Festők azonosítása festményeik stílusjegyei alapján

/Identifying painters by the stylistic features of their paintings/

train_validate_test_repeat

Csapattagok:

Bajkó Norbert

Kohlmann András

Mikulás Bence

Nagy Péter

Kivonat

A mély neurális hálózatk elképesztő fejlődésének köszönhetően egyre újabb és újabb lehetőségek nyílnak meg a képfeldolgozás területein. Ilyen feladat például a képek tartalmának azonosítása, melyben óriási előrelépés történt az elmúlt 4 évben. Első fő célunk ennek egy kissé kicsavart változata; adott festmények stílusának elemzése és festőjének azonosítása. Feladatunk volt egy konvolúciós neurális hálózat betanítása a kiválasztott festők stílusának felismerésére, majd ezáltal a később bemenetként kapott festmények eredetének megállapítására. A kapott eredményeket felhasználva a félév második felében a betanult festői stílusok rekreálását tűztük ki célul, amikor is zajból képesek vagyunk elállítani az adott festő műveiből kinyert jellemzők által definiált stílust. Amilyen izgalmasnak hangzik a téma, annyi buktatóval találhatja szembe magát az ember. Az alábbi dokumentum betekintés nyújt féléves munkánk tervezési és implementálási folyamataiba, nehézségeibe és eredményeibe, valamint ejtünk néhány szót a jövőbeli fejlesztési lehetőségekről is.

Abstract

Due to the exceptional evolution of Deep Neural Networks in the past few years, there are more and more opportunties appearing for application in the scope of image processing. A pracrical example is - which took a huge leap in the last 4 years - to identify the contents of any image. Our main and first goal was a twisted version of content identification, we sought to analyze the style of a given painting and indentify its painter. Our task was to train a convolutional neural network (CNN) to recognize the style of selected painters, and later on to determine the origin of a painting received as input. As fort the second part of the semester using the received results, we decided to recreate the learnt painter styles in a way, that the network is able to produce style from noise defined by the obtained features of given paintings. As exciting as is may sound, the subject contains many pitfalls and quite hard to comprehend. This document offers an insight into our design and implementation process, its difficulties and results. In the end, we also mention some suggestions regarding improving our solution further.

Bevezetés

A feladatot a Deep learning a gyakorlatban Python és LUA alapokon tárgy keretében házi feladatként készítjük a követelmények teljesítése és ismereteink bővítése érdekében. A csapattagok érdeklődésének előzetes felmérése alapján egy képfeldolgozási feladatot próbálunk megvalósítani. Választásunk egy képfelismerési feladatra esett, amelyet a Painter by Numbers nevű Kaggle verseny inspirált.

Első lépésként szeretnénk létrehozni egy rendszert, ami meg tudja állapítani egy festmény stílusát és ennek segítségével a lehetséges alkotót. A megoldás tartalmazza a képek előfeldolgozását, elemzését és a kiértékelést is. Másrészt pedig szeretnénk létrehozni egy alkalmazást, ami az első feladatban létrehozott rendszer segítségével képes egy stílus stilizált reprodukálására, valamint esetleges átvezetését egy bemeneti képre a style-transferhez hasonlóan. Harmadik lépésként szeretnénk a rendszerünket valamilyen módon a “külvilághoz kapcsolni”, például mobilos és/vagy webes kliens alkalmazások képében.

Tématerület ismertetése, korábbi megoldások

Painting Mona Lisa

A tanulmány a kép generálás egy lehetséges módszerét mutatja be, számunkra egy ilyen megoldás megismerése miatt volt hasznos.

Forrás

Style Transfer

Bár eredetileg egy kép tulajdonságainak átvitelére szolgál, jelenleg már ezt használjuk a festők stílusának rekreációjához Deep Dream helyett.

Forrás

Toward Discovery of the Artist's Style

Témakört tekintve ez a munka kapcsolódik legjobban a mi feladatunkhoz. A publikáció témája a képek stílusainak detektálása, akár úgy is, ha a képen több festő dolgozott.

Forrás

Megvalósítás

Adatok beszerzése, előkészítése

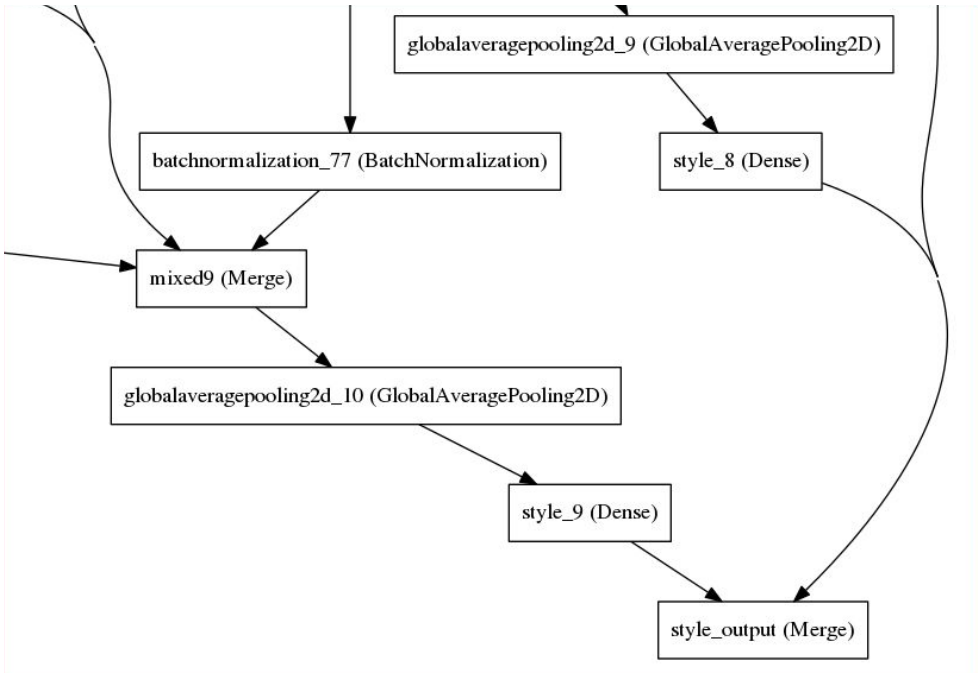
A feladatunk mindkét részéhez ugyanazt a tanító halmazt használtuk, amelyet a Kaggle versenyhez adtak. A tanító adat körülbelül 80 ezer képet tartalmaz, amelyekhez egy csatolt .csv fájlban érhetőek el a metaadatok. A halmaz 20%-át lecsippentettük validációs célokra. Később, a tényleges tanításkor sosem tanítottunk ennyi képpel, valójában körülbelül ennek 1/20 részével (3200/800 képpel) dolgoztunk. Mivel a fórumon elérhető volt egy felhasználó által előfeldolgozott változata a képeknek, amelyek kezelhető méretűre voltak kicsinyítve, ezért a tanításra inkább azokat használtuk (Forrás). Ezzel az egyszerűsítéssel bár a képek részletgazdagsága csökkent nagyon sokat nyertünk mind tanítási időben, mind pedig a feladat komplexitásában. Ennek köszönhetően a képek feldolgozásánál csupán két dolgunk maradt, a képek négyzetes alakra hozása 256x256 pixel méret mellett, valamint a neurális hálózat számára emészthető tömb típusra alakítás.

Festő azonosítás

A tanítást kezdetben Jupyter Notebook-ban végeztük, később sebesség szempontból és azért, hogy a tanszéki szervert tudjuk használni áttértünk a script alapú futtatásra. Az első prototípusban 10 ezer képet tudtunk csak tanításra használni, mert ennyi fért bele a memóriába, de sikerült úgy továbbfejleszteni hogy később már 27 ezer képpel is tudtunk tanítani. Szintén memória szempontból, továbbá a tanított háló minősége indokolta, hogy ne az összes mintával tanítsunk, hanem csak azon festők képeit használjuk, akiknél a lehető legtöbb festmény áll rendelkezésre. Így el tudtuk kerülni, hogy az alulreprezentált festők alkotásai, amelyekre a kevés minta miatt képtelen rátanulni a háló, hibát vigyenek a kiértékelésbe. Célunk tehát a festmények stílusjegyeinek betanítása volt a hálózat számára.

A neurális hálózat tanításának minden iterációját először saját PC-n kezdtük majd, amikor nyilvánvalóvá vált, hogy nincsen szintaktikai és szemantikai hiba, a tanítást átmozgattuk a tanszéki szerverre. A tanítás során többször belefutottunk a túltanulás jelenségbe, ekkor megoldást jelentett Dropout rétegeket beszúrni az egyes Dense rétegek közé. Felmerült, hogy a tanító halmaz számosságban kevésnek bizonyul. Ebben az esetben adatdúsítottunk volna oly módon, hogy a képeket skáláztuk, forgattunk vagy daraboltuk volna, azonban erre

már nem maradt időnk. A rendszerünk Python 3.5 verziójú Keras alapokon nyugszik, ami a motorháztető alatt TensorFlow-t használ. Azért a Keras-ra esett a választásunk, mert viszonylag alacsony a learning curve-je, illetve sok előretanított háló-variáció áll rendelkezésre benne. A scriptünk a *preprocess* illetve *train* logikai részekből áll, nem volt szükség további modulokra bontani az implementációt. Az hálózatunk one-hot encodeing-ot használ a festők osztályokra bontására. Több lehetőséget megfontolva úgy döntöttünk, hogy nem egy teljesen saját, hanem egy előtanított hálóból indulunk ki. A választásunk az Inception V3-ra esett, amely egy bonyolult, több ágú, főleg objektum felismerésre használt modell. A működési elve alapján az elején az első néhány rétegben kell, hogy megjelenjenek olyan alacsony szintű jellemzők, amik segíthetnek azonosítani az alkotóra jellemző apró részleteket. Ezen információk felhasználása érdekében, a modellben található *merge* pontokat előrekötöttük az osztályozást végző fully-connected rétegbe, ezzel javítva a hálózat pontosságán. Ezen pontok szolgálnak bottleneck-ként az Inception-ben, ezért feltételeztük, hogy ezekben a rétegekben akkumulálódik a stílusinformáció.



A hálózat kimenete

Kiértékelés (tanítási, validációs és teszt hiba), hiperparaméter optimalizálás, stb.

A hiperparaméter optimalizálást kézzel végeztük, mellyel ugyan nem értünk el hatalmas javulást, de ~5%-os növekedés volt látható a validációs eredményeket tekintve.

Optimalizációt összefoglaló táblázat

A következő hét folyamán kidolgozunk olyan metrikákat, amelyekkel pontosabb képet kapunk a neurális hálózat teljesítményéről. Elsődleges célunk, egy a festő csoportok közötti tévesztési mátrixot (confusion matrix) és a hozzá tartozó értékeket szeretnénk megalkotni. Majd ezután, ha marad idő, egy a témakörben használt speciális mérőszámot is.

Tesztelés

A képeket egyelőre úgy teszteltük, hogy kiválogattunk a festményeket annak a 10 festőnek a képeiből, akiken a tanítást végeztük. Majd a kiválasztott képeket kétfelé szedtük, egy olyan részhalmazra, amelyek benne voltak a tanítóhalmazban, és egy másik részhalmazra, melyben kizárólag teszt képek szerepeltek, melyeket a hálózat egyáltalán nem látott. Ezeket a képeket végigengedtük a hálón, majd feljegyeztük az eredményeket. A háló képes volt az “ismeretlen” képeket meglepően jól összekapcsolni az azt festő személlyel.

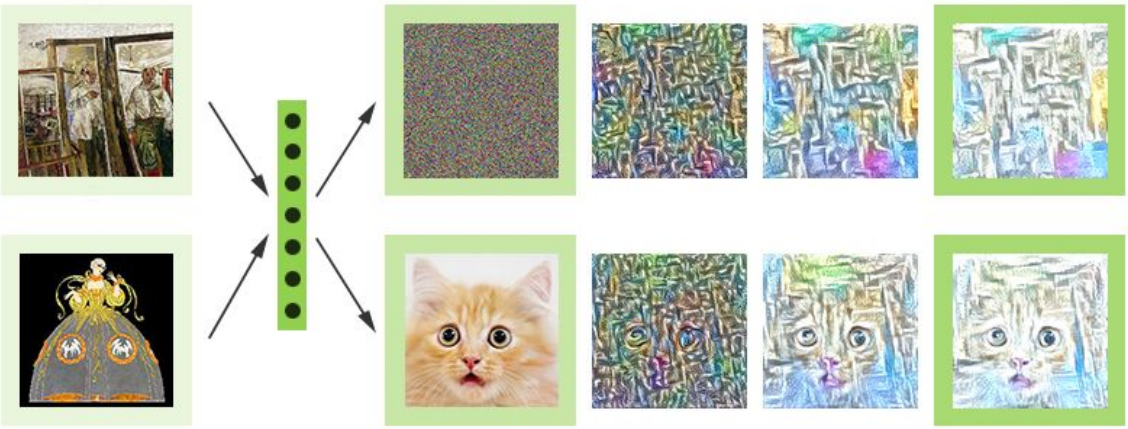
Képgenerálás

A kitűzött második feladatunk során egy festőre jellemző stílussal terveztük képet generálni. Erre két megközelítési módot találtunk, mely lényegében a style-transfernél megismert technikát alkalmazza, miszerint egy kép tartalmának megváltoztatása nélkül a kép stílusát egy forrásképhez igazítja. Az általunk közölt megoldásban értelemszerűen nem egy kép szolgál a stílus alapjául, hanem a tanítás során átlagoljuk a festők képeinek stílusjellemzőit, amelyek praktikusan már egyéb transzformációk nélkül használhatóak is a style-transferben a stílusra vonatkozó veszteség számításához. Az egyik megközelítésben a hagyományosan is alkalmazott metódushoz hasonlóan egy kép tartalmát hagyjuk meg, és a stílust alkalmazzuk rá, a másik megközelítésben pedig egy random zaj szolgál a tartalomért, így a szintetizált kép nem tartalmaz semmit, csupán egy absztrakt kivonata lesz a festő stílusának.

Amennyiben a forrásban is használt hálót (VGG16) használjuk az eredeti feladatban kitűzött célok elérésére, a tapasztalatok szerint nem kapunk kellően jó eredményt. Ez magától értetődik, hiszen az Inception háló architektúrájában egy komplexebb felépítést hoz, mint a VGG16, egy ilyen feladatnál elvárható a magasabb hatásfokú működése. Az egyetlen előny, amely felsorolható mégis az eredeti háló használata mellett, az az, hogy így már adott konvolúciós rétegek vizsgálatával történik a stílusátvitel. Az általunk használt Inception

esetében azonban, a felépítéséből következően más jelleget kapnak az egyes konvolúciós rétegek tartalmai, ezért a hagyományos style-transfernél nem alkalmazható, és az álalános szakirodalom által is ellenjavallott.

Mivel az eredeti feladat eredményességének szempontjából fontos volt, hogy az Inceptiont használjuk, ezért nem cseréltük le végül az alkalmazott hálót, feladva ezzel, hogy egy hálón lehessen a két részfeladatot elvégezni. Így az eredmények, amelyeket a második feladatrészben fel tudunk mutatni mind a style-trasferre alkalmas architektúrában készültek.



A stílust egy köztes adatszerkezettel reprezentálva szintetizáljuk a képet

Interfész

Webszolgáltatás

A hálóhoz készült egy webszolgáltatás és két kliens alkalmazás. A webszolgáltatás a hálóból kiexportált modelt és súlyokat használva fogad egy képet bemenetként, melyet végigvezetve a hálón kimenetként a három legesélyesebb festő azonosítóját, illetve valószínűségüket adja. Mivel a hálózat Python nyelven íródott és az képfeldolgozás abban már implementálva volt, így a könnyű összekapcsolás miatt úgy döntöttünk, hogy azonos nyelven valósítjuk meg a webszolgáltatást is. A választásunk tehát az elterjedt Flask keretrendszerre esett. GitHub-on részleteztük a szolgáltatás használatát, illetve a kimenet formátumát.

GitHub

Webes kliens

A webes kliens első sorban a prezentáció demójára készült. A felület egy képet vár bemenetként, melyet a Keras.js végigvezet a hálón. A Keras.js használatához speciális súlyokat és modelt kellett kiexportálni a hálóból.

GitHub

iOS Kliens

A mobilos kliensünk egy natív, Swift 3-ban megírt alkalmazás, amely funkcionalitásban nem tér el webes klienstől. A felhasználó egy képet tölthet fel a Photo Library-ből, melyet aztán az alkalmazás a webszolgáltatásnak továbbküld, és visszakapja az eredmény valid JSON-ként.

GitHub

Jövőbeli tervek

1. A hálózat bottleneckjeinek felhasználása, mint kivezetési pont egy intuitív megoldás, de közel sem biztos, hogy optimális. A háló belső szerkezetének mélyebb tanulmányozása után levont következtetések után esetleg másik kivezetési pontok alkalmazása lehet, hogy jobb eredmény biztosít
2. Az Inception V3 további tanulmányozásával párhuzamba állítva a VGG16 style-transferben használt rétegeivel felfedezhetünk megfeleltetéseket, amik segítségével a képszintézist végző hálózat architeúráját lecserélhetjük Inception V3-ra a mostani VGG-ről vagy ha nem találunk megfeleltetést, akkor később az egyszerűbb AlexNet-re
3. A tanítást megismételhetjük más tanító és validációs halmazzal, illetve a speciális esetek manuális vizsgálatával pontosíthatunk az eredményeken

Összefoglalás

A projekt során kitűzött célunk volt, hogy ne csak a Kaggle-ön található alapfeladatot oldjuk meg, hanem azt kibővítve egy komplexebb problémát implementáljunk. Ehhez szükséges volt, hogy mindenképpen valamely minőségi újdonságtartalmat is adjunk hozzá a már elérhető megoldásokhoz. Az alapfeladatot, amely két festmény eredetét hasonlítja össze, átfogalmaztuk egy általánosabb, a festő stílusának felismerése köré összpontosuló

problémára, amellyel elértük a célunk, hiszen ez egy összetettebb problémakör, mint az eredeti Kaggle-ös. Ezt a feladatrészt teljességgel sikerült megoldanunk, és a mérhető eredmények vizsgálatakor elégedettek lehetünk. A sok hasonló festmény ellenére képesek vagyunk 10 festő esetén a képek 80%-ánál, 100 festő esetén 60%-ánál az valódi alkotó áll a hálózat által definiált valószínűségi sorban.

Ezen kívül a második feladatrész, amely megvalósítását célul tűztük ki, szintén egy speciális probléma általánosítása, név szerint a style-transfer megoldásának nem egy festmény stílusával operáló változata, hanem az egy festőre jellemző stílus megtalálása. Ez az egy festőre jellemző átlagos stílus szolgált a megoldásunkban alapul a style-transfer során, így átfogalmazva az eredeti funkciót. Ezt a feladatot részben sikerült megoldanunk, hiszen az első részben használt hálót nem tudtuk újra felhasználni, mert annak az architektúrája nem alkalmas a style-transferre, azonban egy kezdetlegesebb hálóval sikerült ezt is implementálnunk. A style-transfer metrikái mindig ködös része volt a tématerületnek, hiszen egy festmény stílusát, illetve két festmény stílusának hasonlóságát felettébb szubjektív megállapítani, ezért a megoldásunk értékeléséhez is csak ennyit tudunk hozzátenni, hogy fel véltük ismerni az alkalmazott festő stílusát a generált képben.

A megoldás során természetesen az eredményeken kívül fontos volt számunkra, hogy a csapat minden tagja megismerkedjen a választott tématerülettel, és elsajátítsa a neurális hálók építésének, valamint tanításának képességét. Reflektálva a féléves munkánkra egyértelműen megállapíthatjuk, hogy ezen kitűzéseket maradéktalanul elértük.

Fájlok:

Előfeldolgozás	(2 db | 91 sor)
Festő azonosítás	(1 db | 109 sor)
Stílus tanulás	(1 db | 387 sor)
Website	(3 db | 338 sor) + 3 modell állomány
Web service	(1db | 104 sor) + 2 modell állomány
iOS app	(5db | 172 sor)

A fejlesztéshez használt eszközök:

Software: Keras, nano, Atom, PyCharm, Keras.js, npm, Xcode, Flask

GPU: GT 740M (saját), GTX 660 Ti (saját), Titan X (SmartLab)

Hivatkozások

1. Painting Mona Lisa

   GitHub: https://github.com/evolvingstuff/MonaLisa
   Blog: evolvingstuff.blogspot.hu/2012/12/generating-mona-lisa-pixel-by-pixel
   evolvingstuff.blogspot.hu/2012/12/learning-to-generate-mona-lisa-animated
   Reddit: www.reddit.com/r/programming/comments/15qj3p/

2. Style Transfer

   Paper: https://arxiv.org/pdf/1508.06576v2.pdf
   GitHub: https://github.com/jcjohnson/neural-style

3. Toward Discovery of the Artist's Style

   IEEE: http://ieeexplore.ieee.org/document/7123719/

   Paper: http://nanne.github.io/papers/Noord2015.pdf

4. Train/test data

   GitHub: https://github.com/zo7/painter-by-numbers/releases/tag/data-v1.0

5. Web Service

   GitHub: BME-SmartLab-Education / 2016-train-validate-test-repeat / Web Service

6. Web Client

   GitHub: BME-SmartLab-Education / 2016-train-validate-test-repeat / Web Client

7. iOS App

   GitHub: BME-SmartLab-Education / 2016-train-validate-test-repeat / iOS