VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ KOLOROVÁNÍ VIDEA

AUTOMATIC VIDEO COLORIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **TOMÁŠ MIKESKA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2020

Abstrakt

Tato práce se zabývá plně automatickým kolorováním videa a fotografií pomocí konvolučních neuronových sítí. Shrnuje dosavadní přístupy, architektury a různé chybové funkce. V rámci práce jsou navrženy a natrénovány různé varianty neuronových sítí s různými chybovými funkcemi pro automatické kolorování. Nejlepší kombinace zvládá kolorovat velké množství scén a je demonstrována její schopnost kolorovat dostatečně koherentní video.

Abstract

This thesis deals with fully automatic colorization of video and images using convolutional neural networks. It summarizes existing approaches, architectures and several objective functions. In this thesis several neural networks with different objective functions are implemented and trained for automatic colorization. The best of them is able to colorize large diversity of scenes and is sufficient to produce coherently colorized video.

Klíčová slova

neuronové sítě, konvoluční neuronové sítě, automatické kolorování, GAN, počítačové vidění

Keywords

neural networks, convolutional neural networks, automatic colorization, GAN, computer vision

Citace

MIKESKA, Tomáš. *Automatické kolorování videa.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Automatické kolorování videa

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, PhD. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Mikeska
28. května 2020

Poděkování

Tímto bych chtěl velmi poděkovat mému vedoucímu práce Ing. Michalovi Hradišovi, PhD., za skvělé vedení, ochotu, věcné připomínky a rady, které mi poskytl při řešení této práce.

Obsah

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Mikeska Tomáš**
Program: Informační technologie
Název: **Automatické kolorování videa**
Automatic Video Colorization
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte základy neuronových sítí typu Generative Adversarial Networks.
2. Vytvořte si přehled o současných metodách generování a transformace obrazu pomocí neuronových sítí se zaměřením na kolorování fotografií a videa.
3. Vyberte konkrétní metodu vhodnou pro kolorování videa a navrhněte její možná vylepšení.
4. Implementujte navrženou metodu a proveďte experimenty.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát prezentující vaši práci, její cíle a výsledky.

Literatura:

- Tero Karras, Samuli Laine, Timo Aila: A Style-Based Generator Architecture for Generative Adversarial Networks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- Satoshi Iizuka, Edgar Simo-Serra, Hiroshi Ishikawa: Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. SIGGRAPH, 2016.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Kapitola 1

Úvod

Kolorování šedotónové fotografie zachycující určitou scénu je pro člověka poměrně snadný úkol, vzhledem k tomu, že člověk víceméně ví, jaké barvy použít tak, aby výsledná fotografie vypadala co nejvíce reálně. Avšak pro počítač je tento úkol naopak velmi obtížný a správné obarvení vyžaduje kompletní vizuální porozumění scény. Jde tedy o značnou výzvu v oblasti počítačového vidění. Zároveň se kolorování dá využít i v praktické činnosti, neboť počítač za nás provede pracný a časově náročný úkol.

Tato práce se zabývá plně automatickým kolorováním obrázků a videí na reálných fotografiích a zároveň zhodnocením dosavadních přístupů.

Restaurování fotografií nebo videa pomocí automatického kolorování zažívá v posledních letech velká zlepšení, především díky pokroku v trénování hlubokých neuronových sítí a stale většímu dostupnému výpočetnímu výkonu. Výsledek systému automatického kolorování je odhad realistického obarvení původního šedotónového obrázku, přičemž přijatelných obarvení může být více a mírou kvality výsledku je uvěřitelnost obarvení hodnocená pozorovateli - lidmi. Pro úspěšné obarvení obrázku potřebuje neuronová síť rozpoznat jednotlivé objekty ve scéně a ty vhodně obarvit tak, aby byla zachována lokální i globální konzistence obrázku a výsledné obarvení působilo co nejvíce reálně. Při automatickém kolorování videa navíc vznikají požadavky na konzistenci obarvení obrázků v čase. Úspěšná neuronová síť by tedy měla být schopna rozpoznat nejrůznější objekty napříč kategoriemi a vyžaduje tak velkou kapacitu parametrů a natrénování na datové sadě velkých rozměrů, aby se mohla setkat s velkým množstvím scén. Zároveň je kladen důraz na barevnost obrázku, kde se často objevují problémy především u objektů, které mohou být potenciálně obarveny velkým množstvím barev. Výsledky obarvení neuronovými sítěmi tak velmi často trpí nízkou barevností a variabilitou.

Potřeba kolorovat šedotónové obrázky nemusí být pouze u starých fotografií či videí pro účely jejich restaurování. Japonský trh s animovanými filmy se každým rokem rozrůstá a v posledních letech přesáhl hodnoty 18 miliard dolarů [32]. Právě při tvorbě některých kreslených filmů či seriálů se obvykle načrtne několik klíčových snímků hlavními tvůrci a ostatní mezisnímky se dokreslují jejich méně zkušenými pomocníky. Takových snímků je obvykle velké množství, a právě jejich obarvení je jednou z nejzdlouhavějších, a tím pádem také nejdražších fází. V této práci se omezíme na kolorování šedotónové fotografie, tedy skutečných scén, přestože kolorování náčrtků by vyžadovalo obdobný přístup a v kapitole 2 bude zmíněno i kolorování jiných domén, jako právě kolorování kreslených filmů.

Kolorování šedotónového obrázku počítačem nemusí vždy probíhat plně automaticky. Existují i další přístupy, kdy obrázek obarvuje stále počítač, ale je možné určitým jednoduchým způsobem ovlivnit, jak bude výsledek vypadat. Může se jednat o kolorování pomocí

specifikování palety barev, kdy je neuronové sítí zároveň poskytnuta informace o cílené paletě několika barev a ty jsou při obarvení upřednostněny. Dále se může jednat o kolorování pomocí referenčního obrázku, kdy požadujeme, aby byl obrázek obarven v podobných barvách jako na obrázku referenčním. Extrakce palety se zde děje tedy automaticky. Jindy může jít o více manuální přístup, při kterém přímo specifikujeme jeden nebo více barevných bodů v šedotónové fotografii a neuronová síť má při kolorování tyto barvy dodržet a dobarvit zbytek fotografie tak, aby bylo celé obarvení konzistentní. Existují však i experimenty, u kterých jsou barvy obarvení specifikovány v textově podobě. Uživatel takového systému tedy specifikuje barvy libovolných objektů ve větě a neuronová síť musí tyto nároky naplnit a zároveň se pokusit obrázek obarvit realisticky. Tato práce se zaměřuje na plně automatické kolorování, přestože zmíněné poloautomatické metody jsou obvykle modifikací popisované architektury a chybové funkce. Dále je v této práci demonstrována efektivita uvedeného řešení i na odlišné úloze, kterou je umělé zvyšování rozlišení obrázků.

Obrázek 1.1: "Migrant Mother", Dorothea Lange, 1936. Nalevo je původní fotografie, napravo obarvená fotografie získaná pomocí neuronové sítě trénováné v této práci.

Kapitola 2

Předchozí práce

V této kapitole budou popsány důležité komponenty tvořící systém pro automatické kolorování obrázků nebo videa ve vybraných pracích. Obvyklá architektura použitá v existujících systémech [4] [40] [35] [2] [14] pro kolorování obrázků se skládá z kolorovací neuronové sítě, která na vstupu přijme šedotónový obrázek a na výstupu generuje buďto přímo obarvený obrázek v RGB, anebo pouze jeho barevné kanály, které musí být doplněny o původní šedotónový obrázek. Výstup kolorovací neuronové sítě je poté poslán do chybové funkce, která může být realizována více způsoby a její volba má velký vliv na kvalitu výsledného obarvení. Chybová funkce často vyžaduje referenci, tedy skutečný barevný obrázek, ale není to vždy pravidlem jako například u generativního trénování. Po dosažení konvergence při optimalizaci takové chybové funkce, můžeme natrénovaný model použít ke kolorování neviděných šedotónových obrázků.

2.1 Architektura

V této sekci jsou prvně vysvětleny základní koncepty konvolučních neuronových sítí, dále jejich běžné vrstvy a některé známé konvoluční architektury, na které bude později v textu odkazováno. Také zde budou popsány některé konkrétní architektury přímo pro kolorování obrázků a videa používané ve vybraných pracích.

2.1.1 Konvoluční neuronové sítě

Základní architekturou, používanou v naprosté většině moderních systémů pro automatické kolorování obrázků a počítačové vidění vůbec, jsou konvoluční neuronové sítě. Jedná se o neuronové sítě složené z konvolučních vrstev, tj. vrstev, kde jsou parametry prostorově sdíleny a tvoří tzv. konvoluční filtry. Typická architektura konvolučních neuronových sítí je několik konvolučních vrstev zapojených za sebou, které jsou následovány jednou nebo více plně propojenými vrstvami. Jedná se o obecnou skupinu sítí, které mohou být používány pro různé účely jako klasifikace, segmentace, apod. Pro účely kolorování jsou používány konvoluční sítě, které mají shodnou šířku a výšku vstupní i výstupní vrstvy. Stejně je tomu tak v sítích používaných pro semantické segmentování scén. Semantické segmentování scén je významný problém počítačového vidění, který si vyžádal velmi velký zájem vědecké komunity, a který proto významně ovlivňuje i výzkum automatického kolorování obrázků a videa.

V práci [29] byl zkoumán význam hloubky neuronové sítě na její výslednou úspěšnost a bylo zjištěno, že hlubší neuronové sítě (tj. sítě s větším počtem vrstev) mají na dostatečně

komplexních úlohách lepší výsledky, než sítě méně hluboké, ale s větším počtem neuronů ve vrstvách, a tedy i podobným počtem parametrů. Hlubší sítě jsou tak často efektivnější při stejném počtu parametrů a dosahují celkově lepších výsledků. Autoři v [29] používají pouze malé 3×3 konvoluční filtry a dosahují velkého zlepšení při použití 16 až 19 vrstev za sebou. Tato architektura je obecně známá jako VGG Network.

Trénování neuronových sítí, o velkém počtu za sebou poskládaných konvolučních nebo plně propojených vrstev, ovšem trpí problémem mizejících gradientů, a tak v praxi nejsou velmi hluboké neuronové sítě natrénovatelné. Proto autoři v [11] používají tzv. reziduální spojení, kdy výstup vrstvy je součtem vstupu a výstupu funkce vrstvy. Autoři empiricky dokazují, že takové sítě lze trénovat několikanásobně hlubší a dosahují výrazně lepších výsledků. V počítačovém vidění je velmi známá právě neuronová síť ResNet z [11], která dosahuje velmi dobrých výsledků v klasifikaci na datové sadě CIFAR-10 a dalších a je možné úspěšně naučit až 152 vrstev této architektury, tedy přibližně 8x hlubší než sítě VGG.

Trénování hlubokých neuronových sítí dále stěžuje fakt, že vstupní distribuce každé vrstvy se během trénování neustále mění, protože se mění i parametry předchozích vrstev. To přináší potřebu trénovat modely s nízkým hyperparametrem learning rate a také zvyšuje nároky na správnou inicializaci parametrů jednotlivých vrstev. Autoři v [15] nazývají tento fenomén *internal covariate shift* a navrhují řešení v podobě normalizační vrstvy nazývané Batch Normalization. Tato vrstva normalizuje výstup předchozí vrstvy tak, že vystřeďuje její průměr a normalizuje varianci. Pomocí pouze dvou přídatných parametrů, tato jednoduše implementovatelná vrstva, výrazně urychluje trénovací proces. Zároveň zlepšuje konečné výsledky, protože je možné trénovat hlubší modely. Mimo zmíněné výhody tato vrstva působí i jako regularizace.

Autoři v práci [30] řeší problém přetrénování, který je významný u hlubokých neuronových sítí. Navrhují přidání nové vrstvy, kterou nazývají Dropout, která náhodně s pravděpodobností p vynuluje některé aktivace předchozí vrstvy. Motivací ke zvolení tohoto řešení je snížení možnosti, že jednotlivé neurony spolu budou příliš spolupracovat. Tato vrstva je aktivní pouze během trénování, při inferenci je její absence kompenzována vynásobením všech vah předchozí vrstvy zvoleným hyperparametrem pravděpodobnosti vynulování p.

2.1.2 Kolorovací neuronové sítě

Cílem generátoru barevné masky je aproximovat funkci, která na vstupu přijme šedotónový obrázek o specifikovaném rozměru a na výstupu vygeneruje obarvený obrázek nebo barevné kanály o stejném rozměru jako vstupní obrázek. Pokud je jako výstup zvolen celý obrázek, jsou obvykle výstupem data v RGB barevném modelu. Jako alternativa se často používá výstup v podobě A a B kanálů barevného modelu CIE L*a*b, které musejí být doplněny o šedotónový kanál L. Použití jiných barevných modelů, například YUV, je v praxi spíše vzácné.

V předchozích pracích [4] jsou použity konvoluční neuronové sítě typu *enkodér-dekodér*. Vstupní obrázek nejprve projde enkodérem, tedy obvykle několika konvolučními a pooling vrstvami, a výstupem je tenzor o menší šířce a výšce, který by měl popisovat příznaky scény na vyšší úrovni abstrakce (ang. *feature vector*). Tento tenzor je poté vstupem dekodéru, který se z něj snaží získat požadovaný výstup celého modelu.

Autoři v článku [4] používají konvoluční neuronové sítě typu *enkodér-dekodér*. Vstupní obrázek nejprve projde enkodérem, tedy obvykle několika konvolučními a pooling vrstvami, a výstupem je tenzor o menší šířce a výšce, který by měl popisovat příznaky scény na vyšší úrovni abstrakce (ang. *feature vector*). Tento tenzor je poté na vstupu dekodéru, který se z

něj snaží získat požadovaný výstup celého modelu. Autoři dále do střední části modelu, k výstupu enkodéru, přidávají výstup předtrénované neuronové sítě a to ve "spojovací"vrstvě, kterou nazývají Fusion. Jde o síť Inception-ResNet-v2 [31] předtrénovanou jako klasifikátor na datové sadě ImageNet [7]. Parametry této sítě se během trénování modelu nemění, není tedy potřeba počítat gradienty pro vrstvy této sítě. Enkodér sítě slouží k extrakci důležitých prostorových informací o scéně, přičemž výstup předtrénované sítě dává modelu globální kontext.

Autoři v článku [14] používají architekturu podobnou jako v [deep koalarization], která ale předtrénovanou síť Inception-ResNet-v2 nahrazuje vlastní sítí. Tato síť není uzamčená a její parametry se tak optimalizují během trénovacího procesu. První část enkodéru nazývají Low-level Features Network a je sdílená v obou částech enkodéru. Výstup této vrstvy jde v jedné větvi enkodéru do části, která se nazývá Mid-level Features Network. Výstup první části v druhé větvi jde do sítě, která se nazývá Global Features Network. Výstup obou těchto větví se spojí v části, která se nazývá Fusion. Síť dále pokračuje dekodérem, v práci nazývaným Colorization Network. Výstup části nazývané Global Features Network navíc vstupuje i do klasifikátoru a síť je tak zároveň optimalizována i jako klasifikátor kategorie scény. Účelem přidání klasifikátoru je napomoci extraktoru globálních příznaků v lepší generalizaci.

Architektura navržená v práci [35] je podobná té z článku Let there be Color! [14]. Obsahuje společný enkodér, který strukturou odpovídá síti VGG16 [29] a jeho úloha je podobná jako má síť Low-level Features Network v [14]. Výstup této části jde podobně do dvou částí, přičemž jejich výstupy se spojují ve Fusion vrstvě a výstup jedné se používá ke klasifikaci scény. Výstup Fusion nadále prochází dekodérem. V případě ChromaGAN popisuji konkrétně architekturu generátoru, jelikož je model trénován jako WGAN [3], obsahuje dále i další neuronovou síť nazývanou diskriminátor.

Některé práce [2] [19] k obarvení používají architektury konvolučních neuronových sítí převzatých z oblasti sémantické segmentace obrázků. Automatická sémantická segmentace je důležitou úlohou počítačového vidění, jejímž cílem je pro každý pixel na obrázku identifikovat objekt, který je na něm zobrazen.

Oblíbený model používaný ve výzkumu automatické sémantické segmentace, a také používaný pro automatické kolorování obrázků, je *U-net* [28]. Jedná se o architekturu založenou na konvolučních neuronových sítích, která má dvě hlavní části stejně jako modely typu *enkodér-dekodér*. Enkodér postupně posílá výstup předchozí vrstvy přes konvoluční bloky, ve kterých se zvětšují velikosti filtrů. Dekodér pak posílá výstup předchozích vrstev přes konvoluční bloky se zmenšující se velikostí filtrů. Výstup bloků enkodéru se, narozdíl od modelů v této práci popisovaných jako *enkodér-dekodér*, navíc přidává na vstup odpovídajících bloků dekodéru.

Právě *U-net* je základem neuronové sítě trénované pomocí známého repozitáře DeOldify [2], který řeší automatické kolorování obrázků i videa. Repozitář používá framework fast.ai[1] a jeho pomocné struktury právě pro vytvoření a trénování *U-net*u. Tato pomocná struktura přijímá jako parametr enkodér a celý dekodér je vytvořen dynamicky na základě enkodéru. Autoři jako enkodér používají sítě ResNet předtrénované na datové sadě ImageNet o různých hloubkách (repozitář trénuje více modelů).

Pro dosažení lepší globální konzistence kolorovaných obrázků repozitář DeOldify přidává, mimo klasické vrstvy, které můžeme nalézt v konvolučních neuronových sítích, také Self-attention vrstvy. Tato vrstva, která má původ v systémech pro zpracování přirozeného

[1] https://docs.fast.ai/

jazyka [34], se vyznačuje tím, že každý bod výstupu je závislý na všech ostatních bodech vstupu. Self-attention vrstva tak dává trénovanému modelu snadnější způsob, jak sjednotit barvy na souvisejících objektech, které jsou ve scéně daleko od sebe. Použití těchto vrstev je doplněno o spektrální normalizaci modelu, tedy nahrazení jednotlivých konvolučních a plně propojených vrstev jejich spektrálně normalizovanými verzemi, čímž se dosahuje lepší stability při trénování.

Autoři se v článku [38] soustředí na správné kolorování objektů, které nemají velké zastoupení v trénovací datové sadě. Jejich architektura zahrnuje kolorovací neuronovou síť doplněnou o tzv. Memory Network. Externí Memory Network napomáhá modelu ukládat kritické informace na dlouhou dobu. Jedná se o množinu trojic (K, V, A), kde K je klíč (ang. Key), V je hodnota (ang. Value) a A je stáří (ang. age). Klíč slouží k tomu, aby se z paměti vybral takový obrázek, který je nejpodobnější hledanému query obrázku. Klíč se získává jako výstup sítě ResNet18-pool5 předtrénované na datové sadě ImageNet, které jako vstup vložíme právě vybraný obrázek a výsledný vektor znormalizujeme. Tato síť je dále optimalizována pomocí tzv. Threshold Triplet Loss, která nutí pomocnou síť mapovat podobné obrázky na vektory s malou kosinovou vzdáleností a naopak odlišné obrázky na vektory s velkou kosinovou vzdáleností. Celá síť je trénována jako Conditional GAN [21], přičemž generátor této sítě je konvoluční neuronová síť typu *enkodér-dekodér*, která ale ve vnitřních vrstvách obsahuje vrstvy AdaIN [13], díky kterým se modelu přidává informace o barevné paletě obrázku extrahovaného z externí paměti.

2.2 Chybová funkce

Správné zvolení chybové funkce má zásadní vliv na barevnost a variabilitu řešení výsledného modelu a také na efektivitu a paměťové nároky trénovacího procesu. Bohužel v případě automatického kolorování obrázků nelze určit žádnou formální metriku, která přesně určuje kvalitu výsledku jako je například úspěšnost v případě klasifikátorů. Kvalita obarvení je určena hodnocením skupiny pozorovatelů s ohledem na cíl obarvení, kterým je obvykle dosažení maximálně realistického obarvení. Je tedy možné vytvořit pouze chybové funkce, které skutečný cíl aproximují, ale nelze jej optimalizovat přímo.

2.2.1 Per-pixel chybové funkce

V pracích [4] [14] autoři používají střední kvadratickou odchylku (MSE) jako chybovou funkci. V obou zmíněných pracích jsou výstupem kolorovací sítě dva kanály reprezentující barevné kanály v barevném modelu CIE L*a*b. Tato chybová funkce vyžaduje znalost reference, tedy skutečného obarvení vstupu. Hlavním problémem použití střední kvadratické odchylky pro automatické kolorování je, že jejich úspěšná minimalizace nevede k výsledkům, které považujeme za optimální, tedy dostatečně barevné a variabilní obrázky. Modely trénované pomocí těchto chybových funkcí často neobarví některá místa, kde je možné použít velké množství různých barev. Případně často volí tzv. bezpečné barvy, tedy nepříliš odvážné barvy, které průměrují odhadnuté pravděpodobnostní rozložení barev.

$$MSE(x) = \frac{1}{CHW} \sum_{c=0}^{C} \sum_{h=0}^{H} \sum_{w=0}^{W} (x_{w,h,c} - y_{w,h,c})^2$$

Autoři v práci [40] argumentují, že použití střední kvadratické odchylky je nevhodné pro multimodální povahu automatického kolorování obrázků, kdy objekt může být obarven více

barvami. V takovém případě střední kvadratická odchylka volí průměr takové množiny, a proto preferuje desaturované obarvení, které není vnímáno realisticky. Pro nekonvexní množiny možných obarvení může být zvolený průměr dokonce mimo takovou množinu a objekt tak může být obarven nepřijatelnou barvou. V práci [40] je proto ke kolorování přistupováno jako k multinomiální klasifikaci. Autoři kvantizují barevný prostor A a B kanálů do přihrádek o velikosti 10 a ponechávají $Q = 313$ hodnot. Neuronová síť se poté trénuje pro daný vstup X nalézt funkci $\hat{Z} = f(X)$, která odhaduje pravděpodobnostní rozdělení barev $\hat{Z} \in [0,1]^{H \times W \times Q}$. Referenční výstup je na toto rozdělení namapován pomocí soft-encoding schématu, které používá 5 nejbližších sousedů. Objektivní funkcí potom je multinomiální křížová entropie (ang. multinomial cross-entropy).

$$\mathcal{L}_{cl}(Z, \hat{Z}) = -\sum_{h,w} v(Z_{h,w}) \sum_{q} Z_{h,w,q} log(\hat{Z}_{h,w,q})$$

kde $v(Z_{h,w})$ je funkce, která každý pixel vyvažuje mírou rarity dané barevné třídy, které pixel nabývá. Jako motivaci za tímto rozhodnutím autoři v [40] uvádějí fakt, že hodnoty A a B kanálu v přirozených fotografiích, například v datové sadě ImageNet, jsou z velké části velmi nízké hodnoty, kvůli častému výskytu velkých nebarevných pozadí. Tento fakt vede k dalšímu sklonu k desaturovaným obarvením. Autoři tedy tento problém řeší právě přidáním funkce $v(Z_{h,w})$, která přiřazuje vyšší hodnoty méně častým barvám podle výskytu na datové sadě ImageNet [7]. Autoři empiricky demonstrují efektivitu tohoto řešení.

Autoři v [38] [36] používají Huberovu chybovou funkci, také nazývanou smooth L1 chybová funkce. Volba této chybové funkce s sebou přináší nutnost zvolit hyperparametr δ, který určuje dělící hranici v absolutní odchylce reference a výstupu. Tento hyperparametr ovlivňuje velikost chyby v případě, že je absolutní odchylka větší než zvolený parametr. Autoři v článku [36] nastavují tento hyperparametr na hodnotu 0.5, hodnota používaná v článku [38] není uvedena.

$$L_H(x) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{pokud } |y - \hat{y}| < \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2 & \text{jinak} \end{cases}$$

2.2.2 Perceptuální chybová funkce

V pracích [2] [35] je používána chybová funkce převzatá z oblasti přenosu stylu obrázků a Super-Resolution. Jde o tzv. perceptuální chybovou funkci (ang. Perceptual loss) z [16], která porovnává výstup jednotlivých vrstev zvoleného předtrénovaného modelu pro výstupní a referenční obrázek. Tento model je předtrénován jako klasifikátor na velké datové sadě a parametry tohoto modelu se v průběhu trénování nemění. V praxi se často používá například konvoluční neuronová síť VGG16 předtrénovaná na datové sadě ImageNet, především díky jejímu dostatečnému výkonu, malé paměťové a výpočetní náročnosti (síť je používána bez konečných plně propojených vrstev) a zároveň velmi jednoduché architektuře. Výstupy jednotlivých vrstev zachycují vlastnosti obrázku na různých úrovních abstrakce a ke kvalitním výsledkům pro danou aplikaci je potřeba najít správné váhy jednotlivých vrstev. Výstupní vrstva je odstraněna a jsou použity pouze předchozí vrstvy. Obě zmíněné práce [2] [35] používají tuto chybovou funkci s dalšími doplňujícími chybovými funkcemi.

2.2.3 Generative Adversarial Networks

Generative Adversarial Networks [9], zkráceně GAN, jsou zcela odlišným přístupem, který je použitý ke kolorování obrázků v pracích [35] [2] [38] [32]. Spočívá v souběžném trénování dvou modelů, generativního modelu nazývaného generátor, který zachycuje hledanou datovou distribuci, a druhého modelu nazývaného diskriminátor, který se snaží odlišit, jestli vzorek přichází z distribuce aproximované generátorem nebo ze skutečné distribuce. Během trénování jsou tedy diskriminátoru předkládány skutečné a vygenerované obrázky a zároveň poskytována informaci o tom, zdali jde o skutečný vzorek. Tyto dva modely spolu soupeří a snaží se dosáhnout stavu, kdy generátor generuje vzorky nerozpoznatelné od skutečné datové distribuce a diskriminátor není schopný skutečné a vygenerované vzorky odlišit. Po dosažení tohoto optima je možné použít generátor pro vytváření nových vzorků dat, které v trénovací datové sadě neexistují. V původním článku [9] je na vstupu generátoru matice šumu o nějaké náhodné distribuci, například Gaussovské. Ale pro automatické kolorování obrázků nebo videa autoři v [35] [2] [38] [32] používají generátor jako neuronovou síť, která generuje obarvený obrázek z šedotónového.

Formálně je cílem nalézt správnou distribuci generátoru p_g nad datovou distribucí x. K tomu definujeme vstupní proměnnou nějaké náhodné datové distribuce $p_z(z)$, funkci mapující tuto náhodou distribuce do požadované datové distribuce $\mathcal{G}(z;\theta_g)$, kde $\mathcal{G}$ (generátor) je derivovatelná funkce - neuronová síť. Dále definujeme funkci $\mathcal{D}(x;\theta_d)$, která přijímá vzorek z datové distribuce a jejím výstupem je skalární hodnota, která reprezentuje pravděpodobnost, že vzorek x přichází ze skutečné datové distribuce, anebo je uměle vygenerován pomocí funkce $\mathcal{G}(z)$. Tyto dvě funkce $\mathcal{G}(z)$ a $\mathcal{D}(x)$ spolu hrají hru minimax se dvěma hráči s ohodnocovací funkcí $V(\mathcal{G},\mathcal{D})$. Při optimalizaci tak hledáme Nashovo ekvilibrium této nekonvexní hry se spojitými mnohodimenzionálními parametry.

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D},\mathcal{G}) = \mathbf{E}_{x\sim p_{data}(x)}[log(\mathcal{D}(x))] + \mathbf{E}_{x\sim p_z(z)}[log(1-\mathcal{D}(\mathcal{G}(z)))]$$

Sítě GAN v základní podobě představené v [9] často v praxi trpí několika problémy, které je činí těžko trénovatelnými. Mezi problémy s trénováním sítí GAN patří mizející gradienty (ang. vanishing gradients), kdy diskriminátor má výrazně lepší úspěšnost než generátor a gradienty, které poskytuje chybová funkce tak mizejí a nejsou dostatečně velké k efektivnímu zlepšování generátoru. Dále se stává, že generátor během trénování začne generovat stále stejný vzorek nebo množinu vzorků, bez ohledu na vstup, nebo že trénovací proces nekonverguje. Pro stabilizaci trénovacího procesu bylo navrženo několik řešení, která se snaží tyto problémy odstranit.

DCGAN [26] je první práce, kde se autorům podařilo natrénovat hluboké konvoluční neuronové sítě jako GAN na několika datových sadách. Trénovací proces se autorům podařil stabilizovat pomocí několika úprav typického složení neuronových sítí. První z nich je nahrazení pooling vrstev konvolucemi s větším krokem (ang. strided convolution). Takové konvoluce při každém kroku posouvají filtr o více polí. Další úpravou je nahrazení plně propojených vrstev a ponechání pouze vrstev konvolučních. Další úpravou je přidání vrstev Batch Normalization, které obecně stabilizují trénovací proces neuronových sítí. Poslední úpravou je správné použití aktivačních funkcí. Autoři v generátoru, mimo výstupní vrstvy, používají aktivační funkce ReLU, ale v diskriminátoru dosahují lepších výsledků s aktivační funkcí Leaky ReLU, především u modelování ve vyšším rozlišení. Autoři provádí experimenty na datových sadách Large-scale Scene Understanding (LSUN), Imagenet-1k a jimi vytvořené datové sadě fotografií portrétů obličejů.

Wasserstein Generative Adversarial Networks [3], zkráceně WGAN, jsou jedním z nejúspěšnějších řešení pro stabilizaci trénovacího procesu sítí GAN. Autoři předkládají teoretické zdůvodnění, proč trénovací proces klasických sítí GAN, optimalizujících Jensen-Shannon divergenci, selhává a nabízejí řešení v podobě nahrazení právě Jensen-Shannon divergence jinou metrikou. Jde o vzdálenost Wasserstein-1, také nazývanou metrika Kantorovich–Rubinstein nebo Earth-Mover (EM) vzdálenost. Autoři empiricky demonstrují efektivitu tohoto řešení při generování syntetických obrázků trénováním modelu na datové sadě LSUN-Bedrooms. Autoři mimo jiné ukazují, že tato nová metrika lépe koreluje s konvergencí generátoru a s kvalitou vygenerovaných vzorků.

Jednou z nevýhod WGAN trénovacího procesu však je, že funguje pouze pro K-Lipschitz funkce, tj. derivovatelné funkce, jejichž gradient dosahuje maximální hodnoty K pro všechny parametry funkce. Aby se tohoto omezení dosáhlo pro neuronovou síť, která představuje funkci generátoru, navrhují autoři v původnímu článku ořezávání parametrů (ang. weight clipping) během trénování. Právě tento způsob je později nahrazen v další iteraci nad WGAN, která se jmenuje WGAN-GP [10], kde autoři představují tzv. Gradient penalty, která namísto ořezání parametrů upravuje objektivní funkci tak, že penalizuje velké gradienty na náhodně vybraných vygenerovaných vzorcích. Tím dosahují lepších výsledků než původní práce s ořezáváním parametrů.

V práci [22] autoři stabilizují trénovací proces sítí GAN pomocí spektrální normalizace. Jedná se o přímé kontrolování Lipschitzovy konstanty jednotlivých vrstev diskriminátoru. Jelikož spektrální norma matice parametrů je jeho největší singulární hodnota, bylo by potřeba v každé iteraci provádět singulární rozklad a tato metoda by se stala výpočetně velmi náročnou. Proto autoři navrhují rychlou aproximaci spektrální normy. V článku je demonstrována efektivita tohoto řešení.

Self-Attention Generative Adversarial Networks [39], zkráceně SAGAN, je práce, kde se autoři zaměřují na dalekonosnou konzistenci výsledků sítí GAN v úloze generování obrázků. Přidávají mezi klasické konvoluční vrstvy navíc i vrstvy typu Attention, konkrétně Self-Attention vrstvy, jejichž výstup je závislý na celém vstupu takové vrstvy. Tento mechanismus dává generátoru možnost lépe zachovat celkovou realističnost celé modelované scény a navíc z tohoto principu těží i diskriminátor, který naopak může snadněji tuto globální konzistenci obrázku kontrolovat. V práci jsou dále použity některé techniky pro stabilizaci trénovacího procesu, konkrétně spektrální normalizace [22] a two time-scale update rule (TTUR) [12]. Autoři empiricky demonstrují kvalitativní vliv přidání těchto vrstev při modelování na datové sadě ImageNet [7], kde výrazně zlepšují výsledky oproti předchozím pracím.

NoGAN je název, který svému procesu trénování dali autoři v repozitáři DeOldify [2]. Jedná se předtrénovaní generátoru i diskriminátoru sítě GAN pomocí přímého trénování s učitelem na maximální možnou úroveň. Model je poté pouze dotrénován jako síť GAN a toto dotrénování již trvá pouze zlomek času. Velkou výhodou této metody je právě předtrénování pomocí konvenčních způsobů, které je mnohem snazší, efektivnější a méně náchylné na volbu hyperparametrů. S dobrým předtrénovacím procesem jsou autoři schopni dosáhnout velmi dobrých výsledků, které už jsou pouze vylepšeny pomocí GAN trénování. Samozřejmě pro libovolné generativní modelování, pro které jsou sítě GAN často používány, nelze vytvořit takový předtrénovací proces. V případě automatického kolorování obrázků je to však velmi snadné a to pomocí chybových funkcí zmíněných v předchozích podsekcích.

2.2.4 Hybridní chybové funkce

V pracích [35] [2] je používáno více chybových funkcí zároveň pro dosažení nejlepších výsledků. Obvykle jde o kombinaci generativního trénování s některou ze zmíněných chybových funkcí. Tyto pomocné chybové funkce výrazně pomáhají stabilizovat trénovací proces sítí GAN.

Autoři v práci [2] používají kombinaci trénování NoGAN s perceptuální chybovou funkcí. V této práci je tedy nejprve natrénován generátor pomocí perceptuální chybové funkce. Po konvergenci generátoru je natrénován samostatně diskriminátor jako binární klasifikátor rozlišující skutečné a obarvené vzorky. Poté jsou tyto dva modely trénovány jako GAN, konkrétně optimalizují Jensen-Shannon divergenci stejně jako v článku [9]. Autoři také zmiňují, že veškeré užitečné trénování jako GAN je provedeno během několika prvních iterací, kdy křivka učení dosáhne pomyslného inflexního bodu. Po dosažení tohoto bodu se trénování již nezlepšuje a pouze osciluje mezi tímto optimem a přesaturovanými výsledky. Autoři tento inflexní bod hledají manuálně prohlížením obarvení validačních obrázků v průběhu trénování.

Objektivní funkce v práci [35] je kombinací tří chybových funkcí a je definována jako

$$\mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathcal{L}_e(\mathcal{G}^1) + \lambda_g \mathcal{L}_g(\mathcal{G}^1, \mathcal{D}) + \lambda_s \mathcal{L}_s(\mathcal{G}^2)$$

kde $\mathcal{G}^1 : L \longrightarrow (a, b)$ je kolorovací síť a $\mathcal{G}^1 : L \longrightarrow y$ je klasifikační síť. Objektivní funkci tedy tvoří střední kvadratické odchylka $\mathcal{L}_e$ mezi kanály A a B barevného modelu CIE L*a*b. Dále distribuční chybová funkce $\mathcal{L}_s$, která je definována jako Kullback-Leiblerova divergence mezi výstupem model a předtrénované VGG16 sítě. Dále se jedná o objektivní funkci $\mathcal{L}_g$ na základě článku WGAN-GP [10].

2.3 Koherence kolorování videa

Metoda udržení koherence kolorování videa v práci [2] je inspirována článkem [27], který se zabývá koherencí přenosu stylu obrázků ve videu, kde je problém udržení koherence ještě markantnější. Autoři volí jednoduchou metodu, kdy aplikují snadnou augmentaci vstupu - přidání šumu například Gaussovské distribuce. Hlavní intuicí za zvolením tohoto řešení je fakt, že pro jednotlivé po sobě jdoucí snímky videa se vstup modelu mění pouze trochu, ale v případě nekonzistentního obarvení se výstup mění radikálně. Přidání gaussovského ruchu na vstup se tak simuluje právě tato změna a neuronová síť je tak optimalizována robustnějším způsobem, který je agnostický k malým změnám a výsledné obarvení je tak více koherentní při sekvenčním obarvení jednotlivých snímků. V práci [2] autoři také demonstrují, že jimi natrénovaný model je dostatečně robustní i bez přidání této augmentace, a je tak možné s ním kolorovat snímky videa sekvenčně při zachování dostatečné koherence.

Autoři v článku [32] se zabývají automatickým kolorováním videa pro obarvování anime seriálů. V práci používají Conditional GAN [21], kde vstupem je šedotónový obrázek, který je podmíněný předchozím barevným snímkem. Vynucení podobnosti aktuálního a předchozího snímku není nijak dále vynucováno v objektivní funkci. Nevýhodou tohoto přístupu je potřeba mít ke každému obrázku i jeho předchozí snímek, což vytváří nároky na datovou sadu, kterou již není tak snadné získat.

Kapitola 3

Metoda

Tato kapitola popisuje navrhované řešení pro úlohu automatického kolorování videa. Prvně budou diskutovány obecné uvažované přístupy ke kolorování, následně některá praktická omezení a poté podrobněji jednotlivé komponenty předloženého návrhu.

Obrázek 3.1: Nalevo je původní obrázek, napravo je obarvení jeho šedotónového ekvivalentu pomocí neuronové sítě trénované v této práci. Přestože je obloha na pozadí obrázku obarveny zcela jinou barvou než v obrázku referenčním, je toto obarvení přijatelné.

3.1 Obecný přístup

Na základě zjištění v práci [2], kde autoři demonstrovali velkou robustnost trénovaného systému pro sekvenční automatické kolorování obrázků, které může být použito i pro automatické kolorování videa, jsem se rozhodl, že i navrhovaná metoda v této práci vyjde ze systému pro automatické kolorování obrázků. Kolorování videa tak probíhá pomocí sekvenčního obarvování jednotlivých snímků. Pomocí úprav zmíněných v následujícím textu se budu snažit o vytvoření dostatečně robustního systému, který bude následující snímky obarvovat se stejnou zvolenou paletou.

Cílem trénovaného modelu je tedy z šedotónového obrázku, tedy obrázku s jedním kanálem, získat barevný obrázek, tedy obrázek se třemi kanály, který je realisticky obarvený a obsahově shodný s obrázkem původním. Formálně je tedy cílem najít funkci f, pro kterou

platí $R = f(G)$, kde $G \in \mathbb{R}^{H \times W \times 1}$ reprezentuje vstupní šedotónový obrázek a $R \in \mathbb{R}^{H \times W \times 3}$ reprezentuje výsledný obarvený obrázek.

Pro nalezení této funkce bude použita konvoluční neuronová síť trénovaná na dostatečně obecné datové sadě stejně jako v zmiňovaných pracích [4] [38] [35] [2] [14]. V následujících podsekcích jsou podrobněji představeny konkrétní trénované architektury a chybové funkce. Vstupem trénovaných neuronových sítí je šedotónový obrázek, který je zmenšen na velikost 256×256. Je použito zmenšení nezachovávající poměr stran původního obrázku, vstupní obrázek tedy není pouze výřezem původního, ani neobsahuje prázdná místa. Výstupem je obarvený obrázek o stejném rozměru jako obrázek vstupní, ale s více kanály pro reprezentaci barvy.

3.2 Praktická omezení

Trénování hlubokých neuronových sítí na velké datové sadě a často za pomoci dalších pomocných sítí (jako v případě perceptuální chybové funkce nebo GANu) je velmi výpočetně náročné. Takové modely se tedy obvykle učí na jedné nebo více GPU často i po dobu několika dní. Mimo jiné vznikají velké nároky na velikosti paměti použité GPU a tak se často některé architektury stávají nenatrénovatelné z praktických důvodů.

Paměťové nároky učeného modelu jsou lineárně závislé na velikosti vstupního obrázku. To je samozřejmě v protikladu s praktickými požadavky, kdy je potřeba obarvený obrázek v ideálním případě získat v původní velikosti. Aby se dalo částečně vyhnut tomuto problému, lze využít faktu, že lidské oko je méně senzitivní na barevné změny než na změny jasu. Je proto možné použít původní šedotónový obrázek v jeho nativním rozlišení a ten doplnit o vygenerované zvětšené barevné kanály. Díky tomu lze obrázky obarvovat ve výrazně menším rozlišení s velmi dobrými výsledky i při obarvování obrázků s vysokým rozlišením. Přesto je výhodou mít větší rozlišení na vstupu, nejen kvůli přesnějšímu výstupu, ale také kvůli tomu, že ve větším rozlišení je učená neuronová síť schopna rozlišit více objektů, které při velké pixelizaci nemusí být rozlišitelné.

Pro další zefektivnění trénování na větším rozlišení lze neuronovou síť nejprve předučit na menším rozlišení, které se vždy po konvergenci může zvýšit a tento proces několikrát opakovat. Výhodou je, že v malém rozlišení je síť schopna naučit se rozpoznat velké množství objektů a trénování v tomto rozlišení je výrazně rychlejší a má menší paměťové nároky. Při zvýšení rozlišení už je model schopný se adaptovat rychle. Dále je možné při inferenci obarvit obrázek po částech, ale v takovém případě model ztrácí globální kontext a výsledky nemusí být konzistentní.

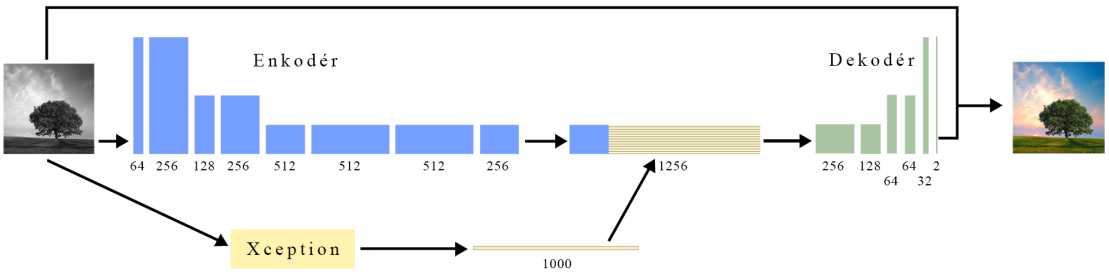
3.3 Architektura

V této sekci je popsána architektura používané konvoluční neuronové sítě. Tato práce trénuje dvě základní architektury a jejich varianty, které obsahují oproti základní verzi některé další vrstvy. První architekturou je *enkodér-dekodér* s fůzí výstupu předtrénovaného modelu. Druhou architekturou je *U-net*. V kapitole 4 jsou popsány jednotlivé experimenty s popsanými neuronovými sítěmi a jsou hodnoceny rozdíly mezi jejich výsledky.

3.3.1 *Enkodér-dekodér*

První trénovaný model vychází z práce Deep Koalarization [4]. Jedná se o architekturu typu *enkodér-dekodér*, kde do střední části je přidán výstup předtrénovaného modelu Inception-

Obrázek 3.2: Diagram sítě *enkodér-dekodér* používané v této práci s Xception předtrénovanou neuronovou sítí jako extraktorem příznaků.



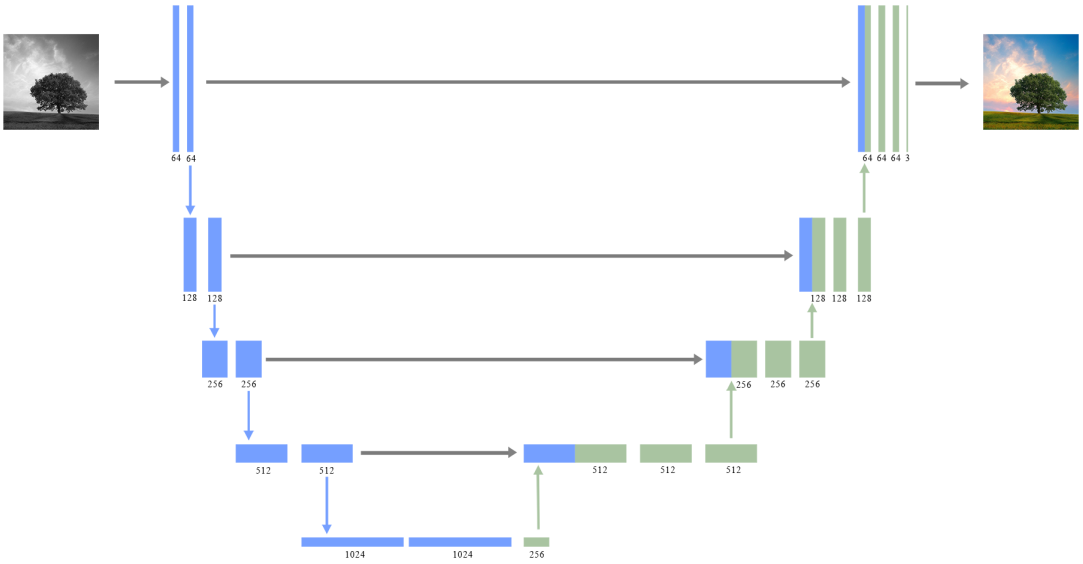
ResNet-v2. Původní architektura popsaná v [4] má velké paměťové nároky, mimo jiné právě kvůli použití předtrénované sítě Inception-ResNet-v2. V této práci je tento předtrénovaný model nahrazen o něco novější a méně paměťově a výpočetně náročnou architekturou Xception [6], která dosahuje srovnatelných validačních výsledků v klasifikaci na datové sadě ImageNet [7]. Natrénování modelu s Xception, jako extraktorem příznaků, dává pocitově srovnatelné výsledky, ale trénování je rychlejší, má menší paměťové nároky a rychlejší inferenci.

Enkodér modelu se skládá z 8 konvolučních vrstev s ReLU [1] aktivačními funkcemi za každou z nich. Každá vrstva používá filtry velikosti 3×3. Enkodér neobsahuje pooling vrstvy, ale ke zmenšení vstupu používá krok o velikosti 2 v první, třetí a páté konvoluční vrstvě. Dojde tedy k osminásobnému zmenšení.

Vstup modelu je zároveň zaslán do sítě Xception, která je předtrénovaná jako klasifikátor na datové sadě ImageNet. Parametry této předtrénované sítě jsou po celou dobu našeho trénování zamčeny. Výstup enkodér a zároveň předtrénované Xception sítě vstupují do vrstvy Fusion, kde jsou spojeny. Při vstupním obrázku o velikosti 256×256 je velikost výstupního vektoru z enkodéru $32 \times 32 \times 256$. Jako výstup předtrénované sítě Xception používáme její konečný výstup používaný ke klasifikaci, tedy tenzor o velikosti 1001×1. Pro spojení těchto dvou tenzorů o různých rozměrech se postupuje stejně tak, jako u autorů v [4] a replikuje se výstupní vektor sítě Xception tolikrát, aby mohl výstupní tenzor z enkodéru doplnit do hloubky. Pro uvedené velikosti tenzorů tedy výstupní tenzor Fusion vrstvy bude mít rozměr $32 \times 32 \times 1256$.

Výstup Fusion vrstvy je jediným vstupem dekodéru, který již na výstupu generuje výstupní RGB obrázek o požadované velikosti. Dekodér je složen znovu z konvolučních vrstev s velikostí filtru 3×3 s aktivační funkcí ReLU [1]. K zvětšení vstupu je používána vrstva UpSampling2D, která používá jednoduchou interpolaci pomocí nejbližšího souseda. Poslední konvoluční vrstva používá aktivační funkci tanh. Celá síť obsahuje 30 milionů parametrů, přičemž pouze 7,5 milionu z nich je trénovaných. Celý model je znázorněn na obrázku 3.2.

Obrázek 3.3: Diagram sítě *U-net* používané v této práci.



3.3.2 *U-net*

Další trénovaný model v této práci je založený na architektuře *U-net* [28]. První část tohoto modelu, která vstup postupně zmenšuje, bude nazývána enkodér. Druhá část, která vstup zvětšuje a přijímá výstup z příslušné části enkodéru, bude nazývána dekodér.

Enkodér je složen z 10 konvolučních bloků. Jeden takový blok se skládá z jedné konvoluční vrstvy s velikostí filtru 3×3, následuje normalizační vrstva BatchNormalization z [15] a poté aktivační funkce ReLU [1]. Druhý, čtvrtý, šestý a osmý konvoluční blok zároveň zmenšuje vstup a to použitím kroku o velikosti 2 v konvoluční vrstvě. Enkodér je zakončen jednou konvoluční vrstvou s 256 filtry, velikostí filtru 1×1 a aktivační funkcí ReLU [1].

Dekodér je složen ze 4 bloků, které jsou strukturou složitější než bloky v enkodéru, a proto také počet vrstev v dekodéru je výrazně větší než v enkodéru. Jeden takový blok kombinuje výstup předchozího bloku dekodéru, případně poslední vrstvy enkodéru v případě prvního bloku, a výstup příslušného bloku enkodéru, který pracuje s tenzory o stejném rozměru. Poslední vrstvou dekodéru je konvoluční vrstva, s třemi filtry s velikostí filtru 1×1, která zaručuje, že výstupem modelu je tenzor se třemi kanály. Celý *U-net* je znázorněn na obrázku 3.3.

Attention *U-net*

Variantou sítě *U-net* je poté model Attention *U-net*, který obsahuje stejnou strukturu, ale za každým blokem dekodéru následuje vrstva Self-Attention. Dále je celý model spektrálně normalizovaný, tedy všechny konvoluční vrstvy jsou nahrazeny jejich spektrálně normalizovanou variantou. Spektrální normalizace stabilizuje trénování sítí GAN a proto je zvolena v případě, že je tento model trénovaný jako GAN.

3.4 Chybová funkce

V této sekci jsou popsány chybové funkce použité v této práci. Volba chybové funkce má v případě automatického kolorování obrázků zásadní vliv na výsledky natrénovaného modelu. Chybová funkce ovlivňuje především barevnost výsledných obarvení. Vyhodnocení experimentů s chybovými funkcemi je provedeno v kapitole 4.3.

První chybovou funkcí, která je v této práci používána, je střední kvadratická odchylka mezi kanály A a B barevného modelu CIE L*a*b. Při použití této chybové funkce je výstupem trénované sítě tedy tenzor s 2 kanály, ne celý RGB obrázek se 3 kanály. Formálně je tedy cíl trénování pozměněn a je potřeba najít funkci f, pro kterou platí $R = f(G)$, kde $R \in \mathbb{R}^{H \times W \times 2}$ reprezentuje A a B kanály obarveného obrázku. Výhodou této chybové funkce je absence dalších hyperparametrů.

Další chybovou funkcí je perceptuální chybová funkce, která vychází z Feature Reconstruction Loss z práce [16]. Avšak pro účely automatického kolorování jsou použity jiné hyperparametry, protože v původní práci je tato chybová funkce používána k přenosu stylu obrázku a k Super-Resolution. Dále je zároveň zahrnuta střední kvadratická odchylka přímo mezi výstupem sítě a referenčním obrázkem. Tato chybová funkce používá síť VGG16 předtrénovanou jako klasifikátor na datové sadě ImageNet. Minimalizuje se eukleidovská vzdálenost mezi výstupy některých jejích vrstev pro výstup kolorovací sítě a referenčního výstupu. K výpočtu jsou používány vrstvy ϕ_i předtrénované sítě VGG16, které jsou následovány Pooling vrstvou.

$$\mathcal{L}_p(y, \hat{y}) = \sum_i \lambda_i (\phi_i(y) - \phi_i(\hat{y}))^2$$

Další a poslední možnou variantou je trénování modelu jako sítě GAN s pomocnou perceptuální chybovou funkcí. Objektivní funkce generátoru je dána

$$\mathcal{L}(\mathcal{G}, \mathcal{D}) = \mathcal{L}_p(\mathcal{G}) + \lambda_g \mathcal{L}_g(\mathcal{G}, \mathcal{D})$$

kde $\mathcal{L}_p$ je pomocná perceptuální objektivní funkce

$$\mathcal{L}_p(\mathcal{G}) = \mathbf{E}_{(x,\hat{y}) \sim p_{data}} \sum_i \lambda_i (\phi_i(\mathcal{G}(x)) - \phi_i(\hat{y}))^2$$

a $\mathcal{L}_g$ je objektivní funkce generátoru. V této práci minimalizuji Jensen-Shannon divergenci mezi generovanou a skutečnou datovou distribucí při trénování GAN stejně jako v práci [9], proto

$$\mathcal{L}_g(\mathcal{G}, \mathcal{D}) = \mathbf{E}_{x \sim p_{data}(x)}[log(\mathcal{D}(x))] + \mathbf{E}_{x \sim p_z(z)}[log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

kde p_{data} je skutečná datová distribuce a p_z je generovaná datová distribuce.

3.5 Koherence videa

Pro zlepšení koherence kolorovaného videa tato práce používá stejný přístup jako [2], tedy řešení založené na práci týkající se přenosu stylu videa [27]. Do vstupu neuronové sítě přidávám šum s rovnoměrnou distribucí, přičemž tento šum není přidán do referenčního výstupu. Neuronová síť se tak snaží tento šum na vstupu odstranit. Nezaznamenal jsem výrazné zlepšení koherence videa, ale výsledky, které navrhovaná metoda produkuje, jsou uspokojivé. Některé ukázky obarvených videí jsou v příloze.

3.6 Datová sada

Tato práce používá dvě datové sady. První z nich je menšího rozsahu a byla používána k hledání hyperparametrů a zkoušení jednotlivých variant. Druhá datová sada je velkého rozsahu a slouží ke konečnému natrénování řešení nalezených pomocí menší datové sady.

Datová sada malého rozsahu používaná v této práci je Tiny-ImageNet[1]. Jedná se o volně dostupnou část datové sady ImageNet [7], kterou poskytuje Standfordova univerzita pro kurz CS231n. Tato datová sada obsahuje 200 různých tříd a pro každou z nich 500 trénovacích, 50 testovacích a 50 validačních obrázků. V této práci jsou modely optimalizovány na trénovací datové sadě, tedy na 10 tisících obrázcích, proto je trénování velmi rychlé - epocha trvá obvykle 5-20 minut. Výsledky trénování hodnotíme pouze na evaluační datové sady poskytnuté k Tiny-ImageNet.

Základem hlavní datové sady velkého rozsahu v této práci je datová sada Places365 [41]. Tato volně dostupná datová sada obsahuje 1,8 milionu trénovacích obrázků z 365 kategorií scén. Jedná se o datovou sadu interiérových a exteriérových scén, ale v této datové sadě nenajdeme kategorie jako portréty obličejů nebo fotografie zvířat a květin. Protože především portrétní fotografie jsou důležitou doménou, kterou je často potřeba kolorovat, byla tato datová sada doplněna o datovou sadu CelebA [20]. CelebA je volně dostupná datová sada obsahující 200 tisíc portrétních obrázků celebrit. V této práci používáme originální snímky bez dalšího ořezávání a upravování, které je vhodné spíše pro rozpoznávání obličejů.

[1] https://tiny-imagenet.herokuapp.com/

Kapitola 4

Experimenty a vyhodnocení

V této kapitole je věnován prostor pro představení experimentů, které byly v rámci této práce provedeny a následně demonstrace výsledků včetně prezentace výstupních obrázků.

4.1 *Enkodér-dekodér* a MSE

V tomto experimentu byla trénovaná architektura *enkodér-dekodér* popsaná v 3.3.1 se střední kvadratickou odchylkou (MSE) jako chybovou funkcí. Model byl trénován na datové sadě Tiny-ImageNet.

Problémem této kombinace je nízká barevnost a také konzistence obarvení. Předměty jsou obvykle obarvené pouze z části, což se jeví jako ještě výraznější problém při kolorování videa, kde neustále se měnící barvy působí velmi nezdařile.

Při dalších experimentech jsem zjistil, že fůze výstupu z předtrénované architektury nemá velký vliv na výsledné obarvení a model naučený bez této části dosahuje srovnatelných výsledků. Protože nelze stanovit formální metriku pro podporu toho tvrzení, je možné, že jde pouze o můj osobní předsudek založený na empirickém pozorování na vlastní validační datové sadě.

Výsledky tohoto experimentu jsou ukázány na obrázku 4.1. Trénovaná síť produkuje desaturované výsledky a zároveň často obarvuje pouze část nalezených objektů. Jako hlavní problém tohoto řešení hodnotím právě použití architektury *enkodér-dekodér*. Úkolem této architektury není pouze obarvit obrázek, ale zároveň veškeré potřebné informace o obrázku shrnout v nízkodimenzionálním vektoru, který je výstupem enkodéru, ze kterého poté dekodér vytvoří celou barevnou masku. To představuje další výzvu, kterou model musí během trénování překonat, přestože je to zbytečné, protože úkolem trénovaného modelu není vnitřně získávat nízkodimenzionální vektor popisující danou scénu jako například při učení modelů typu Autoencoder. Dalším problém je použití velmi jednoduché MSE chybové funkce, která podporuje nižší saturaci [40].

Obrázek 4.1: Vybrané validační obrázky kolorované výsledným modelem trénovaným pomocí architektury *enkodér-dekodér* se střední kvadratickou odchylkou. Obrázky jsou desaturované a objekty často pouze z části.

4.2 *U-net* a perceptuální chybová funkce

V tomto experimentu byla trénována architektura *U-net* popsaná 3.3.2 a perceptuální chybová funkce popsaná v 3.4. Architektura *U-net* na rozdíl od architektury *enkodér-dekodér* přidává spojení mezi vrstvami enkodéru a příslušnými vrstvami dekodéru, které pracují se stejnou velikostí vstupu. Právě díky těmto spojením je odstraněna potřeba modelu vnitřně modelovat scénu v nízkodimenzionálním vektoru. Zároveň používaný *U-net* nepřijímá výstup z žádné předtrénované neuronové sítě a namísto toho jsem výrazně zvětšil počet parametrů samotné trénované neuronové sítě.

Kombinace sítě *U-net* s perceptuální chybovou funkcí dosahuje výrazně lepších výsledků než *enkodér-dekodér* s MSE. Celková barevnost obrázků se na první pohled zvýšila, stejně tak obrázky dosahují větší lokální konzistence. Předměty jsou často obarvené celé a nedochází k barevným skokům. Tato architektura mimo jiné dosahuje velmi dobrých výsledků již po velmi krátké době v řádech několika desítek minut.

Přestože s použitím *U-net*u a perceptuální chybové funkce trénovaný model dosahuje poměrně barevných výsledků, není obarvení rozhodně dokonalé. Hlavním problémem této metody je nově vzniklá chyba, která se objevuje na obarvených fotografiích. Jde o zelené skvrny na některých objektech a zelený artefakt lemující hrany obrázku.

Obrázek 4.2: Vybrané validační obrázky kolorované výsledným modelem trénovaným pomocí architektury *U-net* pouze s perceptuální chybovou funkcí. Na obrázcích se vyskytuje zvláštní chyba v podobě zelených skrvn.

4.3 GAN s pomocnou perceptuální chybovou funkcí

V tomto experimentu byla architektura *U-net* popsaná v 3.3.2 trénovaná jako GAN s pomocnou perceptuální chybovou funkcí. Objektivní funkce je přesněji popsána v 3.4. Ke stabilizaci trénovacího procesu jsou použita doporučení ze známého paperu DCGAN [26] a z článku [5]. Mezi použité vylepšení patří tzv. smooth positive labels, kdy namísto binárních štítků 0 a 1 jsou použity vyhlazené hodnoty jako například 0.1 a 0.9. Díky tomuto triku se dá částečně předejít přetrénování, protože se trénovaný model nenaučí přiřazovat hodnoty blízké jedničce, kde je gradient sigmoidy velmi malý. Dále byly podle doporučení nahrazeny nelinearity typu ReLU [1] nelinearitou typu Leaky ReLU [37] a upraveny výchozí hodnoty optimalizátoru Adam [17]. Dále jsem po vzoru práce TTUR [12] použil odlišný learning rate v optimalizátoru generátoru a diskriminátoru.

Po určité době trénování v tomto nastavení došlo ke kolapsu, kdy například generátor tvořil pouze celočervenou barevnou masku pro všechny vstupy. Nepodařilo se mi s jistotou zjistit příčinu neúspěchu. Je možné, že nebyly použity správné hyperparametry, při kterých by docházelo ke konvergenci. Trénování bylo potřeba včas zastavit, protože pokud byla váha GAN chybové funkce velká, docházelo po čase k až přesaturovaným a příliš odvážným obarvením, které nepůsobily realisticky. Typickým příkladem byly příliš oranžové obličeje postav. Při ještě delším ponechání trénování došlo k úplnému kolapsu a model začal generovat zcela nerealistické barevné masky, které často ani neodpovídaly struktuře obrázku. Použití GAN s pomocnou perceptuální chybovou funkcí pomohlo částečně odstranit zelené artefakty, které produkovalo trénování pouze s perceptuální chybovou funkcí.

Obrázek 4.3: Vybrané validační obrázky architektury *U-net* trénované jako GAN s pomocnou perceptuální chybovou funkcí. GAN pomohl odstranit zelené skrvny, které vznikly při použití pouze perceptuální chybové funkce. Výsledky jsou ovšem přesaturované a při další trénování došlo ke kolapsu trénovacího procesu.

4.4 SAGAN

V tomto experimentu jsou aplikovány úpravy uvedené v práci [39]. Do architektury sítě GAN jsou tedy přidány vrstvy Self-Attention pro lepší globální konzistenci a zároveň je trénovaný model spektrálně normalizován pro stabilizaci trénovacího procesu. V tomto experimentu je tedy trénována upravená varianta sítě *U-net*, kterou nazývám Attention *U-net*. Tato architektura je detailněji popsána v 3.3.2. Tento model se tedy trénuje jako síť GAN s pomocnou perceptuální chybovou funkcí stejně jako v experimentu popsaném v 4.3.

Výsledky této kombinace dosahují nejlepších výsledků ze všech uvedených experimentů. Obarvení je realistické, dostatečně barevné a zároveň lokálně i globálně konzistentní.

Obrázek 4.4: Vybrané validační obrázky architektury *U-net* trénované jako SAGAN s pomocnou perceptuální chybovou funkcí. Jedná se o nejlepší trénovanou variantu v této práci.

Kapitola 5

Technické detaily

Pro implementaci všech experimentů používám programovací jazyk Python 3. Veškeré neuronové sítě používané v této práci jsou implementovány v open-source frameworku Keras [1], který poskytuje vysokoúrovňové rozhraní pro rychlou tvorbu neuronových sítí s podporou trénování na GPU za použití různých backendů. Jako backend pro Keras používám TensorFlow verze 1.14. TensorFlow [2] je open-source knihovna pro strojové učení od společnosti Google, která umožňuje tvorbu výpočetních grafů a mimo jiné jejich efektivní optimalizaci na GPU. TensorFlow je dnes dominantní volbou pro backend frameworku Keras a od verze TensorFlow 2.0 je Keras zaintegrován přímo do jádra TensorFlow.

Jelikož trénování velmi hlubokých neuronových sítí na velkých datových sadách, jako v mém případě, je velmi výpočetně náročné a je potřeba jej provádět často i několik desítek hodin na výkonných GPU. Neuronové sítě jsem trénoal v cloudové platformě Paperspace [3], která se zaměřuje právě na poskytování GPU instancí pro experimenty s umělou inteligencí. Makefile dostupný v kořenovém adresáři zdrojových kódu obsahuje i cíl, který umožňuje vytvořit cloudovou instanci a spustit na ní trénování pomocí jednoho příkazu. Celý proces trvá pouze několik desítek sekund. Samozřejmě se nahrávají pouze zdrojové kódy, datová sada je v cloudu fixně uložena a trénované instance si ji pouze připojují do svého souborového systému.

Pro snadnější údržbu používaného ekosystému, který vyžaduje pro trénování na GPU správné nastavení knihoven CUDA a předinstalování dalších podpůrných systémových balíčku, a umožnění efektivního trénování v cloudu, používám nástroj Docker [4] - software pro efektivní uzavření aplikací do kontejnerů, které mohou být deterministicky replikovány kdekoliv, kde je nainstalovaný Docker Runtime. V kořenovém adresáři zdrojových kódů se nachází soubor Dockerfile, který definuje celý proces vytváření takového obrazu kontejneru. Mimo to je tento vytvořený kontejner uložen v repozitáři Docker Hub [5], odkud je snadnější a rychlejší tento obraz získat. Instrukce pro stažení obrazu a spuštění kontejneru se nacházejí v souboru README.md. Obraz je založen na operačním systému Ubuntu 18.04, je v něm nainstalována Nvidia CUDA 10.0, programovací jazyk Python 3, knihovny OpenCV a TensorFlow s podporou GPU.

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] https://www.paperspace.com/
[4] https://www.docker.com/
[5] https://hub.docker.com/r/tomikeska/ml-box

Pro sledování experimentů a správu jejich výsledků používám open-source nástroj Comet.ml [6]. Pomocí několika snadných úprav trénovacího procesu nám umožňuje na jednom místě sledovat výsledky nebo aktuální trénovací proces všech experimentů prováděných na libovolných zařízeních. Přímo v konzoli webové aplikace Comet.ml je možné pozorovat grafy všech sledovaných metrik, evidovat hyperparametry daných experimentů, sledovat logované evaluační obrázky a další. Jednotlivé experimenty lze označovat štítky pro lepší filtrování a poté křivky různých skupin porovnávat v jednom grafu. V původních experimentech na lokálním zařízení jsem používal vizualizační nástroj TensorBoard, který nabízí i mnoho pokročilejších funkcí, ale Comet.ml dokázal nahradit všechny mnou používané funkce. Na rozdíl od TensorBoard je tato správa centralizována ve webové aplikaci a jednotlivé experimenty k ní data posílají. Použití je tedy vhodnější pro distribuované trénování více experimentů v jedem okamžik na více zařízeních.

Mimo frameworky Keras a TensorFlow využívám v práci několik dalších knihoven pro usnadnění práce. Jedná se o knihovny sklearn [25] a skimage [33] pro některé pomocné funkce a především pro práci s fotografiemi jako převod reprezentace mezi různými barevnými modely. Dále využívám pomoci knihovny PIL [7] pro snadné načítání fotografií z disku. Samozřejmostí je také použití knihovny Numpy [24] pro práci s vektory, maticemi a tenzory.

[6] https://www.comet.ml/
[7] https://pillow.readthedocs.io

Kapitola 6

Single Image Super-Resolution

V této kapitole bude demonstrována efektivita nalezeného řešení na odlišné úloze, kterou je transformace obrázků. Jedná se o problematiku zvyšování rozlišení obrázku. Rekonstrukce detailní fotografie o vysokém rozlišení z předchozí fotografie s nízkým rozlišením byla velmi dlouho velká výzva v oblasti počítačového vidění. Tento úkol je ještě náročnější, pokud je k dispozici pouze jeden obrázek bez jakéhokoliv kontextu. Taková disciplína se nazývá Single Image Super-Resolution (SR) a je významným problémem v oblasti restaurování obrázků. Trénování pomocí technik strojového učení používá obrázky s vysokým rozlišením, jejichž zmenšená verze slouží jako vstup a originál jako referenční výstup. Existující výzkum [8] [18] se obvykle snaží o zvýšení rozlišení 2x nebo 4x.

Mimo Super Resolution existuje i několik dalších technik restaurování fotografií, které si vyžádaly pozornost. Jedním z nich je například Image Denoising - odstraňování šumu z obrázku vzniklého chybou fotoaparátu nebo kamery ve špatných světelných podmínkách. Dále to může být Image Deblurring - jde o zostření rozostřených částí obrázku, které vznikají chybou fotoaparátou nebo kamery při zachycování příliš rychlého pohybu. Další podobnou disciplínou je Image Inpainting, kdy je úkolem realisticky doplnit na zachyceném snímku chybějící místa, která mohou vznikat například u velmi starých a nezachovalých fotografiích. Stejně jako u kolorování šedotónových fotografií jde u všech těchto úloh o transformaci obrázku v obrázek, tudíž se domnívám, že by mělo být možné pro jejich řešení použít modifikaci metody popsané v 3. Nyní se omezím na Super Resolution a pokusím se demonstrovat efektivitu navrženého řešení pro tento problém.

6.1 Historie Super-Resolution

První generací řešení SR byly techniky založené na interpolaci pomocí okolních pixelů. Jedná se především o interpolaci bilineární, která výsledný pixel odhaduje pomocí lineární kombinace okolních bodů. Dále interpolace bikubická, která používá kubické interpolace okolních bodů, anebo interpolace pomocí nejbližších souseda, která přiřadí pixelu hodnotou stejnou jako nejbližší původní pixel, přičemž se zvolí směr zaokrouhlení pro vyřešení konfliktů. Hlavní výhodou těchto přístupů je jejich efektivita, ale samozřejmě tyto techniky nedosahují příliš dobrých výsledků, právě protože jsou velmi jednoduché, vychází pouze z lokální informace a nesnaží se o žádné sémantické pochopení scény.

Jedním z prvních úspěšných použití neuronových sítí pro tento problém je práce [8], kde je představena architektura SRCNN. Jedná se o plně konvoluční neuronovou síť bez jakýchkoliv plně propojených vrstev. Vstupem této sítě je původní zmenšený obrázek, který

se zvětším zpět na původní velikost pomocí bikubické interpolace. Dále tento vstup projde několika konvolučními vrstvami. Mezi výstupem této neuronové sítě a referenčním obrázkem se počítá střední kvadratická odchylka (MSE) jako chybová funkce. Výhodou tohoto přístupu je především velmi nízká výpočetní náročnost, díky které je tento model velmi snadno a rychle natrénovatelný, má nízké paměťové nároky a inference je také velmi rychlá a může probíhat velmi rychle i na obyčejném CPU osobního počítače. Přestože je efektivnější než jednoduché interpolace, nevýhodou je nižší efektivita. To pramení především z důvodu, že tato neuronová síť má velmi malý počet vrstev a receptivní pole je velmi malé. Tato síť tedy stejně jako jednoduché interpolace používá spíše pouze lokální informace a nedochází ke kompletnímu vizuálnímu pochopení scény.

Po příchodu reziduálních neuronových sítí ResNet [11] a jejich velkém úspěchu v klasických disciplínách počítačového vidění se začaly tyto reziduální bloky používat i pro problém Super-Resolution. Autoři v práci SRResNet [18] nahradili jednoduché konvoluční vrstvy v SRCNN právě bloky reziduálními a výrazně zvětšili hloubku této sítě a tedy i počet parametrů. Další výraznou změnou v této práci oproti předchozí generaci je nahrazení bikubické interpolace v modelu transponovanou konvolucí, někdy také nazývanou dekonvoluce, která síti umožňuje naučit se mechanismus zvětšení rozlišení. Vstupem této architektury je tedy nadále zmenšený původní obrázek, ale nedochází k následné bikubické interpolaci.

Jednou z hlavních nevýhod předchozích přístupů je použití kvadratické střední odchylky (MSE) jako chybové funkce. Jelikož jde o per-pixel chybovou funkci, která vždy podporuje spíše řešení, která průměrují přijatelná řešení, výsledky sítí s použitím této chybové funkce jsou často příliš vyhlazené a nepůsobí dostatečně realisticky. Autoři v práci [16] zkoumají již zmíněnou perceptuální chybovou funkci, která používá předtrénované sítě VGG16 a počítá vzdálenost mezi výstupy různých mezivrstev pro výstupní a referenčních obrázek. Tato chybová funkce tedy podporuje spíše větší shodu v abstraktnějších kvalitách obou obrázků a výsledky jsou proto reálnější.

Dalším přístupem pro dosažení nejlepší uvěřitelnosti výsledku je použití sítí GAN, které se přímo snaží o generování výsledků nerozpoznatelných jinou neuronovou sítí od těch skutečných. Autoři v článku [8] trénují dále i model nazývaný SRGAN, kde používají právě sítí GAN. Generátor je reziduální konvoluční neuronová síť, která používá vrstev PixelShuffle pro zvětšení vstupu. Dále objektivní funkci pro optimalizaci sítí GAN kombinuji právě s perceptuální chybovou funkcí pro stabilizaci trénovacího procesu, stejně jako v mnou navrhované architektuře v předchozích kapitolách. Přestože výsledky této metody jsou velmi fotorealistické, velký důraz na nerozpoznatelnost od reálných fotografií je zároveň nevýhodou tohoto přístupu. Problém totiž nastává v případě, kdy se ve scéně nachází objekt, který není v nízkém rozlišení rozpoznatelný. V takovém případě si sítě trénované jako GAN často kompletně domyslí obsah, což nemusí být přijatelné v závislosti na účelu použití.

6.2 Experimenty

V prováděných experimentech vycházím z architektury navržené pro automatické kolorování videa, veškeré zdrojové kódy jsou znovu použity s pouze malými úpravami a změnami hyperparametrů. Experimenty jsou prováděny na vstupních obrázcích o velikosti 128x128 a cílem je dvojnásobné zvětšení, tedy výstupní velikost je 256x256. Jelikož požadujeme obrázky o této velikosti, můžeme použít stejnou datovou sadu jako pro trénování modelů pro automatické kolorování obrázků.

6.2.1 *U-net* s perceptuální chybovou funkcí

V tomto experimentu je použita základní architektura *U-net* s perceptuální chybovou funkcí s upravenými vahami pro jednotlivé vrstvy předtrénované neuronové sítě VGG16. Používám *U-net* bez vrstev Self-Attention a bez spektrální normalizace, jelikož důvodem k jejich přidání by bylo docílení větší globální konzistence, což není problém v případě Super-Resolution a stabilizace trénovacího procesu sítí GAN, které v tomto experimentu nejsou použity. Jako vrstvu, dynamicky zvětšující velikost vstupního tenzoru pro zvýšení rozlišení, zde používám bilineární interpolaci následovanou konvoluční vrstvou po vzoru článku [23], kde autoři porovnávají různé vrstvy tohoto typu.

Jako evaluační metriku jsem přidal výpočet PSNR (peak signal-to-noise ratio), přestože tato metrika přímo nekoreluje s kvalitou výsledků a právě pokročilejší metody používající perceptuální chybovou funkci nebo SRGAN mohou mít PSNR nižší než metody jako SRResNet. Přesto v úvodu trénování se PSNR snižuje, což mi pomohlopři hledání vhodných hyperparametrů.

Výsledky tohoto experimentu jsou velmi dobré, obrázky nejsou příliš vyhlazené, často je realisticky doplněna textura a zároveň nejsou výsledky příliš odvážné - nepřidávají se žádné zcela nové objekty. Jako prostor pro zlepšení vidím především zvětšení rovných čáry u objektů, kdy ve výsledných obrázcích jsou často tyto čáry zubaté.

Obrázek 6.1: Vstupní obrázky v levém sloupci v rozlišení 128×128, výstupní obrázky z trénované neuronové sítě ve velikosti 256×256 uprostřed, referenční výstup ve velikosti 256×256 napravo.

Kapitola 7

Závěr

V této práci bylo navrženo řešení pro automatické kolorování šedotónových fotografií a videa pomocí hlubokých konvolučních neuronových sítí inspirovaných posledním výzkumem v oblasti počítačového vidění. Za pomoci správné volby architektury, chybové funkce a dostatečně rozsáhlé datové sady byla empiricky ukázána schopnost tohoto řešení kolorovat fotografie s různými objekty poměrně realistickým a přijatelným způsobem. Při kolorování videa bylo dosaženo ve velkém případě dostatečné koherence.

Pro dosažení nejlepších výsledků jsem iteroval nad různými volbami architektur a chybových funkcí, které jsem kvalitativně porovnával na vlastní evaluační datové sadě. Během tohoto vývoje jsem se setkal s několika problémy, které se úplně nebo částečně podařilo vyřešit pomocí úprav trénovacího procesu nebo architektury zvolené neuronové sítě.

Přestože navrhovaná neuronová síť provádí kolorování uspokojivě, lze ji dále zlepšovat různými způsoby. Jako příležitost pro další zlepšení se jeví další zvětšení datové sady, především o další kategorie scén, které nejsou dostatečně zachyceny naší současnou zvolenou datovou sadou a tudíž nad nimi sebelepší neuronová síť nemůže generalizovat a tudíž je ani správně obarvit. Ačkoliv by bylo získání větší datové sady poměrně snadné, trénování neuronové sítě na větší datové sadě je výpočetně náročnější, proto jsem musel zvolit limit pro velikost datové sady, na které je, v mých podmínkách, trénovací proces ještě únosný. Dalším zlepšení, které je nasnadě, je zvětšení kapacity neuronové sítě. Takové zvětšení podobně zvyšuje výpočetní náročnost a zároveň i paměťově nároky, proto byla zvolena největší možná neuronová síť, která je v mých podmínkách natrénovatelná.

Mimo aplikaci automatického kolorování videa jsem provedl i experimenty s použitím stejné metody pro Single Image Super-Resolution, kde bylo dosaženo velmi dobrých výsledků a demonstroval jsem tak efektivitu tohoto řešení i pro odlišnou doménu, která také pracuje s transformací obrázku na obrázek. Domnívám se tedy, že navržená metoda je vhodná i pro jiné úlohy, zahrnující restaurování fotografií a videa, jako je odstraňování šumu, rozmazaných částí nebo chybějících částí.

Literatura

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu), 2018.

[2] Jason Antic. Deoldify. https://github.com/jantic/DeOldify. Accessed: 2020-05-24.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[4] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017.

[5] Jason Brownlee. Tips for training stable generative adversarial networks. Accessed: 2020-05-24.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2014.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017.

[13] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[14] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4):110:1–110:11, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[18] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2016.

[19] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity, 2019.

[20] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[21] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[22] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[23] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[24] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[27] Jeffrey Rainy and Archy de Berker. Stabilizing neural style transfer for video. https://medium.com/element-ai-research-lab/stabilizing-neural-style-transfer-for-video-62675e203e42. Accessed: 2020-05-24.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[31] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016.

[32] Harrish Thasarathan, Kamyar Nazeri, and Mehran Ebrahimi. Automatic temporally coherent video colorization, 2019.

[33] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[35] Patricia Vitoria, Lara Raad, and Coloma Ballester. Chromagan: Adversarial picture colorization with semantic class distribution, 2019.

[36] Yi Xiao, Peiyao Zhou, and Yan Zheng. Interactive deep colorization with simultaneous global and local inputs, 2018.

[37] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015.

[38] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory-augmented networks, 2019.

[39] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018.

[40] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[41] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Seznam příloh

- **deep-colorization/** - adresář se zdrojovými kódy pro automatické kolorování
- **superresolution/** - adresář se zdrojovými kódy pro Super-Resolution
- **models/** - adresář obsahující parametry natrénovaného modelu
- **videos/** - adresář obsahující ukázková obarvená videa