# Festők azonosítása festményeik stílusjegyei alapján

## *Identifying painters by the stylistic features of their paintings*

Bajkó Norbert, Kohlmann András, Mikulás Bence, Nagy Péter

***Kivonat* — A mély neurális hálózatok 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 opportunities appearing for application in the scope of image processing. A practical 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 analyse the style of a given painting and identify its painter. Our task was to train a convolutional neural network (CNN) to recognize the style of selected painters, and later to determine the origin of a painting received as input. As for 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 it 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.**

Ez a dokumentum a Budapesti Műszaki és Gazdaságtudományi Egyetem fent megnevezett tárgyának keretein belül keletkezett, 2016 őszén.
Az összes szerző egyenlően kontributált a dokumentumba, csapatmunka révén.

## I. 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[1].
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[6]. Harmadik lépésként szeretnénk a rendszerünket valamilyen módon a "külvilághoz kapcsolni", például mobilos és/vagy webes kliens alkalmazások képében.

## II. Tématerület ismertetése, korábbi megoldások

A Painting Mona Lisa tanulmány a kép generálás egy lehetséges módszerét mutatja be, számunkra egy ilyen megoldás megismerése miatt volt hasznos. Az iterációkban elkészülő megoldás alapgondolatait használjuk fel a képszintézisnél.[2][3][4][5]

Bár eredetileg egy kép tulajdonságainak átvitelére szolgál a Style Transfer, jelenleg már ezt használjuk a festők stílusának rekreációjához Deep Dream helyett.[6][8][12]

Témakört tekintve a Toward Discovery of the Artist's Style című munka kapcsolódik legjobban a mi feladatunkhoz. A publikáció témája a képek stílusainak detektálása, akár úgy is, ha a képen több festő dolgozott.[7]

## III. Megvalósítás

### *A. Adatok beszerzése, előkészítése*

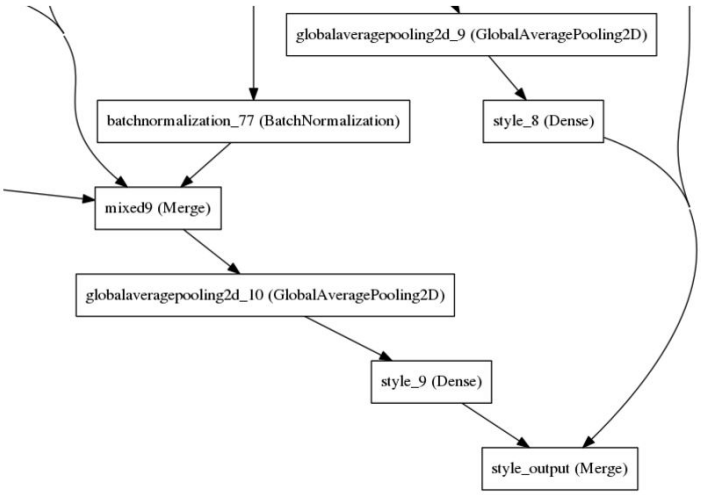
A feladatunk mindkét részéhez ugyanazt a tanító halmazt használtuk, amelyet a Kaggle versenyhez adtak. A tanító adat

körülbelül 80 ezer képet tartalmaz, amelyekhez egy csatolt .csv fájlban érhetőek el a metaadatok. A halmaz 20%-át elizoláltuk validációs célokra. Később, a tényleges tanításkor sosem tanítottunk ennyi képpel, valójában körülbelül ennek 1/20 részével (3200/800 képpel) dolgoztunk. A csökkentett tanítóhalmazt a leginkább reprezentált festők műveiből szelektáltuk, ezáltal egyszerűsítve a feladatot. Mivel a fórumon elérhető volt egy felhasználó által előfeldolgozott változata a képeknek, amelyek kezelhető méretűre voltak kicsinyítve, ezért a tanításra inkább azokat használtuk.[14] Ezekkel az egyszerűsítésekkel bár a képek részletgazdagsága csökkent, valamint a tanítóhalmaz is redukálva lett, 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.

### *B. 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 Google által fejlesztett TensorFlow-t használja. 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 encoding-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. [9] 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.




*1. ábra*

Ezen információk felhasználása érdekében, a modellben található merge pontokat előrekötöttük az osztályozást végző fully-connected rétegbe, ahogyan azt az 1. ábra mutatja, ezzel javítva a hálózat pontosságán. Ezen pontok szolgálnak bottleneckként az Inception-ben, ezért feltételeztük, hogy ezekben a rétegekben akkumulálódik a stílusinformáció.

## IV. KIÉRTÉKELÉS (TANÍTÁSI, VALIDÁCIÓS ÉS TESZT HIBA), HIPERPARAMÉTER OPTIMALIZÁLÁS

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

### *A. Az optimalizációt összefoglaló táblázatok*

| Dense | Dropout | Dense | F1 batch | F2 batch |
|---|---|---|---|---|
| 512 | 0.5 | 512 | 64 | 64 |
| 1024 | 0.5 | 512 | 64 | 64 |
| 1024 | 0.5 | 1024 | 64 | 64 |
| 2048 | 0.5 | 1024 | 64 | 64 |
| 2048 | 0.5 | 2048 | 64 | 64 |
| | | | | |
| 2048 | 0.3 | 1024 | 64 | 64 |
| 2048 | 0.5 | 1024 | 64 | 64 |
| 2048 | 0.7 | 1024 | 64 | 64 |
| | | | | |
| 1024 | 0.3 | 1024 | 64 | 64 |
| 1024 | 0.5 | 1024 | 64 | 64 |
| 1024 | 0.7 | 1024 | 64 | 64 |
| | | | | |
| 1024 | 0.5 | 1024 | 16 | 16 |
| 1024 | 0.5 | 1024 | 32 | 16 |
| 1024 | 0.5 | 1024 | 32 | 32 |
| 1024 | 0.5 | 1024 | 64 | 32 |
| 1024 | 0.5 | 1024 | 64 | 64 |

| | | | | |
|---|---|---|---|---|
| 1024 | 0.5 | 1024 | 128 | 64 |
| 1024 | 0.5 | 1024 | 128 | 128 |
| 1024 | 0.5 | 1024 | 256 | 128 |
| 1024 | 0.5 | 1024 | 256 | 256 |

*1. táblázat*

Az *1. táblázat* foglalja össze a hiperparaméter optimalizáláskor alkalmazott paraméterértékeket. A *2. táblázat* és a *3. táblázat* sorai rendre megfelelnek ezen táblázat sorainak.

| loss | acc | val_loss | val_acc |
|---|---|---|---|
| 0.1397 | 0.9619 | 0.5934 | 0.8277 |
| 0.0816 | **0.9780** | 0.5379 | 0.8491 |
| 0.0895 | 0.9739 | **0.5178** | **0.8503** |
| **0.0795** | 0.9758 | 0.6135 | 0.8151 |
| 0.0887 | 0.9733 | 0.6059 | 0.8226 |
| | | | |
| **0.0551** | **0.9865** | **0.5507** | **0.8289** |
| 0.0795 | 0.9758 | 0.6135 | 0.8151 |
| 0.1276 | 0.9591 | 0.6479 | 0.8176 |
| | | | |
| **0.0542** | **0.9868** | 0.5558 | **0.8415** |
| 0.0895 | 0.9739 | **0.5178** | 0.8252 |
| 0.1695 | 0.9509 | 0.5591 | 0.8214 |
| | | | |
| 0.1834 | 0.9440 | 0.7214 | 0.8226 |
| 0.1671 | 0.9515 | 0.7904 | 0.8151 |
| 0.1121 | 0.9704 | **0.5823** | **0.8327** |
| 0.1112 | 0.9666 | 0.6793 | 0.8239 |
| 0.1069 | 0.9717 | 0.6413 | 0.8264 |
| 0.1048 | 0.9688 | 0.6495 | 0.8164 |
| **0.0918** | **0.9745** | 0.6123 | 0.8252 |
| 0.1455 | 0.9604 | 0.5913 | 0.8138 |
| 0.1802 | 0.9462 | 0.7559 | 0.7585 |

*2. táblázat*

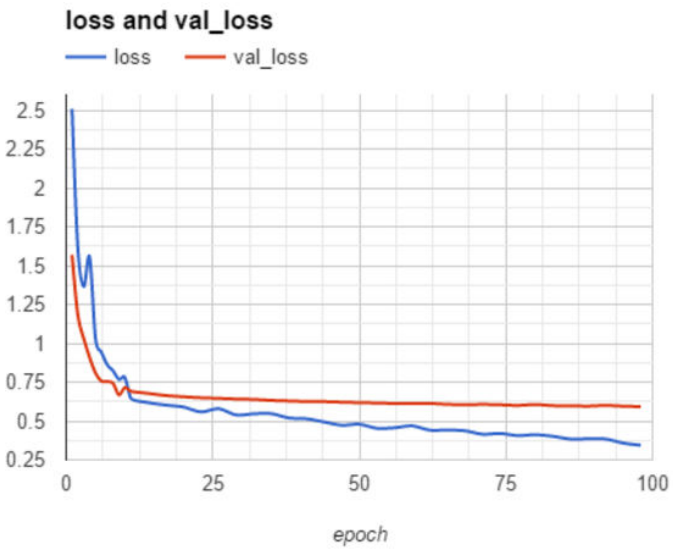
A *2. táblázat* az eredmények változásainak táblázata, melyben látható, melyik paraméterkombináció volt a legalkalmasabb.

| loss change | acc change | val_loss change | val_acc change |
|---|---|---|---|
| 0.00% | 0.00% | 0.00% | 0.00% |
| 41.59% | 1.67% | 9.35% | 2.59% |
| 35.93% | 1.25% | 12.74% | 2.73% |
| 43.09% | 1.45% | -3.39% | -1.52% |
| 36.51% | 1.19% | -2.11% | -0.62% |
| | | | |
| 0.00% | 0.00% | 0.00% | 0.00% |
| -44.28% | -1.08% | -11.40% | -1.66% |
| -131.58% | -2.78% | -17.65% | -1.36% |
| | | | |
| 0.00% | 0.00% | 0.00% | 0.00% |
| -65.13% | -1.31% | 6.84% | -1.94% |
| -212.73% | -3.64% | -0.59% | -2.39% |
| | | | |
| 0.00% | 0.00% | 0.00% | 0.00% |
| 8.89% | 0.79% | -9.56% | -0.91% |
| 38.88% | 2.80% | 19.28% | 1.23% |
| 39.37% | 2.39% | 5.84% | 0.16% |
| 41.71% | 2.93% | 11.10% | 0.46% |
| 42.86% | 2.63% | 9.97% | -0.75% |
| 49.95% | 3.23% | 15.12% | 0.32% |
| 20.67% | 1.74% | 18.03% | -1.07% |
| 1.74% | 0.23% | -4.78% | -7.79% |

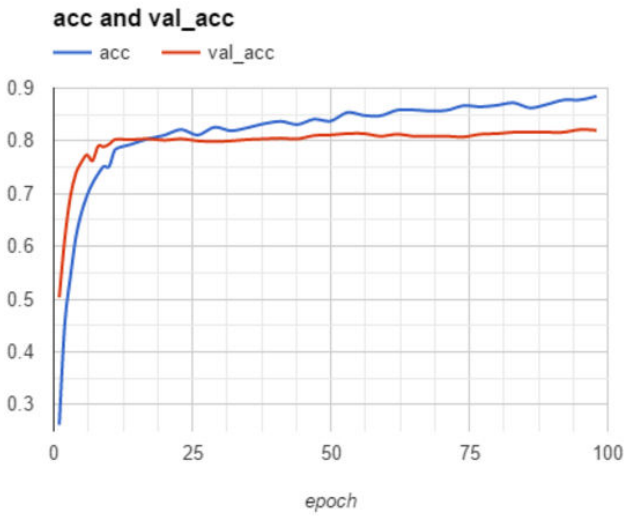
*3. táblázat*

Végül pedig a *3. táblázat* az eredmények változását bemutató táblázat, amelyen a színkód jelzi a változás milyenségét.

*B. A veszteség és a pontosság görbéje*




*2. ábra*




*3. ábra*

A 2. ábra a hiba, a 3. ábra pedig a pontosság görbéit mutatja. Ennél a tanításnál a paraméterek az alábbiak voltak: 3 db Dense: 1024, 3 db Dropout: 0.7, Batch: 32, ADAM + SGD, Epoch: 300 + EarlyStopping

### *C. 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.

### *D. Képgenerálás*

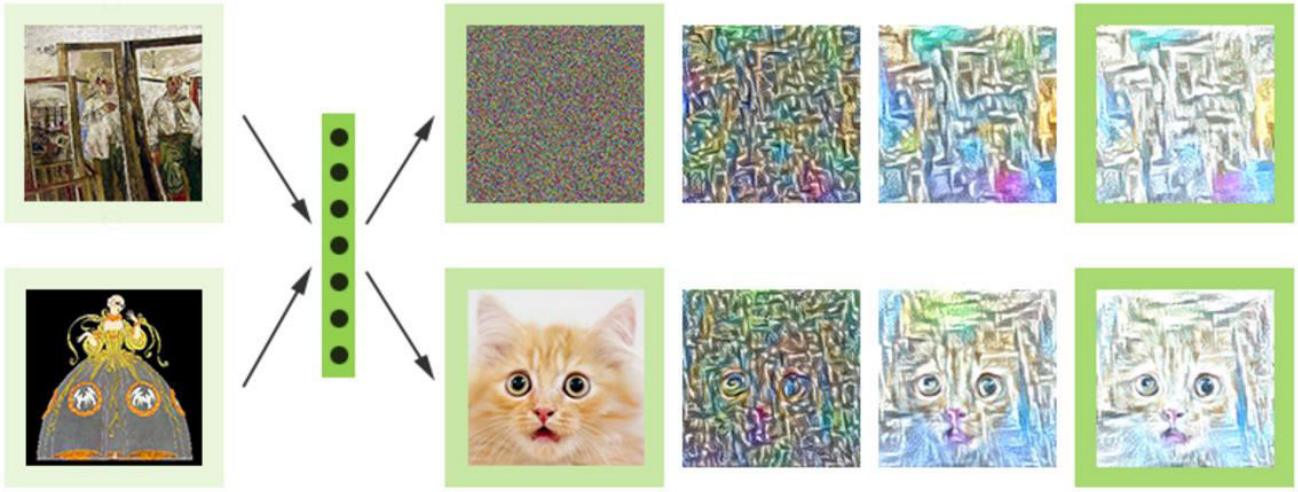
A kitűzött második feladatunk során egy festőre jellemző stílussal terveztük képet generálni. Erre két megközelítési módot találtunk, mely lényegében a style-transfernél megismert technikát alkalmazza[6], miszerint egy kép tartalmának megváltoztatása nélkül a kép stílusát egy forrásképhez igazítja. Az 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.

A metódus alapgondolata, hogy a festményekkel tanított hálót, amely elkészült az első feladatban használjuk fel a style-transferre, így biztosítva egy pontosabb stílusjellemzőkkel bíró végeredményt.

Amennyiben a Style Transferben is használt hálót (VGG16)[10] 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.

A stílust egy köztes adatszerkezettel reprezentálva szintetizáljuk a képet, ahogyan az a 4. ábrán is látszik, a baloldalon szereplő képekből kinyertük a stílusinformációt, majd a jobboldal első sorában zajt, a második sorában pedig egy képet használva fel tartalmi alapul az oszlopokban jobbra haladva az egyes iterációk láthatóak.

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.



*4. ábra*

### *E. Interfész*

#### *1) Webszolgáltatás*

A hálóhoz készült egy webszolgáltatás és két kliens alkalmazás. A webszolgáltatás a hálóból kiexportált modellt 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.[15]

#### *2) 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 modellt kellett kiexportálni a hálóból.[16]

#### *3) 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.[17]

## V. JÖVŐBELI TERVEK

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.

Az Inception V3 további tanulmányozásával párhuzamba állítva a VGG16 style-transferben használt rétegeivel felfedezhetünk megfeleltetéseket, amik segítségével a képszintézist végző hálózat architektúráját lecserélhetjük Inception V3-ra a mostani VGG-ről vagy ha nem találunk megfeleltetést, akkor később az egyszerűbb AlexNet-re.[13]

A tanítást megismételhetjük más tanító és validációs halmazzal, illetve a speciális esetek manuális vizsgálatával pontosíthatunk az eredményeken.

További célunk, hogy egy a festő csoportok közötti tévesztési mátrixot (confusion matrix) és a hozzá tartozó értékeket szeretnénk megalkotni. Majd ezután, egy a témakörben használt speciális mérőszámot is szeretnénk alkalmazni az eredmények kiértékelésére.

## VI. 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, hiszen egy többosztályos osztályozási feladatnál az eredményeink megfelelő pontosságúak, ráadásul az eredeti feladatban egy döntési feladat volt, amit kiegészítettünk. A sok hasonló festmény ellenére képesek voltunk elérni, hogy 10 festő esetén a képek 80%-ánál, 100 festő esetén 60%-ánál a valódi alkotó áll a hálózat által definiált valószínűségi sor legelső helyen.

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] "Painter by Numbers," 2016. [Online]. Available: https://www.kaggle.com/c/painter-by-numbers
[Accessed: 17-Dec-2016].

[2] "MonaLisa," 2013. [Online]. Available: https://github.com/evolvingstuff/MonaLisa
[Accessed: 17-Dec-2016].

[3] Tom Lahore, "Generating Mona Lisa Pixel By Pixel," 2012. [Online]. Available: http://evolvingstuff.blogspot.hu/2012/12/generating-mona-lisa-pixel-by-pixel.html
[Accessed: 17-Dec-2016].

[4] Tom Lahore, "Learning to Generate Mona Lisa, Animated," 2012. [Online]. Available: http://evolvingstuff.blogspot.hu/2012/12/learning-to-generate-mona-lisa-animated.html
[Accessed: 17-Dec-2016].

[5] "Learning to generate image of Mona Lisa using deep neural networks," 2012. [Online]. Available: http://www.reddit.com/r/programming/comments/15qj3p/
[Accessed: 17-Dec-2016].

[6] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge, "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015). [Online]. Available: https://arxiv.org/abs/1508.06576
[Accessed: 17-Dec-2016].

[7] Nanne van Noord, Nanne, Ella Hendriks, and Eric Postma, "Toward Discovery of the Artist's Style: Learning to recognize artists by their artworks." IEEE Signal Processing Magazine 32.4 (2015): 46-54. [Online]. Available: https://arxiv.org/abs/1508.06576
[Accessed: 17-Dec-2016].

[8] Nanne van Noord, Nanne, Ella Hendriks, and Eric Postma, "Image Style Transfer Using Convolutional Neural Networks" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. APA [Online]. Available: http://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Gatys_Image_Style_Transfer_CVPR_2016_paper.html
[Accessed: 17-Dec-2016].

[9] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision" arXiv preprint arXiv:1512.00567 (2015). [Online]. Available: https://arxiv.org/abs/1512.00567
[Accessed: 17-Dec-2016].

[10] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition" arXiv preprint arXiv:1409.1556 (2015). [Online]. Available: https://arxiv.org/abs/1409.1556
[Accessed: 17-Dec-2016].

[11] Matthew D Zeiler, Rob Fergus, "Visualizing and Understanding Convolutional Networks" arXiv preprint arXiv:1311.2901 (2013). [Online]. Available: https://arxiv.org/abs/1311.2901
[Accessed: 17-Dec-2016].

[12] Alexander Mordvintsev, Christopher Olah, Mike Tyka, "Inceptionism: Going Deeper into Neural Networks" Google Research Blog (2015). [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
[Accessed: 17-Dec-2016].

[13] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks" In NIPS, 2012. [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[Accessed: 17-Dec-2016].

[14] "Train/test data" [Online]. Available: https://github.com/zo7/painter-by-numbers/releases/tag/data-v1.0
[Accessed: 17-Dec-2016].

[15] "Web Service" [Online]. Available: https://github.com/BME-SmartLab-Education/vitmav45-2016-train-validate-test-repeat/tree/master/web_service
[Accessed: 17-Dec-2016].

[16] "Web Client" [Online]. Available: https://github.com/BME-SmartLab-Education/vitmav45-2016-train-validate-test-repeat/tree/master/demo_website
[Accessed: 17-Dec-2016].

[17] "iOS App" [Online]. Available: https://github.com/BME-SmartLab-Education/vitmav45-2016-train-validate-test-repeat/tree/master/iOS_Client
[Accessed: 17-Dec-2016].