VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Jiří Holub

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ZVÝŠENÍ KVALITY FOTOGRAFIE S POUŽITÍM HLUBOKÝCH NEURONOVÝCH SÍTÍ
SUPERRESULUTION OF PHOTOGRAPHY USING DEEP NEURAL NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Jiří Holub

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Radim Burget, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Jiří Holub ***ID:*** 158144
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Zvýšení kvality fotografie s použitím hlubokých neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou konvolučních neuronových sítí a generativními sítěmi, dále také vrstvami 2D konvoluce, MaxPooling, dávková normalizace, transpozice, a nástroji Tensorflow a Keras. Vytvořte databázi nejméně 1000 snímků, kde ke každému snímku s vysokou kvalitou a rozlišením bude existovat snímek s nízkou kvalitou a 2x nižším rozlišením. Navrhněte několik modelů neuronových sítí, model natrénujte a ohodnoťte a výsledky vhodně reprezentujte.

DOPORUČENÁ LITERATURA:

[1] Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." European Conference on Computer Vision. Springer, Cham, 2014.

[2] Santana, E., Subpixel Framework [Online], https://github.com/tetrachrome/subpixel

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: doc. Ing. Radim Burget, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato diplomová práce se zabývá zvyšováním rozlišení obrázků při zachování jejich dobré kvality. Jsou zde popsány současné metody řešení tohoto problému, dále jsou popsány principy fungování neuronových sítí se zaměřením na sítě konvoluční. Konečně je popsáno několik modelů konvoluční neuronové sítě pro zvýšení rozlišení obrazu na dvojnásobek, které byly natrénovány, otestovány a porovnány na nově vytvořené databázi fotografií lidí.

KLÍČOVÁ SLOVA

Zvýšení rozlišení obrazu, hluboké učení, umělá neuronová síť, generativní síť, konvoluční vrstva, transponovaná konvoluční vrstva, TensorFlow

ABSTRACT

This diploma thesis deals with image super-resolution with conservation of good quality. Firstly, there are described state of the art methods dealing with this problem, as well as principles of neural networks with focus on convolutional ones. Finally, there is described a few models of convolutional neural network for image super-resolution to double size, which have been trained, tested and compared on newly created database with pictures of people.

KEYWORDS

Image super-resolution, deep learning, artifical neural network, generative network, convolutional layer, transposed convolutional layer, TensorFlow

HOLUB, Jiří. *Zvýšení kvality fotografie s použitím hlubokých neuronových sítí*. Brno, 2018, 60 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Radim Burget, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Zvýšení kvality fotografie s použitím hlubokých neuronových sítí“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Už od dob vzniku digitální reprezentace fotografií a jiných obrázků ve formě bitmapového obrazu je řešen problém, jak tento obraz zvětšit a přitom zachovat jeho kvalitu. Bitmapový obraz má pevný počet obrazových bodů, při jeho zvětšení je nutné doplnit obraz body novými. Informace o jejich hodnotě v obrázku není, je nutné si vypomoci jinak, např. interpolováním této hodnoty z okolních pixelů nebo složitějšími metodami obnovení chybějící informace na základě dříve naučených vzorů.

Zvyšování rozlišení obrázků najde uplatnění jak v běžném životě, např. při přiblížení obrázků v této práci nebo při úpravě fotografií k tisku, tak pro specifické aplikace jako je přiblížení satelitních a leteckých snímků nebo snímků z lékařských přístrojů jako je magnetická rezonance. Se stále větším rozšířením bezpečnostních kamer je také často potřebné přiblížení z nich pocházejících záznamů pro lepší identifikaci pachatele trestného činu.

Cílem této diplomové práce je navrhnout metodu pro zvyšování rozlišení obrázků s využitím hlubokého učení. K tomu je nutné vytvořit rozsáhlou databázi trénovacích snímků. Navržená metoda by měla být použita pro zvyšování rozlišení fotografií osob a tomu odpovídají i trénovací data.

První kapitola se zabývá současnými metodami pro zvyšování rozlišení obrazu, ať už jde o jednoduchou interpolaci používanou ve většině fotografických editorů nebo moderní metody publikované v posledních letech v odborné literatuře jako je využití neuronových sítí, adaptivních filtrů nebo řídké reprezentace signálů.

Druhá kapitola pojednává o neuronových sítích obecně, vysvětleny zde jsou jejich základní principy se zaměřením na sítě konvoluční, které jsou v dnešní době často používány pro práci s obrázky, ať už pro detekci a klasifikaci objektů nebo pro vytváření obrázků např. s vyšším rozlišením.

Ve třetí kapitole jsou popsány konkrétní modely konvolučních neuronových sítí pro zvýšení rozlišení obrazu, které byly v rámci této práce natrénovány. Najdete zde popis jejich funkce z matematického pohledu a také jsou zde popsány jejich architektury a parametry.

Ve čtvrté kapitole jsou uvedeny informace o tom, jak připravit prostředí nutné k práci s neuronovými sítěmi, je zde popsána implementaci modelů popsaných v kapitole předchozí a také je zde popsáno trénování a sada fotografií, která k němu byla použita.

V poslední kapitole jsou shrnuty výsledky práce. Je zde uvedeno srovnání jednotlivých modelů pomocí objektivních metod hodnocení kvality obrázků jak na nově vytvořené testovací sadě tak na sadách běžně používaných v odborných článcích. Nakonec je popsána jednoduchá aplikace s grafickým uživatelským rozhraním, která slouží pro jednoduší zvětšování obrázků.

1 SOUČASNÉ METODY PRO ZVÝŠENÍ ROZLIŠENÍ OBRAZU

Zvýšení rozlišení obrazu (anglicky Image Super Resolution) je proces získávání obrazu s vysokým rozlišením (High Resolution – HR) z obrazu s nízkým rozlišením (Low Resolution – LR). Tato problematika je studována od dob vzniku bitmapového obrazu a nachází uplatnění v mnoha oblastech, například zvětšení textu, převod videa nebo obrázků do vyššího rozlišení, zvětšení snímků z mikroskopu nebo přiblížení záznamů z bezpečnostních kamer.

Základní metodou pro získání obrazu ve vysokém rozlišení je kombinace několika obrazů téže scény s nízkým rozlišením, které jsou vzájemně porovnány se sub-pixelovou přesností. Ne vždy však je k dispozici několik obrázků a je tak třeba vycházet z jediného obrazu. V této kapitole jsou popsány současně používané metody pro zvýšení rozlišení samostatného obrazu (Single Image Super Resolution – SISR).

1.1 Převzorkování

Základní a nejjednodušší metodou pro zvýšení rozlišení obrazu je převzorkování. Jedná se o změnu počtu pixelů, nově vytvořeným pixelům je poté nutné přiřadit hodnoty. Tyto jsou interpolovány z hodnot původních pixelů. Existuje několik algoritmů, každý má své výhody a nevýhody a jinou oblast použití.

1.1.1 Nejbližší soused

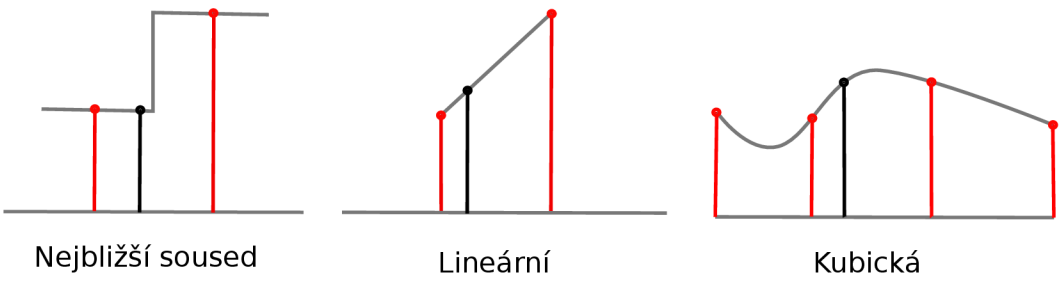
Jedná se o nejjednodušší metodu interpolace, která použije hodnotu nejbližšího pixelu. Má tedy stejný výstup jako zvětšení plochy pixelu. Je rychlá, ale velmi nepřesná, hodí se proto jen pro rychlé náhledy. [1]

1.1.2 Bilineární

Hodnota nového pixelu je vypočítána jako lineární kombinace čtyř hodnot, tudíž je také rychlá a poskytuje lepší výsledky než nejbližší soused. Jelikož se jedná o průměrování hodnot, je výsledný obraz rozostřený a proto je dobré jej doostřit vhodným filtrem. [1]

1.1.3 Bikubická

Pro výpočet nové hodnoty používá 16 okolních pixelů, přičemž ty bližší mají větší váhu než ty vzdálenější. Může být spočítána pomocí Lagrangeových polynomů, kubickými spliny nebo algoritmem kubické konvoluce. Tato metoda je pomalejší než dvě předchozí, zachovává však více detailů a výsledný obraz je hladší a obsahuje méně artefaktů. Výsledek je opět mírně rozostřený a je dobré je doostřit. [1]



Obr. 1.1: Porovnání interpolačních metod, pro názornost v 1D

1.1.4 Fraktálové

Jedná se o adaptivní metodu, pracuje tedy v závislosti na právě zpracovávaném obraze. Tento obraz je kódován do fraktálů pomocí fraktálové komprese a následně dekódován ve vyšším rozlišení. Využívá se zde matematických vlastností fraktálů, díky čemuž je možné ho obnovit v libovolné velikosti. Tato metoda je vhodná pro přírodní obrazy, ve kterých se dají snáze najít fraktály. [1]

Tuto metodu dále rozvíjí např. nedávno publikovaná práce [2]. Tato metoda kombinuje vlastnosti racionální interpolace a fraktálové interpolace. Obraz je rozdělen na texturové (hrany) a strukturální oblasti, s každou touto oblastí je pak pracováno jinak. Díky odlišnému způsobu zpracování různých oblastí je možné dosáhnou lepší rekonstrukce obrazu. Poté je obrázek rozdělen na bloky 5×5 pixelů, pro každý blok je vypočteno měřítko na základě jeho obsahu a poté je interpolován racionální fraktálovou interpolací pro texturové oblasti a běžnou racionální interpolací pro strukturální oblasti. Následně jsou interpolované pixely mapovány na požadované výsledné měřítko. [2]

Všechny výše popsané interpolační metody mají výhodu oproti metodám popsaným ve zbytku této kapitoly a sice to, že jsou nezávislé na zpracovávaných datech. Pracují pouze s hodnotami pixelů v aktuálním obrázku, zatímco ostatní metody potřebují pro svou funkci nejprve data pro natrénování (vytvoření filtrů, slovníků

nebo vah neuronové sítě) a proto může jejich výkonnost kolísat pro různé typy dat. Jinak řečeno, interpolační metody jsou univerzální, zatímco metody pracující podle naučených vzorů mohou pracovat pouze na specifických datech.

1.2 Slovníkové metody

Tyto metody jsou založené na vytvoření dvojice slovníků – odpovídající si páry dat z obrazů o nízkém a vysokém rozlišení, slovníky jsou tudíž vytvářeny z trénovacích dvojic obrázků. Tyto data ve slovnících mohou být různého typu, obvykle jsou jimi vektory příznaků extrahované z obrázků ve formě gradientů nebo jiných vyjádření geometrické struktury na obrázku. Způsob vytváření těchto slovníků a následná rekonstrukce obrazu se liší v různých metodách, v této kapitolce budou uvedeny některé z nich.

1.2.1 Řídká reprezentace

Tato metoda popsaná v práci [3] je založena na řídké reprezentaci signálů, což lze chápat jako nedourčený systém lineárních rovnic, kde existuje řešení, které má velmi málo nenulových proměnných [4]. V praxi to znamená, že je nalezena řídká reprezentace každé oblasti (patch) v obraze s nízkým rozlišením a podle jejích koeficientů je vygenerována oblast obrazu s vysokým rozlišením.

Trénování slovníků probíhá tak, že obrázky s vysokým i nízkým rozlišením jsou rozděleny na odpovídající si menší části ve kterých jsou nalezeny určité příznaky (features). V této práci jsou jako příznaky použity první a druhá derivace v horizontálním i vertikálním směru. Tímto jsou vytvořeny čtyři mapy gradientů, které jsou poté sloučeny a pak slouží jako výsledný příznak. Následně je nalezena řídká reprezentace tohoto příznaku, tak aby byla stejná pro oblast vysokého i nízkého rozlišení. Tyto operace se neprovádí na původním obrázku s nízkým rozlišením, ale na jeho bikubicky interpolované verzi.

Při zvyšování rozlišení je pak obrázek interpolován, rozdělen na oblasti, v nich jsou nalezeny příznaky, je nalezena jejich řídká reprezentace a v předtrénovaném slovníku je nalezen odpovídající příznak s vysokým rozlišením. [3]

1.2.2 Sdružování sousedů

Tyto metody jsou založeny na podobné lokální geometrii. Jedná se opět o metodu učící se z příkladů. Obrázky ve vysokém a nízkém rozlišení se stejně jako u předchozí metody rozdělí na odpovídající si oblasti (patches). Z těchto částí je vytvořen příznakový vektor, který je uložen ve slovníku. Při rekonstrukci pak není použit pouze

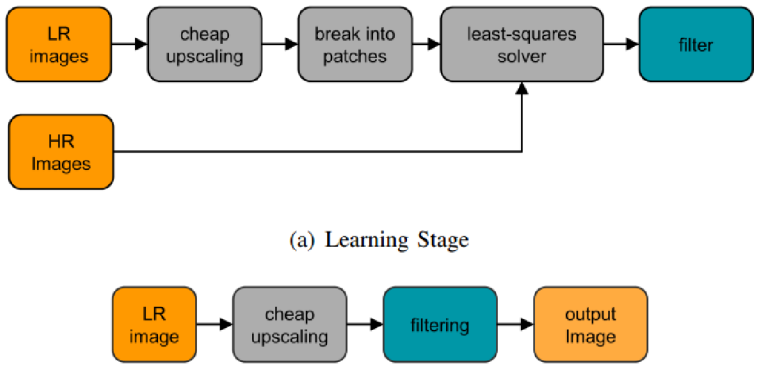
jeden příznakový vektor, ale v oblasti LR je nalezeno K nejbližších sousedů, jejich odpovídající protějšky v HR oblasti jsou pak sloučeny do jednoho příznaku.

Bylo publikováno více metod, liší se zejména extrakcí příznakových vektorů a výběrem nejbližších sousedů. Například v práci [5] jsou jak příznaky použity gradienty prvního a druhého řádu a nejbližší sousedé jsou vybráni podle euklidovské vzdálenosti. Poté je vypočítáno váhové zastoupení jednotlivých příznakových vektorů a pro rekonstrukci jsou použity jim odpovídající příznakové vektory v HR oblasti s vypočítanými váhami.

Další metoda publikovaná v práci [6] používá jako příznaky jasovou složku obrázku, pro výběr sousedů také euklidovskou vzdálenost, ale váhy jednotlivých sousedů jsou počítány pomocí metody nejmenších čtverců a jsou vždy větší než nula. Další úspěšná metoda, publikovaná v [7], využívá slovníku naučeného pomocí řídké reprezentace ve kterém poté vybírá nejbližší sousedy podle euklidovské vzdálenosti.

1.3 Metody založené na filtraci

Tyto metody fungují tak, že obraz s nízkým rozlišením je nejprve nadvzorkován na požadovanou velikost např. bikubickou interpolací a tento obraz je následně vyfiltrován speciální sadou filtrů. Nejjednodušším příkladem je zostřovací filtr typu horní propust, který zvýrazní hrany.



Obr. 1.2: Základní schéma algoritmu RAISR [8]

Více sofistikované řešení je metoda RAISR: Rapid and Accurate Image Super Resolution, publikovaná v článku [8]. Jedná se o algoritmus vyvinutý v laboratořích Google a je založena na natrénování sady filtrů z dvojic trénovacích obrázků s nízkým a vysokým rozlišením. Při učení jsou obrázky rozděleny bloky a ty jsou pak rozděleny do skupin s podobnou geometrií, filtr je následně naučen pro každou skupinu zvlášť. Při zvyšování rozlišení je na základě geometrie obrázku vybrán patřičný předtrénovaný filtr. Potlačení artefaktů vzniklých při rekonstrukci je dosaženo

smísením bikubicky interpolovaného a vyfiltrovaného obrazu. Schéma této metody můžete vidět na obrázku 1.2. [8]

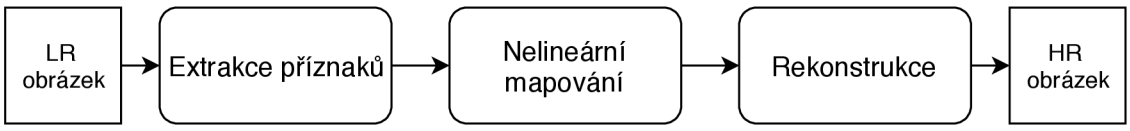
Tato metoda je v současné době používána např. pro kompresi obrázků. Obrázek je podvzrokován na čtvrtinovou velikost, při této operaci jsou vytvořeny vhodné filtry, které jsou uloženy spolu s podvzorkovaným obrázkem. Při rekonstrukci je poté obrázek nadvzorkován pomocí bikubické interpolace a vyhlazen dříve vytvořenými filtry. Údajně je díky tomu možné ušetřit 75 % objemu dat. [9]

1.4 Metody založené na hlubokém učení

Použití hlubokého učení je v principu podobné předchozím metodám. Také se učí z trénovacích dvojic obrázků s nízkým a vysokým rozlišením, výstupem však není filtr nebo slovník, ale naučená neuronová síť, jejíž váhy odpovídají příznakům vyskytujícím se v obraze.

Metody využívající hlubokého učení můžeme rozdělit na dva typy podle způsobu rekonstrukce obrazu: buď je obraz zvětšen např. bikubickou interpolací a následně je neuronovou sítí filtrován pro zvýraznění hran a rekonstrukci detailů nebo je vstupem sítě původní obrázek s nízkým rozlišením a neuronová síť na základě naučených příznaků generuje obrázek s vysokým rozlišením (generativní síť).

Téměř všechny neuronové sítě pro zvyšování rozlišení se dají zjednodušeně popsat blokovým schématem na obrázku 1.3. Ze vstupního obrázku s nízkým rozlišením (nebo jeho bikubické interpolace) jsou extrahovány příznaky (hrany, složitější textury), které jsou poté nelineárně mapovány do HR prostoru. Poslední částí sítě je rekonstrukční, kdy jsou data převedena do zpět barevného prostoru a požadované velikosti. Různé modely implementují tyto části sítě různě ve snaze dosáhnout co nejlepších výsledků. Ve zbytku této podkapitoly budou popsány principy některých z nich.



Obr. 1.3: Blokové schéma obecné neuronové sítě pro zvýšení rozlišení obrazu

Jedním z prvních algoritmů využívajících neuronové sítě pro zvýšení rozlišení je SRCNN (Super-Resolution Convolutional Neural Network), popsaný v práci [10]. Tato síť se skládá ze tří vrstev, první konvoluční vrstva, jejímž vstupem je bikubicky interpolovaný obrázek, extrahuje mapy příznaků, druhá vrstva je nelineárně mapuje

na mapy příznaků v oblasti HR a poslední vrstva produkuje finální obraz z predikovaných příznaků. Tato metoda produkuje relativně dobré výsledky, nicméně její výkonnost už byla překonána a nyní často slouží jako referenční metoda pro srovnávání nově vyvíjených metod. [10]

Dalším zajímavým modelem je síť DCSCN (Deep CNN with Residual Net, Skip Connection and Network in Network), publikovaná v práci [11]. Jedná se o generativní síť, jejím vstupem je obrázek s nízkým rozlišením a ne jeho interpolace. přičemž zvětšení rozlišení probíhá v rámci průchodu sítí. Díky tomu je model méně výpočetně náročný. Síť se skládá ze sedmi konvolučních vrstev, které zajišťují extrakci příznaků. Další, rekonstrukční, část sítě využívá paralelní strukturu pro nadvzorkování dat. Její výstup je poté sečten s bikubickou interpolací vstupu a tím je získán výsledný obrázek. Síť se tedy neučí celý obrázek, ale pouze rozdíl mezi interpolací a skutečným obrázkem s vysokým rozlišením (reziduální učení). [11]

Zajímavý přístup je zvolen také v modelu nazvaném Large Receptive Field Net (LRFCNN) představeném v práci [12]. Zde jsou pro extrakci příznaků použity vrstvy s velkými rozměry konvolučních jader, konkrétně 61×1 a 1×61, zatímco většina modelů používá obvykle velikosti 3×3 nebo 5×5. Poté následují běžné vrstvy pro rekonstrukci. Kvůli velké velikosti konvolučních jader je trénování a vybavování sítě pomalé, nicméně dle autorů dosahuje dobrých výsledků. [12]

V posledních letech bylo publikováno mnoho dalších modelů, některé z nich byly použity v praktické části této práce a proto jsou detailněji popsány v kapitole 3. Dále uveďme ve zkratce několik dalších zajímavých myšlenek a modelů: V práci [13] je představen model, který pro konečnou rekonstrukci využívá jak vysokoúrovňové tak i nízkoúrovňové příznaky a zároveň navrhuje víceparametrovou chybovou funkci. Kombinaci hlubokého učení, slovníkových metod a řídké reprezentace představili autoři v práci [14]. Zvyšování rozlišení hyperspektrálních obrázků, neboli obrázků obsahujících více kanálů elektromagnetického spektra, se zabývá práce [15].

1.5 Další metody

Mezi další metody, které používají odlišný přístup než předchozí můžeme zařadit např. práci [16], ve které je využita vlnková transformace soběpodobnostních vlastností obrázků. Vstupní obrázek je zde transformován pomocí vlnkové transformace a k existujícím subpásmům je predikováno další, které reprezentuje detaily. Využívá se zde toho, že bloky, které jsou si podobné v prostorové oblasti, vykazují stejnou podobnost i ve vlnkové oblasti a toho, že vlnkové koeficienty v nižších pásmech jsou jen zvětšenou verzí těch ve vyšších pásmech. Nakonec je provedena zpětná vlnková transformace a tím vznikne obrázek s vysokým rozlišením. [16]

2 NEURONOVÉ SÍTĚ PRO ZPRACOVÁNÍ OBRAZU

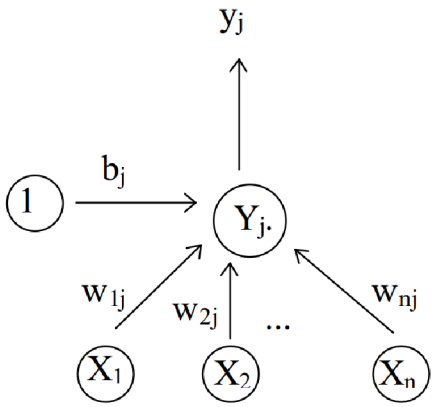
Umělé neuronové sítě jsou výpočetním modelem jehož chování je odvozeno z chování biologických neuronů v živých organismech. Existují různé druhy neuronových sítí, v této kapitole bude popsán jejich základní princip a poté budou popsány konvoluční sítě vhodné pro zpracování obrazu.

2.1 Obecný princip neuronových sítí

Neuronové sítě se skládají z jednoho nebo více umělých neuronů uspořádaných v jedné nebo více vrstvách, které mohou být navzájem pospojované a předávají si signály, které následně vyhodnocují pomocí svých přenosových funkcí.

2.1.1 Neuron

Základem modelu neuronové sítě je formální neuron. Obecně má n vstupů $x_1, ..., x_n$, které jsou ohodnoceny synaptickými váhami $w_{1j}, ..., w_{nj}$, které určují jejich propustnost. Práh b_j neboli bias určuje prahovou hodnotu aktivace neuronu. [17]



Obr. 2.1: Formální neuron [17]

Vážená suma vstupních hodnot představuje vnitřní potenciál neuronu:

$$Y_j = \sum_{i=1}^{n} w_{ij} x_i + b_j, \tag{2.1}$$

výstup neuronu je poté dán výstupem aktivační funkce jejímž argumentem je vnitřní potenciál neuronu:

$$y_j = S(Y_j). \tag{2.2}$$

Aktivační funkce

Aktivační funkce zavádí nelinearitu na výstup neuronu. Následuje výčet nejpoužívanějších aktivačních funkcí:

- Skoková funkce:
$$f(x) = \begin{cases} 0 \text{ nebo } -1, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{2.3}$$
- Sigmoida:
$$f(x) = \frac{1}{1 - \mathrm{e}^{-x}} \tag{2.4}$$
- Hyperbolický tangens:
$$f(x) = \frac{1 - \mathrm{e}^{-x}}{1 + \mathrm{e}^{-x}} \tag{2.5}$$
- ReLU (Rectified Linear Unit):
$$f(x) = max(0, x) \tag{2.6}$$
Případně její varianty Leaky ReLU:
$$f(x) = \begin{cases} x, & x \geq 0 \\ 0,01x, & x < 0 \end{cases} \tag{2.7}$$
a Parametrized ReLU:
$$f(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \tag{2.8}$$

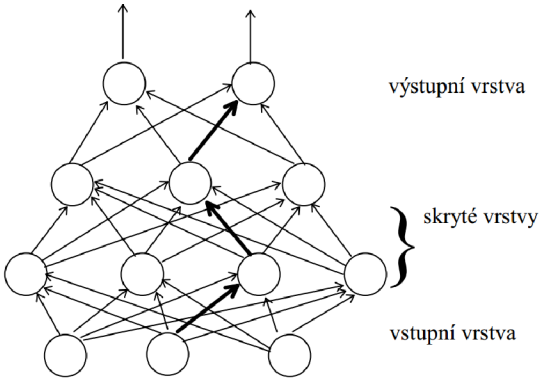
2.1.2 Neuronová síť

Samotný neuron dokáže pouze rozdělit data do dvou tříd, tedy rozdělit například rovinu přímkou. Proto se neuronové sítě skládají z více neuronů uspořádaných do několika vrstev. Příklad takovéto sítě můžete vidět na obrázku 2.2. Vstupy neuronů v jedné vrstvě jsou spojeny s výstupy neuronů ve vrstvě předchozí (kromě vstupní vrstvy) a jejich výstupy jsou připojeny na vstupy neuronů vrstvy následující (kromě výstupní vrstvy). [18]

Takováto síť se nazývá dopředná, protože data jsou posouvána směrem od vstupu k výstupu beze smyček. Dalším typem sítí jsou rekurentní sítě, ve kterých se nacházejí smyčky a informace se tak mohou přenášet i od vrstev vyšších k vrstvám nižším. Tento typ sítí má specifické využití a v této práci nebude popisován.

2.1.3 Trénování

Aby neuronová síť správně fungovala musí být nejprve natrénována. Existuje více učících algoritmů, ten nejběžnější a nejdůležitější pro tuto práci se nazývá učení



Obr. 2.2: Příklad vícevrstvé neuronové sítě [17]

s učitelem. V praxi to znamená předložení vstupních dat na vstup sítě, tyto data jsou sítí zpracována a na výstupu porovnána s požadovanými výstupními daty, které zde vystupují jako učitel. Proces trénování pak znamená úpravu vah sítě tak, aby se výstup co nejvíce blížil požadovanému výstupu. Dále bude vysvětleno několik základních pojmů.

Tréninková množina: obsahuje tréninkové vzory popisující danou problematiku. Skládá se ze vstupů a požadovaných výstupů.

Chyba sítě: míra naučenosti, je odchylka mezi skutečnými a požadovanými výstupy neuronové sítě. Tato hodnota je určena pomocí chybové funkce (loss function), která bývá nejčastěji realizována jako střední kvadratická odchylka:

$$f(w) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, w) - t_n)^2, \tag{2.9}$$

kde $f(w)$ je chybová funkce, $y(x_n, w)$ je výstup sítě závislý na vstupu x_n a hodnotách vah w a t_n je požadovaný výstup. Tato funkce může být modifikována pro použití na různých typech dat, například pro rozdíl pixelů u obrázků. Existují další chybové funkce jako je například Cross entropie používaná v sítích pro klasifikaci vstupních dat k jedné z výstupních tříd (např. rozpoznávání číslic).

$$E = \sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c}\left(\frac{y_{n,c}}{t_{n,c}}\right), \tag{2.10}$$

kde y je výstup sítě, c je třída a t je požadovaný výstup. [19]

Adaptace vah: proces, jehož cílem je minimalizace chyby sítě. Nejčastěji je používán algoritmus snižování hodnoty gradientu (gradient descent), tedy nalezení směru, ve kterém ztrátová funkce klesá nejrychleji (v závislosti na měnících se parametrech sítě). Gradient se získá jako součet parciálních derivací chybové funkce. Tímto způsobem je nalezeno lokální minimum chybové funkce sítě.

Zpětné šíření chyby: back propagation, je algoritmus, který vypočtenou chybu sítě distribuuje zpět přes celou síť a promítá ji jako úpravu váhových hodnot.

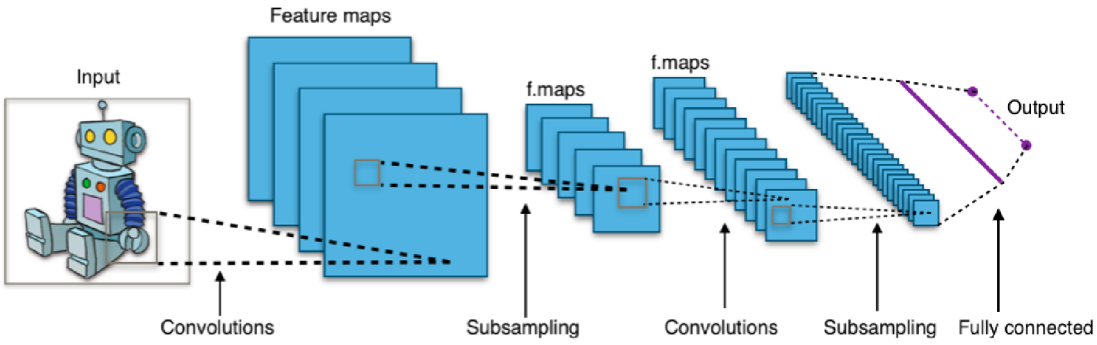
Koeficient učení: learning rate, obvykle značen jako α, určuje velikost kroku při úpravě vah. Nabývá hodnot 0–1, přičemž 0 znamená, že k žádné úpravě vah nedochází. Větší koeficient učení znamená rychlejší konvergenci, nicméně natrénovaná síť může být méně přesná. Naproti tomu při nízkém koeficientu učení je konvergence pomalejší, ale výsledek může být přesnější. [17]

2.2 Konvoluční sítě

Jedná se o speciální neuronové sítě, které používají operaci konvoluce místo maticového násobení alespoň v jedné z vrstev. Příklad konvoluční sítě můžete vidět na obrázku 2.3. Pro zpracování obrazu se využívá 2D diskrétní konvoluce, která je definovaná vztahem:

$$f(x,y) * h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-i) \cdot h(i,j), \tag{2.11}$$

kde $f(x,y)$ a $h(x,y)$ jsou 2D signály (obrázky). [20]



Obr. 2.3: Příklad typické konvoluční neuronové sítě [21]

2.2.1 Konvoluční vrstva

Konvoluční vrstva se skládá z konvolučních jader (kernels), což jsou číslicové filtry v prostorové oblasti. Tyto jádra mají určitou velikost, např. 3×3. Konvoluční jádro je posouváno po vstupním obrázku s krokem s (stride) a pro každý krok je vypočítána konvoluce mezi konvolučním jádrem a odpovídající částí obrázku. Pokud filtr zasahuje mimo oblast obrázku, chybějící pixely jsou nahrazeny nulami (zero-padding). Jednotlivé filtry mají za úkol detekovat v obraze různé příznaky (např. hrany), jejichž přítomnost se projeví vyšší odezvou na tento filtr (vyšší výsledek konvoluce).

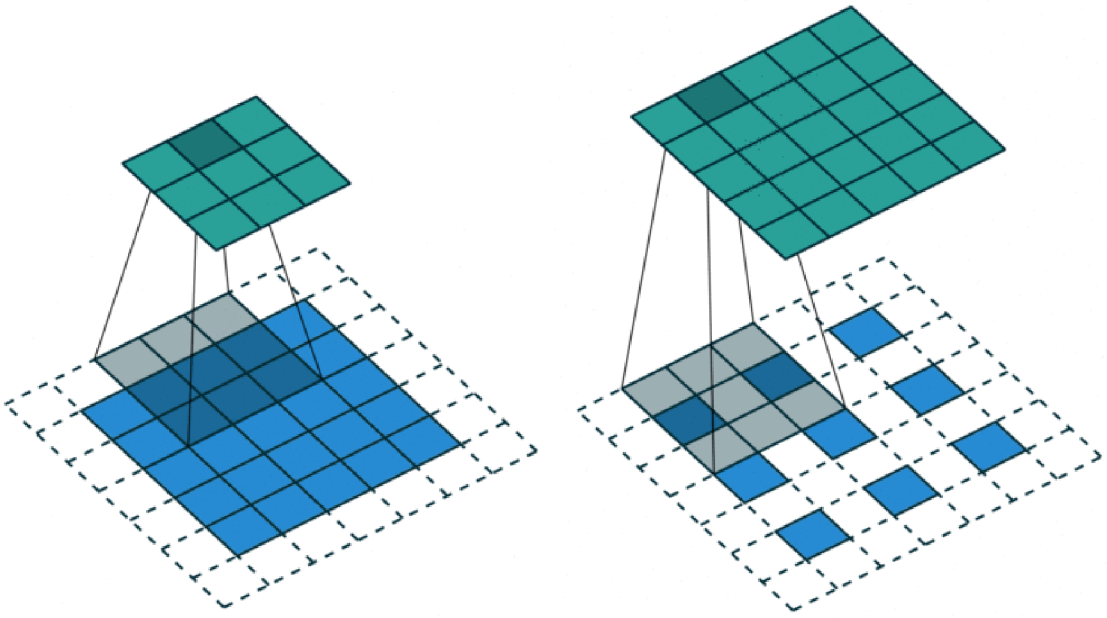
Výstupem konvoluční vrstvy jsou mapy příznaků odpovídající jednotlivým druhům filtrů (hrany nebo složitější objekty). Počet filtrů pak udává počet map příznaků. Tyto jsou pak dále zpracovávány dalšími vrstvami. [22]

Transponovaná konvoluční vrstva

Jinak též nazývaná sub-pixelová konvoluční vrstva nebo dekonvoluční vrstva. Rozdíl je v tom, že zatímco konvoluční vrstva snižuje velikost vstupních dat, transponovaná konvoluční vrstva jejich rozměr zvyšuje. Používá stejnou operaci konvoluce, ale využívá vyplňování (padding) vstupních data a různě velký krok. Odtud také název sub-pixelová konvoluce, neboť při této operaci jsou indexy dat ve tvaru zlomků a chybějící data na neceločíselných souřadnicích jsou doplněna nulami. [23]

Pokud jsou na vstup konvoluční vrstvy s jádrem o velikosti k vložena data o rozměru n, její výstup je poté přiveden na vstup transponované konvoluční vrstvy se stejným jádrem, výstupem budou data o stejném rozměru jako na vstupu konvoluční vrstvy, ale jejich hodnota bude odlišná (nejedná se o matematickou operaci dekonvoluce). Tato vlastnost může být využita např. při zvyšování rozlišení obrázku.

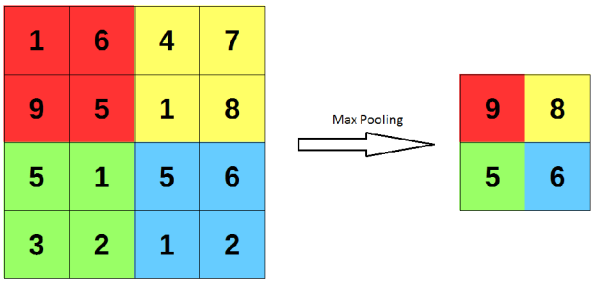
Příklad operace konvoluční vrstvy (vstup 5×5, jádro 3×3, krok 2, vyplňování, výstup 3×3) a transponované konvoluční vrstvy (vstup 3×3, jádro 3×3, krok 2, vyplňování, výstup 5×5) můžete vidět na obrázku 2.4.



Obr. 2.4: Příklad operace konvoluční a transponované konvoluční vrstvy [24]

2.2.2 Sdružovací vrstvy

Neboli pooling vrstvy, obvykle v architektuře sítě následují po konvolučních vrstvách a jejich úkolem je snížit rozměry příznakových map. Slučovací vrstva rozdělí příznakovou mapu na menší čtvercové části, např. 2×2, které se vzájemně nepřekrývají a z každé této části produkuje jednu výstupní hodnotu, kterou může být maximum (max pooling) nebo průměrná hodnota (average pooling). Jinými slovy sdružovací vrstva se skládá z filtru, který je posouván po příznakové mapě s krokem rovným velikosti filtru, odezvou na tento filtr je pak např. maximální hodnota této oblasti. Další výhodou redukce rozměrů dat sdružovací vrstvou je větší nezávislost na posunutí vstupního obrazu. [25]



Obr. 2.5: Ukázka vrstvy Max pooling

2.2.3 Další typy vrstev

Výše popsané základní typy vrstev bývají doplněny o další druhy vrstev, které mají obvykle za úkol úpravu dat před dalším zpracováním. Následuje výčet některých z nich.

Plně propojená vrstva

Fully-connected layer nebo dense layer. Jedná se o běžnou vrstvu neuronových sítí znázorněných na obrázku 2.2. Provádí lineární operaci se vstupním vektorem a váhami, každý prvek vstupu je připojen na prvek výstupu. Tato operace bývá následována aktivační funkcí. [26]

Normalizační vrstva

Slouží k normalizaci výstupů předchozí vrstvy. Např. posunutím střední hodnoty blízko k nule a standardní odchylky k jedné. [26]

Slučovací vrstva

Merge layers, slouží ke sloučení výstupů více vrstev, např. k jejich sečtení po jednotlivých prvcích (Element Wise Sum). [26]

2.3 Knihovny pro hluboké učení

Implementace hlubokých neuronových sítí může být velmi složitá, z toho důvodu vznikla řada frameworků a knihoven, které tuto práci usnadňují. Tyto knihovny přicházejí s jednotlivými vrstvami ve formě funkcí, které se předají vstupní data a parametry vrstvy. Tato funkce poté vrací patřičně zpracovaná výstupní data, která mohou být použita opět jako vstup další vrstvy. Vývojář se díky tomu nemusí zabývat programováním vrstev na nejnižší úrovni, ale využívá vrstvy jako funkční bloky, díky čemuž je jeho práce jednodušší a výzkum mnohem rychlejší. V dnešní době lze vybírat z velkého množství těchto knihoven, např.

- TensorFlow,
- Keras,
- Theano,
- Torch,
- Caffe,
- CNTK,
- Matlab Neural Network Toolbox.

2.3.1 TensorFlow

TensorFlow je open-source softwarová knihovna pro různé matematické úlohy, včetně strojového učení. Pro výpočty využívá grafy datových toků, uzly v grafu reprezentují matematické operace, hrany pak vícerozměrné datové pole (tensory) na které jsou operace aplikovány. TensorFlow byl vyvinut týmem Google Brain pro interní potřeby společnosti Google. Později byl uvolněn pod licencí open-source. Je napsán v Pythonu a umožňuje výpočty na procesorech i na grafických kartách. [27]

2.3.2 Keras

Keras je vysokoúrovňové API pro práci s neuronovými sítěmi. Je napsán v jazyce Python a může být použit jako nástavba na knihovny TensorFlow, Theano nebo CNTK. Vyvinut byl s cílem zjednodušit a zrychlit výzkum v oblasti neuronových sítí. Mezi jeho hlavní vlastnosti patří jednoduché a rychlé prototypování díky uživatelské přívětivosti a modularitě, podpoře konvolučních i rekurentních sítí a schopnosti provádět výpočty na procesoru i grafické kartě. [26]

3 POUŽITÉ MODELY NEURONOVÝCH SÍTÍ

Pro zvyšování rozlišení obrázků bylo v této práci použito několik existujících modelů neuronových sítí, některé byly převzaty i se zdrojovým kódem, jiné byly implementovány autorem práce podle informací v publikacích. Vybrány byly na základě uváděné úspěšnosti zvyšování rozlišení, originalitě návrhu a také podle data publikování.

Konkrétně se jedná o modely: SRGAN (Super Resolution Generative Adversarial Network) [28], DRCN (Deeply–Recursional Neural Network for Super Resolution) [29], ESPCN (Efficient Subpixel Convolutional Network) [30] a VDSR (Very Deep Super Resolution) [31]. Na základě těchto modelů byl následně navržen vlastní model neuronové sítě kombinující některé jejich vlastnosti ve snaze dosáhnout lepších výsledků.

3.1 SRGAN

Tento model byl publikován v roce 2017 v práci [28]. Dle autorů dosahuje dobrých výsledků na obvyklých tetovacích sadách a také jeho architektura je velmi zajímavá. Jako základ práce na tomto modelu byl použit zdrojový kód dostupný zde:[32], který byl dále upravován.

Jedná se o dopřednou konvoluční síť G_{θ_G} pro generování obrazu s vysokým rozlišením I^{HR} z obrazu s nízkým rozlišením I^{LR}. Parametry této sítě jsou $\theta_G = \{W_{1:L}; b_{1:L}\}$, což jsou váhy a prahy jednotlivých vrstev, které jsou získány pomocí chybové funkce l^{SR}. Funkce této sítě je vyjádřena vzorcem:

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=1}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR}). \tag{3.1}$$

3.1.1 Chybová funkce

Na rozdíl od většiny současných metod, kde je chybová funkce založená na optimalizaci MSE, zde je navržena jako vážená kombinace chyby obsahu a protikladné chyby (Adversarial loss):

$$l^{SR} = l_{VGG}^{SR} + 10^{-3} l_{Gen}^{S} R, \tag{3.2}$$

kde l_{VGG}^{SR} je chyba obsahu a $l_{Gen}^{S}R$ je protikladná chyba. [28]

Chyba obsahu

Metody založené na optimalizaci MSE mají problémy s obnovením vysokofrekvenčních detailů. Proto je zde navíc použita chybová funkce VGG založená na ReLU (Rectified Linear Unit) aktivačních vrstvách předtrénované sítě popsané v [33].

$$l^{SR}_{VGG/i,j} = \frac{1}{W_{i,j}Hi,j}\sum_{x=1}^{W_{i,j}}\sum_{y=1}^{H_{i,j}}(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2, \tag{3.3}$$

kde $\phi_{i,j}$ je mapa příznaků (feature map) j-té konvoluce před i-tou poolingovou vrstvou sítě VGG19 a $W_{i,j}$ q Hi,j jsou rozměry mapy příznaků. Chybová funkce je tedy definována jako euklidovská vzdálenost mezi reprezentací rekonstruovaného obrazu a originálního obrazu. [28]

Protikladná chyba

Pro zajištění výstupu co nejpodobnějšího přirozeným obrázků je v síti přidaná tato složka chybové funkce. Je založena na pravděpodobnosti rozpoznání, zda jde o vygenerovaný nebo přirozený obraz:

$$l^{SR}_{Gen} = \sum_{n=1}^{N} -\log D_{\theta_D}(G_{\theta_G}(I^{LR})), \tag{3.4}$$

kde $D_{\theta_D}(G_{\theta_G}(I^{LR})$ je pravděpodobnost, že rekonstruovaný snímek je originální snímek s vysokým rozlišením. [28]

3.1.2 Architektura sítě

Jak již bylo naznačeno, použitá síť se skládá ze dvou částí: generativní sítě G_{θ_G} a diskriminativní sítě D_{θ_D}. Tyto sítě jsou učeny zároveň tak, že je řešen protikladný problém minimum-maximum: minimální chyba generativní sítě a maximální chyba diskriminativní.

V praxi to znamená, že generativní síť ze zadaného obrázku v nízkém rozlišení vytváří obrázek s vysokým rozlišením, ten by měl být co nejpodobnější originálu, proto její chyba musí být co nejmenší. Naproti tomu diskriminativní síť má za úkol rozhodovat, jestli obrázek vložený na její vstup je originální obrázek nebo jen zvětšená verze z výstupu generativní sítě. Čím lepší budou generované obrázky, tím více se bude diskriminativní síť plést, proto by měla mít co největší chybovost. [28]

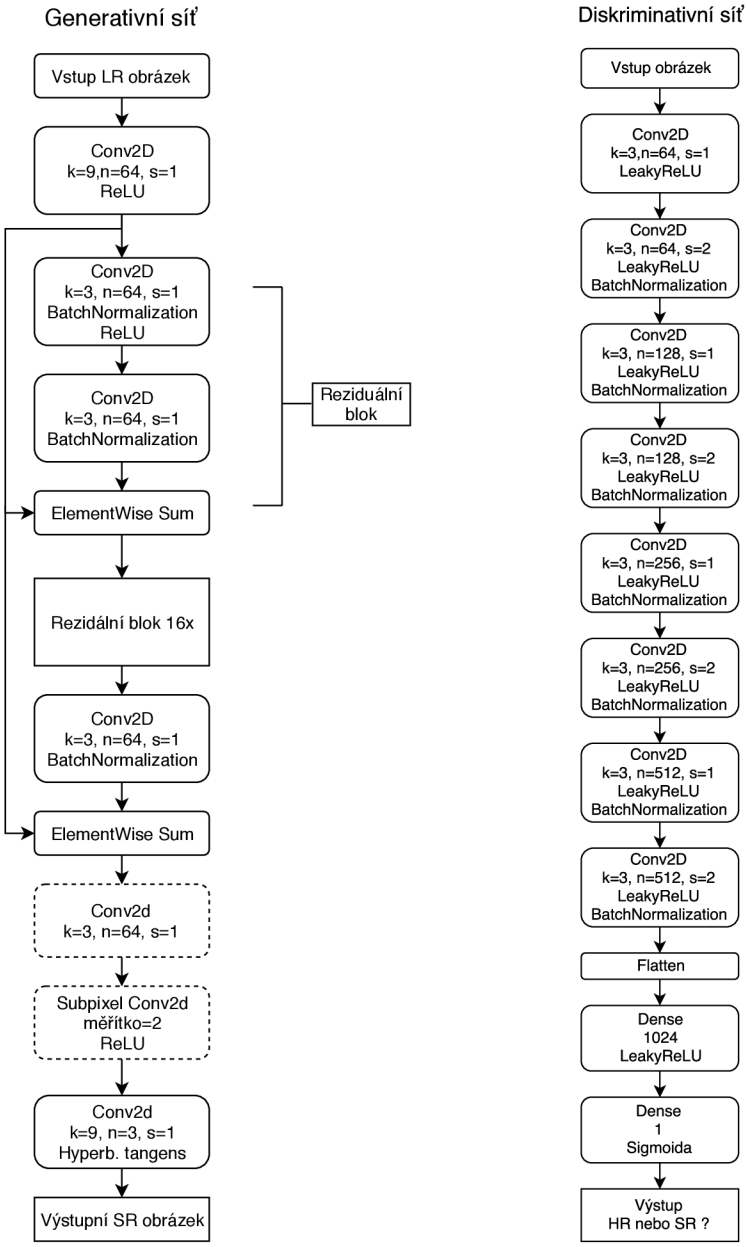
Jinými slovy generativní síť má za úkol svými výsledky přelstít síť diskriminativní, aby si myslela, že jde o skutečné obrázky. Tato síťová architektura byla poprvé představena v práci [34].

Architekturu obou těchto sítí můžete vidět i s parametry jednotlivých konvolučních vrstev na obrázku 3.1, kde k je velikost konvolučních jader, n je počet map příznaků a s je krok. Vlevo je síť generativní, vpravo diskriminativní. Tento model neuronové sítě byl natrénován se dvěma variantami nastavení:

- vstupem sítě je obrázek s nízkým rozlišením, který je na konci výpočtu zvětšen na dvojnásobek pomocí subpixelové konvoluční vrstvy (dále v práci označován jako SRGAN1),

- vstupem sítě je obrázek, který byl bikubicky interpolován na konečnou velikost a následujícím průchodem neuronovou sítí je pouze vylepšován (SRGAN2).

Ve schématu jsou vrstvy tvořící rozdíl mezi těmito modely označeny čárkovanou čarou.



Obr. 3.1: Architektura neuronové sítě SRGAN

Jádrem generativní sítě je proměnný počet reziduálních bloků, zde použito 16, v každém se nachází dvě konvoluční vrstvy se 64 filtry s jádry 3×3 a aktivační funkcí ReLU. Po těchto blocích následuje volitelná subpixelová konvoluční vrstva, která zvyšuje rozlišení obrázku na dvojnásobek. [28]

Diskriminační síť je tvořena osmi konvolučními vrstvami s LeakyReLU aktivační funkcí, s jádry 3×3 a s rostoucím počtem map příznaků od 64 do 512 po násobcích dvou. Po těchto blocích následují dvě plně propojené vrstvy (dense) a výstupem je pravděpodobnost, zda se jedná o přirozený nebo generovaný obraz. [28]

Celkem tedy obrázek od vstupu k výstupu projde přes 35 konvolučních vrstev (respektive 34 konvolučních a jednu subpixelovou konvoluční vrstvu) v generativní síti a přes 8 v síti diskriminační. Toto nastavení má za následek společně se složitou chybovou funkcí velkou výpočetní náročnost jak pro trénování tak pro následné vybavování.

Popis projektu

Pro trénování byl nastaven počet epoch na 2000 s koeficientem učení 10^{-4}, který je po polovině tréninkových epoch snížen na 10^{-5}. Pro minimalizaci chybové funkce je použit Adam optimizer [35] s parametrem $\beta = 0,9$, algoritmus založený na snižování gradientu.

Samotný projekt se skládá ze čtyř souborů: *main.py*, hlavní soubor načítající obrázky a spouštějící terénování nebo testování, *model.py* obsahující samotný model neuronové sítě, *config.py*, ve kterém je možné nastavit některé parametry sítě a *utils.py* obsahující metody pro práci s obrázky.

Před spuštěním je nutné nastavit cestu k trénovacím a testovacím obrázkům v souboru *config.py* a následně je možné skript spustit příkazem `python3 main.py`. Natrénovaný model je průběžně ukládán do složky checkpoints ve formátu *.npz*, což je soubor balíčku Numpy. Testování poté probíhá obdobně příkazem `python3 main.py --mode=evaluate` pro celou testovací sadu nebo je možné pomocí parametru `--file=obrázek` specifikovat jeden obrázek pro zvětšení.

3.2 DRCN

Metoda publikovaná v práci [29] v roce 2016 je zajímavá tím, že využívá rekurzivní konvoluční vrstvu, tzn. že na data je opakovaně aplikována stejná konvoluční vrstva se stejnými vahami. Pro práci s tímto modelem byl jako základ použit zdrojový kód dostupný zde: [36].

3.2.1 Architektura

Tato neuronová síť se skládá ze tří pomyslných částí:

- embedding sítě, která ze vstupního obrázku extrahuje příznakové mapy,
- inference (odvozovací) sítě, která pomocí rekurzivní vrstvy vytváří příznakové mapy odpovídající obrázku s vysokým rozlišením,

- rekonstrukční síť, která příznakové mapy převádí zpět do barevného prostoru vstupního obrázku.

Funkce celé sítě se dá zjednodušeně vyjádřit vzorcem:

$$\hat{\mathrm{y}} = f(\mathrm{x}), \tag{3.5}$$

kde x je vstupní obrázek a ŷ je odhad požadovaného výstupního obrázku y.
Funkci f je možné rozepsat do tří funkcí odpovídajících jednotlivým částem sítě popsaným výše:

$$f(\mathrm{x}) = f_3(f_2(f_1(\mathrm{x}))). \tag{3.6}$$

Embedding síť lze popsat pomocí vzorce:

$$H_{-1} = \max(0, W_{-1} * \mathrm{x} + b_{-1}), \tag{3.7}$$

$$H_0 = \max(0, W_0 * H_{-1} + b_0), \tag{3.8}$$

$$f_1(\mathrm{x}) = H_0, \tag{3.9}$$

kde x je vstupní obrázek, W_n a b_n jsou váhy a prahy, H_n je výstupní matice n-té vrstvy $max(0, .)$ odpovídá ReLU aktivační funkci.
Inference síť, která je složena z rekurzivních vrstev využívajících stejné matice vah a prahů W, b, lze popsat rovnicemi:

$$g(H) = \max(0, W * H + b), \tag{3.10}$$

což je funkce jedné rekurze.

$$H_d = g(H_{d-1}) = \max(0, W * H_{d-1} + b), \tag{3.11}$$

pro d = 1,.. D = počet rekurzí. Celková funkce inferenční vrstvy je tedy složená funkce g:

$$f_2(H) = g^D(H) \tag{3.12}$$

A konečně rekonstrukční síť, která je opakem embedding sítě, lze popsat vzorcem:

$$H_{D+1} = \max(0, W_{D+1} * H_D + b_{D+1}), \tag{3.13}$$

$$\hat{\mathrm{y}} = \max(0, W_{D+2} * H_{D+1} + b_{D+2}), \tag{3.14}$$

$$f_3(H) = \hat{\mathrm{y}}, \tag{3.15}$$

kde ŷ je odhad výstupního obrázku, W_n a b_n jsou váhy a prahy, H_n je výstupní matice n-té vrstvy $max(0, .)$ odpovídá ReLU aktivační funkci. [29]

Další důležitou částí architektury tohoto modelu je Recursive-Supervision (kontrola rekurzí). To znamená, že z výstupu každé rekurzivní vrstvy je pomocí stejné

rekonstrukční sítě vytvořen obrázek s vysokým rozlišením. Tímto je získáno D predikcí výstupního obrázku, které jsou následně zprůměrovány a tak je získán jediný výstupní obrázek. [29]

Díky této metodě je zjednodušen proces učení, kdy je jednak snížen počet parametrů k úpravě tím, že se využívá pořád stejná rekonstrukční síť a také zpětné šíření chyby přes dřívější rekurze je rychlejší. Další výhodou je možnost kontroly, zda je počet rekurzí optimální. Stačí zkontrolovat, jakou měrou přispívají k výslednému obrázku výstupy jednotlivých rekurzí. [29]

Skip-Connection (vynechání spojení) je další z nových metod v tomto modelu. Jedná se o přímé přivedení vstupního obrázku do rekonstrukční sítě s vynecháním embedding a inference sítě. Vstupní interpolovaný obrázek se od výstupního liší jen v detailech, proto je použit při každé rekonstrukci jako základ, ke kterému jsou detaily z rekurzivních vrstev přidány. [29]

Tyto dvě nové metody lze matematicky vyjádřit takto:

$$\hat{\mathrm{y}}_d = f_3(\mathrm{x}, g^d(f_1(\mathrm{x}))), \tag{3.16}$$

kde x je vstupní obrázek a $\hat{\mathrm{y}}_d$ je predikce d-té rekurze.

Za povšimnutí stojí, že funkce f_3 má dva vstupy – vstupní obrázek x a výstup rekurzivní vrstvy. Finální výstup je pak vyjádřen jako vážený průměr predikcí po jednotlivých rekurzivních vrstvách:

$$\hat{\mathrm{y}} = \sum_{d=1}^{D} w_d \cdot \hat{\mathrm{y}}_d, \tag{3.17}$$

kde w_d jsou váhy jednotlivých predikcí. [29]

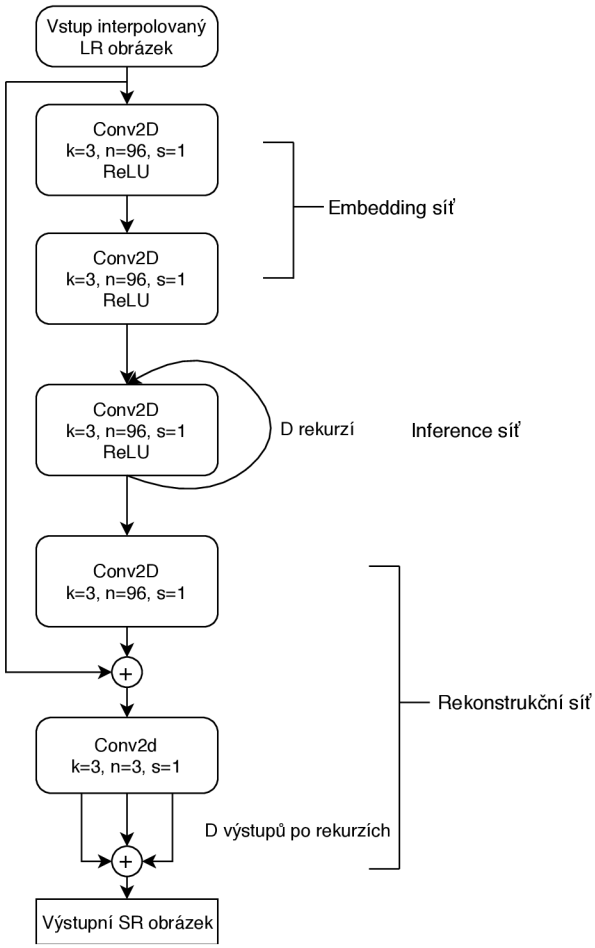
Blokové schéma této neuronové sítě můžete vidět na obrázku 3.2. U jednotlivých konvolučních vrstev jsou vypsány její parametry, kde k je velikost konvolučního jádra, n je počet filtrů a s je krok. Dalším parametrem sítě je D, tedy počet rekurzí. Tento model byl natrénován ve dvou variantách nastavení:

- počet rekurzí $D = 9$ (dále v textu nazýván DRCN1),
- počet rekurzí $D = 16$ (DRCN2).

3.2.2 Chybová funkce

Jako u většiny modelů řešících problém Super–Resolution je použita minimalizace MSE (Mean Square Error) jako chybová funkce. Zde je však rozšířena z jednoho objektu pro minimalizaci chyby na D+1 objektů: D výstupů rekurzí a finální výstup. Chyba výstupu po jednotlivých rekurzích je vyjádřena jako:

$$l_1(\theta) = \sum_{d=1}^{D} \sum_{i=1}^{N} \frac{1}{2DN} (\mathrm{y}^i - \hat{\mathrm{y}}_d^i)^2, \tag{3.18}$$



Obr. 3.2: Architektura neuronové sítě DRCN

kde θ je soubor parametrů, N je počet obrázků v trénovací sadě, y je skutečný obrázek s vysokým rozlišením a $\hat{y}_d$ je výstupní obrázek po d-té rekurzi.

Chyba finálního výstupu po váženém součtu mezivýsledků je:

$$l_2(\theta) = \sum_{i=1}^{N} \frac{1}{2N} (\mathrm{y}^i - \sum_{d=1}^{D} w_d \cdot \hat{\mathrm{y}}_d^i)^2. \tag{3.19}$$

Výsledná chybová funkce je poté definována jako:

$$L(\theta) = \alpha l_1(\theta) + (1 - \alpha) l_2(\theta), \tag{3.20}$$

kde α vyjadřuje míru jakou se na výsledku podílejí mezivýsledky po jednotlivých rekurzích. [29]

Popis projektu

Počáteční koeficient učení je nastaven na 0,001, který je snižován na polovinu vždy, když při čtyřech po sobě jdoucích krocích nedojde ke snížení chyby. Učení je ukončeno poté, co koeficient učení klesne pod 1e−5. Pro optimalizaci je použit opět

Adam optimizer [35]. Trénování probíhalo pouze na jasové složce obrázků, při následném zvyšování rozlišení je tento model použit pro všechny barevné složky. Díky tomu je urychlen proces učení. Natrénovaný model je poté uložen do složky models v nativním tensorflow formátu.

Tento projekt se skládá ze čtyř souborů: *main.py*, hlavní soubor odkud jsou spouštěny ostatní funkce a kde se také nachází měnitelné parametry sítě jako je počet rekurzí, *super_resolution.py*, kde se nachází samotná definice modelu a funkce pro trénování a testování, *super_resolution_utilty.py* ve kterém jsou pomocné funkce pro práci s obrázky a s modelem neuronové sítě a *test.py* pro testování natrénovaného modelu.

Po nastavení parametrů je trénování spuštěno pomocí `python3 main.py`. Trénovací sada je nastavena buď v v souboru *main.py* nebo je možné ji nastavit parametrem `-training_set adresář_s_daty` při spouštění skriptu. Testování probíhá příkazem `python3 test.py --file obrázek`.

3.3 ESPCN

Tento model publikovaný v roce 2016 v práci [30] si klade za cíl snížení výpočetní náročnosti zvyšování rozlišení obrázků při dosažení dobré kvality výstupu. Toho je zde dosaženo tím, že vstupem je obrázek s nízkým rozlišením, nikoli jeho bikubicky interpolovaná verze jak tomu bývá u většiny ostatních modelů. Bikubická interpolace do obrázku stejně nepřináší žádnou přídavnou informaci.

Díky tomu je většina výpočtů prováděna na čtvrtinovém počtu pixelů (v případě zvětšování 2 $\times$) a konečné zvětšení provádí předposlední vrstva celého modelu, konvoluční subpixelová vrstva. Operace zvětšení je pak také podrobena trénování a může se proto přizpůsobovat trénovacím datům na rozdíl od interpolace, která je pořád stejná.

3.3.1 Architektura sítě

Architektura této sítě je velmi jednoduchá. Jak již bylo zmíněno, vstupem je obrázek s nízkým rozlišením, který prochází přes dvě konvoluční vrstvy, jejichž funkce je popsána následovně:

$$f_1(x) = \max(0, W_1 * x + b_1), \tag{3.21}$$

$$f_2(x) = \max(0, W_2 * f_1(x) + b_2), \tag{3.22}$$

kde x je vstupní obrázek, W_n a b_n jsou trénovatelné váhy a prahy jednotlivých vrstev a max(0,.) je ReLU aktivační funkce. [30]

Hlavním prvkem této sítě pak je subpixelová konvoluční vrstva s parametrem scale (měřítko) rovným 2. Tato vrstva provádí klasickou operaci konvoluce s tím rozdílem, že krok (stride) má velikost $\frac{1}{2}$. Tato operace odpovídá operaci konvoluční vrstvy se vstupními daty dvojnásobné velikosti s krokem 1. Matematicky lze funkci této vrstvy vyjádřit takto:

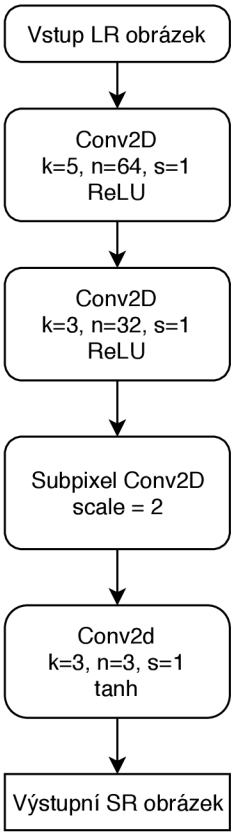
$$f_3(x) = PS(W_3 * f_2(x) + b_3), \tag{3.23}$$

kde PS je operace přeskládání prvků vstupního tensoru pro potřeby subpixelové konvoluce. Tato operace je definována jako:

$$PS(T)_{x,y,c} = T_{[x/2],[y/2],C\cdot 2\cdot mod(y,2)+C\cdot mode(x,2)+c}, \tag{3.24}$$

kde x, y a c jsou souřadnice ve vstupním tensoru. [30]

Posledním prvkem sítě je rekonstrukční konvoluční vrstva, která data převádí zpět do barevného prostoru. Celou architekturu sítě i s parametry jednotlivých vrstev můžete vidět na obrázku 3.3.



Obr. 3.3: Architektura neuronové sítě ESPCN

3.3.2 Chybová funkce

V této síti je jako parametr k minimalizaci použit pouze jednoduchý MSE (Mean Squared Error) vyjádřený vzorcem:

$$l(\theta) = \sum_{i=1}^{N} \frac{1}{2N} (\mathrm{y}^i - f^L(\mathrm{x}^i))^2, \tag{3.25}$$

kde θ je souhrn parametrů sítě, N je počet obrázků v trénovací sadě, x je vstupní LR obrázek a y je skutečný HR obrázek. [30]

Popis projektu

Tento model byl implementován autorem této práce dle informací v článku [30]. Parametry jednotlivých vrstev a chybová funkce byly převzaty, ostatní parametry byly nastaveny autorem. Jedná se o koeficient učení 10^{-4}, 2000 trénovacích kroků a Adam optimizer [35] pro optimalizaci vah.

Struktura projektu je obdobná jako u modelu SRGAN, viz 3.1.2, stejné je i ovládání: spuštění trénování příkazem `python3 main.py` a testování `python3 main.py --mode=evaluate`.

3.4 VDSR

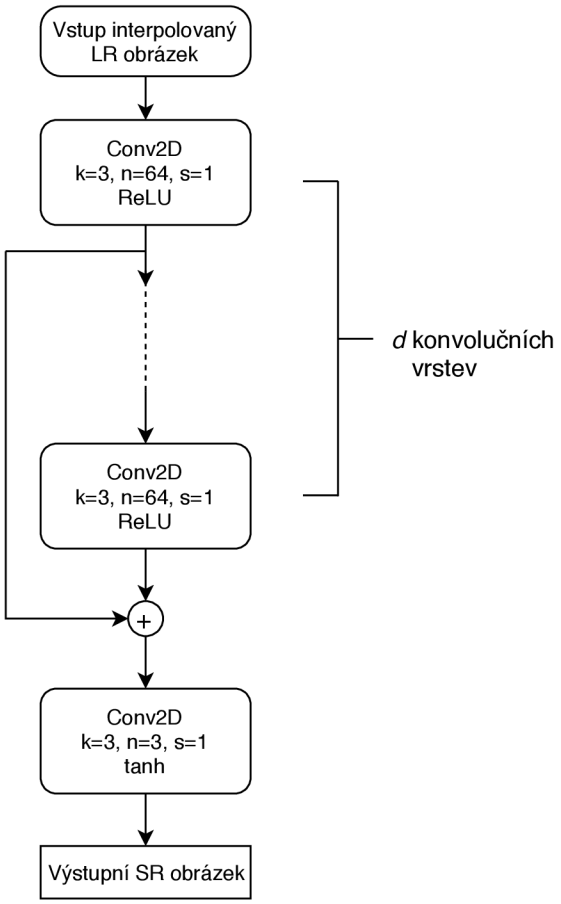
Hlavní myšlenkou této metody, publikované v práci [31] v roce 2016, je použití velmi hluboké neuronové sítě (až 20 vrstev). Návrh tohoto modelu byl založen na vylepšení vlastností jedné z prvních neuronových sítí pro zvyšování rozlišení obrázků SRCNN. Konkrétně se snaží ho vylepšit ve třech aspektech:

- kontext – využívá větší oblasti obrázků,
- konvergence – větší počáteční koeficient učení a zbytkové (residual) učení díky velké podobnosti vstupního LR a výstupního HR obrazu,
- násobek zvětšení – jediný natrénovaný model pro různé zvětšení. [31]

3.4.1 Architektura sítě

Vstupem této sítě je bikubicky interpolovaný LR obraz. Obraz poté prochází přes d stejných konvolučních vrstev se 64 filtry o velikosti 3×3. Tyto vrstvy se neučí celý výsledný obraz, ale pouze detaily, tedy rozdíl mezi výstupním HR a vstupním interpolovaným LR obrazem. Takto naučené rozdíly jsou poté přičteny ke vstupnímu obrazu a tím je získána výstupní predikce HR obrazu. Architekturu tohoto modelu můžete vidět na obrázku 3.4.

Prvním vylepšením tohoto modelu je využití širšího kontextu ve vstupním obrázku, neboli větší obl,ast působení jednotlivých filtrů. To znamená, že síť může



Obr. 3.4: Architektura neuronové sítě VDSR

použít více informací pro předpovězení detailů v obraze. Větší oblast působení lze obsáhnout použitím větších filtrů. To je ale výpočetně náročné, lepší je zřetězení více menších filtrů za sebou. První vrstva konvoluce pracuje v oblasti 3×3 pixelů, s každou další vrstvou se oblast působení rozšiřuje o 2. Pro síť s hloubkou d je výsledná oblast působení $(2d + 1) \times (2d + 1)$, pro hloubku 20 vrstev tedy 41×41 pixelů, oproti síti SRCNN, která pracuje pouze přes 13×13 pixelů. [31]

V původní práci [31] je pokusy prokázáno, že čím je síť hlubší, tím lepších výsledků dosahuje, obzvlášť pro větší zvětšení.

Dalším vylepšením je reziduální učení: síť předpovídá pouze rozdíl mezi vstupním a požadovaným výstupním obrázkem, tedy jeho vysokofrekvenční složky. Část obrazu je stejná jak ve vstupní tak ve výstupním obraze (neobsahuje vysokofrekvenční detaily) a po průchodu nelineární aktivační funkcí je odezva na tyto oblasti nulová. Díky tomu a také díky vysokému koeficientu učení je konvergence této sítě mnohem rychlejší a produkované výsledky jsou lepší. Výsledky experimentů s reziduálním učením a koeficientem učení jsou opět k dispozici v originální práci [31].

Posledním vylepšením je možnost zvětšování obrázků na různé násobky původní

velikosti s použitím jediného natrénovaného modelu. Toho je dosaženo jednoduše tím, že trénovací sada je rozšířena o vstupní obrázky zmenšené na různé velikosti a poté interpolované zpět na požadovanou velikost. Při testování se poté obrázek jen interpoluje na požadovanou velikost a síť je schopna ho zostřit, ať už byl násobek zvětšení jakýkoliv. [31]

3.4.2 Chybová funkce

Jako chybová funkce je zde opět použita minimalizace MSE, nepočítá se však přímo mezi výstupním a požadovaným výstupním obrázkem, ale kvůli reziduálnímu učení, kdy výstupem sítě je pouze rozdíl obrázků, MSE se počítá mezi součtem vstupního a výstupního obrázku a požadovaným výstupním obrázkem následovně:

$$l(\theta) = \sum_{i=1}^{N} \frac{1}{2N} (\mathrm{y}^i - (\mathrm{x}^i + \hat{\mathrm{y}}^i))^2, \tag{3.26}$$

kde θ je souhrn parametrů sítě, N je počet obrázků v trénovací sadě, x je vstupní LR obrázek, ŷ je výstup sítě a y je skutečný HR obrázek. [31]

Popis projektu

Tento model byl opět implementován autorem této práce. Základní architektura byla použita stejná jako v původním modelu, některé parametry však byly pozměněny. Počet vrstev byl zvolen $d = 10$ jako optimální pro zvětšování $2\times$ požadované v této práci. Také trénovací sada byla vytvořena pouze pro zvětšování $2\times$. Počáteční koeficient učení byl nastaven na 10^{-4} se snížením na desetinu 1000 krocích, učení bylo ukončeno 1600 krocích. Pro optimalizaci vah byl opět použit Adam optimizer [35]. Struktura a ovládání projektu je opět shodné s předchozím modelem.

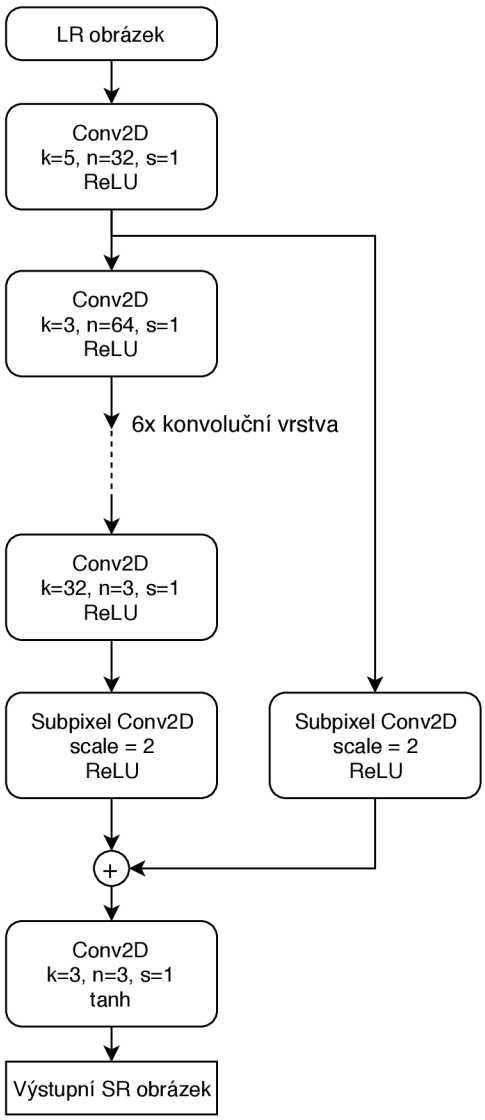
3.5 Vlastní návrh

Na základě znalostí získaných implementací předchozích modelů a dle jejich výkonnosti a efektivity byla navržena vlastní neuronová síť. Cílem návrhu bylo snížit výpočetní nároky modelu při zachování nebo zvýšení výkonnosti sítě.

Vstupem této sítě je obrázek s nízkým rozlišením, díky čemuž jsou výpočty rychlejší. Následuje konvoluční vrstva se 32 filtry 5×5 pro úvodní extrakci příznaků. Po ní následuje 6 stejných konvolučních vrstev inspirovaných sítí VDSR se 64 filtry velikosti 3×3 a jednou konvoluční vrstvou shodnou s první vrstvou modelu. Poté následuje zvýšení rozměru dat na dvojnásobek subpixelovou konvoluční vrstvou inspirovanou sítí ESPCN. K jejímu výstupu jsou přičteny další subpixelovou

vrstvou zvětšené mapy příznaků z první vrstvy modelu. Nakonec jsou data převedena zpět do barevného prostoru konvoluční vrstvou se 3 filtry velikosti 1. Schéma této architektury můžete vidět na obrázku 3.5.

Chybovou funkcí je opět MSE mezi výstupním a požadovaným výstupním obrázkem, obdobně jako u předchozích modelů. Je zde využit opět Adam optimizer [35]. Koeficient učení je na začátku nastaven na 10^{-2} snižující se na desetinu poprvé po 20 krocích, dále pak vždy po trojnásobku předchozího počtu kroků pro změnu, tedy po 60, 180 atd. Učení bylo ukončeno při velikosti koeficientu učení 10^{-6}. Struktura a ovládání projektu je shodná s předchozími modely.



Obr. 3.5: Architektura neuronové sítě dle vlastního návrhu

4 ŘEŠENÍ

Všechny použité modely jsou napsány v jazyce python s využitím knihovny TensorFlow popsané výše. Tyto modely byly poté natrénovány, některé ve více variantách nastavení parametrů, na nově vytvořené trénovací databázi snímků. Tato kapitola obsahuje informace nutné pro spuštění trénování a testování neuronových sítí, informace o jejich implementaci a popis trénovací databáze.

4.1 Příprava prostředí

Trénování a testování modelů proběhlo na serveru s procesorem Intel Xeon E5-2640, grafickou kartou GeForce GTX 1080 Ti a 64 GB operační paměti. Nainstalovaný zde byl operační systém Ubuntu 16.04. Následuje popis instalace potřebného softwaru a balíčků jazyka python.

Základním požadavkem pro spuštění modelů je knihovna TensorFlow, existující ve dvou verzích – *tensorflow*, s podporou výpočtu pouze na procesoru a *tensorflow-gpu*, která podporuje výpočty na grafických kartách Nvidia s podporou technologie CUDA, což výrazně urychluje výpočty. Při využití knihovny s podporou grafických karet je nutné nainstalovat CUDA® Toolkit ve verzi 9.0, aktuální ovladače grafické karty, cuDNN SDK verze 7.0 a knihovnu *libcupti-dev*. Zde uvedené verze softwaru jsou aktuální v době dokončení této práce, po vydání nové verze TensorFlow budou pravděpodobně zapotřebí také novější verze podpůrného softwaru a knihoven.

Další postup je stejný pro obě varianty. Nejprve je nutné nainstalovat Python verze 3 nebo vyšší. Následně je nainstalovaná samotná knihovna TensorFlow pomocí správce balíčků *pip* a to ve verzi buď s nebo bez podpory GPU. Podrobný postup instalace naleznete na stránkách projektu TensorFlow [27].

Skripty použité v této práci vyžadují pro své spuštění další přídavné balíčky, konkrétně se jedná o *scipy*, *numpy*, *matplotlib*, *easydict*, *scikit-image* a *Pillow* instalované opět přes správce balíčků *pip*.

4.2 Implementace modelů

Všechny výše popsané modely byly naprogramovány v jazyce Python s využitím knihovny pro strojové učení TensorFlow. U prvních dvou modelů byly využity veřejně dostupné zdrojové kódy, ostatní byly implementovány autorem. V této podkapitole bude uveden zjednodušený přehled základních kroků nutných k vytvoření modelu neuronové sítě, který je možno použít jako návod. Kompletní zdrojové kódy je možné nalézt na přiloženém CD.

Nejprve je nutné importovat základní knihovny:

```
import numpy as np
import tensorflow as tf
import tensorlayer as tl
```

Poté následuje načtení všech trénovacích obrázků do pole s využitím pomocné funkce `read_all_imgs`:

```
train_hr_imgs = read_all_imgs(train_hr_img_list,
    path=config.TRAIN.hr_img_path, n_threads=16)
```

Dalším krokem je definování vstupních a výstupních dat, zde dvou tensorů pro vstupní a výstupní obrázek. Tensor je základním datovým typem knihovny TensorFlow, jedná se o vícerozměrné pole, zde čtyřrozměrné, kde jednotlivé dimenze obsahují velikost dávky obrázků, sířku a výšku obrázku v pixelech a počet barevných kanálů. Dalším řádkem je zadefinován model `espcn` a na vstup je mu vložen tensor vstupních obrázků.

```
t_image = tf.placeholder('float32', [batch_size, 225, 225, 3],
    name='t_image_input_to_generator')
t_target_image = tf.placeholder('float32', [batch_size, 450, 450, 3],
    name='t_target_image')
net = espcn(t_image, is_train=True, reuse=False)
```

Přičemž vlastní model je definován jako funkce, která volá další funkce reprezentující jednotlivé vrstvy a předává vstupní vrstvě vstupní tensor a následujícím vrstvám vždy výstup předchozí vrstvy.

```
def espcn(t_image, is_train=False, reuse=False):
    w_init = tf.random_normal_initializer(stddev=0.02)
    b_init = None
    g_init = tf.random_normal_initializer(1., 0.02)
    size = t_image.get_shape().as_list()
    with tf.variable_scope("net", reuse=reuse) as vs:
        tl.layers.set_name_reuse(reuse)
        n = InputLayer(t_image, name='in')
        n = Conv2d(n, 64, (5, 5), (1, 1), act=tf.nn.relu,
            padding='SAME', W_init=w_init, name='n64s1/c')
        n = Conv2d(n, 32, (3, 3), (1, 1), act=tf.nn.relu,
            padding='SAME', W_init=w_init, b_init=b_init,
            name='n64s1/c/m')
        n = SubpixelConv2d(n, scale=2, n_out_channel=None,
            act=tf.nn.relu, name='pixelshufflerx2/1')
        n = Conv2d(n, 3, (1, 1), (1, 1), act=tf.nn.tanh,
            padding='SAME', W_init=w_init, name='out')
        return n
```

Poté je definovaná trénovací operace, zde minimalizace MSE (Mean Squared Error) pomocí optimizeru Adam:

```
mse_loss = tl.cost.mean_squared_error(net.outputs ,
    t_target_image, is_mean=True)
g_vars = tl.layers.get_variables_with_name('net', True, True)
g_optim = tf.train.AdamOptimizer(lr_v, beta1=beta1)
    .minimize(mse_loss, var_list=g_vars)
```

Následně je vytvořena Session, neboli prostředí ve kterém jsou vykonávány operace a jsou inicializovány proměnné:

```
sess = tf.Session(config=tf.ConfigProto(allow_soft_placement=True,
    log_device_placement=False))
tl.layers.initialize_global_variables(sess)
```

Poté už je jenom spuštěno učení. To probíhá průchodem dvěma cykly for: první pro opakování trénovacích kroků (epoch) a druhý pro průchod přes všechny trénovací obrázky. Následuje jen předzpracování dat a spuštění trénovací relace:

```
for epoch in range(0, n_epoch+1):
    for idx in range(0, len(train_hr_imgs), batch_size):
        b_imgs_450 = tl.prepro.threading_data(
            train_hr_imgs[idx : idx + batch_size],
            fn=crop_sub_imgs_fn, is_random=True)
        b_imgs_225 = tl.prepro.threading_data(b_imgs_450,
            fn=downsample_fn)
        errG, errA, _ = sess.run([g_loss, mse_loss, g_optim],
            {t_image: b_imgs_225, t_target_image: b_imgs_450})
```

Po stanoveném počtu kroků jsou aktuální natrénované váhy modelu ukládány do souboru `.npz`.

```
tl.files.save_npz(net.all_params, name=checkpoint_dir+
    '/model_epoch'+str(epoch)+'.npz', sess=sess)
```

Po natrénování je možné pro zadaný vstupní obrázek vygenerovat dvakrát větší podobu. Vstupní data jsou předzpracována, je vytvořena relace a jsou inicializovány parametry jak bylo uvedeno výše. Poté jsou do modelu načteny natrénované váhy:

```
tl.files.load_and_assign_npz(sess=sess, name=checkpoint_dir+
    '/model_epoch_2000.npz', network=net)
```

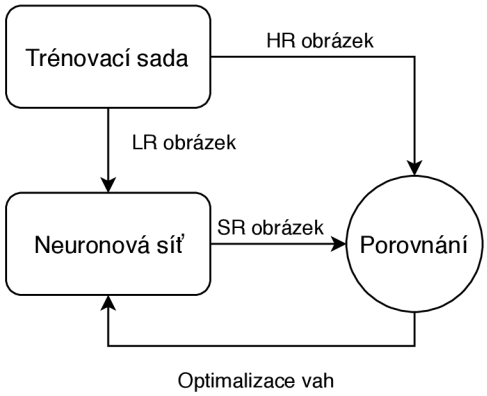
Poté je už jen spuštěno vybavování sítě a výstupní obrázek je uložen.

```
out = sess.run(net.outputs, {t_image: [valid_lr_img]})
tl.vis.save_image(out[0], save_dir+'/'+filename+'_valid_gen.png')
```

4.3 Trénování

Trénování je spolu s návrhem, implementací a testováním základním prvkem při práci na modelu jakékoli neuronové sítě. Trénováním se rozumí vkládání vstupů

a požadovaných výstupů do sítě tak, aby síť těmto datům přizpůsobila svoje vnitřní parametry (váhy). Blokové schéma pro trénování zde použitých modelů můžete vidět na obrázku 4.1. Zde jsou vstupní data ve formě obrázku s nízkým rozlišením (LR), výstupními daty je obrázek po operaci super-resolution (SR) a požadovaným výstupem je původní obrázek ve vysokém rozlišení (HR). Síť poté přizpůsobuje své parametry tak, aby minimalizovala rozdíly mezi SR a HR obrázky.



Obr. 4.1: Blokové schéma trénování modelu

Základním požadavkem úspěšného trénování je vhodně zvolená trénovací sada. Ta musí obsahovat kvalitní, pro dané použití relevantní a komplexní data v dostatečném množství. Při malé velikosti trénovací množiny síť není dostatečně komplexní a není schopna dobře reagovat na neznámá data. Velká velikost trénovací množiny zase prodlužuje trénování jak z pohledu výpočetní náročnosti tak i z pohledu dosažení konvergence.

4.3.1 Trénovací data

Cílem této práce je vytvoření metody pro zvýšení rozlišení obrazu osob, využitelné například pro bezpečnostní složky pro zlepšení záznamů z bezpečnostních kamer nebo jiných fotografií zájmových osob.

Z toho důvody jsou jako trénovací data použity snímky lidí v běžných situacích, ale i při pózování pro fotografa. Snímky jsou pořízeny na různých veřejných místech jako jsou nádraží nebo ulice, ale i z interiéru obytných budov a zábavních podniků. Ukázku trénovacích obrázků (respektive výřezy z nich, z důvodu dobré viditelnosti rozdílů v detailech) můžete vidět na obrázku 4.2, vlevo je obrázek s nízkým rozlišením (zde pouze zvětšen z důvodu srovnání), vpravo obrázek s vysokým rozlišením.

Zdrojem obrázků byla jednak databáze obrázků pro vědecké účely ImageNet [37], dále databáze PIPA (People in Photo Album), vytvořená v rámci práce [38] zabývající se detekcí obličejů a veřejná databáze obrázků a fotografií `pixabay.com`.

Obr. 4.2: Ukázka trénovacích obrázků

Tyto obrázky mají tedy společné to, že se na nich nacházejí jednotlivci nebo skupiny osob. Pro trénování neuronových sítí je potřeba mít obrázky o stejné velikosti. Na základě analýzy stažených obrázků byla zvolena velikost 450×450 pixelů. Při této velikosti jsou osoby na snímcích dostatečně viditelné a přitom si zachovávají detaily, jako jsou obličeje. Pro menší obrázky by se ztratily buď zmíněné detaily nebo lidské postavy jako celek. Při větší velikosti by bylo trénování výpočetně velmi náročné.

Pro zmenšení a ořezání obrázků byl vytvořen jednoduchý skript v jazyce Python, využívající knihovnu pro práci s obrázky Pillow [39]. Vzniklé obrázky byly poté podvzorkovány na poloviční velikost, tedy 225×225 pixelů. Tímto vznikly páry odpovídajících si obrázků s nízkým a vysokým rozlišením, sloužící jako vstup a požadovaný výstup pro trénování neuronové sítě. Vytvořená trénovací databáze obsahuje 1000 párů takovýchto obrázků.

5 VÝSLEDKY

Výsledkem této práce je celkem sedm natrénovaných modelů, jejichž parametry a architektury byly popsány výše. Po natrénování modelů neuronových sítí bylo provedeno testování na testovací sadě obrázků, která byla vytvořena zároveň s trénovací sadou. Testovací obrázky jsou stejného typu i velikosti jako trénovací, jsou na nich však nezávislé (nebyly použity pro trénování). Vlastní testovací sada obsahuje 200 párů obrázků.

Modely byly také otestovány na obvyklých testovacích datasetech Set5, Set14, B100 a Urban100 a porovnány s výsledky uváděnými autory původních modelů. Pro srovnání byly výsledky též srovnány s bikubickou interpolací.

Dalším výstupem této práce je aplikace s grafickým uživatelským rozhraním, která umožňuje jednoduché zvětšování obrázků s pomocí některého z natrénovaných modelů.

5.1 Srovnávací kritéria

Výsledky testování byly porovnány pomocí běžných objektivních ukazatelů kvality obrázků, konkrétně se jedná o MSE (Mean Squared Error), PSNR (Peak Signal to Noise Ratio) a SSIM (Structural Similarity). Porovnána byla také časová náročnost zpracování jednoho obrázku různými modely. V následující podkapitole je také vyjádřeno subjektivní hodnocení kvality dle pohledu autora práce.

MSE

Neboli střední kvadratická odchylka se vypočítá podle vztahu:

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2, \tag{5.1}$$

kde I je originální obrázek, K je právě hodnocený obrázek a m, n jsou rozměry obrázků.

Jedná se tedy o mocninu průměrného rozdílu mezi jednotlivými pixely porovnávaných obrázků. Čím nižší MSE, tím více jsou si porovnávané obrázky podobné, pro naprosto stejné obrázky je hodnota MSE nulová.

PSNR

Neboli odstup signálu od šumu. Měří se na logaritmické stupnici a je závislý na MSE podle vztahu:

$$PSNR = 10 \cdot \log \left(\frac{(2^n - 1)^2}{MSE} \right), \tag{5.2}$$

kde n bitová hloubka obrázků, obvykle 8 bitů na pixel a barevný kanál.

Čím jsou si porovnávané obrázky podobnější, tím vyšší je hodnota PSNR, pro naprosto stejné obrázky vznikne ve výrazu dělení nulou, v takovém případě je PSNR definováno na hodnotu 100. PSNR i MSE mají jednu nevýhodu a sice to, že pracují pouze s matematickou podobností obrázků a nezohledňují lidské vnímání obrazu a proto ne zcela koreluje se subjektivním hodnocením.

SSIM

Tato metoda pracuje se strukturálním rozložením měřených dat, zohledňuje tedy lidské vnímání obrazu. Vychází z předpokladu, že jas objektů je dán osvětlením, ale jejich struktura je na jasu nezávislá. SSIM index dosahuje hodnot v intervalu -1 až 1, vyšší hodnota znamená lepší kvalitu. Je počítán pouze pro jasové složky podle vztahu:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{5.3}$$

kde μ je vážený průměr a σ je vážená variance, resp. kovariance a c_1, c_2 jsou konstanty. Hodnoty $\mu_x, \sigma_x, \sigma_{xy}$ jsou vypočítány dle vztahů:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{5.4}$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}, \tag{5.5}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y), \tag{5.6}$$

kde N je počet pixelů v obraze, x_i, y_i jsou hodnoty pixelů a μ_x, μ_y jsou průměrné hodnoty pixelů. Hodnoty konstant c_1, c_2 jsou určeny vzorci:

$$c_1 = (k_1 L)^2, \tag{5.7}$$

$$c_2 = (k_2 L)^2, \tag{5.8}$$

kde konstatny $k_1 = 0,01$, $k_2 = 0,03$ a L je dynamický rozsah hodnot pixelů, pro 8 bitů na pixel tedy 255. [40]

Čas

Pro porovnání jednotlivých modelů byl také použit čas, který potřebují pro zpracování jednoho obrázku. Tímto testovacím obrázkem byla Lena o rozměrech 480×480 pixelů. Zde nezáleží na obsahu a rozměrech obrázku, důležité je použít stejný pro všechny metody.

Testování časové náročnosti proběhlo na běžném notebooku bez dedikované grafické karty s procesorem Intel Core i3 3110M. Při výpočtu na grafické kartě budou výpočty několikanásobně rychlejší, zde však více vynikne rozdílná výpočetní náročnost a zároveň je tím simulováno použití modelů na běžných pracovních stanicích dostupných běžnému uživateli.

5.2 Hodnocení

Po natrénování byly každým z modelů zvětšeny všechny obrázky z vlastní testovací sady, na které byly poté aplikovány výše popsané metody měření kvality. Jejich průměrné výsledky můžete vidět v tabulce 5.1. Nejlepší výsledky pro každou sledovanou statistiku jsou zobrazeny červeně, druhé nejlepší modře.

Tab. 5.1: Srovnání výsledků na vlastní testovací sadě

Model	MSE	PSNR	SSIM	Čas [s]
Bikubická	51,19	32,312	0,924	–
SRGAN1	90,27	30,634	0,8917	25,38
SRGAN2	47,29	32,732	0,9371	93,78
DRCN1	28,38	35,003	0,9517	129,99
DRCN2	28,41	35,000	0,9517	225,04
ESPCN	43,97	33,079	0,9424	0,69
VDSR	35,74	34,219	0,9517	26,74
Vlastní	48,8	32,269	0,9399	4,22

Jak je vidět nejlepších výsledků při použití kvalitativních metod dosahuje model DRCN v obou variantách nastavení, zvýšení počtu rekurzí z 9 na 16 nepřineslo téměř žádné navýšení kvality, spíše naopak, zatímco výpočetní náročnost vyjádřená časem výpočtu vzrostla téměř dvakrát, přičemž i jednodušší model je pomalejší nežli všechny ostatní modely.

Druhý v pořadí, co se týče kvality, je model VDSR. SSIM index má stejnou hodnotu jako pro modely DRCN, PSNR a MSE jsou jen o málo horší, přičemž časová náročnost je téměř pětkrát nižší než pro model DRCN1.

Co se týče výpočetní náročnosti tak nejlepších výsledků dosahuje model ESPCN s časem pouhých 0,69 s na zpracování obrázku při zachování dostatečně dobré kvality. Druhým modelem v žebříčku výpočetní náročnosti je model navržený autorem práce se 4,22 s na testovaný obrázek. Jeho výsledky jsou mírně horší než u předchozích modelů, nicméně stále ucházející.

Největším zklamáním je pak model SRGAN, který dle autorů dosahoval velmi dobrých výsledků. Varianta SRGAN1 pracující se vstupními obrázky v nízkém rozlišení a zvětšující je až na konci výpočtu subpixelovou konvoluční vrstvou dosahuje dokonce horších výsledků nežli bikubická interpolace jak podle objektivních metod tak podle subjektivního hodnocení.

(a) Bikubická 29,55/0,924 (b) SRGAN1 29,42/0,905 (c) SRGAN2 31,50/0,941

(d) DRCN1 32,88/0,957 (e) DRCN2 32,90/0,957 (f) ESPCN 31,84/0,949

(g) VDSR 32,88/0,958 (h) Vlastní 31,67/0,945 (i) Originál PSNR/SSIM

Obr. 5.1: Ukázka výsledných obrázků

Upravená varianta SRGAN2 pracující s interpolovanými vstupními obrázky za cenu vyšší výpočetní náročnosti dosahuje lepších výsledků, ale stále nedostatečných

v porovnání s ostatními modely. Původní model SRGAN byl navržen pro zvětšování čtyřikrát, zde byl upraven pouze pro dvojnásobné zvětšování, to by však nemělo takto zhoršit jeho výkonnost. Na vině bude spíše nedostatečně rozsáhlá trénovací databáze, kdy autoři původního modelu použili 350 tisíc náhodně vybraných obrázků. Nicméně ostatním modelům takto redukovaná trénovací sada nijak neublížila.

(a) Bikubická 29,02/0,917 (b) SRGAN1 28,46/0,883 (c) SRGAN2 30,61/0,932

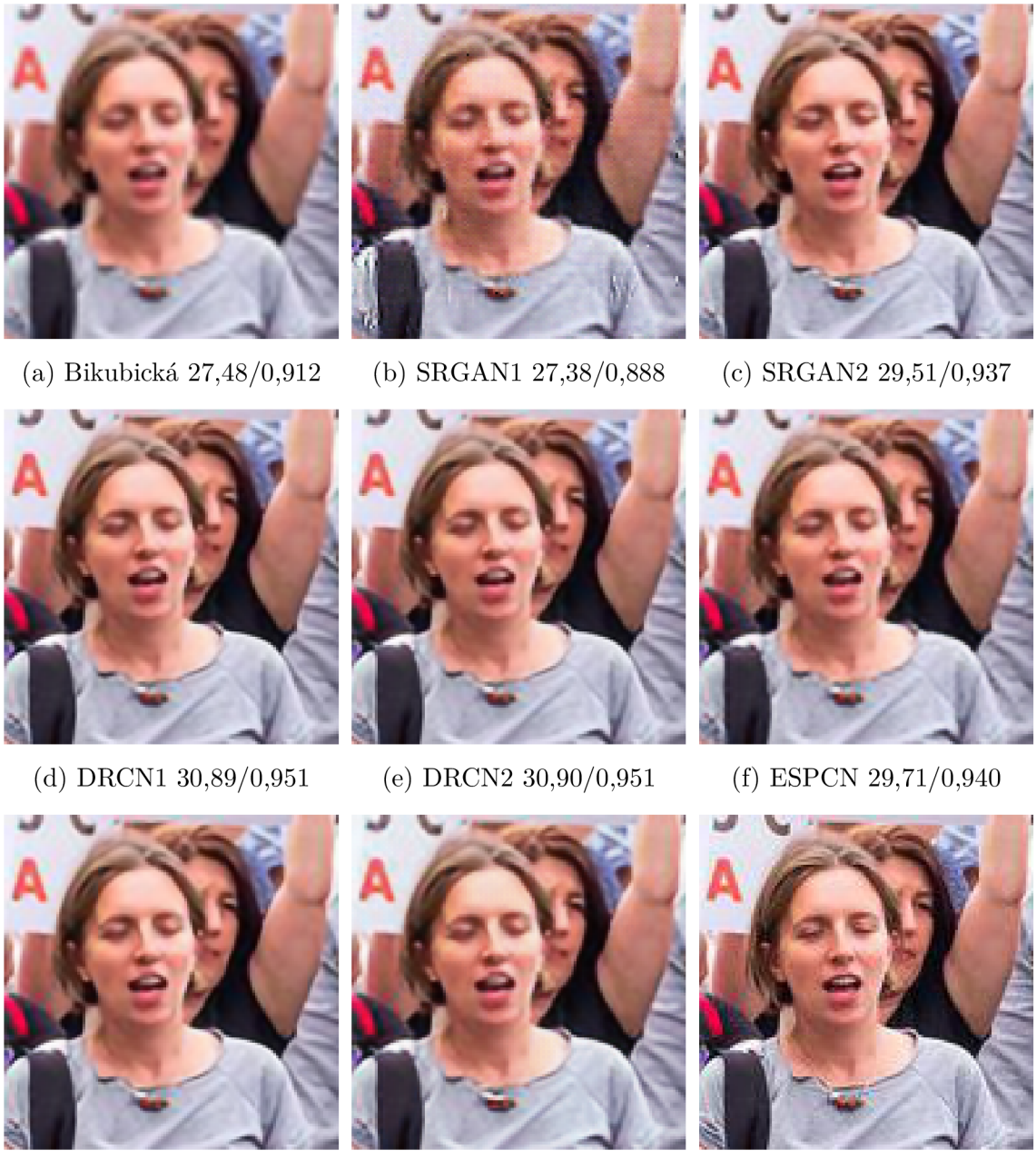
(d) DRCN1 31,81/0,950 (e) DRCN2 31,76/0,950 (f) ESPCN 31,00/0,940

(g) VDSR 31,86/0,951 (h) Vlastní 30,83/0,931 (i) Originál PSNR/SSIM

Obr. 5.2: Ukázka výsledných obrázků

Když se na výsledné obrázky podíváme subjektivním pohledem člověka, oba modely DRCN i model VDSR produkují téměř totožné obrázky, u některých se však

zdá, že výstupy modelu VDSR vypadají o málo lépe. Na druhém místě subjektivního hodnocení jsou modely ESPCN a vlastní model autora, jejichž výsledky jsou na pohled téměř stejné. Nejhorší je pak model SRGAN1, který v obrázcích vytváří různé drobné artefakty složené z černých a bílých pixelů.



(a) Bikubická 27,48/0,912 (b) SRGAN1 27,38/0,888 (c) SRGAN2 29,51/0,937

(d) DRCN1 30,89/0,951 (e) DRCN2 30,90/0,951 (f) ESPCN 29,71/0,940

(g) VDSR 30,83/0,950 (h) Vlastní 29,72/0,939 (i) Originál PSNR/SSIM

Obr. 5.3: Ukázka výsledných obrázků

Ukázku výřezů z obrázků generovaných všemi testovanými modely můžete spolu s bikubicky interpolovaným obrázkem a originálem vidět na obrázcích 5.1, 5.2, 5.3,

v popiscích těchto obrázků jsou uvedeny i výsledky hodnocení kvality pro tyto konkrétní obrázky.

Dále bylo provedeno testování na výzkumníky běžně používaných datasetech Set5, Set14, BSD100 a Urban100. Tyto sady obrázků se skládají z několika fotografií lidí, ale hlavně z fotografií přírody (BSD100) nebo budov a měst (Urban100). Výsledky natrénovaných modelů můžete vidět v tabulkách 5.2 a 5.3 včetně výsledků uváděných autory originálních modelů. Nejlepší výsledky pro každý dataset a každou srovnávací metodu jsou zvýrazněny červeně, zvlášť pro nově natrénované modely a zvlášť pro originální modely. Ukázku generovaných obrázků ze sady Set5 můžete vidět na obrázku 5.4.

Tab. 5.2: Srovnání výsledků na testovacích sadách Set5 a Set14

	Set5		Set14	
Model	PSNR	SSIM	PSNR	SSIM
Bikubická	29,141	0,8410	26,314	0,7610
SRGAN1	29,279	0,8326	26,152	0,7672
SRGAN2	30,000	0,8527	26,677	0,7928
DRCN1	30,149	0,8690	27,227	0,7958
DRCN2	30,126	0,8686	27,218	0,7957
ESPCN	30,213	0,8562	26,664	0,7990
VDSR	30,127	0,8661	26,537	0,7998
Vlastní	29,450	0,8366	26,170	0,7953
Původní modely				
SRCNN [10]	36,66	0,9542	32,45	0,9067
DRCN [29]	37,63	0,9588	33,04	0,9118
VDSR [31]	37,53	0,9587	33,03	0,9124

Jak je vidět, nově natrénované modely nedosahují zdaleka takových výsledků jako modely původní. Toto má jednoduché vysvětlení: nově trénované modely byly trénovány na specifických datech – obrázcích lidí, na nichž dosahují dobrých výsledků, zatímco originální modely byly trénovány na obecných datech, nejčastěji náhodně vybraných a použitých ve velkém množství (až stovky tisíc oproti tisíci u nově trénovaných).

Nicméně ani výsledky nově trénovaných modelů nejsou na těchto obecných datech špatné. Objektivní hodnocení nevypadá moc dobře, při subjektivním pohledu jsou obrázky sice rozostřené, ale pořád kvalitnější než bikubická interpolace. Stále zde platí, že nejlepších výsledků dosahují modely DRCN a VDSR.

Tab. 5.3: Srovnání výsledků na testovacích sadách B100 a Urban100

	B100		Urban100	
Model	PSNR	SSIM	PSNR	SSIM
Bikubická	26,485	0,7360	23,291	0,7349
SRGAN1	24,831	0,7490	23,584	0,7476
SRGAN2	24,777	0,7658	24,122	0,7850
DRCN1	27,437	0,7807	24,405	0,7905
DRCN2	27,431	0,7805	24,398	0,7901
ESPCN	24,800	0,7706	24,219	0,7877
VDSR	24,759	0,7741	24,255	0,7940
Vlastní	24,967	0,7706	24,125	0,7881
Původní modely				
DRCN [29]	31,85	0,8942	30,75	0,9133
VDSR [31]	31,9	0,896	30,76	0,914

Shrnutí výsledků

Jak bylo uvedeno výše, některé z nově natrénovaných modelů dosahují dobrých výsledků, jak po stránce objektivního tak po stránce subjektivního hodnocení, na specifických datech – nově vytvořené databázi obrázků lidí, což je v pořádku, protože to bylo cílem této práce. Při použití na obecných datech reprezentovaných testovacími sadami Set5, Set14, BSD100 a Urban100 jsou výsledky o poznání horší, hlavně po stránce objektivních ukazatelů. Subjektivní dojem je vcelku dobrý, zcela jistě lepší než u bikubické interpolace, ale z porovnání s výsledky nejnovějších modelů vycházejí zde natrénované modely jednoznačně hůře. Nejnovější modely těží z velmi rozsáhlých trénovacích databází obrázků všeobecného charakteru.

Dalšího vylepšení generovaných obrázků by bylo možné dosáhnout rozšířením trénovací sady, případně dalšími experimenty s nastavením sítě, jako je třeba velikost a počet filtrů v jednotlivých vrstvách, koeficient učení, parametry optimizeru nebo použití komplexnější chybové funkce. Tyto další vylepšení bohužel nebylo možné v rámci diplomové práce uskutečnit z důvodu časové a výpočetní náročnosti – rozšíření trénovací sady úměrně prodlouží dobu trénování, stejně tak opakované trénování pro různé nastavení sítě stojí další čas. Nicméně bylo prokázáno, že některé modely dosahuje dobrých výsledků a jsou bez problémů použitelné pro zvětšování obrázků v praxi, další prací však může být jejich výkonnost ještě vylepšena.

(a) Bikubická 25,85/0,807 (b) SRGAN1 26,31/0,808 (c) SRGAN2 27,27/0,847

(d) DRCN1 27,30/0,853 (e) DRCN2 27,27/0,853 (f) ESPCN 27,71/0,859

(g) VDSR 27,39/0,855 (h) Vlastní 27,40/0,853 (i) Originál PSNR/SSIM

Obr. 5.4: Ukázka výsledných obrázků z datasetu Set14

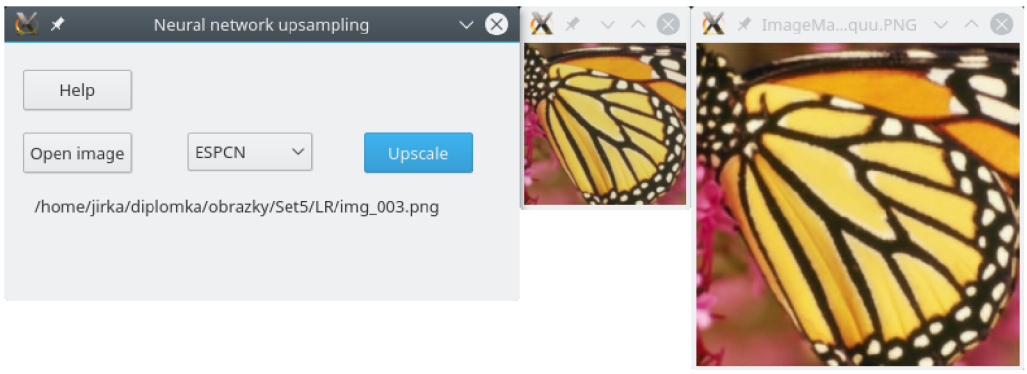
5.3 Aplikace s grafickým rozhraním

Skripty pro generování obrázků pomocí natrénovaných neuronových sítí byly opatřeny grafickým rozhraním, aby bylo možné tyto modely jednoduše testovat. Zároveň je tímto umožněno tyto metody, které jsou lepší než interpolační metody dostupné v běžných grafických editorech, využívat i uživatelům bez znalosti práce s příkazovým řádkem.

Tato aplikace je napsaná opět v jazyce Python s využitím knihovny *PyQt5*, instalované přes správce balíčků *pip*. Tuto aplikaci můžete nalézt na přiloženém CD i se skripty pro Windows a Linux, které zajistí instalaci balíčků potřebných pro spuštění aplikace. Těmito skripty je nainstalován také balíček *tensorflow* ve verzi bez podpory grafické karty. Verze s touto podporou je složitější na instalaci a vyžaduje konkrétní verze dalšího softwaru, zkušenější uživatelé si ji mohou nainstalovat podle návodu na stránkách projektu TensorFlow [27].

Použití aplikace je velmi jednoduché: spouští se zavoláním skriptu gui.py pomocí Pythonu nebo spuštěním skriptu *run.bat* (případně *run.sh*). Po výběru jednoho nebo více obrázků v dialogu po kliknutí na tlačítko Open image je možné z vysouvací nabídky zvolit jeden z natrénovaných modelů. Pod označením SRGAN se nachází model SRGAN2, protože model SRGAN1 nedosahoval dobrých výsledků a po výběru modelu DRCN je použit model DRCN1, protože DRCN2 dosahoval téměř stejných výsledků, ale za téměř dvojnásobný čas. Ostatní označení odpovídají modelům popsaným v předcházející kapitole.

Stisknutím tlačítka Upscale dojde ke spuštění výpočtu pomocí zvoleného modelu. Po skončení operace (která může v závislosti na zvoleném modelu a velikosti obrázku trvat i několik minut) je obrázek zobrazen a uložen do složky *output*. V případě dávkového zpracování více obrázků je zobrazen pouze poslední z nich, všechny jsou však uloženy do složky *output*. Po kliknutí na tlačítko Help je zobrazena jednoduchá nápověda k používání aplikace obsahující jednoduchý popis modelů sloužící jako možné kritérium pro výběr. Vytvořené grafické rozhraní můžete vidět na obrázku 5.5.



Obr. 5.5: Aplikace pro zvyšování rozlišení

Z pohledu programování je aplikace také jednoduchá: obsahuje několik funkcí pro obsluhu tlačítek a jednu funkci pro spuštění TensorFlow modelu. Zde jsou na základě vybraného modelu načteny jeho natrénované váhy ze složky *models* a je spuštěno vybavování, po jehož skončení je obrázek uložen a zobrazen. Práce s neuronovými sítěmi je popsána výše v části 4.2, proto zde nebude dále rozebírána.

6 ZÁVĚR

Cílem této práce bylo navrhnout a natrénovat několik modelů neuronové sítě, která by zvyšovala rozlišení obrázků na dvojnásobek beze ztráty kvality. Po seznámení se s problematikou zvyšování rozlišení obrázků (kapitola 1) a neuronových sítí se zaměřením na konvoluční sítě (kapitola 2), byly vybrány čtyři již existující modely slibující dobré výsledky. Jedná se o modely SRGAN, DRCN, ESPCN a VDSR, na jejichž základě byla navržen vlastní model. Všechny tyto modely (některé ve více variantách nastavení) byly implementovány a natrénovány na nově vytvořené databázi fotografií lidí, čítající 1000 obrázků pro trénování a dalších 200 pro testování.

Architektura těchto sítí s použitými parametry je popsána v kapitole 3. Informace o implementaci a testování nabízí následující kapitola 4. Natrénované modely byly poté otestovány na vytvořené sadě fotografií lidí a také na obvyklých testovacích datasetech Set5, Set14, BSD100 a Urban100. Výsledky těchto srovnání jsou k dispozici v poslední kapitole 5.

Z výsledků testování na sadě fotografií lidí je patrné, že některé modely dosahují dobrých výsledků. Mezi nejlepší patří modely DRCN a VDSR, jejichž výsledky jsou dobré z pohledu objektivního i subjektivního hodnocení kvality. S mírně horší kvalitou následují modely ESPCN a model navržený autorem práce, které zase mají několikanásobně menší výpočetní náročnost, hlavně pak model ESPCN, který může pracovat téměř v reálném čase a najde tak uplatnění např. zvyšování rozlišení video sekvencí.

Testování na obecných obrázcích dopadlo o poznání hůře. Důvodem je specificky vybraná trénovací databáze, která však byla cílem této práce. Nicméně i na těchto obecných datech dosahují modely uspokojivých výsledků, výrazně lepších něž metody dostupné v běžných grafických editorech a proto je možné je v praxi používat, minimálně na amatérské úrovni. Z toho důvodu bylo vytvořeno jednoduché grafické rozhraní umožňující pohodlné používání těchto modelů i uživatelům bez hlubších znalostí práce s příkazovým řádkem a jazykem Python.

Navržené modely je možné dále vylepšit, zejména rozšířením trénovací sady, běžné je trénování na stovkách tisíc obrázků. Také je možné provádět další experimenty s architekturami modelů a nastavením jejich parametrů.

LITERATURA

[1] RAJMIC, Pavel. Základy počítačové sazby a grafiky. Brno: Vysoké učení technické v Brně, 2012. ISBN 978-80-214-4451-5.

[2] ZHANG, Yunfeng, Qinglan FAN, Fangxun BAO, Yifang LIU a Caiming ZHANG. Single-Image Super-Resolution Based on Rational Fractal Interpolation. *IEEE Transactions on Image Processing* [online]. 2018, **27**(8), 3782-3797 [cit. 2018-05-17]. DOI: 10.1109/TIP.2018.2826139. ISSN 1057-7149. Dostupné z: `https://ieeexplore.ieee.org/document/8336891/`

[3] JIANCHAO YANG, John WRIGHT, Thomas S HUANG a YI MA. Image Super-Resolution Via Sparse Representation. IEEE Transactions on Image Processing [online]. 2010, 19(11), 2861-2873 [cit. 2017-12-05]. DOI: 10.1109/TIP.2010.2050625. ISSN 1057-7149. Dostupné z: `http://ieeexplore.ieee.org/document/5466111/`

[4] HRBÁČEK, R.; RAJMIC, P.; VESELÝ, V.; ŠPIŘÍK, J. Řídké reprezentace signálů: Úvod do problematiky. Elektrorevue – Internetový časopis (`http://www.elektrorevue.cz`), 2011, roč. 2011, č. 50, s. 1-10. ISSN: 1213- 1539.

[5] HONG CHANG, DIT-YAN YEUNG a YIMIN XIONG. Super-resolution through neighbor embedding. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004 [online]. IEEE, 2004, s. 275-282 [cit. 2017-12-14]. DOI: 10.1109/CVPR.2004.1315043. ISBN 0-7695-2158-4. Dostupné z: `http://ieeexplore.ieee.org/document/1315043/`

[6] BEVILACQUA, Marco, Aline ROUMY, Christine GUILLEMOT a Marie-line Alberi MOREL. Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. In: Procedings of the British Machine Vision Conference 2012 [online]. British Machine Vision Association, 2012, 135.1-135.10 [cit. 2017-12-14]. DOI: 10.5244/C.26.135. ISBN 1-901725-46-4. Dostupné z: `http://www.bmva.org/bmvc/2012/BMVC/paper135`/index.html

[7] TIMOFTE, Radu, Vincent DE a Luc Van GOOL. Anchored Neighborhood Regression for Fast Example-Based Super-Resolution. In: 2013 IEEE International Conference on Computer Vision [online]. IEEE, 2013, s. 1920-1927 [cit. 2017-12-14]. DOI: 10.1109/ICCV.2013.241. ISBN 978-1-4799-2840-8. Dostupné z: `http://ieeexplore.ieee.org/document/6751349/`

[8] ROMANO, Yaniv, John ISIDORO a Peyman MILANFAR. RAISR: Rapid and Accurate Image Super Resolution. IEEE Transactions on Computational Imaging [online]. 2017, 3(1), 110-125 [cit. 2017-12-13]. DOI: 10.1109/TCI.2016.2629284. ISSN 2333-9403. Dostupné z: http://ieeexplore.ieee.org/document/7744595/

[9] Saving you bandwidth through machine learning. Blog.google.com [online]. [cit. 2017-12-13]. Dostupné z: https://blog.google/products/google-plus/saving-you-bandwidth-through-machine-learning/

[10] Chao Dong, Chen Change Loy, Kaiming He, Xiaoou Tang. Image Super-Resolution Using Deep Convolutional Networks. CoRR, abs/1501.00092, 2015, Dostupné z:http://arxiv.org/abs/1501.00092.

[11] YAMANAKA, Jin, Shigesumi KUWASHIMA a Takio KURITA. Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network. LIU, Derong, Shengli XIE, Yuanqing LI, Dongbin ZHAO a El-Sayed M. EL-ALFY, ed. *Neural Information Processing* [online]. Cham: Springer International Publishing, 2017, 2017-10-26, s. 217-225 [cit. 2018-05-17]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-70096-0_23. ISBN 978-3-319-70095-3. Dostupné z: http://link.springer.com/10.1007/978-3-319-70096-0_23

[12] WANG, Qiang, Huijie FAN, Yang CONG a Yandong TANG. Large receptive field convolutional neural network for image super-resolution. In: *2017 IEEE International Conference on Image Processing (ICIP)* [online]. IEEE, 2017, 2017, s. 958-962 [cit. 2018-05-18]. DOI: 10.1109/ICIP.2017.8296423. ISBN 978-1-5090-2175-8. Dostupné z: http://ieeexplore.ieee.org/document/8296423/

[13] CHU, Jinghui, Jiaqi ZHANG, Wei LU a Xiangdong HUANG. A Novel Multi-Connected Convolutional Network for Super-Resolution. *IEEE Signal Processing Letters* [online]. , 1-1 [cit. 2018-05-18]. DOI: 10.1109/LSP.2018.2820057. ISSN 1070-9908. Dostupné z: http://ieeexplore.ieee.org/document/8326520/

[14] LIU, Yang, Qingchao CHEN a Ian WASSELL. Deep network for image super-resolution with a dictionary learning layer. In: *2017 IEEE International Conference on Image Processing (ICIP)* [online]. IEEE, 2017, 2017, s. 967-971 [cit. 2018-05-18]. DOI: 10.1109/ICIP.2017.8296425. ISBN 978-1-5090-2175-8. Dostupné z: http://ieeexplore.ieee.org/document/8296425/

[15] MEI, Shaohui, Xin YUAN, Jingyu JI, Shuai WAN, Junhui HOU a Qian DU. Hyperspectral image super-resolution via convolutional neural network.

In: *2017 IEEE International Conference on Image Processing (ICIP)* [online]. IEEE, 2017, 2017, s. 4297-4301 [cit. 2018-05-18]. DOI: 10.1109/ICIP.2017.8297093. ISBN 978-1-5090-2175-8. Dostupné z: http://ieeexplore.ieee.org/document/8297093/

[16] SUN, Dong, Teng LI, Qingwei GAO a Yixiang LU. A novel single-image super-resolution algorithm based on self-similarity in wavelet domain. In: *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)* [online]. IEEE, 2017, 2017, s. 639-644 [cit. 2018-05-17]. DOI: 10.1109/RCAR.2017.8311935. ISBN 978-1-5386-2035-9. Dostupné z: http://ieeexplore.ieee.org/document/8311935/

[17] VOLNÁ, Eva. *Neuronové sítě 1*. Ostrava: Ostravská univerzita, 2002.

[18] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. Second. Upper-Daddle River, New Jersey 07458: Williams Publishing House, 2006. ISBN 0-13-273350-1

[19] KLINE, Douglas M. a Victor L. BERARDI. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing and Applications* [online]. 2005, 14(4), 310-318 [cit. 2017-12-12]. DOI: 10.1007/s00521-005-0467-y. ISSN 0941-0643. Dostupné z: http://link.springer.com/10.1007/s00521-005-0467-y

[20] SMÉKAL, Zdeněk. *Analýza signálů a soustav - BASS*. Brno: Vysoké učení technické v Brně, 2012. ISBN 978-80-214-4453-9.

[21] Convolutional neural network. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-12]. Dostupné z: https://en.wikipedia.org/wiki/Convolutional_neural_network

[22] ZEILER, M. Rob FERGUS. ZEILER, M. Visualizing and Understanding Convolutional Networks. New York University, USA: Dept. of Computer Science.

[23] Shi Wenzhe, Caballero Jose, Theis Lucas, Huszar Ferenc, Aitken Andrew, Ledig Christian, Wang Zehan. (2016). Is the deconvolution layer the same as a convolutional layer?.

[24] Conv_arithmetic. Github [online]. [cit. 2017-12-12]. Dostupné z: https://github.com/vdumoulin/conv_arithmetic

[25] DUMOULIN, Vincent; VISIN, Francesco. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285, 2016.

[26] Keras: The Python Deep Learning library. [online]. [cit. 2017-12-13]. Dostupné z: https://keras.io/

[27] TensorFlow: An open-source software library for Machine Intelligence. [online]. [cit. 2017-12-09]. Dostupné z: https://www.tensorflow.org/

[28] LEDIG, Christian, Lucas THEIS, Ferenc HUSZAR, et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2017, 2017, s. 105-114 [cit. 2018-05-18]. DOI: 10.1109/CVPR.2017.19. ISBN 978-1-5386-0457-1. Dostupné z: http://ieeexplore.ieee.org/document/8099502/

[29] KIM, Jiwon, Jung Kwon LEE a Kyoung Mu LEE. Deeply-Recursive Convolutional Network for Image Super-Resolution. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2016, s. 1637-1645 [cit. 2017-12-14]. DOI: 10.1109/CVPR.2016.181. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780550/

[30] SHI, Wenzhe, Jose CABALLERO, Ferenc HUSZAR, Johannes TOTZ, Andrew P. AITKEN, Rob BISHOP, Daniel RUECKERT a Zehan WANG. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2016, s. 1874-1883 [cit. 2017-12-14]. DOI: 10.1109/CVPR.2016.207. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780576/

[31] KIM, Jiwon, Jung Kwon LEE a Kyoung Mu LEE. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2016, 2016, s. 1646-1654 [cit. 2018-05-17]. DOI: 10.1109/CVPR.2016.182. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780551/

[32] SRGAN. *Github* [online]. [cit. 2017-12-09]. Dostupné z: https://github.com/zsdonghao/SRGAN

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations (ICLR), 2015

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014.

[35] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), 2015. 6

[36] deeply-recursive-cnn-tf. *Github*: [online]. [cit. 2018-04-30]. Dostupné z: `https://github.com/jiny2001/deeply-recursive-cnn-tf`

[37] ImageNet [online]. [cit. 2017-12-09]. Dostupné z: `http://www.image-net.org/`

[38] Zhang, N., Paluri, M., Taigman, Y., Fergus, R., Bourdev, L. (2015). Beyond frontal faces: Improving Person Recognition using multiple cues. 4804-4813. 10.1109/CVPR.2015.7299113.

[39] *Pillow: The friendly PIL fork* [online]. [cit. 2018-05-08]. Dostupné z: `http://python-pillow.org/`

[40] ČÍKA, Petr. *Multimédia* [online]. Brno: Vysoké učení technické v Brně, 2014 [cit. 2018-05-08].

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

HR	High Resolution – Vysoké rozlišení
LR	Low Resolution – Nízké rozlišení
SISR	Single Image Super Resolution – Zvýšení rozlišení jednoho obrázku
RAISR	Rapid and Accurate Image Super Resolution
SRCNN	Super-Resolution Convolutional Neural Network
DCSCN	Deep CNN with Residual Net, Skip Connection and Network in Network
LRFCNN	Large Receptive Field Net
ReLU	Rectified Linear Unit
SRGAN	Super Resolution Generative Adversarial Network
SRGAN1	SRGAN, varianta nastavení parametrů 1
SRGAN2	SRGAN, varianta nastavení parametrů 2
DRCN	Deeply–Recursional Neural Network for Super Resolution
DRCN1	DRCN, varianta nastavení parametrů 1
DRCN2	DRCN, varianta nastavení parametrů 2
ESPCN	Efficient Subpixel Convolutional Network
VDSR	Very Deep Super Resolution
MSE	Mean Squared Error – Střední kvadratická chyba
PSNR	Peak Signal to Noise Ratio – Špičkový odstup signálu šumu
SSIM	Structural Similarity – Strukturální podobnost