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

REKONSTRUKCE ŘÍDCE VZORKOVANÉHO OBRAZU POMOCÍ HLUBOKÉHO UČENÍ

RECONSTRUCTION OF SPARSE SAMPLED IMAGES WITH DEEP LEARNING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. HOANG ANH LE**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ROMAN JURÁNEK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Le Hoang Anh, Bc.**
Program: Informační technologie
Obor: Vývoj aplikací
Název: **Rekonstrukce řídce vzorkovaného obrazu pomocí hlubokého učení**
Reconstruction of Sparse Sampled Images with Deep Learning
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s možnostmi rekonstrukce řídce vzorkovaných obrazů (kde informace není ve všech pixelech)
2. Navrhněte a implementujte metodu pro rekonstrukci. Zaměřte se na multispektrální data.
3. Demonstrujte funkčnost implementované metody a vyhodnoťte její přesnost na vhodném datasetu.
4. Vytvořte prezentační materiály.
5. Zhodnoťte vaši práci a navrhněte možnosti pokračování.

Literatura:

- Dai et al, Adaptive Image Sampling using Deep Learning and its Application on X-Ray Fluorescence Image Reconstruction, IEEE Transactions on Multimedia, 2019

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

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

Vedoucí práce: **Juránek Roman, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem práce bylo zlepšit kvalitu rekonstrukce řídce vzorkovaných mikroskopických snímků pomocí neuronových sítí. V práci budou popsány různé přístupy k rekonstrukci obrazu. Budou zde popsaný i použité implementace, nad kterými bylo provedeno vyhodnocení z pohledu rekonstrukce a segmentace, která je právě jejich hlavní možnou aplikací.

Abstract

The main goal of this thesis was to increase reconstruction quality of sparse sampled microscopic images by using neural networks. The thesis will cover various approaches for image reconstruction and will also include descriptions of implementations, which were used. Implementations will be evaluated based on quality of reconstruction, but also based on segmentation, which could be their main possible application.

Klíčová slova

Neuronová síť, GAN, U-Net, rekonstrukce obrazu, segmentace, strojové učení

Keywords

Neural network, GAN, U-Net, image reconstruction, segmentation, machine learning

Citace

LE, Hoang Anh. *Rekonstrukce řídce vzorkovaného obrazu pomocí hlubokého učení*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Roman Juránek, Ph.D.

Rekonstrukce řídce vzorkovaného obrazu pomocí hlubokého učení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Romana Juránka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Hoang Anh Le
18. května 2021

Poděkování

Chtěl bych poděkovat svému vedoucímu za jeho ochotu pomoci a věnovaný čas. Dále bych chtěl také poděkovat své rodině za jejich podporu. Výpočetní zdroje byly poskytnuty projektem "e-Infrastruktura CZ" (e-INFRA LM2018140) v rámci programu Projects of Large Research, Development and Innovations Infrastructures.

Obsah

Kapitola 1

Úvod

Cílem této práce je najít nový způsob pro rekonstrukci řídce vzorkovaného obrazu na datech poskytnutých od firmy TESCAN Brno, s.r.o.[1], která se zabývá vytvářením snímků zkoumaných materiálů získávaných pomocí elektronových mikroskopů. Samotné snímání materiálu je velice časově náročné a proto je nezbytné zaznamenat jen specifické pozice a zbytek určit na jejich základě, aby se celý tento proces urychlil až na jednotky vteřin. Firma využívala jednodušší metodu pro zpětnou rekonstrukci, která by ale měla jít vylepšit pomocí strojového učení. Pro správné určení z jakých prvků se zkoumaný vzorek skládá a také za účelem určení, na jakých přesných pozicích se v daném vzorku nachází, je nezbytná kvalitní segmentace, která vychází právě z rekonstruovaných dat. Rekonstrukce pomocí později zmíněné výchozí metody však způsobuje nadsegmentaci, která komplikuje správné určení informací o vzorku tím, že je nutné pak zpětně spojit vícero vytvořených segmentů, které spolu sousedí a obsahují stejné složení prvků. Za tímto účelem by pak bylo vhodné použít a otestovat konvoluční neuronové sítě. Po průzkumu byla nalezena architektura U-Net, která byla aplikována přímo na podobný typ dat, a nebo také generativní adversariální sítě (GAN), které byly často aplikované na řešení problematiky rekonstrukce obrazu, avšak ne přímo na tento typ dat. Generativní sítě jsou ale známé tím, že umožňují vytvářet obrázky, které velice dobře napodobují cílové obrázky.

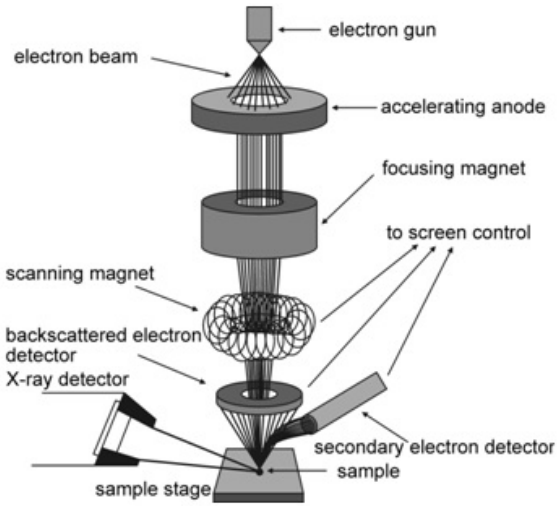
V kapitole 2 budou popsány základní informace o elektronové mikroskopii a jejich datech. Dále v kapitole 3 budou popsány data, které se budou v práci používat a také návrh postupu řešení. V kapitole 4 budou primárně popsány různé architektury neuronových sítí zaměřené na rekonstrukci obrazů. V kapitole 5 pak budou popsány reimplementace vybraných architektur a nebo z nich použité jejich klíčové principy. V kapitole 6 budou popsány použité metriky pro vyhodnocení a budou zde vyhodnoceny implementace architektur z pohledu rekonstrukce obrazu a také z pohledu segmentace.

[1] https://www.tescan.com/

Kapitola 2

Elektronová mikroskopie

Cílem elektronové mikroskopie [3] je získat přesné informace o složení a topologii zkoumaného vzorku. K tomu se používají elektronové mikroskopy, které na rozdíl od optických mikroskopů nepoužívají světelné paprsky k vytvoření snímku zkoumaného vzorku, ale místo toho využívají svazky elektronů. Elektronové mikroskopy pak umožňují větší zvětšení oproti optickým mikroskopům a tedy poskytují možnost zkoumat jemnější detaily vzorku. Elektronové mikroskopy lze rozdělit na dva typy, kde každý pracuje na jiném principu. První se nazývá *transmission electron microscope* (dále jen TEM) a druhý *scanning electron microscope* (dále jen SEM). V této kapitole bude rozebrána jen druhá varianta, ze které byla vytvořena použitá data pro tuto práci.

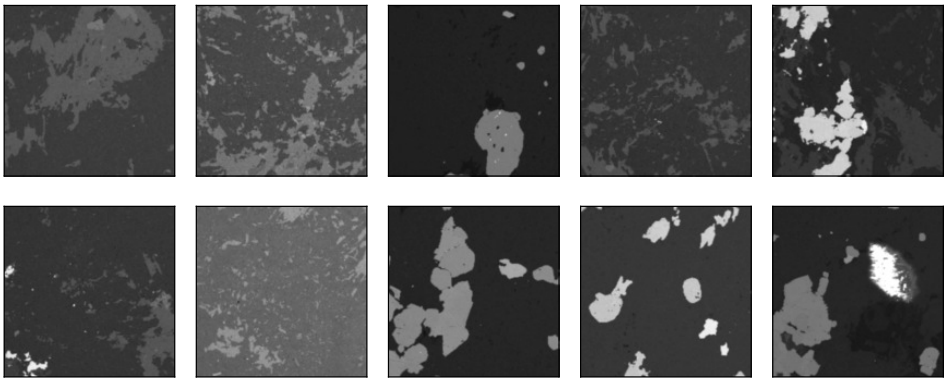


Obrázek 2.1: Ukázka SEM z jakých částí se skládá. převzato z odkazu [1].

Schéma SEM lze vidět na obrázku 2.1. Princip fungování spočívá v části *electron gun*, obsahující filament nebo špičku většinou z wolframu, jenž se poté zahřívá pomocí protékajícího proudu. To poté pomocí jevu termoemise vyzařuje svazky elektronů, které procházejí sadou různých čoček až ke zkoumanému vzorku. Při kontaktu svazku elektronů se vzorkem vznikají sekundární elektrony, zpětně rozptýlené elektrony (dále jen BSE) a rentgenové záření, které jsou zachyceny pomocí k nim příslušným detektorům.

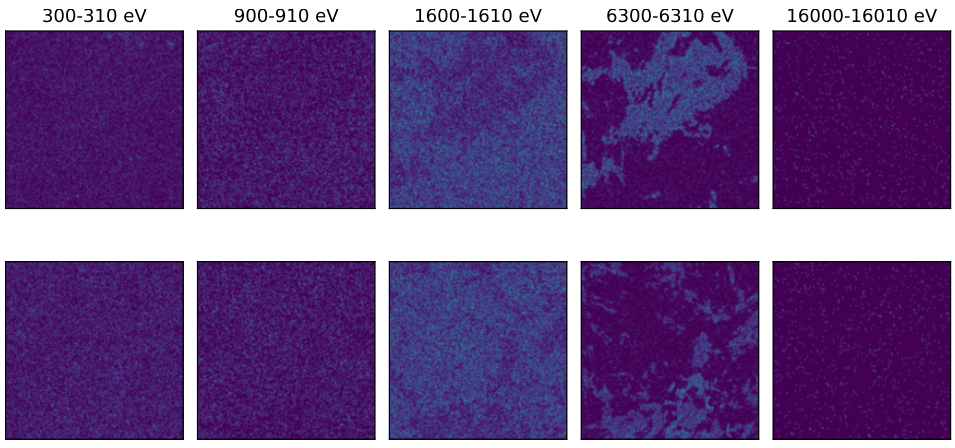
BSE vzniká tak, že se nad každým bodem vzorku vysílají svazky elektronů, které jsou odraženy zpět. To je důsledkem pružné srážky mezi vysílanými elektrony a atomy vzorku, která způsobí změnu trajektorie elektronů. Počet BSE je závislý na hmotnosti prvku, kde těžší

materiály, které budou mít většinou větší atomové jádro, jsou schopny odrazit zpět více elektronů než lehčí materiály. Z detekovaných BSE pak vzniká proud, který je naměřen a převeden do výstupního snímku. Ukázky lze pak vidět na obrázku 2.2, kde světlejší oblasti odpovídají těžšímu materiálu a tmavší oblasti lehčímu.

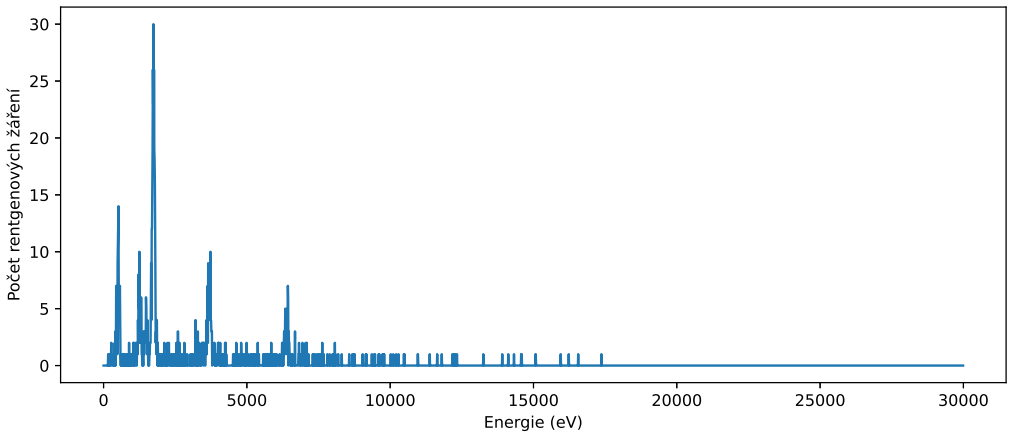


Obrázek 2.2: Ukázka některých BSE dat. Světlejší oblasti značí, že obsahuje těžší materiál a tmavší naopak lehčí.

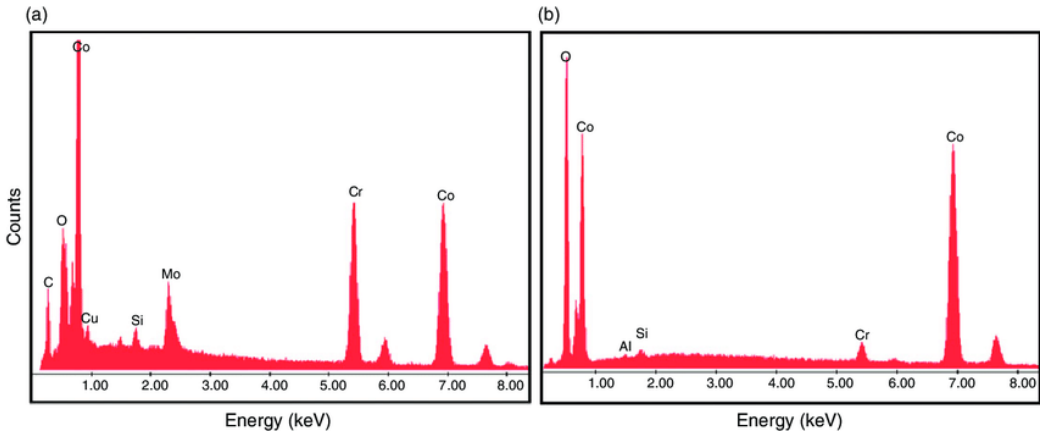
Zpracování rentgenových záření se nazývá *Energy-dispersive X-ray spectroscopy* (dále jen EDS), kde výstupem je pak prvková kompozice vzorku vyjádřena pomocí spektra. Spektrum EDS vzniká tak, že se opět nad každým bodem vzorku vysílají svazky elektronů podobně jak při BSE, avšak místo toho se po určitý čas provádí měření jednotlivých energií zachycených rentgenových záření, které jsou vytvořeny reakcí vzorku s elektrony. V průběhu se také měří počet záření, který je poté rozdělen do intervalů podle jejich velikosti energie. Pro ukázku lze na obrázku 2.3 vidět vybrané intervaly, kde má jeden tento interval rozsah 10 eV. Každý interval poté odpovídá jednomu kanálu v multispektrálních datech o stejných rozměrech jako má vzorek. Ukázku spektra v jednom bodě vzorku lze vidět na obrázku 2.4.



Obrázek 2.3: Na každém řádku je jiný vzorek a v sloupcích jsou pak ukázky EDS kanálů o určitých intervalech energie rentgenových záření. Světlejší barva pak označuje pozice s vyšším počtem rentgenových záření o velikosti energie patřící do daného intervalu.



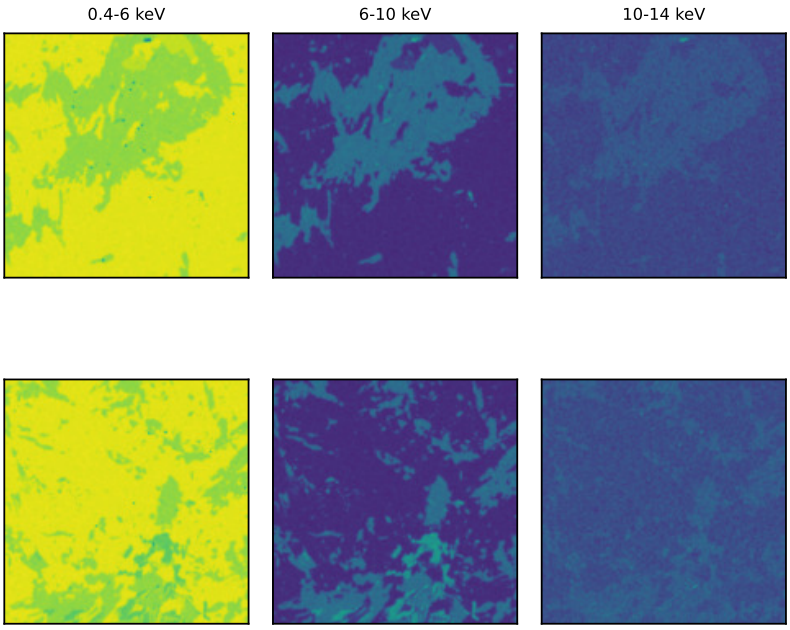
Obrázek 2.4: Ukázka spektra EDS, které vychází z dat EDS s celkovým počtem 3000 intervalů o rozsahu 10 eV, obsahující počet rentgenových záření s energii patřící do daného intervalu.



Obrázek 2.5: Ukázka 2 spekter EDS různých vzorků, kde ke každému vrcholu je určeno, ke kterému prvku patří. Převzato z odkazu [2].

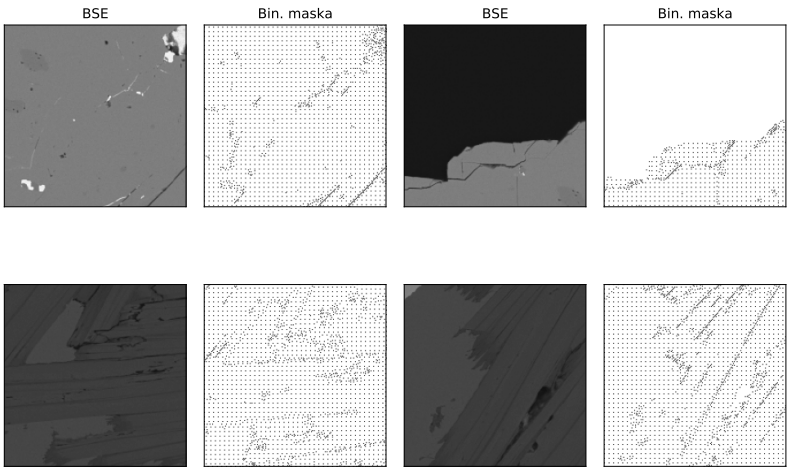
Z vrcholů spektra lze určit, jaké prvky obsahuje vzorek v určitém bodě, příklady lze vidět na obrázku 2.5. Avšak stanovení materiálu v každém bodě může způsobit chybnou identifikaci způsobenou nepřesnostmi během měření. Za tímto účelem je nutné rozsegmentovat vzorek na podoblasti stejných materiálů a teprve až z jejich kumulativního spektra stanovit přesný materiál. Korektní segmentace je ale vázaná na data EDS a jak lze vidět na obrázcích kanálů EDS, data obsahují velké množství šumu, které komplikují proces segmentace. Šum lze ale eliminovat pomocí sčítání určitých rozsahů kanálů, jak ukazuje obrázek 2.6.

BSE lze získat celkem rychle, na druhou stranu získat EDS může trvat velice dlouhou dobu. Například pokud je potřeba zjistit informace o vzorku s rozměry 200×200 bodů, bude nutné nad každým bodem provádět měření přibližně 5 ms. Celkový čas měření pak odpovídá přibližně 3 minutám a 20 sekundám, což je nepřijatelně dlouho pro tak malé rozměry vzorku. Pro vyřešení tohoto problému nelze zredukovat čas strávený nad každým bodem, protože by to ještě více znepřesnilo výsledky měření. Z toho důvodu se provádí měření jen na určitých pozicích definovaných s pomocí binární masky. Tato binární maska vzniká na



Obrázek 2.6: Ukázka kanálů EDS po rozšíření intervalů energie, které pak umožní eliminovat šum. Na každém řádku je jiný vzorek.

základě předběžné segmentace BSE. To je umožněno tím, že na rozdíl od EDS lze BSE měřit nad každým bodem díky rychlosti dané metody. Obrázky přiblížených výřezů binarních masek a příslušných BSE lze vidět na obrázku 2.7. Až dále z EDS naměřených na daných pozicích se vytváří zrekonstruované EDS, které je definováno ve všech bodech, pomocí metody popsané v sekci 4.1.



Obrázek 2.7: Ukázka přiblížených výřezů BSE a k nim příslušných binárních masek. Černé pixely pak představují pozice kde bylo provedeno měření EDS. Na každém řádku je vidět jiný vzorek.

Kapitola 3

Analýza problému

Následující sekce bude zaměřena na vstupní data, která mi byla poskytnuta a na nutné úpravy nad nimi, aby byla vhodná pro trénování neuronových sítí. Dále bude rozebrán postup pro řešení problematiky rekonstrukce obrazu.

3.1 Vstupní data

Vstupní data se skládají ze 2 hlavních částí, kterými jsou BSE a spektra EDS. Tyto dvojice dat se pak dále člení podle EDS na EDS definované v každém bodě a vzorkované EDS. V druhém případě je tedy ještě dostupná binární maska určující pozice validních pixelů, kterých je celkem jen 4.2% z celkového počtu pixelů. Rozměry BSE odpovídají velikosti $x \times x \times 1$ pixelů, kde x značí šířku a zároveň výšku. Velikost ale není stejná u všech dat z poskytnuté datové sady. Ke každému BSE pak přísluší EDS o rozměrech $x \times x \times 3000$. BSE nabývá hodnot z rozsahu $\langle 0, 1 \rangle$ a v obrázku 2.2 pak světlejší oblasti odpovídají hodnotám blíže 1 a naopak tmavší odpovídají hodnotám blíže 0. Je dostupno celkem 110 dat 1. typu a jejich rozložení velikosti lze vidět v tabulce 3.1. Na 10 datech 2. typu dosahují až rozměrů 1000 pixelů na výšku i šířku. Kromě toho jsou ještě 3 vzorky dat, které mají dostupné vzorkované EDS, binární masku a BSE, ale mají navíc dostupnou *ground-truth* (dále jen GT) segmentační mapu, která už ale odpovídá přesné segmentaci do podoblastí stejných materiálů. Mapa obsahuje také oblasti s nevalidními pixely, protože se nepodařilo určit k jakému segmentu mají být přiřazeny. Také jsou k dispozici 2 referenční segmentační mapy, kde 1. vznikla metodou firmy TESCAN a 2. byla vytvořena metodou GMRF zmíněnou v článku [10].

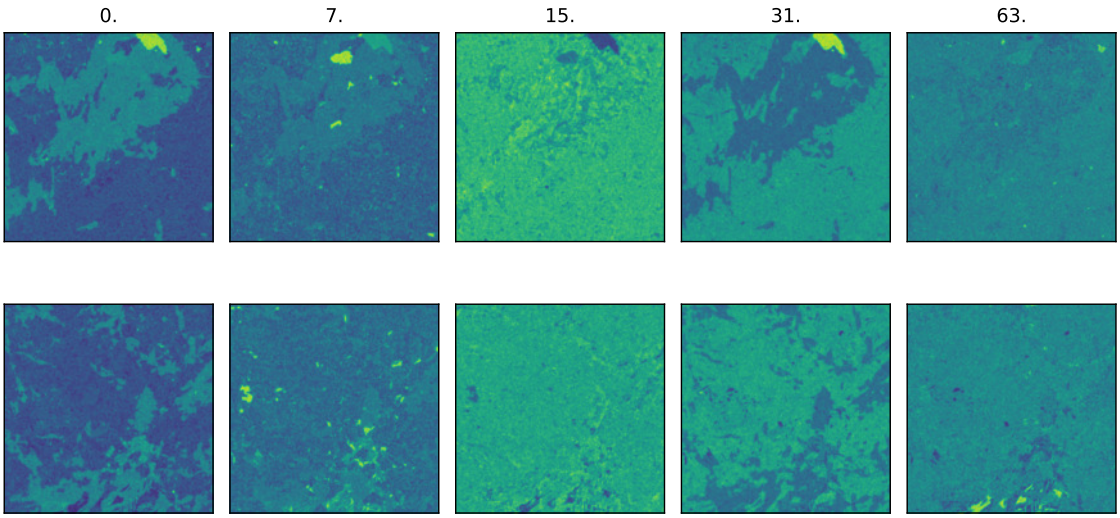
Tabulka 3.1: Tabulka možných rozměrů dat v pixelech, které mají EDS definované ve všech bodech a jejich počet o daných rozměrech.

Velikosti (px)	Počet
150	72
214	12
300	3
375	13
500	10

BSE lze díky omezenému rozsahu hodnot použít bez úprav jako součást vstupu sítí pro rekonstrukci EDS, avšak samotné EDS není vhodné použít jako vstup a bude nutné ho

upravit nebo zpracovat. Důvodem je, že použití celého EDS, které obsahuje 3000 kanálů oproti běžným 3 kanálům, by pro konvoluční sítě znamenalo i velký počet trénovatelných parametrů a k tomu navíc většina kanálů obsahují jen šum. Tyto problémy by šly eliminovat sčítáním vybraných kanálů, jak je ukázáno například na obrázku 2.6. V tom případě by ale bylo nutné normalizovat hodnoty EDS do určitého omezeného rozsahu hodnot, protože obsahují počty rentgenových záření.

Další alternativou, ke které mi byla poskytnuta neuronová síť, je vytváření *embedding*u, který obsahuje na rozdíl od spektra EDS o 3000 hodnotách jen 64 hodnot. Síť přiřazuje podobný vektor hodnot mezi blízkými materiály a naopak velice odlišné mezi vzdálenými materiály. Rozměry upravených dat pak odpovídají velikosti $x \times x \times 64$ a jednotlivé hodnoty *embedding*u nabývají hodnot z rozmezí $\langle -1, 1 \rangle$. Ukázky kanálů lze vidět na obrázku 3.1. Avšak jak lze vidět i vytvořený *embedding* obsahuje nežádoucí šum.



Obrázek 3.1: Ukázka kanálů *embedding*u EDS, kde na každém řádku je jiný vzorek a kde sloupce reprezentují jinou dimenzi s daným indexem z celkového počtu 64 hodnot. Hodnoty ve všech pozicích pak náleží do intervalu $\langle -1, 1 \rangle$.

3.2 Návrh řešení

Cílem práce je vytvořit metodu na rekonstruování obrazu, který pak lze použít k vytvoření segmentací. Místo toho aby se jako vstupní data používalo celé spektrum EDS, budou místo nich použity jejich *embedding*y (dále jen EDS, pokud nebude specifikováno). Ten se tedy vytvoří jak pro spektra EDS definované ve všech bodech tak i pro vzorkovaná data. Dále bude nezbytné zjistit, jaké metody se používají pro rekonstrukci obrazů. Mohou to být metody zaměřeny specificky na podobná data nebo také na obecné obrázky. Poté vybrané z nich reimplementovat a případně upravit. Metody si budou muset poradit s nízkým procentem pixelů s validní hodnotou a budou také muset zpracovávat jiný typ dat, kde běžné obrázky mají 3 kanály, zatímco EDS má 64 kanálů. Výsledné metody pak budou použity pro vyhodnocení a budou navzájem porovnány mezi sebou i se základní metodou, která bude odpovídat metodě používané firmou TESCAN. Způsob trénování a tvorby datových sad bude popsán v sekci 5.1.

Kapitola 4

Současná řešení

V následující kapitole budou rozebrány všechny použité metody a modely z teoretického pohledu.

Problém *inpainting* [4, 12] spočívá ve vytvoření chybějící části nebo obnovení poškozené části obrázků, aby výsledný obrázek vypadal realisticky. Rekonstrukce může řešit problémy mezi které patří například odstranění šumu nebo odstranění textu, který obraz překrývá. Metody lze rozdělit na 2 hlavní skupiny na základě jejich vstupu. 1. skupinou jsou obrazy s regulární maskou, kde je cílem rekonstruovat chybějící prostřední obdélníkovou část, která se nachází vždy na stejné pozici obrazu o stejné velikosti. Pak existuje 2. skupina obsahující obrazy s iregulární maskou, kde mohou být rekonstruované oblasti libovolně veliké a také různých tvarů. Data použitá v této práci pak odpovídají 2. skupině.

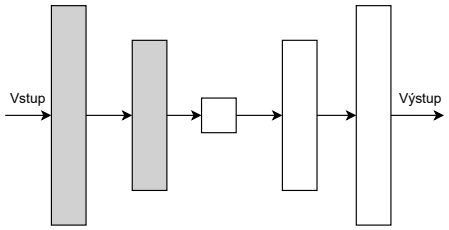
4.1 Rekonstrukce za pomocí jednoho nejbližšího souseda

Jednoduchá metoda, která není založená na strojovém učení. Spočívá v postupném určení pixelů s neznámou hodnotou pomocí jakéhokoliv jednoho sousedního pixelu s již známou hodnotou. Hodnota sousedního pixelu se pak pouze zkopíruje do sousedních pixelů s ještě neurčenou hodnotou.

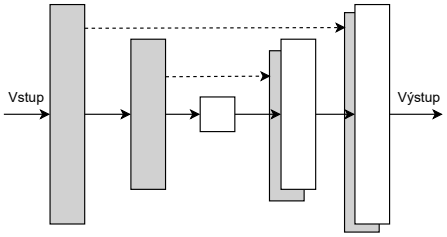
4.2 Autoenkodér

Autoenkodér [4] je architektura umělé konvoluční neuronové sítě, která se skládá ze dvou hlavních částí. Architekturu lze vidět na obrázku 4.1.

Cílem první části zvané enkodér je vytvořit reprezentaci vstupního obrázku s výrazně menšími rozměry. Motivací pro to je, aby se síť naučila rozpoznat vysoko úrovňové příznaky o vstupním obrázku. Takto získané příznaky umožňují robusnost vůči poškození nebo šumu. Určení vhodného poměru rozměrů mezi vstupním obrázkem a výstupní reprezentací zvanou *bottleneck* v enkodéru je závislé na rozměrech vstupního obrázku, pokud je snahou dosáhnout co nejlepších výsledků rekonstrukce. V článku byl zmíněn rozsah od 3 do 6 na obrázcích o velikosti 32×32, kvůli obtížnosti rozpoznat koherentní objekty a elementy. Oproti tomu pokud by zmíněná síť byla natrénována na datové sadě *ImageNet*, poměr mezi vstupem a výstupem enkodéru by mohl dosahovat hodnoty větší než 12, kde průměrná velikost obrázků je větší a je tedy jednodušší extrahovat vysokoúrovňové příznaky na rozdíl od předchozí datové sady. Daná část se skládá z konvolučních vrstev a k postupnému zmenšení šířky a výšky vstupu dochází pomocí **Maxpooling** vrstev a nebo pomocí nastavení



Obrázek 4.1: Ukázka architektury autoenkodér. Šedé bloky značí výstup po konvolučních vrstvách v enkodéru a bílé značí po konv. vrstvách dekodéru. Inspirováno na základě článku [8].



Obrázek 4.2: Ukázka architektury U-Net. Šedé bloky značí výstup po konv. vrstvách v enkodéru, které se konkatenují na výstup konv. vrstev v dekodéru. Inspirováno na základě článku [8].

parametru posuvu v konvoluční vrstvě na hodnotu větší než 1. Počet kanálů se ale po každé konvoluci postupně zvyšuje. V článku také zmiňují, že konvoluce s větším jádrem o velikosti 5 nebo 7 lze nahradit za několik menších o velikosti 3 se zachováním velikosti vnímaného prostoru. A právě s tímto zahrazením dosáhli lepších výsledků.

Pomocí druhé části zvané dekodér se poté z *bottleneck*u vytvoří výstup, který může být o stejné velikosti jako je vstup a nebo o velikosti chybějícího výřezu. Dekodér se podobně jako enkodér skládá z konvolučních vrstev, ale na rozdíl od enkodéru je jeho snahou vytvořit ze zkomprimované reprezentace postupným zvětšováním rozměrů a snižováním počtu kanálů výstup. Výstup se ale liší v závislosti na implementaci.

V článku [4], který se zabývá doplněním chybějící části do obrázku, je vstupem obrázek, který má odebranou vnitřní čtvercovou část. Výstupem sítě je pak už samotný chybějící výřez. Mezi enkodérem a dekodérem také dávali plně propojené vrstvy nebo konvoluční vrstvy.

V článku [5] popisují 2 oddělené sítě, kde první síť se zabývá tvorbou masky pro obrázek s cílem odebrat co nejvíce pixelů ze vstupního obrázku, ale zachovat možnost rekonstrukce do původní podoby. Druhá se právě pak zabývá rekonstrukcí a výstupem jsou obrázky o stejné velikosti, jako je vstup. Dále je zajímavostí, že propojení mezi enkodérem a dekodérem je vyřešeno pomocí vrstvy, kterou v článku nazývají *channel-wise fully connected layer* [19]. Jedná se plně propojenou vrstvu, kde se počet vstupů rovná počtu výstupů s tím rozdílem, že v tomto případě nejsou všechny vstupy napojeny na všechny výstupy, ale každý vstup je v jednom kanálu připojen na všechny výstupy v daném kanálu. Pro vstup o rozměrech $n \times n$ s počtem kanálů rovným m je pro běžnou plně propojenou vrstvu, která má počet výstupů roven počtu vstupů, počet trénovatelných parametrů roven n^4m^2. Pro zmíněnou variantu vrstvy je počet parametrů roven n^4m. Motivací bylo snížit počet parametrů a tím urychlit průběh trénování. Nevýhodou dané architektury ale je, že se nejedná o plně konvoluční architekturu a nelze jako vstup použít libovolné velikosti.

Enkodér a dekodér se trénují dohromady a cílem je minimalizovat rozdíl mezi skutečným výstupem sítě a očekávaným výstupem, tohle je pak vyjádřeno snahou minimalizovat chybovou funkci, kterou může být střední kvadratická chyba[1] (MSE nebo také L_2), která odpovídá vzorci:

$$L_2 = \frac{1}{n}\sum_{i=1}^{n}(I^{out} - I^{gt})^2,$$

[1] https://en.wikipedia.org/wiki/Mean_squared_error

nebo také střední absolutní chyba[2] (MAE nebo také L_1), která odpovídá vzorci:

$$L_1 = \frac{1}{n}\sum_{i=1}^{n}|I^{out} - I^{gt}|.$$

Podle článku [8] sítě natrénované s MAE vytváří výstupy, které jsou méně rozmazané oproti sítím natrénovaným s MSE.

4.3 U-Net

U-Net [20, 8] je architektura podobná autoenkodéru, avšak hlavním rozdílem je přidání dalších propojů nazývaných *skip-connection*, které vedou z výstupů konvolučních vrstev v části enkodéru do vstupů konvolučních vrstev v dekodéru, které se konkatenují s výstupem z předchozí konvoluční vrstvy v dekodéru. Architekturu lze vidět na obrázku 4.2.

Motivací pro to je, že vstup a výstup mohou sdílet některé nízko úrovňové informace a přes samotný autoenkodér by musely procházet veškeré informace přes všechny jednotlivé konvoluční vrstvy enkodéru až do *bottleneck*u, které by kvůli omezené velikosti nemusel zahrnovat všechny důležité informace. Je tedy nutné tyto informace nějakým způsobem přenést do dekodéru, ke kterým právě slouží zmíněné propoje, které přeskakují *bottleneck* a vedou přímo do konvolučních vrstev v dekodéru.

Způsob trénování je pak stejný jak pro autoenkodér popsaný v sekci 4.2. Architektura se pak dále používá například pro segmentaci obrazů [20] a nebo se také často používá jako generátor pro generativní adversariální sítě (dále jen GAN) popsané v následující sekci.

Architekturu také použili v článku [21], ve kterém ji využívají k experimentům nad různými problémy obrazového zpracování jako je rekonstrukce hyperspektrálních dat. Síť ale nebyla natrénována běžným způsobem, který používá datovou sadu obsahující GT rekonstruovaného obrazu kvůli nedostatku dat. Místo toho využívají principu *Deep image prior* popsaného v článku [22]. Využívají náhodně inicializovanou síť s fixním vstupem, který je náhodný šum a očekávaným výstupem je pak vzorkovaný obraz. Chybovou funkcí je pak:

$$L = ||(x - x_0) \odot m||^2,$$

kde x je výstup sítě, x_0 je vzorkovaný obraz a m je binární maska.

V článku dále také zkouší využívat místo bežných 2D konvolucí 3D konvoluce, kde motivací bylo pokus o zachování informací, které se mohly ztratit hned po 1. konvoluční vrstvě v enkodéru. Výsledky však ukázaly, že s použitím 3D konvoluce dosáhli horších výsledků a zpracování stejných dat oproti sítě využívající 2D konvoluce je paměťově a časově mnohem náročnější, kvůli nutnosti pracovat s tenzory, které mají přidanou další dimenzi.

4.3.1 Architektura využívající partial konvoluce

V článku [14], který je zaměřený na problém rekonstrukce obrazu s iregulární maskou, popisují možný nedostatek běžných konvolucí. Faktem je, že prováděná konvoluce nebere v potaz to, jestli se nachází zrovna v oblasti obrázku, který je nutný rekonstruovat nebo v oblasti, kde jsou už pixely určené. To podle článku mohlo vést na rozmazané výstupy a také na možnou barevnou nesouladnost. K řešení tohoto problému představili v článku

[2] https://en.wikipedia.org/wiki/Mean_absolute_error

vrstvu, která se nazývá *partial* konvoluční vrstva. Jedná se o konvoluční vrstvu, která zpracovává jen už určené pixely a se kterou je pak zároveň spojená operace, která postupně upravuje binární masku určující, na kterých pozicích už je známá hodnota a na kterých ještě ne. Celý tento postup lze rozdělit do dvou hlavních kroků. Mějme váhy konvolučního jádra označené jako W, a k nim odpovídající bias b. Mějme dále vstupní příznaky X, které budou aktuálně zpracovávány a k nim příslušnou binární masku M. První krok lze vyjádřit následovně, kde x' představuje novou hodnotu po konvoluci na aktuální pozici:

$$x' = \begin{cases} W^T(X \odot M)\frac{\text{suma(l)}}{\text{suma}(M)} + b, & \text{pokud suma(M)} > 0. \\ 0, & \text{jinak.} \end{cases}$$

$\odot$ je maticové násobení po složkách, **l** je pak matice obsahující 1 o stejných rozměrech jako je **M**. Vynásobení vstupních příznaků s binární maskou způsobí, že se budou brát v potaz jen už určené hodnoty. Jako kompenzaci za chybějící pixely obsahuje rovnice ještě poměr sum, který v případě že maska ještě obsahuje chybějící pozice, bude poměr větší než 1 a tím navýší danou hodnotu x'.

V 2. kroku po každé konvoluci je pak nová hodnota m' v masce na stejné pozici, kde byl proveden aktuální krok konvoluce, získána následovně:

$$m' = \begin{cases} 1, & \text{pokud suma}(\mathbf{M}) > 0. \\ 0, & \text{jinak.} \end{cases}$$

Po postupném aplikování těchto konvolucí bude maska obsahovat jen samé 1 a výstupní tenzor bude tedy obsahovat jen pixely s platnými hodnotami.

Jako architekturu generátoru opět použili architekturu U-Net, avšak konvoluční vrstvy nahradili za vrstvy s *partial* konvolucemi. Jako část chybové funkce použili L_1, kde však ale více penalizovali oblasti, které mají být rekonstruovány. Jako další části využili také *Perceptual* chybu, *Style* chybu a také *Total variation* chybu.

Perceptual chyba byla poprvé představena v článku [9] a slouží k tomu, aby se příznaková reprezentace rekonstruovaného obrazu co nejvíce podobala referenčnímu obrazu. Druhá podobná varianta nazývaná *Style* [6] chyba se pak snaží o to, aby se barvy a textury rekonstruovaného a skutečného obrazu opět mezi sebou shodovaly. Tyto chyby jsou získány pomocí výstupů vybraných vrstev předtrénovaného modelu **VGG19** nebo **VGG16** na datové sadě **ImageNet**. Porovnávají se výstupy modelu **VGG** pro vstupy, kterými jsou rekonstruovaný výstup a očekávaný výstup generátoru. Vzorec pro výpočet *Perceptual* chyby odpovídá následujícímu vzorci:

$$\mathcal{L}_{perceptual} = \sum_{l=1}^{L} \frac{||\phi_l^{I_{out}} - \phi^{I_{gt}}||_1}{N_{\phi_l^{I_{gt}}}},$$

kde L odpovídá počtu výstupních vrstev, $\phi_l^{I^{out|gt}}$ značí výstup pro rekonstruovaný obraz a očekávaný výstup. $N_{\phi_l^{I_{gt}}}$ odpovídá počtu prvků v dané vrstvě **VGG** sítě. Pro *Style* chybu pak platí následující vzorec:

$$\mathcal{L}_{style} = \sum_{l=1}^{L} \frac{1}{C_l C_l} \frac{||(\phi_l^{I_{out}})^T \phi_l^{I_{out}} - (\phi^{I_{gt}})^T \phi^{I_{gt}}||_1}{N_{\phi_l^{I_{gt}}}},$$

kde značení je stejné jako výše, ale ještě navíc přibylo C_l, které značí počet kanálů daného výstupu.

Snahou *Total variation* chyby je zamezit možný šum a vizuální artefakty, které mohou být zavedeny pomocí *Style* a *Perceptual* chyby. K výpočtu této chyby využívají sumu absolutních vzdáleností od sousedních pixelů ve výstupním obrázku R. To je vyjádřeno následující rovností:

$$\mathcal{L}_{tv} = \sum_{(i,j)\in R,(i,j+1)\in R} \frac{||I^{i,j+1}_{comp} - I^{i,j}_{comp}||_1}{N_{I_{comp}}} + \sum_{(i,j)\in R,(i+1,j)\in R} \frac{||I^{i+1,j}_{comp} - I^{i,j}_{comp}||_1}{N_{I_{comp}}}.$$

Předchozí složky chybové funkce jsou vynásobeny konstantami, které dále ovlivňují velikost vlivu daných složek.

4.4 Generativní adversariální sítě

Cílem Generativních adversariálních sítí [23, 19, 8] (dále jen GAN) je vytvářet výstupy, které by měli být k nerozeznání od stanovených dat. To je zajištěno architekturou GAN, skládající se ze dvou neuronových sítí nazývané generátor a diskriminátor, které se ale trénují zároveň. Cílem generátoru je vytvářet falešná data se snahou, aby vypadala co nejdůvěryhodněji za účelem obelstít diskriminátor. Cílem diskriminátoru je pak rozeznávat falešná data od skutečných. Základní GAN je nepodmíněný a nevyužívá žádné další informace pro obě sítě. Generátor jako vstup využívá jenom náhodný šum a diskriminátor využívá jenom samotný výstup generátoru. Existuje však verze, která se jmenuje *conditional* GAN, umožňující přidat další informace o vstupu do obou sítí. Tento typ je pak vhodný pro problémy převodu z jednoho obrazu do jiného.

Existuje mnoho variant GAN architektur, každá pro řešení jiných problémů počítačového vidění, ale následující části budou zaměřeny na architektury převádějící vstupní obraz na jiný.

4.4.1 Pix2Pix

V článku [8] popisují architekturu GAN, kterou lze aplikovat na mnoho problémů převodu obrazu do jiného, mezi které patří například vytvoření obrazů ze zadaných hran, obarvování obrázků nebo také právě rekonstrukce obrazu. V článku poukazují také na nedostatky běžných konvolučních sítí (CNN), v případě že by se aplikovaly samostatně na tyto úlohy. Snahou CNN je minimalizovat Euklidovu vzdálenost mezi výstupem sítě a očekávaným výstupem. Vzdálenost se minimalizuje pomocí zprůměrování nad všemi možnými výstupy sítě. Důsledkem tohoto ale je, že jsou výstupy rozmazané. Pro získání ostřejších výstupů sítě jsou nutné expertní znalosti.

Obecně cílem generátoru GAN architektury je obelstít diskriminátor, ale z předchozích přístupů zjistily, že je vhodné přidat do objektivní funkce generátoru také L_1 nebo L_2 chybu, aby se výstupy generátoru více podobaly očekávaným výstupům z pohledu těchto chybových funkcí. Součástí vstupu GAN generátoru by měl být také šum, aby negeneroval deterministický výstup, ale z provedených experimentů došli k závěru, že při přidání šumu jako vstup se síť pak naučila ignorovat daný šum a ve finální verzi místo toho, aby dávali na vstup i šum, se ho snažili zavést pomocí *dropout* vrstev, které použili jak během trénování tak i testování. Avšak i po jejím použití vypozorovali jen malé rozdíly mezi výstupy sítě nad stejným vstupem.

K získání ostřejších výstupu z generátoru právě slouží diskriminátor, který se snaží rozpoznat skutečný obraz od uměle vytvořeného. Aby se však diskriminátor zaměřil na vysoko

frekvenční strukturu obrázku, omezují rozsah odkud může čerpat informace do menších výřezů. Z toho důvodu použili architekturu diskriminátoru, která se nazývá **PatchGAN**, jehož cílem je ohodnocovat strukturu jen na úrovni menších výřezů. V článku provedli experimenty z různými velikostmi těchto výřezů, které lze nastavit pomocí počtu konvolučních vrstev měnící velikostí pomocí parametru posuvu nebo lze také provešt změnu velikosti pomocí **Maxpooling** vrstev. Velikost vnímaného výřezu může být také ovlivněna velikostí konvolučního jádra.

Při postupném zvětšování velikosti výřezů na velikosti 1×1, 16×16, 70×70 a 286×286, kde poslední velikost odpovídá velikosti celého obrázku, dostávali ostřejší výstupy až do chvíle než dosáhli velikosti celého obrázku, kdy se výsledek o proti předchozímu výřezu zhoršil. To si vysvětlují tím, že při použití výřezu o velikosti celého obrazu se do sítě zavede mnohem větší počet parametrů a natrénovat ji je pak náročnější. Výhodou architektury **PatchGAN** je to, že se jedná o plně konvoluční architekturu, takže může přijímat na vstupu různé velikosti obrázků. Jako poslední vrstvu použili konvoluční vrstvu redukující počet kanálů na 1 s aktivační funkcí **sigmoid**.

Trénování se sice provádělo zároveň, ale chybové funkce jsou pro obě podsítě počítány jiným způsobem. Diskriminátor byl natrénován jako binární klasifikátor s chybovou funkcí *binary crossentropy*[3]. Pro diskriminátor použili jako vstup kombinaci vstupu pro generátor a výstupu z generátoru nebo očekávaného výstupního obrázku z generátoru. Pro trénování se pro výstup z generátoru předpřipraví tenzor o stejných rozměrech jako je výstup diskriminátoru obsahující jen samé 0 a to stejné se provede i pro očekávané výstupy, ale místo toho budou obsahovat jen samé 1.

Generátor využívá chybovou funkci, která počítá rozdíl pixelů mezi vstupem a výstupem, ale navíc využívá vypočtenou chybu z diskriminátoru pouze nad umělými výstupy vytvořené generátorem. To lze vyjádřit následujícím vzorcem:

$$L_G = L_D + \lambda \times L_{1|2},$$

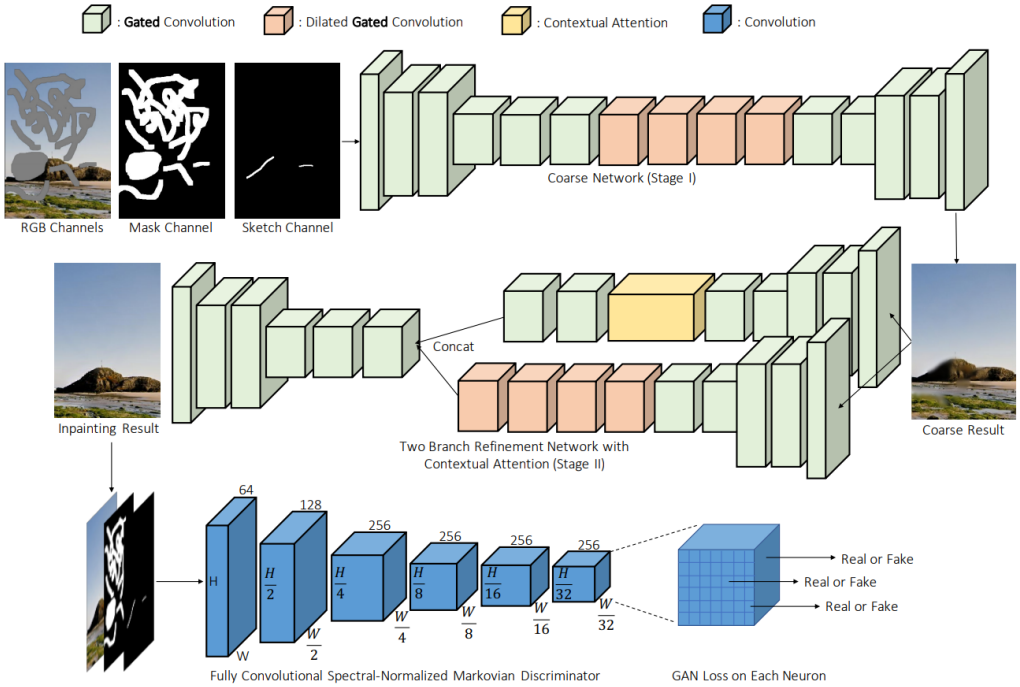
kde L_D vyjadřuje adversariální chybu a $L_{1|2}$ vyjadřuje vypočtený rozdíl mezi vstupem a výstupem za pomocí chybových funkcí zmíněných v sekci 4.2, který se vynásobí určitou konstantou λ. V článku provedli expermimenty s různými hodnotami λ od 0 do 100, kde došli k výsledku, že s $\lambda = 0$ získali na výstupu ostré výstupy, avšak obsahovaly artefakty. Ty se jim ale podařilo eliminovat s nastavením $\lambda = 100$.

4.4.2 Architektura využívající gated konvoluci

V článku [26], který je také zaměřený na rekonstrukci obrazu, představili *gated* konvoluci jako vylepšenou variantu *partial* konvoluce. Motivací pro tuto vrstvu je, že *partial* konvoluce masku pro následující vrstvy určuje jen heuristicky pomocí již určených pozic. Další nevýhodou také je, že maska je binární a pevně tedy určuje, kde se nachází validní/nevalidní pixel.

Místo toho se *gated* konvoluční vrstva dynamicky učí masku a místo toho, aby obsahovala jen hodnotu 0 nebo 1, může nabývat hodnoty z rozmezí $\langle 0, 1 \rangle$. Maska je dále dostupná zvlášť pro všechny kanály na rozdíl od *partial* konvoluce, kde je maska stejná pro všechny kanály. Pro její definování mějme 2 konvoluční jádra W_g a W_f. Pro získání výstupní hodnoty x' platí následující:

[3] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy



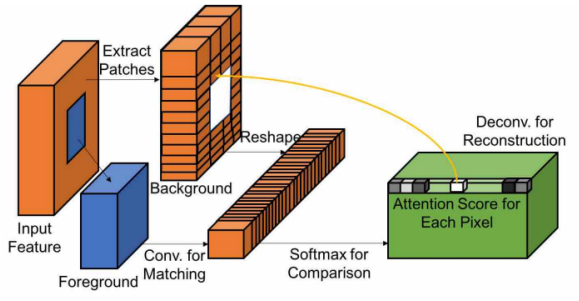
Obrázek 4.3: Ukázka architektury využívající *gated* konvoluci, převzato z článku [26].

$$
\begin{aligned}
G_{y,x} &= W_g.X, \\
F_{y,x} &= W_f.X, \\
x' &= \phi(F_{y,x}) \odot \sigma(G_{y,x}),
\end{aligned}
$$

kde ϕ může být aktivační funkce jako je **ReLU**, **Leaky ReLU** a podobné a σ pak odpovídá aktivační funkci **sigmoid** pro vytváření masky. Nevýhodou této vrstvy ale je, že bude obsahovat větší počet parametrů oproti běžné konvoluci.

Použitou architekturu lze vidět na obrázku 4.3. Generátor obsahuje 2 autoenkodéry, kde v 1. vytváří hrubě rekonstruovaný obraz a ve 2. už vytvářejí ostrý výstup. Všechny konvoluční vrstvy jsou nahrazeny *gated* konvolucemi a v obou autoenkodérech využívají konvoluce s dilatací na nejnižší úrovni, které umožňují zachytit vzdálenější závislosti. Ve 2. fázi rozdělují nejnižší vrstvy enkodéru do dvou větví, kde jedna z nich také obsahuje *contextual attention*.

Konvoluce jsou schopny získat informace jen z menšího okolí, a proto byl použit *attention* mechanismus, který slouží k zachycení vzdálených závislostí. Varianta zvaná *contextual attention* [19] pak z oblastí s určenými hodnotami vytváří menší výřezy, které pak využívají jako konvoluční jádra. Z oblastí, které mají být rekonstruovány jsou také vytvářeny výřezy a pak jsou na nich provedeny konvoluce se všemi vytvořenými jádry. Hodnoty podobnosti jsou dále upraveny pomocí *softmax* aktivační funkce a tím se získá *attention* mapa pro každý pixel vstupu. Konvoluční jádra a *attention* mapa jsou pak použity pro rekonstrukci chybějících oblastí. Tento přístup ale v článku považují za vhodný pouze pro čtvercové chybějící oblasti a není tedy schopen dobře generalizovat nad ireguláními maskami. Princip fungování této vrstvy lze vidět na obrázku 4.4.



Obrázek 4.4: Ukázka *contextual attention* vrstvy, převzato z článku [19].

Diskriminátor je **PatchGAN**, ve kterém navíc využívají spektrální normalizaci na konvolučních vrstvách. Spektrální normalizace [16] je normalizační způsob za účelem stabilizace trénování architektury GAN. Snahou je normalizovat hodnoty vah **W** všech jednotlivých vrstev sítě, aby pro Lipschitzovo omezení platilo $\sigma(W) = 1$:

$$\bar{W}_{SN}(W) := W/\sigma(W).$$

Spektrální norma $\sigma(W)$, použitá pro regularizaci jednotlivých vrstev by měla být největší singulární hodnota vah **W**, avšak nalezení této hodnoty je časově náročné a místo jeho přesného určení se provádí rychlá aproximace.

Na výstupu diskriminátoru je konvoluční vrstva s 256 filtry bez aktivační funkce. Chybovou funkcí, kterou použili pro její natrénování je *hinge* chyba. Chybová funkce pro diskriminátor je následující:

$$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}\left[ReLU\left(-1 + D\left(x, y\right)\right)\right] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}\left[ReLU\left(-1 - D\left(G\left(z\right), y\right)\right)\right],$$

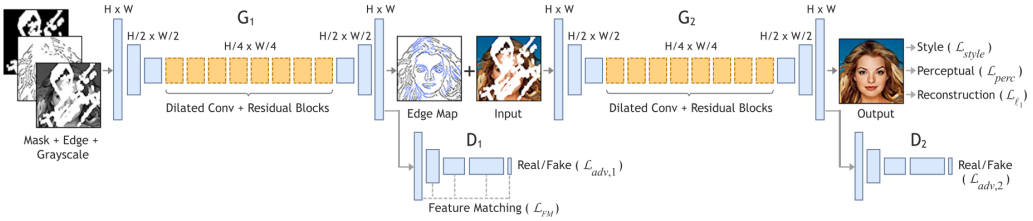
kde x je očekávaný výstup generátoru a z je vstup pro generátor. V chybové funkci lze nahradit **ReLU** také za minimum v porovnáním s 0 nebo za jinou aktivační funkci. Pro adversariální chybu 2. fáze generátoru platí:

$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D\left(G\left(z\right), y\right).$$

Jako součást chybové funkce pro 1. fázi a 2. fázi generátoru už pak použili dále jen L_1 chybu. Nevyužívají další chybové funkce jako je *Perceptual* chyba a to odůvodnili tak, že informace na úrovni menších výřezů už je obsažena v diskriminátoru. Součástí vstupu o celkem 5 kanálech použili binární masku, vzorkovaný obraz, ale také vstup, který se nazývá *sketch*. *Sketch* umožňuje upravit vygenerovaný výstup na základě dokreslených čar nacházejících se uvnitř oblastí, které se mají rekonstruovat.

4.4.3 EdgeConnect

V článku [15], který se zabývá rekonstrukcí mikroskopických snímků, využívají metodu s rekonstrukcí hran. Použitou architekturu lze vidět na obrázku 4.5 a vychází z článku [17], ve kterém využívají opět dva autoenkodéry jako generátor, kde 1. má vstup obsahující binární masku, obraz převedený do šedotónu a jeho hrany, které byly detekovány pomocí Cannyho hranového detektoru. Výstupem je poté rekonstruovaná mapa hran, kde GT hran je získán



Obrázek 4.5: Ukázka architektury EdgeConnect, převzato z článku [17].

stejným způsobem jako byl vstupní obraz hran. K 1. autoenkodéru je přiřazen diskriminátor, který dostává jako vstup samotnou mapu hran. Druhý autoenkodér pak dostává na vstupu výstup 1. sítě a také vstupní obraz. Jeho snahou je tedy rekonstruovat vstupní obraz i na základě domyšlených hran. Druhá podsíť má taktéž přiřazený diskriminátor, avšak na vstupu dostává pouze rekonstruovaný obraz. Trénování obou podsítí probíhá odděleně, kde 1. podsíť bere v potaz pouze adversariální chybu a také *feature-matching* chybu [24], která je také získána na základě diskriminátoru.

Feature-matching chyba slouží ke stabilizaci trénování díky tomu, že nutí generátor vytvářet výstupy se stejnými vlastnostmi jako má očekávaný výstup generátoru. K určení této chyby je využito několika prostředních vrstev diskriminátoru, které se použijí jako výstupy. Získají se tedy výstupy jak pro očekávaný výstup generátoru tak i pro vytvořený výstup generátoru a následně se tyto výstupy porovnávají. Tento způsob připomíná použití předtrénované sítě **VGG**, ale podle článku se autorům nepodařilo s ní dosáhnout dostatečně dobrých výsledků, kde jako důvod uvedli, že nebyla natrénovaná na produkování hranových informací. Chybová funkce pro druhou podsíť zahrnuje L_1 chybu, adversariální chybu, *Perceptual* chybu a také *Style* chybu, kde pak dále taky váhují jednotlivé složky výsledné chybové funkce.

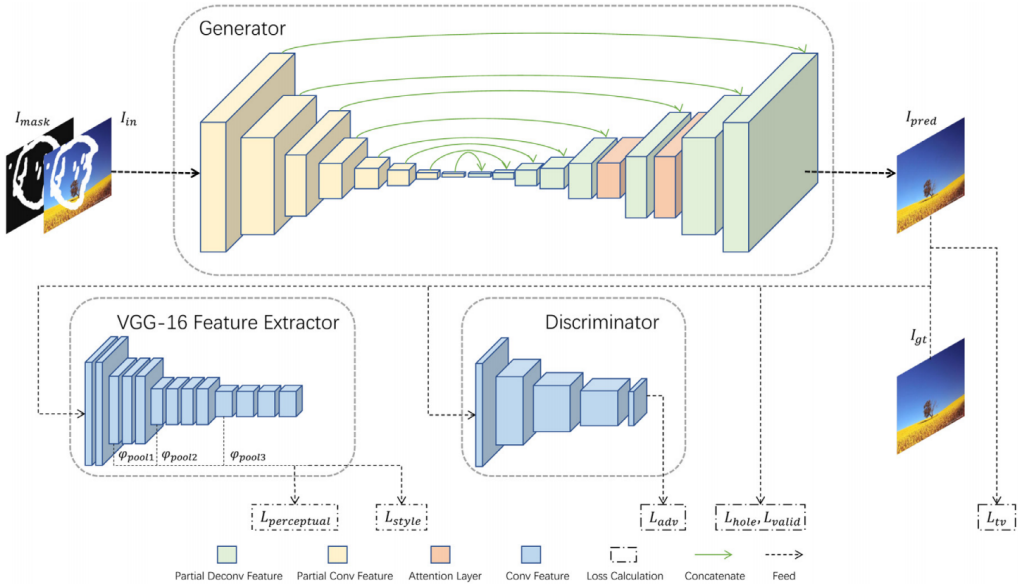
4.4.4 Multistage attention

V článku [18], který se zabývá rekonstrukcí obrazu s iregulární maskou, využívají také architekturu U-Net, kde však nahradili všechny konvoluční vrstvy *partial* konvolucemi. Jako součástí generátoru také využívají *contextual attention* po 2 konvolucích v části dekodéru, která by po použití na vícero různých úrovních měla zvýšit kvalitu příznakových map. Architekturu lze vidět na obrázku 4.6.

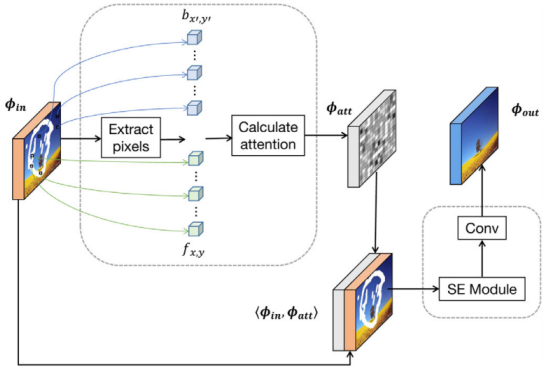
Jelikož by pro *contextual attention* bylo náročné určit výřezy, které mají sloužit jako konvoluční jádra z důvodu, že je maska iregulární se místo porovnání výřezů, porovnávají na úrovni pixelů. Po určení *attention* mapy ji poté konkatenují se vstupním tenzorem a tu pak využívají jako vstup SE modulu. Schéma tohoto principu lze vidět na obrázku 4.7. Funkce SE modulu lze rozdělit do tří kroků. V 1. kroku se provede globální *average pool* přes všechny kanály zvlášť. tím se získá *embedding* o rozměrech $1 \times 1 \times C$, kde C odpovídá počtu kanálů. V druhém kroku se nastavuje důležitost jednotlivých kanálů vyjádřena hodnotami z rozmezí $\langle 0, 1 \rangle$, které jsou získány pomocí dvouvrstvého perceptronu. Tento krok lze vyjádřit následovně:

$$w_c = \sigma(W_2 \psi(W_1, z)),$$

kde z značí vstup, W_1 a W_2 jsou trénovatelné váhy, ψ značí aktivační funkci **ReLU** a σ aktivační funkci **sigmoid**. Ve třetím kroku se vynásobí všechny hodnoty v každém kanálu vstupního tenzoru s hodnotou v odpovídajícím kanálu w_c. To lze vidět na obrázku 4.8.



Obrázek 4.6: Ukázka architektury Multistage Attention, převzato z článku [18].



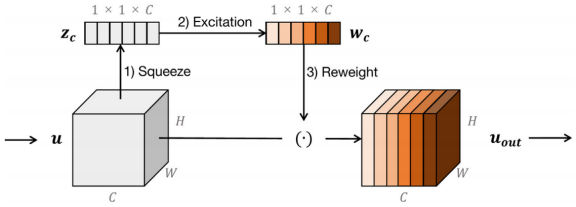
Obrázek 4.7: Ukázka principu vytvoření *attention* mapy pro pixelovou variantu *contextual attention*, převzato z článku [18].

Potom následuje úprava výstupního počtu kanálů na stejný počet jak před aplikováním *attention* vrsty pomocí konvoluční vrstvy s velikostí konvolučního jádra 1×1.

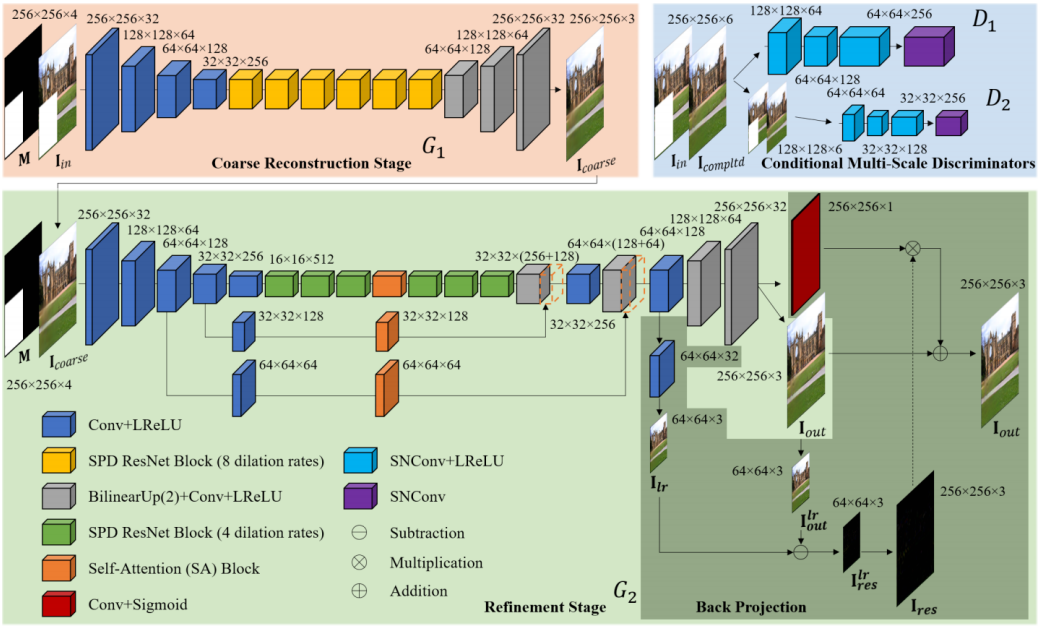
Diskriminátor je opět **PatchGAN** s jedním výstupním kanálem a se spektrální normalizací. Součástí chybové funkce pro generátor je L_1, která více penalizuje rekonstruované oblasti. Dále je součástí *Style* chyba, *Perceptual* chyba a také *Total variation* chyba, které jsou opět vynásobeny konstantami.

4.4.5 DeepGIN

V článku [13] zaměřeném na rekonstrukci obrázků, které obsahují jen nízké množství validních pixelů, využívají architekturu generátoru, která se skládá prvně s autoenkodéru a poté z architektury U-Net, kde se v 1. části vytváří hrubý obraz a ve 2. už ostrý obraz. V 2. části využívají další variantu *attention* zvanou *self attention* [28, 27] po konvolucích v nejhlubší části sítě a také v propojích *skip-connection*. Architekturu lze vidět na obrázku 4.9.



Obrázek 4.8: Ukázka principu SE modulu použitého pro pixelovou variantu *contextual attention*, převzato z článku [18].

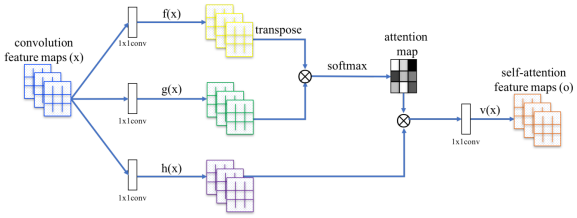


Obrázek 4.9: Ukázka architektury DeepGIN, převzato z článku [12].

Varianta *attention* zvaná *self attention* nepracuje nad výřezy, ale jen nad pixely a dále už také nerozlišuje, jestli leží pixel v oblasti s určenými nebo neurčenými hodnotami a porovnává tedy všechny pixely se všemi ostatními. Vstupní tenzor je zpracován 2 konvolucemi s velikostí jádra 1×1 a tím se získají 2 příznakové mapy, kde jedna z nich je transponována. Jejich pronásobením a aplikováním *softmax* funkce se získá *attention* mapa s rozměry $N \times N$, kde N odpovídá počtu hodnot vstupního tenzoru. Z *attention* mapy pak lze zjistit odkud pro danou pozici čerpat aktivace. Princip této vrstvy lze pak vidět na obrázku 4.10.

V architektuře využívají také 2 **PatchGAN** diskriminátory se spektrální normalizací natrénovány pomocí *hinge* chyby. Každý diskriminátor má vstup obsahující rekonstruovaný obraz s různou velikostí, druhý dostává o polovinu menší obraz. Zdůvodněním bylo, že chtěli zlepšit detaily a textury lokálních vzorů v různých měřítkách.

V článku také využívají principu *back projection* [7] (dále jen BP), která byla původně použita jako součást sítě pro zvětšování rozlišení obrázků. Tento princip by měl zajistit lepší zarovnání mezi vzory generovaného a referenčního obrazu. V článku na super rezoluci využívají bloků, které opakovaně zmenšují a zvětšují obraz za účelem určit nelineární vztah mezi zmenšeným a zvětšeným obrazem. Místo toho v tomto článku využívají vážený BP reziduál za účelem snížení počtu trénovatelných parametrů.



Obrázek 4.10: Ukázka *self attention* vrstvy, převzato z článku [27].

Jako další vrstvy pak využívají různé varianty **ResNet** bloků. Jedná se o vrstvu obsahující několik konvolučních vrstev, kde každá zpracovává určitou část vstupních dat. Ty jsou rozděleny rovnoměrně podle kanálů za účelem snížit počet trénovatelných parametrů. Každá daná konvoluce má jinou hodnotu *dilation* pro využití různě vzdálených příznaků při rekonstrukci.

Kapitola 5

Postup řešení

Trénování a testování sítí probíhalo na serveru *Metacentrum*[1] a všechny použitelné příkazy jsou dostupné na následující stránce[2]. Sítě, které zde budou popsány jsou implementovány v jazyce Python pomocí knihovny Keras.

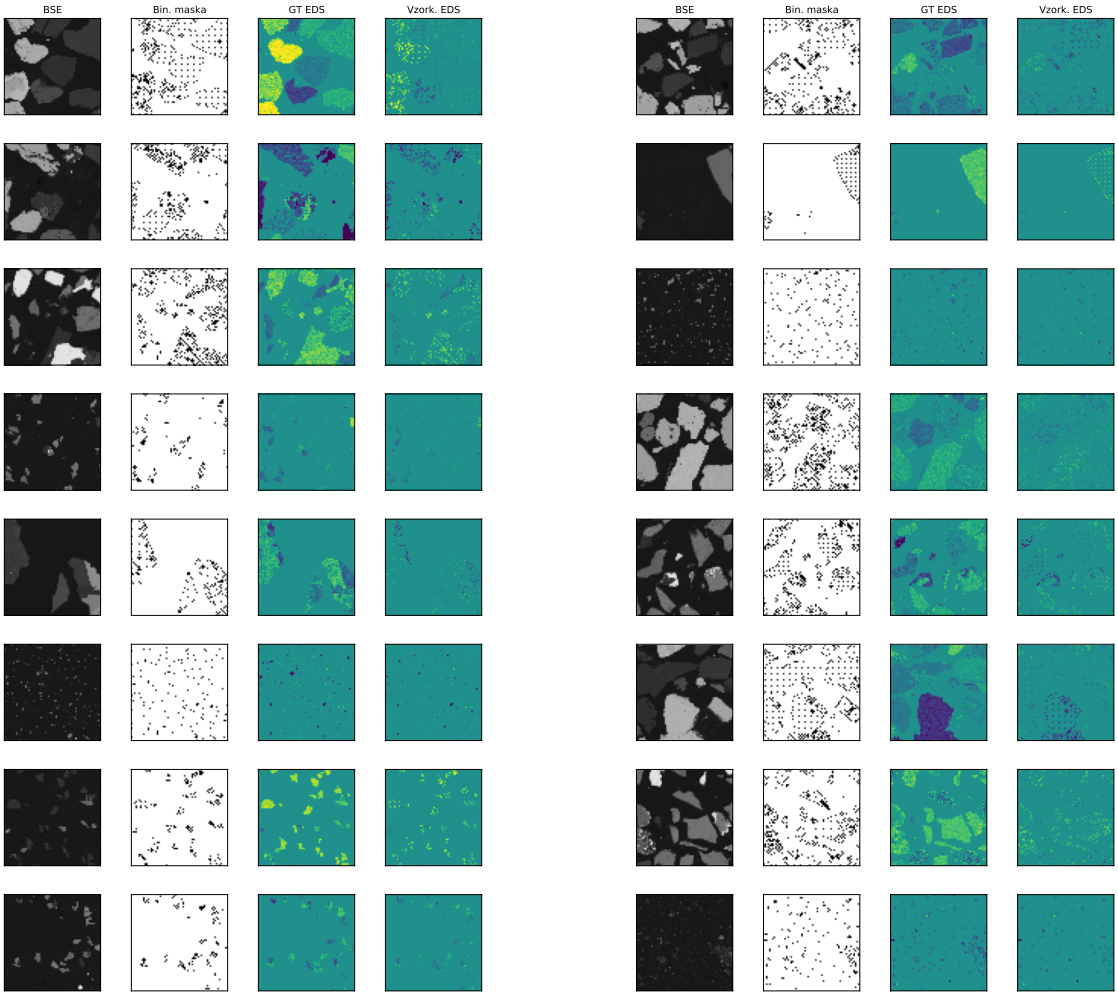
5.1 Vytvoření datových sad a trénování

Jelikož neexistuje datová sada obsahující vzorkované EDS a k nim příslušný očekávaný rekonstruovaný výstup, byla mi pro trénování neuronových sítí poskytnuta funkce pro vytváření binárních masek z BSE, která by se měla podobat skutečnému vzorkování dat. Na základě zadaných parametrů je schopna vytvářet binární masky zachovávající 3 až 15 procent pixelů v závislosti také na vstupním BSE. Z poskytnuté datové sady se spektry EDS určenými v každém bodě byl pro každý vzorek vytvořen *embedding* a ty společně s příslušným BSE byly uloženy do souborů v plném rozlišení rozděleny do trénovací, validační a do testovací datové sady v poměru 90 : 10 : 10. Jelikož jsou všechny implementované sítě plně konvoluční, jsou schopny pracovat s libovolnými rozměry vstupu. Jediné omezení je, že musí být rozměry vstupu dělitelné mocninou 2 v závislosti na počtu **pooling** vrstev v dané architektuře. Jako společná mocnina 2 byla zvolena pro všechny sítě hodnota 8. Způsob trénování funguje tak, že se načte ze souboru trénovací datová sada s daty různých velikostí a ty se během trénování postupně načítají. Velikost *batch* je nastavena na 1, za účelem umožnění co největších a různě velikých výřezů. Načítá se tedy náhodně zvolený výřez co s největší možnou velikostí z načítaných dat a na ten se během trénování aplikuje binární maska vygenerovaná funkcí s náhodnými parametry a jako GT pro rekonstrukci bude sloužit jejich podoba před aplikováním masky. Ukázky přiblížených výřezů lze vidět na obrázku 5.1.

5.2 Použité metody

V následující sekci budou popsány implementované metody, které byly použity pro experimentování a vyhodnocení. Všechny sítě byly natrénovány na 300 epochách.

[1] https://metavo.metacentrum.cz/
[2] https://wiki.metacentrum.cz/



Obrázek 5.1: Ukázky vytvořených výřezů, kde skupinky o 4 sloupcích jsou po řadě zleva BSE, vygenerovaná binární maska, kde černé pixely značí pozice, které mají být zachovány. Dále *embedding* EDS určený ve všech bodech a *embedding* EDS po aplikování masky.

5.2.1 Baseline - Metoda využívající 1 nejbližšího souseda

Jako Baseline metoda byla vytvořena výchozí metoda založená na principu popsané v sekci 4.1, která bude odpovídat způsobu rekonstrukce firmou TESCAN. Metoda vyplňuje chybějící pixely postupně pomocí zkopírování hodnoty sousedního pixelu, který už má určenou hodnotu. Implementace pak bude sloužit pro srovnání s ostatními metodami pomocí neuronových sítí a cílem bude dosáhnout lepších výsledků rekonstrukce obrazu oproti této metodě. Hlavním rozdílem bude, že v tomto případě bude rekonstruován *embedding* EDS.

5.2.2 U-Net

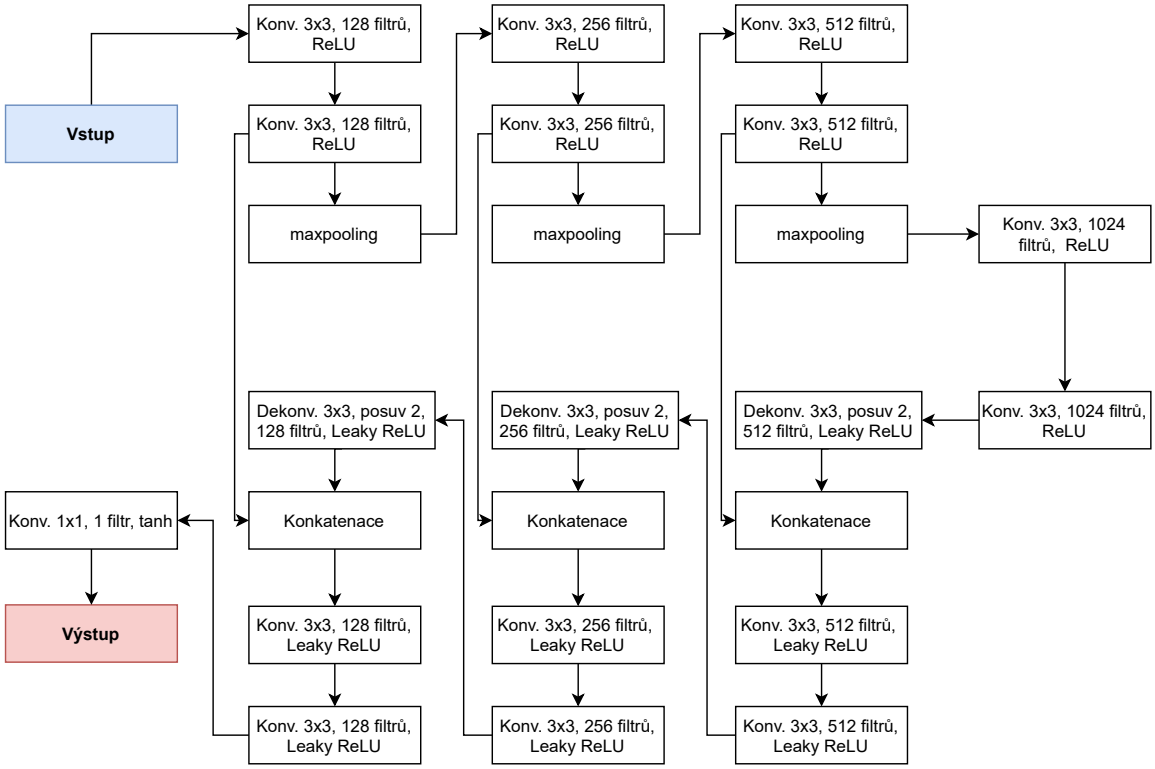
Jako zástupce pro tuto architekturu byla použita síť popsaná v článku [20], kde využívají architekturu U-Net pro segmentaci medicínských obrazů. Architekturu jsem zvolil z důvodu, že byla použita jako generátor pro GAN sítě zaměřené na rekonstrukci obrazu [18, 8], ale byla i přímo využita pro rekonstrukci obrazu na hyperspektrálních datech [21]. Zdrojový kód byl převzat a upraven z následujícího repozitáře[3]. Konvoluční vrstvy mají velikosti jader rovno 3. Byly dále odebrány vrstvy na nejvyšší úrovni v enkodéru a dekodéru za účelem snížení počtu parametrů a také z důvodu, že konvoluční vrstvy na dané úrovni obsahovaly méně filtrů než má vstup kanálů. Dále byly také odebrány **Dropout** vrstvy. Implementace využívá **Maxpooling** vrstev pro zmenšení rozměrů v enkodéru a naopak v dekodéru pro zvětšení byly **UpSampling2D**[4] vrstvy nahrazeny za **Conv2DTranspose**[5] vrstvy, což jsou dekonvoluční vrstvy obsahující trénovatelné parametry. Konvoluční vrstvy v enkodéru využívají aktivační funkci **ReLU** a v dekodéru jsou pak **Leaky ReLU**, kde tato kombinace vychází z článku [18]. Součástí sítě jsou také propoje *skip-connection* propojující výstupy vrstev enkodéru s výstupy vrstev dekodéru, které jsou implementované pomocí vrstvy **Concatenate**. Vstupem je trojice, kterou tvoří binární maska, BSE a vzorkované EDS se všemi 64 kanály. Výsledný počet kanálů vstupu pak odpovídá počtu 66. Na výstupu sítě je konvoluční vrstva s velikostí jádra rovno 1 a výstupem o stejné šířce a výšce jako jsou vstupní data. Vrstva obsahuje 64 filtrů a použitou aktivační funkcí je pak **tanh**. Jako GT pro dané vstupy pak slouží EDS před aplikovaním vygenerované binární masky. Jako chybová funkce byla použita L_1, ta byla zvolena na základě toho, že v článku [8] se jim podařilo dosáhnout lepších výsledků než s L_2 chybou. a jako optimalizační algoritmus byl použit **Adam** [11] s *learning rate* rovno 10^{-4} a parametrem β_1 rovno 0.5. Architekturu lze vidět na obrázku 5.2.

Druhý experiment byl proveden se stejnou architekturou, chybovou funkcí a optimalizačním algoritmem. Využívá ale jako vstup už jen dvojici o celkovém počtu 65 kanálů obsahující binární masku a vzorkované EDS. Cílem tohoto experimentu je zjistit, jak moc a také jakým způsobem se BSE projevuje na rekonstruovaném obrazu.

5.2.3 Pix2Pix

Jako další architekturu, která byla reimplementovaná, vychází z GAN architektury popsané v článku [8]. Jako generátor využívá U-Net architekturu, která je implementačně stejná jako je v sekci 5.2.2. Za diskriminátor je pak použita architektura **PatchGAN**. Obsahuje konvoluční vrstvy s aktivačními funkcemi **Leaky ReLU** a pro zmenšení rozměrů je pak použit parametr posuvu o velikosti 2 v konvolučních vrstvách. Na výstupu sítě je poté

[3] https://github.com/zhixuhao/unet
[4] https://keras.io/api/layers/reshaping_layers/up_sampling2d/
[5] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose



Obrázek 5.2: Schéma U-Net architektury. Všechny konvoluční využívají výplň.

Tabulka 5.1: Architektura naimplementovaného diskriminátoru **PatchGAN** pro trénování s *binary cross-entropy loss*.

Architektura diskriminátoru					
Vrstva	Velikost Jádra	Počet filtrů	Stride	Padding	Aktivační funkce
Konv. 1	3	128	2	Same	Leaky ReLU
Konv. 2	3	256	2	Same	Leaky ReLU
Konv. 3	3	512	2	Same	Leaky ReLU
Konv. 4	3	1	2	Same	Tanh

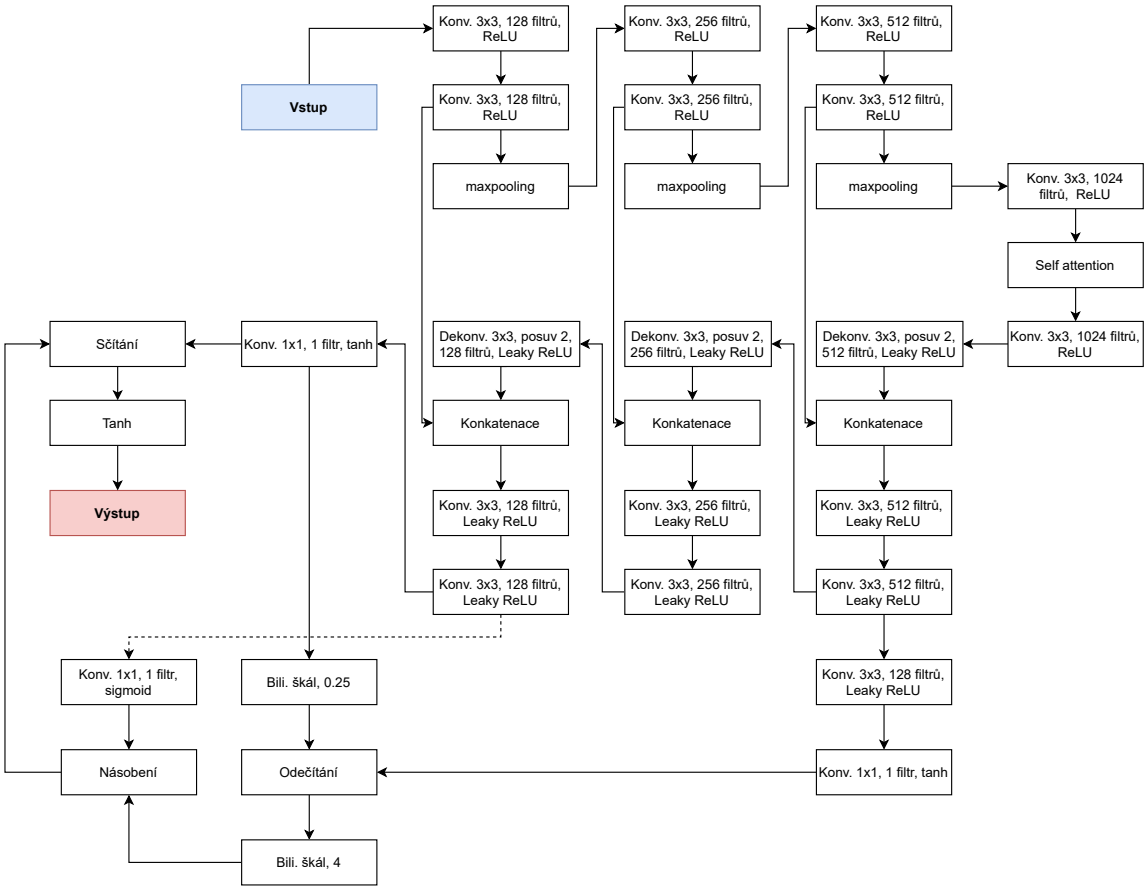
konvoluční vrstva s jedním filtrem a aktivační funkcí **sigmoid**. Vstupem pro generátor je opět trojice o celkovém počtu 66 kanálů a vstupem pro diskriminátor je dvojice o 65 kanálech obsahující rekonstruované EDS a BSE. GT pro diskriminátor nad vygenerovaným EDS je tenzor o stejných rozměrech jako je jeho výstup obsahující jen samé 0 a v případě, že na vstupu dostane původní EDS, použije se tenzor obsahující samé 1. Vrstvy diskriminátoru lze vidět v tabulce 5.1.

Součástí chybové funkce pro generátor je L_1 chyba, která je vynásobena 100 a *binary crossentropy* chyba diskriminátoru nad vygenerovanými daty. Chybová funkce diskriminátoru je pak *binary crossentropy* chyba nad vygenerovanými a skutečnými daty.

Optimalizační algoritmus pro generátor byl použit **Adam** s *learning rate* rovno 10^{-4} a parametrem β_1 rovno 0.5. Pro diskriminátor byl použit stejný optimalizační algoritmus, avšak nastaven s *learning rate* rovno $4 * 10^{-4}$

5.2.4 Pix2Pix + self att. + BP

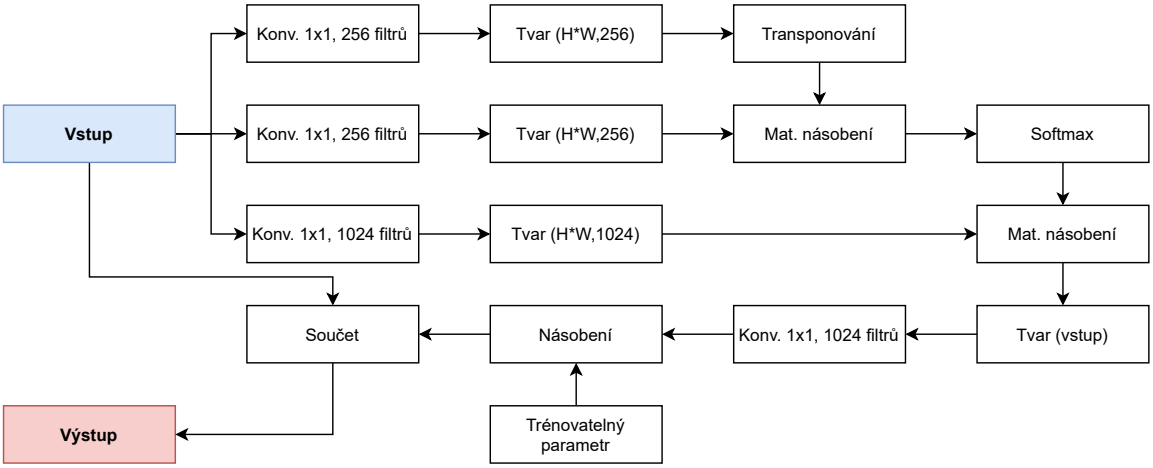
Jedná se o síť, která využívá U-Net jako v předchozích dvou architekturách. Je však doplněna o *self attention* vrstvu a také využívá princip *back projection*, který byl použit v článku popisující architekturu DeepGIN [12]. Jelikož je *self attention* vrstva pamětově náročná, je aplikována pouze v nejhlubší části generátoru na pomezí enkodéru a dekodéru. Schéma architektury lze vidět na obrázku 5.3.



Obrázek 5.3: Schéma architektury Pix2Pix, doplněná o *self attention* a BP.

Vrstva *self attention* byla reimplementována na základě článku [27] a lze ji vidět na obrázku 5.4. Obsahuje na začátku celkem 3 konvoluční vrstvy s jádrem o velikosti 1, kde 2 z nich obsahují čtvrtinu filtrů předcházející konvoluční vrstvy a budou tedy mít 256 filtrů. Třetí konvoluční vrstva zachovává počet kanálů a žádné z nich nepoužívají aktivační funkce. Výstupy vrstev se sníženým počtem kanálů se upraví na tvar $D \times 256$, kde D odpovídá počtu prvků původního vstupního tenzoru v jednom kanále. Dále je jeden z nich transponován a poté se maticově vynásobí mezi sebou. Tím se získá matice o velikosti $D \times D$. Po aplikování aktivační funkce *softmax* po řádcích se získá *attention* mapa. *Attention* mapa se poté maticově vynásobí s výstupem třetí konvoluční vrstvy upravené na tvar $D \times 1024$ a tím se získá tenzor o stejných rozměrech jako je upravený výstup třetí konvoluce. Tento výstup se upraví na tvar odpovídající původnímu tvaru a poté na něj znovu aplikuje konvoluce s jádrem o velikosti 1 a počtem filtrů rovno 1024 bez aktivační funkce. Výstup vrstvy se poté přičte k původnímu vstupu *self attention* vrstvy. Součástí vrstvy je ale ještě

trénovatelný parametr inicializovaný na 0, který určuje jak moc se má výstup projevit na daném původním vstupu. Tento součet je pak naimplementován pomocí *custom* vrstvy[6].



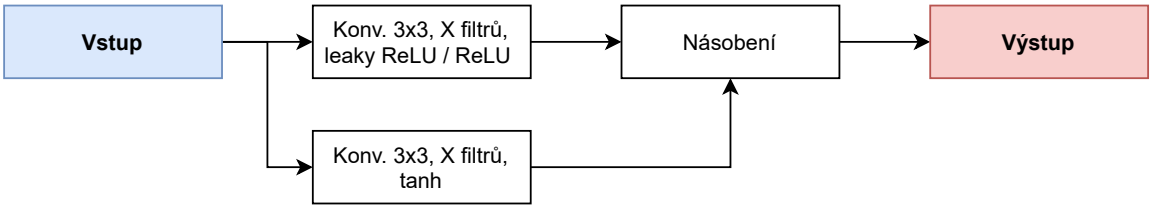
Obrázek 5.4: Schéma *self attention*.

V článku DeepGIN chyběly detaily ohledně implementace spojené se způsobem zmenšení obrázku a binárních masek a s nim spojený výpočet v chybové funkci, avšak z oficiálních zdrojových kódů[7] implementovaných v knihovně *PyTorch*, se mi podařilo zjistit, že pro implementaci BP používají bilineární úpravu velikosti jak pro zmenšení tak i zvětšení.

Na konvoluční vrstvu v dekodéru, který má rozměry 4-krát menší než původní vstup sítě, jsou napojeny sekvenčně 2 konvoluční vrstvy, kde poslední má aktivační funkci **tanh** a 64 filtrů. Tím se získá zmenšené rekonstruované EDS, které bude označeno jako I_{lr}. Z výstupu generátoru vracejícího 64 kanálů EDS v původním rozlišení je přidána *custom* vrstva umožňující bilineárně škálovat vstup s měřítkem rovno 0.25, což zmenší daný vstup na čtvrtinu původní velikosti označené jako I_{lr}^{out}. Provede se rozdíl mezi oběma zmenšenými obrazy a poté se na něj aplikuje bilineární škálování s měřítkem rovno 4. Výstup této operace bude označen jako I_{res}. Na předposlední vrstvu v dekodéru je dále kromě konvoluční vrstvy s výstupem EDS také napojena další konvoluční vrstva s 1 filtrem a s jádrem o velikosti 1, kde použitou aktivační funkcí je **sigmoid**. Tato vrstva je pak použita pro vynásobení s každým kanálem I_{res}. Tento součin se pak přičte k původnímu vstupu generátoru a na daný součet se použije opět aktivační funkce **tanh** a nový výstup bude označen jako I_{out}. Na diskriminátor byla také aplikována spektrální normalizace, která se často objevovala v implementacích diskriminátoru u mnoha článků [18, 26, 17]. Její implementaci jsem převzal z následujícího odkazu[8].

Součástí chybové funkce generátoru je opět L_1 a adversariální chyba, která je vynásobena konstantou rovno 10^{-3}. Výpočet L_1 chyby probíhá mezi I_{lr} a 4-krát zmenšeným EDS před aplikováním binární masky a dále mezi I_{out} a nezmenšeným EDS. V oblastech kde má probíhat rekonstrukce obrazu je vypočtená chyba vynásobena 5 a v případě zmenšeného EDS bude jeho odpovídající binární maska také bilineárně zmenšena. To však způsobí, že nebude už nabývat binárních hodnot, ale místo toho může nabývat hodnot z rozmezí $\langle 0, 1 \rangle$.

[6] https://www.tensorflow.org/guide/keras/custom_layers_and_models
[7] https://github.com/rlct1/gin
[8] https://github.com/thisisiron/spectral_normalization-tf2



Obrázek 5.5: Schéma *gated* konvoluce.

Tabulka 5.2: Architektura naimplementovaného diskriminátoru **PatchGAN** pro trénování s *hinge* chybou.

Architektura diskriminátoru					
Vrstva	Velikost jádra	Počet filtrů	Stride	Padding	Aktivační funkce
Konv. 1 + SN	3	128	2	Same	Leaky ReLU
Konv. 2 + SN	3	256	2	Same	Leaky ReLU
Konv. 3 + SN	3	512	2	Same	Leaky ReLU
Konv. 4 + SN	3	512	2	Same	-

V tomto případě se chyba na jedné pozici ve zmenšeném EDS bude počítat následovně:

$$L_{lr}^{pixel} = |(I_{lr}^{gt} \times M_{lr}) - (I_{lr}^{out} \times M_{lr})| + \lambda_{hole}|(I_{lr}^{gt} \times (1 - M_{lr})) - (I_{lr}^{out} \times (1 - M_{lr}))|,$$

kde $\lambda_{hole} = 5$ odpovídá zvětšení chyby nad rekonstruovanou oblastí, I_{lr}^{gt} odpovídá očekávané hodnotě, I_{lr}^{out} odpovídá výstupní hodnotě a M_{lr} odpovídá hodnotě masky. Diskriminátor je natrénován stejně jak u předchozího modelu.

5.2.5 Gated GAN

Jako poslední metoda byla opět použita GAN architektura, kde jako generátor byla použita U-Net architektura, kde jsou ale konvoluční vrstvy nahrazeny za *gated* konvoluce použité v článku [26]. Z důvodu že *gated* konvoluce obsahuje 2 konvoluční jádra, bylo nutné v nejhlubších vrstvách snížit počet filtrů na polovinu. Schéma implementace *gated* konvoluční vrstvy lze vidět na obrázku 5.5. Konvoluční vrstvy diskriminátoru také využívají spektrální normalizaci jak u předchozího modelu. Na výstupu diskriminátoru je konvoluční vrstva se spektrální normalizací a s 256 filtry, ale bez aktivační funkce. Vrstvy diskriminátoru lze vidět v tabulce 5.2. Diskriminátor je pak natrénován pomocí *hinge* chybové funkce a chybovou funkcí pro generátor je L_1 chyba, kde v oblastech které se mají rekonstruovat je vypočtená chyba vynásobena $\lambda_{hole} = 5$. Dále je její součástí adversariální chyba opět vypočtená nad rekonstruovanými obrazy, která je vynásobena $\lambda_{adv} = 10^{-3}$.

Kapitola 6

Vyhodnocení

Vyhodnocení je rozděleno do 3 částí. V 1. části bude provedeno vyhodnocení rekonstruovaných obrazů implementovaných metod nad daty EDS, ke kterým byly vygenerovány uměle masky. V 2. části budou ukázány rekonstruované EDS, ke kterým není GT. V poslední části pak bude vyhodnocení z pohledu segmentace na 3 různých vzorcích, ke kterým není dostupné GT pro EDS, ale je naopak dostupná pro segmentace. Zároveň je taky dostupná segmentační mapa z výstupu metody od firmy TESCAN a také ze segmentační metody popsané v článku [10].

6.1 Použité metriky

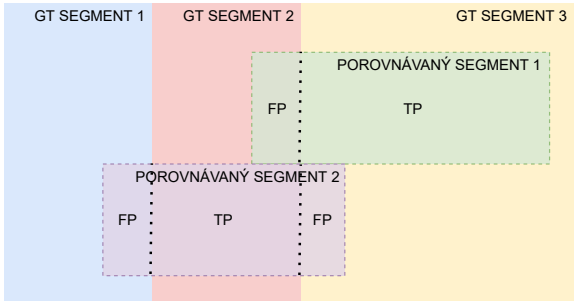
první dvě metriky pro vyhodnocení výsledků rekonstrukce různých metod, které budou použity pro 1. fázi vyhodnocení, vyžadují referenčního obrázku pro určení, jak moc se od něho liší. První z nich je špičkový poměr signálu k šumu (PSNR), který je spjat s MSE a v tomto případě, kde se maximální hodnota pixelu rovná 1, lze PSNR vyjádřit následujícím vzorcem:

$$PSNR = 10 \log_{10} \left(\frac{1}{MSE} \right).$$

Vyšší hodnota pak znamená větší podobnost ke GT.

Druhou metrikou je poté strukturální podobnost [25] (SSIM), která už bere více v potaz lidské vnímání a zohledňuje osvětlení, kontrast a strukturu. Může nabývat hodnot z rozmezí $\langle 0, 1 \rangle$, kde vyšší hodnota znamená větší podobnost ke GT.

Pro vyhodnocení segmentací je použito 5 hlavních metrik [10], kde prvními 3 jsou *true positive rate* (TPR), *false positive rate* (FPR) a nadsegmentační poměr. Způsob výpočtu TPR je takový, že se pro každou oblast v GT segmentaci najdou v porovnávané segmentaci (PS) oblasti, které ho ve většině překrývají. Pro výpočet TPR se pak určuje poměr mezi částmi segmentu v PS s názvem *true positive* (TP), které překrývají GT daný segment, s částmi stejného segmentu, které leží mimo GT zvané *false positive* (FP). FPR je pak opačná hodnota k TPR a platí rovnost $TPR + FPR = 1$, kde TPR a FPR mohou nabývat hodnot z intervalu $\langle 0, 1 \rangle$. Nadsegmentační poměr pak udává poměr mezi počtem segmentů v PS a počtem v GT segmentaci. Při srovnání GT sama se sebou by platilo $TPR = 1$, $FPR = 0$ a nad. poměr $= 1$. Ukázka kde se nachází TP a FP lze vidět na obrázku 6.1. Jelikož GT segmentace obsahuje už segmenty rozdělené podle materiálů, je očekávané, že výstup segmentačních metod bude obsahovat více segmentů než GT, ale ty lze pak zpětně spojit a je tedy spíše důležité, aby hodnota TPR dosahovala co nejvyšší hodnoty.



Obrázek 6.1: Na obrázku lze vidět, že větší část porovnávaného segmentu 1 leží v GT segmentu a bude tedy s ním porovnán. Na stejném příkladě pak tečkované černé čáry oddělují oblasti, které daný GT překrývají a budou tedy započítány do TPR od oblastí, které ho naopak nepřekrývají a budou místo toho započítány do FPR.

Další dvě jsou *precision* a *recall*, které pro jejich určení využívají pixelů tvořící hrany oddělující jednotlivé segmenty v PS nebo GT segmentaci od sebe. *Precision* udává poměr hran, které se v PS nachází a jsou také skutečně v GT segmentaci s celkovým počtem hran v PS. *Recall* pak udává poměr hran, které se v GT segmentaci nachází v PS s celkovým počtem hran v GT segmentaci. Označme $P(G, S)$ jako *precision* a $R(G, S)$ jako *recall*, kde G jsou hrany GT segmentace a S jsou hrany PS. Bude tedy platit:

$$R(G,S) = \frac{TP(G,S)}{TP(G,S) + FN(G,S)},$$

$$P(G,S) = \frac{TP(G,S)}{TP(G,S) + FP(G,S)},$$

kde $TP(G, S)$ jsou hrany, které se nachází v G a zároveň v S. $FN(G, S)$ odpovídá hranám, které se nachází v G ale ne v S a $FP(G, S)$ odpovídá hranám, které se nachází v S ale ne v G. Pro výpočet metrik pro segmentaci mi byl poskytnut zdrojový kód.

6.2 Výsledky

Naměřené metriky pro rekonstrukci obrazu lze vidět v tabulce 6.1 a ukázky přiblížených výřezů o velikosti 32 pixelů lze vidět na obrázku 6.2. Na obrázcích architektur U-Net lze vidět, že se BSE projevuje na rekonstrukci EDS tím, že napomáhá určit hranice mezi oddělenými oblastmi. Nejlepších výsledků rekonstrukce dosáhla právě architektura U-Net, kde však ostatní generativní sítě nebyly schopny zreplikovat šum uvnitř určitých oblastí, ale spíše je naopak více vyhladily. Hrany však zůstaly stále ostré.

Ukázky rekonstrukce EDS na datech, které nemají GT lze vidět na obrázku 6.3. Na ukázkách lze vidět zmenšené výřezy o velikostech 64 pixelů, kde na výstupech neuronových sítí se vytváří pravidelné artefakty. Ty však nejsou tak viditelné na výstupu **Gated GAN**.

Z důvodu že nebyla dostupná segmentační metoda, která je použitá firmou TESCAN, byla místo ní naimplementovaná jednoduchá segmentační metoda za pomocí metody **Watershed**. Pro vytvoření výškové mapy se na každý kanál EDS aplikoval Sobelův filtr a poté se přes všechny kanály vyhledala pro každou pozici maximální hodnota gradientu. Pro vytvoření značek se nejprve určily maximální hodnoty přes všechny kanály EDS a z nich se poté pomocí Cannyho hranového detektoru s hodnotou parametru $\sigma = 0$ vyhledaly hrany. Tím se získá binární mapa kde hrany nabývají hodnoty 1. Mapa se invertuje a poté se provede

Tabulka 6.1: Tabulka s naměřenými metrikami na testovací datové sadě vytvořené pomocí *embedding*u EDS definovaném ve všech pixelech a pomocí uměle vytvořených bin. masek.

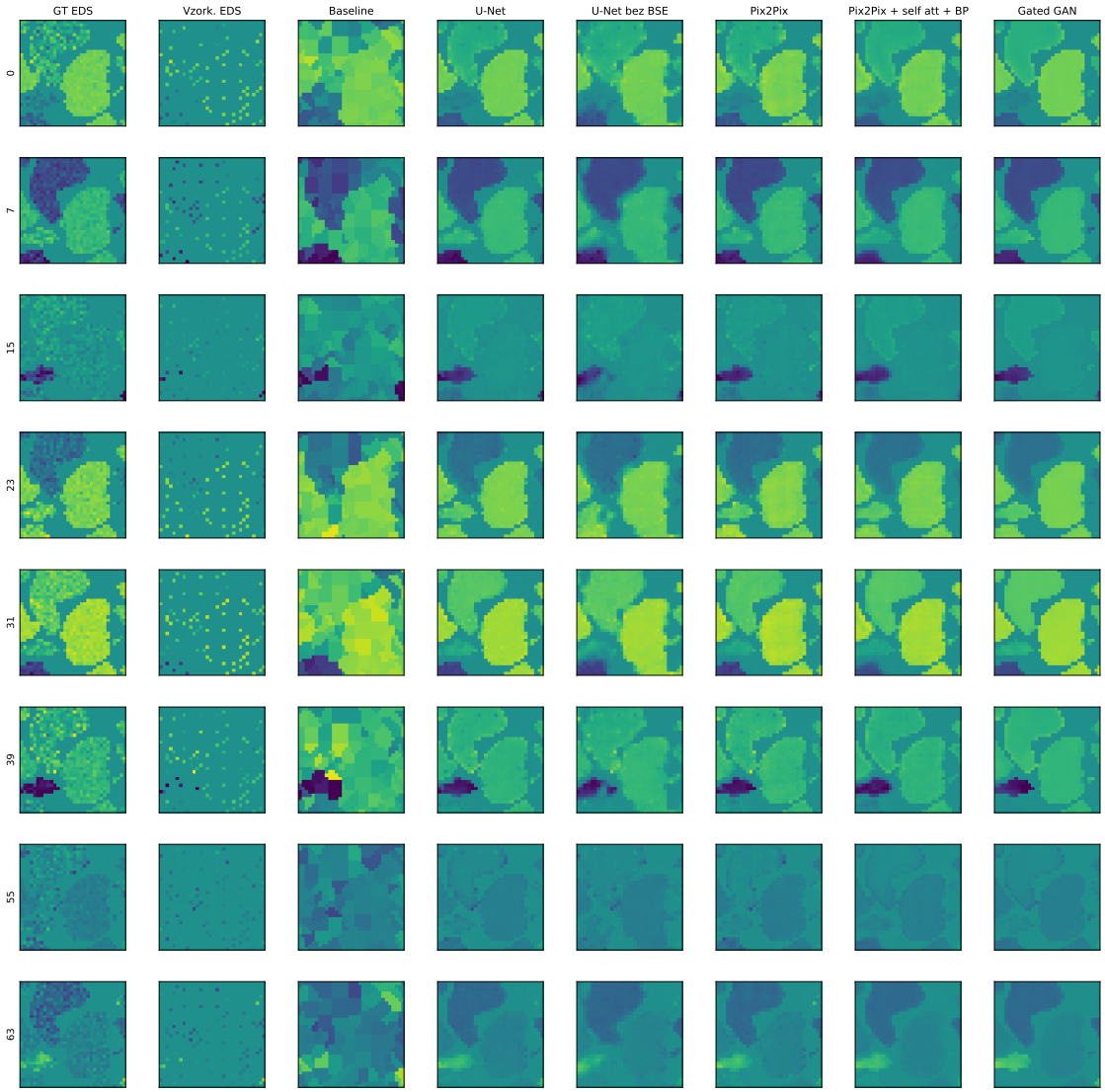
Metody	SSIM	PSNR
Baseline	0.268	25.2957
U-Net	0.9145	39.7161
U-Net bez BSE	0.8919	38.009
Pix2Pix	0.9113	39.4549
Pix2Pix + self att + BP	0.8952	38.5643
Gated GAN	0.9054	38.9344

Euklidovská vzdálenostní transformace a vyhledaná lokální maxima, pak budou odpovídat značkám pro segmentaci. Měnitelnými parametry pro tento způsob segmentace jsou pak nastavení prahů pro Cannyho hranový detektor.

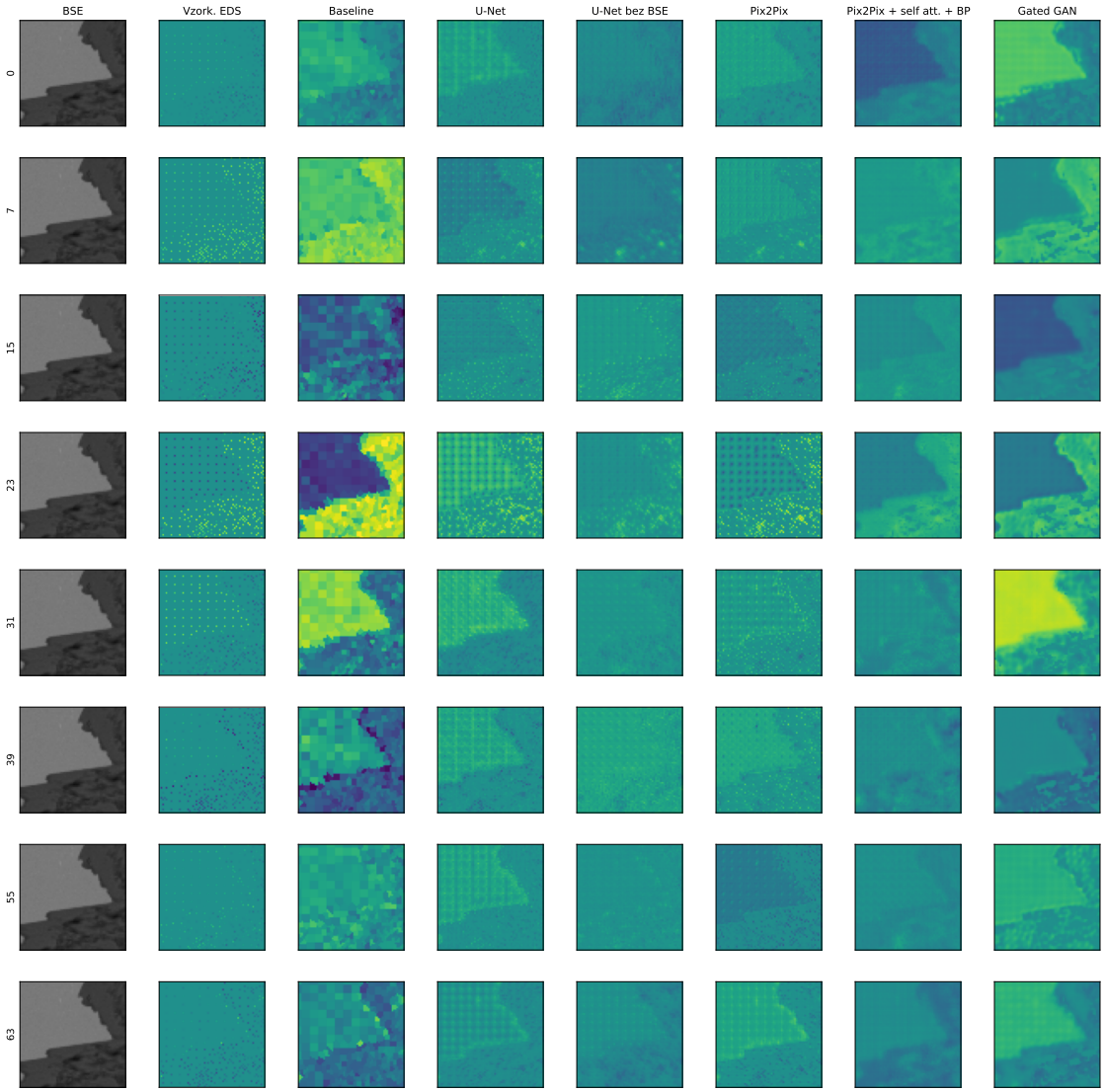
V grafech 6.4 lze v levém sloupci vidět naměřené TPR a k němu příslušný počet segmentů pro každý vzorek zvlášť. Tyto hodnoty byly naměřeny pomocí postupné úpravy prahů v segmentační metodě nad rekonstruovaným EDS. Měření bylo provedeno jen pro určité prahy a body v grafu odpovídající naměřeným metrikám byly spojeny do lomených čar, které lze vidět v grafech. Lepších výsledků o stejném počtu segmentů se podařilo dosáhnout se sítí **Gated GAN** na 1. vzorku. Avšak ve všech ostatních případech dosáhly implementované metody horších výsledků. Z porovnání výsledků mezi *baseline* a TESCAN lze vidět, že by šlo dosáhnout lepších výsledků pokud by byla naimplementována lepší segmentační metoda. Samozřejmě je třeba vzít i v potaz rozdíl typu dat nad kterými byla provedena segmentace, kde TESCAN segmentace vznikla z plného spekra EDS, zatímco baseline segmentace vznikla z *embedding*u EDS.

V druhém sloupci pak lze vidět naměřené metriky *recall* a *precision*, opět získané úpravou prahů u segmentační metody. Naměřené výsledky *precision* se stejnou hodnotou *recall*, jako má bod GMRF, dopadly hůře než daná metoda GMRF.

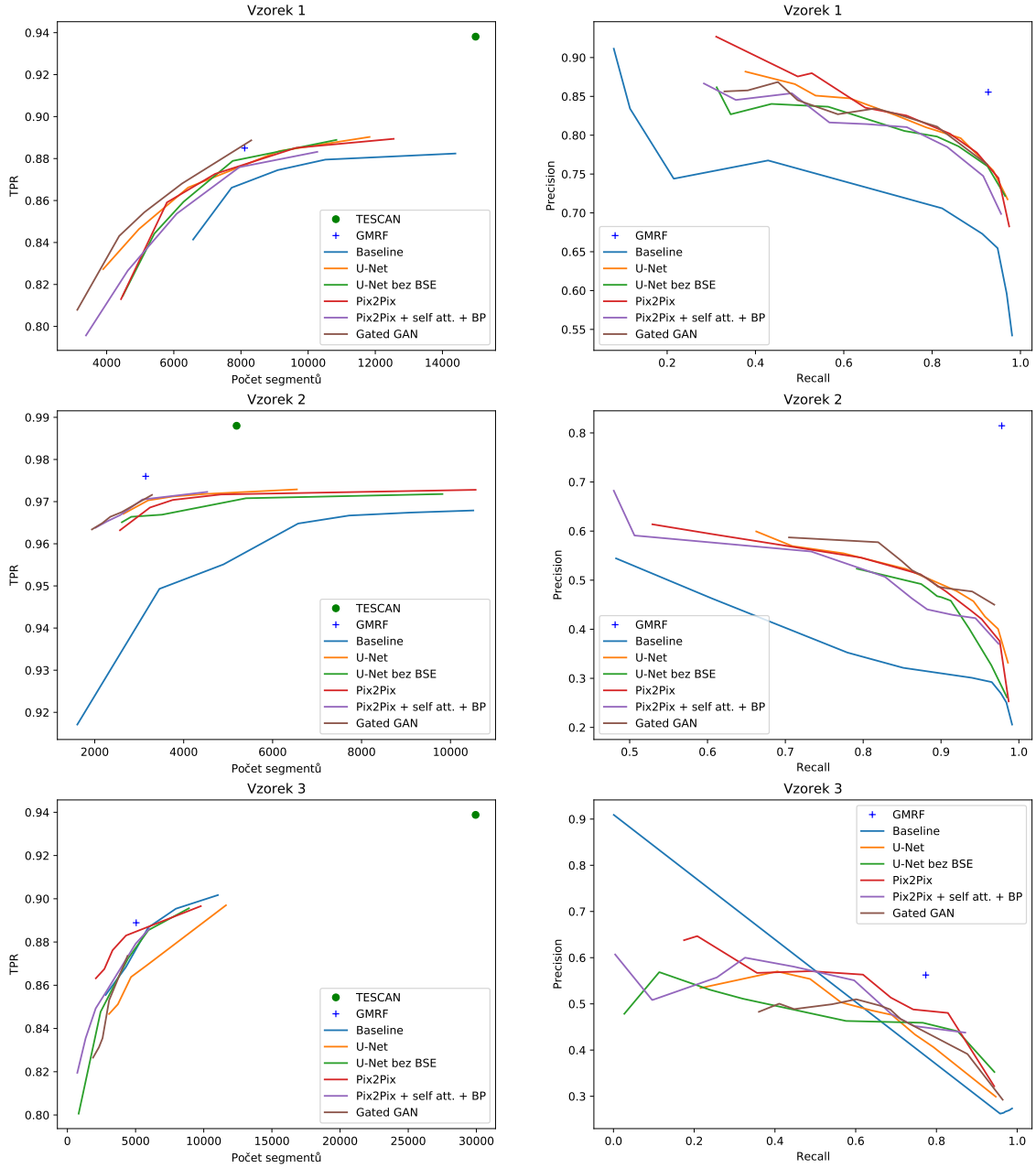
Tabulka 6.2 pak obsahuje hodnoty metrik segmentací TPR, FTP a nadsegmentační poměr (dále jen NP) s použitím stejného nastavení prahů v segmentační metodě nad rekonstruovanými daty. Obsahuje také metriky dvou referenčních segmentací vytvořených firmou TESCAN a metodou GMRF. Ukázky příslušných segmentací lze pak vidět na obrázcích 6.5 a 6.6.



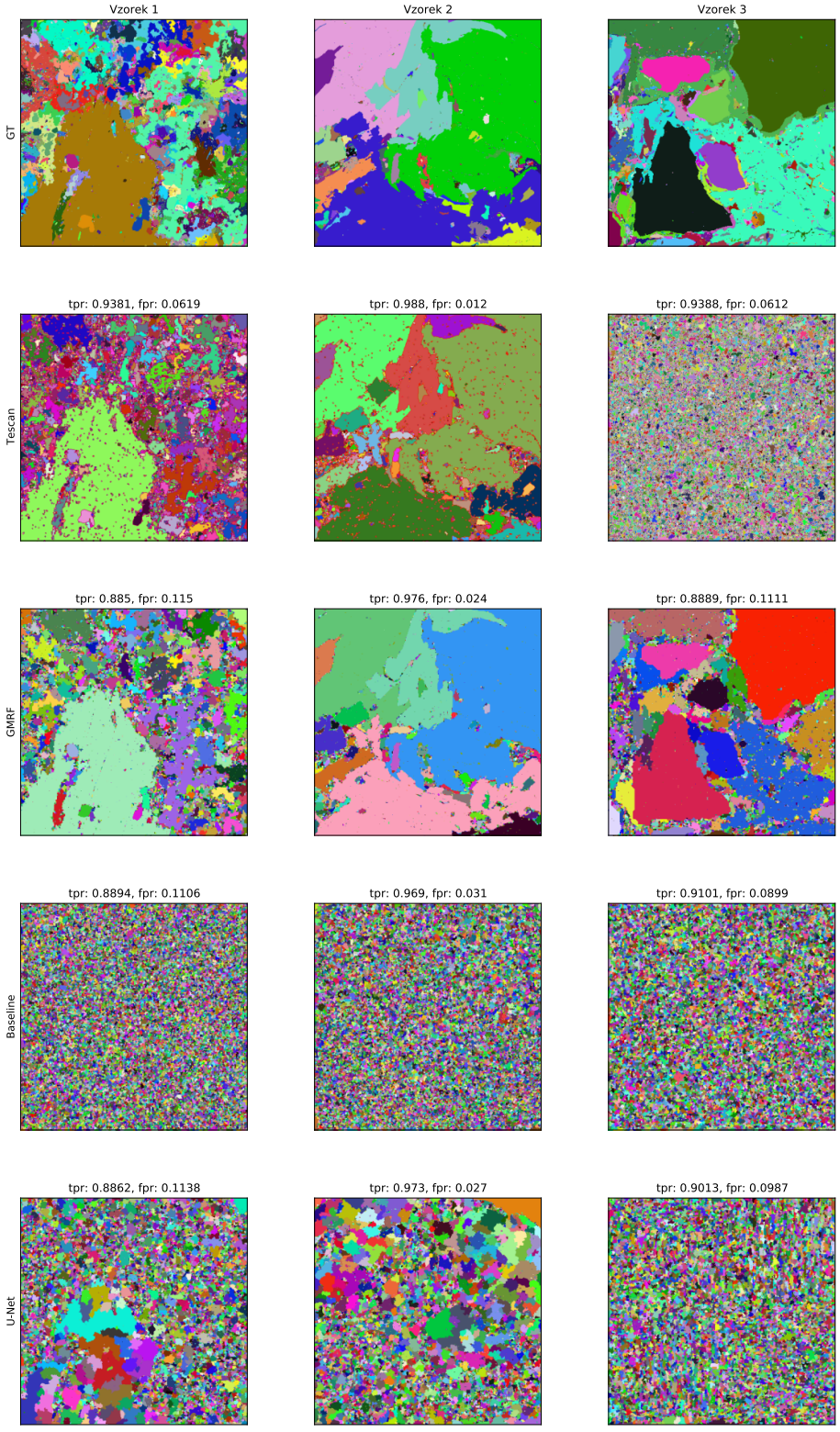
Obrázek 6.2: Ukázky GT EDS, vzorkovaného EDS a k nim příslušné rekonstruované EDS z různých implementovaných metod na vybraných kanálech. Velikosti přiblížených výřezů odpovídají 32 pixelům.



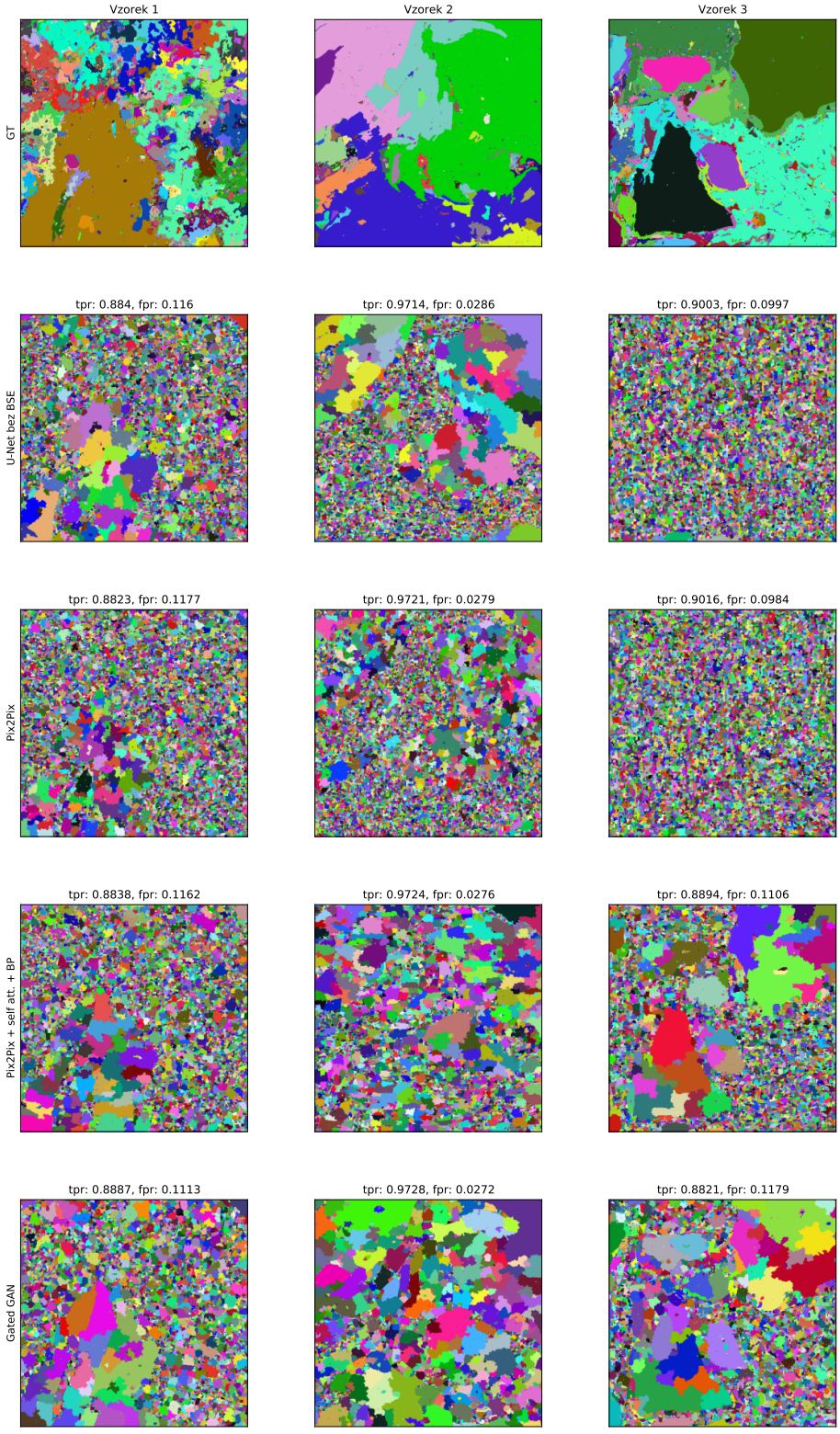
Obrázek 6.3: Ukázka přiblíženého výřezů o velikosti 64 pixelů na vzorkovaném EDS bez GT, kde v 1. sloupci je BSE.



Obrázek 6.4: Naměřené metriky pro segmentaci, kde na každém řádku je jiný vzorek. Lomené čáry uvnitř grafů odpovídají naměřeným metrikám získaných nastavením různých prahů v segmentační metodě. Body odpovídají metrikám poskytnutých referenčních segmentací.



Obrázek 6.5: Ukázky segmentačních map referenčních metod a map vytvořených z rekonstruovaných EDS získaných pomocí implementovaných metod.



Obrázek 6.6: Ukázky segmentačních map vytvořených z rekonstruovaných EDS pomocí implementovaných metod.

Tabulka 6.2: Tabulka obsahující metriky spojené se segmentací rekonstruovaných EDS porovnávané oproti referenční segmentaci. NP je zkratkou po nadsegmentační poměr.

	TPR	FPR	NP
Vzorek 1			
TESCAN	0.9381	0.0619	3.0041
GMRF [10]	0.885	0.115	1.6255
Baseline	0.8894	0.1106	5.6881
U-Net	0.8862	0.1138	2.6783
U-Net bez BSE	0.884	0.116	2.6013
Pix2Pix	0.8823	0.1177	2.9557
Pix2Pix + self att. + BP	0.8838	0.1162	2.3265
Gated GAN	0.8887	0.1113	1.6651
Vzorek 2			
TESCAN	0.988	0.012	6.3052
GMRF [10]	0.976	0.024	3.8222
Baseline	0.969	0.031	29.5577
U-Net	0.973	0.027	11.4435
U-Net bez BSE	0.9677	0.0323	7.5018
Pix2Pix	0.9714	0.0286	12.2151
Pix2Pix + self att. + BP	0.9724	0.0276	8.6209
Gated GAN	0.9728	0.0272	4.4848
Vzorek 3			
TESCAN	0.9388	0.0612	12.5177
GMRF [10]	0.8889	0.1111	2.1084
Baseline	0.9101	0.0899	7.0602
U-Net	0.9013	0.0987	5.2341
U-Net bez BSE	0.9003	0.0997	5.1187
Pix2Pix	0.9016	0.0984	5.0962
Pix2Pix + self att. + BP	0.8894	0.1106	2.9703
Gated GAN	0.8821	0.1179	2.273

Kapitola 7

Závěr

Podařilo se mi naimplementovat metody, které jsou schopny rekonstruovat *embedding* EDS dat, avšak neočekávaným výsledkem bylo, že naimplementované generativní sítě spíše zašumělé oblasti vyhladily, než že se je snažily napodobit. Nejlépe tedy dopadla architektura U-Net natrénovaná s L_1 chybou bez použití adversariální chyby s naměřenými metrikami SSIM=0.9145 a PSNR=39.7161 na vygenerovaných maskách. Porovnáním výsledků mezi architekturami U-Net s různými vstupy, kde jedna z nich obsahuje navíc BSE, lze vidět že BSE napomáhá ke kvalitnější rekonstrukci. Druhá síť bez BSE dosahovala hodnot SSIM=0.8919 a PSNR=38.009.

Z pohledu segmentace se síti **Gated GAN** podařilo na jednom ze vzorků, na kterém bylo prováděno vyhodnocení, dosáhnout vyšší hodnoty TPR při stejném počtu segmentů oproti metodě GMRF, avšak všechny ostatní výsledky naimplementovaných metod v porovnání s referenčními segmentacemi dopadly hůře. Výsledky segmentace by mohly být ale zlepšeny pomocí lepší implementace segmentační metody.

Výsledky rekonstrukce by mohly být zlepšeny hlubší sítí nebo použitím bloků obsahujících konvoluční vrstvy s různými hodnotami parametru *dilation*, ale omezujícím faktorem je velikost vstupních dat, které mohou nabývat délky a šířky až 1000 pixelů. Výsledky by mohly být také vylepšeny rozšířením datové sady například pomocí augmentace dat. Vyhodnocení by mohlo také obsahovat metriky pro vzorkovaná EDS s oficiální binární maskou, ke kterým bude dostupné EDS definované ve všech bodech.

Literatura

[1] *Obrázek - SEM*. https://www.technoorg.hu/news-and-events/articles/high-resolution-scanning-electron-microscopy-1/.

[2] *Obrázek - spektra EDS*. https://www.researchgate.net/figure/Energy-dispersive-X-ray-spectroscopy-EDX-results-of-a-bare-Co-Cr-and-b-CO-3-O-4-NWs_fig3_327726647.

[3] Abd Mutalib, M., Rahman, M., Othman, M., Ismail, A. a Jaafar, J. Chapter 9 - Scanning Electron Microscopy (SEM) and Energy-Dispersive X-Ray (EDX) Spectroscopy. In: Hilal, N., Ismail, A. F., Matsuura, T. a Oatley Radcliffe, D., ed. *Membrane Characterization*. Elsevier, 2017, s. 161 – 179. DOI: https://doi.org/10.1016/B978-0-444-63776-5.00009-7. ISBN 978-0-444-63776-5. Dostupné z: http://www.sciencedirect.com/science/article/pii/B9780444637765000097.

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

[5] Dai, Q., Chopp, H., Pouyet, E., Cossairt, O., Walton, M. et al. Adaptive Image Sampling Using Deep Learning and Its Application on X-Ray Fluorescence Image Reconstruction. *IEEE Transactions on Multimedia*. Institute of Electrical and Electronics Engineers (IEEE). Oct 2020, sv. 22, č. 10, s. 2564–2578. DOI: 10.1109/tmm.2019.2958760. ISSN 1941-0077. Dostupné z: http://dx.doi.org/10.1109/TMM.2019.2958760.

[6] Gatys, L. A., Ecker, A. S. a Bethge, M. *A Neural Algorithm of Artistic Style*. 2015.

[7] Haris, M., Shakhnarovich, G. a Ukita, N. Deep Back-Projection Networks For Super-Resolution. *CoRR*. 2018, abs/1803.02735. Dostupné z: http://arxiv.org/abs/1803.02735.

[8] Isola, P., Zhu, J.-Y., Zhou, T. a Efros, A. A. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018.

[9] Johnson, J., Alahi, A. a Fei Fei, L. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. 2016.

[10] Juránek, R., Výravský, J., Kolář, M., Motl, D. a Zemčík, P. Graph-based Deep Learning Segmentation of EDS Spectral Images for Automated Mineralogy Analysis. *Computers and Geosciences (in review)*. 2020.

[11] Kingma, D. P. a Ba, J. *Adam: A Method for Stochastic Optimization*. 2017.

[12] Li, C.-T., Siu, W.-C., Liu, Z.-S., Wang, L.-W. a Lun, D. P.-K. *DeepGIN: Deep Generative Inpainting Network for Extreme Image Inpainting*. 2020.

[13] Li, C., Siu, W., Liu, Z., Wang, L. a Lun, D. P. DeepGIN: Deep Generative Inpainting Network for Extreme Image Inpainting. *CoRR*. 2020, abs/2008.07173. Dostupné z: https://arxiv.org/abs/2008.07173.

[14] Liu, G., Reda, F. A., Shih, K. J., Wang, T., Tao, A. et al. Image Inpainting for Irregular Holes Using Partial Convolutions. *CoRR*. 2018, abs/1804.07723. Dostupné z: http://arxiv.org/abs/1804.07723.

[15] Ma, B., Gao, M., Wang, Z., Ban, X., Huang, H.-Y. et al. Deep learning based automatic inpainting for material microscopic images. *Journal of microscopy*. Září 2020, sv. 281. DOI: 10.1111/jmi.12960.

[16] Miyato, T., Kataoka, T., Koyama, M. a Yoshida, Y. *Spectral Normalization for Generative Adversarial Networks*. 2018.

[17] Nazeri, K., Ng, E., Joseph, T., Qureshi, F. Z. a Ebrahimi, M. *EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning*. 2019.

[18] Ning, W., Ma, S., Li, J., Zhang, Y. a Zhang, L. Multistage Attention Network for Image Inpainting. *Pattern Recognition*. Květen 2020, sv. 106, s. 107448. DOI: 10.1016/j.patcog.2020.107448.

[19] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. a Efros, A. A. *Context Encoders: Feature Learning by Inpainting*. 2016.

[20] Ronneberger, O., Fischer, P. a Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015.

[21] Sidorov, O. a Hardeberg, J. *Deep Hyperspectral Prior: Denoising, Inpainting, Super-Resolution*. Únor 2019.

[22] Ulyanov, D., Vedaldi, A. a Lempitsky, V. Deep Image Prior. *International Journal of Computer Vision*. Springer Science and Business Media LLC. Mar 2020, sv. 128, č. 7, s. 1867–1888. DOI: 10.1007/s11263-020-01303-4. ISSN 1573-1405. Dostupné z: http://dx.doi.org/10.1007/s11263-020-01303-4.

[23] Wang, S. *Generative Adversarial Networks (GAN): A Gentle Introduction [UPDATED]*. Duben 2017.

[24] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J. et al. *High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs*. 2018.

[25] Wang, Z., Bovik, A., Sheikh, H. a Simoncelli, E. Image Quality Assessment: From Error Visibility to Structural Similarity. *Image Processing, IEEE Transactions on*. Květen 2004, sv. 13, s. 600 – 612. DOI: 10.1109/TIP.2003.819861.

[26] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. et al. Free-Form Image Inpainting with Gated Convolution. *CoRR*. 2018, abs/1806.03589. Dostupné z: http://arxiv.org/abs/1806.03589.

[27] Zhang, H., Goodfellow, I., Metaxas, D. a Odena, A. *Self-Attention Generative Adversarial Networks*. 2019.

[28] Zheng, C., Cham, T.-J. a Cai, J. Pluralistic Image Completion. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, s. 1438–1447. DOI: 10.1109/CVPR.2019.00153.

Příloha A

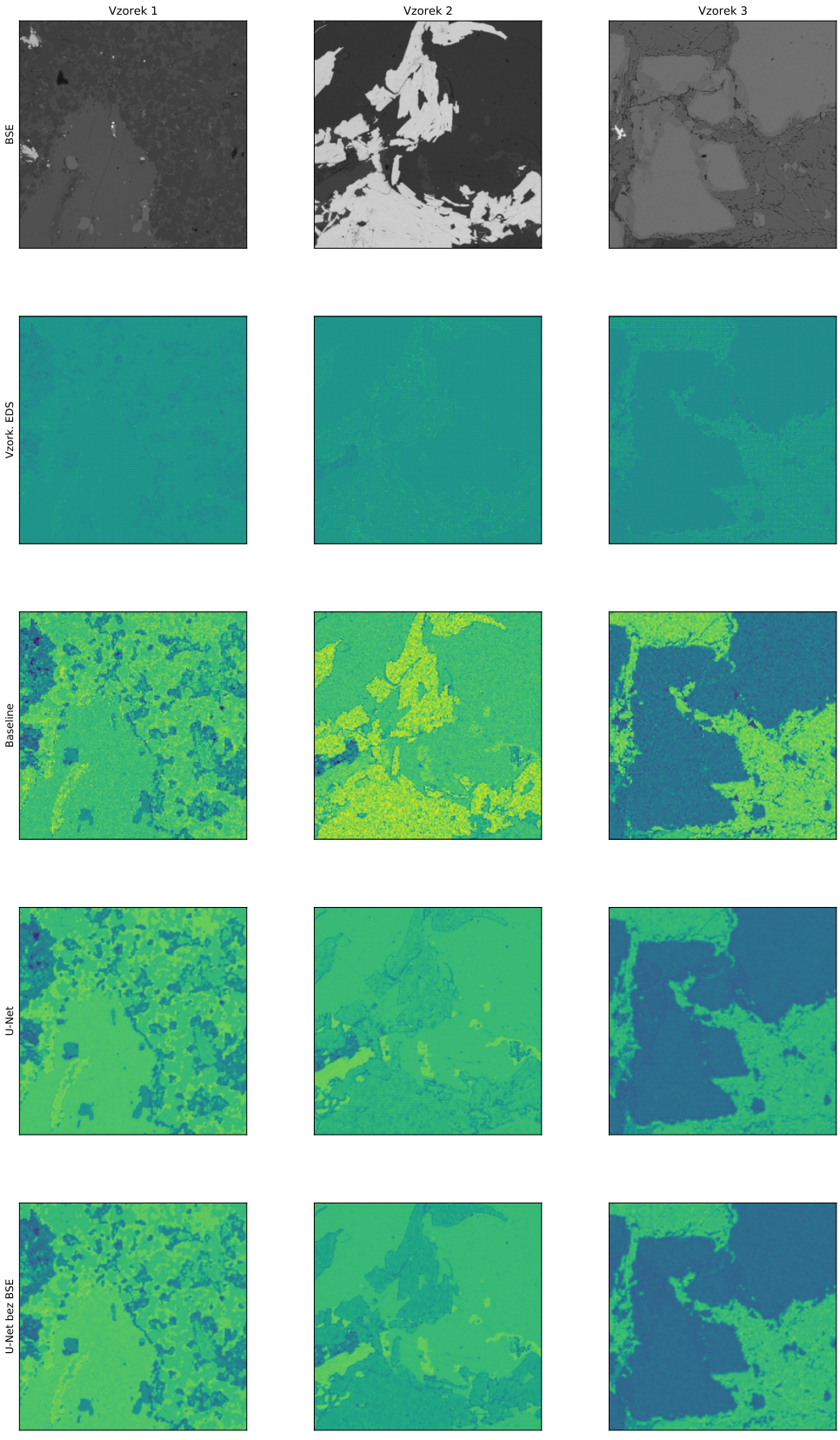
Obsah přiloženého paměťového média

Médium obsahuje složky:

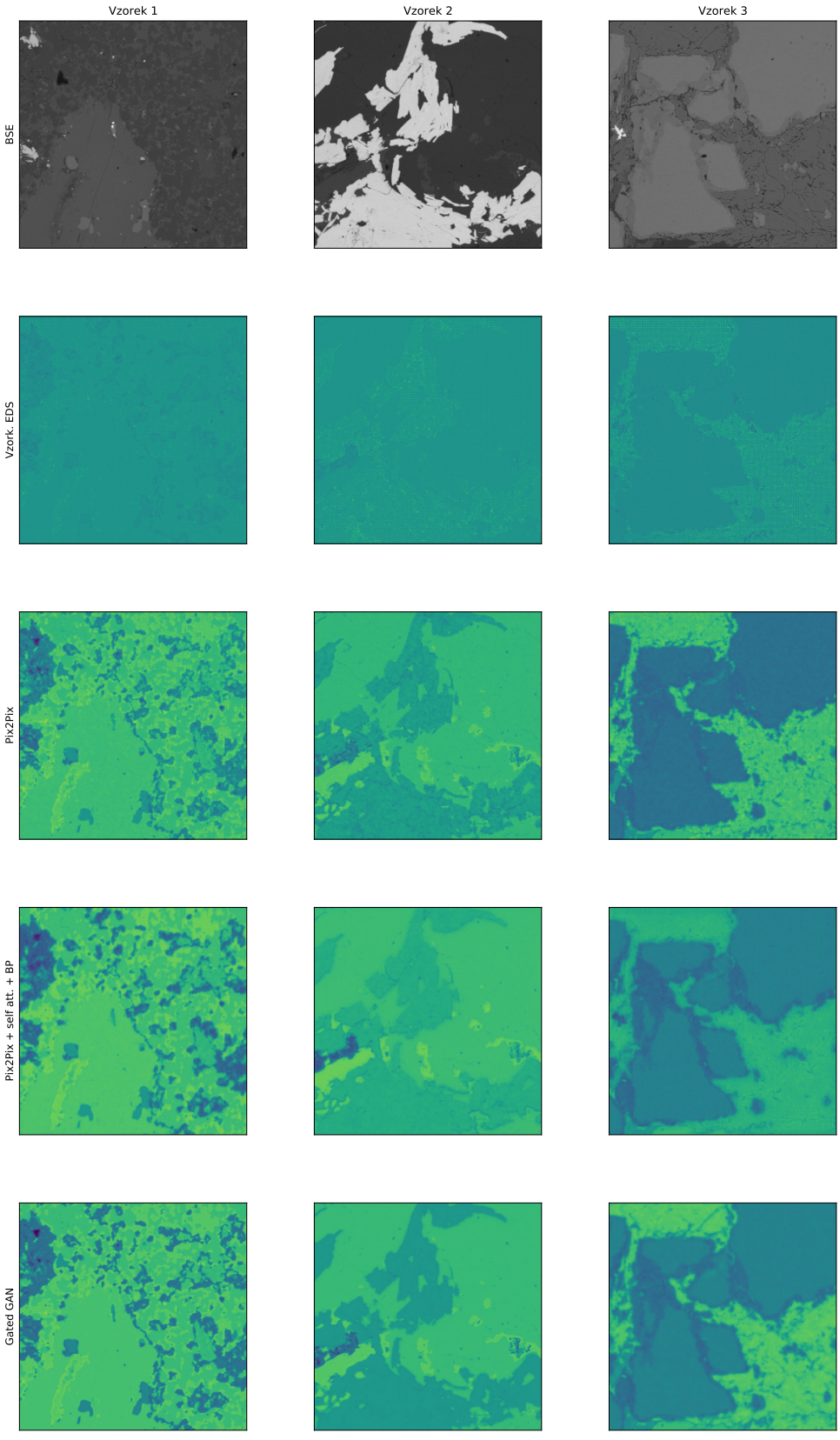
src Obsahuje zdrojové kódy.

add Obsahuje plakát.

doc Obsahuje PDF soubor práce.



Obrázek A.1: Rekonstrukce dat použité pro vyhodnocení segmentace.



Obrázek A.2: Rekonstrukce dat použité pro vyhodnocení segmentace.