. 2020. "Automatic Extraction of Road Intersection Points from USGS Historical Map Series Using Deep Convolutional Neural Networks." *International Journal of Geographical Information Science* 34 (5). doi:10.1080/13658816.2019.1696968.

Saha, Sumit. 2018. "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way." *Towardsdatascience.Com*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Skalski, Piotr. 2019. "Make Sense." https://www.makesense.ai/

Stanford University. 2021. “Convolutional Neural Networks (CNNs / ConvNets).” https://cs231n.github.io/convolutional-networks/.

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, és Chunfang Liu. 2018. “A Survey on Deep Transfer Learning.” In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11141 LNCS. doi:10.1007/978-3-030-01424-7_27.

Tímár, Gábor. 2008. “Georeferencia-Térképi Vetületek És Geodéziai Dátumok Szabatos Használata a Térinformatikában.” *Budapest*. http://sas2.elte.hu/tg/georeferencia.htm

Verma, Shiva. 2019. “Understanding Different Loss Functions for Neural Networks.” https://shiva-verma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718.

Weng, Lilian. 2017. “Object Detection for Dummies Part 3: R-CNN Family.” *GitHub*. https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html.

Köszönetnyilvánítás

Szeretném megköszönni a témavezetőmnek, Dr. Gede Mátyásnak az ötletet, a megvalósításban nyújtott sok segítséget és adatot. Emellett hálás vagyok neki a folyamatos támogatásért, amit a nehéz pillanatokban nyújtott.

Köszönöm továbbá Árvai Valentinnek, aki végig segítőkész javaslatokkal látott el a programkód írása közben. Végül köszönöm a családomnak, akik egész idő alatt támogattak és segítettek kikapcsolódni.

EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén – A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Mellékletek

A dolgozat mellékleteként feltöltött ZIP fájlban találhatók a használt programok kódjai. A ZIP fájl a következő állományokat tartalmazza:

- kepfelvago_atfedessel_sajat.py – A kép darabolására szolgáló program
- Detektalas_tanitas.ipynb – A detektálás és a tanítás során használt munkafüzet
- csv_egybe_vetülettel.py – A CSV összeillesztő program

DIPLOMAMUNKA LEADÁSI
és
EREDETISÉG NYILATKOZAT

Alulírott **Vassányi Gergely**, Neptun-kód: **K9T26J**

az Eötvös Loránd Tudományegyetem Informatikai Karának, Térképtudományi és Geoinformatikai Intézetén

Pontszerű jelek automatikus felismerése archív térképeken konvolúciós neurális háló használatával

című diplomamunkámat a mai napon leadtam.

Témavezetőm neve: **Dr. Gede Mátyás**

Büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom, hogy jelen szakdolgozatom/diplomamunkám saját, önálló szellemi termékem; az abban hivatkozott szakirodalom felhasználása a szerzői jogok általános szabályainak megfelelően történt.
Tudomásul veszem, hogy szakdolgozat/diplomamunka esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

A témavezető által benyújtásra elfogadott szakdolgozat PDF formátumban való elektronikus publikálásához a tanszéki honlapon

HOZZÁJÁRULOK NEM JÁRULOK HOZZÁ

Budapest, 2021. december 15.

Vassányi Gergely

……..

hallgató aláírása