Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Kettős expozíciós képek készítése mély tanulás alapon

Tudományos Diákköri Konferencia
2018.

Készítette
Almási Péter Béla

Konzulens
Dr. Gyires-Tóth Bálint

2018. október 28.

Tartalomjegyzék

Kivonat

A mesterséges intelligencia területei közül az elmúlt években a neurális hálózatok mentek keresztül talán a legnagyobb fejlődésen (*deep learning*). A grafikus számítási egységek (GPU, Graphical Processing Unit) fejlődése, az algoritmikus fejlesztések és a felhőszolgáltatásokban elérhető óriási számítási kapacitás lehetővé tették, hogy hatékonyan tudjunk dolgozni velük. Neurális hálózatok felhasználásával számos területen sikerült minden eddiginél jobb (és sok esetben az ember címkézésénél is pontosabb) eredményeket elérni: például a kép- és beszédfelismerés, nyelvfeldolgozás, beszédszintézis, idősor-analízis, ajánlórendszerek, valamint a megerősítéses tanulás területén is átütő sikereket értek el az utóbbi években.

A különböző területeken általában speciális hálózatokat használnak; képfeldolgozásban a konvolúciós hálózatok terjedtek el. Ezekkel számos különféle képfeldolgozási feladatot meg lehet oldani: képen és videón objektumok felismerésére, képszegmentációra, képek hibáinak javítására, vagy akár képgenerálásra is használhatók ilyen hálózatok. Bizonyos konvolúciós architektúrák képesek egy képen pixelpontosan meghatározni a rajta levő objektumok helyét. Ilyen hálózatok használhatóak például az önvezető autók kameraképeinek elemzéseihez, ahol nem csak annyit kell eldönteni, hogy van-e valahol másik autó vagy gyalogos a képen, hanem azok pontos helyét is meg kell határozni.

Dolgozatomban ismertetem egy olyan hálózat elkészítését és tanítását, amely képes pixelpontosan meghatározni, hogy az adott képen hol találhatók állatok, és ez alapján képes körbevágni a képről az állatokat, eltávolítva az eredeti hátteret. Ezt felhasználva elkészítettem egy demonstrációs mintarendszert, amellyel művészi hatású, úgynevezett kettős expozíciós (*double exposure*) képeket lehet készíteni. Ez egy rendkívül kedvelt és népszerű képkészítési eljárás grafikusok és fotósok körében. A kép rendszerint egy figura és egy természeti kép áttűnését jeleníti meg; előbbi lehet állat, emberi arc, vagy egy tetszőleges alak sziluettje. Jelen esetben a képeken egy állat és egy tájkép alkotja a kompozíciót. A dolgozat keretében elkészített megoldásom tetszőleges állatalakot tartalmazó képből és tájképből automatikusan képes kettős expozíciós, művészi hatású képeket generálni.

Abstract

Among the areas of artificial intelligence, neural networks have undergone some of the greatest advances in recent years. The improvement of GPUs (Graphical Processing Units), algorithmic developments and the large computing power available in cloud computing services have made training neural networks possible in reasonable time. Neural networks have reached state of the art results in many areas, including image and speech recognition, natural language processing, speech synthesis, recommendation systems, time series analysis, and reinforcement learning. These results are often more accurate than those produced by humans.

The different areas often use specialized networks to reach the best results: in image processing, convolutional neural networks have become the most successful ones. Using these networks, we can solve many image processing tasks, including, but not limited to, image recognition, image segmentation, error correction on images, and image generation. Certain convolutional networks are capable of determining the exact location of objects on a picture with pixel-level precision. Such networks can be used to analyze the input from cameras in self-driving cars, where it is necessary not only to recognize that there is a pedestrian or another car somewhere in the picture, but its exact location has to be identified as well.

In this work, I present a neural network that was trained to segment the animals on images, which automatically allows removing the background of an animal. Using this network I created an application which creates artistic double exposure images. It is a very popular image creating method among designers and photographers. These images are usually made of two pictures: one of a figure, which can be an animal, human face, or any kind of silhouette, and one of a landscape. I used animal images and landscapes to create the compositions. The proposed program can generate double exposure images by automatically combining almost any kind of animal image and landscape image.

1. fejezet

Bevezető

A mesterséges intelligencia napjainkban jelentős fejlődésen megy keresztül. Ennek köszönhetően számtalan olyan programban használják, amikkel a mindennapjaink során rendszeresen találkozunk: legyen szó akár intelligens személyi asszisztensekről, zene- vagy videószolgáltatók ajánlórendszeréről, többnyelvű fordítóprogramról vagy akár egy okostelefonos kameraalkalmazásról, mindegyikben megtalálhatók ilyen algoritmusok.

A mesterséges intelligencia területei közül az elmúlt években a mély tanulás paradigmán alapuló neurális hálózatok kapták talán a legnagyobb figyelmet (*deep learning*). A grafikus számítási egységek (GPU, Graphical Processing Unit) fejlődése, az algoritmikus fejlesztések, a rendelkezésre álló hatalmas adatbázisok és a felhőszolgáltatásokban elérhető óriási számítási kapacitás lehetővé tették, hogy hatékonyan tudjunk dolgozni velük. Neurális hálózatok felhasználásával számos területen sikerült minden eddiginél jobb (és sok esetben az ember címkézésénél is pontosabb) eredményeket elérni: például a kép- és beszédfelismerés, nyelvfeldolgozás, beszédszintézis, idősor-analízis, ajánlórendszerek, valamint a megerősítéses tanulás területén is átütő sikereket értek el az utóbbi években.

A különböző területeken általában speciális hálózatokat használnak; képfeldolgozásban a konvolúciós hálózatok terjedtek el. Ezekkel számos különféle képfeldolgozási feladatot meg lehet oldani: képen és videón objektumok felismerésére, képszegmentációra, képek hibáinak javítására, vagy akár képgenerálásra is használhatók ilyen hálózatok. Bizonyos konvolúciós architektúrák képesek egy képen pixelpontosan meghatározni a rajta levő objektumok helyét. Ilyen hálózatok használhatóak például az önvezető autók kameraképeinek elemzéseihez, ahol nem csak annyit kell eldönteni, hogy van-e valahol másik autó vagy gyalogos a képen, hanem azok pontos helyét is meg kell határozni.

A klasszikus gépi tanulási feladatok mellett a mély tanulás egyik rendkívül érdekes ága lett az ún. neurális művészet (*neural art*). Ez annyit jelent, hogy a mély neurális hálózatok által egy adott adathalmazból megtanult valószínűségi eloszlásokat különféleképpen paraméterezve és felhasználva művészi hatású produktum előállítására használjuk fel. Ilyen produktum lehet kép, videó, hangeffektus vagy például zene. A neurális művészet talán legismertebb változata az úgynevezett neurális stílustranszformáció. Ennek lényege, hogy a neurális hálózatok képesek megtanulni például egy festő képeire jellemző jellegzetességeket, a festő által használt egyedi stílusjegyeket, és ezek alapján képesek bármilyen képet

átrajzolni olyan stílusúvá, mintha az adott festő festette volna azt.

Dolgozatomban ismertetem egy olyan hálózat elkészítését és tanítását, amely képes pixelpontosan meghatározni, hogy egy képen hol találhatók állatok, és ez alapján képes körbevágni a képről az állatokat, eltávolítva az eredeti hátteret. Ezt felhasználva elkészítettem egy demonstrációs mintarendszert, amellyel művészi hatású, úgynevezett kettős expozíciós (*double exposure*) képeket lehet készíteni. Néhány példa művészek által készített kettős expozíciós képekre az 1.1. ábrán látható. Ez egy rendkívül kedvelt és népszerű[1] képkészítési eljárás grafikusok és fotósok körében. A kép rendszerint egy figura és egy természeti kép áttűnését jeleníti meg; előbbi lehet állat, emberi arc, vagy egy tetszőleges alak sziluettje. Jelen esetben a képeken egy állat és egy tájkép alkotja a kompozíciót. A dolgozat keretében elkészített megoldásom tetszőleges állatalakot tartalmazó képből és tájképből automatikusan képes kettős expozíciós, művészi hatású képeket generálni. Az általam elkészített rendszer újdonságát az adja, hogy korábban még nem használtak mély neurális hálózatokat kettős expozíciós képek készítésére; így az eddigieknél sokkal gyorsabban, művészi hozzáértés és képszerkesztő programok ismerete nélkül is bárki készíthet ilyen képeket. A rendszer kipróbálható a *http://deep2.tmit.bme.hu:5000/* címen.

1.1. ábra. *Művészek által készített kettős expozíciós képek. Források: https://www.boredpanda.com/double-exposure-animal-portraits-by-norwegian-photographer/, https://www.demilked.com/double-exposure-animal-portraits-faunascapes-whatwedo-denmark (Letöltés időpontja: 2018. 10. 03.)*

A dolgozat felépítése a következő: a 2. fejezetben bemutatom a kapcsolódó tudományos eredményeket a képszegmentáció és a neurális művészet témaköréből, a 3. fejezetben röviden bemutatom az elkészített rendszer működését és részeit, a 4. fejezetben részletesen ismertetem a képszegmentációt megvalósító neurális hálózatot, az 5. fejezetben pedig az állatfejek irányát meghatározó neurális hálózatot, a 6. fejezetben bemutatom a művészi hatású kettős expozíciós kép elkészítésének folyamatát, végül a 7. fejezetben összefoglalom az elvégzett munkát.

[1] Egy Google keresés a *double exposure* kulcsszóra 209 millió találatot eredményez, az elmúlt hétre szűkített keresés pedig 312 ezret (2018. 10. 26-i adatok alapján).

2. fejezet

Irodalomkutatás

Ebben a fejezetben ismertetem a dolgozatom témájához kapcsolódó tudományos cikkeket és a *state of the art* megoldásokat. A fejezetben először képfelismerést, illetve képszegmentációt megvalósító neurális hálózatok kerülnek bemutatásra, majd szó esik a neurális művészet területén elért fontosabb eredményekről is.

2.1. Képszegmentáció

A képszegmentációt megvalósító neurális hálózatok előtt röviden bemutatom a mély tanulás képfelismeréssel és objektumdetekcióval kapcsolatos fontosabb eredményeit az elmúlt évekből.

2.1.1. Képfeldolgozás neurális hálózatokkal

A számítógépes képfeldolgozás különböző területein az elmúlt években számtalan, minden eddiginél jobb eredményt értek el mély neurális hálózatok felhasználásával. Az egyik első ilyen eredmény a 2012-es ImageNet verseny megnyerése volt a Torontói Egyetem kutatói által elkészített, AlexNet névre keresztelt neurális hálózattal [1]. Az ImageNet egy rendkívül nagyszabású verseny, amelyen az elsődleges feladat képeken található objektumok felismerése[1]. A versenyen több millió kép alkotja a tanításhoz felhasználható adatbázist, melyek több, mint húszezer kategóriába vannak beosztva a rajta található objektum szerint, így egy-egy kategóriába jellemzően néhány száz vagy ezer kép tartozik. Néhány példa ezekre a 2.1a. ábrán látható. Az AlexNet egy konvolúciós neurális hálózat; képfeldolgozáshoz azóta is ezek a hálózattípusok bizonyulnak legsikeresebbnek. A konvolúciós rétegek mellett még MaxPooling és előrecsatolt rétegek alkotják a hálózatot.

Az AlexNet sikere után az elkövetkező években is a konvolúciós neurális hálózatok arattak győzelmet az ImageNet versenyen. 2013-ban az Oxfordi Egyetemen elkészített VGG-16 nevű hálózat lett a legjobb [2]. Ez a hálózat ez előbbinél mélyebb, azaz több konvolúciós réteget használ. Egyszerű struktúrája miatt gyakran használják komplexebb feladatok megoldására készített hálózatok alapjául.

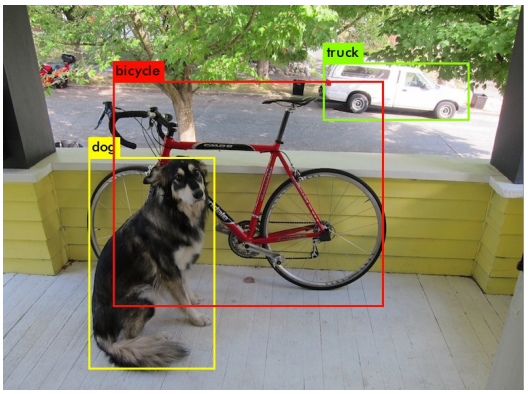
[1] A verseny azóta kiegészült, az elmúlt évben már objektumlokalizáció és videó alapján történő objektumfelismerés is szerepelt a feladatok között.

A Google mérnökei által készített GoogLeNet a 2014-es évben érte el a legnagyobb pontosságot az ImageNet versenyen [3]. Ez az előzőeknél már komplexebb felépítésű hálózat; a jobb eredmények eléréséhez úgynevezett *inception module*-okból építettek fel egy, az eddigiekhez képest még több rétegből álló hálózatot. Az *inception module*-ok kisebb blokkok a hálózatban, amelyekben különböző szűrőméretű konvolúciós rétegekkel elemzik az előző réteg eredményeit, majd ezeket a részeredményeket egyesítik.

A Microsoft által kifejlesztett ResNet hálózat 2015-ban nyerte meg az ImageNet versenyt [4]. A hálózat minden eddiginél több, 152 rétegből áll. Az ilyen mélységű hálózat tanítása komoly problémát jelenthet, ennek megoldására úgynevezett *residual block*-okat használtak, amelyek lényege, hogy az egyes rétegek képesek a bemenetüket közvetlenül átadni a következő rétegnek, így kiküszöbölve a túlságosan mély hálózatok tanításakor jelentkező problémákat (eltűnő és felrobbanó gradiens). Ez a hálózat a verseny során a tesztképeket 3,57%-os hibaaránnyal ismerte fel – ez jobb, mint amire egy átlagos ember képes (kb. 5%-os hiba).

Ezek a hálózatok – amellett, hogy az ImageNet versenyen kiemelkedő eredményeket értek el – rengeteg tudást hordoznak magukban. A hálózatok alsóbb rétegei képesek felismerni egyszerűbb elemeket (pl. vonalak, görbék, egyszerű alakzatok) a képeken, amely elemekből a későbbi rétegek felismerik a képen látható objektumot. Ezt a tudást felhasználhatjuk, ha olyan képfeldolgozási problémát szeretnénk megoldani, amely esetében a feldolgozandó képek tematikájukban hasonlítanak az ImageNet versenyben levőkhöz. Ezért képfeldolgozási feladatoknál gyakori módszer, hogy ezeket a hálózatokat használjuk kiindulásként, és bizonyos rétegeit "továbbtanítva" módosítjuk a hálózatot, hogy az aktuális feladatnak megfelelően működjön (*transfer learning*). Így a hálózatok gyorsabban lesznek képesek megoldani a kívánt feladatot, mint ha egy teljesen új hálózatot kezdenénk el tanítani véletlenszerű kiindulási súlyokkal.

(a) *Képek klasszifikálása (AlexNet által készített predikciók).*



(b) *Objektumdetekció (YOLO hálózat által).*

2.1. ábra. *Képek klasszifikációja és objektumdetekció. (A képek az AlexNet cikkből, illetve a YOLO projekt weboldaláról (https://pjreddie.com/darknet/yolo/) származnak.)*

2.1.2. Objektumok helyének meghatározása

A képek osztályozása után a következő lépés a képszegmentáció felé a képeken található objektumok helyének meghatározása (*object localization*). Ez annyit jelent, hogy a hálózatnak, amellett, hogy felismeri a képen levő objektumokat, azok helyét is meg kell határoznia (tkp. egy téglalapot elhelyezve köré). Erre látható egy példa a 2.1b. ábrán.

Egyszerűbb esetben csak egy objektum található a képen, amit fel kell ismerni. Ekkor a korábbiakban bemutatott hálózatok kiegészíthetők további kimenetekkel úgy, hogy ezekben az objektumok helyét is megbecsüljék.

Bonyolultabb esetben több (változó számú) objektum található a képeken, amelyek mindegyikét fel kell ismerni, és a helyüket is meg kell határozni. Ez az eddigieknél jóval komplexebb feladat, megoldásához összetettebb eszközökre van szükség. A korábbi megoldások azért sem használhatók fel itt egy az egyben, mert több objektum esetén a kimenet mérete nem rögzített: míg az egyszerű képfelismerési feladatnál egy adott hosszúságú vektor a neurális hálózat kimenete, addig itt a kimenet mérete attól függ, hogy hány darab objektum található a képen.

Objektumdetekciós feladatok megoldására két hálózatot mutatok be: az R-CNN-t, és a YOLO algoritmust. Az R-CNN [5] működése a következő: a képen kijelölnek bizonyos régiókat, ezeken futtatják a konvolúciós neurális hálózatokat, és a konvolúciós hálózatok kimenete alapján SVM-ekkel (Support Vector Machine) határozzák meg az objektumok helyét az eredeti képen. A régiók kiválasztásához az úgynevezett szelektív keresés algoritmust használják, ami kijelöl a képeken olyan régiókat, ahol a felismerendő objektumok potenciálisan elhelyezkedhetnek, így nem kell minden lehetséges téglalapot végignézni a képen. A probléma ezzel a módszerrel az, hogy a régiók kijelölése és az ezeken történő predikció nagyon lassúvá teszi a működést. Ezért elkészítették az R-CNN módosított, fejlettebb változatait. A Fast R-CNN [6] a régiók helyett az egész képet használja a konvolúciós hálózat bemeneteként, így megspórolva azt a többletszámítást, ami az átlapolódó régiók esetén jelentkezik az első változatban. Ebben a régiók kijelölése még mindig a szelektív keresés algoritmussal történt, ami nagymértékben lelassította a predikciók készítését. A további gyorsítás érdekében a régiók kijelölését egy konvolúciós neurális hálózatra bízták, így megszületett a Faster R-CNN [7]. Ezzel már közel valós időben megvalósítható az objektumdetekció, miközben a pontossága is javult a korábbi változatokhoz képest.

A különböző R-CNN változatok mellett egy másik, objektumdetekciót megvalósító hálózat a YOLO (You Only Look Once) [8]. Ez az előzőekben bemutatott R-CNN változatokhoz képest teljesen új módon közelíti meg a feladatot: régiók kijelölése és azokon klasszifikáló konvolúciós hálózatok futtatása helyett itt regressziós feladatként kezelik az objektumdetekciót. Egyetlen neurális hálót használnak, aminek a bemenete a kiindulási kép, amin az objektumokat fel kell ismerni, kimenetén pedig egyből az objektumokat tartalmazó téglalapok koordinátái és a hozzájuk tartozó valószínűségek jelennek meg. Ezáltal a korábbi módszerekhez képest sokkal gyorsabban működik a felismerés (valós idejű feldolgozást is lehetővé téve), az egész hálózat egyben tanítható. Valamint azáltal, hogy az objektumok felismerésekor nem csak azok közvetlen környezetét látja a neurális hálózat, hanem az egész

képet, jobban kontextusba tudja helyezni az objektumokat, így pontosabb lesz a felismerés, csökken a háttérben lévő véletlenszerű mintázatokra adott téves predikciók száma.

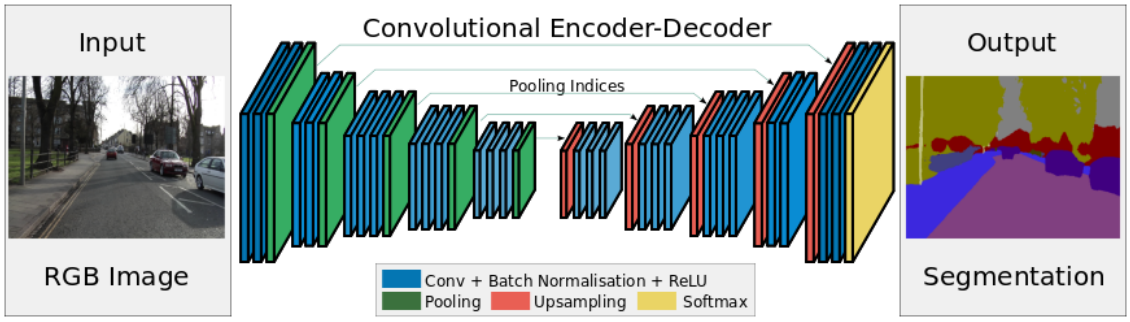
2.1.3. Képszegmentáció

A képek klasszifikációja és a képeken található objektumok helyének detekciója után a következő lépés a képek még pontosabb feldolgozásához a képszegmentáció (*semantic segmentation*). Ez azt jelenti, hogy a képen található objektumok helyét nem csak közelítőleg kell megjelölni egy-egy őket határoló téglalappal, hanem pixelpontossággal meg kell állapítani, hogy a képen egészen pontosan hol helyezkednek el. Erre látható egy példa a 2.2. ábrán. Másképp úgy is fogalmazhatnánk, hogy a képet különböző részekre (szegmensekre) kell felbontani úgy, hogy minden szegmens egy-egy objektumot tartalmazzon; innen származik az elnevezése. Ez a neurális hálózatok szempontjából azt jelenti, hogy nem csak a képre vagy egy-egy régióra kell predikciókat készíteni, hanem a kép minden egyes pixelét be kell sorolni az adott kategóriák valamelyikébe (háttér és felismerendő objektumok).

Képszegmentációval már a deep learning elterjedése előtt is foglalkoztak, de sok más területhez hasonlóan itt is jelentős javulást értek el a mély tanuló algoritmusok felhasználásával, ezért itt csak a mély tanulás alapú képszegmentációs eredményeket ismertetem. A korábbiakhoz hasonlóan képszegmentáció esetében is a konvolúciós neurális hálózatok meghatározóak, egy ilyen látható [10]-ben is. Kiindulási alapnak az ImageNet adatbázison tanított hálózatokat használták (pl. AlexNet, VGG, GoogLeNet), amelyeket átalakítottak teljesen konvolúciós hálózatokká (az előrecsatolt rétegek konvolúciós réteggé transzformálásával). Az így kapott hálózatokat end-to-end módon tudják tanítani, vagyis nincs szükség további előfeldolgozási lépésekre és utómunkálatokra ahhoz, hogy megkapjuk a modell predikcióját. Mivel a kiindulási ImageNet-es hálózatok tartalmaznak MaxPooling rétegeket, ezért az előállított kimenet az eredetinél kisebb felbontású lenne; ennek kiküszöbölésére úgynevezett dekonvolúciós rétegeket használnak. Ezek úgy működnek, mintha "megfordítanánk" a konvolúciós rétegek működését, felcserélve az előre- és visszairányt. Mivel a Pooling rétegeknél jelentős mennyiségű információvesztés történik, ezért önmagukban a de-

(a) *Eredeti kép.* **(b)** *A képen megjelölt objektumok.*

2.2. ábra. *Képszegmentáció során a képen található objektumok helyét pixelpontossággal kell meghatározni. (A kép az MSCOCO adatbázisból származik ([9].))*

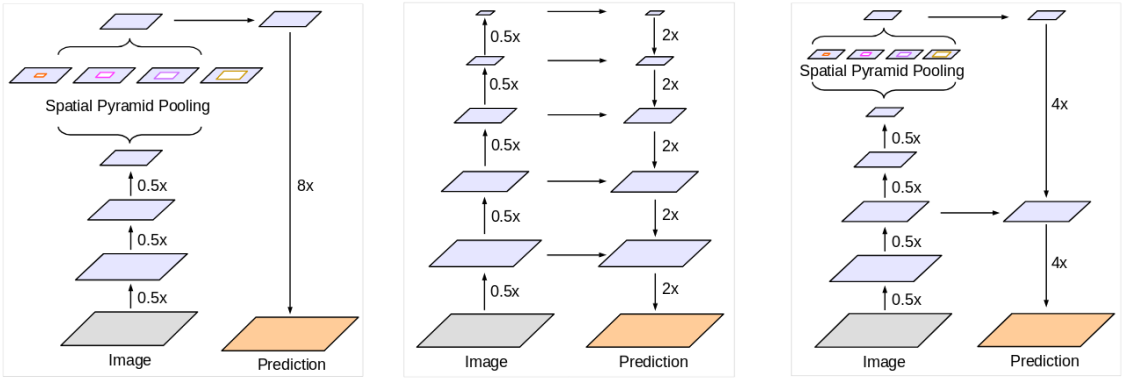


2.3. ábra. *A SegNet hálózat felépítése: enkóder-dekóder architektúra, a részek között átkötésekkel. (A kép az eredeti cikkből ([11]) származik.)*

konvolúciós rétegekkel nem lehet teljesen pontos szegmentációt előállítani; ezért az alsóbb szinteken levő Pooling rétegek kimenetét "előrevitték", hozzáadták a megfelelő dekonvolúciós rétegek kimenetéhez. Így az eredeti felbontás visszaállításakor a durva felbontású, magas szintű és a precízebb, alacsonyabb szintű információ is a hálózat rendelkezésére áll, ami pontosabb eredményeket jelent.

A SegNet névre keresztelt hálózat [11] egy enkóder-dekóder architektúrát használ, mindkét rész konvolúciós rétegekből áll. A hálózat felépítése látható a 2.3. ábrán. Az enkóder rész topológiája megegyezik a VGG-16 hálózat konvolúciós rétegeivel (az előrecsatolt rétegeket elhagyták a VGG hálózatból), a dekóder rész pedig ennek a "megfordítottjaként" képzelhető el, Pooling helyett Upsampling rétegekkel. Ez azért is előnyös, mert így az enkóder tanítását nem véletlenszerűen inicializált súlyokkal kell elkezdeni, hanem felhasználhatóak a már betanított VGG hálózatban levő súlyok. Az enkóder rész feladata a képből a magasabb szintű jellemzők kinyerése, míg a dekóder rész ezek alapján a pontos szegmentáció előállításáért felelős. Az enkóder és a dekóder rész megfelelő szintjei között itt is találhatóak átkötések, de míg az előző esetben a Pooling rétegek teljes kimenetét átadták ezeknél az előrecsatolásoknál, addig jelen esetben csak az indexeket továbbítják, vagyis azt, hogy a MaxPooling esetében a 2×2-es ablakokból a négy közül melyik érték lett kiválasztva. Az átadott indexeket a dekóderben az Upsampling rétegek használják fel: a nagyobb méretű kép előállításakor az indexnek megfelelő helyre bemásolják a kisebb képen levő értéket, a többi pozíciót pedig nullákkal töltik fel. Ezzel a módszerrel lényegesen kevesebb információt kell átadnunk, ami kisebb memóriaigényt jelent, miközben az elért eredmények pontossága nem csökken számottevően az előbbi esethez képest.

Kicsit másféleképpen közelítik meg a képszegmentáció problémáját [12]-ben. Az előbbi két hálózat a képek osztályozására használt hálózatokból kiindulva készített képszegmentációra képes modelleket; habár ennek megvoltak az előnyei, a képszegmentáció e cikk szerzői szerint mégis másféle architektúrájú hálózatokat igényel. A legnagyobb problémát a MaxPooling rétegek jelentik: míg a képek osztályozásánál ezek célja, hogy a képen kissé eltolva elhelyezkedő objektumokat is felismerje a hálózat, addig képszegmentációnál ez információveszteséget jelent az objektumok pontos helyéről. Ezért az itt bemutatott hálózatban egy újfajta elemet használnak, az úgynevezett nyújtott konvolúciót (*dilated convolution, atrous convolution*). Ennek az a lényege, hogy a konvolúciós szűrők nem közvetlenül egymás mel-



(a) *Spatial Pyramid Pooling* **(b)** *Általános Enkóder-Dekóder architektúra.* **(c)** *DeepLab Enkóder-Dekóder nyújtott konvolúcióval.*

2.4. ábra. *DeepLab architektúrák: nyújtott konvolúció használata Spatial Pyramid Pooling-hoz; általános enkóder-dekóder felépítésű hálózat architektúrája és a DeepLab hálózat dekóder résszel kiegészítve. (Forrás: [14].)*

letti pontok értékeit vizsgálják, hanem köztük kihagynak valamennyi pontot. (Például egy 3×3-as szűrő 1-es kihagyással egy 5×5-ös részből a (páratlan, páratlan) koordinátákon levő 9 db pontot fogja felhasználni.) Ez lehetővé teszi, hogy a szűrők nagyobb részét lássák a képnek anélkül, hogy a kép felbontását csökkentenénk, vagy a szükséges paraméterek számát növelnénk.

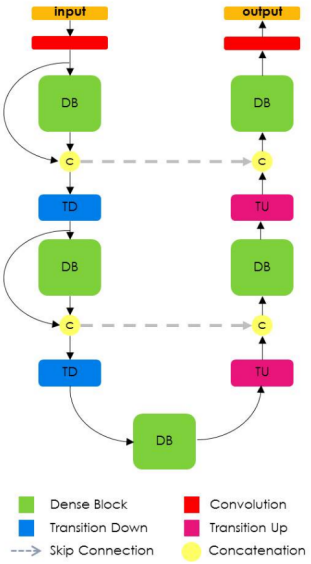
Ugyancsak nyújtott konvolúciót használnak a DeepLab esetében is ([13], [14]). Itt a kép feldolgozásakor egy-egy ponton több nyújtott konvolúciót használnak különböző kihagyásokkal, így a pont közvetlen és tágabb környezetét is tudják vizsgálni (*Atrous Spatial Pooling Pyramid, ASPP*). Ez a hálózat az eredeti képnél kisebb felbontásban készíti el a szegmentációt, amit először interpolálással nagyítottak fel, majd később kiegészítették egy dekóder hálózattal, aminek a feladata az eredeti felbontás visszaállítása. A DeepLab hálózat felépítése látható a 2.4. ábrán.

A RefineNet [15] hálózatban az eredeti képnek különböző felbontású változatait is felhasználják. Így a kisebb felbontású képekből képesek magasabb szintű információk meghatározására a képen elhelyezkedő objektumokról, amihez a nagyobb felbontású kép hozzávételével képesek az objektumok pontos körvonalát meghatározni.

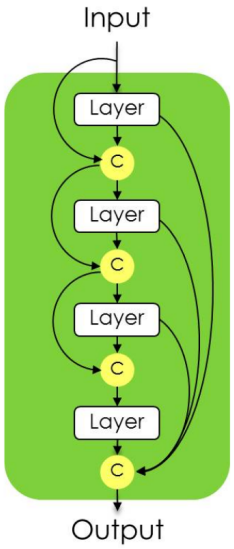
A képszegmentáció jelenlegi egyik legelterjedtebb célterülete az önvezető autók kameraképeinek elemzése, ahol kiemelten fontos, hogy a kép alapján a rajta levő objektumok pontos helye meghatározható legyen. Emellett más alkalmazási területei is vannak, például orvosi képfeldolgozásra is használhatóak ilyen hálózatok. Erre egy példa [16]. Az itt bemutatott, U-Net névre keresztelt hálózatot elektronmikroszkóp-felvételeken levő sejtek elkülönítésére tanították be. A hálózat felépítése hasonlít a korábbiakban bemutatottakhoz: konvolúciós és MaxPooling, majd pedig konvolúciós és Upsampling rétegek alkotják, valamint a két rész között átkötések találhatóak. Az orvosi képfeldolgozás során általában kisebb méretű adatbázisok állnak rendelkezésre, mint más képszegmentációs feladatoknál, ezért az U-Net tanításakor nagy hangsúlyt fordítottak a megfelelő adatdúsításra, amely segítségével a hálózatot kevés kép felhasználásával is pontosabban lehet tanítani.

A konvolúciós neurális hálózatoknál megfigyelhető, hogy az egyre jobb eredményeket általában egyre mélyebb, több rétegből álló hálózatokkal érik el. Ez látható a Tiramisu [17] hálózat esetében is, amelyet az eddigieknél részletesebben mutatok be, ugyanis ezt használtam fel a dolgozatomban elkészített rendszerben a képszegmentáció megvalósítására. Ez a hálózat a DenseNet [18] architektúrájához hasonló blokkokból épül fel. A DenseNet képfelismerő hálózat, de ahhoz, hogy jobban megértsük a Tiramisu hálózat felépítését, először vizsgáljuk meg ezt. A DenseNet abból a feltevésből indul ki, hogy a mélyebb konvolúciós hálózatok pontosabb eredményeket tudnak elérni. Azonban a túlságosan mély hálózatok tanításakor különféle problémák jelentkezhetnek. Ilyen az eltűnő gradiens (*vanishing gradient*), amikor a hiba-visszaterjesztés során a hálózat alsóbb rétegeibe már túl kicsi gradiensértékek jutnak el, így az itt levő súlyok nem tudnak módosulni, a hálózat ezen rétegei nem tanulnak meg semmit, vagy csak sokkal lassabban tanulnak. A másik probléma lehet a felrobbanó gradiens (*exploding gradient*), amikor az alsóbb rétegekbe túl nagy gradiensértékek jutnak el, így itt a súlyok túl nagy ugrásokkal változnak. Ezen problémák esetében az alsóbb rétegek, azon túl, hogy nem tanulnak, a véletlenszerű inicializálás miatt az eredeti képen felismerendő jellemzők helyett nagyjából véletlen értékeket adnak tovább a következő rétegeknek, amelyek így ha jól is működnek, sokkal rosszabb hatékonysággal tudják felismerni a képen levő objektumot. Ezen problémák kezelésére használható sok esetben a ReLU aktivációs függvény: $max(0, x)$, illetve ennek további változatai, amik más nemlineáris függvényeknél kedvezőbb tulajdonságokkal rendelkeznek a hiba-visszaterjesztés szempontjából. Nagyon mély hálózatoknál azonban ez is kevés lehet, ezért a DenseNet architektúrában bevezették az úgynevezett *dense block*-ok használatát. Ezekben a konvolúciós rétegek bemenetükön az összes megelőző réteg kimenetét megkapják, és minden réteg közvetlenül továbbítja a kimenetét a blokk végére. Így a hiba visszaterjesztésekor minden réteg közvetlenül is megkapja a blokk végén meglévő gradienseket, ezáltal elkerülhetőek a fentebb leírt problémák.

A Tiramisu [17] hálózat úgy képzelhető el, mint a DenseNet hálózat átalakítása képszegmentációs célra. A Tiramisu hálózat is *dense block*-okból épül fel, a hálózat architektúrája és a blokkok struktúrája a 2.5. ábrán látható. A *dense block*-ok felépítése hasonló a DenseNet-ben használtakhoz. A blokkokban konvolúciós rétegek találhatóak, kiegészítve BatchNormalization, ReLU és Dropout rétegekkel a regularizáció és a nemlinearitás érdekében. A hálózat első felében (*downsampling path*) a blokkok között leskálázások történnek (*Transition Down*), melyek 1×1-es konvolúciót és 2×2-es MaxPooling-ot tartalmaznak, kiegészítve a blokkokban is megtalálható regularizációs és aktivációs elemekkel. A blokkok kimenetéhez hozzá vannak csatolva az adott blokk bemenetén kapott értékek is, így egy-egy blokk az őt közvetlenül megelőző által előállított jellemzők mellett a korábbi blokkok által megtanult jellemzőket is megkapja. Ezáltal a jellemzők száma a hálózatban előrehaladva folyamatosan növekszik. A Tiramisu hálózat második felét (*upsampling path*) is *dense block*-ok alkotják, melyek között felskálázó (*Transition Up*) elemek találhatóak. A felskálázást úgynevezett *Transposed Convolution* rétegekkel valósítják meg. A hálózat két része között átkötések találhatóak, amik segítségével a magasabb felbontású képekből kinyert, alacsonyabb szintű információkat is felhasználja a hálózat a felskálázás során. Itt



(a) *A Tiramisu hálózat struktúrája. A hálózat úgynevezett dense block-okból áll. A blokkok között le- és felskálázó komponensek találhatók. A hálózat két ága között átkötések találhatóak, melyeket szaggatott nyíl jelez. A kis körökben a beérkező adatok összefűzése (konkatenáció) történik. Megfigyelhető, hogy a dense block-ok eltérően vannak összekötve a le- és felskálázó részben: a hálózat elején a blokkok kimenetéhez mindig hozzáfűzik a bemenetét, így az előállított jellemzők száma itt folyamatosan növekszik.*



(b) *Dense block felépítése. A blokkban az egyes szinteket a konvolúciós rétegek mellett Batch-Normalization, ReLU aktivációs réteg és Dropout alkotják. A szintek az eredeti bemenet és a felettük lévő szint által előállított jellemzők alapján készítik el a saját jellemzőiket, melyeket átadnak a következő szintnek, valamint közvetlenül a kimenetre is.*

2.5. ábra. *Tiramisu hálózat felépítése és a dense block. (Forrás: [17].)*

már a blokkok között nincsenek előrecsatolások, ugyanis attól nagymértékben megnőne a hálózat paramétereinek száma, ami lényegesen növelné a hálózat tanításához szükséges időt (a felbontás az első részben folyamatosan csökkent, ezért lehetett ott előrecsatolásokat használni; itt viszont a felskálázások miatt a felbontás folyamatosan növekszik). A teljes Tiramisu hálózatban összesen 103 darab konvolúciós réteg található, öt darab *Transition Up* és öt *Transition Down* köti össze a blokkokat a le- és felskálázás során. A sok réteg ellenére, a hálózat speciális szerkezete miatt, a paramétereinek száma más hálózatokénál alacsonyabb.

A képszegmentáció megvalósításakor felhasználási területtől függően lehet szempont a hálózat futásához szükséges idő is. Ha például önvezető autók kameraképeinek elemzését szeretnénk mély neurális hálózatokkal megvalósítani, akkor elengedhetetlen a sokszor valós időnél is gyorsabb képfeldolgozás. Ez megvalósítható a [19]-ban bemutatott módszerrel. Az itt leírt ICNet nevű hálózat nagy felbontású (1024×2048-as) képeken valós időben (30 kép másodpercenként) képes a szegmentáció megvalósítására. Ehhez a hálózat az eredeti, nagy felbontású kép mellett annak lekicsinyített változatait is felhasználja. Először ezek alapján elkészít egy hozzávetőleges szegmentációs képet, amit azután az eredeti felbontású kép segítségével pontosít. Ezzel a módszerrel a pontosság jelentős romlása nélkül sikerült

felgyorsítani a képszegmentáció folyamatát.

2.2. Neurális művészet

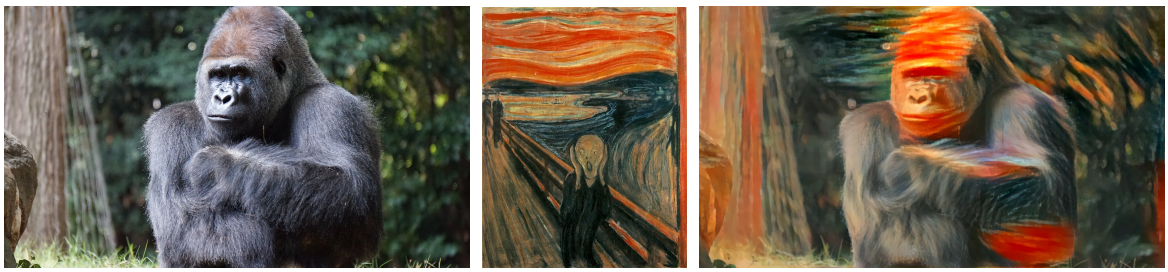
A mély neurális hálózatok egyik érdekes felhasználási területe lett az utóbbi években az úgynevezett neurális művészet (*neural art*). Ez azt jelenti, hogy mély neurális hálózatok felhasználásával művészi hatású produktumot (ez általában kép, de lehet például zene is) állítanak elő. A feladat nehézségét az adja, hogy nehéz pontosan, matematikailag megfogalmazni, hogy mitől lesz egy kép "művészies", valamint – a hagyományos, felügyelt tanítással ellentétben – olyan kimenetet kell előállítani, amelyhez hasonló korábban még nem létezett, így itt komplexebb megoldásokra van szükség, nem elég összegyűjteni egy nagy, felcímkézett adatbázist, amelyben levő összefüggéseket a hálózat képes megtanulni.

Az egyik legismertebb megvalósítása a neurális művészetnek a stílustranszformáció (*style transfer*), ami során egy objektumot és egy stílust tartalmazó képből előállítunk egy olyan képet, ami az adott objektumot tartalmazza az adott stílusban (például stílusként felhasználhatjuk ismert festők műveit, így bármilyen fényképből készíthetünk olyan képet, ami olyan hatást kelt, mint ha egy híres festő készítette volna). A stílustranszformáció mellett szintén közismert a Google DeepDream (mély álom) nevű alkalmazása, amivel álomszerű, kicsit hallucinogén megjelenésűvé alakíthatunk egy képet. Ezeken kívül számtalan további felhasználása van a neurális művészetnek, melyek közül néhányat bemutatok a következő alfejezetekben.

2.2.1. Stílustranszformáció

A stílustranszformáció (*style transfer*) [20] lényege, hogy egy művészi kép stílusát átvigyük egy másik képre. Ez látható a 2.6. ábrán. A kép generálásához két képre van szükség: az egyik a tartalmat, a másik pedig a stílusjegyeket szolgáltatja.

A generált kép előállításához definiálunk egy hibafüggvényt, amely leírja, hogy a generált kép mennyire hasonlít a megadott tartalomra és stílusra. A hibafüggvény felhasználja a kiindulási két képet és a generált képet, amik alapján egy értékkel jellemzi az utóbbit: minél kisebb az érték, annál jobb a generált kép. A cél az, hogy ezt a hibaérték minél alacsonyabb legyen. Ehhez a neurális hálózatoknál szokásos hibavisszaterjesztés (*backpropagation*) algoritmusát a megszokottól eltérően használjuk: a változtatható paraméterek a generált kép képpontjai lesznek, a hiba kiszámításakor felhasznált neurális hálózatok súlyai pedig rög-



2.6. ábra. *Stílustranszformáció során egy kiindulási képet egy adott témához hasonlóvá alakítunk át. (Festmény: Edvard Munch - A sikoly.)*

zített értékűek (a neurális hálózatok tanításakor ez éppen fordítva szokott lenni: a hálózat súlyait módosítjuk, de a bemeneti képet nem változtatjuk meg). A hibafüggvény felhasználásával a neurális hálózatok tanításakor szokásos optimizáló algoritmusok segítségével több lépésen keresztül egyre jobb generált képet állíthatunk elő. Kiindulásként a generált kép pixeleit véletlenszerű értékeknek választjuk meg.

A hibafüggvény két részből áll: tartalmi hibából (*content loss*) és stílushibából (*style loss*). Ezek súlyozott összege adja a végső hibát. Mindkét hibaérték kiszámolásához felhasználjuk az ImageNet képek felismerésére betanított VGG hálózatot.

A tartalmi hiba azt mutatja meg, hogy a generált képen levő objektum mennyire hasonlít az eredeti képen levőhöz. Ezt az értéket úgy határozzuk meg, hogy az első bemeneti képet és a generált képet végigfuttatjuk a VGG hálózaton, és megvizsgáljuk a hálózat valamely konvolúciós rétegének (vagy rétegeinek) aktivációit: ha a két kép hasonló, akkor a hálózatban számolt értékek is hasonlóak lesznek, hiszen a hálózat ekkor ugyanazt a dolgot ismeri fel a két képen. A hasonlóságot a két kép esetében mért aktivációs értékek között átlagos négyzetes hibával (*mean squared error*) számoljuk.

A stílushiba azt mutatja meg, hogy a generált kép stílusa mennyire hasonló az elvárt stílust megadó kép stílusához. Ennek kiszámolásához is a VGG hálózat valamely belső rétegének aktivációit használjuk, de a számolás némileg bonyolultabb az előzőnél. A konvolúciós réteg kimenetét csatornánként "kilapítjuk" vektorokká, majd kiszámoljuk e vektorok páronként vett skaláris szorzatát, és felírjuk ezeket egy mátrixba (ez a Gram mátrix). Az ebben lévő értékek a különböző szűrők közötti korrelációt mutatják meg. A bemeneti kép és a generált kép stílusa közötti hasonlóságot a Gram mátrixuk közötti átlagos négyzetes hiba írja le, tehát minél hasonlóbb a két kép stílusa, annál inkább hasonlóak lesznek a belőlük előállított Gram mátrixok is.

2.2.2. DeepDream

A stílustranszformáció mellett a másik, széles körben ismert iránya a neurális művészetnek a DeepDream [21], amellyel az elkészített képek olyan hatásúak lesznek, mintha a számítógép "álmában" rajzolta volna őket. A kép bizonyos részei kissé eltorzulnak, és itt különböző formák (például állatok) homályos képe jelenik meg.

A DeepDream eredeti célja a neurális hálózatok vizualizációja volt: azt vizsgálták, hogy mi alapján ismeri fel az egyes objektumokat, és egyáltalán milyen jellemzőket tanulnak meg a hálózat különböző rétegei. Ehhez a hálózat súlyait rögzítették, és a bemeneti képet úgy változtatták, hogy a hálózat egy adott kimenete minél nagyobb legyen, vagyis olyan képet akartak generálni, melyet a hálózat minél nagyobb magabiztossággal mond egy adott kategóriába tartozónak. Ehhez hasonlóan valamelyik belső réteg egyik kimenetét is maximalizálhatjuk, ekkor azt láthatjuk, hogy az a réteg milyen formákat képes felismerni.

A mély álom-szerű képek generálásakor is ez történik: kiválasztunk egy kiindulási képet, és ennek a képpontjait módosítjuk úgy, hogy a hálózat valamelyik kimenete minél nagyobb legyen. Ez látható a 2.7. ábrán is: a képen bizonyos állatok körvonalai figyelhetők meg. A megjelenő formák komplexitása attól függ, hogy a hálózat melyik rétegének kimenetén levő

(a) *Kiindulási kép.*

(b) *DeepDream effektussal módosított kép.*

2.7. ábra. *DeepDream segítségével olyan hatást érhetünk el, mintha a számítógép "álmában" készített volna a képet.*

értéket maximalizáltuk: az első néhány réteg esetében csak egyszerűbb vonalak, görbék, hullámok jelennek meg, míg a későbbi rétegek választásakor bonyolult alakzatok tűnnek elő a képen.

2.2.3. További alkalmazások

Az előbbi két példán kívül még számos további cikk foglalkozik neurális művészettel. [22]-ben generatív neurális hálózatok (*GAN, Generative Adversarial Network*) segítségével készítenek kreatív művészi képeket. A GAN-ok ([23]) két neurális hálózatból állnak: egy generátorból és egy diszkriminátorból. A generátor feladata, hogy megadott mintához hasonló

képeket állítson elő, a diszkriminátor feladata pedig a valós és a generált képek megkülönböztetése. A két hálózatot egyszerre tanítják, addig, amíg a generatív hálózat képes lesz az eredetiektől megkülönböztethetetlen képek előállítására; tanítás során a diszkriminátor segíti a generátort abban, hogy minél jobb képeket produkáljon. A művészi képek generálásához a GAN kissé módosított változatát használták, amely így képes kreatívabb képek készítésére.

A neurális művészet népszerűségét az is mutatja, hogy a 2017-es NIPS (*Neural Information Processing Systems*) konferencián – ami az egyik legnevesebb konferencia gépi tanulás témakörében – egynapos workshop-ot rendeztek a gépi tanulás kreatív, művészi felhasználási lehetőségeiről[2]. Ezen a generatív modelleket kutató tudósok mellett olyan művészek is előadtak, akik saját alkotásaikhoz használják fel a mély neurális hálózatok nyújtotta lehetőségeket. Művészi képek generálása mellett olyan témák is előkerültek itt, mint például zenék generálása, automatikus szövegírás, divatos ruhák tervezése vagy éppen új parfümillatok előállítása. Habár ezek az eredmények korántsem annyira professzionálisak, mint a korábbi fejezetben ismertetettek, sokszínűségük jól mutatja a mély neurális hálózatok művészi irányú felhasználásában rejlő lehetőségeket.

[2] https://nips2017creativity.github.io/

3. fejezet

Rendszerterv

Dolgozatomban egy olyan rendszert tervezek és valósítok meg, amivel lehetséges kettős expozíciós képek generálása szinte bármilyen kiindulási állatalakot tartalmazó képből és tájképből.

A kettős expozíciós kép előállításának első lépése, hogy az állatfigurát tartalmazó képről körbe kell vágni az állatot, el kell távolítani az eredeti hátterét. Ezt a feladatot egy képszegmentációra képes neurális hálózattal oldottam meg, ami képes elkülöníteni a képeken levő állatokat a hátterüktől. A hálózat tanításához a publikusan elérhető Microsoft COCO: Common Objects in Context [9] adatbázist használtam fel. Az adatbázisból kiválogattam az állatokat tartalmazó képeket, és a hozzájuk megadott címkék alapján elkészítettem a kép–maszk párokat, ahol a maszkok jelzik az adott képeken az állatok elhelyezkedését. Ezek felhasználásával betanítottam egy Tiramisu [17] hálózatot, ami így képessé vált arra, hogy felismerje és körbevágja a képeken levő állatokat.

A kettős expozíciós képeket úgy szeretném elkészíteni, hogy az állatnak a feje megmaradjon, és csak a teste legyen összemosva a tájképpel. Ahhoz, hogy ezt meg lehessen csinálni, meg kell határozni, hogy hol található a képen az állat feje. Ezt rábízhatjuk a program felhasználójára is, hogy ő döntse el, merre néz az állat; de még jobb, ha ezt is automatikusan el tudjuk dönteni az állatfigurát tartalmazó kép alapján. Ezért elkészítettem egy másik mély neurális hálózatot is, amely feladata eldönteni, hogy az állatnak a testéhez képest milyen irányban helyezkedik el a feje a képen. E hálózat alapját az ImageNet versenyre készített VGG-16 adta, amelynek a már betanult súlyaiból kiindulva jelen feladatot sokkal gyorsabban megtanulta a hálózat, mintha véletlenszerűen inicializált súlyokkal kezdtük volna a tanítást. Ezt a hálózatot is a COCO adatbázisban levő állatokat tartalmazó képekkel tanítottam, melyeket három csoportra osztottam aszerint, hogy az állat feje balra, jobbra vagy felfelé néz. A tanítás végére ez a hálózat képessé vált arra, hogy meghatározza egy képen levő állat fejének irányát.

A két neurális hálózat elkészültével már minden adott, hogy elkészíthessük a kettős expozíciós képeket. A képek készítéséhez szükség van egy állatalakot tartalmazó képre és egy tájképre. Az állatot tartalmazó képből az első hálózat segítségével kivágjuk az állatot, és eltávolítjuk az eredeti hátteret. Ezután a második hálózat segítségével meghatározzuk, hogy milyen irányba néz az állat. Ezek felhasználásával egy matematikai algoritmus segítségével

elkészíthetjük a két kép összekeverésével kapott kettős expozíciós képet, ügyelve arra, hogy az állat feje megmaradjon, és csak a teste legyen elhalványítva és összemosva a tájképpel. A kép elkészítésének folyamata látható a 3.1. ábrán.

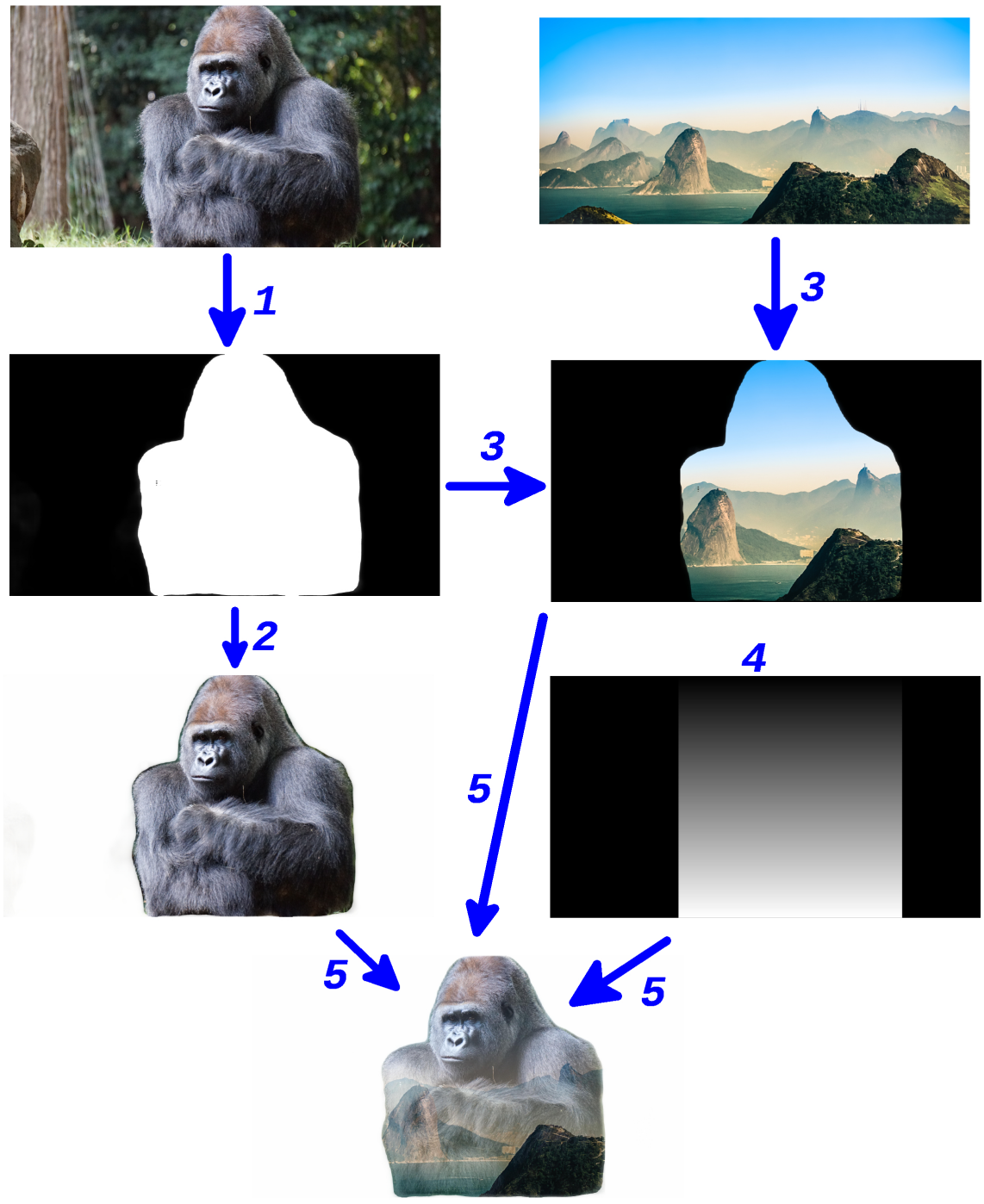
A rendszer a kettős expozíciós képek készítése mellett lehetőséget biztosít arra is, hogy egy állatos képen körbevágjuk a rajta levő állatot. Ehhez a fent leírt folyamatnak csak az első felét hajtjuk végre, a képszegmentációra képes hálózat segítségével meghatározzuk a képen az állat körvonalát, majd körbevágjuk azt. Ez a felhasználási mód akkor lehet hasznos, ha állatos képekkel szeretnénk dolgozni, de nem éppen ilyen kettős expozíciós képeket akarunk készíteni, hanem valami mást, például egy másik képre bevágni őket. Ekkor lehetőségünk van az állatok automatikus körbevágására és az eredeti hátterük eltávolítására.

A neurális hálózatok működése nem mindig tökéletes, néha hibáznak a maszk vagy az állatfej irányának pontos meghatározásában. Ha a fej irányát eltévesztette a hálózat, akkor lehetőségünk van kézzel megadni azt, így továbbra is elkészíthetőek a kettős expozíciós képek. Néha a szegmentációs hálózat által generált maszk sem tökéletes, előfordulhatnak benne kisebb-nagyobb pontatlanságok. Bizonyos jellegű hibák egyszerűen javíthatók szabály alapú módszerrel. Előfordulhat, hogy az állatok helyét jól meghatározza a hálózat, de bizonyos részeken nem annyira volt biztos a dolgában, ezért halvány lesz a kimenet. Olyan is lehetséges, hogy az állat körvonalának meghatározása mellett egyéb területekre is kis valószínűséggel azt hitte, hogy állathoz tartoznak. Ilyen jellegű hibák javíthatók a generált maszkban levő értékek szétválasztásával: bizonyos szint alatti értékeket le-, vagy az afölöttieket felkerítjük a szebb kép előállítása érdekében.

Az egyszerű használhatóság érdekében a programhoz elkészítettem egy webes felületet, ahol könnyen lehet kettős expozíciós képeket generálni. A felületen kiválaszthatjuk az állatos képet és a tájképet, amiből a rendszer automatikusan elkészíti a kettős expozíciós képet. Ha nem vagyunk elégedettek az eredménnyel, akkor az előző bekezdésben leírt korrekciókat is elvégezhetjük. Emellett a webes felületen lehetőség van arra is, hogy csak körbevágjuk az állatfigurát tartalmazó képet, de ne készítsünk belőle kettős expozíciós képet.

A kettős expozíciós képek előállítását végző rendszer minden részét (a webes felület felhasználóoldali részének kivételével) Python (3.5.2) nyelven készítettem el. A neurális hálózatok elkészítéséhez és tanításához a Keras (2.0.8) könyvtárat használtam, ami mögött TensorFlow (1.3.0) backend futott. Ezek mellett felhasználtam még a NumPy, Pillow, Matplotlib és Pickle könyvtárakat (ezek rendre számítási műveletekhez, képfeldolgozáshoz, diagramok és ábrák készítéséhez, valamint Python objektumok szerializációjához használhatóak). A webes felület Flask alapon működik.

A neurális hálózatok tanítása rendkívül erőforrásigényes, ehhez a tanszéken rendelkezésre állt egy GPU-s szerverkörnyezet, ahol a tanítások hatékonyan futtathatók. A szerver kellő erőforrásokkal rendelkezik nagyobb méretű hálózatok tanítására is (8 x Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz, 32GB RAM, GeForce GTX Titan X 12GB grafikus kártya), valamint már fel voltak rá telepítve az általam használt szoftverkönyvtárak is.



3.1. ábra. *A kettős expozíciós kép előállításának lépései:* ***1.*** *az állathoz tartozó maszk elkészítése az első neurális hálózattal;* ***2.*** *a maszk alapján az állat körbevágása;* ***3.*** *a maszk alapján a tájképből állat-alakú rész kivágása;* ***4.*** *az állatfej irányának meghatározása a második neurális hálózattal (itt: felfelé);* ***5.*** *körbevágott állat és tájkép felhasználásával kettős expozíciós kép előállítása, ahol a két kép közötti áttűnést a fej irányának figyelembevételével valósítjuk meg.*

4. fejezet

Képszegmentáció megvalósítása

A kettős expozíciós kép elkészítéséhez első és legfontosabb lépés az állatalakot tartalmazó képen az állat körvonalának pontos felismerése és ez alapján az állat körbevágása. Ez egy képszegmentációs feladat: a képen el kell különíteni az állatot a háttértől. A képszegmentáció megoldására egy neurális hálózatot tanítottam be, amely sok, mintául szolgáló kép alapján megtanulta felismerni, hogy egy képen mely részek tartoznak az állathoz és melyek a háttérhez. A hálózatot a Microsoft COCO: Common Objects in Context adatbázisban lévő képek segítségével tanítottam.

4.1. Adatbázis

A neurális hálózatok felügyelt tanításához nagy méretű, lehetőleg címkézett adathalmazokra van szükség. Ez jelen esetben azt jelenti, hogy olyan állatokat tartalmazó képekre van szükség, amelyekhez meg van adva, hogy a képen pontosan hol helyezkedik el az állat. Több olyan adatbázis is létezik, amelyekben a képekhez tartoznak szegmentációs jelölések (például: COCO [9], PASCAL Visual Object Classes [24], Cityscapes [25]). Ezek általában utcai felvételeket tartalmaznak, rajta megjelölve az utak, épületek, autók, gyalogosok stb., amelyek például egy önvezető autó fejlesztésekor lehetnek hasznosak. Ezen adatbázisok közül a Microsoft COCO adatbázist választottam, ugyanis a benne lévő képek alapján ez a legalkalmasabb a jelenlegi feladat megoldásához. Az adatbázisban összesen több mint kétszázezer felcímkézett kép található, amelyeken összesen 80 kategóriába tartozó objektumok vannak bejelölve. A kategóriák között megtalálhatóak például az utcai képekre jellemző tárgyak (különféle járművek, táblák), beltéri képeken előforduló objektumok (többféle bútor, háztartási gép, ételek és egyéb tárgyak), valamint van tíz darab állatos kategória is (madár, kutya, macska, ló, bárány, tehén, elefánt, medve, zebra, zsiráf). A képekhez meg van adva a rajta levő objektumok pontos körvonala, ami alapján a képszegmentációt meg lehet valósítani.

A mostani feladat megoldásához csak az állatokat tartalmazó képekre van szükség, ezért az adathalmazból a képekhez tartozó jelölések alapján kiválogattam azokat a képeket, amelyeken vannak állatok. Amelyik képeken túl kicsi méretűek voltak az állatok, azokat kihagytam, ugyanis a várható felhasználás során a kép nagy részében az állat lesz, ezért azok

a képek, melyek esetében például csak az egyik sarokban található egy kis állat, valószínűleg nem tudnák javítani a hálózat pontosságát. Az adatbázisban levő képek különféle méretűek voltak, ezért – a hálózat egyszerűbb tanítása érdekében – minden képet átméreteztem egységes, 224×224-es méretűre.

A képeket szétosztottam tanító, validációs és tesztelő adathalmazokra (közelítőleg 70% – 20% – 10% arányban). A tanító adathalmazban levő képekkel a modellt tanítottam, a validációs képekkel tanítás közben mértem a modell pontosságát, a tesztelő képekkel pedig a tanítás végén mértem meg a hálózat sikerességét. Az egyes adathalmazokba eső képek száma a 4.1. táblázatban látható. Néhány kép a tanító adatbázisból a 4.1. ábrán látható.

4.1. táblázat. *Az adathalmazban levő képek darabszáma.*

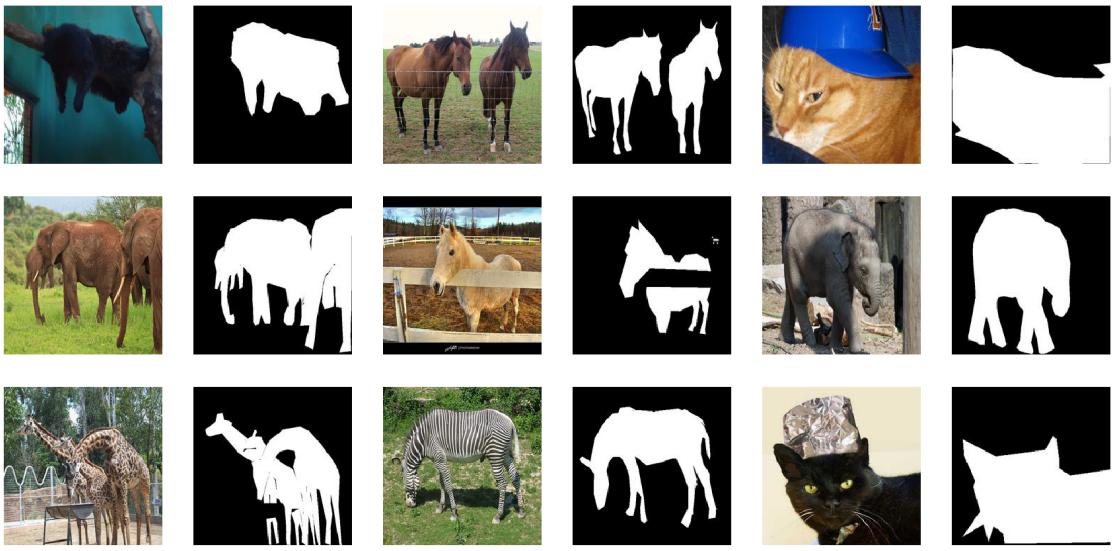
Tanító adathalmaz	11268
Validációs adathalmaz	3406
Teszt adathalmaz	1642

A képeket a tanítás előtt az alábbi képlet szerint normalizáltam:

$$x' = \frac{x - avg(X)}{std(X)}, \tag{4.1}$$

ahol $avg(X)$ és $std(X)$ a tanító adathalmazban levő képek alapján csatornánként számolt átlag és szórásérték.

A tanításhoz használtam adatdúsítást (*data augmentation*) is. Ezzel a technikával azt lehet elérni, hogy kisebb méretű adathalmazzal is jobb eredményt érjen el a neurális hálózat. Lényege, hogy tanítás közben az adatokat kismértékben változtatjuk (ez képek esetében jelenthet például forgatást, tükrözést, zoom változtatást, nyírást vagy egyéb, ezekhez hasonló transzformációt). Így a modell ugyanazt a képet több változatban is látja, ezáltal jobban megtanul általánosítani, így pontosabb lesz a felismerés a modell által korábban nem látott



4.1. ábra. *Képek a tanító adatbázisból a hozzájuk tartozó maszkkal.*

képeken. Az adatdúsítás jelentőségéről [26]-ban találhatóak további részletek. Én a felsorolt transzformációk közül csak a függőleges tengely menti tükrözést alkalmaztam.

4.2. Neurális hálózat tanítása

A képeken található állatok felismeréséhez a Tiramisu [17] nevű hálózatot használtam[1]. A hálózat felépítésének ismertetése megtalálható a 2.1.3 fejezetben. Az eredeti cikkben a hálózatot városi képek szegmentációjára használták. A jelenlegi feladat ennél egyszerűbb, ugyanis csak két osztályt kell megkülönböztetni: az állatot és a hátteret. Ehhez a hálózatban csak az utolsó konvolúciós réteg paramétereit kell megváltoztatni, ami meghatározza, hogy hány osztályba sorolja be a kép képpontjait; a hálózat többi része megegyezik az eredeti Tiramisuéval. A teljes hálózatban összesen 103 darab konvolúciós réteg található, és kicsivel több, mint 22 millió paraméterből áll. Az eredeti hálózathoz képest a konvolúciós rétegekben található szűrőket 3×3-asról 5×5-ösre cseréltem, így a hálózat nagyobb pontossággal ismerte fel a képeken levő állatokat (természetesen így a hálózat paramétereinek száma is növekedett).

A modell tanításához a képeket az előzőekben leírtak szerint dolgoztam fel, a képeken alkalmaztam adatdúsítást (*data augmentation*). A modell bemenete egy 224×224 felbontású, 3 csatornás kép, kimenete pedig 224×224 becslés: a kép minden pixelére meghatározza, hogy milyen valószínűséggel esik az egyes kategóriákba (állat és háttér).

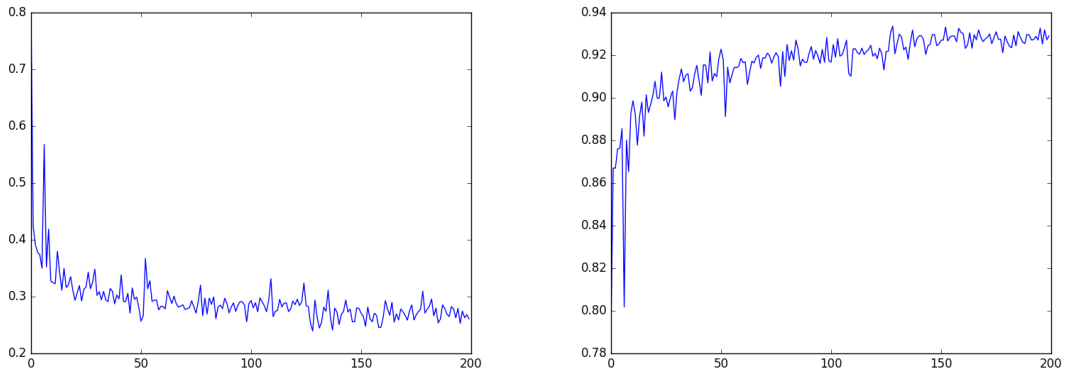
A modell tanítását véletlenszerűen inicializált súlyokkal kezdtem. A tanításhoz 4-es méretű *batch size*-t használtam (azaz tanítás közben a hálózat súlyait minden lépésben négy kép alapján számolt értékek átlaga szerint módosítja), a tanításhoz használt optimizáló algoritmus RMSprop [27] volt $1 \cdot 10^{-4}$-es kiinduló tanulási ráta (*learning rate*) értékkel. A tanítás generátorral történt (túl sok kép van ahhoz, hogy egyszerre betöltsük mindet a memóriába), egy-egy epochban 300 batch-nyi képpel tanítottam a hálózatot, és 250 batch-nyi képpel validáltam (ezek a képek rendre a tanító és a validációs adathalmazból lettek kiválasztva, minden lépésben véletlenszerűen). Az epoch-ok optimális számának meghatározásához *early stopping*-ot használtam (ez azt jelenti, hogy tanítás során minden epoch után megmérjük a validációs hibát, és amikor az eddigieknél alacsonyabb értéket kapunk, akkor elmentjük a modellt; ha pedig már egy ideje nem csökken tovább ez az érték, akkor megállítjuk a tanítást, és visszatöltjük azt az állapotot, amikor a legkisebb volt a validációs hiba). A tanítás során használtam Dropout-ot 0,2-es értékkel (ez azt jelenti, hogy tanításkor minden lépésben bizonyos valószínűséggel figyelmen kívül hagyunk egy-egy súlyt, így többféle információból tanulja meg felismerni a modell a képeken levő állatokat), illetve l2 regularizációt 10^{-4}-es paraméterrel.

A modellt két fázisban tanítottam. Először véletlenszerűen inicializált súlyokkal kiindulva (HeUniform [28] inicializációval), amíg az *early stopping* alapján meg nem állt; majd második lépésben ebből a már betanított modellből kiindulva kezdtem újra a tanítást, az előbbinél kisebb *learning rate* értékkel, így a hálózat által elért pontosság tovább tudott

[1] A Tiramisu hálózat Keras alapú implementációja megtalálható GitHub-on a https://github.com/fastai/courses/blob/master/deeplearning2/tiramisu-keras.ipynb oldalon (elérés időpontja: 2018. 10. 12.)

növekedni.

A modell teljes tanítása nagyságrendileg 300 epoch-ig tartott, ami a GPU-s szerverkörnyezetben egy napot vett igénybe. A tanítás közben epoch-onként mért validációs hiba és pontosság alakulása a 4.2. ábrán látható.



(a) *Validációs hiba a tanítás második fázisában epoch-onként.* **(b)** *Validációs pontosság a tanítás második fázisában epoch-onként.*

4.2. ábra. *A tanítás második fázisában mért validációs hiba és pontosság.*

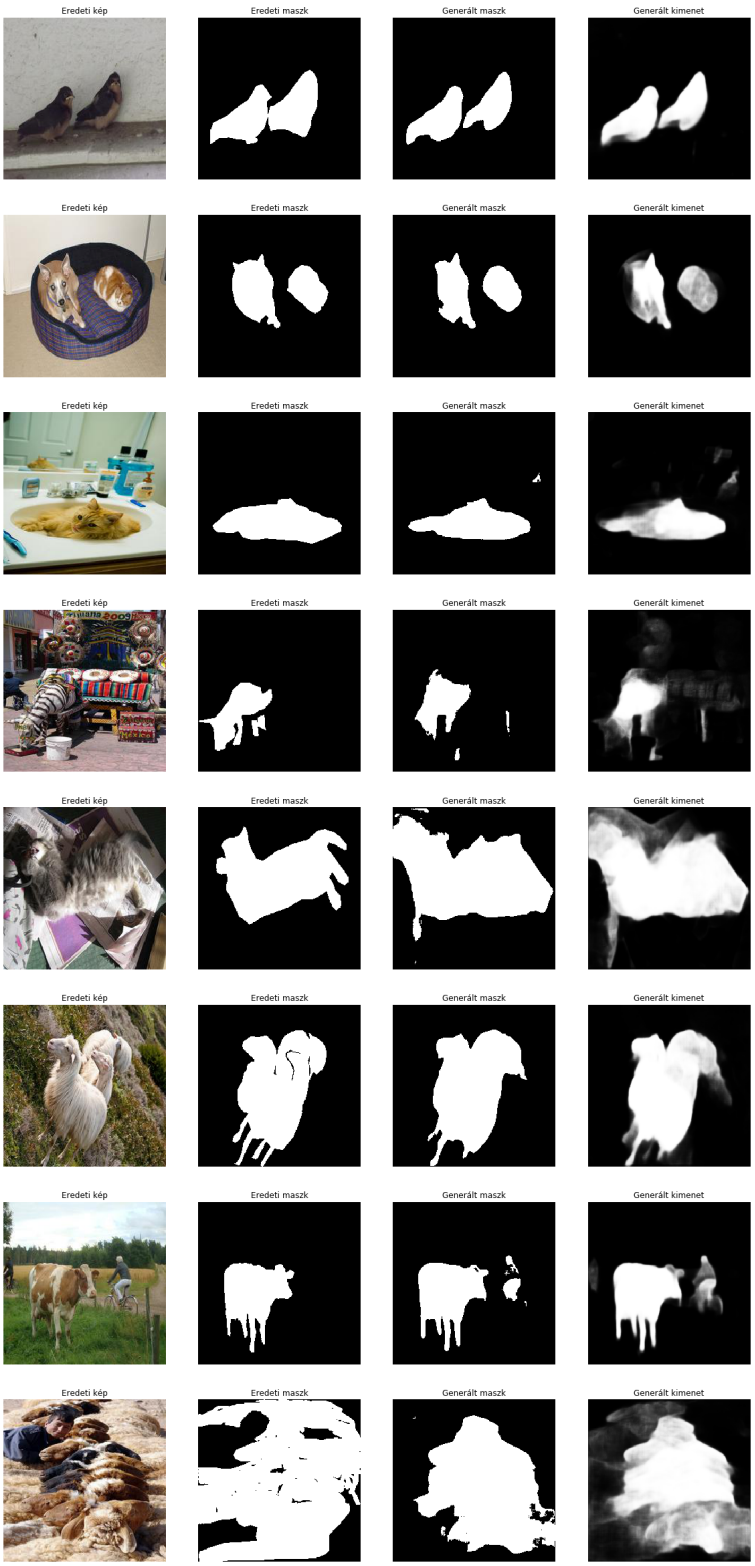
4.3. Kiértékelés

A tanítás végére a hálózat a tesztképeken 93%-os pontosságot ért el – vagyis a képen levő pixelek ekkora hányadát tudta helyesen bekategorizálni.

A 4.3. ábrán látható a modell által generált maszk néhány, a tesztadatbázisból véletlenszerűen választott képre. A képek mellett látható az eredeti maszk, a modell által generált maszk és a modell kimenete. A képeken is látható, hogy a hálózat az esetek nagy részében helyesen megtalálja az állatok helyét a képen, de néha előfordulnak kisebb-nagyobb tévedések: az utolsó előtti képen például az embert is állatnak kategorizálta be, az utolsó, sok állatot tartalmazó képen pedig pontatlan lett a generált maszk.

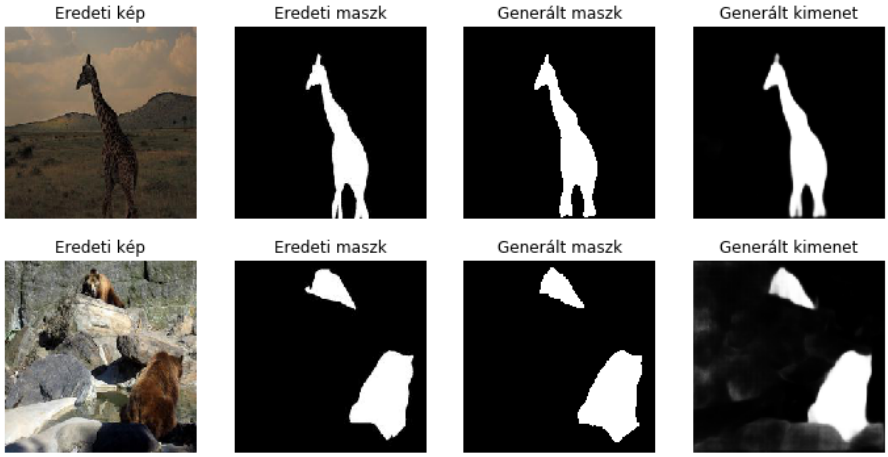
A 4.4. ábrán látható kettő-kettő a modell által készített legpontosabb és legrosszabb predikciók közül. A rosszul elkészített maszkok esetében jellemzően összetett a kép (sok objektum van rajta), takarásban van az állat egy része, vagy beleolvad a környezetébe.

A tesztadatbázisban levő képek mellett olyan képekre is megvizsgáltam a modell teljesítményét, amelyek az adatbázisban lévőknél "jobb" képek olyan értelemben, hogy nagyobb felbontású, jobban beállított képek (azaz jellemzően jól láthatóak rajta az állatok, nincsenek félig-meddig takarásban). Ezek láthatóak a 4.5. ábrán. Ezek a képek változatosabbak olyan szempontból, hogy többféle olyan állat is van köztük, amik nem szerepelnek az adatbázisban, tehát amiket nem látott a modell tanulás közben (pl. mosómedve, szurikáta). Ezek a képek azért is relevánsak, mert ezekből készítettem kettős expozíciós képeket, és várhatóan a leendő felhasználás során is ehhez hasonló képeket fognak használni (olyan képekből, amiken az állatok félig takarásban vannak, vagy nem jól láthatóak, nem lehet jól kinéző kettős expozíciós képeket készíteni).

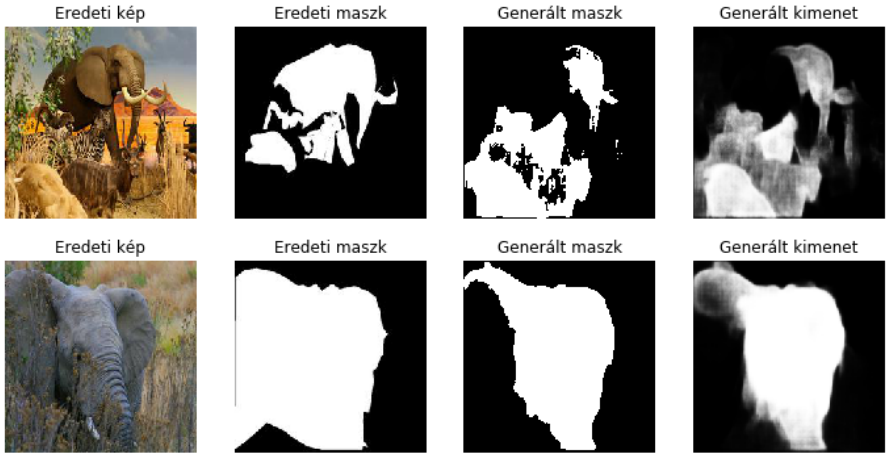


4.3. ábra. *A modell által készített predikciók a tesztadatbázisban levő képekre. Az oszlopokban rendre az eredeti kép, az eredeti maszk, a modell által generált maszk és a modell pontos kimenete látható. A modell által generált maszkban a* $0,5$*-nél magasabb valószínűséggel állathoz tartozónak mondott képpontok szerepelnek.*

A tanító adatbázisban összesen tízféle állatról szerepeltek képek, ami némileg korlátozza azt, hogy milyen állatokat képes felismerni a hálózat a képeken. Habár sok olyan állat körvonalát is pontosan meg tudja határozni, amelyek nem szerepeltek a tanításhoz használt képek között – például szurikáta, mosómedve, delfin –, vannak olyanok is, melyeket jellemzően nem tud behatárolni a képeken. Így például a halakat és a rovarokat nem ismerte fel az általam kipróbált tesztképeken. Ennek az az oka, hogy a halak és rovarok nem hasonlítanak egyik állathoz sem azok közül, amelyekkel tanítottam a hálózatot, míg a szurikáta és mosómedve némileg hasonlít például egy medvéhez, ezért ezeket felismeri a hálózat.



(a) *Helyes predikciók.*



(b) *Pontatlan predikciók*

4.4. ábra. *Példák jó és pontatlan szegmentációra.*

4.5. ábra. *A hálózat tesztelése olyan képekre, amiken az állatok jól beállított szögben vannak fényképezve, és jól láthatóak. A kettős expozíciós képek készítéséhez is ilyen kiindulási képeket érdemes használni.*

5. fejezet

Állatfejek irányának meghatározása

A kettős expozíciós képek készítésekor fontos szempont, hogy a képeken merre néz az állatok feje, ugyanis az állat és a tájkép közötti áttűnést úgy szeretném elkészíteni, hogy az állat feje látható maradjon, és csak a teste legyen összemosva a tájképpel. Egészen pontosan azt kell meghatározni, hogy a testéhez képest milyen irányban helyezkedik el a feje. Az 5.1. ábrán látható egy-egy példa olyan esetekre, amikor az állatok feje balra, jobbra vagy felfelé néz. Az állatfejek irányának meghatározására egy konvolúciós neurális hálózatot készítettem el, amely a megfelelően felcímkézett képek alapján egy osztályozási feladatot hajt végre: a képeket három kategóriába sorolta aszerint, hogy a rajtuk lévő állat feje balra, jobbra vagy felfelé néz (olyan eset nagyon ritka, hogy az állat feje lentebb van, mint a teste, ezért nem csináltam negyedik osztályt). A végső képet így e neurális hálózat által meghatározott irányt figyelembe véve tudjuk elkészíteni.

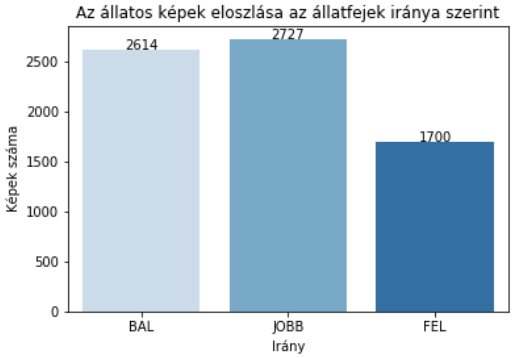
5.1. Adathalmaz

A hálózat tanításához állatokat tartalmazó képeket kell megjelölni aszerint, hogy rajta az állatok feje milyen irányba néz. Nem találtam olyan adathalmazt, amelyben ezek be lennének jelölve a képeken (ez egy elég speciális osztályozás; általában például az alapján

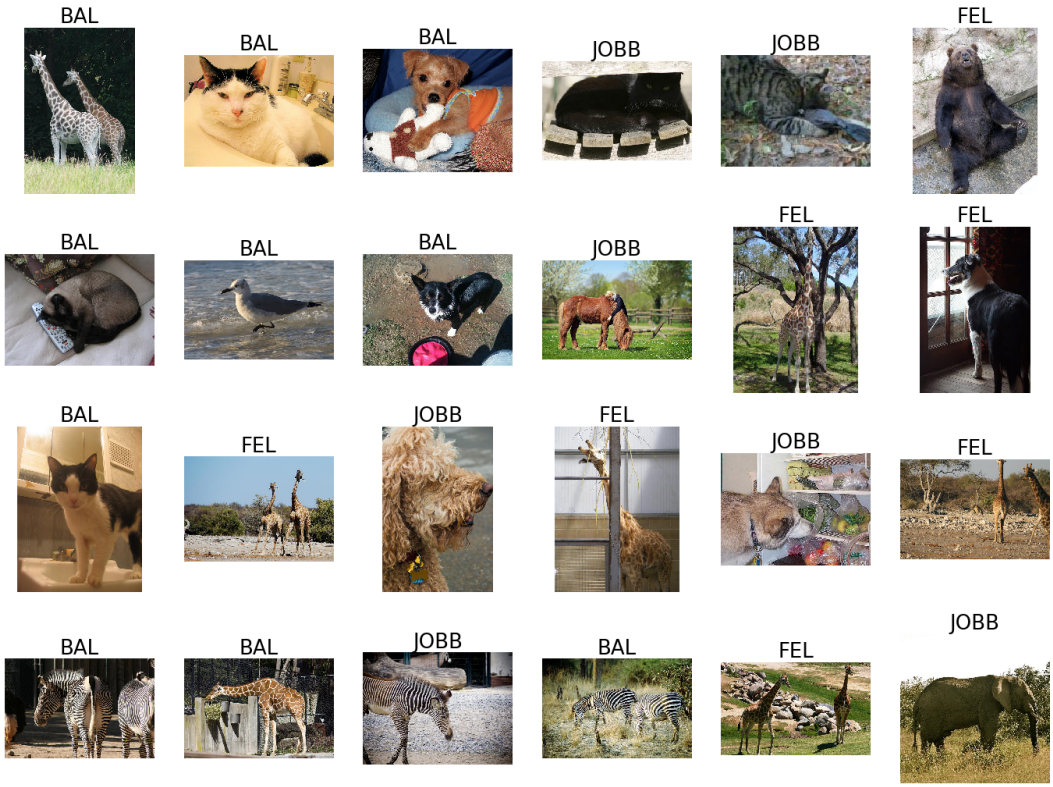
(a) *Balra néző fej* **(b)** *Jobbra néző fej* **(c)** *Felfelé néző fej*

5.1. ábra. *Az állatok fejének irányától függően készítjük el az átmenetet az állatfigurát tartalmazó kép és a tájkép között. Amerre az állat feje néz, ott csak a fej látható, és a kép többi részén, vagyis az állat testénél lesz áttűnés a tájképpel.*

osztályozzák a képeket, hogy milyen állat van rajtuk), ezért kézzel címkéztem fel jó néhány képet. A képeket, mint a másik neurális hálózat esetében, most is a COCO [9] adatbázisból válogattam ki. Azokat a képeket használtam fel, amelyeken egy, vagy legfeljebb két állat található (két állatot tartalmazó képekből csak azokat, ahol az egyik nagyon kicsi, vagy mindkettő egyértelműen ugyanarra néz). Azokat a képeket, ahol nem volt egyértelmű az állatfej iránya (például takarásban volt az állat, vagy csak az állat feje látszott a képen, a teste nem), elhagytam az adatok közül. Így 7041 darab kép maradt, amelyeket megcímkéztem a három kategória valamelyikével. Az egyes kategóriákba került képek száma látható az 5.2. ábrán. Néhány kép a hozzájuk tartozó címkékkel az 5.3. ábrán látható.



5.2. ábra. *Az állatok fejének iránya alapján a különböző osztályokba került képek darabszáma.*



5.3. ábra. *Képek az adatbázisból a hozzájuk tartozó fej-iránnyal.*

A képeket szétosztottam tanító, validációs és teszt halmazokra $70\% - 20\% - 10\%$ arányban. Tanításhoz a tanító adathalmazban levő képeken használtam adatdúsítást: a képeknek a tükrözött változatát is felvettem az adathalmazba, a BAL-JOBB címkék cseréjével, valamint használtam kismértékű elforgatást, zoom változtatást, nyírást és eltolást is. Így tanítás közben a modell mindig kicsit másképpen látta ugyanazt a képet, ezáltal jobban megtanult általánosítani, kevésbé lett túltanítva a tanító képekre. A képeket tanítás előtt a 4.1. képlet szerint normalizáltam. A képek itt is egységes, 224×224-es felbontásúra voltak átméretezve. A címkéket tanítás során OneHot kódolással reprezentáltam, vagyis az elvárt kimenet mindig egy háromelemű vektor, amelyben egyetlen egyes érték található a címkének megfelelő helyen, a többi érték a vektorban nulla.

5.2. Neurális hálózat tanítása

A hálózat feladata a képeken levő állatfejek irányának meghatározása. A lehetséges irányok: balra, fel és jobbra; tehát a hálózatnak minden képet e három kategória valamelyikébe kell besorolnia. Ilyen jellegű hálózatok tanítása során érdemes *transfer learning*-et alkalmazni, vagyis teljesen saját hálózat helyett kiindulásként felhasználni valamelyiket a 2.1.1-ben bemutatott képfelismerő hálózatok közül. Egyszerű felépítése miatt én a VGG-16 hálózatot választottam kiindulásként. Ehhez a hálózathoz elérhetőek a betanított súlyok is, amelyekben rengeteg tudás van: az alsóbb rétegek már képesek a képeken levő egyszerűbb vonalak, formák felismerésére, amely információkat felhasználva a későbbi rétegek felismerik a képen levő konkrét objektumokat. Az ImageNet adatbázisban sok állatos kategória megtalálható, ezért ez a hálózat az eredeti súlyokkal már sok olyan alacsony szintű jellemzőt ismer, ami az állatok felismeréséhez szükséges, és így az állatfejek irányának meghatározásakor is hasznos lehet.

A hálózatot az eredetihez képest kismértékben átalakítottam, a következők szerint: a hálózat végén található előrecsatolt rétegeket eltávolítottam, majd az utolsó konvolúciós réteget lecseréltem egy olyanra, ami 5×5-ös konvolúciót használ 2-es eltolásokkal (*stride*). Ez azért előnyösebb az eredetinél, mert a hálózatban levő paraméterek számát nagyrészt az első előrecsatolt rétegben levő paraméterek száma határozza meg, ami nagyban függ az utolsó konvolúciós réteg kimeneteinek számától; a 2-es eltolás használatával pedig csökkenthető a réteg kimeneteinek száma. Ezután még három előrecsatolt réteget fűztem a hálózathoz, rendre 512, 512 és 3 db kimenettel (az utolsó már az osztályozást hajtja végre). Az előrecsatolt rétegek között használtam BatchNormalization-t, Dropout-ot $0,3$-as értékkel, valamint ReLU aktivációt. Az utolsó réteg Softmax aktivációt használ az osztályozáshoz. A teljes hálózatnak így 42 millió paramétere lett (szemben az eredeti VGG-16-tal, amiben 138 millió paraméter volt; tehát jelentős csökkenést sikerült elérni az utolsó rétegek átalakításával).

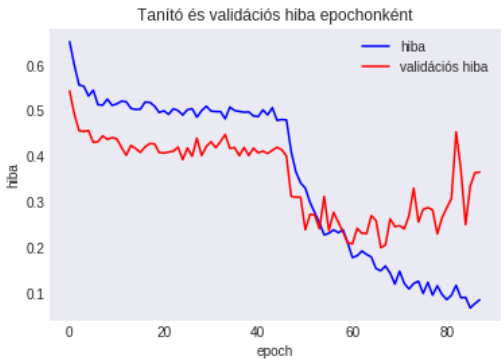
A hálózatot két fázisban tanítottam. Az első fázisban az eredeti VGG hálózat súlyait tartalmazó rétegeket befagyasztottam, és csak a többi réteget tanítottam, így az eredeti, már megtanult jellemzőkből kiindulva megtanulta a modell a klasszifikációt. Ezután a második fázisban már az egész hálózat súlyait módosítottam a tanítás során, így az

alacsonyabb szintű jellemzők finomhangolása is megtörtént, ezáltal még pontosabb lett a modell. A tanításhoz 16-os *batch size*-t használtam, a hálózatok súlyainak módosításához az SGD algoritmust használtam 10^{-3}-os tanulási rátával, valamint $0,9$-es momentummal. A hibafüggvény kategorikus keresztentrópia (*categorical crossentropy*) volt. A tanítás itt is generátorral történt, egy-egy epochban 150 batch-nyi képet használtam fel. Az epochok számának optimális meghatározásához a tanítás mindkét fázisában *early stopping*-ot használtam, 20-as *patience* értékkel (vagyis akkor állt meg a tanítás, ha 20 epoch-ig nem javult a validációs hiba). A tanítás a két fázisban 46 és 42 epochig tartott, ami összesen körülbelül két órát vett igénybe a GPU-s szerverkörnyezetben.

5.3. Kiértékelés

A tanítás végére a hálózat 88%-os pontosságot ért el, vagyis a képek 88%-a esetében tudta helyesen meghatározni az állatfejek irányát.

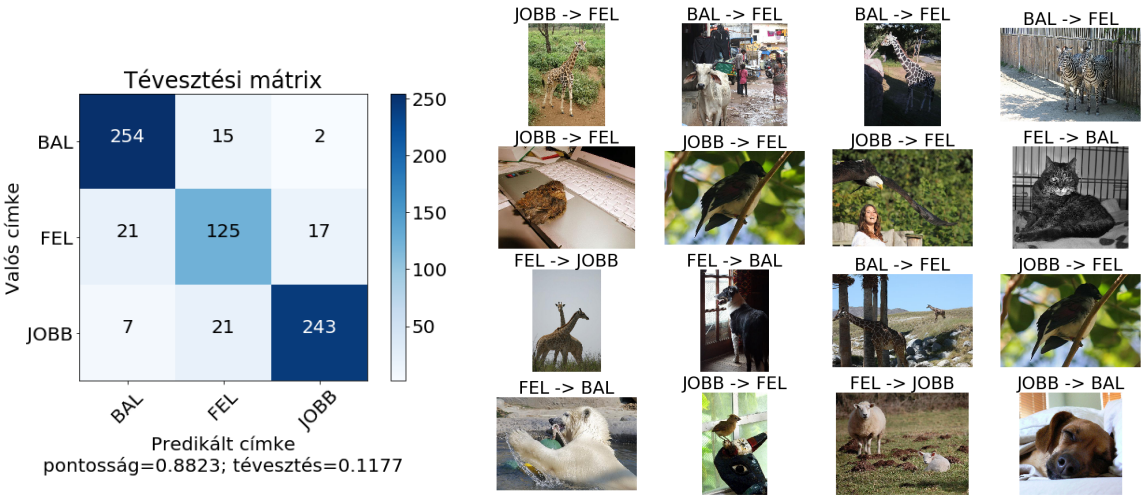
Tanítás közben a hibafüggvény (*loss*) mértéke látható az 5.4. ábrán. Az ábrán megfigyelhető, hogy a 46 epochig tartó első fázis után a második fázis kezdetekor jelentősen lecsökkent a mért hiba. A tanítás végére a hálózat már túltanult, ekkor visszatöltöttem azt az állapotot, amikor minimális volt a validációs hiba.



5.4. ábra. *A tanító és validációs hiba értéke tanítás közben. 46 epochig tartott a tanítás első fázisa, ezután kezdődött a második.*

A tesztadatokra elkészített tévesztési mátrix az 5.5a. ábrán látható. Az ábrán megfigyelhető, hogy az esetek döntő többségében a modell jól határozza meg az állatfej irányát; a legtöbb esetben a felfelé irányt téveszti össze a balra vagy a jobbra iránnyal (a balt és jobbat nagyon ritkán keveri össze). Ennek az is lehet az oka, hogy egy-egy képhez néhány esetben több irány is jó lehet: például egy balra néző zsiráf esetében a fej egyrészt balra néz, másrészt fent van. Néhány hibás címkézés látható az 5.5b. ábrán. Ezeken a képeken is megfigyelhetőek olyan esetek, amikor a képekhez a valós és a modell által predikált címke is elfogadható lenne.

A tesztképek mellett megvizsgáltam a modell által készített predikciókat azokra a jól beállított képekre, amelyekből a kettős expozíciós képeket is készítettem. Ez látható az 5.6. ábrán. Az itteni húsz kép esetében a modell mindössze egyetlen esetben tévedett, és itt is elég bizonytalan volt a döntésében (43% volt a tippje a BAL irányra, ami a helyes lenne); a legtöbb esetben magabiztosan helyesen állapította meg az állatfej irányát.



(a) *Tévesztési mátrix a tesztadatok alapján.*

(b) *Tévesen meghatározott címkék (valós → predikált).*

5.5. ábra. *A modell eredményességének szemléltetése: tévesztési mátrix és néhány hibás predikció. A hibás predikciók esetében sok esetben nem egyértelmű a képhez tartozó címke.*

5.6. ábra. *A modell predikciói jól beállított állatos képekre, amelyekből a kettős expozíciós képek készülhetnek, és a döntés bizonyossága. A modell az itteni 20 képből mindössze egy esetben tévedett, de ott is bizonytalan volt a döntésében.*

6. fejezet

Kettős expozíciós kép előállítása

A két neurális hálózatot felhasználva elkészíthetjük a kettős expozíciós képet.

6.1. Kép előállítása

Művészi hatású kettős expozíciós képeket általában képszerkesztő programokkal (pl. Adobe Photoshop) szoktak készíteni. A képek elkészítéséhez sok leírást lehet találni az interneten, az egyszerűbbektől kezdve a bonyolultabb effekteket is tartalmazókig. Én most egy olyan algoritmust használok, ami viszonylag egyszerűen működik, nem használ egyedi speciális effekteket, így bármilyen képekkel használható. (Habár egy egyedileg elkészített, az adott kép témájához illő speciális effekteket tartalmazó kép látványosabb lesz, az automatikus generáláshoz egyszerűbb általános módszert használni, amellyel gyorsabban készülhetnek a képek.)

A kettős expozíciós kép elkészítéséhez egy állatot tartalmazó képet és egy tájképet használunk fel. A kép fő témáját az állat adja, amelynek a teste áttűnést képez a tájképpel. A kép elkészítésének folyamata a 3.1. ábrán látható. Az elkészítés első lépése, hogy az állatalakot tartalmazó képről körbevágjuk az állatot, eltávolítva az eredeti hátteret. Ehhez a 4. fejezetben bemutatott neurális hálózatot használjuk fel. Az állatot tartalmazó képet átméretezzük 224×224-es méretűre, majd a hálózat segítségével elkészítjük a hozzá tartozó maszkot. A maszkot ezután felnagyítjuk az eredeti kép méretére, és abból vágjuk ki az állatot; így a kép az eredeti felbontásban megmarad, míg a nagyítás egy jól generált maszk esetében legfeljebb néhány pixelnyi pontatlanságot jelent a nagyobb felbontású képen, ami általában nem észrevehető.

A generált maszk alapján a tájképből is kivágunk egy, az állat alakjával megegyező formájú részt. Ha szükséges, akkor ehhez a tájképet felnagyítjuk vagy lekicsinyítjük. Így van egy körbevágott állatunk, és egy ugyanilyen alakra vágott tájképünk, amikkel elkészíthetjük az átmenetet.

Az átmenet elkészítéséhez generálunk egy gradienst tartalmazó képet, ami az állat helyén egy 0-ból 1-be tartó átmenetet tartalmaz, és azt jelenti, hogy az adott ponton mennyire erősen fog megjelenni a tájkép az állat mellett. A gradiens az állat fejétől indul nullás értékekkel, és az állat teste felé haladva egyre nagyobb értékek kerülnek bele, ahogy egyre

erősebben jelenik majd meg a tájkép az átmenetben. A gradiens irányának meghatározásához tudnunk kell, hogy melyik irányba néz az állat feje (pontosabban, hogy a testéhez képest merre helyezkedik el a feje). Ennek eldöntésére használható az 5. fejezetben bemutatott neurális hálózat, ami meghatározza, hogy az állat feje balra, jobbra vagy felfelé néz. A meghatározott irány alapján elkészíthetjük a gradiensátmenetet tartalmazó képet.

A kettős expozíciós kép elkészítése a következő képlet szerint történik:

$$im = (1 - grad) \cdot msk \cdot cropped + grad \cdot msk \cdot land + (1 - msk), \tag{6.1}$$

ahol:

- im: az elkészülő kép,
- $grad$: a generált gradiensátmenet,
- msk: az állatos képhez tartozó maszk,
- $land$: a tájkép,
- $cropped$: az állatos kép.

Vagyis az állatfigurát tartalmazó képet és a tájképet körbevágjuk a maszk segítségével, majd a gradiensátmenet szerinti súlyozott összegüket vesszük. A végén hozzáadott $(1 - msk)$ a fehér hátteret biztosítja.

A 6.1. egyenletben szereplő minden tag egy, az állatot tartalmazó kép felbontásával megegyező méretű mátrix; a képek és a maszk is 0 és 1 közé skálázott értékeket tartalmaznak; a szorzások elemenkénti szorzásokat jelentenek. Ahhoz, hogy a műveletek elvégezhetőek legyenek, a tájképből ki kell vágni egy részt, amely megegyezik az állatos kép méretével.

6.2. Korrekciók

A képek generálásakor előfordul, hogy a neurális hálózatok nem végeznek tökéletes munkát, és tévednek a predikcióik során. Azonban vannak olyan, kisebb tévedések, melyek még egyszerű szabály alapú módszerrel javíthatóak, ezekből mutatok be itt egyet-egyet a két hálózat esetében.

6.2.1. Fej-irány korrekció

Az állatfej irányát meghatározó mély neurális hálózat tévedéseit egyszerűen tudjuk javítani: ehhez csak be kell állítani kézzel a generálás során, hogy merre néz az állat. Így ha ez a hálózat téved, de a másik pontosan felismeri az állat körvonalát, akkor továbbra is el tudjuk készíteni a kettős expozíciós képeket.

6.2.2. Maszk korrekció

Kissé bonyolultabb a helyzet, ha a maszkot generáló hálózat ad pontatlan eredményt: mivel itt sokkal komplexebb a kimenet, ezért ezt nehezebb automatikusan javítani. Vannak

olyan esetek a maszk generálásakor, amikor a hálózat nagyjából helyesen felismeri az állat körvonalát, de az állat belsejében bizonyos részeket "halványabban" címkéz (azaz kisebb bizonyossággal állítja, hogy ott is állat található), vagy az állat mellett a háttér egyes részeit kis valószínűséggel állatnak hiszi. Az ilyen jellegű hibák javíthatóak úgy, hogy a generált maszkban bizonyos küszöbérték alatti értékeket 0-ra, vagy az efölöttieket 1-re állítjuk. Ezáltal ezekben a speciális esetekben a képszegmentációt megvalósító hálózat tévedéseit is tudjuk korrigálni.

6.3. Eredmények értékelése

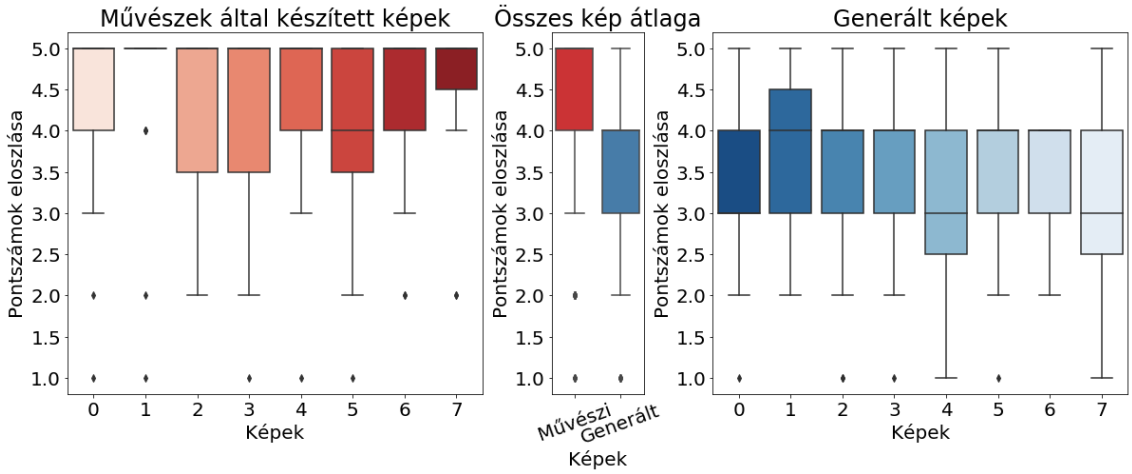
A generált képek minőségét, szépségét több tényező határozza meg. Ezek közül a legfontosabb az, hogy a képszegmentáció mennyire volt sikeres, vagyis mennyire volt pontos az első neurális hálózat. Ha ez nem találja meg elég pontosan a képen az állatot, és nem tudja jól körbevágni, akkor az előállított kép biztosan nem fog jól mutatni. Ha viszont ez a hálózat pontosan működik, akkor az elkészített kép minőségét az ezt előállító algoritmus további lépései már nem befolyásolják jelentősen: a fej irányát meghatározó hálózat esetleges tévedései kézzel javíthatóak, a két képet összefésülő algoritmus pedig mindig ugyanúgy működik, nem rontja vagy javítja az elkészült eredményt. A kettős expozíciós képek szépségét ezenkívül az határozza meg, hogy milyen állatos képet és tájképet választottunk kiindulásként: sok olyan állatalakot tartalmazó kép van, melyből sehogy sem lehet szép kettős expozíciós képet csinálni. Az állatot tartalmazó kép kiválasztásánál fontos, hogy az állat jól látható legyen a képen, és a kép hátterét eltávolítva is jól felismerhető legyen; vagyis például ne lógjanak be különböző tárgyak az állat elé, amiket kivágva a kapott kép "lyukas" lenne, vagyis hiányoznának az állat egyes részei (például egy magas fűben álló állat esetén körbevágás után a a fű eltűnik, így a lába rövidebbnek fog tűnni a valóságosnál a kivágott képen). A tájképet célszerű úgy megválasztani, hogy a színvilága illeszkedjen az állatéhoz, vagy a témája kapcsolódjon az állat jellemző előfordulási helyéhez – például egy farkast ábrázoló képhez passzol egy téli, hegyvidékes táj, de egy zebrához nem.

6.1. ábra. *Sikertelen kettős expozíciós képek. A sikertelen generálás általában abból adódik, hogy nem sikerül helyesen körbevágni az állatot; de ha például az állat lábai nem látszanak az eredeti képen, akkor is furcsa végeredményt kaphatunk.*

A 6.1. ábrán látható néhány példa sikertelen képgenerálásra. A legtöbb esetben a sikertelenség oka az, hogy az állatos képeket nem sikerült pontosan körbevágni (első három kép); a negyedik képen pedig egy olyan példa látható, ahol az eredeti képen nem látszanak az állat lábai, így a körbevágott képen lábak nélkül nem annyira jól néz ki az állat.

Sikeresen generált képekre a függelékben láthatóak példák az F.1.1. és az F.1.2. ábrán.

A generált kettős expozíciós képek szubjektív kiértékeléséhez egy tesztet készítettem, melyben a tesztalanyok a tesztben szereplő képeket 1-től 5-ig osztályozták aszerint, hogy mennyire találják szépnek őket. Az értékelendő képek között szerepeltek művészek által, manuálisan készített kettős expozíciós képek és a megoldásom alapján automatikusan generált képek is, mindkettőből 8-8 darab. Az értékelés célja az volt, hogy összehasonlíthatóak legyenek a generált képek a művészek által készített képekkel. Az értékelést 23 ember végezte el, az általuk a képekre adott pontszámok eloszlása külön a művészek által készített képekre és a generált képekre a 6.2. ábrán látható.



6.2. ábra. *A felhasználók értékelték a művészek által készített és a program által generált képeket. Az ábrákon az egyes képekre adott pontszámok eloszlása látható. A bal és jobb oldali diagramon látható a művészek által manuálisan készített és a dolgozatban bemutatott rendszer segítségével automatikusan generált képekre adott értékelések megoszlása képenként külön-külön, a középső ábrán pedig a nyolc-nyolc képre együttvéve. A dobozok színes részei a képre adott, sorba rendezett értékelések 25 és 75%-a között lévő értéket fedik le (helyenként a dobozok belsejében lévő vonal a mediánt mutatja), a dobozból kinyúló szakaszok és különálló pontok a kiugró értékeléseket mutatják.*

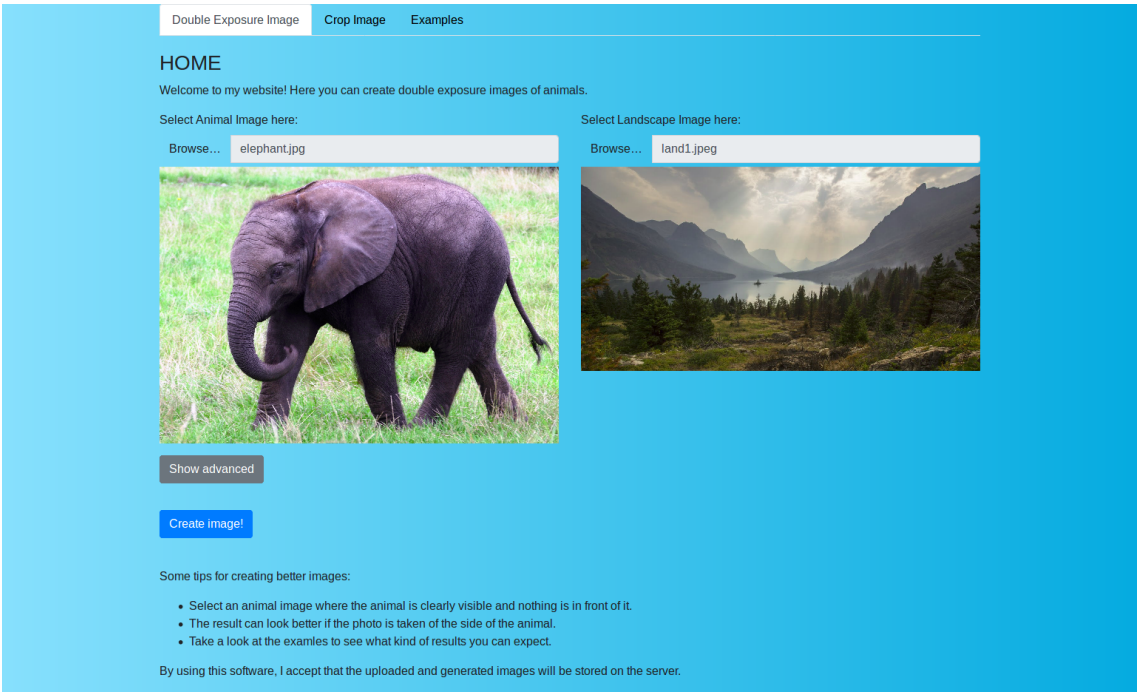
Az ábrán megfigyelhető, hogy a művészek által készített képek általában magasabb pontszámot kaptak (átlagosan $4,29$-at, míg a generált képek $3,44$-t). Ennek az elsődleges oka valószínűleg az, hogy a művészek egyedi effektusokat is használtak, amelyekkel látványosabbá tehetőek a képek; valamint a művészek által készített képeken jellemzően jobban passzol egymáshoz az állatos kép és a tájkép, mint az általam kiválasztott párosítások esetében, ami szebb összhatást eredményez. Mindemellett az automatikusan generált képek is sok 5-ös és 4-es pontszámot kaptak, ami azt mutatja, hogy ezek között is vannak olyanok, amelyek elnyerték a felhasználók tetszését.

6.4. Webes interfész

A kettős expozíciós képeket elkészítő algoritmust Python nyelven implementáltam, így egy parancssorból futtatható programot kaptam. Az egyszerűbb használhatóság érdekében elkészítettem hozzá egy webes interfészt, amin lehetőségünk van a képek elkészítésére. A felületen kiválaszthatjuk a két képet, amiket fel szeretnénk használni, és ezekből automatikusan elkészül a kettős expozíciós kép. Ha nem vagyunk elégedettek az eredménnyel, akkor lehetőségünk van a 6.2. fejezetben leírt korrekciók végrehajtására. A felület emellett lehetőséget ad arra is, hogy egy állatos képen levő állatot körbevágjunk, így ha például ezt az állatot egy másik képre szeretnénk odaszerkeszteni, ehhez is segítséget nyújt a rendszer. Az elszigetelés és a könnyebb hordozhatóság érdekében a webes alkalmazást egy Docker konténerbe helyeztem el, amiben telepítve vannak a futáshoz szükséges Python könyvtárak, és a webalkalmazás kódja mellett bele vannak csomagolva a betanított modellek is, így ez a konténer tartalmaz mindent, ami a webalkalmazás futtatásához szükséges.

A kettős expozíciós képek elkészítése nagyjából fél-egy percet vesz igénybe. Az idő körülbelül felét az állat körbevágása, a maszk generálása, vagyis a neurális hálózatok futtatása, másik felét pedig az összevágott kép elkészítése veszi igénybe. Ez utóbbi részhez szükséges idő erősen függ a képek felbontásától is, hiszen itt már az eredeti felbontású képpel kell dolgozni; én nagyobb, akár 5000×3000 pixel felbontású képekkel is teszteltem a rendszert, de kisebb képek esetében gyorsabb a végeredmény előállítása.

A webes felület kinézete a 6.3. ábrán látható. A felület a *http://deep2.tmit.bme.hu:5000/* címen érhető el.



6.3. ábra. *A rendszerhez elkészített webes felhasználói felület kinézete. A képek kiválasztása után a Create image! gombra kattintva elkészíthetjük a kettős expozíciós képet.*

7. fejezet

Összefoglalás

Dolgozatomban egy olyan rendszer elkészítését és működését mutattam be, mellyel egyszerűen és gyorsan lehet szinte bármilyen állatos képből és tájképből művészi hatású kettős expozíciós (*double exposure*) képet készíteni.

A képek elkészítéséhez egy állatos képre és egy tájképre van szükség, melyek kombinációjából automatikusan előállítható a kettős expozíciós kép. A kép elkészítési folyamatának legfontosabb lépése az állat körbevágása a képen, ezt egy konvolúciós mély neurális hálózattal oldottam meg. A hálózat képszegmentációt valósít meg, azaz képes pixelpontosan meghatározni, hogy a képen hol helyezkedik el az állat. A hálózat tanításához a COCO adatbázisban lévő képeket használtam fel. A kettős expozíciós képek készítéséhez emellett meg kell határozni, hogy a képen milyen irányba néz az állat feje, ugyanis az állatos kép és a tájkép közötti átmenetet úgy valósítom meg, hogy az állat feje látható maradjon, és csak a teste legyen összemosva a tájképpel. Ehhez egy másik neurális hálózatot tanítottam be, amely képes megállapítani, hogy a képen az állat balra, jobbra vagy felfelé néz. A hálózat tanításához állatos képeket címkéztem fel aszerint, hogy azokon merre néznek az állatok. A körbevágott állatos képből és a tájképből ezután az állatfej irányának figyelembevételével elkészíthetőek a kettős expozíciós képek.

A rendszer egyszerű használhatósága érdekében elkészítettem egy webes felületet, ami lehetőséget ad kettős expozíciós képek készítésére. A felületen az állatos kép és a tájkép feltöltése után automatikusan elkészül a generált kép. A szebb eredmények érdekében végrehajthatunk bizonyos korrekciókat az előállított képen. A kettős expozíciós képek készítése mellett a weboldalon lehetőség van az állatos képek körbevágására is.

7.1. Lehetséges jövőbeli fejlesztések

A kettős expozíciós képek jelenleg egy több lépcsőből álló folyamat végeredményeként állnak elő. Megvizsgálható, hogy lehetséges-e egyetlen neurális hálózattal, *end-to-end* módon kettős expozíciós képek készítése. Ekkor egy neurális hálózat a kiválasztott állatos képet és tájképet kapná meg bemenetként, és ezekből közvetlenül előállítaná a végső képet. A feladat nehézsége, hogy e hálózat felügyelt tanításához nem áll rendelkezésre adathalmaz, és ilyet összegyűjteni – még az itt bemutatott rendszer használatával is – rendkívül nehézkes

és időigényes folyamat.

A jelenlegi rendszer alapjául szolgáló képszegmentációnak más felhasználási területei is vannak állatos képek körbevágásán kívül. Egy kapcsolódó alkalmazás lehet például képeken (vagy akár videókon) levő emberek körvonalának felismerés és kivágása, majd másik képre (vagy videóra) áthelyezése. Így a filmes világból ismert *green screen*-hez hasonló módszerrel lecserélhető az eredeti háttér akkor is, ha a felvételt nem zöld vászon előtt készítettük.

Ezenkívül a képszegmentációhoz kapcsolódó alkalmazási terület az önvezető autók kameraképeinek elemzése vagy orvosi képek analizálása, ahol bár a feladat bonyolultabb, mint az itt bemutatott, állatokat felismerő hálózat esetében, de a megoldás ehhez hasonló módszerekkel kapható meg.

Köszönetnyilvánítás

Szeretném kifejezni köszönetemet mindazoknak, akik a dolgozat elkészítéséhez segítségükkel hozzájárultak. Elsősorban szeretnék köszönetet mondani a konzulensemnek, Dr. Gyires-Tóth Bálintnak, aki magas szintű szakmai tudásával segítette a munkámat, türelemmel és készséggel válaszolt felmerülő kérdéseimre, javaslataival és értékes tanácsaival iránymutatást adott a dolgozatom elkészítéséhez. Köszönöm emellett a családomnak és barátaimnak, hogy ötleteikkel, javaslataikkal segítették az elkészült projekt még jobbá tételét.

Irodalomjegyzék

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[2] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[3] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Ross Girshick et al. "Region-based convolutional networks for accurate object detection and segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 142–158.

[6] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[7] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[8] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[9] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[11] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *arXiv preprint arXiv:1511.00561* (2015).

[12] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[13] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[14] Liang-Chieh Chen et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *arXiv preprint arXiv:1802.02611* (2018).

[15] Guosheng Lin et al. "RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation." In: *Cvpr*. Vol. 1. 2. 2017, p. 5.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[17] Simon Jégou et al. "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE. 2017, pp. 1175–1183.

[18] Gao Huang et al. "Densely Connected Convolutional Networks." In: *CVPR*. Vol. 1. 2. 2017, p. 3.

[19] Hengshuang Zhao et al. "Icnet for real-time semantic segmentation on high-resolution images". In: *arXiv preprint arXiv:1704.08545* (2017).

[20] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *arXiv preprint arXiv:1508.06576* (2015).

[21] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. "Inceptionism: Going deeper into neural networks". In: *Google Research Blog. Retrieved June* 20.14 (2015), p. 5.

[22] Ahmed Elgammal et al. "CAN: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms". In: *arXiv preprint arXiv:1706.07068* (2017).

[23] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[24] Mark Everingham et al. "The pascal visual object classes challenge: A retrospective". In: *International journal of computer vision* 111.1 (2015), pp. 98–136.

[25] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[26] Alexey Dosovitskiy et al. "Discriminative Unsupervised Feature Learning with Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 766–774.

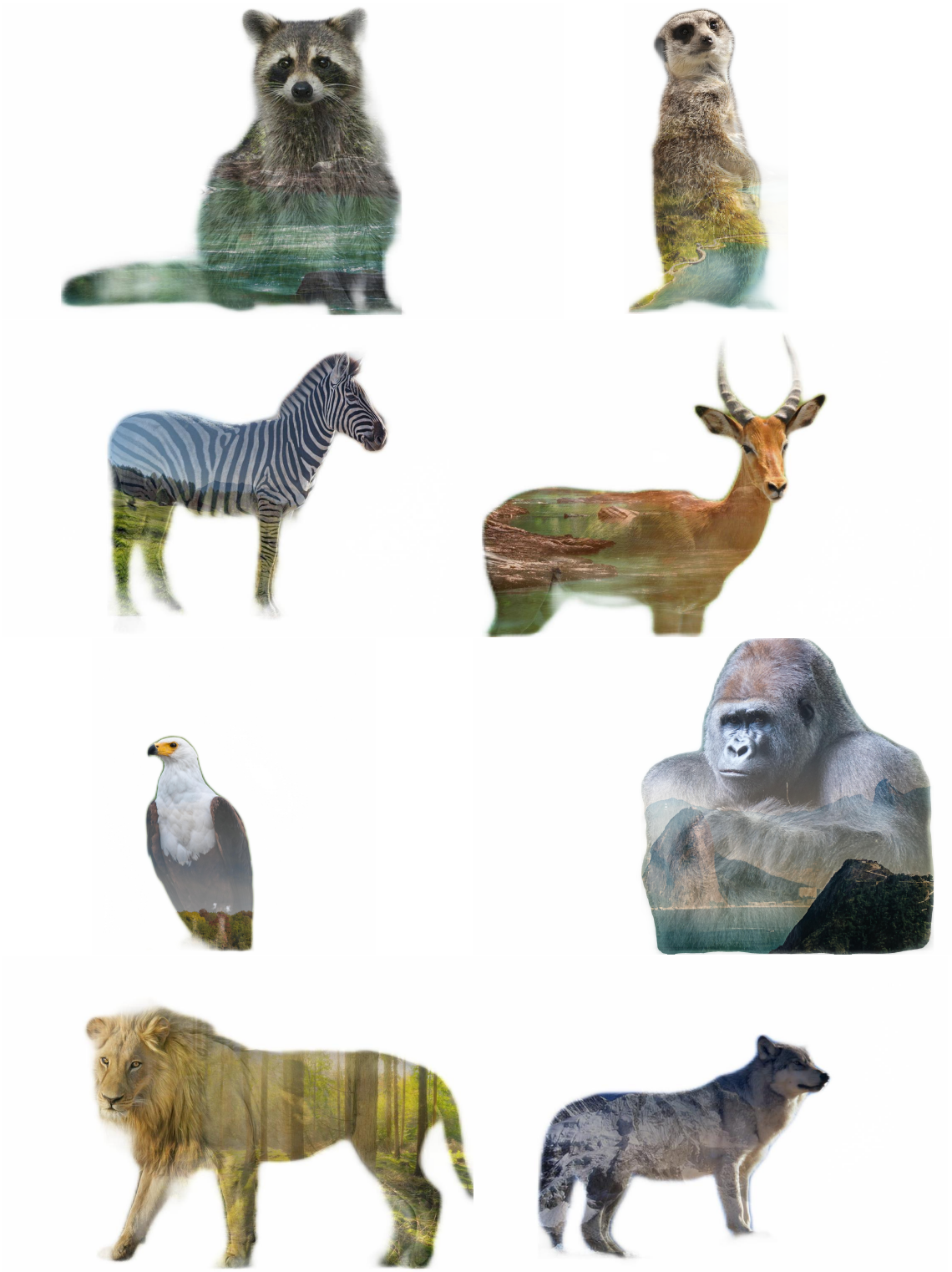
[27] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[28] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

Függelék

F.1. Generált kettős expozíciós képek

F.1.1. ábra. *A program által generált kettős expozíciós képek.*



F.1.2. ábra. *A program által generált további kettős expozíciós képek.*