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

KOMPRESE OBRAZU POMOCÍ NEURONOVÝCH SÍTÍ

IMAGE COMPRESSION WITH NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. LUKÁŠ TEUER**
AUTHOR

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

BRNO 2018

Abstrakt

Tento dokument se zabývá kompresí obrazu za pomoci různých druhů neuronových sítí. Jsou zde probrány vlastnosti použitých druhů neuronových sítí, jako jsou konvoluční a rekurentní neuronové sítě. V dokumentu jsou ukázány a podrobně popsány architektury neuronových sítí, které se dají použit ke kompresi obrazu, a vysvětluje, jakým způsobem pracují. Dále jsou zde provedeny experimenty nad různými strukturami a parametry neuronových sítí za cílem najít nejvhodnější vlastnosti sítě pro kompresi obrazu. Navrhují se zde nové koncepty pro kompresi obrazu pomocí neuronových sítí, které jsou hned otestovány. Na závěr je zde navržena síť skládající se z nejlepších konceptů a částí otestovaných během experimentování.

Abstract

This document describes image compression using different types of neural networks. Features of neural networks like convolutional and recurrent networks are also discussed here. The document contains detailed description of various neural network architectures and their inner workings. In addition, experiments are carried out on various neural network structures and parameters in order to find the most appropriate properties for image compression. Also, there are proposed new concepts for image compression using neural networks that are also immediately tested. Finally, a network of the best concepts and parts discovered during experimentation is designed.

Klíčová slova

Komprese obrazu, neuronové sítě, konvoluční sítě, rekurentní sítě

Keywords

Image compression, neural networks, convolutional networks, recurrent networks

Citace

TEUER, Lukáš. *Komprese obrazu pomocí neuronových sítí.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Komprese obrazu pomocí neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Teuer
22. května 2018

Poděkování

Děkuji mému vedoucímu Ing. Michalu Hradišovy, Ph.D. za odborné vedení, cenné rady a trpělivost, kterou mi v průběhu zpracování diplomové práce poskytl.

Obsah

Kapitola 1

Úvod

Komprese obrazu je téma, jež se těší stále velké pozornosti moderní vědecké komunity, která se pořád snaží hledat postupy a algoritmy, které zdokonalují současné výsledky, a je pravděpodobné, že ještě po nějakou dobu to bude pokračovat, než se najde optimální řešení pro všechny možné situace, pokud může také řešení vůbec existovat. Aktuálně se používá velká řada různých algoritmů a postupů pro kompresi obrazu, které dosahují rozdílných výsledků podle vlastností samotného obrazu a vlastností algoritmu.

U neuronových sítí se už delší dobu předpokládalo, že budou v oblasti komprese obrazu přinejmenším stejně úspěšné, jako jsou dosavadní používané postupy, což bylo v minulých letech potvrzeno.

Tato práce se tedy zabývá problematikou neuronových sítí a jejich použití pro kompresi obrazu. Obsahuje jednoduché uvedení do tématu konvolučních a rekurentních neuronových sítí, vysvětluje použité metody pro porovnání komprimovaného obrazu, neboť standardní metody pro porovnávání jsou uzpůsobené dosavadním komprimačním algoritmům, a představuje modely neuronových sítí aktuálně využitelných pro komprimaci obrazu.

Následující kapitola 2 obsahuje lehký úvod do problematiky a historie neuronových sítí a vysvětlení jejich základních principů. Dále jsou zde podrobněji popsány konvoluční neuronové sítě, které se hojně využívají v oblasti zpracování obrazu a budou se dále používat i v této práci. Je zde popsána architektura konvolučních a rekurentních sítí a jak se odlišují od obyčejných neuronových sítí. Popsány jsou i typické prvky, které se v těchto sítích používají, a to včetně dekonvoluční vrstvy nebo LSTM jednotek.

V další kapitole 3 je rozbor dřívější práce zabývající se kompresí obrazu za pomocí neuronových sítí. Budou zde hlavně popsány modely sítí, se kterými bylo experimentováno a celkově teorie a poznatky se kterýma jejich autoři přišly. Podrobně budou rozebrány jednotlivé modely sítí a budou popsány i společné principy, které využívají.

Následující kapitola 4 ukazuje můj návrh architektur a parametrů, které budu testovat a nad kterými budu experimentovat. Kapitola taky popisuje vhodné metody pro zhodnocení sítě, tak aby se hodnotící kritéria korespondovala s tím, jakým způsobem vidí člověk, použité nástroj pro tvorbu sítě a použitý dataset. Popisuje také návrh otestování konceptů, které by mohly zlepšit kvalitu stávajících kompresních neuronových sítí.

Poslední kapitola 5 popisuje implementaci nejdůležitějších částí a jednotlivé experimenty, které byly navrhnuty v předchozí kapitole. Experimenty jsou doprovázeny grafickým znázorněním jejich výsledků v podobě grafů a obrázků, které představují příklad kvality dané sítě. Na konci kapitoly je stručné vyhodnocení výsledků a porovnání s předchozími pracemi.

Kapitola 2

Neuronové sítě

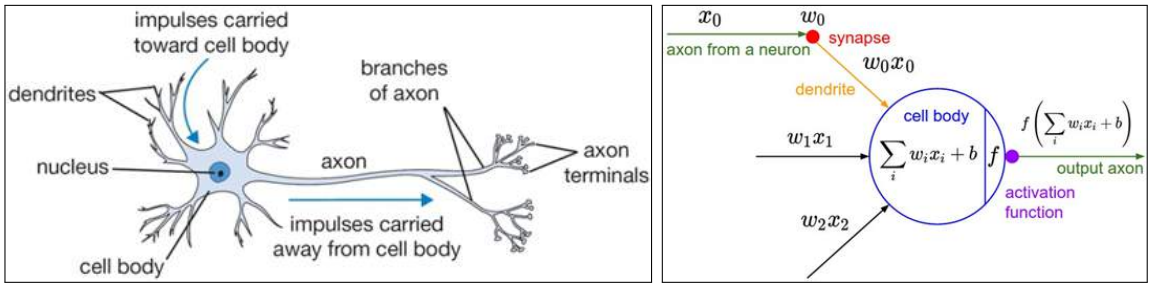
Neuronové sítě [10] jsou výpočetní nástroj, který je modelován podle struktury neuronů v mozkové kůře savců (s určitou mírou abstrakce), ale v podstatně menším měřítku. Zatímco velké neuronové sítě můžou obsahovat tisíce výpočetních jednotek, mozky savců mají miliardy neuronů s odpovídacím nárůstem ve schopnosti vzájemně spolupracovat a projevovat znaky chování. Tyto neurony bývají v sítích vysoce propojené a spolupracují spolu na řešení specifických problémů. Tyto sítě se, podobně jako lidé, učí pomocí příkladů. Stejně jako v biologických systémech, učení představuje drobné úpravy v synaptických propojeních, které existují mezi neurony.

Přestože se neuronové sítě dostávají do popředí teprve nedávno, toto odvětví bylo zavedeno ještě před vzestupem éry počítačů. Množství důležitých objevů a pokroků v tomto poli bylo však možné až s tímto rozvojem levné výpočetní síly, kterou počítače představovaly. Přesto v této době ještě nebyla výpočetní síla na takové úrovní, aby mohly neuronové sítě vzkvétat, a až na pár výjimek, většina vědců je nepovažovala jako budoucnost. Až s moderním pokrokem v technologiích (GPU a paralelizace), vynálezu backpropagation metody a dostupnosti trénovacích dat v podobě velkých datasetů, se neuronové sítě dostávají do pozornosti současných vědců, a to nejenom díky tomu, že dokázaly překonat mnohé dosavadní state-of-the-arts postupy a algoritmy v mnoha odvětvích.

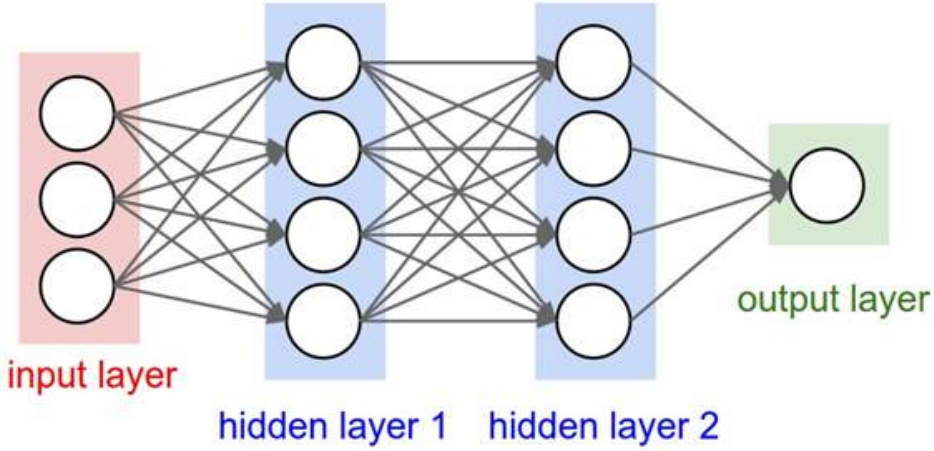
2.1 Struktura neuronových sítí

Tak jak je uvedeno v [1], odkud je v této části hlavně čerpáno, je základní jednotkou neuronové sítě neuron. Na obrázku 2.1 je představen biologický neuron a jeho zjednodušený matematický model. Tyto neurony jsou dále typicky organizovaný do více vrstev, které jsou určitým způsobem uspořádané za sebou v modelu, a každý neuron je typicky propojen s každým neuronem sousedící vrstvy. Každý jednotlivý neuron získává vstupní signály ze svých dendritů, což představuje výstupní hodnoty neuronů z předcházející vrstvy, a vysílá signál skrze svůj axon, kterým posílá hodnotu signálu do dendritů neuronů následující vrstvy.

Ve výpočetním modelu neuronu, signály, které putují skrze axony (např. x0), se násobí s hodnotou přidělenou napojenému dendritu (x0w0) ostatních neuronů, závislou na synaptické energii dané synapse (w0). Myšlenka je takové, že synaptická energie (představuje váhu w) se dá učit a následně může kontrolovat vliv vzájemného působení neuronů mezi sebou (jak pozitivního působení, tak i negativního). V základním modelu, dendrity přenášejí



Obrázek 2.1: **Vlevo:** Biologické zobrazení neuronu lidského mozku. **Vpravo:** matematické zobrazení neuronu



Obrázek 2.2: Topologie neuronové sítě s plně propojenými vrstvami

signál do těla neuronu, kde se všechny hodnoty signálu sečtou. Jestliže je výsledná suma nad určitou hranící (bias - b0), neuron může vyslat signál skrze svůj axon dále.

Ve výpočetním modelu se předpokládá, že přesné načasování vysílání signálů nehraje významnou roli a jedině frekvence zasílání signálů ovlivňuje informaci. V našem modelu je tato frekvence představena jako aktivační funkce. Historicky se běžně používala sigmoid funkce, kvůli svým reálným vstupům, které změní do intervalu mezi 0 a 1.

Jak bylo dříve řečeno, neurony každé neuronové sítě se organizují do vrstev, což se dá také představit jako skupiny neuronů spojených v acyklickém grafu. Cykly nejsou v základních modelech povoleny, protože by představovaly nekonečnou smyčku při dopředném procházení neuronovou sítí. Obyčejné neuronové sítě nejčastěji využívají plně propojené vrstvy, ve kterých jsou neurony dvou sousedních vrstev propojeny plně každý s každým, ale žádný neuron není propojen s žádným jiným neuronem ve stejné vrstvě. Obrázek 2.2 ukazuje příklad topologie sítě, která používá plně propojené vrstvy. Vrstvy, které se nenacházejí na začátku nebo konci modelu jsou nazývané skryté vrstvy.

Poslední vrstva celé sítě se většinou nazývá výstupní vrstva a její úkol je zhodnotit její vstupní vektor a typicky podat jako svůj výstup hodnotu nebo vektor hodnot, které představují skóre sítě pro daný vstup. To, co skóre znamená, je závislé na specifické úloze neuronové sítě, a je tedy úlohu od úlohy jiné.

2.2 Konvoluční neuoronové sítě

Konvoluční neuronové sítě jsou velmi podobné obyčejným neuronovým sítím z předchozí části. Jsou tvořeny z neuronů, které mají naučitelné váhy (w0) a biasy (b0). Každý neuron přijímá vstupy, získá jejich sumu a typicky nad ní následně provede nelineární aktivační funkci. Výsledek celé sítě je stále vyjádřen výstupem z poslední vrstvy pomocí skórovací funkce.

Na rozdíl od obyčejných neuronových sítí, konvoluční sítě nabízejí nové možnosti pro práci s více-dimenzionálními daty (typicky obrázky). Konvoluční sítě mají namísto vzájemného propojení všech neuronů, propojeny při aktivaci neuronů pouze okolní neurony v rámci dimenzí daného vstupu. Toho je dosaženo pomocí lokálních propojení a provázání vah, následovaných formou poolingu, což se nakonec projeví ve vytvoření posunem nezávislých vlastností (features). Další výhodou konvolučních neuronových sítí je zjednodušení trénování sítí, které je způsobené menším množstvím parametrů, než je u plně propojených sítí se stejným množstvím vrstev.

2.2.1 Architektura konvolučních sítí

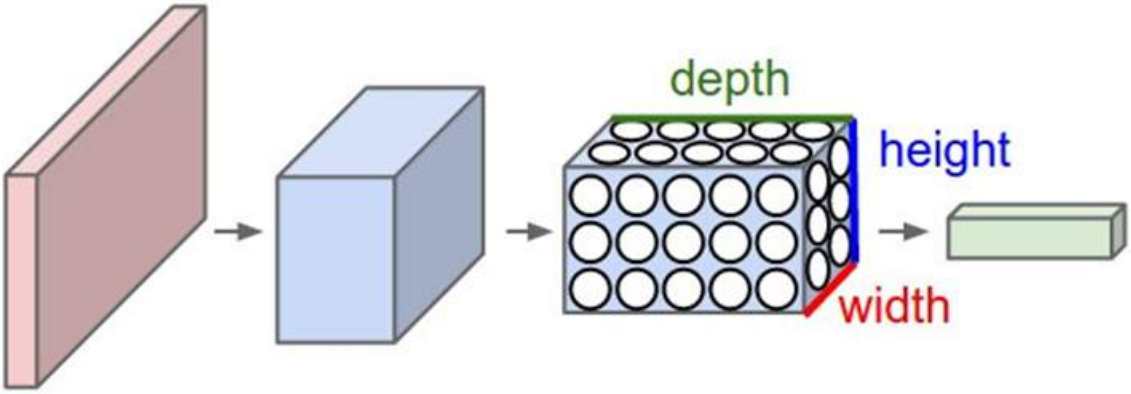
Tak jak je vysvětleno v [1], obyčejné neuronové sítě nejsou vhodné pro obrázky s proměnným měřítkem. Například pro obrázky o velikosti pouze 32x32x3 (32 pixelů šířka, 32 pixelů výška a 3 barevné kanály), jeden plně propojený neuron v první skryté vrstvě obyčejné neuronové sítě by měl 32x32x3 = 3072 vah. Toto množství je stálé zpracovatelné, ale je jasně zřejmé, že při použití většího obrázků (např. 200x200x3), každý neuron by měl mít 200x200x3 = 120 000 vah. Dále, typicky budeme potřebovat větší množství neuronů a v takovém případě množství parametrů velmi rychle narůstá. Je jasně vidět, že plné propojení všech neuronů způsobuje zbytečné plýtvaní zdrojů a takovéto velké množství parametrů by velmi rychle vedlo k přetrénování.

Konvoluční neuronové sítě [1] využívají skutečnosti, že vstup má vždy 2D strukturu dat a omezují svojí architekturu rozumnějším způsobem. Zejména, na rozdíl od obyčejných neuronových sítí, konvoluční sítě mají své neurony uspořádány do tří dimenzí: výška, šířka a hloubka (hloubka zde nepředstavuje velikost celé neuronové sítě ale pouze třetí dimenzi v ohledu prostoru jedné vrstvy). Neurony v jedné vrstvě jsou propojeny pouze s malou oblastí předchozí vrstvy, místo toho aby byly propojeny plně. Každá vrstva konvoluční neuronové sítě přemění vstupní 3D prostor na 3D výstupní prostor neuronových aktivací. Na obrázku 2.3 vidíte červenou vstupní vrstvu, která obsahuje vstupní obraz, takže její šířka a výška budou představovat rozměry obrázku a její hloubka bude představována množstvím barevných kanálu.

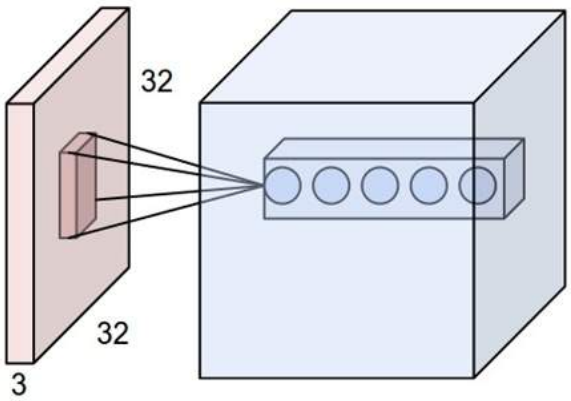
2.2.2 Konvoluční vrstva

Konvoluční vrstva je základním stavebním prvkem jakékoli konvoluční sítě a představuje nejvýznamnější prvek konvolučních sítí.

Tak jak jsou popsány v [1], parametry konvoluční vrstvy se skládají ze sady filtrů s naučitelnými parametry. Každý filtr je typicky malý v dimenzích šířky a výšky, ale sahá přes celou hloubku vstupního obrazu. Např. běžný filtr první vrstvy konvoluční sítě mívá velikost 5x5x3, kde 5x5 představuje výšku a šířku obrazu v pixelech a 3 představuje všechny barevné kanály, tedy hloubku). Při procházení sítí vezmeme postupně každý filtr a provedeme s ním konvoluci skrze celý vstupní obraz. Tímto se získá 2D aktivační mapa, kde každá bod představuje odpověď filtru na danou pozici vstupního obraz. Síť se naučí takové filtry, které



Obrázek 2.3: Grafické zobrazení architektury konvoluční sítě. Převzato z [1]



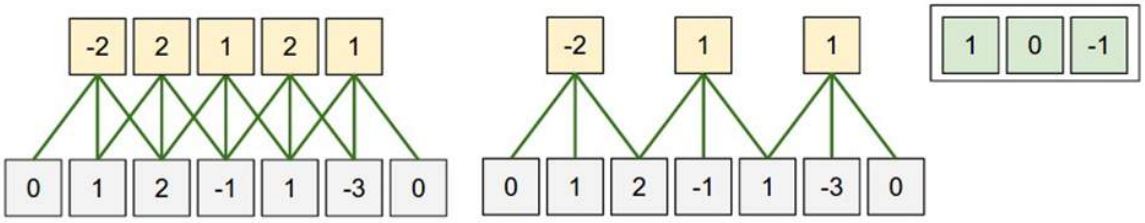
Obrázek 2.4: Lokální propojení konvoluční vrstvy. Převzato z [1]

se aktivují, pokud dostanou na vstup nějaký druh vizuálního rysu, jako je třeba hrana o určitém natočení nebo skupina bodů stejné barvy. Obecně platí, že pozdější konvoluční vrstvy v modelu se dají naučit reagovat na složitější vizuální prvky či celé objekty. Takto se zpracují všechny filtry v konvoluční vrstvě, kde každý vyprodukuje jednu 2D aktivační mapu. Tyhle mapy se poskládají na sebe (ve směru osy představující hloubku), čímž získáme výstup konvoluční vrstvy.

Obrázek 2.4 ukazuje propojení vrstev. Zatímco ve směru výšky a šířky jsou konvoluční vrstvy propojeny vždy lokálně (propojeny jen na omezeném prostoru), ve směru hloubky se vždy použijí všechny kanály (platí pro každou konvoluční vrstvu, nezávisle na vstupu). Každý filtr produkuje rozdílný výstup, který se poskládá za sebe (na obrázku 2.4 je tedy 5 filtrů).

Aby se dále zmenšil počet zpracovávaných parametrů, kterých by bylo stále více než je přijatelné i s použitím pouze lokálního propojování, konvoluční vrstvy podle využívají tzv. sdílení vah, kde každý filtr má pouze jednu sadu vah, která se nemění v závislosti na pozici vstupního obrazu. Vychází se z předpokladu, že jakýkoli prvek se může objevit na kterémkoli místě v obraze.

Velikost výstupu konvoluční vrstvy tedy závisí podle [1] na třech parametrech: hloubka, stride a padding. Jak už bylo výše řečeno, hloubka výstupu závisí na počtu filtrů, které



Obrázek 2.5: Ukázka parametru stride na 1D poli. Levá část má parametr stride jedna, pravá část má stride dva. Úplně vpravo je použitý konvoluční filtr. Převzato z [1]

konvoluční vrstva aplikuje na vstup. Druhý parametr je stride, který určuje mezery mezi nanášením konvolučních filtrů na vstupní obraz. V případě stride, který je 2 nebo více, se bude vždy postupovat o tolik pixelů při procházení obrazem. Stride způsobuje zmenšení výstupního obrazu v porovnání se vstupním obrazem, podle horizontálního a vertikálního směru.

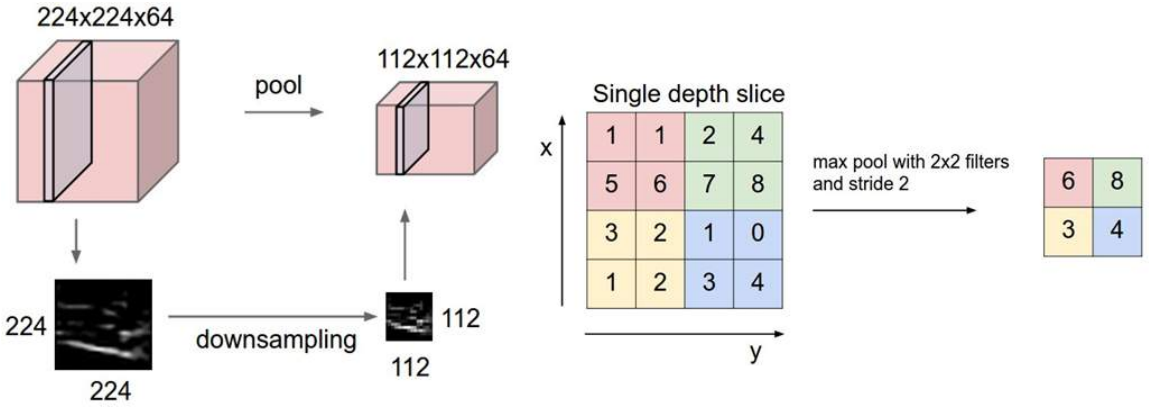
Posledním parametrem je padding, který určuje přidání nul na okrajích vstupního obrazu, čehož se používá pro vyrovnání ztráty velikosti způsobené velikostí filtru. Například použitím 5x5 filtru, zpracováváme první pixel až na třetím řádku a třetím sloupci, a ztratíme tedy 2 pixely z každé strany ve výstupu.

2.2.3 Pooling vrstva

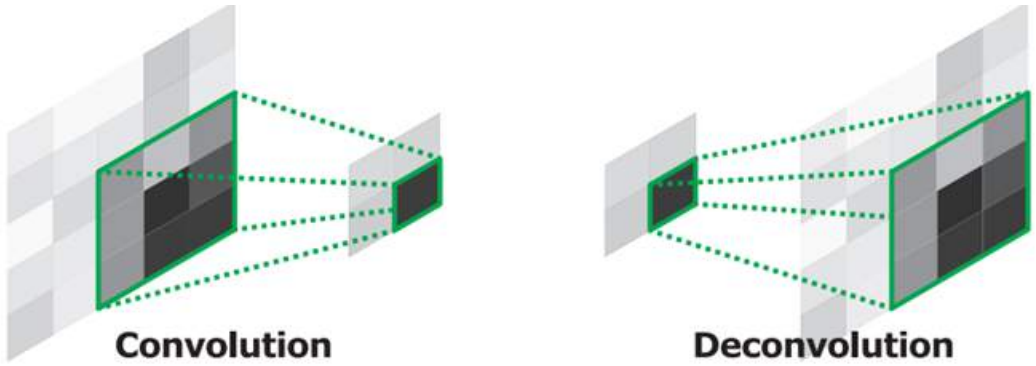
Jak je popsáno v [1], pooling vrstva slouží jako funkce pro progresivní zmenšení obrazu v horizontálním a vertikálním směru, čímž sníží množství parametrů a výpočtů v síti a tedy i umožňuje lépe se vyvarovat přetrénování. Je běžné vkládat pooling vrstvu mezi dvě následné konvoluční vrstvy v konvolučních sítích. Pooling vrstva pracuje nezávisle na hloubce vstupu a zmenšuje ho pouze ve zbylých dvou dimenzích, za použití MAX operace. Nejběžnější forma této vrstvy má filtry o velikosti 2x2 se stride 2, což způsobí zmenšení vstupu na polovinu v obou směrech, s výslednou redukcí 75 %. MAX operace v takovémto případě vezme největší hodnotu z daného 2x2 pole a zbylé zahodí. Hloubka zůstává vždy nezměněna. MAX funkce není povinná, může se používat i jiná operace jako Average nebo L2-norm, ale MAX bývá nejčastěji používanou.

2.2.4 Dekonvoluční vrstva

Tato vrstva není zcela běžná u běžných konvolučních sítí, je zde však zmíněna alespoň jedna verze, protože se v této práci bude dále často objevovat. Její název není úplně přesný, neboť tak jak je to vysvětleno v [8], dekonvoluce není operace, která v dekonvoluční vrstvě probíhá, ale používá se zde transponovaná konvoluce. I přes tuto nepřesnost se však začalo používat jméno dekonvoluční vrstva. Dekonvoluční vrstva je určitým způsobem míněna jako inverzní vrstva ke konvoluční vrstvě, tím způsobem, že zatímco konvolučení vrstva spojuje několik vstupních aktivací a vytváří jednu výslednou aktivaci, dekonvoluční vrstva přetváří jednu vstupní aktivaci na několik výstupních. Výstup dekonvoluční vrstvy je pak zvětšená aktivační mapa. Naučené filtry v těchto sítích se snaží rekonstruovat tvar vstupního objektu, a tedy podobně jako v konvolučních vrstvách, hierarchické struktury dekonvolučních vrstev se používají pro zachytávaní různých úrovní tvarových detailů. Filtry na nižších úrovních



Obrázek 2.6: Funkce pooling vrstvy. V levé části ukázka změny dimenzí po průchodu pooling vrstvou. V pravé části ukázka MAX operace s filtrem 2x2 a stride 2. Převzato z [1]



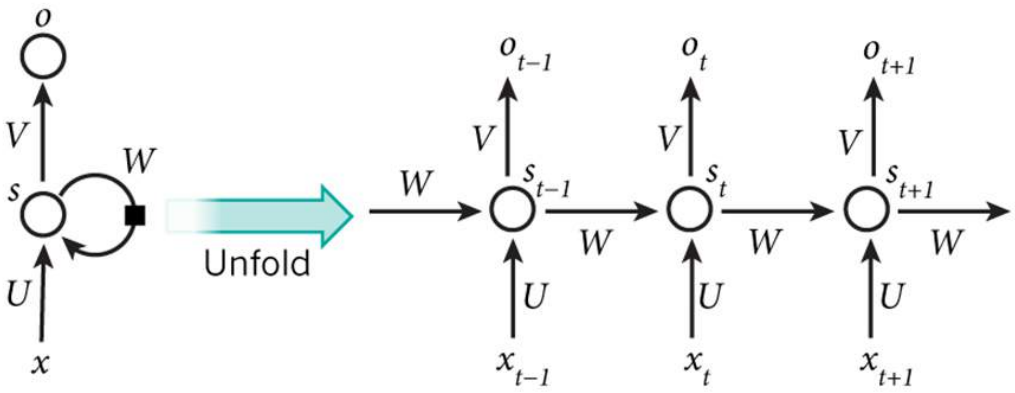
Obrázek 2.7: Znázornění rozdílu mezi konvoluční a dekonvoluční vrstvou. Převzato z [8]

se přiklání k zachytávání všeobecného tvaru objektu, zatímco přesnější a jemnější detaily jsou naučeny ve filtrech vyšších vrstev.

2.3 Rekurentní neuronové sítě

Rekurentní neuronové sítě [5] patří do rodiny neuronových sítí pro zpracování sekvenčních dat. Podobně jako konvoluční neuronové sítě jsou specializované sítě pro zpracování 2D dat jako je třeba obraz, rekurentní sítě jsou sítě specializované pro práci s posloupností hodnot. Stejně jako se konvoluční sítě mohou používat i na obrázky s velkou šířkou a výškou, a některé konvoluční sítě mohou zpracovávat i obrázky s proměnnou velikostí, rekurentní sítě se umí přizpůsobit mnohem větším sekvencím dat, než by bylo praktické pro obyčejné neuronové sítě, které k tomu nejsou sestrojeny. Většina rekurentních sítí je také schopná zpracovávat posloupnosti s proměnnou délkou.

Myšlenka za rekurentními neuronovými sítěmi, tak jak je uvedeno v [2], je mířena na využití informací, které objevují postupně. Pokud je potřeba předpovědět příští slovo ve větě, je vhodné vědět, jaké slova se objevily před ním. Rekurentní sítě se jmenují rekurentní, protože provádějí stejnou úlohu na každý prvek posloupnosti, a jejich výstup je závislý i



Obrázek 2.8: Rekurentní síť a ukázka jejího rozbalení v čase. Parametr t představuje jednotlivé kroky. Převzato z [4]

na předchozích průbězích. Jiná možnost jak si představit rekurentní sítě je taková, že mají formu "paměti", která zachycuje informace, které byly prozatím vypočteny. Teoreticky si mohou rekurentní sítě pamatovat neomezeně dlouhé posloupnosti informací, prakticky jsou však omezeny na zapamatování si pouze několika předcházejících kroků.

2.3.1 Architektura rekurentních sítí

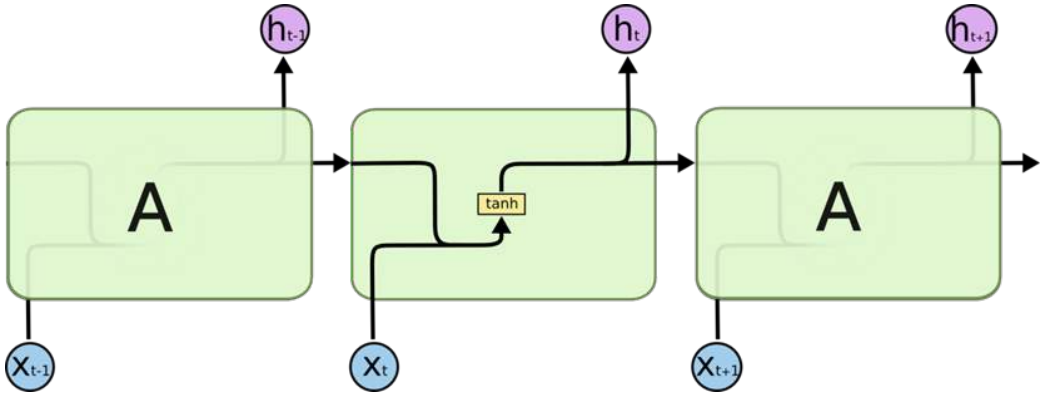
Rekurentní neuronové sítě jsou stavěny podobně jako ostatní dopředné neuronové sítě, liší se však svou architekturou a jak jsou její neurony vzájemně propojeny. Dopředné neuronové sítě jsou organizovány ve vrstvách a těmito vrstvami se postupuje pouze jedním směrem od vstupní vrstvy po výstupní. V propojení sítě se neobjevují a ani nesmí objevit žádné cykly. Přesto, že samotný mozek obsahuje ve svojí struktuře neuronů cykly, tyto cykly se z obyčejných neuronových sítí odstranily, kvůli zjednodušení trénovacího procesu na úkor výpočetní flexibility. Rekurentní neuronové sítě, tak jak jsou popsány v [4], ve svojí struktuře tyto cykly povolují, což umožňuje informaci přetrvávat použitou dobu v síti a sítím tedy umožňuje vynikat v úlohách, kde ostatní typy sítí nejsou vhodné. Na obrázku 2.8 je ukázáno typické představení rekurentní neuronové sítě a její rozbalení.

Diagram 2.8, jak je popsaný v [4], ukazuje rozbalení rekurentní sítě do posloupnosti vrstev. Rozbalení sítě znamená, že se síť přepíše pro celou aktuální délku posloupnosti do stejných propojených modulů. Pokud se například v sítích zajímáme o posloupnosti pěti slov, výsledná rozbalená síť by obsahovala pět modulů (vrstev), každá modul určený pro jedno slovo.

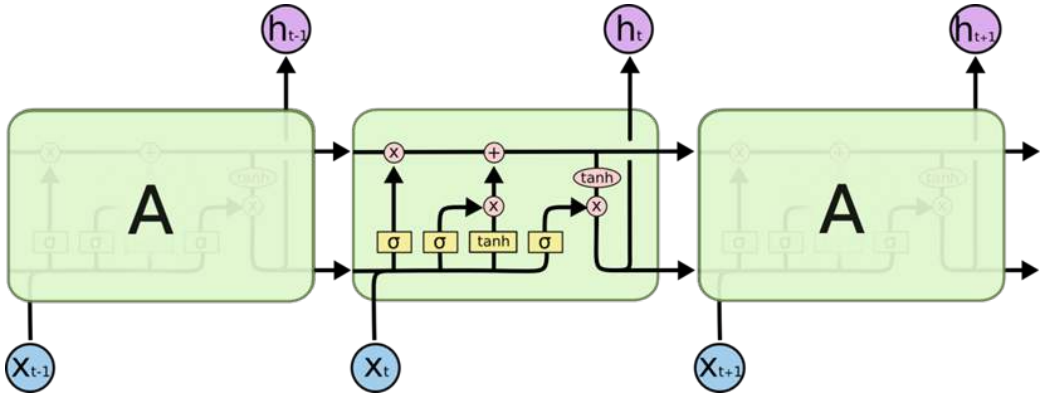
2.3.2 Long Short Term Memory (LSTM)

Long Short Term Memory sítě, neboli také LSTM, jsou podle [2], odkud celá tato kapitola čerpá, speciálním druhem rekurentních sítí, které se umí naučit dlouhodobé závislosti. Poprvé byly představeny v práci, kterou napsali Hochreiter a Schmidhuber [6], a následně byly velmi zpopularizovány. LSTM sítě jsou velmi verzatilní, zvládají řešit širokou škálu problému a nyní patří mezi nejběžněji používané rekurentní sítě.

LSTM sítě byly specificky vytvořeny pro to, aby se vyhnuly problémům s dlouhodobými závislostmi, proto je jejich schopnost pamatovat si informace po dlouhé časové úseky prakticky zabudovaná v jejich struktuře a není ji potřeba složitě učit.



Obrázek 2.9: Opakující se modul v běžných rekurentních sítích. Převzato z [2]



Obrázek 2.10: Opakující se modul v běžných LSTM sítích. Převzato z [2]

Všechny rekurentní sítě tvoří formu posloupnosti opakujících se modulů neuronových sítí. V běžných rekurentních sítích, tento opakující se modul mívá velmi jednoduchou strukturu (Obr. 2.9).

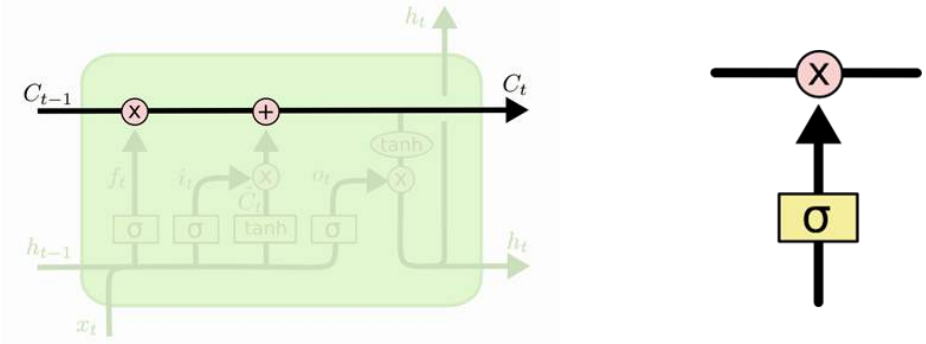
LSTM sítě také obsahují podobnou zřetězenou strukturu, na rozdíl od běžných rekurentních sítí má však jejich modul odlišnou strukturu. Místo, aby obsahoval prvky odpovídající jedné vrstvě obyčejné neuronové sítě, jsou v něm prvky odpovídající čtyřem vrstvám, které spolu speciálně spolupracují.

Funkčnost LSTM sítí

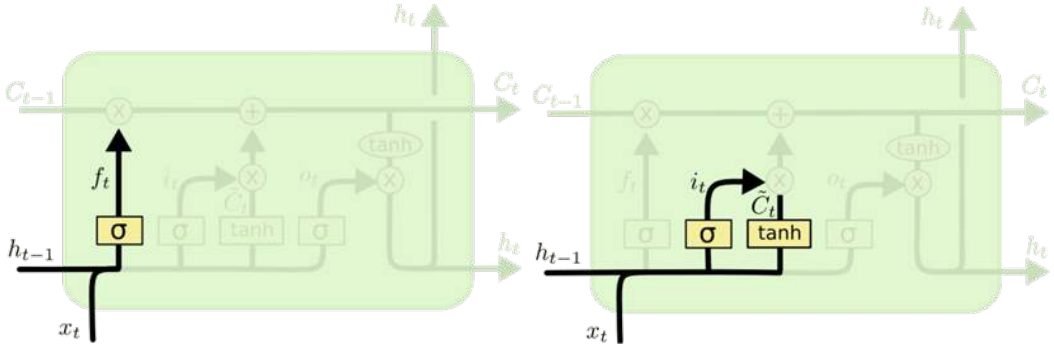
Hlavním bodem LSTM sítí je jeho jednotkový stav (cell state), který je v obrázku 2.10 představován horizontální linií vedoucí skrze horní část. Cell state probíhá přes celou posloupnost modulů a samostatně na něm probíhají pouze minimální lineární interakce. Pro informace je snadné, aby nebyly přes celou svou cestu vůbec změněny.

LSTM sítě mají schopnost přidávat a odebírat informace do (popřípadě z) cell state. Tato schopnost je však velmi opatrně regulována za pomocí bran (gates). Brány představují cestu jak volitelně povolovat informacím vstup do cell statu. Skládají se z sigmoidové vrstvy neuronové sítě a násobící operace, která pracuje bod po bodu.

Z vrstvy se sigmoidové funkcí vystupují čísla mezi nulou a jedničkou, které popisují, jak moc má každý komponent nechat projít. Hodnota nula znamená, že by se nemělo propustit



Obrázek 2.11: **Vlevo:** Cell state probíhající jedním modulem **Vpravo:** Gate prvek. Převzato z [2]



Obrázek 2.12: **Vlevo:** Zapomínající vrstva pro odstraňování dat z aktuálního cell state. **Vpravo:** Část modulu zabývající se tvorbou aktualizačních dat. Převzato z [2]

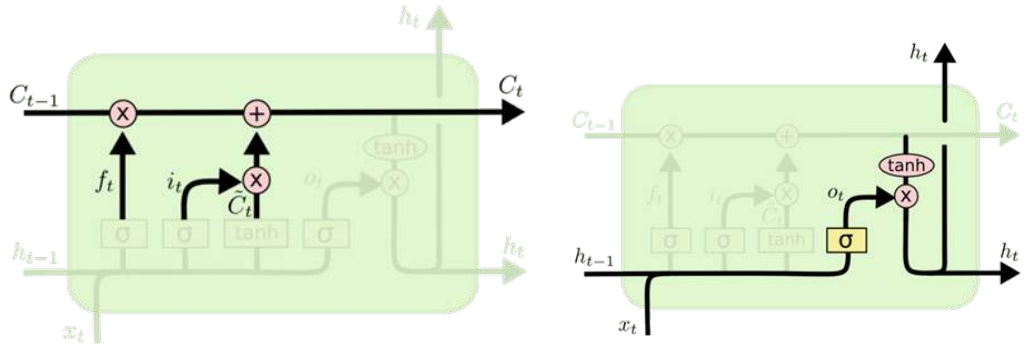
nic, zatímco hodnota jedna způsobí, že se nechá projít všechno. LSTM má celkově tři tyto brány, což umožňuje kontrolovat cell state.

První krok v LSTM síti, jak vysvětlují v [2], je rozhodnout, kterou informaci je potřeba odstranit z cell state. Toto rozhodnutí je učiněno sigmoidovou vrstvou nazývanou "zapomínající vrstva"(forget gate layer). Tato vrstva se podívá na výstup z minulého modulu (h_{t-1}) a aktuální vstup (x_t) a vydá číslo mezi 0 a 1 pro každé číslo v cell statu(C_{t-1}), který se získal z minulého modulu. Toto číslo je v intervalu 0 až 1 a určuje, jak důležité je danou informaci si zapamatovat (funkce brány z minulé sekce).

Například pokud je cílem předpovídat nové slovo podle předchozích slov, cell state může obsahovat informaci pohlaví, aby bylo možné vybírat vhodné zájmena. Pokud se ale na vstupu objeví nová osoba, pravděpodobně bude nutné zapomenout pohlaví předchozí osoby.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2.1}$$

Dalším krokem je podle [2] rozhodnutí, které nové informace se budou ukládat do cell statu. Toto se dělí na dvě části. První, sigmoidová vrstva nazývaná "vstupní vrstva"(input gate layer) rozhodne, které hodnoty se budou aktualizovat. Druhá, tanh vrstva vytvoří vektor nových kandidátních hodnot ($\tilde{C}_t$), který může být přidán do cell state. V dalším kroku se tyto dvě části zkombinují, čímž se vytvoří aktualizační data pro cell state. Jako



Obrázek 2.13: **Vlevo:** Celá část LSTM modulu zabývající se aktualizací cell state. **Vpravo:** Část modulu, ve které probíhá tvorba výstupu a zasílání do dalšího modulu, popřípadě na jiný výstup. Převzato z [2]

příklad, pokud se v minulém kroku odstranily informaci o pohlaví osoby, pravděpodobně se bude chtít vytvořit nová, které ji nahradí.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{2.3}$$

V [2] je dále řečeno, že následně je potřeba aktualizovat minulý cell state (C_{t-1}) a vytvořit z něho nový cell state (C_t). V minulých krocích se už provedla rozhodnutí o datech, které budou aktualizovány, a je nyní tedy nutné pouze provést tyto změny. Vynásobí-li se minulý stav vektorem f_t, způsobí to odstranění informací, které se zapomenuli v prvním kroku. Následně se přičte vektor $\tilde{C}_t$, který obsahuje nové kandidátní hodnoty, zmenšené podle toho, jak moc je potřeba danou hodnotu aktualizovat. V tomto kroku se tedy provádí aktualizace cell statu, a to jak odstraňování informací, tak jejich přidání.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.4}$$

Nakonec je podle [2] potřeba se rozhodnout co půjde na výstup modulu. Tento výstup bude založen na aktuálním cell state, bude se ale ještě přefiltrovávat. Nejprve použijeme sigmoid funkcí brány, což rozhodne, které části cell state do výstupu přejdou. Cell state následně proložíme funkci tanh, která přemění hodnoty cell state na hodnoty mezi -1 a 1, a následně je vynásobíme vektorem sigmoid brány, což propustí pouze ty hodnoty, u kterých se rozhodlo, že je chceme ve výstupu.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \tanh(C_t) \tag{2.6}$$

Kapitola 3

Související práce

Jak je řečeno v [11], jedna z prvních prací zabývajících se komprimaci obrazu pomocí neuronových sítí byla napsána již v roce 1999 [7], ve které neuronové sítě nahrazují některé části standardních řetězců pro komprimaci, a následně se učí lepším parametrům. V roce 2006 se objevily autoenkodéry, které umožňovaly implementovat celou síť jako kompresní nástroj.

Mezi hlavní práce, ze kterých se zde vychází, jsou však práce George Todericiho a kol. [11][12], které se přímo zabývají různými architekturami pro kompresi obrazu. V jeho první práci o kompresi obrazu pomocí neuronových sítí z roku 2016 [11], se zabývá redukcí zmenšených obrázků na velikosti 32x32 pixelů, neboť pro tyto typy obrázků moderní kompresní metody, jako je např. JPEG, nejsou nejvhodnější. V této práci představili několik architektur sítí založených převážně na principu autoenkodéru, se kterými experimentovali. Tyto sítě jsou podrobněji představeny v následujícím segmentu.

V druhé práci George Toderici o kompresi obrazu [12] se zabývá rozšířením sítí, aby dokázaly komprimovat obrazy s jakýmkoli rozlišením. V této práci používají jednu architekturu rekurentní sítě, u které experimentují s různými druhy rekurentních modulů a s více metodami rekonstrukce obrazu. Tyto prvky budou také ukázány v příští části.

3.1 Stávající architektury kompresních sítí

Tato kapitola obsahuje popis architektur sítí a metod používaných pro kompresi obrazu pomocí neuronových sítí. Informace pocházejí hlavně z prací George Todericiho [11][12], kde na těchto sítích experimentuje. Pro každou architekturu probereme funkci E, která představuje enkodér a přijímá na vstupu obrázkové data a produkuje zakódovanou reprezentaci těchto dat. Tato data jsou dále zpracována binarizační funkcí B, která je stejná na všech architekturách, a vytváří binární reprezentaci dat. Pro každou architekturu je pak ještě potřeba funkce dekodéru D, která z binárních dat vyprodukuje rekonstruovaný obraz. Dohromady tyto tři funkce tvoří základ autoenkodéru (3.1).

$$x_{rst} = D(B(E(x))) \tag{3.1}$$

Tyto konceptuální prvky jsou sdíleny ve všech následujících architekturách a v případě některých modelů se používá i postupného reziduálního výsledku, kdy se pro každý vstup opakuje průběh sítě v cyklech, ale jako vstup se pak používá chyba (reziduum) předchozího cyklu. Cílem je snížení reziduální chyby, jak dostáváme více a více informací pro dekodér. Řetězení kopií reziduálního autoenkodéru se dá vyjádřit jako rovnice 3.2 popsaná v [11]

$$F_t(r_{t-1}) = D_t(B(E_t(r_{t-1}))) \tag{3.2}$$

Kde F_t představuje reziduální autoenkóder, r_0 představuje původní obrázek a r_t pro t > 0 obsahuje reziduální chybu po t cyklech. Pro modely sítí, které neobsahují LSTM prvky, F_t nemá žádnou paměť a předpovídá pouze samotné reziduum. Plná rekonstrukce je pak vytvořena součtem všech reziduí. Pokud ale F_t má paměť, autoenkodér vypočítává samotný obrázek v každém cyklu.

3.1.1 Binarizační funkce

Binarizační funkce, tak jak je uvedená v [11], je stejná pro všechny uvedené architektury a její výstup představuje komprimovaný obraz. Používá se, protože bitové vektory jsou velmi snadně serializovatelné a de-serializovatelné. Dále můžeme měnit míru komprese pouze omezením na velikost výsledných bitů a také pomáhá síti lépe se naučit efektivnější reprezentaci.

Binarizační proces se skládá ze dvou částí. První část generuje potřebný počet výstupů (tak velký, jaký chceme výstupní počet bitů) v intervalu reálných čísel $[-1, 1]$. Druhá část vezme tuto reprezentaci jako vstup a produkuje diskrétní výstup s hodnotami {-1, 1} pro každou vstupní hodnotu.

Pro první část v binarizaci se může použít plně propojená vrstva s tanh aktivační funkcí. Druhá část se může řešit podle metody, kterou popsal Raiko a kol. [9], kde binarizaci $x \in [-1, 1]$ provádíme jako rovnice 3.3 a 3.4, které jsou také popsány v [11].

$$b(x) = x + \epsilon \in \{-1, 1\} \tag{3.3}$$

$$\epsilon \sim \begin{cases} 1 - x, & \text{s pravděpodobností } \dfrac{1+x}{2} \\ -x - 1, & \text{s pravděpodobností } \dfrac{1-x}{2} \end{cases} \tag{3.4}$$

Kde ϵ představuje kvantizační šum. Regularizace, kterou dodává náhodná kvantizace, se dá využít pro backpropagaci gradientů zpátky skrz binarizační vrstvu.

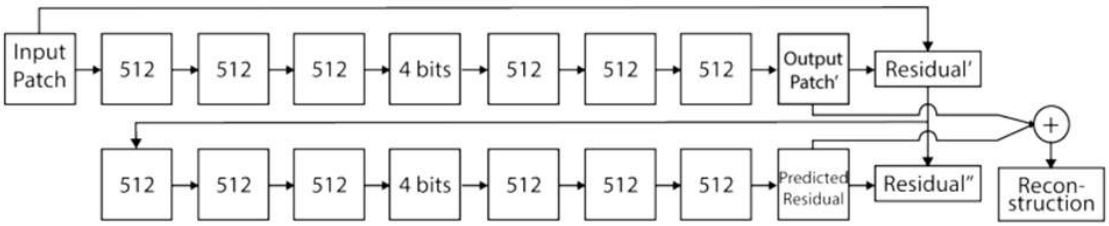
$$B(x) = b(\tanh(W^{bin}x + b^{bin})) \tag{3.5}$$

V kompletní binární funkci 3.5 z práce [11] jsou W^{bin} a b^{bin} standardní lineární váhy a bias z minulé vrstvy v síti. Ve všech následujících modelech se formule 3.5 používá při dopředném průchodu sítí. Pro backpropagaci se použije derivace předpokladu, a protože předpoklad je pro všechny $x \in [-1, 1]$ stejný jako x, gradienty se můžou nechat projít přes tuto vrstvu nezměněny.

Aby mohla existovat vždy stejná reprezentace pro každý konkrétní vstup, jakmile se síť natrénuje, používá se pouze nejpravděpodobnější výsledek b(x) a b může být nahrazeno b^{inf}, které je definováno jako rovnice 3.6.

$$b^{inf}(x) = \begin{cases} -1 & \text{pokud } x < 0 \\ 1 & \text{jinak} \end{cases} \tag{3.6}$$

Velikost komprese je určena počtem bitů generovaných v každé cyklu, což odpovídá počtu řádků v matici W^{bin}, a počtem cyklů, pokud je používán reziduální autoenkodérový model.



Obrázek 3.1: Struktura plně propojeného reziduálního autoenkodéru. Je zde znázorněna struktura prvních dvou iterací, kde cílem první iterace je zakódování originálního vstupního obrázku, zatímco druhý cyklus se snaží o zakódování reziduální chyby prvního cyklu. Každý cyklus má výstup na 4 bitech a podle velikosti komprese, které se chce dosáhnout, se pouze zvolí odpovídající množství cyklů. Převzato z [11]

3.1.2 Plně propojený reziduální autoenkodér

Plně propojený reziduální autoenkodér [11] je nejjednodušší ze všech zde popsaných modelů. Enkodér (E) a dekodér (D) jsou složeny čistě z plně propojených vrstev. Počet výstupu v každé plně propojené vrstvě je nastaven na konstantních 512 a jako aktivační funkce se používá pouze tanh. Jako loss funkce se používá L2.

Kvůli tomu, že rezidua mění svoje vlastnosti v závislosti na aktuální iteraci autoenkodéru, existují dva postupy jak přistupovat k tomuto modelu. První sdílí váhy skrze všechny iterace, zatímco druhý se učí váhy v každé iteraci nezávisle na sobě.

3.1.3 Plně propojený LSTM reziduální enkodér

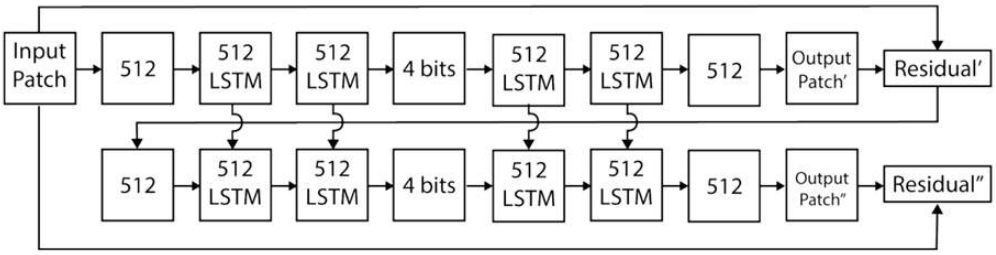
V tomto modelu z [11] se používat LSTM vrstev jak v enkodéru, tak i v dekodér. V enkodéru se používá jedna plně propojená vrstva následovaná dvěma LSTM vrstvami. Struktura dekodéru je opačná, nejprve leží dvě LSTM vrstvy a za nimi plně propojená vrstva s tanh aktivační funkcí, která předpovídá barevné složky (není uvedená v obrázku). Na rozdíl od předcházejícího modelu je výstupem každé iterace enkodéru předpověď skutečného obrázku a ne pouze předchozí reziduum, jak už bylo zmíněno dříve.

3.1.4 Konvoluční/dekonvoluční reziduální enkodér

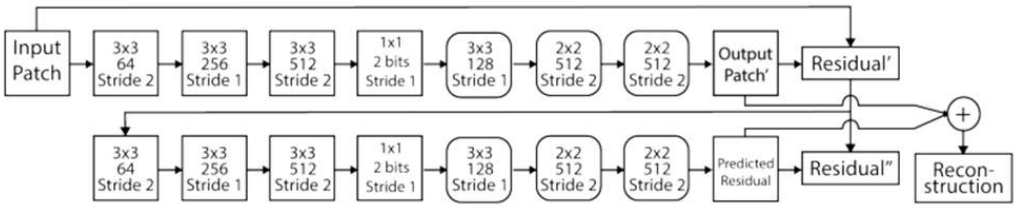
Konvoluční/dekonvoluční reziduální enkodér, tak jak je popsán v [11], je velmi podobný plně propojenému autoenkodéru z části 3.1.2. Změny jsou takové, že plně propojené vrstvy v enkodéru byly nahrazeny konvolučními vrstvami a v dekodéru byly nahrazeny dekonvolučními vrstvami popsanými v části 2.2.4. Poslední vrstva dekodéru obsahuje dekonvoluční masku o velikosti 1x1 s třemi filtry, které konvertují aktuální informaci do tří RGB barevných kanálů.

3.1.5 Konvoluční/dekonvoluční LSTM síť

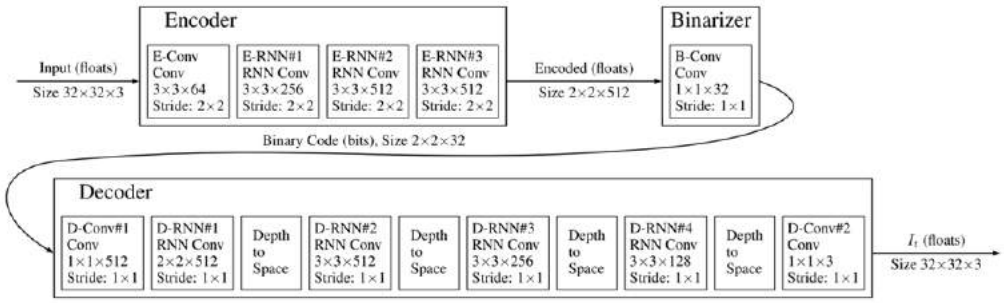
Tato síť z [11] kombinuje konvoluční a dekonvoluční vrstvy společně s LSTM prvky dvou předchozích sítí, čímž získává některé výhody obou sítí. Struktura sítě je velmi podobná jako předchozí konvoluční/ dekonvoluční síť, v enkodéru se pouze vymění druhá a třetí konvoluční vrstva za speciální LSTM konvoluční vrstvu, která má oproti obyčejné LSTM



Obrázek 3.2: Struktura plně propojeného LSTM reziduálního enkodéru. Jsou zde opět znázorněny první dva cykly rozbaleného reziduálního enkodéru. Samotný enkodér by obsahoval pouze první řádek modelu a následná funkčnost by se zajistila předkládáním reziduální chyby z předchozího cyklu zpátky na vstup. Podobně jako předcházející model má bloky s 512 výstupy v každé vrstvě. Šipky propojující LSTM bloky znázorňují paměťové závislosti, které jsou běžné u rekurentních sítí. Jako loss funkce se opět používá L2. Převzato z [11]



Obrázek 3.3: Struktura plně konvolučního/dekonvolučního reziduálního enkodéru(pouze první dvě iterace). Konvoluční vrstvy jsou na obrázku znázorněny jako čtverce s ostrými rohy, zatímco dekonvoluční vrstvy mají zaoblené rohy. Síť neobsahuje rekurentní prvky, v každém cyklu se tedy vypočítává pouze reziduum. Převzato z [11]



Obrázek 3.4: Struktura reziduální sítě s prvky rekurentních a konvolučních sítí. Základní struktura pro experimentování s různými druhy rekurentních jednotek. Převzato z [12]

vrstvě upravenou strukturu, aby mohla pracovat s 2D parametry jako je stride, šířka, výška apod. Podobným způsobem je řešen dekodér, kde je také vyměněna druhá a třetí vrstva.

Tato síť byla používána pro obrázky o rozlišení 32x32 a ukázala se být jako nejefektivnější z předchozích sítí. Následná síť je vylepšenou verzí této sítě a používá se pro obrázky s libovolným rozlišením.

3.1.6 Rekurentní/konvoluční síť pro kompresi obrazu s libovolným rozlišením

Tato síť, použitá a popsaná v práci [12], podobně jako minulá, obsahuje prvky jak konvolučních tak rekurentních sítí. Na rozdíl od minulé sítě je tato síť však plánována pro obrazy s jakýmkoli rozlišením. Této schopnosti však není dosaženo pomocí zvětšení vstupního obrazu, který zůstává pořád pro obrazy 32x32, ale strukturou sítě, která je více uzpůsobená různorodým datům a umí lépe navazovat části při rekonstrukci.

Do enkodérové i dekodérové části sítě byla oproti minulým sítím přidána jedna další vrstva a byly upraveny některé další parametry sítě, jako je například velikost konvolučního jádra nebo stride. Největší změnou však je, že se s touto sítí vyzkoušely různé druhy rekurentních jednotek (modulů) mezi které patří běžné LSTM, upravené pro lepší práci s daty konvolučních vrstev (místo násobení se používá konvoluce), asociativní LSTM, které používají holografickou reprezentaci a pracují s komplexními čísly, nebo Gated Reccurent Units (GRU).

Kapitola 4

Návrh

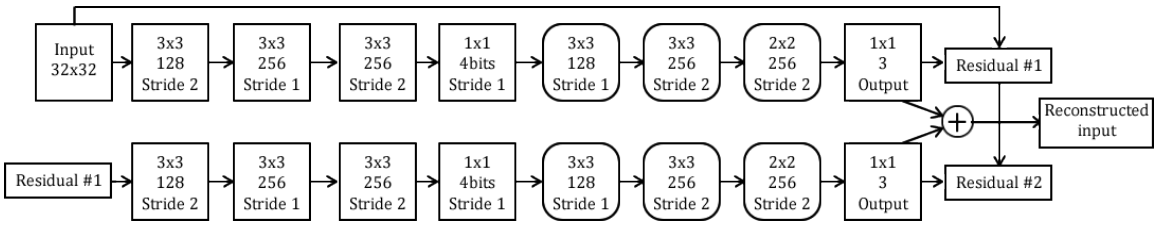
Tato kapitola popisuje základní návrh modelů a experimentů, které budou použity v této práci. Jako základ využiji model neuronové sítě, který používá jak principy rekurentních sítí (tím zde myslím, že budu využívat principy rekurentních jednotek, což znamená opakované využívání dříve použitých dat v pozdějších částech sítě, a nemusí to být např. kompletní LSTM jednotka), tak i principy konvolučních sítí (konvoluční vrstvy, správně zakomponované s rekurentními principy, tj. vytvořené tak, aby se informace dala předávat mezi jednotlivými vrstvami).

Podobně jako v práci George Todericiho, ze které budu v první řadě vycházet, hlavní síť bude založena na principu autoenkodéru. Bude tedy složena z enkodéru (část pro zakódování vstupního obrazu), bottleneck části, která vytváří reprezentační výstup obrazu (komprimovaný obraz), a dekodéru, který převádí reprezentační výstup zpátky do podoby co nejvíce podobné vstupnímu obrazu. Je možné, že budu experimentovat i jinými druhy architektur, této architektuře se však budu věnovat nejvíce.

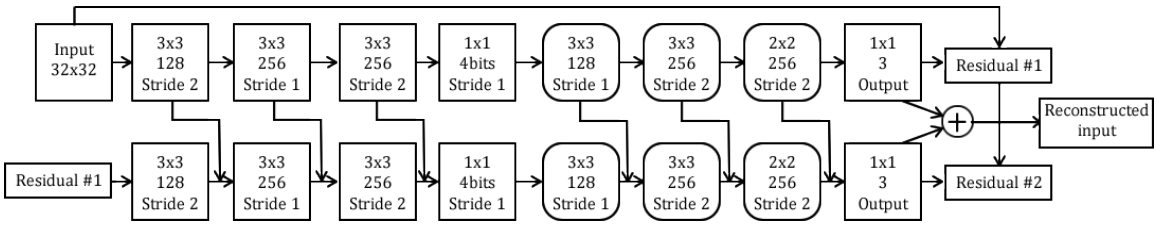
Všechny sítě budou stejně jako dříve zmíněné sítě (část 3.1) v každé iteraci získávat z binarizační vrstvy množinu bitů, které představují část celkové zakódované informace. Pro moje sítě budu používat 16iterační modely sítí s finální kompresí čtyř bitů na pixel. Toto je dvojnásobné množství bitů, než je použito v práci George Todericiho, kvůli možnosti použít pouze data z méně iterací však toto nepředstavuje problém a při finálním srovnání kvality sítí budou nejlepší modely otestovány i při větších kompresních poměrech.

4.1 Výchozí modely sítí

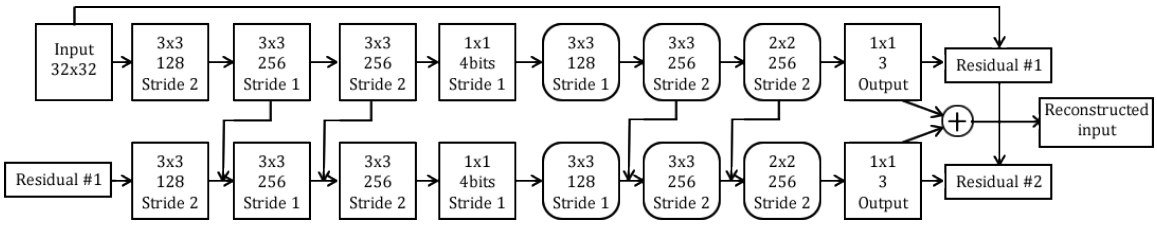
Dva základní principy modelů, na kterých budu experimentovat, jsou čistě reziduální konvoluční sítě (4.1) a reziduální konvoluční sítě s rekurentními prvky. Po otestování výsledků mezi těmito dvěma sítěmi, budu následně experimentovat se změnami parametrů na tom typu sítě, který bude dávat lepší výsledky v hodnotách MS-SSIM. U sítě s rekurentními prvky (předávání informace do budoucích iterací) budu navíc testovat různé způsoby předání této informace. Jako jedna z možností bude testováno předávání výsledku vrstvy součtem se vstupem vrstvy v budoucí iteraci. Jinou možností je předávat jako konkatenaci k danému vstupu. Další možnosti testování u těchto sítí je množství budoucích iterací, do kterých se informace výstupu vrstvy bude předávat, nebo pozice v iteraci (před kterou vrstvu), do které se informace přidá.



Obrázek 4.1: Dvě iterace modelu reziduální konvoluční sítě na principu autoenkodéru. Jedna iterace je složena s tří konvolučních vrstev, binarizařní vrstvy, tří dekonvolučních vrstev a konvoluční vrstvy převádějící na tří kanálový obraz. Výchozí reziduální síť s parametry, které se budou měnit v závislosti na experimentovaném parametru.



Obrázek 4.2: Dvě iterace modelu sítě s rekurencí. Výstup konvolučních a dekonvolučních vrstev se kombinuje s výstupem té samé vrstvy v následující iteraci, jinak je kvůli porovnatelnosti výsledku totožná ve všech parametrech se sítí 4.1



Obrázek 4.3: Dvě iterace modelu sítě. Zasílá výstup vrstvy na vstup stejné vrstvy v následující iteraci, jinak totožná se sítí 4.2. Kvůli rozdílnosti parametrů výstupu a vstupu první konvoluční a dekonvoluční vrstvy, tyto vrstvy neobsahují rekurenci.

4.2 Zakladní experimenty nad parametry

Přesto, že v předcházejících pracích bylo vyzkoušeno značné množství sítí s různými parametry a metodami, zůstává velké množství prvků, které se dají dále upravovat. V experimentech plánuji vyzkoušet měnit různé parametry, a to jak základní, tak i prvky, které mění celou strukturu sítě. Postupně budu z výsledků jednotlivých experimentů brát všechny koncepty, které nějakým způsobem zlepšují výslednou kompresi obrazu nebo jiné aspekty sítě.

Mezi základní prvky, které budu měnit, patří počet vrstev v jednotlivých prvcích (enkodér, dekodér) sítě. V uvedených modelech se používají převážně tři nebo čtyři vrstvy pro každý prvek. Při zvětšení počtu vrstev by se teoreticky mohla zvýšit kvalita dekódovaného obrazu, kterou by síť měla umět vyprodukovat, díky možnosti reakce na větší počet kombinací příznaků v obraze. To je však způsobeno na úkor rychlosti sítě a hlavně také paměťových nároků, které je potřeba na uložení modelu. V mém návrhu bych chtěl docílit takových paměťových nároků, aby se daly snadno používat i s grafickými kartami, které jsou i několik let staré, ale zároveň aby byla rychlost dostatečná. V případě zlepšení výsledků při použití většího počtu vrstev v jedné iteraci sítě budu experimentovat s více možnostmi počtu vrstev pro určení, kdy se zlepšování vlastností sítě zastaví. V případě zlepšování vlastností budu také experimentovat nad rozdílným počtem vrstev v dekodéru a enkodéru a také v jednotlivých iteracích.

Dalším prvkem, nad kterým budu experimentovat, je změna velikosti rozlišení vstupního obrazu. Ve všech dříve zmíněných modelech byl jako vstup používán obraz o rozlišení 32x32. Testováním na různých velikostech rozlišení (pravděpodobně stále čtvercového tvaru, kvůli funkčnosti konvolučních filtrů v konvolučních sítích) bych chtěl zjistit vhodnou velikost rozlišení pro optimální výsledky. Velikost rozlišení navíc hodně ovlivňuje ostatní prvky sítě. Například s menším rozlišením je pravděpodobné, že se zmenší paměťové nároky sítě. Celkově hlavní rozdíl v kvalitě výsledků tohoto experimentu očekávám, že se zobrazí na místech v obraze, kde se při rekonstrukci obrazu jednotlivé části spojují. V případě menších vstupů se budou tyto oblasti vyskytovat častěji než při použití větších vstupů. Zde bude záviset na samotné síti, jak kvalitně bude umět zakódovávat okraje vstupu, i když kvůli nekompletní informaci obrazu se do určité míry budou určité grafické artefakty pravděpodobně objevovat vždy. Při kompresi větších obrazů bude navíc těžší velikostně přizpůsobit obraz na správnou velikost, pokud použijeme větších vstupy, na rozdíl od menších vstupů, na které se dá obraz lépe rozložit.

Plánuji také otestovat změny, které přinese zmenšení a zvětšení konvolučních oken (v zásadě se hodně využije v kombinaci se změnou rozlišení vstupního obrazu). Větší konvoluční jádro by teoreticky mělo přinést zlepšení kvality výsledného obrazu díky zvýšení dostupných dat pro zakódování jednoho pixelu. Na druhou stranu díky naučení se spoléhat na všechny body v konvolučním oknu může způsobit zvětšení chyby na okrajích výstupného obrazu, což je způsobené nekompletní informaci o pixelech za okrajem obrazu (v případě komprese obrazu složeného z více oblastí).

Společně s velikostí konvolučních filtrů budu experimentovat i se samotným počtem konvolučních filtrů. Zvětšení počtu filtrů by mělo zlepšit kvalitu výsledného obrazu, za cenu lineárního navýšení potřebné paměti. Měl by existovat určitý limit počtu konvolučních filtrů, který by měl pokrýt všechny možnosti, které mohou nastat z daného vstupu sítě (závislé na velikosti konvolučního okna). V části dekodéru je tento limit mnohem nižší kvůli bottleneck vrstvě, která významně snižuje celkový počet potencionálních možností. Tento princip jde ale prakticky aplikovat pouze na originální vstup se třemi kanály (RGB), popřípadě na

vrstvu zpracovávající výstup enkodéru. Následující vrstvy mají teoreticky exponenciálně větší množství možností.

Možnost otestovat je i kvalitu sítě v závislosti na množství detailů ve trénovaných datech. Určitým zvětšením či zmenšením rozlišení těchto dat se sníží, respektive zvýší množství rychlých přechodů mezi barvami v obraze. Síť by následně měla umět lépe zkomprimovat data o vyšších respektive nižších frekvencích, na úkor těch druhých, které se následně budou objevovat v trénujících datech méně často. Celkově tedy bude výsledek pravděpodobně záviset na tom, jak velké procento běžných obrázků obsahuje rychlé přechody či souvislé plochy.

Další testovací parametr představuje loss funkce podle které se daná síť trénuje. Plánuji otestovat loss funkce L1, L2 a také možnost trénovat síť přímo podle MS-SSIM parametru. Zároveň je možné otestovat i learning rate sítě a optimalizér, i když tyto prvky nepředstavují praktický dopad na výslednou kvalitu komprese jako spíše na rychlost trénování sítě.

4.3 Pokročilé experimenty

Kromě změny základních parametrů sítě, mám taky v plánu zkusit upravit celou strukturu sítě tak, aby síť obsahovala větve nebo části, které by se specializovaly na vstupní obraz (již rozdělený na díly) podle jeho určitých vlastností. Každá tato větev by se teoreticky měla natrénovat pouze na vstupních obrazech, které přísluší její specializaci, což by teoreticky mělo zlepšit kvalitu obrazu. Pro dekodér by se informace, která větev byla použita, zapsala ve formě několika bitů (v případě čtyř větví by byla potřeba 2 bitů). Tento princip by měl teoreticky dobře spolupracoval se vstupy s větším rozlišením, které by obsahovaly poměrově více zakódovaných bitů a metadata, která by určovaly větev, by se využívaly méně často a mohlo by se teda použít i více větví (9 nebo 16 pro 3 resp. 4 bity).

Při experimentování se budou níže uvedené úpravy kombinovat s některými dříve zmíněnými změnami v předchozí části 4.2. Nejdůležitější vlastnost, která se bude měřit je kvalita výsledku. Ostatní charakteristiky, jako je rychlost výpočtu nebo paměťová složitost, jsou pouze doplňujícími vlastnostmi.

4.3.1 Rozdělení barev

První experiment s podobným principem bude rozdělení enkodéru na čtyři různé větve, kde každá větev bude mít na vstup jednu základní barvu RGB nebo jasovou složku obrazu. Jasovou složku vypočítám pomocí rovnice. Provedu celkově dva experimenty, kde jeden bude mít tyto čtyři větve spojené buď pomocí konkatenace nebo součtu před vstupem do binarizační vrstvy, zatímco druhý bude mít v binarizační vrstvě každou větev zakódovávat separátně.

Cílem těchto experimentů je zjistit, zda specializování sítě na úrovni pouze části modelu sítě povede ke zlepšení výsledků.

4.3.2 Plně specializované sítě

Pro tento experiment, který slouží jako test pro zjištění, jak velký rozdíl způsobí použití velmi specializovaného datasetu, jsem se rozhodl použít geografický dataset map. Důkladnější popis datasetu je popsán v části ref, ale důraz povedu i přes nutnost specializace na možné pozdější využití kompresních sítí na kompresi mapových dat, takže se pokusím pokrýt veškeré hlavní přírodní prostředí (tj. lesy, pole, hory, pouště města apod.). I částečná

specializace datasetu by však díky redukci možností, jakých může nabrat podoba vstupu, měla způsobit zlepšení kompresních vlastností sítě na daném specializovaném datasetu. Problémy v praktickém využití by ale následně mohly nastat v nekompletním trénujícím datasetu, kdy se síť nenatrénuje na velmi vzácných přírodních úkazech (např. zasněžené vrcholky hor), nebo přetrénování, kdy je síti předloženo nepoměrně větší množství dat z jednoho biotopu (např. pole či oceány).

4.3.3 Umělé specializované mapy

Posledním prvkem sítí, se kterým budu experimentovat, je umělé specializování sítí, což je do určité míry kombinace dvou předchozích principů 4.3.1, 4.3.2. V zásadě se jedná o rozdělení vstupu do různých větví. Místo toho aby se ale rozdělilo do různých větví v jednom modelu sítě, jako je tomu u dříve zmíněných sítí 4.3.1, rozdělí se vstup do různých modelů sítí, které jsou předem natrénované pouze na daný interval vstupů rozhodnutých podle určité metriky (z tohoto umělá specializace).

Principiálně se tedy musí rozdělit interval všech možných vstupů na nejlépe 2^x částí, kdy se pro každou tuto část natrénuje síť pouze na vstupech spadajících do toho intervalu. Při kompresi celého obrazu se tento obraz rozdělí na části a následně se jako první možnost všechny části zpracují všemi modely sítí a následně se pomocí hodnotící metriky (např. MS-SSIM) zrekonstruují do jednoho obrazu a pro zkomprimovaný obraz se vyberou jen tyto části. Druhá možnost je pomocí určité metriky zhodnotit vstupní část, a zpracovat ji pouze tou sítí, která je na to specializovaná.Pro obě možnosti bude potřeba přidat ke každému bloku x bitů, kterými se určí zpracovávající větev. Toto množství je však oproti velikosti celého bloku zanedbatelné.

Pro nejlepší funkčnost této sítě je nejpodstatnější vhodně vybrat metriku, podle které se bude rozdělovat interval vstupů a také vhodně vybrat množství a hranice těchto intervalů. Metrika by měla umět vhodně popsat celý interval tak, že konec intervalu nebude ležet v nekonečnu.

Experimentováním budu zkoušet zjistit, jak kvalitní bude kombinace některých dříve vzájemně vylučujících se vlastností (trénování na vstupech s nízkou a vysokou frekvenční složkou). Další experimenty budou s metrikou zaměřenou na ostrost, která je odvozena od gradientu obrazu, nebo také jak velké budou rozdíly při různém rozdělení trénujících intervalů.

4.4 Encoding

Kódování a komprese je funkce binarizační vrstvy a představuje převod hodnot sítě v dané vrstvě na bity v závislosti na binarizační funkci, která je použita stejná jako použil George Toderici (3.1.1). Nad touto částí nebudu provozovat žádné experimenty, podívám se však na možnosti následné komprese výstupu této binarizační vrstvy. V zásadě budu hledat možnost jak přeskupit daná data tak, aby se dala jejich velikost dále podle nějaké metody zmenšit. Vycházím z principu, že v případě podobnosti okolních pixelů v obraze, bude i jejich zakódovaná reprezentace vykazovat určitou podobnost. Testování provedu pomocí zakódování do formátu .7z, který používá mimo jiné i metody entropického kódování, a budu hledat uskupení dat, které bude přinášet největší kompresi.

4.5 Dataset

Abych mohl porovnávat své výsledky s výsledky práce George Todericiho, bylo by vhodné použít stejný trénovací a testovací dataset. V jeho práci je však pouze uvedeno, že používají náhodně vybraných šest milionů obrázků získaných z internetu, které se rozdělily na díly s 32x32 rozlišením. Z těchto dílů se následně vybíraly ty, které dosahovaly nejhorší míry komprese při zakódování PNG algoritmem.

Protože nemám k dispozici přímo tento dataset, pro experimenty budu používat MIRFLICKR dataset, který obsahuje jeden milión barevných obrázků získaných z portálu Flickr, všechny pod Creative Commons licencí.

Dataset pro plně specializované sítě 4.3.2 bude tvořen mapami dostupnými z aplikace Google Earth Pro. Je vybrána plocha mezi rovnoběžkami o zeměpisné šířce 30° a 45° a poledníky o zeměpisné délce -115° a -80°. Dataset je tvořen satelitními snímky o rozměrech 256x256 pixelů a je focen z výšky pohledu přibližně 20 km. Tato oblast spadá pod uzemí Spojených států amerických a obsahuje různé přírodní prostředí, včetně pohoří, lesů, pouští ale i množství urbanizovaných ploch. Z největší části však dataset obsahuje oblasti pro zemědělskou výrobu. V experimentech se ukáže jak velký rozdíl je mezi kvalitou zkomprimovaných částí s velkým procentem výskytu a částí, které se vyskytují v datasetu vzácněji.

Pro testovací účely bude využit LIVE dataset, který obsahuje 29 obrázku s vysokým rozlišením. Všechny výsledné hodnoty, které budu v pozdějších experimentech uvádět, budou získané z testování nad touto sadou. Jedinou výjimkou budou experimenty nad plně specializovanými sítěmi, kde je potřeba vytvořit speciální dataset. Ten bude vytvořen náhodným vybráním 200 obrázků z trénovacího datasetu.

4.6 Tensorflow

Jako nástroj, který použiju pro tvorbu sítě a následně experimentováním nad ní, jsem zvolil TensorFlow, což je open source knihovna pro matematické výpočty za použití dataflow grafů. Knihovna umožňuje využívat jak procesor, tak grafickou kartu pro své výpočty. TensorFlow se dá použít s programovacími jazyky jako je C++ a Java, ale největší rozšíření má jazyk Python, který použiju pro tvorbu sítě i veškeré logiky.

4.7 Hodnotící metody

Protože neuronové sítě komprimují a následně rekonstruují tak, aby byl výsledek nejlepší pro lidské vidění, měla by se pro zhodnocení výsledků sítě používat metoda nebo metody, které toto berou v potaz. Současně neexistuje žádná metoda, nad kterou by byl všeobecný souhlas, že nejlépe hodnotí toto kritérium. Proto budu v budoucích experimentech používat MS-SSIM (Multi Scale Structural Similarity), algoritmus, který se často používá pro porovnávání ztrátových algoritmů. Jedná se o stejný algoritmus jako byl použil v pracích George Todericiho [11] .

MS-SSIM se musí aplikovat zvlášť na každý barevný kanál mezi originálním a měřeným obrazem a dává výsledky v intervalu mezi 0 a 1. Vyšší hodnoty výsledku znamenají větší podobnost k originálnímu obrazu.

Kapitola 5

Experimenty a implementace

Tato kapitola představuje soupis většiny experimentů, které byly v průběhu práce testovány. Některé sítě, které byly otestovány, ale nepřinášení žádné nové relevantní informace vzhledem k ostatním sítím nejsou v této kapitole přestaveny. Experimenty jsou převážně představeny v podobném sledu v jakém byly testovány.

U většiny experimentů je ukázán graf učení sítě s hodnotami MS-SSIM, které byly naměřeny během trénování sítě. Tyto hodnoty, na rozdíl od konečné hodnoty MS-SSIM, kterou měříme kvalitu sítě, nejsou měřeny na testovacím vzorku, ale jsou vypočítány na datech, kterými je síť aktuálně trénovaná, proto je nutné brát v potaz určitou nepřesnost v hodnotách, i když je tato nepřesnost převážně zanedbatelná.

Zároveň je zde popsána i implementace některých důležitých části práce, mezi které patří popis implementace výpočtu MS-SSIM, implementace encodingu, u kterého bude vysvětlena i implementace binarizační vrstvy, a je zde také popsána implementace samotného trénování a testování. Kapitola taky obsahuje stručný soupis knihoven Python a Tensorflow knihoven, které byly použity v rámci projektu.

5.1 Implementace trénování a testování

Pro trénování a testování neuronových sítí byla použita knihovna Tensorflow v jazyce Python. Díky stálému vývoji této knihovny bylo potřeba využít průběžně několika knihoven se začátkem ve verzi 0.9 a končíce u verze 1.3. Dále byly využity knihovny numpy pro práci s matematickými strukturami a daty, scipy a pillow pro práci s obrázky a knihovna binlayer, která je použita pro vytvoření zkomprimovaných dat.

Pro trénování bylo potřeba zajistit dodávání obrázků síti v požadovaném formátu. V případě použití trénování pouze na částech obrázků, kdy potřeba rekonstruovat obraz zpátky do původní struktury, tak Tensorflow knihovna poskytuje nástroje pro snadný přísun dat. Pokud ale je potřeba otestovat výsledný obraz, který je složený s proměnlivého počtu částí, musí se vytvořit dávka ručně. Přestože dávka (batch) může být v neuronových sítích proměnlivá, v Tensorflow knihovně je specifický případ, kdy dekonvoluční síť potřebuje znát kompletní rozměry svého vstupu a to i velikost dávka již při inicializaci. Velikost dávky se tedy nemůže přizpůsobovat aktuálnímu obrázku, ale musí být konstantní během celého trénování i testování. Poslední dávka při zpracování obrazu nemusí tedy mít dostatečné množství bloků a musí se doplnit. Jako řešení jsem použil první bloky daného obrázku znovu jako doplnění chybějících bloků.

Pro trénování byly použit learning rate 10^{-5} společně s Adam optimalizérem, který adaptivně upravuje tuto learning rate během trénování podle potřeby. Nižší hodnoty learning rate způsobovaly občasné výskyty extrémních hodnot. Společně se snížením learning rate se musely podstatně snížit iniciální hodnoty všech biasů v konvolučních i dekonvolučních sítích, neboť způsobovaly i při běžné inicializaci přes xavier_inicializátor, kdy se bralo v ohled množství vstupů a výstupů pro aktivaci neuronu, se stále objevovaly extrémní hodnoty při trénování.

Modely se testovaly tak, že jedna epocha představuje celý jeden obraz nezávisle na množství bloků v jedné dávce nebo rozměrech bloku. Je to kvůli vytvoření určitého standardu, podle kterého bude možné porovnávat jednotlivé experimenty. Například u sítí, které mají složitější strukturu, nemohly být použity tak velké dávky kvůli větším paměťovým nárokům než u sítí s jednoduší strukturou.

Specifické úpravy pro jednotlivé experimenty jsou vždy popsané u daného experimentu. Modely se ukládaly každých 250 epoch a přestaly se trénovat v případě viditelného nezlepšování sítě. Musí se brát v úvahu, že cílem práce není nalezení nejlepšího výsledku pro každou experimentovanou síť, ale najít strukturu sítě, která bude mít nejlepší výsledky, proto není důležité nechat trénovat síť několikanásobně dlouho pro získání pouze marginálního vylepšení.

5.2 Encoding a komprese

Pro implementaci binarizační vrstvy bylo potřeba zajistit, aby při trénování sítě backtracking propouštěl správné hodnoty. Po zpracování dat pomocí sigmoidy, která připraví data na zaokrouhlení, čímž se získají požadované hodnoty, nastávala v tensorflowu situace, kdy se při backtrackingu nepropouštěly správné hodnoty, což způsobovalo nesprávné trénování. Pro odstranění této nežádoucí funkčnosti bylo potřeba specificky operaci zaokrouhlení i sigmoidě napevno nastavit požadované hodnoty, které je třeba posílat při backtrackingu. Toto se zařídilo pomocí override mapy nad oběma funkcemi, která změnila výchozí hodnoty na identitu.

Formát výstupních dat z binarizační vrstvy je ve formě tří-dimenzionálního pole s hodnotami z množiny {-1, 1}. Tento výstup je v každé iteraci jeden(výjimka u sítí 4.3.1, kde existují až čtyři binarizační vrstvy, každá na jednu větev sítě) s tím, že všechny výstupy se v průběhu zpracování složí do jednoho seznamu, který je předán na konci zpracování. Následně je nutné přetvořit hodnoty z množiny {-1, 1} na množinu {0, 1}, aby tyto hodnoty mohly být reprezentovány jako binární hodnoty, což dělám již mimo tensorflow knihovnu nastavením minimální hodnoty v celém seznamu na hodnotu 0.

Samotný zápis již takto vygenerovaných a upravených zkomprimovaných dat je řešen pomocí postupného rozebrání numpy pole na jednotlivé hodnoty a následného spojení do řetězce nul a jedniček. Toto pole ještě není ve formě bitů, a proto můžou nastávat problémy při debugu, kdy se překladač snaží zkrátit numpy pole pro lepší zobrazení a vynechává celé sledy znaků. Toto jsem řešil přes nastavení parametru *np.set_printoptions(threshold=sys.maxsize)*, který zakázal zkracování zobrazení numpy polí. Samotný rozklad pole bylo třeba optimalizovat, neboť při větších obrázcích vzniká velké množství dat a zpracování znaků po jednom není uskutečnitelné.

Následně se daný řetězec 0 a 1 převede za pomocí knihovny bitarray na řetězec bitů (hodnot True a False v Pythonu). Ty se následně převedou pomocí té stejné knihovny do řetězce znaků složených z 8 bitů, který se již dá zapsat do souboru.

5.2.1 Seskupení dat v poli

Při pokusu dále zkomprimovat výsledný řetězec do formátu .7z, což probíhá za využití několika metod založených na entropickém kódování (např. DEFLATE), se ukázalo, že nevzniká prakticky žádné další zmenšení dat. Jeden z důvodů mohl být kvůli tomu, že daná síť zkomprimovala obrázek s velmi vysokými hodnotami entropie, a tak soubor nelze dále takto zmenšit.

Jako experiment jsem však ještě otestoval, zda přeskupení dat tak, aby se za sebou objevovala podobná data, nezpůsobí rozdíl při použití entropického kódování. Základní verze uskupení dat je taková, že jsou za sebou uloženy skupiny bloků, kde počet bloků ve skupině odpovídá velikosti batche. Pro každou skupiny jsou postupně uloženy data každé iterace pro celou skupinu bloků. Nejlepší uskupení dat by mělo nastat tehdy, když za sebou jdou data, která jsou si podobná. Data z jednotlivých iterací tedy musí být pospolu, neboť každá iterace je rozdílná a neměly by mít moc společného. Další podobnost by mohla nastat v datech, která reprezentují místa, jež jsou v původním obrázku blízko u sebe, neboť tyto místa mají větší pravděpodobnost se navzájem podobat.

Po experimentu se však ukázalo, že měnění uskupení dat nepřináší viditelné zmenšení dat. Byly vyzkoušeny různé způsoby urovnání dat kde brali nejprve všechny bloky za sebou v jednotlivých iteracích, jiný způsob zkoušel nejprve každý blok přes všechny iterace za sebou. Byly použity i postupně řádky celého komprimovaného obrazu (samostatně po iteracích) a postupně každá pozice ve všech blocích (také samostatně po iteracích). Všechny pokusy byly testovány na více komprimačních stupních formátu .7z včetně té nejvýkonnější.

5.3 Experimenty nad základními strukturami

V této sekci jsou uvedeny experimenty nad výchozími modely jednotlivých architektur použitých sítí.Patří mezi ně obyčejná reziduální síť, tvořená pouze konvolučními a dekonvolučními vrstvami a konvoluční síť s reziduálními prvky, která je dále rozdělena podle pozice, kde je předáván výstup rekurentního prvku v příští iteraci (tj. následující nebo aktuální vrstva). Zároveň je v této části popsán experiment mezi sítěmi se součtem výstupu rekurentní jednotky a konkatenací tohoto výstupu.

5.3.1 Reziduální konvoluční síť

Obyčejná reziduální síť je nejjednodušší model, nad kterým se budou provádět experimenty. Obsahuje výchozí složení autoenkodéru s třemi konvolučními vrstvami a třemi dekonvolučními vrstvami pro každou iteraci. Na každém konci iterace je vytvořeno reziduum, které je v další iteraci zkomprimováno, což se opakuje ve všech 16 iteracích.

Výsledky testování jsou ukázané na obrázku 5.1 a na grafu 5.3.2 jasně ukazují, že používání rekurentních prvků v sítí způsobuje mnohem větší kvalitu zkomprimovaných obrázků než použití čisté reziduální sítě. Obrázek je samotný velmi nekvalitní a vznikají na něm velmi viditelné artefakty. U použití menšího počtu iterací je navíc rekonstruovaný obrázek prakticky nepoužitelný (viz 5.4).

Průměrná hodnota MS-SSIM po testu na LIVE datasetu je 0.57445.

5.3.2 Reziduální konvoluční síť s rekurentními prvky

Reziduální konvoluční síť s rekurentními prvky má totožnou strukturu jako reziduální konvoluční sít 4.1, pouze navíc obsahuje propojení jednotlivých iterací za pomocí rekurentních

Obrázek 5.1: **Vlevo:** Obrázek čisté reziduální sítě bez rekurentních prvků. Kvalita obrazu je velmi nekvalitní a v obraze jdou vidět často se opakující artefakty. **Uprostřed:** Obrázek reziduální sítě s rekurentními prvky. **Vpravo:** Originální obrázek

prvků, které jsou uskutečněny jako součet nebo konkatenace výstupu vrstev iterace s určitou částí následující iterace. Jsou otestovány tři druhy výchozích sítí, ze kterých budou vycházet budoucí experimenty v této práci. Dvě sítě (4.2, 4.3) jsou zaměřeny na umístění propojení rekurentního jednotky v další iteraci, třetí síť je zaměřena na zjištění rozdílů kvality při použití konkatenace místo součtu při předání dat do následující iterace.

Rekurentní síť s výstupem za vrstvu

Cílem testování této sítě je zjistit, jak velký rozdíl způsobují rekurentní prvky v síti oproti čistému reziduálnímu modelu sítě bez rekurentních prvků. Model sítě obsahuje tři konvoluční vrstvy v enkodéru a tři dekonvoluční vrstvy spolu jednou konvoluční vrstvou v dekodéru. První vrstva v obou částech má 128 filtrů, druhá a třetí mají po 256 filtrech. V obou částech se dvakrát objevuje stride 2 a všechny konvoluční okna jsou velikosti 3x3, až na poslední dekonvoluční vrstvu, která má velikost okna 2x2. Rekurentní prvky se používají za každou konvoluční vrstvou v enkodéru a každou dekonvoluční vrstvou v dekodéru. Spojení těchto dat se provádí za pomocí obyčejného součtu. Experiment se provádí pomocí loss funkce L2 a na blocích o rozměrech 32x32 pixelů. Celkový počet iterací je 16, kde z každé je získán 1 bit na pixel původního obrazu.

Graf výsledků testování 5.3.2 ukazuje, že rekurentní prvky v síti výrazně zvyšují hodnotu MS-SSIM rekonstruovaných obrázků. Pro následující experimenty bude proto použita tato síť často jako výchozí. Obrázek 5.1 ukazuje vizuální srovnání mezi čistou reziduální sítí a touto sítí. Finální hodnota MS-SSIM po otestování na LIVE datasetu je 0.85112.

Rekurentní síť s výstupem před vrstvu

Téměř totožná síť se sítí 4.2 až na místo v iteraci, kde vstupují data z rekurentních jednotek předchozí iterace. Rozdíl mezi sítěmi jde vidět na obrázcích 4.2 a 4.3.

Výsledky tohoto experimentu 5.3.2 jasně ukazují, že zasílání výstupu rekurentních jednotek na vstup stejné vrstvy v následující iteraci má mnohem lepší výsledky než síť 4.2.

Obrázek 5.2: **Vlevo:** Rekonstruovaný obraz pomocí rekurentní sítě se zasíláním za vrstvu **Vpravo:** Rekonstruovaný obraz pomocí rekurentní sítě se zasíláním před vrstvu

Protože byl experiment na této síti proveden až později během experimentování, když už mnoho dalších experimentů bylo provedeno na výchozím modelu sítě 4.2, byla tato síť využita pouze až v konečných experimentech pro získání nejkvalitnějších kombinací sítí.

Obrázek 5.3 ukazuje rozdíl mezi touto a předchozí sítí. Finální výsledek testování je 0.91365.

Konkatenace místo součtu

Při použití konkatenace místo součtu u rekurentních jednotek je potřeba počítat se zvýšenou paměťovou náročností. Pro trénování této sítě, bylo nutné pro zachovaní běžných hodnot u parametrů, které více ovlivňují paměťové nároky, použít grafickou kartu s větší dostupnou pamětí než 4GB. Při použití pouze 4GB grafické karty bylo potřeba snížit počet iterací na 8. Celkově tato architektura měla ze všech ostatních modelů sítí největší paměťové nároky.

Jako výchozí model byla použita síť 4.2, ve které se pouze zaměnila operace součtu za operaci konkatenace. Konkatenace probíhala na třetí ose, která představuje filtry.

Výsledek sítě 5.3.2 ukazuje, že rozdíl mezi součtem a konkatenací není příliš velký, možná je i lehce horší, to se ale může připsat i fluktuaci výsledků během trénování. Navíc i přesto, že se o něco rychleji trénuje, má ve trénování podstatně větší rozptyl dat, co není vhodné pro stabilní výsledky sítě. I vizuálně není v obrázcích vidět podstatný rozdíl. Finální výsledek sítě po testování na LIVE datasetu je 0.83237

Obrázek 5.3: **Vlevo:** Rekonstruovaný obraz pomocí rekurentní sítě se sčítáním výstupu rekurentních jednotek **Vpravo:** Rekonstruovaný obraz pomocí rekurentní sítě s konkatenací výstupu rekurentních jednotek.

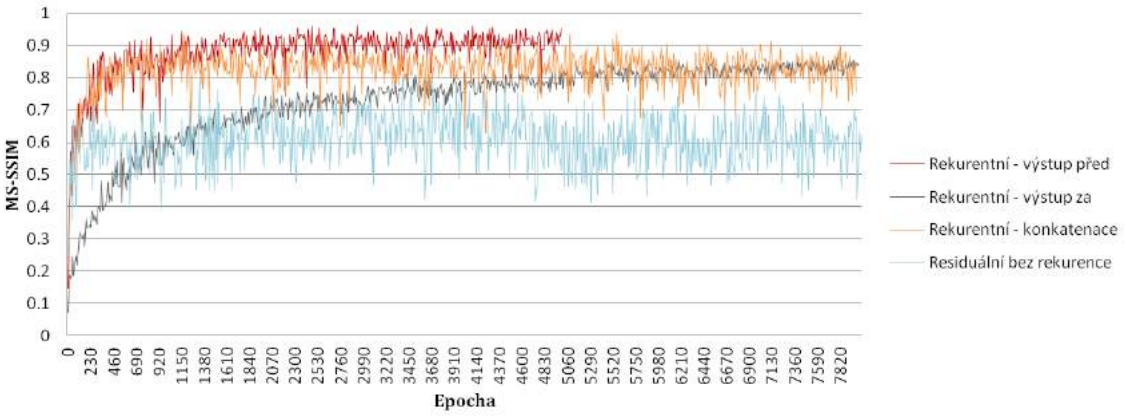
Výsledné porovnání architektur

Tabulka 5.3.2 představuje výsledky předchozích sítí a jasně ukazuje, že nejlepších výsledků dosahuje použití sítě s rekurentními prvky. U čistě reziduální sítě se navíc jasně ukazuje, že při nižším množství iterací se projevuje relativně větší zhoršení obrazu než u sítí s rekurentními prvky, což jde vidět na obrázku. Na grafu 5.3.2 je vidět, že první iterace čistě reziduální sítě komprimují pouze nejhrubější detaily obrázku a obrázek se začne dostávat správné podoby až ke konci přidělených iterací. Na rozdíl od toho, rekurentní sítě už jen při použití nejnižších iterací dosáhly stejných výsledků jako čistě reziduální síť a všechny následující iterace pouze dále vyhlazují detaily.

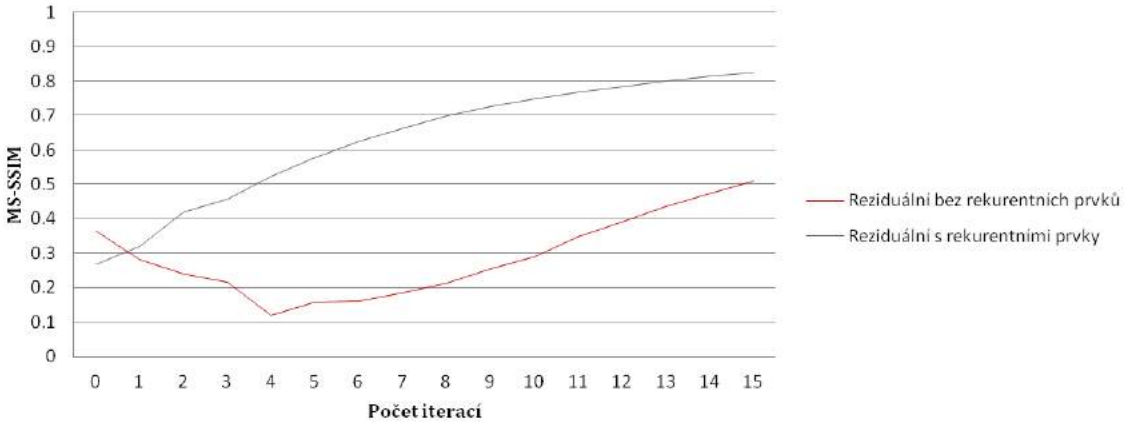
Na grafu 5.3.2 jde vidět průběh trénování jednotlivých architektur. U čistě reziduální sítě jde vidět, že kromě nízkých výsledků, má i velkou fluktuaci v závislosti na různých obrázcích, které jsou ji předloženy. Tato fluktuace se v průběhu trénování viditelně nezlepšovala. Síť s konkatenací dosahovala podobných výsledků jako s síť se sčítáním rekurentních výstupů, měla však také méně stabilnější výsledky v závislosti na různých obrázcích. Rekurentní síť s výstupem rekurentních jednotek do stejné vrstvy dosahovala nejlepších výsledků a i v jejich nejhorších případech přesahovalo kvalitu ostatních architektur. V konečných experimentech, kdy se pokusím získat síť s nejlepšími výsledky se bude používat tato výchozí architektura.

	MS-SSIM - 8it	MS-SSIM - 16it
Residuální síť bez rekurence	0.21794	0.57445
Rekurentní síť s výstupem za vrstvu	0.70576	0.85112
Residuální síť s výstupem před vrstvu	0.78626	**0.91365**
Residuální síť s konkatenací	0.65464	0.83237

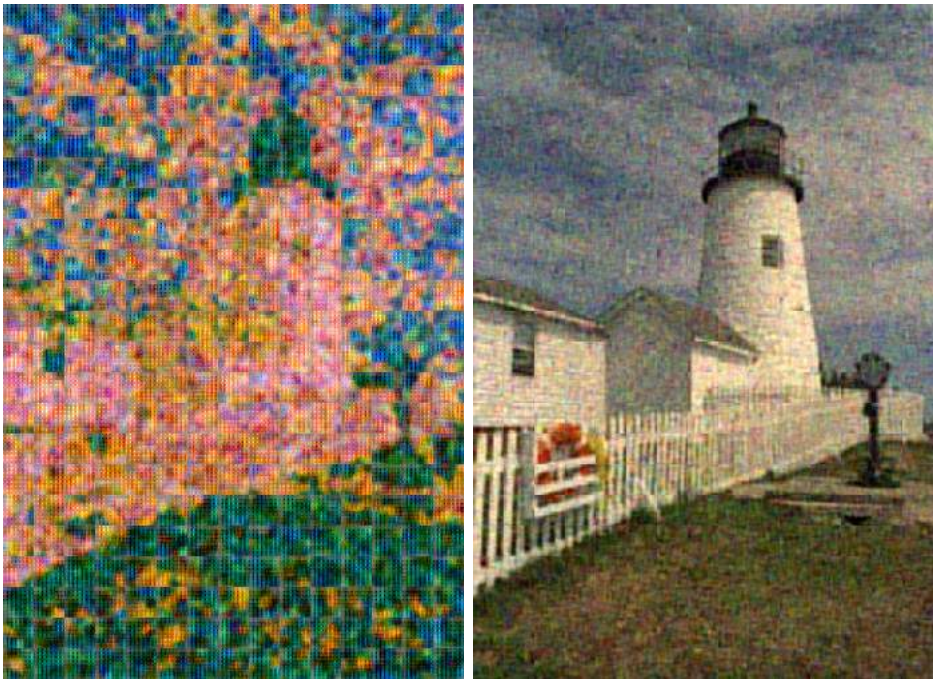
Tabulka 5.1: Výsledky MS-SSIM čtyř základních architektur testovaných na LIVE datasetu. Výsledky jsou pro 8 a 16 iterací, kdy každá iterace představuje 4bity na 8x8 (2x stride 2) pole, což je 0.25 bitu na pixel pro každou iteraci



Graf 5.1: Graf s porovnáním trénování základních architektur sítí **Červená:** Rekonstruovaný obraz pomocí rekurentní sítě se sčítáním výstupu rekurentních jednotek , má jednoznačně nejlepší výsledek, i při natrénování na měnším množství epoch. **Oranžová:** Rekonstruovaný obraz pomocí rekurentní sítě s konkatenací výstupu rekurentních jednotek, namísto sčítání. Způsobí rychlé naučení sítě oproti sčítání, ale v závěru má horší výsledky. **Šedá:** Rekonstruovaný obraz pomocí rekurentní sítě se zasíláním za vrstvu. Je zde vidět, že oproti ostatním architekturám má pomalejší učení, ale i přesto dosahuje druhého nejlepšího výsledku. **Světle modrá:** Reziduální síť bez rekurentních prvků. Má podstatně horší výsledky než ostatní architektury, má však téměř okamžitý narůst na svou maximální hodnotu.



Graf 5.2: Graf ukazující hodnotu MS-SSIM v závislosti na počtu použitých iterací při rekonstrukci obrázku. **Červená:** Čistě reziduální síť, větší hodnoty u nižších iterací jsou způsobeny počítáním MS-SSIM na prakticky jedno až dvou barevných obrázcích, což je jak tyto obrázky v těchto iteracích vypadají. **Šedá:** Reziduální síť s rekurentními prvky 4.2



Obrázek 5.4: **Vlevo:** Rekonstruovaný obraz pomocí čistě reziduální sítě s použitím pouze 8 iterací (celkově použity 2 bity na pixel) MS-SSIM - 0.18512 **Vpravo:** Rekonstruovaný obraz pomocí sítě s rekurentními prvky za použití pouze 8 iterací.(celkově použito 2 bity na pixel) MS-SSIM - 0.66162

5.4 Experimenty nad parametry

Tato sekce se bude zabývat testováním nejdůležitějších parametrů, které byly blíže přiblíženy v kapitole 4.2. Jako výchozí síť, ke které se budou testované sítě přirovnávat, bude síť 4.2, pokud nebude řečeno jinak. Na konci sekce jsou ukázány grafy 5.4.7, 5.4.7 a 5.4.7, které obsahují výsledky jednotlivých experimentů. U každého experimentu je navíc ukázána ukázka rekonstruovaných obrázků pro vizuální představu a model, na kterém je vidět jak vypadá struktura sítě (pouze u sítí, u kterých je model znatelně jiný).

5.4.1 Velikost vstupu

První testovaný parametr je velikost vstupu, na kterém se bude síť trénovat. Místo výchozího bloku o rozměrech 32x32 se využije blok o velikosti 128x128 pixelů a blok o velikosti 256x256 pixelů. Takovéto navýšení ve velikosti vstupu by při zachování velikosti trénovací dávky způsobilo minimálně 16krát větší paměťové nároky, což je pro neserverové stanice nereálné. Pro trénování se tedy snížila velikost dávky, což by ale kvůli tomu, že epochy jsou vázané na celé obrazy místo na dávky, nemělo mít žádný vliv na případnou fluktuaci hodnot MS-SSIM v grafu.

Na obrázku 5.5 jde pozorovat, že sítě natrénované na blocích o velikosti 128x128 a 32x32 mají velmi podobné výsledky, síť natrénovaná na blocích o rozměrech 256x256 už vykazuje výraznější šum v obraze. Na grafu s výsledky 5.4.7 je ukázáno, že výsledky všech tří sítí mají velmi podobný průběh trénování. Větší rozměry bloku mají ale větší odchylku. Test na LIVE datasetu ukázal, že větší bloky mají lehce lepší výsledky i přes viditelnější šum, což může být způsobeno menším množstvím přechodů mezi jednotlivými bloky. Na druhou stranu, větší bloky způsobují horší zarovnání bloků na obraz, což způsobuje výrazné osekání částí obrazu, jak jde poznat na 5.5, kde obraz ze sítě s 256x256 bloky je ustřižen v horizontální rovině.

5.4.2 Počet vrstev v iteraci

Dekodér a enkodér výchozího modelu sítě (4.2) byl rozšířen o dvě konvoluční vrstvy v případě enkodéru a dvě dekonvoluční vrstvy u dekodéru na celkový počet 5 vrstev v každé části. Všechny přidané vrstvy mají 256 filtrů a stride 1. Síť má kvůli zvětšení počtu vrstev větší paměťové nároky a musí se trénovat po menších dávkách. Zároveň je rychlost trénování pomalejší. Na obrázku 5.6 je znázorněna podrobnější struktura sítě.

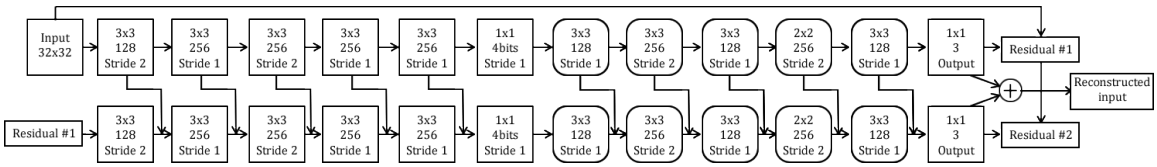
Natrénovaná síť má výrazně horší výsledky (MS-SSIM 0.67201) než výchozí síť. Na obrazu 5.8 vznikají viditelné opakující se artefakty a obraz má lehce rozdílnou barvu.

5.4.3 Množství konvolučních filtrů

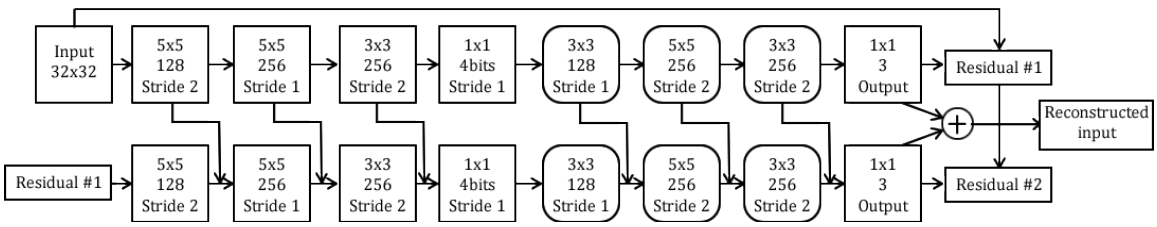
V této síti byl u všech konvolučních vrstev enkodéru a dekonvolučních vrstev dekodéru zvýšen počet filtrů na dvojnásobek, což v případě druhých a třetích vrstev obou částí znamená 512 filtrů a u prvních vrstev 256 filtrů.

Výsledek sítě 0.82721 5.4.7 je horší než výchozí síť, což znamená, že za určitým množstvím filtrů už další filtry nezlepšují výsledek sítě. Zhoršení však není dostatečně velké, aby se dalo usuzovat, že toto zhoršení bylo způsobeno množstvím filtrů a ne náhodnou fluktuací v trénování sítě. Na obrázku 5.9 je zobrazen obrázek zkomprimovaný touto sítí.

Obrázek 5.5: **Vlevo:** Rekonstruovaný obraz se sítí natrénovanou na bloky o velikosti 128x128 pixelů **Uprostřed:** Rekonstruovaný obraz se sítí natrénovanou na bloky o velikosti 256x256 pixelů **Vpravo:** Výchozí rekonstruovaný obraz se sítí natrénovanou na bloky o velikosti 32x32 pixelů



Obrázek 5.6: Model výchozí sítě rozšířený o dvě vrstvy v enkodéru a dekodéru. V enkodéru jsou přidány dvě konvoluční vrstvy na konec celé části, zatímco v dekodéru jsou po jedné vrstvě přidáno za původní druhou a třetí vrstvu



Obrázek 5.7: Model s upravenými velikostmi konvolučních a dekonvolučních oken. Všechny aktuální 5x5 konvoluční okna byly rozšířené z 3x3. Poslední dekonvoluční vrstva, která je aktuálně 3x3, byla rozšířena z rozměrů 2x2

Obrázek 5.8: **Vlevo:** Rekonstruovaný obraz se sítí natrénovanou se dvěmi vrstvami navíc v dekodéru a enkodéru **Vpravo:** Výchozí rekonstruovaný obraz se reziduální sítí s rekurentními prvky 4.2

5.4.4 Velikost konvolučního okna

Původní konvoluční okna 3x3 u první a druhé vrstvy enkodéru byly zvětšeny na rozměry 5x5. U třetí vrstvy se konvoluční okno zanechalo stejné kvůli menšímu vstupnímu obrazu. Ze stejného principu se zanechalo stejné i konvoluční okno i první dekonvoluční vrstvy dekodéru. Druhá dekonvoluční vrstva měla okno změněno také na 5x5, zatímco poslední dekonvoluční vrstva měla z původního okna 2x2 rozšířeno na 3x3.

Síť má velmi podobné výsledky (MS-SSIM 0.84648) s výchozí síti jak v testu na LIVE datasetu, tak i v podobě zkomprimovaného obrázku, který je zobrazen na obrázku 5.9.

5.4.5 Loss funkce

Na této sítí bylo vyzkoušeno trénování podle loss funkce L1, a celý experiment je zaměřený nejenom na výslednou kvalitu sítě, ale i na rychlost natrénování sítě. Loss funkce L1 na rozdíl od funkce L2, která počítá kvadratický rozdíl dvou hodnot pixelů, počítá pouze absolutní rozdíl. To způsobuje, že vetší hodnoty rozdílu mezi výslednou a originální hodnotou pixelu brány, nebudou sítí brány v takovou důležitost jako u funkce L2.

Trénování pomocí loss funkce L1 způsobuje podle grafu 5.4.7 rychlejší trénování sítě, v závěru však dosahuje horších výsledků MS-SSIM - 0.82528 (5.4.7).

Jako součást tohoto experimentu bylo vyzkoušet trénovat síť i podle samotné metriky MS-SSIM. Pokus o natrénování této sítě byl neúspěšný, neboť síť se vždy dostávala do extrémních hodnot vah.

5.4.6 Množství detailů

Testování probíhá na dvou různých sítích. Jedna síť je trénovaná na obrazech, které jsou 8x rozšířeny do výšky i šířky. To způsobuje, že se trénovací bloky skládají pouze z 4x4 pixelů původního obrazu. Většina těchto bloků je tedy složena z pixelů pouze několika málo barev,

Obrázek 5.9: **Vlevo:** Rekonstruovaný obraz se sítí natrénovanou s dvojnásobným množstvím filtrů **Uprostřed:** Rekonstruovaný obraz se sítí natrénovanou s většími konvolučními okny (5x5) **Vpravo:** Rekonstruovaný obraz se sítí natrénovanou pomocí loss funkce L1

které mezi sebou navíc jen velmi pozvolna přecházejí, a jen velmi málo kdy se objevují ostré přechody. Takto trénovaná síť je tedy natrénovaná na jednolité plochy. Kvůli testování na větších obrázcích proběhlo celkově menší množství epoch. Zároveň díky specializovanosti datasetu má graf 5.4.7 rozdílné výsledky než finální zhodnocení přes LIVE dataset.

Druhá síť se opačně trénuje na zmenšených obrázcích, takže se měla naučit lépe ostré přechody mezi barvami a hůře jednolité plochy. Podobně i u této sítě má graf 5.4.7 a výsledné MS-SSIM 5.4.7 hodnoty značné rozdílné hodnoty.

Jako výchozí síť byla použita rekurentní síť 4.3, která dosahuje o něco lepších výsledků než síť 4.2. Je to kvůli tomu, že tyto sítě budou použity i v budoucích experimentech, kde je už větší důraz na kvalitu výsledného obrazu.

Výsledky experimentu ukazují, že síť natrénovaná na zvětšených obrázcích dosahuje lepších hodnot MS-SSIM (0.92683) i přes jasně viditelné zhoršení kvality obrazu v místech rychlých přechodů mezi barvami (5.10), což je ale vyváženo lepší kvalitou obrazu v jednolitých oblastech. Síť trénovaná na zmenšených obrazech i přes vylepšenou kvalitu u rychlých přechodů dosáhla horší celkové kvality (MS-SSIM 0.90429) (5.4.7).

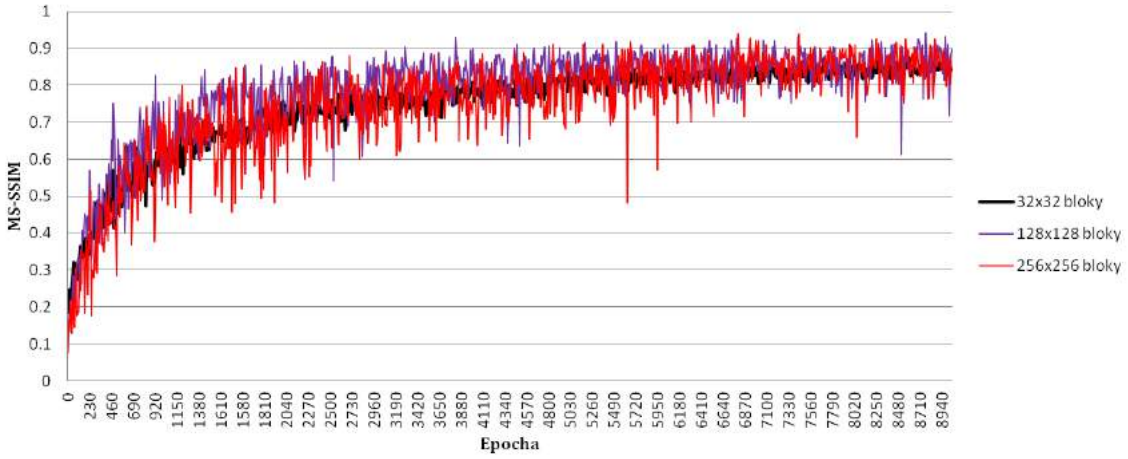
5.4.7 Ostatní

Mezi zbylé experimenty, které ještě stojí za zmínku patří testování zvýšení parametrů stride, což způsobí snížení plochy, se kterou vrstvy pracují. Pro zachování stejných efektivních dat, se kterými síť pracuje, byl počet filtrů u ovlivněných sítí zdvojnásoben. Další otestovaná síť měla rozložena výstupní data do dvojnásobného množství iterací. Překvapivě obě sítě dosahují podstatně horších výsledků (5.4.7) než výchozí síť, kde síť s upraveným parametrem stride dosahuje MS-SSIM 0.76666 a síť s dvojnásobkem iterací 0.79710. U sítě s dvojnásobnou stride toto není zas tak překvapující, neboť omezujeme výhodu konvolučních sítí tím, že zmenšujeme pracovní prostor. U sítě s dvojnásobným množstvím iterací by to mohlo být způsobeno nedostatečným natrénováním sítě.

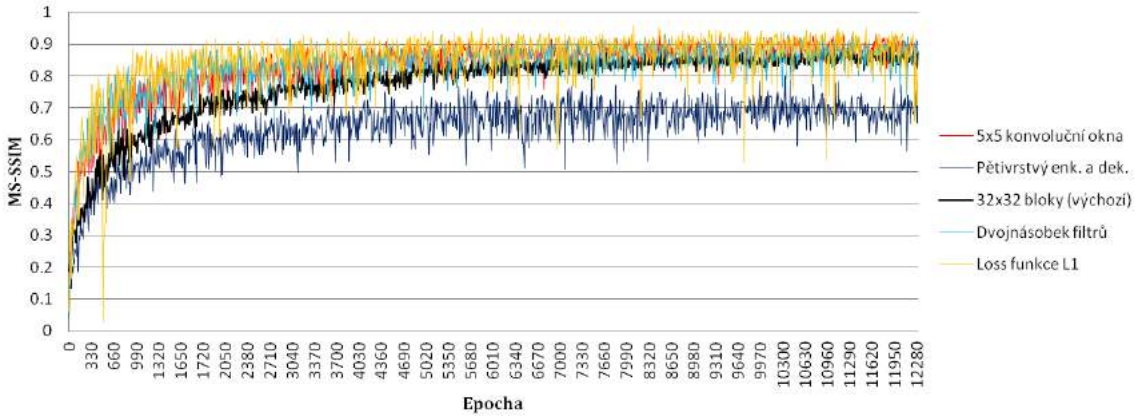
Obrázek 5.10: **Vlevo:** Rekonstruovaný obraz se sítí natrénovanou na 8krát zmenšených obrázcích **Uprostřed:** Rekonstruovaný obraz se sítí natrénovanou na 8krát zvětšených obrázcích **Vpravo:** Rekonstruovaný obraz s výchozí reziduální sítí s rekurentními prvky 4.3 (výstup na vstup)

Typ testovaného parametru	MS-SSIM - 16 it.
Bloky 32x32 (výchozí)	0.85112
Bloky 128x128	0.85567
Bloky 256x256	0.86043
5 vrstev	0.67201
5x5 konvoluční okna	0.84648
Dvojnásobek filtrů	0.82721
Loss funkce L1	0.82528
Dvojnásobná stride	0.76666
Dvojnásobná iterace	0.79710
Rekurentní síť 4.3 (výchozí)	0.91365
Zvětšené vstupy	**0.92683**
Zmenšené vstupy	0.90429

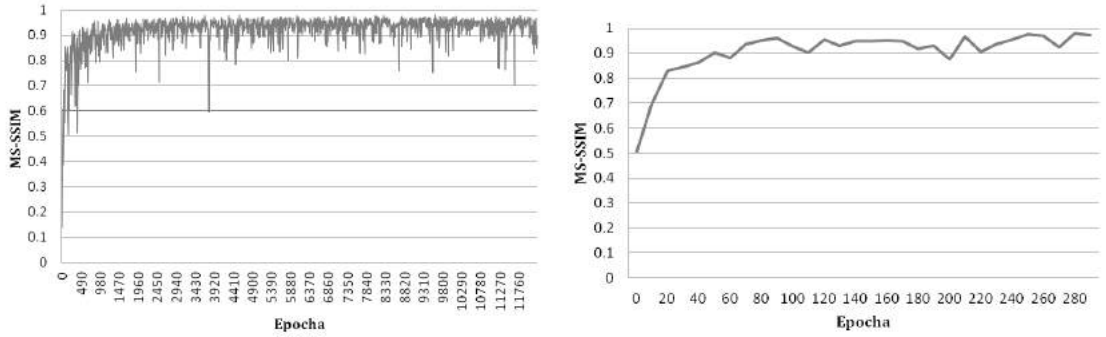
Tabulka 5.2: Výsledné hodnoty MS-SSIM pro nejdůležitější testované parametry. Pro prvních devět experimentů byla jako výchozí síť použita síť 4.2, zatímco na poslední tři experimenty byla použita síť 4.3



Graf 5.3: Průběh hodnot MS-SSIM při trénování sítí podle velikosti jejich vstupních bloků **Černá:** Síť trénovaná na blocích 32x32 pixelů (výchozí) **Červená:** Síť trénovaná na blocích 256x256 pixelů **Fialová:** Síť trénovaná na blocích 128x128 pixelů



Graf 5.4: Průběh hodnot MS-SSIM při trénování sítí podle různých parametrů s výchozí sítí 4.2 **Černá:** Výchozí síť **Žlutá:** Síť trénovaná s loss funkcí L1 namísto funkce L2 **Modrá:** Síť trénovaná se dvěma vrstvami navíc v dekodéru a enkodéru **Tyrkysová:** Síť trénovaná s dvojnásobným množstvím filtrů **Červená:** Síť trénovaná s většími konvolučními okny (5x5)



Graf 5.5: **Vlevo:** Trénování sítě 4.3 na 8krát zmenšených obrázcích, za účelem naučení se rychlých přechodů mezi barvami **Vpravo:** Trénování sítě 4.3 na 8krát (na výšku i šířku, takže 64krát větší) zvětšených obrázcích, za účelem naučení se pomalých popřípadě žádných přechodů mezi barvami (epochy ukazují počet obrázků na kterých se trénovalo, u obrázků se 64krát více daty jich není tolik potřeba)

5.5 Pokročilé sítě

Experimenty v této sekci jsou zaměřeny na změny určitých strukturních prvků modelů sítí a nebo se zabývají zjištěnými informacemi z kapitol 5.3 a 5.4. Hlavní zaměření je na možnost specializace datasetu pomocí různých metod od úpravy struktury sítě až po rozdělení vstupu samotného. Zároveň se kapitola zabývá jak efektivní je přímo použití specializovaného datasetu pro komprimaci obrázků. Pro testování takovýchto sítí byl vybrán specializovaný dataset map.

5.5.1 Specializované sítě - mapy

Tato sekce obsahuje sítě, které se trénují nad datesetem popsaným v 4.5. Cílem je zjistit jak moc dokáže specializovaný dataset zlepšit hodnoty MS-SSIM. Zároveň zde jsou experimenty nad sítěmi, které byly upraveny podle poznatků zjištěných v předchozích experimentech. Jsou zde ukázany i sítě, které se budou používat v budoucích experimentech.

Základní experiment na výchozích architekturách

První testování je uděláno na sítích 4.2 a 4.3 pro zjištění výchozích hodnot MS-SSIM, kolem kterých se tyto sítě pohybují, a také jak velký rozdíl specializovaný dataset způsobí.

Výsledné obrázky obou sítí jsou už v takové kvalitě, že pouhým okem začíná být problém rozeznat rozdíly (5.11). Výsledky 5.5.1 (0.931320 resp. 0.972070) ukazují, že specializovaný dataset významně zvyšuje kvalitu rekonstruovaného obrazu. Z toho budu vycházet při pozdějších experimentech, kdy se pokusím replikovat tento jev uměle na všeobecném datasetu.

Už jen z výsledků této sítě je vidět, že neuronové sítě mají obrovský potenciál při kompresi specifických dat.

Zvětšené obrazy

Pro pozdější experimenty, které budou využívat více sítí najednou je natrénována ještě síť na zvětšených obrazech. Na rozdíl od nespecializované sítě, kde se kvalita sítě výrazně zlepšila, síť trénovaná na mapách dosáhla výrazného zhoršení (MS-SSIM 0.93206) při trénování na

Obrázek 5.11: **Vlevo:** Rekonstruovaný obrázek ze sítě 4.2 natrénované na specializovaném datasetu **Uprostřed:** Rekonstruovaný obrázek ze sítě 4.3 natrénované na specializovaném datasetu **Vpravo:** Originální obrázek pro porovnání

Typ testovaného parametru	**MS-SSIM - 16 it.**
Rekurentní síť 4.2 bez specializace	0.85567
Rekurentní síť 4.2 se specializací	0.931320
Rekurentní síť 4.3 bez specializace	0.91365
Rekurentní síť 4.3 se specializací	**0.972070**
Rekurentní síť 4.3 se specializací a zvětšenými obrazy	0.93206

Tabulka 5.3: Výsledné hodnoty MS-SSIM pro sítě trénované na specializovaném datasetu + sítě pro porovnání.

zvětšených obrazech. Pravděpodobně je to zapříčiněno tím, že na vybraných mapách se objevuje méně jednolitých ploch než u běžných obrázků.

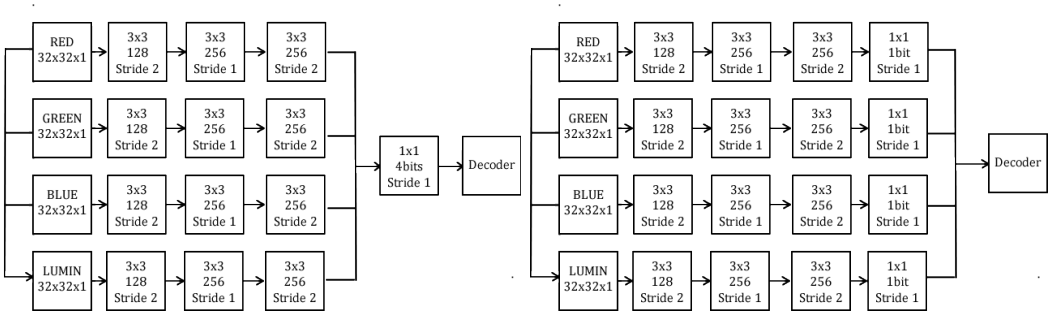
5.5.2 Větvení v síti

V této části jsou vyzkoušeny dvě sítě, které se snaží rozdělit vstupní data ve svém modelu tak, aby vznikly v sítí části, které jsou specializované na určité data, což by teoreticky mohlo zvýšit kvalitu výsledného obrazu. Změna struktury sítě je zaměřena na enkodér a binarizační vrstvu, kde se oba, popřípadě pouze enkodér rozdělí do čtyř identických větví, které budou zpracovávat každý různou barvu z původního vstupu (+ jas, kvůli zachování čtyř bitů na jednu iteraci). Jas je vypočten pomocí rovnice 5.1.

$$Lum = 0.299 * R + 0.587 * G + 0.114 * B \tag{5.1}$$

Typ testovaného parametru	MS-SSIM - 16 it.
Referenční síť 4.2 (výchozí)	0.85112
Pouze enkodér	0.81472
Enkodér + binarizační vrstva	0.82935

Tabulka 5.4: Výsledné hodnoty MS-SSIM pro sítě s větvenou strukturou enkodéru a binarizační vrstvy.



Obrázek 5.12: **Vlevo:** Rozvětvený enkodér pro separátní zpracování barevných kanálů vstupu (a jas). V obrázku pro jednoduchost nejsou zobrazeny výstupy do následující iterace (rekurence) **Vpravo:** Rozvětvený enkodér rozšířený o rozvětvenou binarizační vrstvu

Rozvětvený enkodér

První z testů používá rozvětvení pouze na část enkodéru a výsledky ze všech čtyř větví pomocí operace součtu zkombinuje dohromady před zasláním dat do binarizační vrstvy. Na obrázku 5.12 vlevo je znázorněna struktura enkodéru. Rekurentní prvky stále existují a jsou použity na každé větvi zvlášť.

Výsledky sítě (tabulka 5.5.2) dosahují o něco horších hodnot (0.81472) než má výchozí síť (0.85112), takže po započítání, že síť má navíc vyšší časové i paměťové nároky na trénování, to vypadá, že tohle není správný směr pro vylepšení kvality komprese. Pro úplnost je však ještě proveden test i s rozvětvenou binarizační vrstvou.

Rozvětvený enkodér a binarizační vrstva

Pokračování předchozího experimentu, kdy je navíc přidána rozvětvenému enkodéru i rozvětvená binarizační vrstva. Struktura těchto dvou částí jde vidět na obrázku 5.12. Binarizační vrstva má svůj původní výstup o velikosti čtyř bitů (na každou jednotku v 8x8 poli) rozdělen na čtyři výstupy o jednom bitu. Výstupy jednotlivých vrstvě jsou následné spojeny přes svou poslední dimenzi (vznikne struktura 8x8x4 bitů jako v původní binarizační vrstvě) a tento výstup se pošle do dekodéru. Rekurentní prvky fungují stejně jako v předchozím experimentu.

Výsledky sítě jsou oproti sítí s pouze rozvětveným enkodérem lepší (MS-SSIM 0.82935), stále však nedosahují výsledků referenční sítě. Tento experiment tedy ukazuje, že pro využití specializace v obecném datasetu, rozvětvení modelů a rozdělení vstupu není vhodná cesta.

5.5.3 Uměle specializované sítě - multibranch

Experimenty v této sekci se snaží zakomponovat specializaci datasetu, ale na rozdíl od rozdělení vstupu až v modelu, které se neukázalo jako úspěšné, se zde využívá trénování více sítí, kde každá síť je specializovaná na určitou část z celého datasetu. Dataset se dá takto rozdělit mnoha způsoby, ale pro následující experimenty byly použity poznatky z předcházejících experimentů. První experiment se zaměří na množství detailů v obraze, zatímco druhý bude využívat speciální metriku ostrosti, podle které bude rozebírat vstupy do jednotlivých sítí.

Kombinace sítí zaměřených na množství detailů v obraze

Následující experimenty vycházejí z experimentů 5.4.6, kde je použito zmenšování a zvětšování vstupních obrázků pro částečnou specializaci sítě podle rychlosti přechodů mezi barvami. Jsou využity sítě, které byly natrénovány ve zmíněných experimentech, společně s referenční rekurentní sítí 4.3, která není specializovaná, ale bude sloužit jako vyplnění intervalu všech vstupů, na které nejsou specializované zbývající sítě.

Zhodnocení, který výstup (z množiny aktuálně testovaných sítí) se má využít pro kompresi je určen za pomocí výpočtu hodnoty MS-SSIM pro každý vstupní blok. Pro komprimaci obrazu se tedy musí vstupní obraz zpracovat všemi sítěmi, čímž získáme množinu rekonstruovaných obrazů, kde porovnáváme každý blok přes všechny tyto obrazy pro získání nejlepší hodnoty MS-SSIM. velikost bloků je teoreticky nejlepší zvolit co možná nejmenší, kvůli častější aplikaci rozdělování mezi sítěmi, to však přináší teoreticky větší metadata pro zakódování použitých větví. V následujících experimentech byly vždy použity bloky o velikosti 32x32.

Jsou provedeny následující experimenty a v závěru jsou shrnuty a zhodnoceny jejich výsledky:

2-větvová Low/All Detail síť: Experiment kombinuje síť natrénovanou na zvětšených obrázcích (Low - malé detaily, pomalé přechody) a obyčejnou nespecializovanou síť (All), která by měla pokrýt všechny ostatní vstupy. Výsledek MS-SSIM je 0.92675, což je velmi podobné s Low sítí.

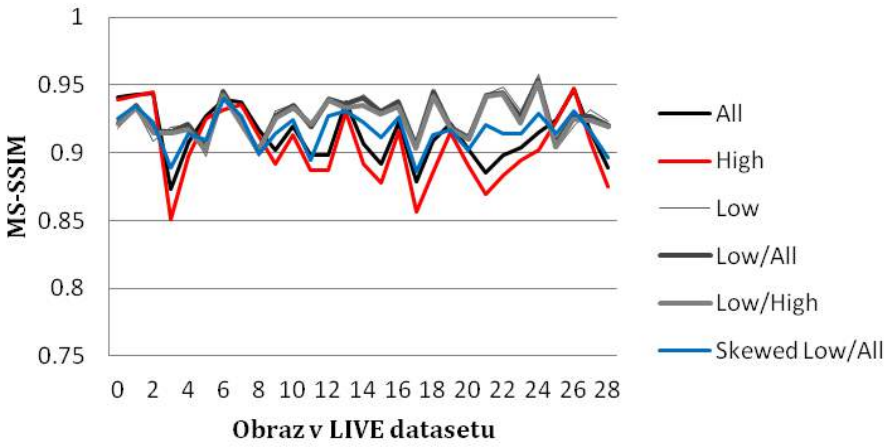
2-větvová Low/All Detail specializovaná síť - mapy: Stejná jako předchozí experiment, používá však navíc už předem specializovaný dataset. Cílem je zjisti, jestli bude mít tato multibranch struktura větší nebo menší efekt v případě přirozené specializace datasetu. Výsledek sítě je 0.96461 MS-SSIM.

2-větvová Low/All Skewed Detail síť: Při porovnávaní MS-SSIM u jednotlivých bloků je v případě, že jedna síť má mnohem lepší výsledky i samostatně než druhá síť, častý případ, kdy se využije bloku ze sítě, která není na daný blok přímo specializovaná. Tento experiment nachýlí měření MS-SSIM o 0.01 ve prospěch All (referenční) sítě za cílem posílat více bloků přes tuto síť. Finální výsledek je však 0.91599 MS-SSIM, což je zhoršení.

2-větvová Low/High Detail síť: Použití dvou sítí, kde jedna je trénovaná čistě na zvětšených obrázcích s pomalými přechody a druhá je trénovaná na zmenšených obrázcích s velmi rychlými a častými přechody mezi barvami. Výsledek 0.92466 MS-SSIM je opět velmi podobný výsledkům Low sítě.

Typ testovaného parametru	MS-SSIM - 16 it.
Samotná rekurentní síť All 4.3 (výchozí)	0.91365
Samotná rekurentní síť Low	0.92683
Samotná rekurentní síť High	0.90429
2-větvová Low/All Detail	0.92675
2-větvová Low/All Skewed Detail	0.91599
2-větvová Low/High Detail	0.92466
Samotná rekurentní síť All - mapy	0.97207
Samotná rekurentní síť Low - mapy	0.93206
2-větvová Low/All Detail - mapy	0.96461

Tabulka 5.5: Výsledné hodnoty MS-SSIM pro sítě trénované pomocí multibranch konceptu zaměřeném na detaily v obraze + referenční sítě pro porovnání.



Graf 5.6: Hodnoty MS-SSIM u jednotlivých obrázků v LIVE databázi u sítí z multibranch experimentu (5.5.3). Na grafu je znázorněné, že hodnoty MS-SSIM pro sítě obsahující Low síť (trénovaná na zvětšených obrázcích) mají téměř totožné výsledky (kromě skewed sítě, která testovala přímo tuto skutečnost).

Zhodnocení výsledků sítí

Z vyzkoušených testů jde vidět, že výsledky sítí (tabulka 5.5.3) se vždy přizpůsobovaly výsledkům té nejlepší sítě ve skupině (Low síť), což je jasné patrné na grafu 5.5.3. U map nastala stejná situace, zde se však výsledky srovnali s All sítí. I přesto, že na grafu je to nemožné zpozorovat, při pohledu na přesné čísla výsledků jde vidět, že kombinační sítě mají často drobný stabilizační efekt, kdy hodnoty s vysokou odchylkou od průměrné hodnoty byly lehce nakloněny k této průměrné hodnotě.

I přesto, že tyto sítě mají pomocí měření MS-SSIM téměř totožné výsledky s Low sítí, na obrázcích 5.15 jde vidět občasné artefakty, které vznikají mezi bloky, které byly použity z různých sítí. Ve zmíněných blocích je na první pohled patrná rozdílná barva.

Cekově se tedy dá říct, že tyto sítě mají potenciál, je však zapotřebí použití sítí s lépe rozděleným trénováním, což může být prakticky těžko uskutečnitelné. Zároveň je zde potenciál stabilizačního efektu.

Obrázek 5.13: **Vlevo:** Rekonstruovaný obrázek pomocí multibrnch sítě Low/All. Na žluté čepici vlevo jde vidět přechod dvou rozdílných odstínu žluté, způsobených rozdílnou kompresí barev u Low a All sítě.**Vpravo:** Originální obrázek pro porovnání

Kombinace sítí trénovaných podle dané metriky

Tato část je zaměřena na kombinaci sítí, které byly speciálně natrénované na vstupech, které byly rozděleny podle dané metriky, a podle stejné metriky se jim bude předkládat vstup. To je rozdíl od sítí v minulé části, které rozhodovaly jaká jejich vnitřní síť bude použita podle toho, která dokázala daný vstupní blok zakódovat nejlépe. Jedna z výhod tohoto přístupu od minulého je nepotřebnost zpracovávat blok přes celou síť, pro zjištění, kterou sítí se bude komprimovat, ale je možné síť vybrat apriorně pomocí použití metriky na daný vstupní blok. Další výhodou je možnost absolutního rozdělení celého universa obrázků do zvolitelných intervalů pomocí dané metriky. To navíc způsobuje, že jednotlivé vnitřní sítě se nikdy nemusí trénovat na špatných datech, což bylo u předchozích sítí prakticky nesplnitelné.

Metriky pro trénování vnitřních sítí, kterou jsem zvolil je metrika ostrosti (sharpness), která se vypočítává za pomocí horizontálního a vertikálního gradientu ve vstupním bloku (který je převeden pro tuto metriku na černobílý obraz). Následně se vypočítá vzdálenost mezi pozičně souhlasnými elementy obou gradientů a zjistí se vážený průměr všech takto získaných hodnot, což je výsledek měřené metriky ostrosti.

Pro multibranch sítě se musí ještě natrénovat zvlášť jednotlivé vnitřní sítě. Rozhodl jsem se použít rozdělení na čtyři intervaly vstupních bloků: <0, 3), <3, 10), <10, 20) a (20, inf), kde interval <0, 3) představuje téměř jednolité plochy s téměř žádnými přechody barev, a ostatní jsou postupně více a více zaměřeny na větší množství rychlých přechodů barev. Následně je vyzkoušena dvou-, tří- a čtyř- větvová multibranch síť, kde se postupně přidává jednotlivé vnitřní sítě, aby se dalo zjistit progresivní zlepšení.

Samostatné sharpness sítě

Jako výchozí síť pro tyto vnitřní sítě je použita rekurentní síť 4.3, která poskytuje ze všech vyzkoušených architektur nejlepší výsledky. Zároveň jsou použity bloky 32x32 i přesto, že poskytují podle testů o něco horší výsledky. Je to kvůli tomu, že při výběru z více sítí musíme využít této vlastnosti do plného potenciálu a pokusit se rozdělit původní obraz do co nejvíce bloků. Protože vnitřní sítě nejsou zaměřeny na celé spektrum možných vstupních obrázků, jejich hodnoty výsledků nepřesahují některé ostatní sítě.

Kvůli tomu, že sítě se už netrénují na celých obrázcích, ale jen na specifických vstupních datech, není u grafu 5.5.4 epocha počítána jako jeden obraz, ale pouze jako přibližné množství bloků, které mívá průměrný obrázek v používaném datasetu (96 nebo 144 bloků o rozměrech 32x32 pixelů - převážně 96).

Sharpness síť 0-3, na rozdíl od jí podobné sítě trénované na zvětšených obrázcích, dosahuje nižších hodnot MS-SSIM (0.88420) než je výchozí síť. Podle obrázku 5.16 lze však poznat, že je to kvůli tomu, že nedokáže dobře zkomprimovat nejednolité plochy, které na druhou stranu zvládá obzvlášť dobře.

Sharpness síť 3-10 na druhou stranu již dokáže mnohem lépe zakódovat bloky s více detaily a nemá velké problémy s jednolitými plochami. S výsledkem MS-SSIM 0.92490 dosahuje lepšího výsledku než výchozí síť.

Síť trénovaná na intervalu <10, 20) dosahuje nejlepších výsledků ze samostatných sharpness sítí (0.93861). To může znamenat, že nejčastěji vyskytující se bloky spadají do intervalu <10, 20), síť trénovaná na blocích spadajících do tohoto intervalu umí dobře zpracovávat i bloky z ostatních intervalů, a nebo kombinace obou.

Sharpness síť 20-inf už ukazuje pokles v kvalitě výsledků (MS-SSIM 0.90090), kde jde na obrázku 5.16 vidět, že ztrácí schopnost komprimace jednolitých ploch. Z výsledků této sítě jde vyvodit, že specializování sítě na nejvyšších detailech není zcela optimální pro nejlepší výsledky sítě.

Na grafu 5.5.4 lze navíc vidět, že sítě trénované na intervalech s vyššíma hodnotami ostrosti se trénovaly rychleji. Zároveň všechny sítě dosáhly v závěru na svých specifických vstupech podobných hodnot, což ukazuje na rovnoměrně rozdělený interval hodnot.

2-větvová sharpness síť 0-3-10

Tato síť používá jako vnitřní sítě sharpness síť 0-3, která je trénovaná na blocích s velmi nízkými gradienty a sharpness síť 3-10. Vstupní bloky, které přesahují sharpness 10 jsou poslány na zpracování sítí 3-10.

Výsledky sítě (5.5.4) ukazují, že koncept kombinace sítí pro kompresi obrazu může fungovat. Síť dosahuje výsledků 0.93011 MS-SSIM, což je víc než samostatné vnitřní sítě, a zároveň lepší než síť výchozí.

3-větvová sharpness síť 0-3-10-20

Stejná jako předchozí síť, jen je přidaná vnitřní síť sharpness 10-20, na kterou se budou posílat bloky o ostrosti 10 až nekonečno (teoretická hranice na 1,44*255).

Přidáním třetí a nejlepší ze samostatně otestovaných vnitřních sítí dále zlepšilo výslednou kvalitu komprese. Výsledek 0.94191 MS-SSIM je lepší než všechny vnitřní sítě dohromady.

4-větvová sharpness síť 0-3-10-20-inf

Stejná jako předchozí síť, jen je jako čtvrtá síť přidána sharpness síť 20-inf.

I přes horší kvalitu přidané sítě samostatně, její přidání dále zlepšilo výslednou kvalitu komprese obrazu. Hodnota MS-SSIM 0.94226 je nejlepší ze všech testovaných sítích a dokazuje, že pro kompresi obrazu je celý koncept využitelný.

Obrázek 5.14: **Vlevo nahoře:** Obrázek rekonstruovaný sítí 0-3, jde vidět velmi kvalitní rekonstrukce částí s malým gradientem, ale rozmazání detailů na ostatních částech. **Vpravo nahoře:** Obrázek rekonstruovaný sítí 3-10. Začíná se objevovat narůst šumu v málo detailních částech, ale zlepšuje se kvalita ostatních částí. **Vlevo dole:** Obrázek rekonstruovaný sítí 10-20. Další zlepšení kódování ostrých částí. Malé zvýšení šumu v málo ostrých částech oproti síti 3-10. **Vpravo dole:** Obrázek rekonstruovaný sítí 20-inf. Znatelný nárůst šumu v málo detailních částech. Malé zlepšení některých bloků s velkou ostrostí.

Obrázek 5.15: Porovnání výsledků multibranch sharpness sítí. Rozdíly už jsou pohledem minimální. **Vlevo:** Dvou-větvová sharpness síť 0-3-10 **Uprostřed:** Tří-větvová sharpness síť 0-3-10-20 **Vpravo:** Čtyř-větvová sharpness síť 0-3-10-20-inf

Typ testovaného parametru	**MS-SSIM - 16 it.**
Samostatná rekurentní síť 4.3 (výchozí)	0.91365
Samostatná sharpness síť 0-3	0.88420
Samostatná sharpness síť 3-10	0.92490
Samostatná sharpness síť 10-20	0.93861
Samostatná sharpness síť 20-inf	0.90090
2-větvová sharpness síť 0-3-10	0.93011
3-větvová sharpness síť 0-3-10-20	0.94191
4-větvová sharpness síť 0-3-10-20-inf	**0.94226**

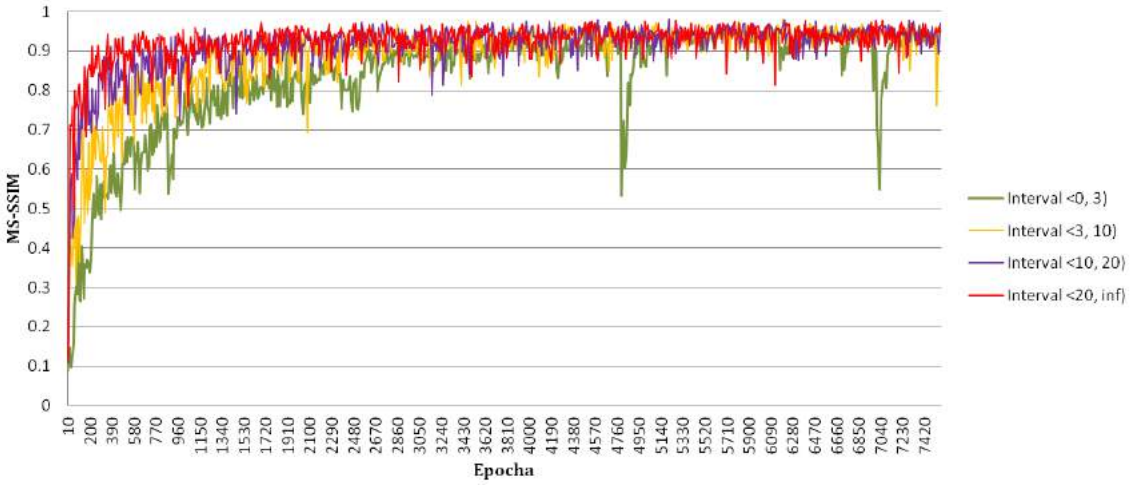
Tabulka 5.6: Výsledné hodnoty MS-SSIM pro sítě trénované pomocí multibranch konceptu zaměřeném na metriku ostrosti a samostatné vnitřní sítě + referenční sítě pro porovnání.

5.5.4 Vyhodnocení sharpness sítí

Předchozí experimenty slouží jako důkaz toho, že použití více sítí s rozdělením a tedy specializací datasetu, může i významně zlepšit výslednou kvalitu obrazu. Metadata, která jsou potřebná pro správný výběr sítě jsou zanedbatelná a navíc i dále zmenšitelná, neboť hodně často se bude opakovat sled bloků, které využily stejnou síť.

Celý tento experiment navíc slouží hlavně jako proof of concept, takže je pravděpodobné, že tyto výsledky i přes snahu nejsou optimální a jdou dále vylepšit například rozdílným rozdělením intervalů, použitím jiné metriky nebo zakomponováním více vnitřních sítí.

Obrázek 5.16: Multibranch sharpness sítě již nemají problémy se spojováním bloků při kombinaci více sítí, na rozdíl od multibranch sítí zaměřených na detaily, jak je vidět na obrázku 5.15



Graf 5.7: Trénování vnitřních sítí podle metriky ostrosti. Intervaly ukazují hodnotu ostrosti u bloků, na kterých byly sítě trénovány. Větší hodnoty intervalu znamenají bloky s větší ostrostí (více rychlých přechodů mezi barvami). Jde vidět, že sítě trénované na výraznějších detailech dosáhnou svých maximálních hodnot rychleji. Jde zároveň vidět, že všechny sítě se na svém specifickém intervalu naučily přibližně na stejné hodnoty MS-SSIM.

5.6 Shrnutí výsledků

Tato kapitola shrnuje všechny výsledky po provedení všech experimentů nad architekturami, parametry a i nad speciálními koncepty. Celkově se ukázalo, že sítě s rekurentními prvky mají významně lepší výsledky než obyčejné sítě (5.3.2), což bylo i základním předpokladem. Z vyzkoušených parametrů se při jejich změnách neukázala mít většina výrazný efekt na výslednou kvalitu sítě, a pokud měly, tak byl spíše negativního charakteru (5.4.7). Mezi jediné parametry, které měly možný pozitivní dopad na kvalitu byly velikost vstupních bloků a trénování zvětšených a zmenšených obrazů, na kterém se založil pozdější koncept specializovaného datasetu. Velikost vstupních bloků i přes lehké zlepšení kvality sítě přináší nevýhody v podobě horšího přizpůsobení sítě různým rozlišením obrazu.

Experimenty nad konceptem specializace obrazu ukázaly, že zde leží možnost významně vylepšit kvalitu sítě. Experimenty nad specializovaným datasetem v podobě map ukázaly sítě, které i ve výchozím stavu bez dalších úprav vysoce předčily všechny ostatní testované sítě (5.5.1). V následujících experimentech, které se snažily napodobit tento efekt, proběhlo několik skupin možností struktur sítí, které neměly optimální výsledky (5.5.2, 5.5.3), což nakonec vyústilo v multibranch sítě trénované podle metriky ostrosti, které využívají více zkombinovaných sítí, každá trénovaná na určitý interval vstupů rozdělených podle dané metriky. Tyto sítě (5.5.4), sloužící jako proof of concept, dokázaly, že využití specializace datasetu je prakticky využitelná, a díky tomu, že pracuje z velké části na úrovni vstupu, může být kombinovaná s velkou částí stávajících sítí.

Výsledky této finální sítě dosahují hodnot MS-SSIM 0.94229 při použití 16 iterací a 0.80262 při použití 8 iterací. V porovnání s výsledky George Todericiho [11],které dosahují výsledků lepších než komprese JPEG, při kompresi za pomocí rekurentní sítě s GRU jednotkami v určitých případech až o 25% zvýšeni komprese oproti formátu JPEG a to při udržení MS-SSIM okolo 0.9. [3]. Při porovnání mojí finální sítě s formátem JPEG, při použití všech 16 iterací se dostávám na stejnou úroveň komprese jako má JPEG při parametru kvality na $\sim$ 90-93. Na obrázku 5.17 jde vidět srovnání.

Jediné sítě, které v moji práci se vyrovnávají kvalitě komprese JPEG a výsledkům George Todericiho, jsou sítě testované na obrázcích map, které v určitých případech i za použití pouze třetiny iterací dosahují výsledků $\sim$0.93 MS-SSIM. Z toho je jasně ukázané, že při použití state-of-the-art sítí na specializovaném datasetu, budou výsledky velmi kvalitní a využitelné i prakticky.

Obrázek 5.17: Porovnání obrázků komprimovaných se stejnou velikostí komprese **Vlevo:** Obrázek komprimovaný pomocí sharpness sítě 0-3-10-20-inf (lehce jiné rozměry kvůli zarovnání na bloky)**Vpravo:** JPEG obrázek komprimovaný s parametrem kvality na 90

Kapitola 6

Závěr

V tomto dokumentu byly probrány základní principy neuronových sítí, včetně rozšiřujících konvolučních a rekurentních neuronových sítí. Zároveň byly ukázány architektury sítí, kterých se dá v současnosti použít pro kompresi obrazu. Na těchto architekturách byly ukázány i základní principy komprese pomocí neuronových sítí.

V druhé části práce byly provedeny experimenty nad různými typy sítí a mnoha kombinací parametrů těchto sítí za účelem zjištění nejefektivnější struktury sítě pro komprimaci obrázků. Ze získaných poznatků z těchto experimentů byly následně navrhnuty a otestovány modely sítí pro získání kvalitnější komprese. Byly otestovány sítě, které byly trénované na specializovaném datasetu a bylo zjištěno, že tyto sítě mají mnohem lepší výsledky při kompresi, což by dalo využít například pro kvalitní komprese map. Testování se zaměřilo i na možnosti jak využít této specializace i na všeobecném datasetu. Na závěr byla navrhnuta a otestována neuronová síť, která kombinovala nejlepší otestované prvky a koncepty jako je například umělé specializování sítě, a dosáhla lepších výsledků než všechny otestované předchozí sítě. I přes to, že tato síť nedosahuje úplně nejlepších výsledků v porovnání se současnými nejlepšími komprimačními sítěmi, koncepty v ní využité by se daly využít pro zvýšení kvality komprese stávajících sítí.

Práce by se dala dále rozšířit vyzkoušením zjištěných konceptů na state-of-the-art sítích pro kompresi, což by ale potřebovalo větší výpočetní sílu. Další možnost je i v rozšíření komprese do časové sféry a pokusit se aplikovat kompresi pomocí neuronových sítí na video, kde by se daly do plné míry využít rekurentní sítě. Podívání se na možnost aplikace více prvků rekurentních sítí je také možnost jak dále rozvíjet tuto práci.

Literatura

[1] *Convolutional Neural Networks for Visual Recognition.* URL http: cs231n.github.io

[2] *Understanding LSTM Networks.* URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] *Image Compression with Neural Networks. Sep 2016.* URL https://ai.googleblog.com/2016/09/image-compression-with-neural-networks.html

[4] *Britz, D.: Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs. Jul 2016.* URL http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[5] *Goodfellow, I.; Bengio, Y.; Courville, A.: Deep learning. MIT Press, 2017.*

[6] *Hochreiter, S.; Urgen Schmidhuber, J.: Long Short-Term Memory.* Neural Computation, *ročník 9, č. 8, 1997: s. 1735–1780, ISSN 0899-7667, doi:10.1162/neco.1997.9.8.1735, 1206.2944.*

[7] *Jiang, J.: Image compression with neural networks - A survey. In* Signal Processing: Image Communication 14, *1999, s. 737–760.*

[8] *Noh, H.; Hong, S.; Han, B.: Learning deconvolution network for semantic segmentation.* Proceedings of the IEEE International Conference on Computer Vision, *ročník 11-18-Dece, 2016: s. 1520–1528, ISSN 15505499, doi:10.1109/ICCV.2015.178, 1505.04366.*

[9] *Raiko, T.; Berglund, M.; Alain, G.; aj.: Techniques for Learning Binary Stochastic Feedforward Neural Networks. 2014: s. 1–10, 1406.2989.*

[10] *Stergiou, C.; Siganos, D.: Neural Networks. [Online; navštíveno 28.04.2017].* URL https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[11] *Toderici, G.; O'Malley, S. M.; Hwang, S. J.; aj.: Variable Rate Image Compression with Recurrent Neural Networks.* International Conference On Learning Representations, *2015: s. 1–9, 1511.06085.*

[12] *Toderici, G.; Vincent, D.; Johnston, N.; aj.: Full Resolution Image Compression with Recurrent Neural Networks.* arXiv:1608.05148, *2016: str. 59, ISSN 08936080, doi:10.4135/9781412985277, 1608.05148.*

Příloha A

Obsah DVD

Přibalené DVD obsahuje složky:

- **SourceFiles** - obsahuje obsahuje zdrojové soubory použité v této práci
- **Video** - obsahuje video prezentující tuto práci
- **Documentation** - obsahuje Tex soubory a PDF verzi tohoto dokumentu