Detekcia a rozpoznávanie mikroskopických objektov v obraze

Lucia Hajduková

3AIb, 2018 - 2019

Abstrakt. Cieľom práce je vytvoriť program schopný lokalizovať mikroskopické objekty na upravených snímkach z mikroskopu. Problém bude riešený pomocou konvolučnej neurónovej siete zvanej U-sieť, ktorá bude objekty detegovať a vyznačovať dohodnutým spôsobom.

Kľúčové slová: strojové učenie, neurónové siete, konvolučné siete, U-siete, rozpoznávanie obrazu, TensorFlow

1 Úvod

1.1. Motivácia

Využitím techník rozpoznávania obrazu sa podarilo vyriešiť už mnoho komplexných úloh od jednoduchej zmeny kontrastu či jasu až po detekciu hrán a objektov. Týmto prístupom je v súčasnosti implementovaný aj program na detekciu častíc, ktorý chceme touto prácou upraviť použitím strojového učenia. S nárastom popularity strojového učenia sa totiž ukázalo, že jeho modely poskytujú pri vhodnej implementácii presnejšie výsledky ako klasický prístup rozpoznávania obrazu. Preto je našou víziou pristúpiť k detekcii mikroskopických častíc v snímkach z pohľadu strojového učenia a vytvoriť model rozpoznávajúci častice pomocou U-siete.

1.2. Úvod do problematiky

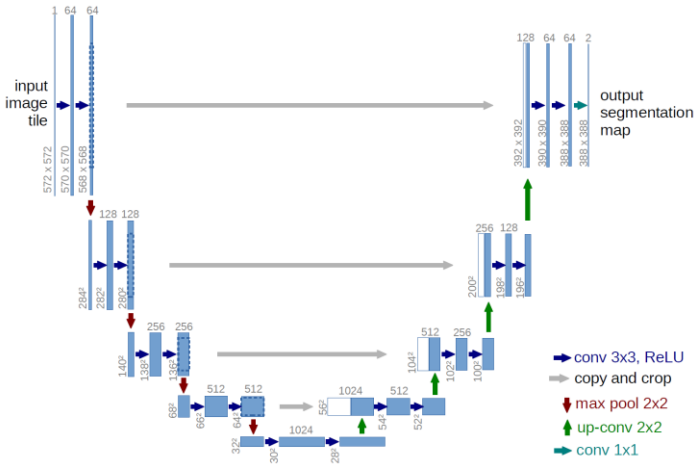
U-siete sú špeciálnym typom konvolučných neurónových sietí. Klasické *konvolučné siete* patria medzi hlboké siete s vrstvami, ktoré nie sú plne prepojené. Majú špeciálnu architektúru prispôsobenú zvlášť na rozpoznávanie a klasifikáciu obrázkov.

Našou úlohou je však klasifikovať každú bunku na snímke, nie obrázok ako celok. Očakávaný výstup teda má byť doplnený o lokalizáciu. Tento problém rieši metóda nazývaná *segmentácia*, ktorá každému pixelu vstupného obrázka priradzuje prislúchajúcu skupinu (segment). Výstupom je segmentovaný obrázok rovnakých rozmerov s vyznačením príslušnosti pixelov do skupín. Na tento problém sú v rozpoznávaní obrazu známe metódy „watershed“, metóda posuvného okna a práhovanie. V roku 2015 však skupina nemeckých vedcov vyvinula U-sieť, ktorá

prekonala výkonnosť dovtedy najspoľahlivejšej *metódy posuvného okna* (sliding-window method).

U-sieť

Názov U-siete plynie z náčrtu architektúry podľa originálneho článku [1] vyobrazeného na Obrázok 1.



Obrázok 1. Architektúra U-siete (príklad pre 32x32 pixelov). Modré rámce znázorňujú multi-kanálovú mapu čŕt, nad nimi je uvedený počet kanálov. Rozmery x,y sú uvedené v ľavom dolnom rohu rámcov. Biele rámce reprezentujú skopírovanú mapu čŕt a šípky predstavujú jednotlivé operácie.

U-sieť v *prvej fáze* postupuje podobne ako konvolučná sieť. Striedajú sa vrstvy vykonávajúce aplikáciu *konvolúcie* a *zhromažďovania*. Zhromažďovaním sa redukuje veľkosť vstupu s ponechaním najdôležitejších znakov. Takýmto spôsobom je získaný jednoduchý výstup označujúci triedu, do ktorej podľa výpočtu obrázok patrí. Prvá fáza zodpovedá ľavej časti nákresu architektúry.

Cieľom *druhej fázy* je doplniť výstup o pixely zredukované v prvej časti. Postup je symetrický s prvou fázou s tým rozdielom, že v každom kroku je miesto konvolúcie aplikovaná *opačná konvolúcia* a zhromažďovanie je nahradené *nadvzorkovaním*. Týmito dvoma operáciami je získaný predbežný výstup, ktorý je ešte potrebné kombinovať s mapou čŕt z prvej fázy na tej istej úrovni (v i-tom kroku 2. fázy je výstup kombinovaný s mapou čŕt z (n-i)-teho kroku 1. fázy, kde n je počet krokov a i = {1,2,…n}). Vďaka tomu sú zachované dôležité črty pôvodného obrázka.

Tento proces nemusí byť aplikovaný na celý obrázok naraz. V praxi sa používa *„stratégia prekrývajúcich sa dlaždíc“* (z anglického overlap-tile strategy). Na výpočet pixelov vybranej časti obrázku sú použité aj pixely z jej okolia, aby bol zachovaný kontext. Ak je vybraná časť na okraji obrázku, okolie je vytvorené

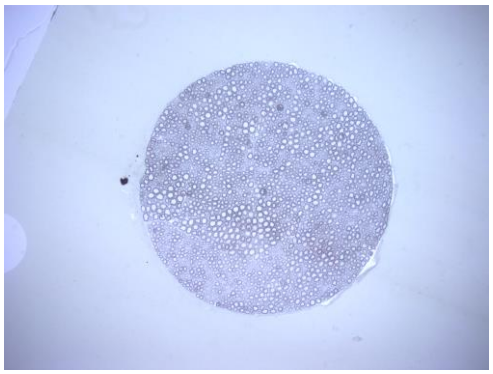
doplnením zrkadlového obrazu tejto časti. Vďaka tomu U-sieť dokáže rýchlo segmentovať aj veľké snímky.

Dáta pre U-sieť

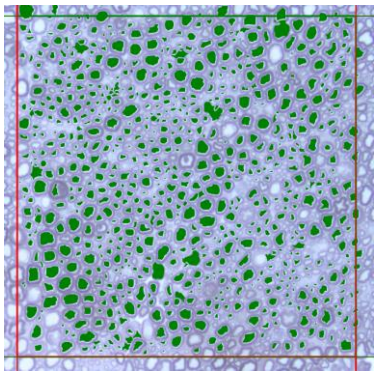
Pre neurónové siete vo všeobecnosti platí, že je to silný nástroj na riešenie rôznych komplexných úloh, ktorý však za svoju presnosť vďačí obrovskému množstvu dát. Naproti tomu U-sieť si postačí s omnoho menšou tréningovou množinou. To je možné vďaka *rozšíreniu* každého tréningového vzoru aplikovaním elastických deformácií. To znamená, že z jedného vzoru je vytvorených niekoľko nových vzorov, pričom sieti stačí pamätať si len originálne vzory a deformácie, ktorými vzniknú nové. Takýto spôsob rozšírenia dát je aplikovateľný predovšetkým pre biomedicínske snímky, ktoré sú si navzájom veľmi podobné a preto je možné aj generovaním vytvoriť vierohodnú snímku.

1.3. Ciele práce a súčasný stav

Cieľom tejto bakalárskej práce je implementovať program, ktorý bude schopný na snímke z mikroskopu označiť bunky, ktoré sa na snímke nachádzajú. Predpokladaný vstup je snímka zobrazujúca bunky (Obrázok 2) a očakávaný výstup je obrázok s vyznačením buniek. V súčasnosti je problém riešený metódou „watershed“, no keďže metóda nie je dostatočne rýchla, segmentácia prebieha po častiach (Obrázok 3). Motiváciou bakalárskej práce je segmentovať snímky pomocou U-siete a výsledky tohto prístupu porovnať so súčasným riešením. Očakávaním je, že U-sieť urýchli proces označovania buniek.



Obrázok 2. Vstuný obrázok.



Obrázok 3. Segmentácia časti vstupného obrázka.

Model siete má byť navrhnutý v TensorFlow a importovaný do projektu v C++, v ktorom bude možné segmentovať ľubovoľnú snímku buniek toho istého typu ako boli použité na tvorbu modelu.

2 Návrh riešenia

Predpokladaný postup práce bude prebiehať v týchto bodoch:

- Implementácia U-siete v jazyku Python použitím TensorFlow
- Trénovanie siete
- Použitie pripravenej siete na segmentáciu obrazov v jazyku C++
- Porovnanie s metódou „watershed“

Implementácia v TensorFlow

TensorFlow [2] je open-souce knižnica pre výskum a produkciu zameraná na strojové učenie. Používa jazyk Python a kód je spustiteľný ako v ľubovoľnom programovacom prostredí pre Python, tak aj v online editore Colab. Budeme používať rozhranie Keras, ktoré obsahuje mnoho užitočných funkcií pre modelovanie hlbokých sietí.

Všeobecný postup pri tvorbe siete podľa [3] vyzerá nasledovne:

1. Zber dát
2. Tvorba modelu
3. Trénovanie
4. Vyhodnotenie
5. Predikcia

Zber dát

Snímky, pomocou ktorých budeme model trénovať a tie, na ktorých otestujeme funkcionalitu získame z databázy mikroskopických snímok, ktorými disponuje Inštitút experimentálnej fyziky, SAV. Tieto snímky sú predspracované.

Tvorba modelu

Keras poskytuje dva základné typy modelov – sekvenčný a model pre funkcionálne rozhranie nazývaný jednoducho Model [4]. V našej aplikácii použijeme Model podľa vzoru tvorcov U-siete [5]. Inicializácia vyzerá nasledovne:

```
from keras.models import Model
from keras.layers import Input, concatenate, Conv2D,
MaxPooling2D, Conv2DTranspose

img_in = Input((img_rows, img_cols, 1))
# prepare layers
model = Model(inputs = img_in, outputs = img_out)
```

Model (`Model`) obsahuje jednu vstupnú vrstvu (`Input`), ktorá načíta vstupný obrázok (`img_in`) a ostatné vrstvy sú výpočtové. Výstup ľubovoľnej vrstvy ide vždy na vstup nasledujúcej vrstvy, okrem poslednej, ktorej výstup je výsledný obrázok (`img_out`). V prvej fáze používame vrstvy na konvolúciu (`Conv2D`) a výber maxím (`MaxPooling2D`). V druhej fáze sa opakujú vrstvy spätnej konvolúcie

(`Conv2DTranspose`) a spájanie (`concatenate`), ktoré sme popísali v teoretickom úvode. Keď sú všetky vrstvy pripravené, je vytvorený model posledným uvedeným príkazom.

Trénovanie

Model vytvorený v predošlom kroku dostane niekoľko dvojíc typu obrázok (`imgs_train`), očakávaný segmentovaný obrázok (`imgs_mask_train`) a niekoľko ďalších parametrov. Každým tréningovým vzorom sa jeho parametre upravia tak, aby sa čo najviac priblížil očakávaným výstupom.

```
model = get_unet()
model.fit(imgs_train, imgs_mask_train,…)
```

Model budeme trénovať na tej istej tréningovej sade niekoľkokrát (kým nedosiahneme požadovanú presnosť alebo nenájdeme nedostatok v inej časti implementácie). Jednému opakovaniu sa hovorí *epocha*. V každej epoche prepočítame hodnotu *chyby* a *presnosti*. Správne by sa zakaždým mala chyba zmenšovať a presnosť zvyšovať. Vďaka týmto hodnotám budeme vedieť určiť, či je sieť natrénovaná podľa našich predstáv.

Vyhodnotenie

Modelu predložíme celkom nové dáta (*testovaciu sadu*), aby sme overili, či správne vyhodnocuje. Zníženie presnosti oproti tréningovej sade značí, že model funguje podľa očakávaní len na obrázkoch z tréningovej množiny a nie vo všeobecnosti. Tento jav označujeme anglickým termínom „*overfitting*“ a keď sa ho dopustíme, snažíme sa ho eliminovať.

Predikcia

V závere sledujeme ako model počíta s dátami, ktoré neboli použité na trénovanie ani testovanie.

Importovanie modelu

Funkčný otestovaný model má byť následne importovaný do projektu v C++, aby bol kompatibilný so súčasným programom, ktorý deteguje bunky metódou „watershed“.

Porovnanie s metódou „watershed“

Parametre, ktoré budeme pri porovnávaní sledovať budú rýchlosť a presnosť výpočtu, pričom presnosti prikladáme väčšiu prioritu.

3 Záver

V tejto bakalárskej práci sa nám doposiaľ podarilo oboznámiť sa s terminológiou a princípmi modelovania neurónových sietí ako aj so špecifikami konvolučných sietí a U-sietí. Získané poznatky sme spísali. Preštudovali sme originálnu implementáciu U-siete a osvojili si programovanie v C++ s využitím knižnice OpenCV. Momentálne je dôraz kladený na pochopenie práce v TensorFlow pre vytvorenie úspešného modelu.

Literatúra

1. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. Eprint arXiv:1505.04597. (2015) Dostupné na https://arxiv.org/abs/1505.04597
2. Tensorflow, https://www.tensorflow.org/
3. Train your first neural network: basic classification, https://www.tensorflow.org/tutorials/keras/basic_classification
4. Keras Models, https://keras.io/models/about-keras-models/
5. Deep Learning Tutorial for Kaggle Ultrasound Nerve Segmentation competition, using Keras, https://github.com/jocicmarko/ultrasound-nerve-segmentation