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

DOPLNĚNÍ CHYBĚJÍCÍ ČÁSTI OBRAZU POMOCÍ HLUBOKÉHO UČENÍ

IMAGE INPAINTING USING DEEP LEARNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **RADEK ZOBANÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. PETR ŠILLING**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154508

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Zobaník Radek**
Program: Informační technologie
Název: **Doplnění chybějící části obrazu pomocí hlubokého učení**
Kategorie: Zpracování obrazu
Akademický rok: 2023/24

Zadání:

1. Zorientujte se v současných technikách hlubokého učení pro doplnění chybějící části obrazu, např. po odstranění objektu.
2. Seznamte se s architekturami neuronových sítí, které jsou pro tuto úlohu vhodné (autoenkodér, GAN, případně generativní modely) a způsobem jejich učení.
3. Vytvořte datovou sadu pro vlastní experimenty a zvolte vhodnou metodu pro řešení úlohy.
4. Danou metodu implementujte pomocí vybrané knihovny pro modelování a trénování neuronových sítí.
5. Experimentujte s vaší metodou a na vhodných metrikách vyhodnoťte dosažené výsledky. Diskutujte možná rozšíření a kroky pro zlepšení výsledků.
6. Prezentujte vaši práci (její cíle, navrženou metodu a dosažené výsledky) formou plakátu nebo videa.

Literatura:

- Burlin *et al.*, Deep Image Inpainting, technical report, 2017, http://cs231n.stanford.edu/reports/2017/pdfs/328.pdf.
- Yu *et al.*, Generative Image Inpainting with Contextual Attention, CVPR 2018, https://arxiv.org/pdf/1801.07892.pdf.
- Rombach *et al.,* High-Resolution Image Synthesis with Latent Diffusion Models, CVPR, 2022, https://arxiv.org/pdf/2112.10752v2.pdf.

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních tří bodů zadaní a částečně bod 4.

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

Vedoucí práce: **Šilling Petr, Ing.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 9.11.2023

Abstrakt

V této práci vznikla aplikace pro testování a porovnávání metod pro doplnění chybějící části obrazu za využití hlubokého učení a byly natrénovány dvě metody, pconv s konvoluční architekturou, respektive AOT-GAN s GAN architekturou. Práce popisuje návrh výsledné aplikace, její funkcionalitu a důležité body implementace. Byla zvolena datová sada, na které byly vybrané modely optimálně natrénovány. Proběhly experimenty na AOT-GAN modelu, kdy se zkoumal vliv počtu AOT bloků v generátoru na výsledný doplněný obraz. Všechny experimenty byly kvalitativně a kvantitativně porovnány. Výsledky ukázaly úctyhodné výsledky při práci s přírodní scenérií.

Abstract

In this thesis, an application was developed for testing and comparing methods for completing missing parts of an image using deep learning, and two methods were trained, pconv with convolutional architecture, and AOT-GAN with GAN architecture. The thesis describes the design of the finished application, its functionality, and important implementation details. A dataset was selected on which the chosen models were optimally trained. Experiments were made on the AOT-GAN model to investigate the impact of the number of AOT blocks in generator on the resulting completed image. All experiments were qualitatively and quantitatively compared. The results showed respectable outcomes when working with natural scenery.

Klíčová slova

image inpainting, hluboké učení, GAN, konvoluční neuronové sítě, aplikace, Python, datová sada, trénování neuronových sítí

Keywords

image inpainting, deep learning, GAN, convolutional neural network, application, Python, dataset, training neural network

Citace

ZOBANÍK, Radek. *Doplnění chybějící části obrazu pomocí hlubokého učení.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Šilling

Doplnění chybějící části obrazu pomocí hlubokého učení

Prohlášení

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

.....
Radek Zobaník
8. května 2024

Poděkování

Chtěl bych poděkovat mému vedoucímu Ing. Petru Šillingovi za všechen čas, který mi věnoval, za jeho ochotu a velmi rychlou odezvu a hlavně jeho rady, které byly často k nezaplacení.

Obsah

Kapitola 1

Úvod

Už od dob renesance vznikla potřeba editovat poničené obrazy. Tato problematika se s vývojem výpočetní techniky přenesla i do digitální roviny, kde se pro tento problém vžilo označení *image inpainting*. Image inpainting odkazuje na proces vyplňování chybějících dat v určité oblasti vizuálního vstupu. Cílem je rekonstruovat chybějící části takovým způsobem, že vyplněná oblast nemůže být detekována nezávislým pozorovatelem, který nezná původní obraz. V průběhu let postupně vznikaly různé metody, které řeší tento problém, ať už převzaté z jiných oborů, nebo plně vytvořené pro image inpainting.

Cílem této práce je vytvořit aplikaci pro vizualizaci a porovnávání výsledků modelů pro image inpainting. Spolu s tím je cílem natrénovat alespoň dva modely. Byl vybrán jeden zástupce generativních inpainting modelů, na kterém budou prováděny hlavní experimenty ve snaze docílit nejlepšího výsledku a jeden zástupce ryze konvolučních modelů, který slouží pro srovnání výsledků. Oba tyto modely dosahují nejlepších výsledků oproti stejně typovým architekturám.

Nejprve byla vybrána vhodná datová sada, která se využije pro trénování zvolených sítí. Data z trénovací datové sady byla argumentována pro lepší výsledky. Všem obrazům a maskám byla následně upravena velikost na 512×512. Kromě trénovacích dat bylo ještě potřeba vyřešit data pro validaci. Jednou z možností bylo vyjmout z trénovací datové sady určitý výřez a ten následně použít pro validaci, ale nakonec byla použita odlišná datová sada, která je svým kontextem podobná trénovací sadě a díky tomu se tak mohou všechny obrazy použít pro trénování.

Proběhlo hned několik experimentů, kdy byly provedeny úpravy na generativním modelu ve snaze odhalit, jak moc účinné jsou AOT bloky a jaký bude mít vliv zmenšení jejich počtu na výsledný obraz (jak moc bude účinný při změně mohutnosti generátoru). Tyto experimenty byly následně validovány na validační datové sadě a to jak kvalitativním vyhodnocením, tak i kvantitativním vyhodnocením za použití obvyklých metrik. Vyhodnocením všech výsledků se dospělo k závěru, že vytrénovaná síť si velice dobře poradí s přírodní scenérií s hodnotou MAE 0.0103. I modely s menší mohutností dosahují téměř totožné hodnoty, ale ukazuje se, že čím méně se použije bloků, tím více klesá hodnota FID a souběžně s tím i vizuální kvalita obrazu. AOT-GAN také překonává model pconv téměř ve všech metrikách i po vizuálním srovnání.

Celá kapitola 2 popisuje důležité pojmy z oblasti doplnění chybějící části obrazu a významné metody z hlediska postupného vývoje v této oblasti. Na tuto kapitolu navazuje kapitola 3. Zabývá se popisem návrhu aplikace, popisuje její funkcionalitu, navržené rozhraní a práci s vybranými modely pro image inpainting. Navazuje na ni kapitola 4, která popisuje aplikaci z hlediska implementace. Závěrečná kapitola 5 shrnuje dosažené výsledky

s hlavním zaměřením na kvantitativní a kvalitativní zhodnocení natrénovaných sítích. Jsou zde popsány jak číselně, tak vizuálně, kvality výsledných modelů. V příloze A se nachází popis obsahu přiloženého paměťového média a v příloze B plakát, který jednoduše shrnuje tuto práci.

Kapitola 2

Teoretická část

Tato kapitola pojednává o teoretickém základu potřebnému k pochopení této práce, jejímž hlavním tématem je *image inpainting*. Jedná se o metodu modifikace obrazu, jejímž cílem je vyplnit předem stanovenou plochu v obraze tak, aby tyto modifikace náhodný pozorovatel nedokázal rozpoznat. Více informací se nachází v sekci 2.1, která podrobně přibližuje image inpainting jako takový, popisuje vývoj v této oblasti a načrtává problémy, které je třeba v dnešní době řešit.

Následující sekce v této kapitole již pojednávají o různých typech řešení tohoto problému. Začíná se od nejstarších a nejjednodušších až po *state-of-the-art* modely. Sekce 2.2 popisuje tradiční metody pro image inpainting. Tradiční metody nevyužívají neuronové sítě, ale přesto je vhodné je zmínit z hlediska jejich historického významu. Jedná se totiž o první algoritmy, jejichž hlavním účelem bylo doplnit určitou část obrazu.

Sekcí 2.3 již začínají moderní metody využívající hluboké učení. V této sekci je popsán důležitý teoretický základ neuronových sítí ve spojitosti s problematikou doplnění obrazu. Je zde vysvětlena funkce neuronu a velký důraz je kladen na popis trénování těchto neuronových sítí za využití datových sad. Tento obecný úvod je následován jednotlivými architekturami sítí pro image inpainting. Sekce 2.4 popisuje konvoluční sítě, jejich fungování a význam. Konvoluce a konvoluční bloky prostupují i velkou spoustou moderních metod, a proto jsou zde řádně vysvětleny. Detailněji je popsán model *pconv* [15], postavený na parciálních konvolucích, sloužící jako referenční model pro srovnání výsledků této práce. Sekce 2.5 zase prozkoumává generativní sítě, s důrazem na *AOT-GAN* [32], sloužící jako hlavní model této práce. Je zde vysvětlena jejich funkce a odlišnost oproti tradičním konvolučním sítím. Na závěr sekce 2.6 představí nejnovější state-of-the-art modely, které dosahují nejlepších výsledků a nastiňují tak budoucí pokrok.

2.1 Image inpainting

Image inpainting odkazuje na proces vyplňování chybějících dat v určité oblasti vizuálního vstupu. Cílem tohoto procesu je rekonstruovat chybějící části či poškozený obraz takovým způsobem, že vyplněná oblast nemůže být detekována nezávislým pozorovatelem, který nezná původní obraz. [8]

Již v minulosti se objevila potřeba řešit tento problém. Z počátku se jednalo o úpravu poškozených částí obrazu. Tento problém se dal řešit pouze ručně pomocí manuálního dokreslování (nebo též retušování). Později, se vznikem filmu a fotografie, se objevily nové výzvy. Vznikla potřeba umazávat celé objekty ze stále ještě fyzických snímků (např. uma-

zání politických nepřátel [13]). Začala se též využívat technika slepování částí dvou či vícero fotek dohromady.

Vznik a rozvoj výpočetní techniky přenesl tento problém do digitální roviny. Nejprve bylo nutné stále ještě manuálního zacházení a lidské zručnosti, ale pokrok šel kupředu. Postupem času začali vznikat algoritmy řešící právě tento problém, takže již není potřeba ruční práce člověka, ale stačí nám výpočetní stroj s vhodným algoritmem, který už problém vyřeší za nás.

Většina těchto problémů spadá do kategorie odstranění nějakého objektu z obrazu. Je nutné vymazat objekt či část obrazu a zaplnit ji tak, aby odpovídala reálnému obrazu, jako by v něm umazaná část vůbec nebyla. V praxi se tyto modely a na nich postavené nástroje využívají k odstranění vodoznaků, škrábanců či šumu, nebo k obnově poničených digitalizovaných fotografií [2]. S těmito operacemi nad šířkově malými ornamenty si poradí i jednodušší modely. Při pokusu o odstranění větších objektů ze scény, jako jsou například lidé či nábytek, zabírající proporčně větší část obrazu, ovšem mohou jednodušší modely selhat. Tím daly potřebě vzniknout mnohem robustnějším a sofistikovanějším modelům, které si poradí i s velkými objekty.

V dnešní době je také velice populární generovat naprosto nový obsah či vkládat určité objekty na zvolené místo, k čemuž se využívají modely na bázi transformerů [6, 14, 29] a difuzní modely [20, 17] (více v sekci 2.6). Řeší úplně stejný problém, zaplnění určitého místa v obraze, ale zároveň generují úplně nové objekty, které v původním obraze vůbec nebyly. Kromě zábavního aspektu či tvorbě reklam je takové generování vhodné též pro tvorbu návrhů a prototypů.

Aby tyto nástroje mohly vzniknout, musel image inpainting projít značným vývojem, jehož počátky sahají k tradičním metodám, které blíže popisuje sekce 2.2.

2.2 Tradiční metody

První metody pro image inpainting ještě nepracovali s konceptem hlubokého učení. Vznikaly již na počátku tisíciletí, kdy za zmíňku stojí práce Efrose et al. [7], kteří již roku 1999 přišli s z dnešního pohledu vcelku primitivním modelem. Tyto algoritmy se dají rozdělit na dvě podskupiny podle způsobu práce s pixely v obraze.

Diffusion-based (někdy též *propagation-based*) algoritmy využívají informace ze sousedních pixelů. Informaci z ohraničených mezí postupně propagují, než zaplní celou masku. Díky tomu jsou vhodné pro jednoduché problémy, jako je doplnění části pozadí, odstranění malých objektů, či na odstranění šumu či velikostně malých oděrek. Bertalmio et al. [2] byli jedni z prvních průkopníků těchto algoritmů. Jejich algoritmus doplňoval pixely z okolí po směru izofotů[1].

Popsaná funkcionalita této skupiny algoritmů má však jeden velký problém – vůbec nepracují s celkovou strukturou obrazu. Při maskách větších velikostí dochází k deformacím a defektům. Problémy též nastávají, pokud okolní pixely nenesou vhodné informace, což se děje u jakýchkoliv nerepetivních textur či u složitých vzorků. Neporadí si tak s problémy, ve kterých je potřeba využít kontext v komplexních scénách.

Na tento problém navázaly *patch-based* (někdy též *examplar-based* či *search-based*) algoritmy. Tyto algoritmy hledají potřebné informace v celé nezamaskované části obrazu. Snaží se vždy vyhledat úryvky z obrazu podobné místu, které se má zaplnit. Pixely z tohoto úryvku poté nakopírují na prázdné místo. Díky tomu jsou vhodné pro scény s velmi repe-

[1] Čáry spojující místa se stejnou intenzitou světla či v této situaci se stejnou intenzitou jasu. [26]

tetivními vzory. Neporadí si však s obrazy obsahující komplexní vzory, protože nerozumí sémantickým informacím v obraze. Pro nalezení více informací na téma fungování patch-based algoritmů, je možné nahlédnout do práce Barnese et al. [1], kteří přišli s jedním z prvních algoritmů tohoto typu.

(a) Ground truth (b) Zamaskovaný obraz (c) Patch-based algoritmus

Obrázek 2.1: Zde je poměrně dobře vidět síla a zároveň problém patch-based algoritmů. Hledají úryvky, které se nejvíce hodí na chybějící místo. Nedokáží si však domyslet nic, co není v nezamaskované části. Proto při nedostatku informací dochází k problémům jako na obrázku (c). Převzato z [33].

Bylo tak nutné vytvořit modely, které dokáží pochopit kontext obrazu a předpovědět tak obsah jejich umazané části. Vznikla tak nová moderní skupina metod pro image inpainting využívající hluboké učení. Tyto sítě jsou schopné se učit vzorce, díky čemuž si dokáží „domyslet“ i části obrazu, co se v něm původně nenachází. Těmito sítěmi se blíže zabývá sekce 2.3.

2.3 Metody využívající hluboké učení

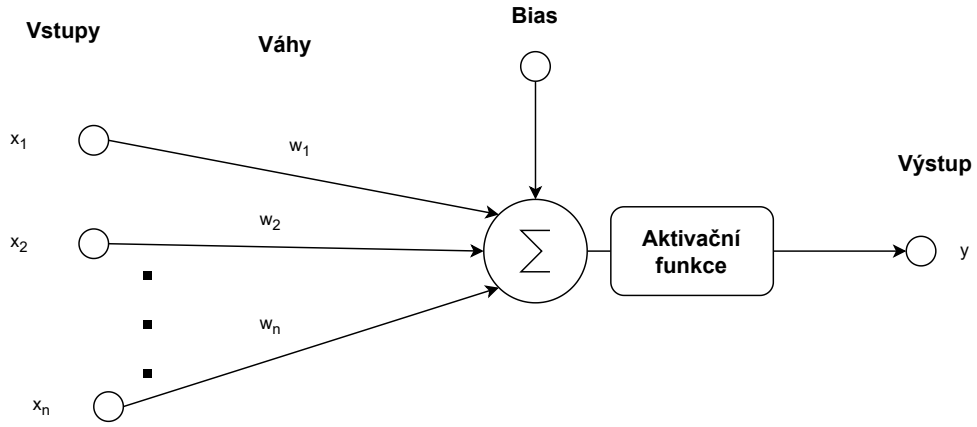
Jedná se o metody, které řeší stejný problém, jako již zmíněné algoritmy, ale řeší jej dosti odlišně. Oproti tradičním metodám je nutné dané modely nejdříve vytrénovat. Jsou využívány neuronové sítě a jejich princip hlubokého učení. Díky nim jsou schopné rozpoznat strukturu a kontext jednotlivých částí obrazu, tyto vzorce se naučit a díky těmto „zkušenostem“ dosáhnout mnohonásobně lepšího výsledku než tradiční metody.

Neuronová síť

Neuronová síť je model postavený na biologickém principu propojenosti neuronů v mozku živých tvorů. Snaží se napodobit fungování lidského mozku z hlediska struktury a konceptu učení a uchovávání si informací.

Tyto sítě se skládají z neuronů (korespondující s neuronem v mozku). Snaží se postupně učit a upravovat váhy jednotlivých neuronů (neuron je podrobně popsán na obrázku 2.2) a dospět tak k nějakému závěru. Cílem trénování těchto sítí je naučit síť pracovat tak, aby závěr, ke kterému dospěje, byl správný.

Ze začátku má každý neuron stanovené náhodné hodnoty vah ovlivňující hodnoty vstupů (udávají jejich „vliv“). Velmi pravděpodobně váhy vstupů u jednotlivých neuronů nejsou



Obrázek 2.2: x_1 –x_n označují vstupní data, w_1 – w_n zase váhy, které tyto vstupy ovlivňují. Kromě vstupních dat pak do neuronu vstupuje též tzv. *bias*, který slouží pro lehké odchýlení výstupu a každý neuron má svůj vlastní. Všechny tyto vstupy se sečtou, načež je výsledek postoupen aktivační funkci, která jej upraví na konečný výsledek y. Ten dále pokračuje sítí.

správně nastavené, a proto je nutné neuronové sítě vytrénovat. Kromě vstupů a jejich vah má ještě každý neuron určitou odchylku (anglicky *bias*), která lehce odchýlí výstup neuronu.

Výstup neuronu je vypočítán součtem všech vstupů a biasu. Tento výsledek se potě transformuje aktivační funkcí pomocí aktivační funkce. Aktivační funkce definuje výstup neuronu při zadání sady vstupů. Upravuje výstupní hodnoty (například je přetransformuje do pouze nezáporných čísel). Aktivační funkce tak určuje, jestli se daný neuron „aktivuje“. Tato informace pak proudí dál do dalších napojených neuronů.

Neurony jsou shlukovány do jednotlivých vrstev, které jsou mezi sebou propojeny. Výstupem neuronové sítě, vyobrazené na obrázku 2.3, je řešení úlohy, které může mít spoustu podob (v našem případě obraz doplněný pixely na místo díry).

Více informací o neuronových sítích, jejich principu a využití lze získat např. v [9], zaměřující se více na matematický aspekt neuronových sítí, nebo v [18] pro více praktické informace.

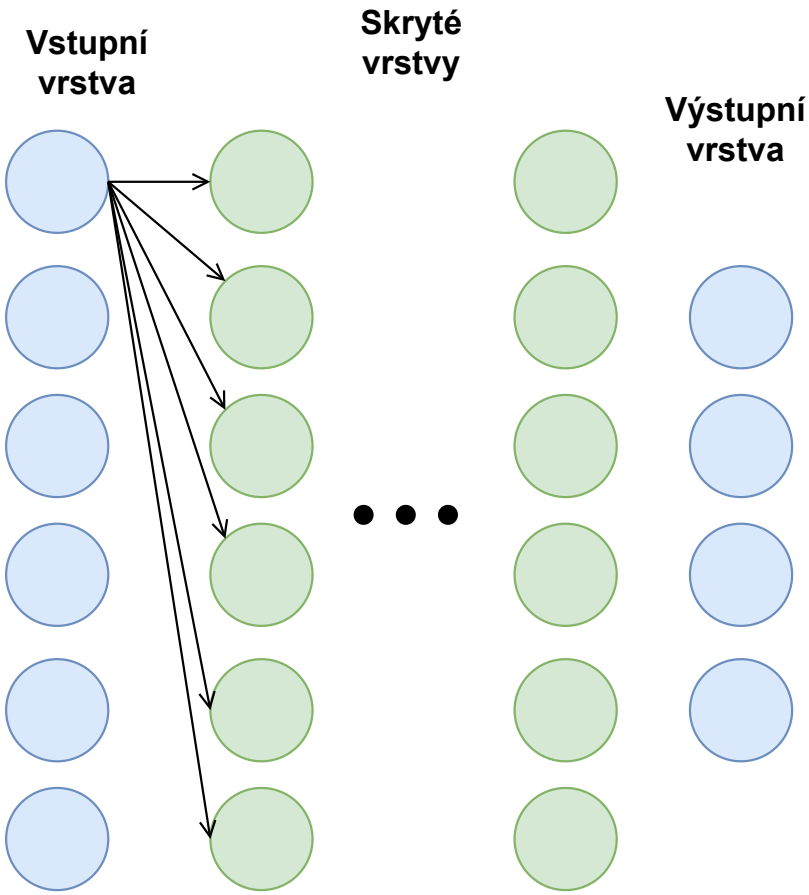
Trénování

Trénování slouží ke změně vah neuronů. Váhy určují výstup sítě, který dostaneme po vložení určitého vstupu. Ze začátku se váhy nastaví na počáteční hodnoty. Výsledky by však neodpovídaly našemu očekávanému výsledku, proto přichází na řadu trénování. Trénováním je třeba upravit váhy každého neuronu tak, abychom pro vložený vstup dosáhli optimálního výsledku.

Váhy se iterativně aktualizují jednotlivými průchody. Jedná se o optimalizační problém jehož cílem je minimalizovat ztrátovou funkci (anglicky *loss function*), čímž se teoreticky čím dál tím víc blížíme ideálnímu výsledku.

Jednou z hlavních metod, která se pro optimalizaci sítě využívá, je tzv. *backpropagation*. Jedná se o algoritmus zpětné propagace chyby, kdy se spočítá gradient ztrátové funkce. Následně se provede gradientní sestup[2] ve směru od poslední vrstvy po první. Při-

[2]Gradientní sestup (anglicky *gradient descent*) je iterativní optimalizační algoritmus prvního řádu pro nalezení lokálního minima diferencovatelné funkce.



Obrázek 2.3: Neurony se shlukují do vrstev, které dohromady tvoří síť. Každá síť má vždy vstupní vrstvu, která přijímá patřičné vstupy (v našem případě zamaskovaný obraz), a výstupní vrstvu, ze které vychází konečný výsledek (v našem případě doplněný obraz). Mezi těmito vrstvami se nacházejí skryté vrstvy. V nich probíhá proces zpracování vstupu na výstup a jejich počet udává šířku modelu.

tom se upraví vstupní váhy jednotlivých neuronů, aby se zmenšila výsledná chyba. Pro více informací o této metodě je možné nahlédnout do práce Rumelharta et al. [22].

Důležitým aspektem trénování jsou datové sady. Jedná se o sady obrazů, které chceme využít pro optimalizaci modelu. Sada se rozdělí na menší vzorky (anglicky *batch*), jejichž velikost je optimálně rovna nějakému násobku dvou, což vede k efektivnějšímu využití procesoru. Tyto obrazy společně s maskou postupně prochází celým modelem. Na výstupu už jsou díry po masce vyplněné, a tyto výsledné obrazy porovnáme s původními *ground truth* obrazy. Vypočítáním rozdílu obou obrazů získáme ztrátovou funkci, která se využívá k optimalizaci sítě. Podle výsledku zpětně upravíme váhy modelu, z datové sady se vybere další vzorek a iterativně se pokračuje. Jedna epocha trénování uběhne v případě, kdy sítí projde celá datová sada. Tyto epochy je nutné provádět do té doby, než je splněn předem definovaný trénovací cíl (například určitý počet epoch či pokud klesne ztrátová funkce pod určitou hodnotu).

Datové sady pro metody na image inpainting obrazu obsahují jednotky až několik desítek milionů obrazů. Datové sady mohou být obecné, které obsahují obrazy zobrazující všechny možné věci, od přírodních scenérií, přes stavby až po portréty lidí. Výsledný vytré-

novaný model by si pak v ideálním světě měl poradit s každým obrazem obsahující libovolný kontext. K tomu je však potřeba spousta dat a úměrně tomu též čas na trénování.

Opakem jsou pak datové sady specifické, které se zaměřují jen na určitý prvek. Příkladem může být datová sada CelebA [16], obsahující pouze lidské obličeje. Tyto datové sady dosahují obvykle lepších výsledků na jeden určitý problém, ale naopak horších výsledků při práci s typově odlišnými obrazy. Výhodou je také to, že není potřeba tolik dat k trénování.

Pro inpainting je nutné k trénování dodat kromě hrubých dat ještě masky. Představují reprezentaci míst v obraze, které byly umazány a je nutné je zaplnit. Masky oproti obrazům s daty obsahují pouze černou a bílou barvu, které korespondují s pravdivostními hodnotami 0 a 1 a odpovídají místům, které je nutné vyplnit, respektive místům nesoucím původní informaci.

Ze začátku se jako maska využíval čtverec nacházející se přesně uprostřed obrazu (které použili např. Burlin et al. [3]), jejíž vytvoření bylo velmi rychlé. Později se začaly využívat generované nepravidelné masky, které lépe odpovídají reálnému použití (využit v práci Liu et al. [15]). Díky tomu si sítě lépe poradí s reálnými daty, kdy se potřebujeme zbavit objektu netriviálního tvaru nacházejícího se i mimo střed scény. Proto se tyto masky snaží napodobit tvary různých objektů, jako kdyby je nakreslila lidská ruka.

Při trénování je samozřejmě ještě nutné mít neuronovou síť, kterou je třeba vytrénovat. Existují různě typy architektur neuronových sítí, které se snaží vyřešit problém doplnění obrazu. Nejjednodušší jsou konvoluční sítě sestavené z bloků pracujících na bázi konvoluce. Podrobně jsou popsány v sekci 2.4. Na ně navázaly GAN sítě, které se prakticky skládají ze dvou sítí, generátoru a diskriminátoru, které spolu soupeří ve snaze vytvořit neodhalitelný syntetizovaný obraz, respektive tento syntetizovaný obraz odhalit. Detailně jsou popsány v sekci 2.5. Snaha oprostit se od konvolučních bloků a zároveň více pracovat s kontextem obrazu vedla ke hledání nových cest, čímž se začali využívat transformery. Následovaly difuzní modely generující obraz z *noise* vektoru a následným iterativním odšumováním. Tyto architektury aktuálně představují to nejlepší na poli metod pro image inpainting a jsou popsány v sekci 2.6.

2.4 Tradiční konvoluční sítě pro image inpainting

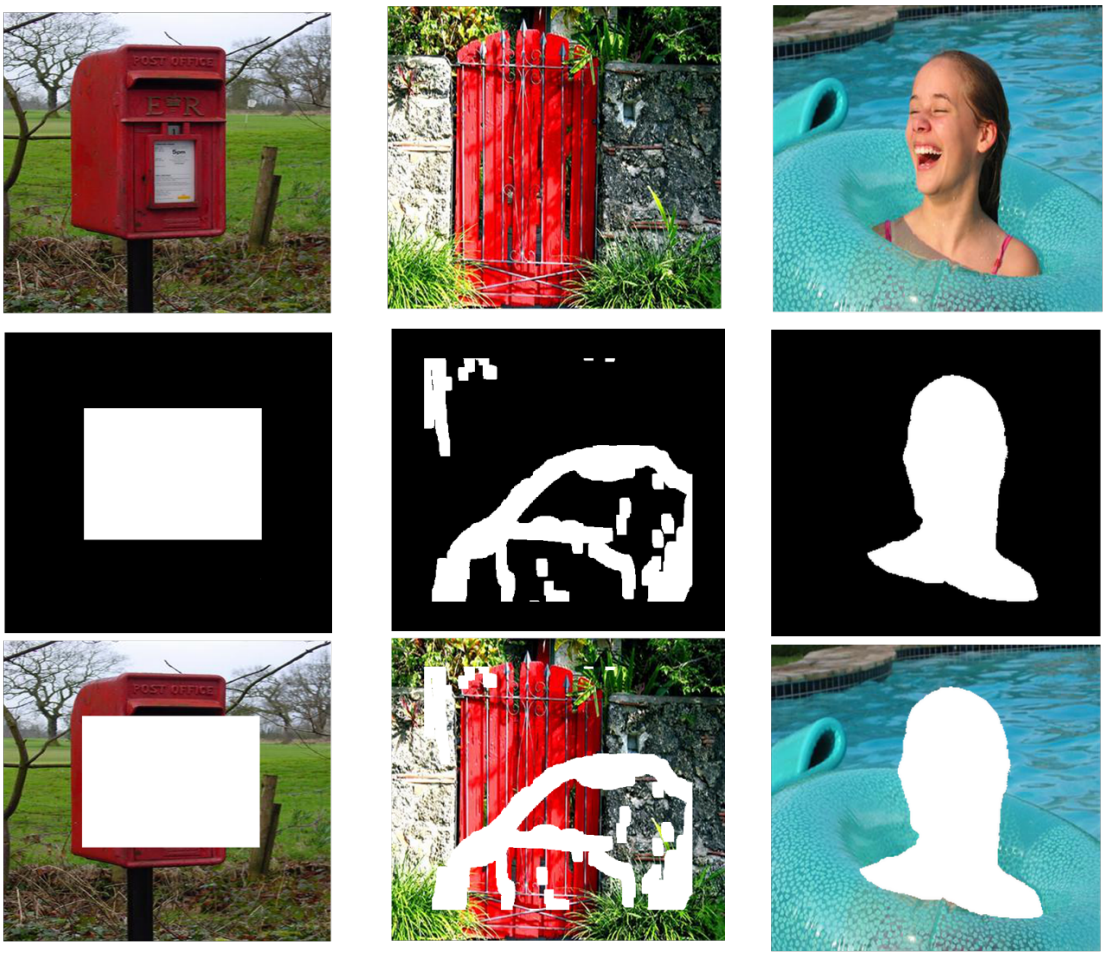
Konvoluční neuronové sítě jsou označovány zkratkou CNN (z anglického *convolutional neural networks*). Svůj název získaly díky tomu, že jejich struktura obsahuje konvoluční vrstvy, skládající se z konvolučních bloků, které provádějí transformaci vstupu na výstup s využitím konvoluce.

Tyto sítě jsou velice vhodné pro rozpoznávání řeči, autonomní rozhodování či klasifikaci vzorců v obraze. Konvoluční bloky se hodí i při řešení našeho problému. Jsou nedílnou součástí většiny modelů sloužící k vyplnění obrazu a jedná se o první architekturu na bázi hlubokého učení pro image inpainting.

Následující sekce popisuje principy fungování konvolučních sítí, na kterou navazuje sekce přibližující fungování vybraného modelu konvoluční sítě, který tyto konvoluční bloky nahrazuje bloky využívající konvoluci parciální. Tento model je v práci využit jako referenční pro porovnání výsledků.

Princip konvoluční sítě

Konvoluční sítě pro image inpainting pracují výhradně s obrazovým vstupem, který má 3 dimenze, a to výšku, šířku a hloubku (popisující počet barevných kanálů vstupu).



(a) Čtvercová maska (b) Nepravidelná maska (c) Maska objektu

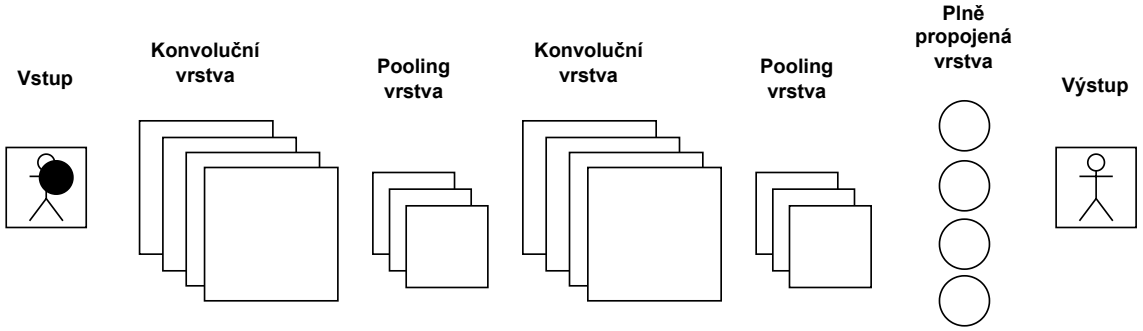
Obrázek 2.4: Ukázka různých typů masek. Čtvercová maska (a) se u novějších modelů již zpravidla nepoužívá. Mnohem častěji se dnes využívají nepravidelné generované masky (b) či masky ve tvaru reálných objektů (c). Obrazy převzaty z datové sady MSRA10K [4].

Skládají se ze tří typů vrstev. Z konvolučních vrstev, *pooling* vrstev a plně propojených vstev. Tyto vrstvy jsou volně poskládány a propojeny za vzniku konvoluční neuronové sítě.

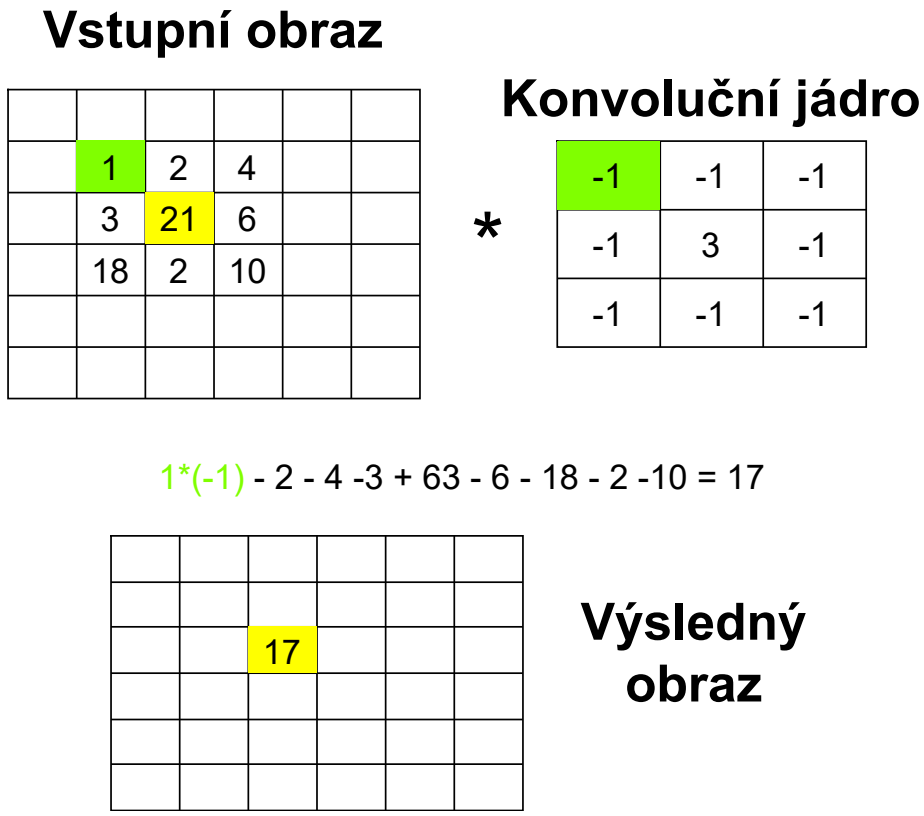
Vstupní vrstva obsahuje obrázek rozebraný na pixely. Tyto data prochází přes konvoluční vrstvu, ve kterých probíhá konvoluce. Pro dvou dimenzionální obraz se dá konvoluce $S(i,j)$ matematicky popsat jako:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n),$$

kde K označuje dvou dimenzionální konvoluční jádro a I zase zvolenou část obrazu. i a j jsou pak indexy výsledné konvoluce a m, n jsou zase indexy obrazu a jádra.[9] V našem případě to znamená, že vezmeme konvoluční jádro, například o velikosti 3×3, a přiložíme jej k obrazu. Pixely, co se překrývají, se mezi sebou vynásobí (viz obrázek 2.6). Tyto výsledky se sečtou a dají nám za vznik hodnotu jednoho pixelu ve výsledném obraze. Iterativně se postupuje pro celý obraz v závislosti na požadované velikosti výstupu.



Obrázek 2.5: Ukázka struktury jednoduché konvoluční sítě. Vstupní obraz projde několika konvolučními a pooling vrstvami. V nich dochází ke zpracování informací z obrazu a úpravám velikosti vstupu. Následně se všechny informace spojí v plně propojených vrstvách, kde každý vstup je napojen na všechny výstupy vrstvy předchozí. V nich dochází ke klasifikaci obrazu, či v našem případě ke zkompletování výsledného obrazu, který se objeví na vstupu.



Obrázek 2.6: Ukázka výpočtu konvoluce pro jeden pixel výstupu. Vezmeme region, který velikostně odpovídá konvolučnímu jádru. Toto jádro poté přiložíme na daný region a hodnotu každého pixelu daného regionu vynásobíme odpovídající hodnotou pixelu v konvolučním jádře. Všechny tyto výsledky následně sečteme a dospějeme tak k hodnotě jednoho výstupního pixelu. Poté se posuneme o předem stanovený krok ve zdrojovém obraze a tuto metodu poté iterativně opakujeme do té doby, než zaplníme celý výsledný obraz požadované velikosti.

Jak je ukázané na obrázku 2.7, neuron pracuje pouze s omezeným regionem (anglicky *receptive field*) vstupního obrazu, čímž se zamezí až příliš velkému zatížení pro jeden neuron. Může se tak také více věnovat jen dané části.

Čím více neuronů se nachází v této vrstvě, tím více vzorů je síť schopna rozlišit, ale zase se zvýší výpočetní doba. Počáteční vrstvy rozpoznávají jednoduché tvary, jako jsou čáry a hrany, hloubější si poradí i se stále složitějšími a abstraktnějšími vzorci.

Důležitý je též zvolený krok (anglicky *stride*) pro pole působnosti neuronů (o kolik pixelů se posune pole působnosti daného neuronu oproti jeho sousedovi). Menší krok znamená větší překrytí a naopak.

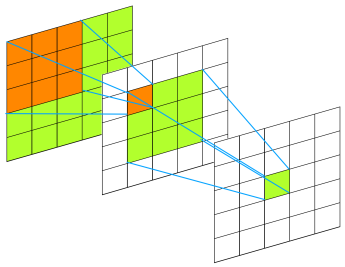
Pro upravení výstupních parametrů se využívá ještě tzv. *zero-padding*, kdy se za hranice obrazu přidá určitý počet nul (černých pixelů), se kterými se pak počítá při výpočtech okrajových oblastí.

Všechny tyto parametry však musí být vhodně nastavené, protože modifikujeme prostorové parametry na výstupu. Musí splňovat následující formuli:

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1,$$

kde n_{out} je počet výstupních parametrů, n_{in} je počet vstupních parametrů, k je velikost jádra, p je velikost okraje a s je velikost kroku. Všechny tyto parametry musíme správně nastavit, aby se výsledek rovnal požadované velikosti. [19]

Pooling vrstva zredukuje počet dat vstupu (jejich parametry). Zmenší se tak výpočetní náročnost sítě, dochází však ke ztrátě určitých dat. Plně propojené vrstvy na konci sítě provádí klasifikaci. Neurony v této vrstvě jsou plně propojeny se všemi výstupy předchozí vrstvy, takže každý výstup je ovlivněn všemi vstupy. Sestavují výsledný obraz, který se objeví na výstupu.



Obrázek 2.7: Ukázka pole působnosti neuronu a postupné propagace informace mezi jednotlivými neurony. Pixely v každé hlubší vrstvě zabírají větší plochu, stále však pracují jen s přesně daným počtem pixelů z předchozí vrstvy.

Ve světě dokreslování obrazů konvoluční sítě již nedosahují takových výsledků jako nejlepší modely v oboru. Problémy mají například při práci s rozměrově velkými maskami. Proto je vhodné konvoluční sítě mírně poupravit či k nim přidat komponenty, které následně zlepší výsledný výstup. Spousta neuronových sítí zabývající se doplněním obrazu v sobě nějaké konvoluční bloky obsahuje. Například v této práci hojně využívaná AOT-GAN [32] síť obsahuje mimo jiné konvoluční bloky ve struktuře obou svých částech (více popsané v sekci 2.5).

Parciální konvoluce

Jako referenční síť spadající do této kategorie byla vybrána síť využívající parciální konvoluci. Její architektura je postavená na síti U-Net[3] struktuře s tím, že všechny konvoluční bloky jsou nahrazeny parciální konvolucí. Více informací o parciálních konvolucích a jejich fungování lze získat přímo z práce Liu et al. [15].

Po každé parciální konvoluci se vždy aktualizuje maska. Pokud byla konvoluce schopna určit výstup podle alespoň jednoho platného vstupu, tak toto místo označíme jako validní. Mějme binární masku M, pak je výsledek vyjádřen jako:

$$m' = \begin{cases} 1, & \text{když } sum(M) > 0 \\ 0, & \text{jinak} \end{cases}$$

Na konci tak jakákoliv výsledná maska m' bude obsahovat samé jedničky v případě alespoň jednoho validního pixelu na vstupu, což znamená úplné vyplnění. Zajistí se tak také to, že se konvoluce nebude provádět na nevalidních pixelech.

Konvoluční sítě představují základ v oborech pracujících s obrazovým vstupem. Dají se poměrně jednoduše vytrénovat a svou jednoduchostí jsou snadné na pochopení. Na konvoluční sítě navázaly GAN sítě, které se prakticky skládají ze dvou oddělených sítí. Pořád ale využívají poznatky získané z vývoje konvolučních sítí. Více informací o GAN sítích je zmíněno v další sekci 2.5.

2.5 GAN sítě

GAN je zkratka pocházející z anglického *generative adversarial network* (česky generativní soupeřící síť). Jedná se o architekturu neuronové sítě, jejíž součástí jsou dva modely, generátor a diskriminátor.

Generátor za vstup přijímá náhodně vygenerovaný vektor šumu (anglicky *noise vector*). Z něj se pak pomocí různých operací vygenerují falešná data, kterými se generátor snaží co nejlépe napodobit data reálná. Diskriminátor dostane původní pravý obraz a falešný vygenerovaný obraz. Následně se snaží odhalit, který byl vygenerovaný generátorem a který je pravý.

Smyslem celé neuronové sítě je to, že spolu „soupeří“ diskriminátor s generátorem. Generátor G se snaží vytvořit falešný obraz tak, aby jej diskriminátor neodhalil. Naopak diskriminátor D se snaží zlepšit v odhalování falešného obrazu. Svým pojetím odpovídá minimax hře pro dva hráče $V(D, G)$, kdy se každý hráč snaží maximalizovat své zisky a naopak minimalizovat zisky toho druhého. Ta se dá matematicky popsat:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))],$$

kde p_z je distribuce šumu, p_{data} distribuce dat. $D(x)$ představuje pravděpodobnost, že x pochází z reálných dat, a ne z p_z, a $G(z)$ zase generaci umělých dat z šumu. Diskriminátor se trénuje pro zvětšení pravděpodobnosti, že správně určí, odkud vstup pochází. Generátor se naopak trénuje pro zmenšení $log(1 - D(G(z)))$. [10].

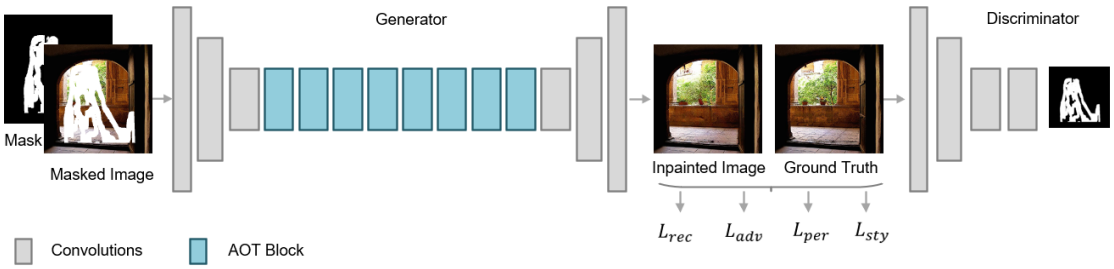
[3] Plně konvoluční neuronová síť vyvinutá pro segmentaci biomedicínského obrazu. Je to hierarchická síť postavená do tvaru písmene U, kdy se obraz nejprve postupně zmenšuje, načež se zase postupně zvětší do původní velikosti. [21].

Po mnoha trénovacích iterací síť teoreticky dojde do bodu, kdy už se ani jeden z nich nemůže dále zlepšovat. Výsledná p_z se pak rovná p_{data} a diskriminátor nedokáže rozlišit mezi oběma distribucemi.

K aktualizaci sítě se opět využívá metoda backpropagation. Je však nutné, aby se generátor i diskriminátor trénovaly společně. Nesmí nastat situace, kdy by se generátor trénoval až příliš, bez aktualizace diskriminátoru. Mohlo by dojít ke kolapsu modelu. Generátor pak generuje jen určitý úsek dat, kterým se mu podařilo obelstít diskriminátor. Z tohoto stavu je pak velice obtížné se dostat. Pro více informací o fungování generátoru a diskriminátoru je možné si přečíst práci Goodfellowa et al. [10].

AOT-GAN

Základní struktura GAN sítí se dá modifikovat. Zeng et al. [32], jejichž model je využíván v této práci, přichází s řadou změn. Standardní GAN neuronová síť generátoru je v jejich práci obohacena o tzv. *AOT* (anglická zkratka pro *aggregated contextual transformation*) bloky, které jsou schopné zachytit širší kontext a opakující se vzory pro zlepšení výsledku. Architekturu lze vidět na obrázku 2.8.



Obrázek 2.8: Architektura AOT-GAN sítě. Generátor má na vstupu a výstupu několik konvolučních vrstev, které se starají primárně o zpracování vstupu, respektive výstupu. Mezi nimi se nacházejí AOT bloky (v základní verzi modelu je jich 8) zachytávající kontext obrazu. Na výstupu se vypočítají ztrátové funkce (detailně jsou popsány v sekci Ztrátová funkce dále v textu), aktualizují se podle nich váhy a doplněný obraz se spolu s ground truth předá diskriminátoru, který odhaduje, které části z vložených obrazů jsou synteticky generované a které jsou reálné. Převzato z [32].

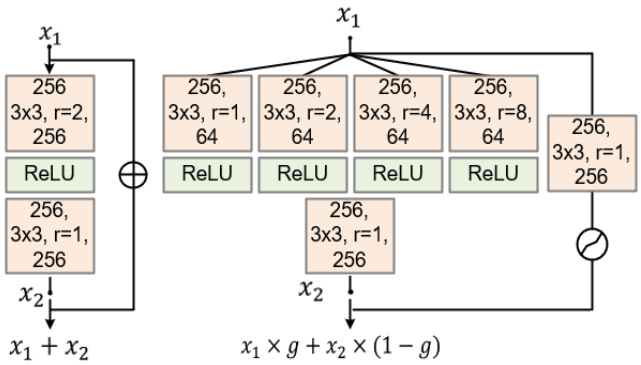
Generátor je poskládán z enkodéru, který je postavený ze standardních konvolučních bloků, sady AOT bloků a následně z dekodéru, který za pomoci transponovaných konvolučních sítí nakonec vygeneruje výsledný obraz.

AOT blok (vnitřní strukturu lze vidět na obrázku 2.9) je založen na strategii rozděl, transformuj a spoj. Nejprve se klasické konvoluční jádro rozdělí na vícero podjader, každé s menším počtem výstupních kanálů. Součtem se však výstupní kanály podjader musí znova rovnat našemu původnímu jádru. Například jádro pracující se 256 výstupními kanály rozdělíme na čtyři podjádra, každé s 64 výstupními kanály.

Každé podjádro pak provádí rozdílnou transformaci vstupu x_1 díky rozdílnému dilatačnímu koeficientu[4]. Použití většího koeficientu umožní podjádru pracovat s větší částí obrazu. Naopak použití menšího koeficientu umožní zaměření na lokální vzorky. Nakonec

[4]Dilatační koeficient určuje počet pixelů, které jsou přeskočeny při každém konvolučním kroku (standardní konvoluce má koeficient 1)

jsou všechny podjádra agregována pomocí konkatenace, následované standardní konvolucí pro propojení všech prvků.



Obrázek 2.9: Nalevo standardní konvoluční blok, vedle něj pro srovnání AOT blok. Na vstup přichází x_1, které obsahuje 256 vstupních kanálů. Každé podjádro má velikost konvolučního jádra 3x3 a počet výstupních kanálů 64, liší se však dilatačním koeficientem r. Všechna podjádra jsou pak spojena konkatenací následovaná standardní konvolucí pro propojení všech prvků. Převzato z [32].

Díky AOT blokům má daná GAN síť větší šanci zaznamenat více různých vzorků obrazu, od celkového kontextu obrazu po opakující se vzorce v lokálních částí, což obecně vede k mnohem kvalitnějším výsledkům. Síť tak jen podle kontextu v obrazu dokáže s vysokou přesností dokreslit i části umazaného objektu.

Další změnou je práce diskriminátoru. Nehodnotí celkový obraz. Místo toho obraz rozdělí na několik podčástí, které poté samostatně ohodnotí na škále od 0 do 1 podle toho, nakolik věří, že se jedná o syntetizovanou část či nikoli. Detailně je hodnocení disktriminátoru znázorněno v obrázku 2.10. Jen části, které jsou ohodnoceny jako syntetizované generátorem, jsou následně použity pro optimalizaci generátoru. Díky tomuto posílení diskriminátoru je generátor nucen generovat realističtější data.

Ztrátová funkce

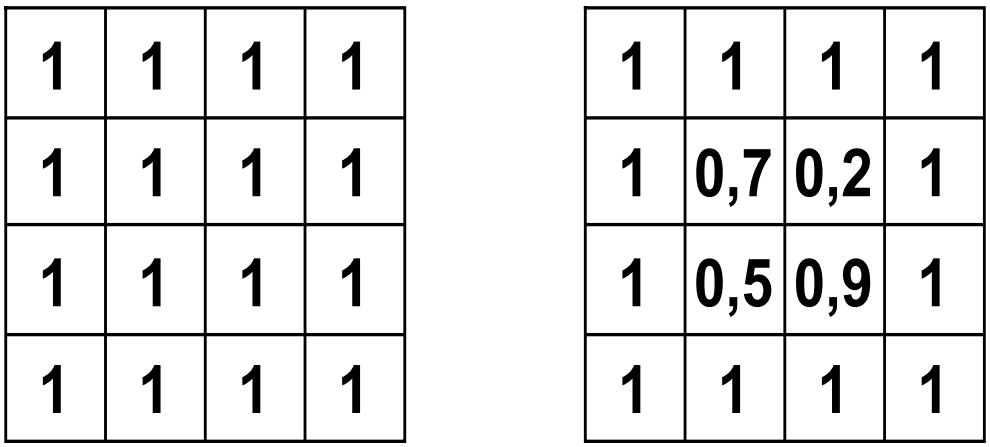
Optimalizace celé sítě je prováděna výpočtem ztrátové funkce a propagací této informace modelem. Hlavní ztrátová funkce je vypočítána součtem čtyř ztrátových funkcí, které se starají o zlepšení kvality práce generátoru s diskriminátorem. Jedná se o ztrátovou funkci generátoru, rekonstrukční, percepční a stylovou.

Ztrátová funkce pro AOT-GAN generátor určuje, jak dobře se daří generátoru přesvědčit diskriminátor, že jím generovaný obraz je pravý. Používají se pouze predikce syntetizovaných částí, což by mělo donutit generátor generovat kvalitnější data. Ztrátová funkce generátoru je definována jako:

$$L_{adv}^{G} = \mathbb{E}_{z \sim p_z}[(D(z) - 1)^2 \odot m],$$

kde m značí masku v binární podobě (kdy 1 označuje chybějící část), $\odot$ značí násobení na úrovni pixelů, z označuje vyplněný obraz a D značí funkci diskriminátoru.

Rekonstrukční ztrátová funkce se snaží docílit pixelové přesnosti generovaného obrazu. Zjednodušeně řečeno se jedná o rozdíl mezi reálným obrazem a obrazem generovaným. Její výpočet je definován jako:



(a) Typický diskriminátor

(b) AOT-GAN diskriminátor

Obrázek 2.10: Oproti původním GAN sítím nepracuje diskriminátor s celým obrazem, ale rozdělí si jej na několik částí. Až následně těmto částem přiřadí hodnotu od 0 do 1 podle toho, nakolik si myslí, že se jedná o reálná data.

$$L_{rec} = ||x - G(x \odot (1 - m), m||_1,$$

kde G značí generátor a x reálný obraz.

Percepční ztrátová funkce se snaží zmenšit na minimum vzdálenost mezi aktivačními mapami reálného a doplněného obrazu. Je definována jako:

$$L_{per} = \sum_i \frac{||\phi_i(x) - \phi_i(z)||_1}{N_i},$$

kde ϕ_i je aktivační mapa i-té vrstvy natrénované sítě (u této sítě autoři využili VGG19[5][24]) a N_i je počet elementů v ϕ.

Podobně pracuje i stylová ztrátová funkce, která se snaží zmenšit na minimum vzdálenost mezi Gramovými maticemi[6] hlubokých detailů reálného a doplněného obrazu. Je definována jako:

$$L_{sty} = \mathbb{E}_i[||\phi_i(x)^T \phi_i(x) - \phi_i(z)^T \phi_i(z)||_1].$$

Všechny tyto ztrátové funkce mají za cíl vylepšit konečný vizuální výsledek vyplnění obrazu. Celá síť se tak snaží dosáhnout co nejmenší celkové ztráty, který je definována následovně:

$$L = \lambda_{adv} L^G_{adv} + \lambda_{rec} L_{rec} + \lambda_{per} L_{per} + \lambda_{sty} L_{sty}.$$

Hodnoty $\lambda_{adv} = 0.01$, $\lambda_{rec} = 1$, $\lambda_{per} = 0.1$ a $\lambda_{sty} = 250$ pro váhy ztrátových funkcí byly autory empiricky zvoleny pro trénování. Díky těmto změnám dosahuje AOT-GAN model

[5]Konvoluční neuronová síť natrénovaná na ImageNet [5] datové sadě. Velice často se využívá pro měření ztrátové funkce.

[6]Matice skalárních součinů vektorů.

lepších výsledků něž většina konvolučních a GAN neuronových sítí pro image inpainting. [32] AOT-GAN sítě budou dále popsány v kapitole 3.3, neboť jsou klíčové pro tuto práci.

2.6 State of the art

Již zmíněné generativní modely (popsané v sekci 2.5) jsou poměrně pokročilou technologií, ale oproti nejnovějším modelům mají svoje limity. Přesto je vhodné zmínit některé modely, které, minimálně v oblasti generativních modelů, dosahují poměrně zajímavých výsledků.

Kromě již zmíněné AOT-GAN sítě [32] sem patří i další modely. Surovov et al. [27] použili ve struktuře generativních modelů rychlou Fourierovu transformaci. Yu et al. [31] zase využili tzv. *gated* konvoluci a Zhao et al. [33] modulované GAN, spojující obrazově podmíněnou a modulově nepodmíněnou architekturu GAN sítí.

Modely na bázi transformerů

Všechny předchozí modely zmíněné v této práci, pracující s konceptem hlubokého učení, byly postaveny na konvolučních blocích, nebo šlo o jejich vhodně upravené varianty. Naopak transformery se od nich snaží odprostit. Původně je vytvořili Vaswani et al. [28] jako nový způsob překladu mezi jazyky, ale díky své efektivnosti se začaly využívat i při práci s obrazovým vstupem.

Transformery zachycují kontext z různých částí obrazu. Tento kontext poté využívají k pochopení obrazu a vztahu mezi jednotlivými pixely. Díky tomu dokáží generovat realistické výsledky. Tyto modely dokáží vytvořit vícero různých výstupů, čehož docílí pomocí změny poměrů vah. Deng et al. [6] postavili svůj model na Taylorově expanzi. Wan et al. [29] využívají transformery pro prvotní rekonstrukci a následně konvoluční síť pro doplnění textur. Liao et al. [14] zase využívají obrazové reference pro lepší generaci výsledku.

Difuzní modely

Difuzní modely [25] jsou nejnovějším přírůstkem v oblasti syntetizace obrazu. Oproti GAN sítím netrpí na tréninkové nestability. Jsou postavené na bázi autoenkodérů [12] pro odstranění šumu, které jsou hierarchicky seřazené. Jejich cílem je se naučit rozdělení dat $f(x)$ pomocí postupného iterativního odšumování obrazu. Využívají přitom Markovův řetězec[7] pro přechod z jedné distribuce dat do druhé. [25] Postupným odšumováním se tak generuje výsledný obraz.

Tyto modely dokáží generovat naprosto nové objekty, které se nenachází nikde ve scéně. Kromě obrazových vstupů (vstupní obraz a maska) dokáží na výsledný obraz převést též vložený text. Rombach et al. [20] upravují proces trénování a oprošťuji se od architektury plně založené na transformerech. Lugmayr et al. [17] zase postavili difuzní architekturu na pravděpodobnostních modelech.

[7] Popisuje pravděpodobnost přechodu z jednoho stavu do druhého jen na základě stávajícího stavu. [25]

Kapitola 3

Návrh

Tato kapitola pojednává o návrhu výsledné aplikace. Cílem výsledné aplikace je vytvořit nástroj na snadné testování a porovnávání metod pro image inpainting na bázi hlubokého učení. Zároveň s tím bude natrénován AOT-GAN model [32] jako hlavní model práce a také konvoluční model na bázi parciální konvoluce [15], který bude sloužit jako model referenční.

Před návrhem jakékoliv aplikace je důležité nejprve si určit nejdůležitější vlastnosti dané aplikace. Typickým uživatelem je výzkumník či student v oblasti hlubokého učení, nebo někdo, kdo má alespoň povědomí o základních informacích z této oblasti a chce si vyzkoušet nějakou inpainting metodu. Aplikace by měla sloužit k co nejintuitivnějšímu testování metod pro image inpainting. Nutné je tedy mít možnost vložit vlastní obraz, na kterém lze jednoduše vyznačit masku. Nejlépe jedním kliknutím následně provést inpainting ve vyznačené části a okamžitě vidět výsledek. Kromě toho uživatel může chtít změnit model pro image inpainting a upravit jeho parametry. Uživatel též může chtít porovnat výsledky dvou různých modelů. Tento výsledek je pak nutné přehledně zobrazit s potřebným popisem. Kromě toho je vhodné uživateli poskytnout základní operace s obrazem, jako je přiblížení a pohyb po scéně. Pro zkoumání vlastností jednotlivých modelů je dobré též přidat několik nástrojů pro úpravu celého obrazu, jako je vyznačování hran pro zkoumání práce s hranami či převod do šedotónového obrazu pro zkoumání vlastností při generaci z černobílého obrazu.

Následující sekce 3.1 nejprve popisuje celkovou funkcionalitu aplikace. Začíná popisem základní funkcionality, bez které by aplikace nemohla plnit svůj účel a nebyla by použitelná. Následně popis sekundární funkcionality umožňující různé úpravy vloženého obrazu.

Dále navazuje sekce 3.2, která popisuje vizuální stránku výsledné aplikace. Na přiložených obrázcích je vhodně ilustrováno rozdělení celkové aplikace, ukázka umístění jednotlivých tlačítek a funkcionalit a interaktivní dialogové okno pro nastavení parametrů pro image inpainting.

Závěrečná sekce 3.3 popisuje v krátkosti propojení neuronových sítí pro image inpainting s výslednou aplikací. Popisuje též plánovanou práci s vybranými modely pro image inpainting.

3.1 Funkcionalita

Hlavním smyslem celé aplikace je vytvořit nástroj pro snadné ověřování kvality různých modelů pro image inpainting. S tím souvisejí základní nástroje, které slouží k operacím potřebným k odstranění nějaké části obrazu. Důležitou složkou je pak doplnění chybějících částí obrazu a porovnání výsledků dvou modelů.

Nástroje

Nástroje si uživatel může vybrat z lišty nástrojů. Slouží k práci s obrazem a maskou. Uživatel si může zvolit z 9 základních nástrojů:

- Nástroj lupy
- Nástroj pohybu
- Nástroj štětce
- Nástroj gumy
- Nástroj bodového výběru
- Nástroj pro vyznačení hran
- Nástroj prahování
- Převod do šedotónového obrazu
- Nástroj pro úpravu barevných kanálů

Zbylá část této sekce se zabývá bližším popisem jejich funkcionality.

Nástroj lupy Nástroj lupy a s ním odpovídající posuvník slouží s přiblížení/oddálení obrazu. Po zamáčknutí levého tlačítka myši se obraz přiblíží, po zamáčknutí pravého tlačítka naopak oddálí. Tato funkcionalita je též implicitně zabudována do kolečka myši.

Nástroj pohybu Nástroj pohybu umožňuje posouvat obrazem po stisknutí levého tlačítka myši. Tento nástroj je použitelný v kombinaci s nástrojem lupy, kdy si uživatel může posouvat přiblíženým obrazem. Implicitně je funkce aktivována při konstantním stlačení kolečka myši a následným pohybem kurzoru.

Nástroj štětce Nástroj na tvorbu masek tvoří štětec, nanášející na obraz bílou barvu. Posuvníkem se upravuje tloušťka štětce. Uživatel nanáší masku kliknutím či podržením levého tlačítka myši. Krokové akce vrací celou akci, která nastala v období mezi zamáčknutím levého tlačítka myši a jejím následným zvednutím.

Nástroj gumy Nástroj gumy slouží k odmazání části masky. Velikost gumy se určí podle posuvníku. Jinak pracuje naprosto stejně jako nástroj štětce.

Nástroj bodového výběru Nástroj bodového výběru umožní uživateli klikáním myši umístit do obrazu body, propojené přerušovanou čárou. Uživatel si tak v případě potřeby může označit objekt tak, že jej jednoduše ohraničí mnohoúhelníkem. Ukončení výběru a vytvoření masky uvnitř vzniklého mnohoúhelníku nastane po propojení s počátečním bodem (kliknutím do blízkého okolí). Tlačítko dokončit instantně spojí poslední nanesený bod s počátkem nejkratší možnou cestou a vytvoří masku.

Následuje popis nástrojů pro editaci obrazu. Tyto funkce pracují s celým obrazem a celkově jej upravují. Jedná se o lehce vedlejší funkcionalitu, ale přesto je vhodné je obsáhnout

v této aplikaci. Tyto úpravy jsou dobré zejména k lepšímu pozorování změn ve struktuře obrazu po doplnění. Hodí se též k prozkoumání vlivu barev na výsledek modelů. V neposlední řadě se jedná o zajímavý nástroj na úpravu obrazů. Efekty těchto modifikátorů obrazu nebudou moci být aktivní zároveň, takže z principu jejich fungování může být aktivní pouze jeden. Uživatel může kdykoliv jejich efekt vypnout. Patří sem:

Nástroj pro vyznačení hran Zakliknutím tlačítka se v obrazu vyznačí hrany za využití laplaceova operátoru[1].

Převod do šedotónového obrazu Vložený obraz se převede do stupnice šedi.

Prahování Převede obraz na černobílý. U této funkce je možné si nastavit práh, kdy se černá překlopí do bílé pomocí posuvníku. Posuvník může nabývat hodnot od 0 do 255. Jestli je hodnota pixelu (součet všech barevných složek děleno třemi) větší než práh, tak se přebarví na bílou, jinak je pixel přebarven na černou.

Úprava barevných kanálů Nástroj slouží k úpravě barevných kanálů obrazu. Uživatel si může pomocí posuvníků nastavit barevný filtr na základní tři barvy (červená, zelená a modrá) a nastavit si tak jejich zastoupení v obraze.

Uživatel si může vrátit svou akci pomocí klávesové kombinace CTRL + Z, případně dopředu CTRL + SHIFT + Z, a to až do 20 kroků. Tyto činnosti bude moci provést i z lišty menu (podle zvoleného nástroje).

Kromě zmíněných tlačítek ke každému z nástrojů se v liště akcí bude permanentně nacházet tlačítko *Vyresetovat*. Zmáčknutí tohoto tlačítka zapříčiní celkové vymazání masky. Tento krok jde vrátit standardním způsobem.

Zbylé funkcionality

Zbytkové funkcionality se nacházejí v nejnižší části lišty. Jedná se o:

Změnu módu Změní se jak hlavní okno, tak doplňovací parametry. V tomto módu vidí uživatel dvě obrazovky, ve kterých se mu vždy po doplnění zobrazí obrazy po inpaintingu. Pořád může provádět všechny akce jako při módu jen s jedním obrazem.

Vizualizace masky Je možné vypnout vizualizaci masky, takže uživatel vidí pouze ground truth obraz.

Vložit obraz Kromě tohoto způsobu může uživatel vložit obraz kliknutím na prázdnou scénu či přetáhnutím obrazu přímo do scény.

Uložit obraz Možnost uložit výsledný obraz na definované místo skrz správce souborů.

Po zmáčknutí tlačítka *Vyplnit* se uživateli zobrazí výsledek v původním okně. Zase bude moci editovat obraz všemi dostupnými nástroji. Obrázek spolu s maskou si bude moci uložit ve vybraném formátu na zvolené místo pomocí souborového dialogového okna.

[1] Jádro je velikosti 3×3 s hodnotou 8 uprostřed a zbytek vyplněn hodnotou -1

Pokud si uživatel zvolí nějaký z porovnávacích módů a zmáčkne tlačítko pro doplnění, tak se mu výsledek zobrazí v upraveném hlavním okně. Zde uvidí vedle sebe oba výsledky, spolu s popisem a možností uložení. V této situaci může stále provádět všechny úpravy tak jako doposud. Jakoukoliv akci, kterou provede v jednom obrazu, tak ji také provede ve druhém.

3.2 Uživatelské rozhraní

Hlavním zaměřením při tvorbě uživatelského rozhraní byla snaha udělat aplikaci co možná nejjednodušší a nejpraktičtější. Jako hlavní barvy byly zvoleny černá s bílou a jejich odstíny. Následně červená barva byla využita pro zvýraznění aktivních prvků, jako je zvolený nástroj, či pro důležitá tlačítka.

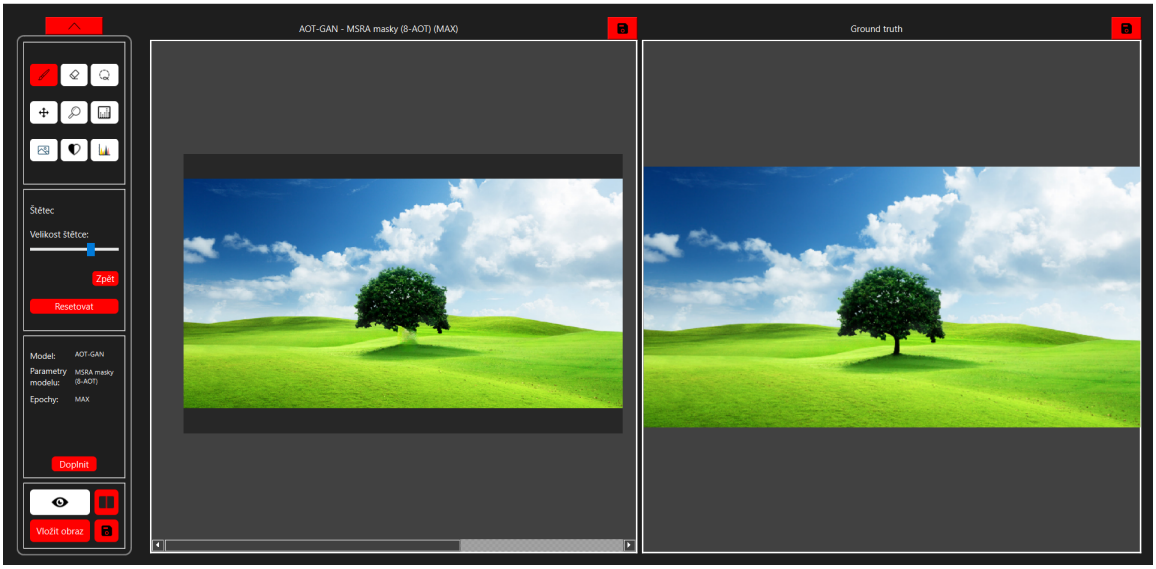
Struktura uživatelského rozhraní

Aplikace má jedno hlavní okno. Slouží jako místo pro práci s obrazem a tvorbu masky (viz obrázek 3.1). V tomto okně uživatel stráví většinu času. Většinu hlavního okna tvoří obdélník, sloužící jako zástupný element, který po vložení uživatelem bude nést obraz připravený k úpravě. Poté se v této scéně budou provádět změny v závislosti na uživateli.

Na levé straně se nachází lišta s editačním menu. Dá se volitelně schovat a zase rozdělat, čímž se uvolní více místa pro obraz. Tato lišta obsahuje všechny důležité funkcionality a též tlačítko sloužící k otevření dialogového okna.

Obrázek 3.1: Ukázka hlavního okna. Na levé straně se nachází lišta s nástroji, na pravé scéna s vloženým obrázkem.

Hlavní okno kromě základního módu s jedním obrazem má ještě tzv. porovnávací mód, kdy se zobrazí místo jednoho obrazu dva. Kvůli nedostatku místa jsou však oproti velikosti jednoho samostatného výrazně zmenšeny. Nad každým obrazem pak je vypsáno nastavení modelu pro daný obraz (viz obrázek 3.2). Uživatel může buď porovnávat výsledky dvou modelů, nebo porovnávat výsledek s ground truth.



Obrázek 3.2: Porovnávací mód. Každé okno má nad sebou popsané, o kterou variantu se jedná. Uživatel může dále pracovat s levým obrazem tak, jak byl zvyklý u jen doplňovacího módu.

Nástrojová lišta

Lišta nástrojů, zobrazená na obrázku 3.3, představuje hlavní místo pro nástroje. Uživatel si zde může zvolit nástroje, upravit jejich vlastnosti, či si upravit parametry modelu a další funkcionality. Je rozdělena na několik částí:

- V horní části se nachází nástroje sloužící k úpravě obrazu či manipulaci s ním. Příkladem je štětec sloužící k nanášení masky. Také obsahuje nástroje sloužící k úpravě celého obrazu, jako je například vyznačovač hran. Zvolený nástroj je rudě zvýrazněn.
- Ve druhém bloku se zobrazí proveditelné akce v závislosti na zvoleném nástroji. Příkladem je štětec, kdy se zde po zvolení zobrazí posuvník řídící velikost nanášené barvy, tlačítko pro vymazání celé masky a tlačítka „Zpět“ a „Vpřed“, která vrací jednu provedenou akci uživatele tam či zpět.
- Třetí blok slouží k nastavení parametrů pro image inpainting. Kliknutí na parametry vyvolává dialogové okno (viz obrázek 3.4. Je zde též tlačítko sloužící k zahájení doplnění.
- Poslední blok obsahuje tlačítka pro vložení obrazu, schování masky, změnu módu a uložení obrázku.

Parametry modelu

V liště nástrojů lze také vidět parametry právě vybraného modelu. Vypsán je vybraný model, zvolené trénovací parametry a počet natrénovaných epoch. Případně se ukazuje výška a šířka výsledného obrazu, pokud si je uživatel nastaví.

Dané hodnoty si uživatel nastaví v dialogovém okně (viz obrázek 3.4). To se mu objeví, když klikne kdekoliv do prostoru této části lišty.

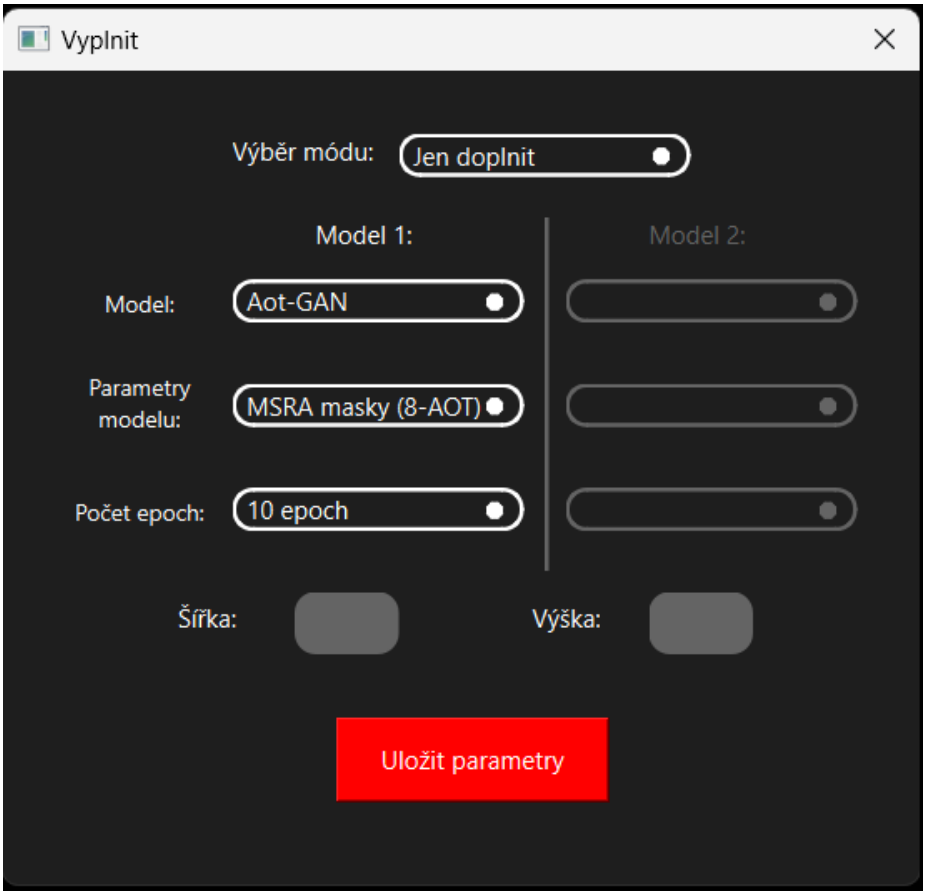
Obrázek 3.3: Nástrojová lišta. V prvním bloku se nacházejí tlačítka s použitelnými nástroji. V druhém bloku se vždy zobrazí proveditelné akce v závislosti na zvoleném nástroji. Následuje blok s parametry pro image inpainting. Kliknutím do této části se vyvolá dialogové okno pro jejich nastavení a tlačítko *Doplnit* zahájí inpainting. V posledním bloku se nacházejí tlačítka pro zapnutí/vypnutí masky, změnu módu, vložení nového obrázku a tlačítko pro jeho uložení.

První dropdown menu přepíná módy. Jedná se o porovnávací módy, při kterých se ve výsledku zobrazí vedle sebe groud truth a vyplněný obraz, respektive dva výsledně zaplněné

obrazy. Při porovnávání dvou modelů je vstup (maska a jí odpovídající ground truth) stejný, ale pro každý z nich si uživatel bude moci zvolit odlišné parametry. Tyto parametry jsou:

- Použitelné modely pro image inpainting.
- Modifikovatelné parametry daného modelu, které popisují nastavitelné parametry modelu či na kterých datech (jak barevné obrazy, tak masky) byla síť trénována.
- Na kolik epoch byl daný model vytrénovaný.
- Dvě pole pro zadání požadované délky a šířky výsledku. Ty uživatel nemusí vyplnit, implicitně se použijí parametry vloženého obrazu.

Po zakliknutí tlačítka *Uložit parametry* se výsledné hodnoty uloží a vypíší do dané sekce nástrojové lišty v hlavním okně aplikace.



Obrázek 3.4: Dialogové okno pro navolení parametrů pro inpainting. Lze si nastavit mód doplnění, model, jeho parametry a počet natrénovaných epoch. Potenciálně si uživatel může ještě nastavit výšku a šířku výsledného obrazu.

3.3 Modely a trénování

Byly zvoleny dva modely, AOT-GAN [32] a pconv [15]. Tyto modely budou vytrénované na datové sadě MSRA10K [4].

AOT-GAN model byl vybrán pro své kvalitní výsledky v poměru s nutnou délkou trénování. Po krátkých experimentech se ukázal jako dobrý model pro situaci, kdy není dostupná výkonná výpočetní jednotka a jsou dostupné jen omezené prostředky. Pconv byl zase vybrán jako jeden z nejlepších modelů s plně konvoluční architekturou. Jedná se o kvalitní komparátor kvality výsledků obrazového doplnění. Datová sada MSRA10K byla zvolena pro svou rozmanitost obrazů. které téměř vždy čítají nějaký hlavní objekt a pozadí. Počet deseti tisíc obrazů je vcelku obstojný z hlediska rozmanitosti. Hlavní výhodou jsou též masky, doprovázející obrazy. Jedná se o masky objektů, které se v těchto obrazech nacházejí a představují tak reálné využití, kdy uživatel chce umazat nějaký objekt, díky čemuž bude síť lépe připravená pracovat s reálnými daty.

Využito bude všech 10 tisíc obrazů na trénování. U trénovacích masek dojde k menší augmentaci dat, čímž vzniknou dvě trénovací sady. Jedna využívá jen masky označující reálné objekty pocházející právě z MSRA10K datové sady. Druhá si z něj bere 7,5 tisíc masek a 2,5 tisíc nepravidelných masek z datové sady použité v [15].

Kromě modifikace trénovací datové sady dojde též k porovnání odlišností při různě robustním modelu AOT-GAN. Základní verze s 8 AOT bloky bude porovnána s verzí obsahující pouze 6, respektive 2 bloky a bude se zkoumat, jak velký vliv bude mít tato změna z hlediska výsledku a časové náročnosti trénování. O přesném fungování AOT bloků a celkové konstrukci AOT-GAN modelu se lze dočíst v sekci 2.5. O architektuře a fungování pconv modelu se lze zase dočíst v sekci 2.4.

Na validaci je poté využita datová sada ECSSD [23]. Na výsledky vyplnění obrazu z této sady budou aplikovány kvantitativní metriky i vizuální kvalitativní porovnání. Více o způsobu testování a validace spolu s detailním popisem experimentů, je popsáno v sekci 5.1.

Cílem aplikace je též vytvořit intuitivní rozhraní, kdy bude jednoduché vložit další modely pro image inpainting.

Z frontendu přijde obraz a maska. Uživatel si vybere model podle libosti a tyto tři informace se pošlou na backendovou část aplikace. Zde se vybere správný modul obsahující model sítě a předá se mu obraz s maskou. Modul příkaz zpracuje a navrátí doplněný obraz jako výsledek. Při porovnávacím módu se tento cyklus provede dvakrát za sebou s potenciálně rozdílným modulem. Výsledný obraz se zase vrátí uživatelskému rozhraní.

V případě potencionálního budoucího rozšíření o další modely tak bude poměrně snadné tyto modely propojit s výslednou aplikací. Stačí vytvořit rozhraní požadující obraz a masku (případně potenciální další parametry), které vrátí výsledný doplněný obraz.

Kapitola 4

Implementace

Tato kapitula popisuje implementaci výsledné aplikace. V sekci 4.1 jsou vypsány všechny použité knihovny a aplikace, které byly při tvorbě aplikace pro image inpainting použity. Zároveň je zde popsána struktura výsledné aplikace.

Následující dvě sekce pak definují jednotlivé moduly, jejich význam a implementaci důležitých prvků. Sekce 4.2 popisuje implementaci logické části aplikace, od práce s obrazem po komunikaci s jednotlivými moduly pro inpainting. Sekce 4.3 popisuje implementaci vizuálních aspektů aplikace a komunikaci mezi jednotlivými interaktivními elementy aplikace.

4.1 Použité knihovny a struktura aplikace

Celá aplikace je napsaná v jazyce *Python* (verze 3.11.5). Knihovna *PyTorch* je hojně využívaná python knihovna pro strojové učení. Byla použita u převzatých modelů pro image inpainting. Pytorch je využit při úpravě obrazu, vytváření modelu a generaci nového obrazu. Pro práci s obrazy je využívána knihovna *PIL* (hlavně její třída *Image*). Tato knihovna byla využita při otevírání obrazů a u nástrojů, které editují celý obraz, jako je prahování či vyznačení hran.

Pomocí PyQt (verze 6) byly implementovány všechny elementy ve scéně, hlavní i dialogové okno. Stejně tak byl využit PyQt systém signálů a slotů pro zachycení komunikace mezi uživatelem, UI a funkcionalitou. PyQt je python odnož Qt, což je multiplatformní framework původně určený pro C++ sloužící pro popis, zobrazení a interakci s GUI.

Pro vizuální stránku aplikace byla využita aplikace *Qt Designer*. Jedná se o aplikaci pro tvorbu uživatelského rozhraní, která je napojená na PyQt. Dají se v ní definovat všechny interaktivní vizuální prvky. V Qt Designeru bylo vytvořeno hlavní okno aplikace a vyskakovací dialogové okno. Výsledné GUI soubory byly překonvertovány příkazem *pyui6*, čímž vznikly soubory *main_win_ui.py* a *dialog_ui.py*, které definují vizuální stránku odpovídající třídy.

Struktura

Aplikace je rozdělena do několika částí. Struktura celé aplikace pak vypadá následovně:

```
/ ..... kořenová složka
├── main.py ..... Zdrojový soubor pro spuštění aplikaci
├── Visual/ ..... Zdrojové soubory se zaměřením na frontend
├── Logic/ ..... Zdrojové soubory se zaměřením na backend
└── Models/ ..... Zdrojové soubory s převzatými modely
```

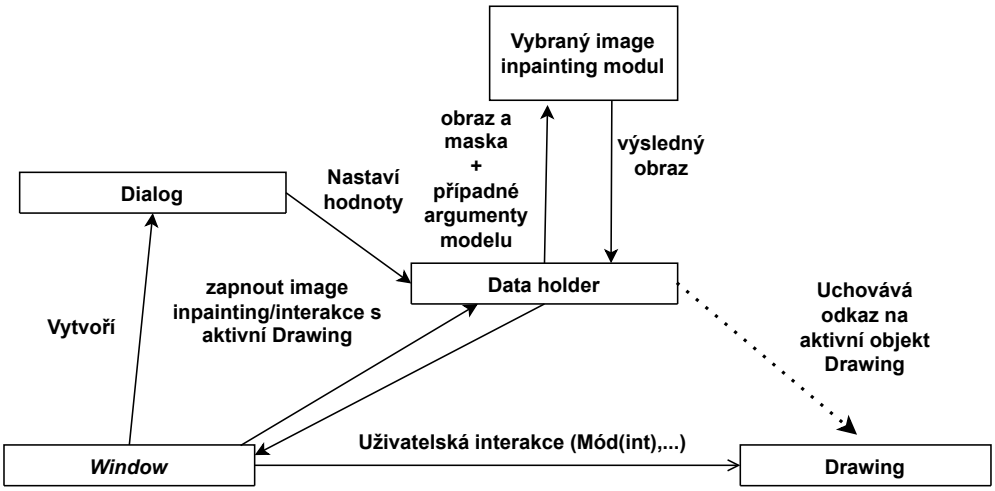
```
└── Checkpoints/ ..... Soubory s vytrénovanými daty
```

Moduly ve složce *Visual* popisují vzhled aplikace či interakci s UI prvky. Moduly v *Logic* naopak popisují logickou část aplikace a její celkovou funkcionalitu. Složka *Models* obsahuje převzaté zdrojové soubory pro AOT-GAN[1] a pconv[2]. Složka *Checkpoints* má strukturu:

```
Checkpoints
└── NÁZEV_MODULU ..... Složka s názvem inpainting modelu
    ├── v{číslo_verze}/..... Složka obsahující checkpointy pro daný typ trénování
    │   └── {číslo_verze}.pt/.pth..... Checkpoint modelu
    └── .../
```

Pro jednoduchou implementaci je každý typ trénování ve svojí složce nesoucí pořadové číslo (číslováno od 0) chronologicky podle pořadí trénování. Chronologické číslování je pak převzaté i u číslování jednotlivých checkpointů, kdy každý nový záznam představuje předem stanovený milník z hlediska počtu epoch (např. u AOT-GAN se jedná o dvě epochy). Tyto záznamy jsou slovně popsány v modulu *data_holder*, přesněji ve struktuře *models* jeho třídy.

Hlavní třídy spolu komunikují způsobem, který je zobrazen na obrázku 4.1.



Obrázek 4.1: Základní komunikace jednotlivých objektů aplikace. Třída *Window* zajišťuje hlavně vizuální aspekt aplikace a komunikaci mezi jednotlivými objekty při uživatelské interakci. Třída *Drawing* se stará o plochu malování a editaci obrazu. Třída *Data_holder* uchovává informace a propojuje logickou stránku s objektem Window. Objekt třídy *Dialog* se vytvoří vždy při uživatelské potřebě změnit explicitní parametry modelu pro image inpainting. Vybraný model spolu s jeho parametry, případně s novou velikostí obrazu jsou předány objektu třídy Data_holder. Při zavolání jsou tyto informace použity pro propojení obrazu a masky se správným modulem pro image inpainting. Navrácen je výsledný obraz, který má místo masky vygenerovaný obsah.

[1] https://github.com/researchmm/AOT-GAN-for-Inpainting.
[2] https://github.com/naoto0804/pytorch-inpainting-with-partial-conv/.

4.2 Logická část aplikace

Moduly nacházející se ve složce *Logic* se primárně zaměřují na funkcionální aspekty aplikace. Řídí tak její činnost v závislosti na informacích přicházejících z frontendu. Následuje popis modulů a jejich nejdůležitějších metod.

Moduly AOT-GAN a PCONV Tyto moduly propojují modely pro image inpainting s celkovou aplikací. Obě obsahují třídu *Args*, která slouží ke zpracování argumentů modelů. Ještě obsahují třídu *Pconv_gen* respektive *Aot_gen*. Ty řeší zpracování obrazu a masky tak, aby byly kompatibilní s danými modely. Modely se nacházejí v adresáři *Models*, moduly AOT-GAN[3] a pconv[4]. Pro potřeby kompatibility byly lehce upraveny tak, aby pracovali pouze s jedním obrazem a maskou a zároveň aby pracovaly s objektem třídy Args, který zajišťuje kompatibilitu vybraných parametrů s modelem (příkladem je počet AOT bloků). Na vstupu očekávají obraz a masku již v požadované velikosti a odkaz na záchytný bod vytrénovaného modelu. Některé modely potřebují nastavit i další parametry (například u AOT-GAN se jedná o počet AOT bloků v generátoru). Velikost obrazu a masky je pro kompatibilitu s modely upravena na hodnotu dělitelnou 4 (v případě potřeby je velikost o 1–3 pixely zmenšena).

Modul data_holder Třída *Data_holder* zde definovaná se stará o primární logickou stránku aplikace. Uchovává v sobě informace o napojených modelech a jejich nastavitelných parametrech ve slovníku *models*, kdy klíčem je zkratka sítě a hodnotou je pole dvojic, kde první prvek nese informaci o možných parametrech a druhý prvek zase pro daný parametr pole s počtem vytrénovaných epoch. Pro uchování nastavených parametrů od uživatele jsou využity dvě tříprvková pole, kde první prvek nese textově název modelu a zbylé dva zase index na dané parametry, respektive počet epoch.

Data_holder též hlídá aktivní *Drawing* objekt ve svém atributu *active_dr_place*. Hlavní okno se tak na něj může odkázat právě skrz tento atribut. Při změně scény (informace přichází od *Main_window*) se zavolá metoda *Change_scene*. V ní se změní ukazatel *active_dr_place* na nově aktivní Drawing objekt a zároveň se přenesou všechny důležité informace z původního objektu do nově aktivního (vrstva masky, obrazu, a zásobníky s uživatelskými akcemi tam a zpět).

Důležitou metodou je ještě *Parse_data*. Tato metoda pracuje se všemi zadanými parametry, které si objekt *Data_holder* právě uchovává. Kromě těchto parametrů ještě potřebuje za vstupní hodnoty obraz a masku, které nejprve přetransformuje na předem danou velikost (či ji ponechá původní v závislosti na uložených datech) a převede je do *PIL* formátu, se kterým pracují všechny moduly pro image inpainting. U masky se ještě vyplní transparentní pozadí za černé. Následně se obraz, maska a odkaz na uložený záchytný bod předají zvolenému modulu (AOT_GAN, respektive pconv). Návratovou hodnotou této funkce jsou dvě proměnné s doplněných obrazem, pokud se jedná o porovnání dvou doplnění, či jen jedem obraz a nulová hodnota jako druhá proměnná.

Modul dialog Obsahuje jedinou třídu *Fill_dialog*, která se stará o práci dialogového okna, které se zobrazí uživateli při úpravě parametrů modelu. Jeho vizuální stránka je definována v adresáři *Visual* a jeho modulu *dialog_ui*, který byl po vytvoření návrhu v aplikaci Qt Designer vygenerován PyQt konvertorem.

[3] https://github.com/researchmm/AOT-GAN-for-Inpainting.
[4] https://github.com/naoto0804/pytorch-inpainting-with-partial-conv/.

Tato třída řeší správné zobrazení parametrů v kontextu zvoleného modelu a módu. Po ukončení editace se z atributů dají vyčíst stanovené parametry pro jednotlivé modely v tříprvkových polích (model, parametry, epochy) *img_one* a *img_two*. Stejně se z atributů dá vyčíst i výška a šířka výsledného obrazu a zvolený móde. Vybrané parametry pak uživateli zobrazí objekt třídy *window*, která je též předá objektu třídy *Data_holder*.

Modul drawing Tento modul představuje asi nejdůležitější část celé aplikace. Je v něm definována všechna logika popisující práci s obrazem a maskou. Z důležitých parametrů je dobré zmínit *QLabel img* a jeho pixmapa *image_layer*, sloužící k vizualizaci ground truth obrazu. *QLabel mask* a jeho pixmapa *draw_layer* zase slouží k zobrazení nanesené masky a *QLabel dotted* a jeho pixmapa *dotted_layer* k zobrazení bodů při bodovém ohraničení.

Metodou *Add_img* se vloží obraz do *image_layer* a všechny kreslící vrstvy změní velikost tak, aby odpovídaly poměru stran právě vloženého obrazu. Všechna předchozí data se vymažou.

Drawing vždy pracuje v určitém módu, který je určen zvoleným nástrojem. Podle něj objekt různě reaguje na uživatelské vstupy. Malování zaopatřuje funkce *Drawing_point*. Při zvoleném nanášení masky se do *draw_layer* vykreslí bod při pouhém kliknutí, případně čára v případě zamáčknutí levého tlačítka myši a následného pohybu. Vždy se uchovává poslední pozice kurzoru (atribut *mk_pos*) do té doby, než je levé tlačítko myši znovu zvednuto. Při každém pohybu kurzoru je tak vždy vykreslena čára mezi *mk_pos* a aktuální pozicí kurzoru, načež se *mk_pos* nastaví na aktuální pozici. Úplně stejně funguje i nástroj gumy, který však z daného místa umaže masku, pokud se zde nachází.

U *dotted* módu je možné jen do prostoru klikat, kdy se do *dotted_layer* vykreslí tečka na aktuální pozici kurzoru. Zase se uchovává poslední pozice kurzoru (všechny body se ukládají do PyQt třídy *Polygon*) pro následné vykreslení přerušované čáry mezi dvěma po sobě jdoucími body. Zároveň se kontroluje, jestli rozdíl pozice kliku (součet absolutních hodnot rozdílů atributů x a y) není menší než 10 od počátečního bodu. Pokud tato možnost nastane, je vyvolán odpovídající *Trigger_event*, vzniklý mnohoběžník se vyplní v *draw_layer* a body ze scény i polygon se smažou.

Funkce *Relative_calcul* přepočítává souřadnice kurzoru v obraze. Při vkládání obrazu je obraz různě zmenšen, ale hlavně má zachovaný poměr stran, takže nezabírá celou velikost objektu Drawing. Proto je nutné každý *MouseEvent* a jeho pozici přepočítat tak, aby klik odpovídal přesně pozici kliku v obraze.

Metody *Updates* a *Trigger_event* zpracovávají vstupy od hlavního okna. Metoda *Updates* zpracovává změnu nástroje a *Trigger_event* zase interakci uživatele s nějakým elementem z nabídky akcí.

Atributy *undo_img* a *redo_img* uchovávají předchozí stav masky. Jedná se o zásobníkové struktury, kdy se využívají funkce *Save_step* a *Pop_step*. Save_step ukládá celkový stav masky až do maximální velikosti zásobníku, po kterém se pro uvolnění místa začnou vyhazovat data, která tam jsou nejdéle. Pop_step pak má dva módy. Při kroku zpět se nejprve uloží stávající stav masky do zásobníku *redo_img*, následně se ze zásobníku *undo_img* vyjme jeden stav, kterým se aktualizuje stávající maska.

4.3 Uživatelské rozhraní aplikace

Uživatelské rozhraní je definováno v agresáři *Visual*. Modul *main_win_ui* popisuje vizuální stránku hlavního okna a modul *dialog_ui* zase vizuální stránku dialogového okna. Tyto dva

moduly byly vygenerovány aplikací Qt Designer, ve které byly dané stránky ručně vyrobeny. Hlavní okno je popsané v modulu *window*.

Modul window Spojuje celou aplikaci dohromady. Při inicializaci vytváří všechny objekty potřebné pro chod aplikace. Propojuje též všechny interaktivní objekty s jejich implementovanou funkcionalitou. Jsou inicializované dvě *drawing_place* (popsány v sekci 4.2), každá pro jednu scénu. Vždy je ale aktivní jen jedna, se kterou se pracuje (skrz odkaz v *Data_holder*).

Pro každou scénu s *drawing_place* je vytvořen pohled (*QGraphicsView*), který je v *Main_window* také definován. Jsou definovány jeho metody pro *mousePressEvent* jako *Press_view*, *mouseReleaseEvent* jako *Release_view*, *mouseMoveEvent* jako *Move_view* a *wheelEvent* jako *Zoom*. Slouží pro nástroje pohybu a přiblížení. Pro přiblížení je využita metoda *scale*, zděděná z QGraphicsView, a to s parametrem $1,2$ pro přiblížení a $\frac{1}{1,2}$ pro oddálení. U pohybu se nejdříve, po zamáčknutí tlačítka (kolečko myši či levé tlačítko v pohybovém módu), uchová pozice kurzoru a nastaví se odpovídající hybný mód pohledu. Při posunu kurzoru na určitou pozici, za stálého držení daného tlačítka, se vždy vypočítá rozdíl původní pozice s pozicí stávající, načež se o tento rozdíl posune celý pohled ve scéně.

Všechna tlačítka nástrojů jsou propojena s metodou *Mode_select*, kde jako lambda parametr vkládají své číselné označení. V metodě *Mode_select* se číselné označení nástroje předá právě využívané *drawing_place*, a zároveň se v boxu pro nástrojové akce vizualizují akce, které je možné s daným nástrojem provádět, ve formě tlačítek, případně posuvníku. Všechny ostatní interaktivní prvky jsou schovány. Také se zde textově vypíše zvolený nástroj (využije se *setText* metoda *ui.tooltip_label*).

Akční tlačítka jsou zase propojena s metodou *Action*, opět s lambda parametrem nesoucím číselné označení (vyjma posuvníku, který má svoji vlastní metodu *Scaler*). Tyto metody jen předávají informace zrovna aktivnímu *drawing_place*.

Pro výběr parametrů modelů se zavolá metoda *Dialog_window*. Hover effekt této sekce byl docílen reimplementováním metod *enterEvent* a *leaveEvent* objektu *ui.hover*. Tento objekt má plně bílé pozadí, kterému se nastaví viditelnost na 0. Po aktivaci *enter* eventu se mu nastaví průhlednost na $0,4$, po opuštění kurzoru (*leaveEvent*) se mu zase vrátí průhlednost na 0. Tlačítko pro zahájení doplnění vyvolá metodu *Fill*. V ní se předá informace o aktivaci doplnění objektu *Data_holder*, který pomocí svých uložených parametrů kontaktuje příslušné moduly, a po zpracování vrátí zpět doplněný obraz. Obraz je následně, po konverzi z PIL obrazu na pixmapu, předán odpovídající *drawing_place*. Podle zvoleného módu se změní scéna.

Tlačítka pro změnu scény, vypnutí masky, vložení obrazu a uložení obrazu mají každá svoji vlastní funkci, která řeší jejich funkcionalitu.

Kapitola 5

Testování a dosažené výsledky

Tato kapitola pojednává o dosažených výsledcích při trénování AOT-GAN modelu. Sekce 5.1 popisuje experimenty, které byly provedeny s datovou sadou, respektive s modelem. Je zde též popsán způsob trénování. Výsledky těchto experimentů jsou popsány hned v následující sekci 5.2. Nejprve jsou modely porovnány z hlediska jejich kvantitativních metrik, které jsou totožné s metrikami využitými v Zeng et al. [32]. Následně je provedeno kvalitativní vyhodnocení, kde jsou mezi sebou porovnány jednotlivé modely a jsou představeny nejlepší výsledky doplněné o případy, kdy se výsledky nesetkaly s odpovídající kvalitou. Toto všechno je shrnuto v závěrečné sekci 5.3.

5.1 Popis experimentů

Všechny experimenty, jak trénování, tak následné testování bylo provedeno na stroji s grafickou kartou NVIDIA GeForce RTX 4060 (16GB), s procesorem 13th Gen Intel(R) Core(TM) i5-13500H a 16GB RAM.

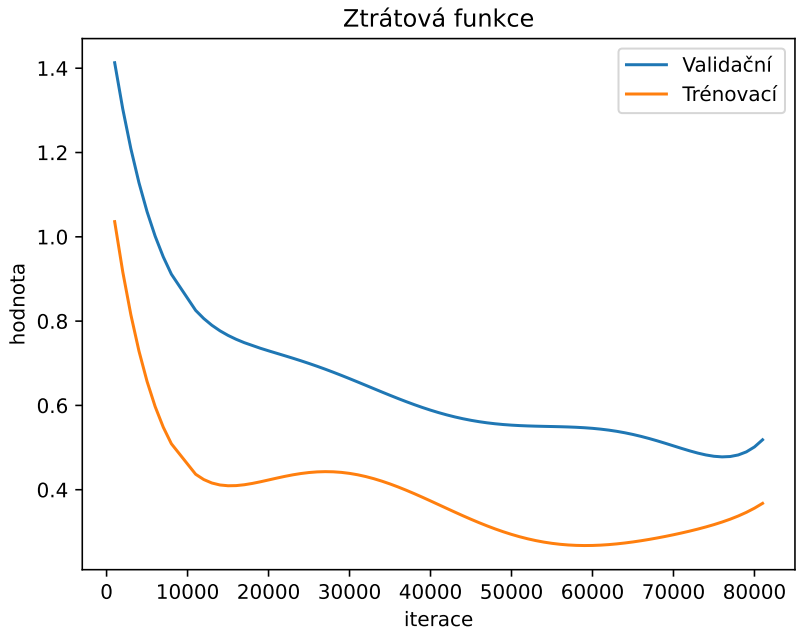
Všechny modely byly trénovány na datové sadě *MSRA 10K* [4]. 7,5 tisíc masek bylo použito právě z této datové sady, kdy tyto masky představují nějaký reálný objekt, který se nachází ve scéně. Při trénování pak byly tyto masky promíchány. Zbylých 2,5 tisíc bylo použito z datové sady *NVIDIA Irregular Mask* [15], aby se tak docílilo větší rozmanitosti masek. Obrazům a maskám byla následně upravena velikost na 512×512 pixelů, aby se následovala doporučená velikost pro trénování [32].

Pconv neuronová síť byla trénována 160 epoch s obvyklou rychlostí učení $0,0002$ a následně 160 *fine-tuning* epoch s rychlostí učení $0,00005$ při *batch size* 16. Doba trvání trénování byla něco málo přes 2 dny.

Doba trénování AOT-GAN byla nejprve stanovena na 8 epoch, což se následně odhalilo jako nedostatečné vytrénování (viz obrázek 5.1), a tak se nakonec doba trénování stanovila na 14 epoch. Trénovací sada byla rozdělena na díly o batch size 2. Proběhlo několik různých natrénování modelu:

- Standardní model s 8 AOT bloky. Jeho trénování trvalo 6 dní.
- Model s 6 AOT bloky, kdy trénování proběhlo dvakrát tak rychleji.
- Model s 2 AOT bloky. U tohoto modelu trénování na 14 epoch zabralo necelých 18 hodin.

Výsledky těchto experimentů jsou popsány v sekci 5.2, kde je provedeno jejich kvantitativní i kvalitativní srovnání.



Obrázek 5.1: Ukázka vývoje ztrátové funkce v poměru k počtu iterací při trénování sítě AOT-GAN. Dolní osa odpovídá počtu iterací (jedna epocha odpovídá 5000 iteracím při batch size 2 a velikosti datové sady 10 tisíc), horní osa odpovídá hodnotě výsledné ztrátové funkce. Oranžová linie značí trénovací ztrátovou funkci, modrá naopak validační ztrátovou funkci na datech, které síť ještě neviděla. Více o použité ztrátové funkci lze nalézt v sekci 2.5 u popisu AOT-GAN modelu. Je patrné, že i po 40000 iteracích (tj. 8 epoch) obě funkce stále klesají. Proto se trénování ukončilo na 14 iteracích, kdy validační ztrátová funkce dosahuje nejnižší hodnoty.

5.2 Dosažené výsledky

Pro kvantitativní vyhodnocení výsledků byly použity stejné metriky jako v práci Zeng et al. [32], a to *MAE*, *PSNR*, *SSIM* a *FID*.

MAE Neboli *mean absolute error* udává rozdíl mezi hodnotami pixelů reálného a generovaného obrazu. Popisuje tak pixelovou přesnost rekonstrukce.

PSNR Neboli *peak signal-to-noise ratio*, vyjadřuje poměr mezi maximální možnou energií signálu a energií šumu.

SSIM Neboli *structural similarity index*, porovnává lokální vzorky intenzit pixelů, které byly normalizovány pro kontrast a jas. [30] Tato metrika se snaží napodobit vizuální vnímání člověka, zvláště jeho reakci na změnu kontrastu a jasu v části obrazu. Pro více informací lze pročíst článek od Wang et al. [30], kteří jsou mimo jiné autory této metriky.

Tabulka 5.1: Kvantitativní porovnání modelů vytrénovaných na MSRA10K datové sadě. Použity byly velké masky, které zabírají v průměru 50% obrazu. Šipka ↑ u metrik značí, že čím vyšší hodnota, tím je model kvalitnější. Naopak při ↓ je menší hodnota lepší. Číslo v závorce u AOT-GAN udává počet AOT bloků v generátoru.

Model	**MAE ↓**	**PSNR ↑**	**SSIM ↑**	**FID ↓**
AOT-GAN (8)	0,041	23,581	0,847	341,146
AOT-GAN (6)	0,046	23,616	0,845	341,197
AOT-GAN (2)	0,0419	23,197	0,843	341,948
pconv	0,339	9,984	0,180	348,493

Tabulka 5.2: Kvantitativní porovnání modelů vytrénovaných na MSRA10K datové sadě, kde masky zabírají pouze 10–20% z celkového obrazu. U ↑ je vyšší hodnota lepší, u ↓ je menší hodnota lepší. Číslo v závorce u AOT-GAN udává počet AOT bloků v generátoru.

Model	**MAE ↓**	**PSNR ↑**	**SSIM ↑**	**FID ↓**
AOT-GAN (8)	0,0100	33,416	0,958	319,848
AOT-GAN (6)	0,0102	33,333	0,957	320,541
AOT-GAN (2)	0,0104	33,070	0,956	320,085
pconv	0,089725	26,098	0,759	312,528

FID Neboli *Fréchet inception distance*, se využívá hlavně pro klasifikaci generativních sítí. Porovnává distribuci generovaných obrazů s distribucí reálných ground-truth obrazů. [11] Hodnotí tak výsledek z hlediska jeho realičnosti.

Kvantitativní vyhodnocení

Na validaci bylo použito 250 obrazů z datové sady *ECSSD* [23]. Ty byly spolu s maskami použity pro generování výsledku jednotlivými modely. Tyto výsledky pak byly validovány porovnáním s ground truth a zapsány do tabulek 5.1 a 5.2.

Tabulka 5.1 popisuje výsledky při doplňování obrazu s velkými maskami znázorňující nějaké reálné objekty (masky převzatě z dané datové sady). Popisuje tak výsledky při zamýšleném používání modelů na odstranění objektu ze scény.

Tabulka 5.2 zase používá masky převzaté z pconv [15], aby se tak docílilo i způsobu kvantitativního vyhodnocení, kde by pconv měla dosahovat lepších výsledků oproti velkým maskám.

Jak jde vidět na tabulce 5.1, AOT-GAN síť dosahuje lepších výsledků ve všech měřených metrikách u masek ve tvaru objektů, které zabírají v průměru 50% celkového obrazu. Dá se též odvodit, že počet AOT bloků v AOT-GAN síti zlepšuje výsledný výsledek. Jedinou anomálií je hodnota PSNR, kde má nejlepší výsledek AOT-GAN se 6 bloky. Je možné, že validační sada obsahovala specifický šum, se kterým si lépe poradil právě AOT-GAN se 6 bloky.

Pro srovnání byla provedena validace i na rozměrově menších maskách (v rozmezí 10–20%), jejíž výsledky se nacházejí v tabulce 5.2. Pconv zde dosahuje mnohem lepších výsledků oproti velkým maskám. Dokonce dosahuje lepších výsledků ve zkoumané metrice FID než AOT-GAN model. FID byl vytvořen hlavně pro kontrolu kvality výsledku u generativních modelů, takže je otázka, nakolik se jedná o odpovídající údaj. Ve všech ostatních metrikách však AOT-GAN předčí pconv, a to o stále marginální rozdíl.

Kvalitativní vyhodnocení

Na dalších několika stránkách jsou vyobrazeny kvalitativní výsledky. Pro kvalitativní vyhodnocení byla použita vytvořená aplikace. Nejprve bylo provedeno porovnání AOT-GAN modelu s různým množstvím AOT bloků a pconv. Některé výsledky jsou vizualizovány na obrázku 5.2. Obrázek 5.3 znázorňuje použití těchto modelů při odstranění náhodně vytvořené masky. Závěrečný obrázek 5.4 poté zobrazuje problémové situace, kdy vytrénované modely selhávají.

Z kvalitativních výsledků vyplývá, že zde vytrénovaná AOT-GAN síť si poměrně s přehledem poradí s přírodní scenérií. Při maskách ve formě různě dlouhých a širokých čar, které zabírají okolo 50% celkového obrazu, si s tímto problémem obstojně poradí oba vytrénované modely. Při odstraňování větších objektů ze scény si pconv často neporadí se správným kontrastem a je tak patrný vizuální rozdíl. Pokud se zaměříme jen na AOT-GAN, tak lze pozorovat vliv mohutnosti generátoru na doplněný výsledek. U modelu s 8 a 6 AOT bloky se dají pozorovat jen drobné odlišnosti, které se projeví případ od případu. U modelu s pouhými dvěmi AOT bloky se však dá poměrně zřetelně pozorovat rozdíl v barvách. Model si nedokáže obstojně propojit celkový kontext obrazu, a tak jsou dost často zřetelné barevné odlišnosti mezi zaplněnou částí a částí původní. Potvrdila se též úvaha, že při větších maskách, případně při větším rozlišení výsledného obrazu již modely s menším počtem AOT bloků nedokáží doplnit nejvnitřnější chybějící oblasti. Utvoří se tam viditelné artefakty jedné barvy. Pravděpodobnou příčinou je nedostatek AOT bloků, kvůli čemuž daná síť nedokáže zachytit všechny potřebné informace. Dva, případně šest bloků v generátoru jednoduše nezvládá takové množství dat, což má za následek zmíněné artefakty.

Natrénované modely si neporadí zejména s lidskými stavbami. Jak je ukázáno na obrázku 5.4, kontext těchto scén (zejména ty z interiéru) model nedokáže pochopit. Nejlépe to jde vidět na rovných hranách, které jsou ve výsledku různě pokroucené a je snadné postřehnout, že se jedná o generovaný obraz.

5.3 Závěrečné zhodnocení

Jak kvalitativním, tak kvantitativním vyhodnocením bylo prokázáno, že AOT-GAN dosahuje lepších výsledků než pconv, hlavně při práci s velkými maskami. Velikost AOT-GAN modelu, přesněji počet jeho AOT bloků, má přímý vliv na výsledný obraz. Z kvantitativního hlediska sice nelze pozorovat velké rozdíly, nicméně z hlediska kvalitativního lze u většiny situací rozpoznat znatelné rozdíly. Méně AOT bloků znamená méně práce s kontextem obrazu, což má za následek horší kompatibilitu doplněné části obrazu s jeho zbytkem, kdy model s pouze dvěma AOT bloky má tendenci vyplňovat chybějící část obrazu mnohem temnějšími barvami, což je hlavně u velkým masek velice patrné.

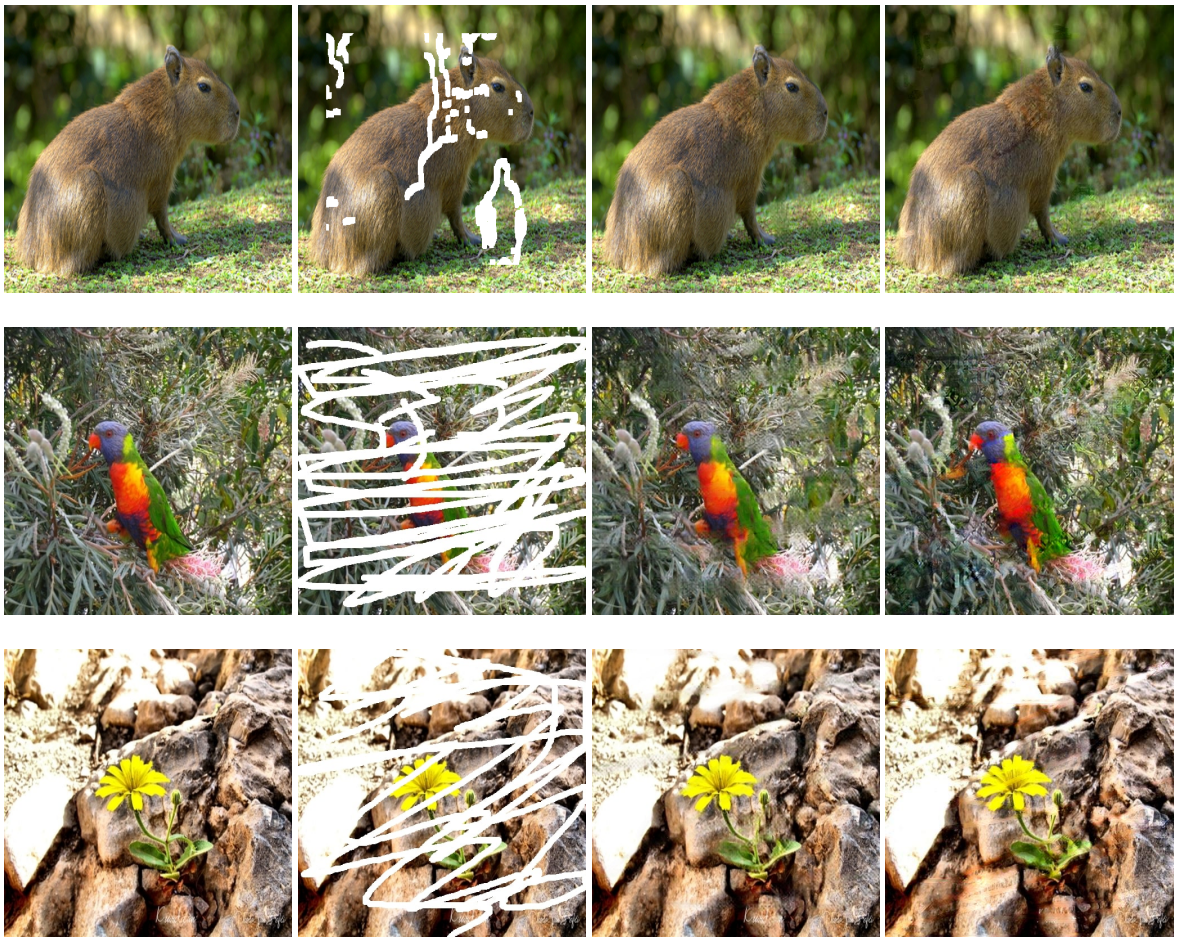
Vytrénované modely si poměrně dobře poradí s přírodní scenérií. Poradí si též při odstraňování menších objektů ze scény. Problémy nastávají zejména u scén s lidskými stavbami, nebo interiéry. V těchto situacích si modely nedokáží poradit zejména s ostrými hranami, které se často snaží různě zakulatit, což je poměrně vizuálně výrazné.

Potenciálním vylepšením by bylo do datové sady zakomponovat i obrazy z městské scenérie, případně vytvořit novou datovou sady, která bude obsahovat pouze obrazy lidských staveb, městských čtvrtí a interiérů.

(a) Zamaskovaný obraz (b) Zamaskovaný obraz (c) Zamaskovaný obraz

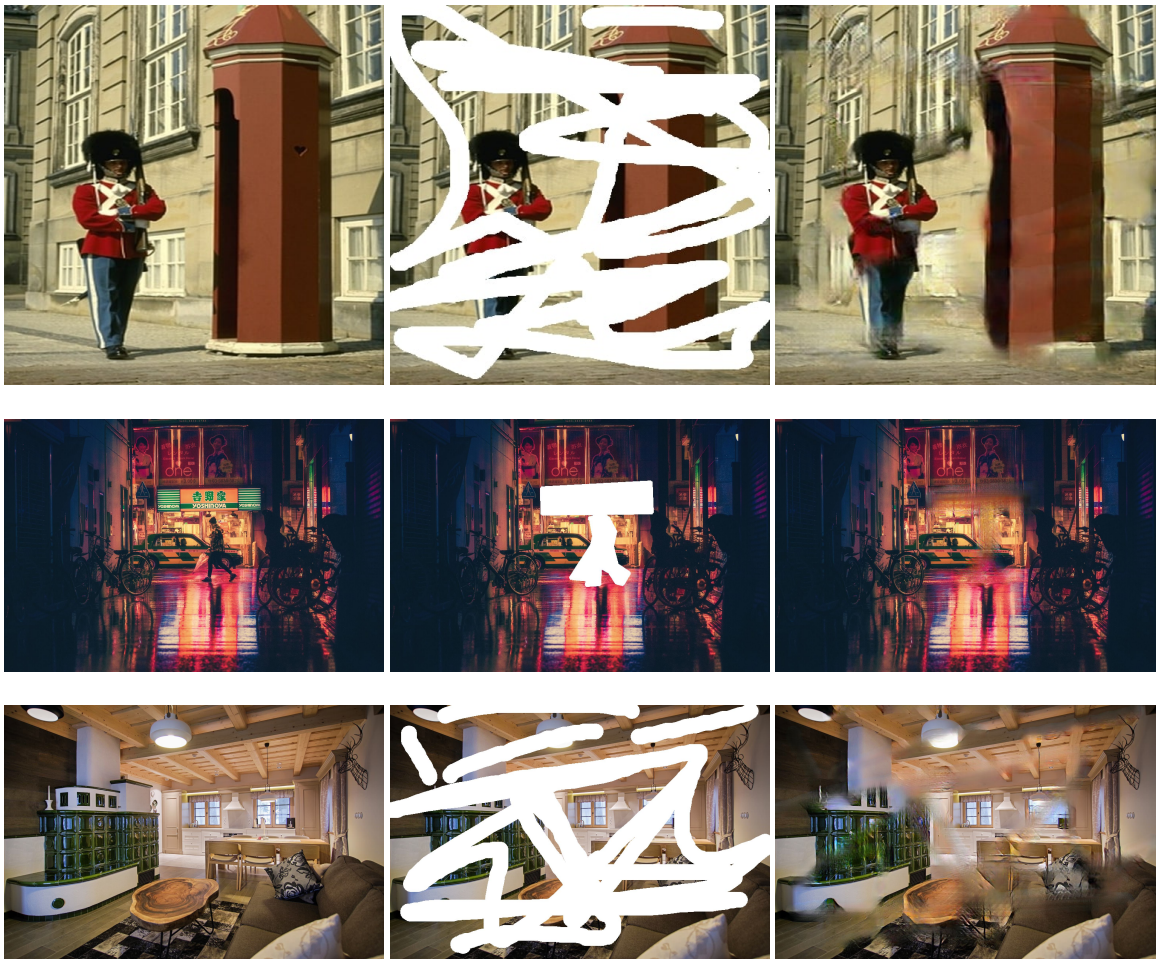
(d) AOT-GAN (2) (e) AOT-GAN (6) (f) AOT-GAN (8) (g) pconv

Obrázek 5.2: Kvalitativní porovnání všech vytrénovaných modelů. Čísla v závorce u AOT-GAN udávají počet AOT bloků v generátoru. Lze vidět, že AOT-GAN s větším množstvím AOT bloků dokáže lépe pracovat s barvami v obraze a výsledky jsou mnohem jasnější. Tento rozdíl je zejména patrný mezi 8 AOT bloky a 2 AOT bloky.



(a) Ground truth (b) Zamaskovaný obraz (c) AOT-GAN (d) pconv

Obrázek 5.3: Ukázka práce modelů na nepravidelných maskách. Vytrénované modely si velice dobře poradí s přírodní scenérií. Dosahují i poměrně kvalitních výsledků u masek zakrývajících velkou část obrazu.



(a) Ground truth (b) Zamaskovaný obraz (c) AOT-GAN (8)

Obrázek 5.4: Ukázky použití, kdy AOT-GAN selhal. Datová sade MSRA obsahuje jen velice málo obrazů města, interiérů či obecně lidských staveb. Často tak dochází k viditelným deformacím právě v těchto situacích.

Kapitola 6

Závěr

Nejprve byl proveden rozsáhlý výzkum metod pro image inpainting. Pomocí případu užití potenciálními uživateli byl vytvořen návrh popisující jak funkcionalitu, tak vizuální stránku aplikace pro testování a porovnání metod pro image inpainting. Byly zvoleny dva modely, AOT-GAN a pconv, které byly vytrénovány na zvolené trénovací datové sadě MSRA10K. Hlavní důraz se kladl na AOT-GAN, kdy byly prováděny experimenty s množstvím AOT bloků v generátoru a byl zkoumán jejich vliv na výsledný obraz.

Natrénované modely podávají ucházející výsledky při odstraňování objektů z přírodní scenérie a bylo zjištěno, že na obrazy velikosti 512×512 lze využít i generátor o mohutnosti pouhých 2 AOT bloků, ale za cenu snížení kvality barev. Při testování kvality výsledků na městských obrazech se objevili vizuálně viditelné artefakty, takže v budoucnu je možné vylepšit výsledky AOT-GAN modelu například vytvořením rozličných datových sad pro různé případy použití, kdy by si uživatel následně mohl zvolit vytrénovaný model pro daný problém. Kvantitativní vyhodnocení potvrdilo očekávaný předpoklad, že AOT-GAN je lepší model pro image inpainting než pconv. Tyto rozdíly byly patrné hlavně na metrice MAE, u které činil rozdíl u velkých masek až 0,3. U masek menších rozměrů se tento rozdíl snížil na pouhých 0,08, kdy dokonce pconv dosahovala lepších výsledků v metrice FID, a to o 8 bodů. V ostatních metrikách však stále výrazně zaostávala.

Při kvalitativním vyhodnocení byla použita vytvořená aplikace, čímž se otestovala její využitelnost pro testování a porovnání výsledků pro image inpainting. V budoucnu je možné aplikaci rozšířit o další metody, například o moderní difuzní metody.

Literatura

[1] Barnes, C., Shechtman, E., Finkelstein, A. a Goldman, D. B. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (Proc. SIGGRAPH).* srpen 2009, sv. 28, č. 3.

[2] Bertalmío, M., Sapiro, G., Caselles, V. a Ballester, C. Image inpainting. In:. Leden 2000, s. 417–424.

[3] Burlin, C., Le Calonnec, Y. a Duperier, L. *Deep image inpainting.* 2017.

[4] Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H. S. a Hu, S.-M. Global Contrast based Salient Region Detection. *IEEE TPAMI.* 2015, sv. 37, č. 3, s. 569–582. DOI: 10.1109/TPAMI.2014.2345401.

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 2009, s. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[6] Deng, Y., Hui, S., Zhou, S., Meng, D. a Wang, J. T-former: An Efficient Transformer for Image Inpainting. In: *Proceedings of the 30th ACM International Conference on Multimedia.* New York, NY, USA: Association for Computing Machinery, 2022, s. 6559–6568. MM '22. DOI: 10.1145/3503161.3548446. ISBN 9781450392037.

[7] Efros, A. A. a Leung, T. K. Texture synthesis by non-parametric sampling. In:. 1999, sv. 2, s. 1033 – 1038.

[8] Furht, B., ed. Image Inpainting. In: Furht, B., ed. *Encyclopedia of Multimedia.* Boston, MA: Springer US, 2006, s. 315–316. ISBN 978-0-387-30038-2.

[9] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning.* MIT Press, 2016.

[10] Goodfellow, I., Pouget Abadie, J., Mirza, M., Xu, B., Warde Farley, D. et al. Generative Adversarial Networks. *Advances in Neural Information Processing Systems.* Červen 2014, sv. 3. DOI: 10.1145/3422622.

[11] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. a Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In: Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R. et al., ed. *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2017, sv. 30.

[12] Hinton, G. E. a Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*. 2006, sv. 313, č. 5786, s. 504–507. DOI: 10.1126/science.1127647.

[13] King, D. *The Commissar Vanishes: The Falsification of Photographs and Art in Stalin's Russia*. Henry Holt and Company, 1997. ISBN 9780805052947.

[14] Liao, L., Liu, T., Chen, D., Xiao, J., Wang, Z. et al. TransRef: Multi-Scale Reference Embedding Transformer for Reference-Guided Image Inpainting. *ArXiv preprint arXiv:2306.11528*. 2023.

[15] Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A. et al. Image Inpainting for Irregular Holes Using Partial Convolutions. In: *The European Conference on Computer Vision (ECCV)*. 2018.

[16] Liu, Z., Luo, P., Wang, X. a Tang, X. Deep Learning Face Attributes in the Wild. In: *Proceedings of International Conference on Computer Vision (ICCV)*. December 2015.

[17] Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R. et al. RePaint: Inpainting using Denoising Diffusion Probabilistic Models. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, s. 11451–11461. DOI: 10.1109/CVPR52688.2022.01117.

[18] Nielsen, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015.

[19] O'Shea, K. a Nash, R. An Introduction to Convolutional Neural Networks. *ArXiv*. 2015, abs/1511.08458.

[20] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. a Ommer, B. High-resolution image synthesis with latent diffusion models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, s. 10684–10695.

[21] Ronneberger, O., Fischer, P. a Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W. M. a Frangi, A. F., ed. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, s. 234–241. ISBN 978-3-319-24574-4.

[22] Rumelhart, D. E., Hinton, G. E. a Williams, R. J. Learning representations by back-propagating errors. *Nature*. 1986, sv. 323, s. 533–536.

[23] Shi, J., Yan, Q., Xu, L. a Jia, J. Hierarchical Image Saliency Detection on Extended CSSD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2016, sv. 38, č. 4, s. 717–729. DOI: 10.1109/TPAMI.2015.2465960.

[24] Simonyan, K. a Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio, Y. a LeCun, Y., ed. *ICLR*. 2015.

[25] Sohl Dickstein, J., Weiss, E. A., Maheswaranathan, N. a Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. JMLR.org, 2015, s. 2256–2265. ICML'15.

[26] SPOLEČNOST Česká meteorologická. izofota. *Elektronický meteorologický slovník (eMS)* [online]. 2017 [cit. 20.4.2024]. Dostupné z: http://slovnik.cmes.cz/heslo/1327.

[27] SUVOROV, R., LOGACHEVA, E., MASHIKHIN, A., REMIZOVA, A., ASHUKHA, A. et al. Resolution-robust Large Mask Inpainting with Fourier Convolutions. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, s. 3172–3182.

[28] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L. et al. Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017, s. 6000–6010. NIPS'17. ISBN 9781510860964.

[29] WAN, Z., ZHANG, J., CHEN, D. a LIAO, J. High-Fidelity Pluralistic Image Completion with Transformers. In: *Proceedings - 2021 IEEE/CVF International Conference on Computer Vision*. IEEE, 2021, s. 4672–4681. DOI: 10.1109/ICCV48922.2021.00465. ISBN 978-1-6654-2813-2.

[30] WANG, Z., BOVIK, A., SHEIKH, H. a SIMONCELLI, E. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*. 2004, sv. 13, č. 4, s. 600–612. DOI: 10.1109/TIP.2003.819861.

[31] YU, J., LIN, Z., YANG, J., SHEN, X., LU, X. et al. Free-Form Image Inpainting With Gated Convolution. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, s. 4470–4479. DOI: 10.1109/ICCV.2019.00457.

[32] ZENG, Y., FU, J., CHAO, H. a GUO, B. Aggregated Contextual Transformations for High-Resolution Image Inpainting. *CoRR*. 2021, abs/2104.01431.

[33] ZHAO, S., CUI, J., SHENG, Y., DONG, Y., LIANG, X. et al. Large Scale Image Completion via Co-Modulated Generative Adversarial Networks. In: *International Conference on Learning Representations (ICLR)*. 2021.

Příloha A

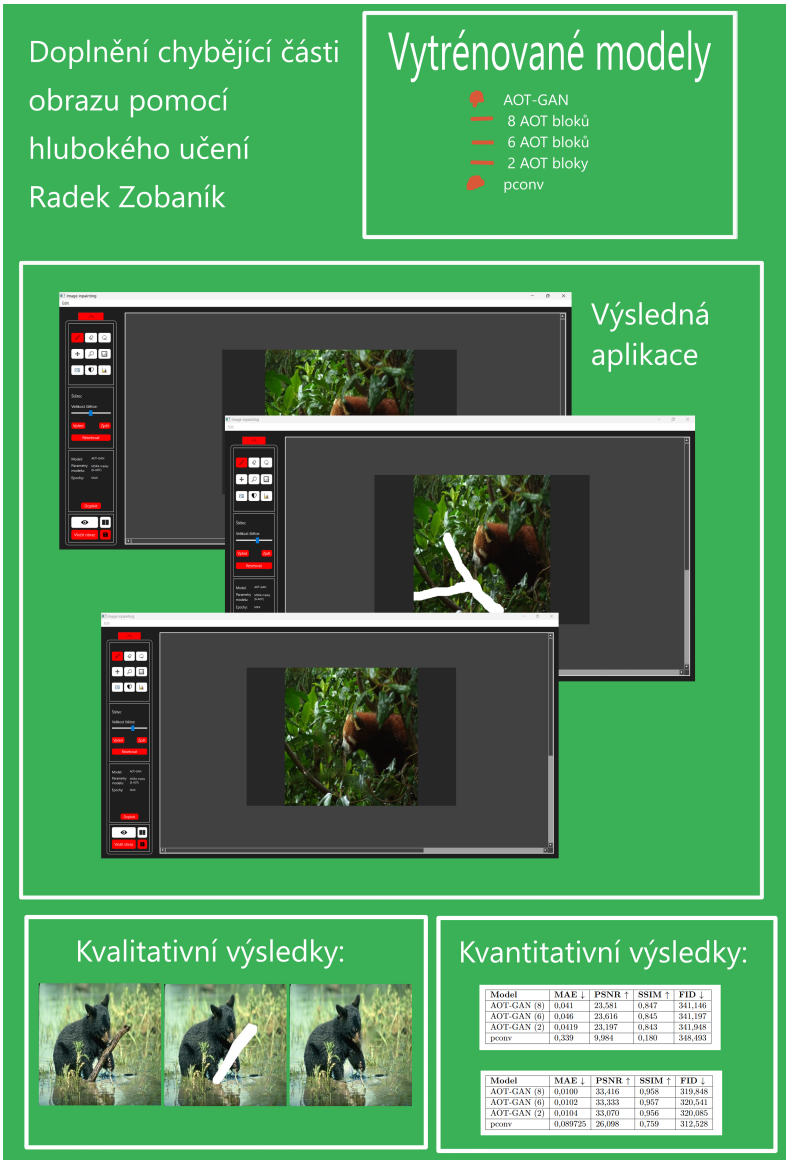
Obsah přiloženého paměťového média

Struktura aplikace je pak následovná:

- `/` ..... kořenová složka
  - `/train` ..... Složka se vším potřebným pro trénování a validaci
    - `/Pconv` ..... Složka s potřebnými moduly pro trénování pconv
    - `/Aot-gan` ..... Složka s potřebnými moduly pro trénování AOT-GAN
    - `/cel` ..... Složka obsahující ukázky dat pro trénování a validaci
  - `/app` ..... Složka obsahující všechny zdrojové soubory aplikace
    - `main.py` ..... Zdrojový soubor pro spuštění aplikaci
    - `Visual/` ..... Zdrojové soubory se zaměřením na frontend
    - `Logic/` ..... Zdrojové soubory se zaměřením na backend
    - `Models/` ..... Zdrojové soubory s převzatými modely
    - `Checkpoints/` ..... Soubory s vytrénovanými daty
  - `/README` ..... Obsahuje instrukce ke zdrojovým kódům
  - `/thesis.pdf` ..... Text práce
  - `/tex` ..... Zdrojové soubory zprávy

Příloha B

Plakát



Obrázek B.1: Plakát prezentující výsledek.