UNIVERZITA HRADEC KRÁLOVÉ
UNIVERSITY OF HRADEC KRÁLOVÉ

FAKULTA INFORMATIKY A MANAGEMENTU
KATEDRA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATICS AND MANAGEMENT
DEPARTMENT OF INFORMATION TECHNOLOGY

METODY PRO ZLEPŠOVÁNÍ OBRAZU S POMOCÍ NEURONOVÝCH SÍTÍ
METHODS FOR IMAGE IMPROVEMENTS WITH THE USE OF NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. MICHAL DOBROVOLNÝ

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. KAREL MLS, Ph.D.

Hradec Králové 2019

ANOTACE

Práce se zaměřuje na problematiku obrazových dat. Konkrétně se jedná o nadvzorkování obrazu pomocí neuronových sítí. Práce zkoumá současné metody a implementuje jejich zlepšení díky optimalizacím. Výstupem práce je porovnání a hodnocení vlivu optimalizační funkce, v souvislosti s množstvím trénovacích dat, na konvolučních a generativních kontradiktorních sítích. Navržená optimalizační funkce vykazuje lepší výsledky na malém množství trénovacích dat. Součástí práce je také přehled teoretických základů neuronových sítí a metod zlepšování obrazu. Trénovací množina se skládá z obrázků použitých při soutěži NTIRE 2017. Skripty jsou psány v jazyce Python, využita byla knihovna pro modelování neuronových sítí Keras a výpočetní knihovna Tensorflow. Kvůli zvýšení rychlosti byly operace učení provedeny na grafických kartách s architekturou CUDA.

ANNOTATION

This thesis focuses on the problematics of image improvements. The main field is super-resolution with the use of a neuron network. The thesis explores current methods and implements training performance improvements. The result of this work is a comparison of influence optimizer to a number of testing samples and training results with use on convolutional and generative adversary networks. The proposed function has better results on a small number of train samples. Part of the thesis is a theory of neuron networks and base image processing. Train dataset is used from competition NTIRE 2017. The programming language is Python. The used framework is Keras with Tensorflow backend. For better performance were used GPUs with CUDA architecture.

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval/zpracovala samostatně a s použitím uvedené literatury.

Hradec Králové ....

podpis autora(-ky)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Karel Mls, Ph.D. za metodické vedení práce, vlídný přístup a odbornou pomoc.

OBSAH

Úvod **4**

1 Umělé neuronové sítě **5**
1.1 Historie neuronových sítí 5
1.2 Teorie neuronových sítí 6
1.2.1 Biologický model neuronu 6
1.2.2 Matematický model neuronu 6
1.2.3 Učení neuronových sítí 7
1.2.4 Hluboké učení 10

2 Úpravy obrazových dat **12**
2.1 Komprese obrazu 12
2.1.1 Bezztrátová komprese 12
2.1.2 Ztrátová komprese 13
2.2 Klasifikace obrazu 13
2.2.1 Neuronové sítě 14
2.3 Oprava obrazových dat 15
2.3.1 Rozšíření kontrastu 15
2.3.2 Prostorové průměrování 15
2.4 Nadvzorkování obrazu 16
2.4.1 Význam nadvzorkování obrazu 16
2.4.2 Klasické metody nadvzorkování obrazu 16
2.4.3 Využití neuronových sítí v oblasti SR 17

3 Aplikace hlubokých sítí **18**
3.1 Optimalizační funkce 18
3.2 Konvoluční sítě 19
3.2.1 SRCNN 21
3.3 Generativní kontradiktorní síť 21
3.4 GAN pro nadvzorkování obrazu 22
3.4.1 SRGAN 22
3.4.2 ESRGAN 24
3.5 Vývojové technologie 25
3.5.1 Hardware 26
3.5.2 Použité technologie 26
3.6 Databáze obrázků 28

4 Výsledky **29**
4.1 Metody hodnocení obrazu 29
4.2 Přehled testovaných sítí 30
4.3 Porovnání konvolučních sítí 31
4.4 SRGAN 34
4.4.1 Model 34
4.5 Porovnání testovaných sítí 37
4.5.1 Porovnání dle optimalizační funkce 37
4.5.2 Porovnání dle rychlosti učení 39
4.5.3 Porovnání dle generátoru 39
4.5.4 Porovnání výstupů na testovací sadě 40
4.6 Naměřené výsledky 42

5 Závěr **43**

Literatura **44**

Seznam symbolů, veličin a zkratek **49**

Seznam příloh **50**

A Přílohy **51**
A.1 Datové médium 53

SEZNAM OBRÁZKŮ

1.1 Model neuronu [1] 6
1.2 Lineárně separabilní problém řešený náhodně inicializovaným neuronem (A) a naučeným neuronem (B) 8
1.3 Funkce exkluzivní disjunkce XOR [1] 8
1.4 Funkce XOR řešená neuronovou sítí 9
3.1 Dvourozměrná konvoluce [2] 20
3.2 Model SRCNN [3] 21
3.3 Koncepční model GAN, na příkladu hledání cesty [4] 22
3.4 Model sítě generátoru pro SRGAN [5] 23
3.5 Model sítě diskriminátoru pro SRGAN [5] 23
3.6 Reziduální blok ESRGAN [6] 24
3.7 Model nahrazení bloků. [6] 24
3.8 Diskriminátory SRGAN a) v porovnání s ESRGAN b) [6] 25
4.1 Ukázkový obrázek z trénovací sady (0044.png, 2040x1140) 31
4.2 Model trénované SRCNN 32
4.3 Model Generativní kontradiktorní síť (GAN) 35
4.4 PSNR v průběhu učení generátoru před přidáním do modelu GAN . . 36
4.5 Průběh ztrátových funkcí sítě NN1006 37
4.6 "Adversarial" ztrátová funkce sítí NN1004 a NN1005 38
4.7 PSNR v průběhu učení sítí NN1000, NN1001, NN1002 a NN1003 . . 39
4.8 Ztrátové funkce sítí NN1004 a NN1006 40
4.9 Vizualizace výstupů obrázku baboon.png 41
A.1 Model generátoru sítě GAN 51
A.2 Model diskriminátoru sítě GAN 52

ÚVOD

Jednou z podnoží počítačového učení je hluboké učení. Výhodou hlubokého učení jsou jeho možnosti. Algoritmy hlubokého učení dokáží hledat a prezentovat data strukturovaná i nestrukturovaná. Algoritmy hlubokého učení již prokázaly svou přidanou hodnotu v mnoha oblastech. Vstupní data mohou mít různé podoby - textová data, obrazová data, časové řady a jiné. V oblasti zpracování obrazových dat jde například o úlohy oprav, nadvzorkování nebo porozumění obrazu. Tato práce se věnuje nadvzorkování obrazu. V textu jsou přiblíženy i principy ostatních příbuzných oblastí.

Hluboké učení a jeho algoritmy častěji a častěji obsazují přední příčky v testech různých oblastí zpracování obrazu. Většina dnešních algoritmů se opírá o konvoluční teorie. Příkladem může být velice rozvinutá oblast porozumění obrazu, oprava obrazových dat, stylizace obrazu nebo také generování chybějících částí. Označení hluboké neuronové sítě lze použít pro jakoukoliv síť o jedné a více skrytých vrstvách. Ve většině případů se však jedná o desítky či stovky vrstev. Jeho publicitě pomohl především rozvoj výpočetního výkonu. Dalším faktorem úspěchu je vývoj teoretických studií algoritmů hlubokého učení.

Pro nadvzorkování obrazu jsou v práci použity dva druhy neuronových sítí: jednodušší konvoluční sítě a složitější generativní kontradiktorní sítě. Nevýhodou konvolučních sítí je nedostatečná kvalita výstupu. Problémem u generativních kontradiktorních sítí je rychlost učení v souvislosti se zapojením dvou nezávislých sítí do jednoho modelu. Práce se tedy věnuje možnostem optimalizace těchto dvou druhů sítí s omezeným množstvím trénovacích prvků.

Cílem práce je zjistit, zda použití optimalizační funkce RMSprop bude mít positivní vliv na výsledky. Současně se práce částečně věnuje optimalizaci současných řešení modifikací hyperparametrů.

V první kapitole je probrána historie neuronových sítí, teorie neuronu a základy učení neuronových sítí.

Druhá kapitola se věnuje metodám zlepšování digitálního obrazu. Popsány jsou základní metody pro klasifikaci, opravu a nadvzorkování.

Třetí kapitola se věnuje teorii konvolučních a generativních kontradiktorních sítí. Popsány jsou různé přístupy k tvorbě architektur, jednotlivé druhy sítí, trénovací data a přehled vývojových prostředků.

Ve čtvrté kapitole jsou shrnuty dosažené výsledky jednotlivých sítí. Kapitola obsahuje také popis parametrů sítí, vyhodnocení jejich vlastností a porovnání.

1 UMĚLÉ NEURONOVÉ SÍTĚ

V kapitole je popsána historie umělých neuronových sítí, teorie neuronových sítí s popisem biologického neuronu, teorie hlubokého učení a nadvzorkování obrazu. V části nadvzorkování obrazu jsou obsaženy alternativní matematické metody nadvzorkování obrazu.

1.1 Historie neuronových sítí

První myšlenka modelace umělých neuronových sítí se objevila v roce 1943 v publikaci práce Waltera Pittse a Warrena McCulloha, kteří vytvořili jednoduchý umělý model neuronu. Ve své práci publikovali architekturu matematického aproximátoru logických funkcí. [7]

Jednalo se hlavně o teoretickou práci bez přímého praktického využití, která položila základ tomuto vědnímu oboru. V roce 1949 byla vydána kniha Donalda Hebba, která vycházela ze studia reflexů. V této knize bylo představeno Hebbovo učení. Jde o učící pravidlo, jež pracuje s inhibičními a excitačními vazbami neuronů, takzvanými synapsemi.

Důležitý milník nastal v roce 1957, kdy Frank Rosenblatt vynalézá zjednodušený model umělého neuronu. Tento model byl nazván perceptron. [8] Hlavní rozdíl mezi původním umělým neuronem byl v učícím algoritmu. Ten dokázal v konečném počtu kroků najít, v případě jeho existence, tahový vektor pro daná trénovací data. V dalších letech byl na tomto principu postaven počítač využívající principu umělých neuronů, Mark 1 Preceptron. Jeho účelem byla detekce znaků promítaných na speciální pole. Tento fakt zapříčinil, že se obor umělých neuronů stal středem vědeckého zájmu.

V 80. letech byl vymyšlen první vícevrstvý učící algoritmus. Do tohoto objevu nebylo možné řešit funkci XOR, jelikož se správné výsledky nedají rozdělit jednou vrstvou. Tento problém byl vyřešen v roce 1986 vydáním článku autorů Rumelharta, Hintona a Williamse. Tento článek popisuje algoritmus, který umožňuje zpětnou distribuci chyby, backpropagation. Na tomto algoritmu je postavena stále většina současných sítí. Tento algoritmus otevřel dveře současným neuronovým sítím.

V posledním desetiletí umožnil vysoký výpočetní výkon a matematické algoritmy automatizaci mnoha činností, které byly dříve dominantou člověka. Příkladem může být řízení auta, analýza dat, hraní her nebo zdravotnictví. Google se svým hlubokým učení dokáže na základě analýzy obrazových dat rozpoznat rakovinu prostaty s přesností 70%. Oproti člověku je to o 9% lepší výsledek. [9]

1.2 Teorie neuronových sítí

Sekce teorie neuronových sítí se zabývá problematikou neuronových sítí a hlubokého učení. Biologické základy neuronu jsou nezbytné pro pochopení fungování umělých neuronů, teoretické informace pak pro pochopení neuronových sítí.

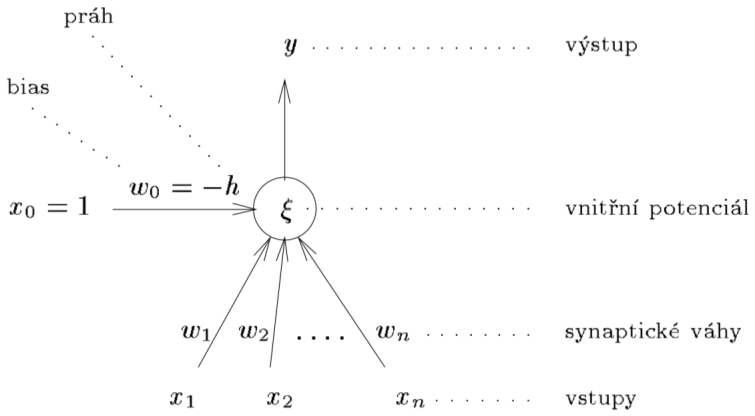
1.2.1 Biologický model neuronu

Neuronová soustava umožňuje živočichům vnímat svět kolem sebe. Základním stavebním prvkem je biologický neuron. Neuronové soustavy jsou schopny zpracovávat obrovské množství dat v reálném čase. Biologický neuron se stal základní inspirací pro matematické modely umělých neuronů.

Neuron se skládá ze tří základních částí, kterými jsou dendrity, soma a axon. Tělo neuronu, tedy soma, zpracovává základní funkci neuronu. Dendrity jsou výběžky, jež slouží k příjímání signálů od ostatních neuronů. Počet dendritů je variabilní. Bližší popis biologie neuronu je nad rámec této práce. Podrobně je tato problematika řešena například v monografii Jiřího Šímy a Romana Nerudy [1].

1.2.2 Matematický model neuronu

Umělé neuronové sítě jsou tvořeny z umělých neuronů (dále jen neuronů). Neuron má `n` vstupů `x` viz. obrázek 1.1. Vstupy simulují dendrity. Každý vstup má vlastní synaptickou váhu `w`. Váha určuje charakter vstupu. Pokud je hodnota váhy záporná, jedná se o inhibiční synapsi. Pokud je hodnota váhy kladná, jde o excitační synapsi. Potenciál neuronu se určuje součtem synaptických vah `w`.



Obr. 1.1: Model neuronu [1]

$$\xi = \sum_{i=1}^{n} w_i x_i - h \tag{1.1}$$

Rovnice 1.1 je rovnicí potenciálu neuronu, kde ξ je vnitřní potenciál neuronu, x jsou vstupy neuronu, w váhy neuronu a h prahová hodnota neuronu.

Aktivační funkce

Přenosová funkce, taktéž označovaná jako aktivační funkce, je dalším důležitým prvkem neuronu. Nejjednodušším typem je ostrá nelinearita reprezentovaná vztahem 1.2. Kromě ostrých aktivačních funkcí jsou používány v praxi také funkce spojité. Dle typu aktivační funkce neuron reaguje odezvou $y = \sigma(\xi)$, kde y je výstupní hodnota a ξ je aktivační funkcí neuronu.

$$\sigma(\xi) = \begin{cases} 1 \text{ jestliže } \xi \geq h \\ 0 \text{ jestliže } \xi < h \end{cases} \tag{1.2}$$

Proces učení

Hlavní výhodou neuronu je jeho schopnost samostatně nalézt rozdělení pro lineárně separabilní skupiny. Tyto skupiny mají konečný počet prvků. Proces učení se skládá ze dvou částí. První část je aktivní a druhá adaptivní. V aktivní fázi neuron počítá výstup pro dané vstupy, jejich váhy a aktivační funkce. Pokud nastane stav, kdy je třeba váhy upravit, neuron přejde do stavu adaptivního. Perceptronu (neuron s učící funkcí) jsou v adaptivní fázi předávány na vstup vzory a vyhodnocuje se, který byl určen špatně.

$$w_i(t+1) = w_i(t) + \eta(d - y)x_i \tag{1.3}$$

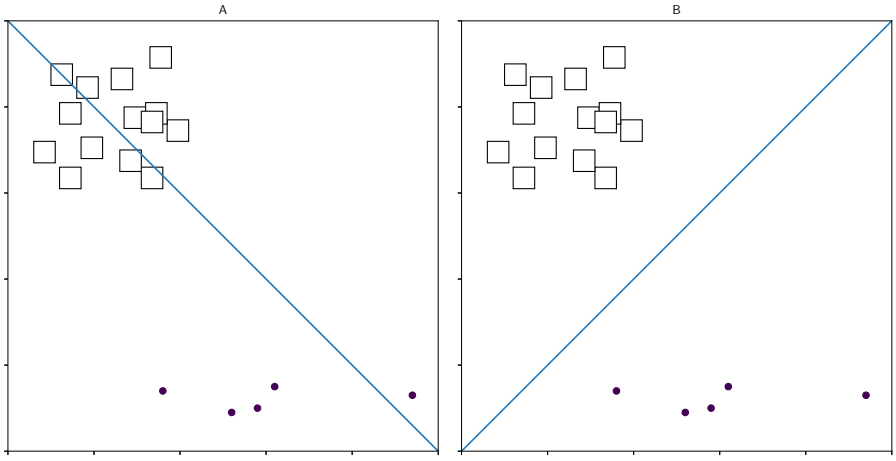
Výpočet váhy w_i, kde t je číslo iterace, η je parametr rychlosti učení, d očekávaný výstup a y je výstupem.

Základní princip učení je postaven na úvaze, že pokud je výstup perceptronu y roven očekávanému výstupu d, příslušná váha se nemění. Obrázek 1.2 znázorňuje graficky v rovině rozdíl náhodně inicializovaného neuronu a naučeného neuronu.

1.2.3 Učení neuronových sítí

V sekci matematického popisu neuronu 1.2.2 je popsán proces učení jednoho perceptronu. Z rovnice potenciálu neuronu (1.1) lze dedukovat, že neuron je schopen řešit pouze lineárně separabilní problémy. To bylo v historii perceptronům často vytýkáno. Například řešení exkluzivní disjunkce XOR nebylo možné.

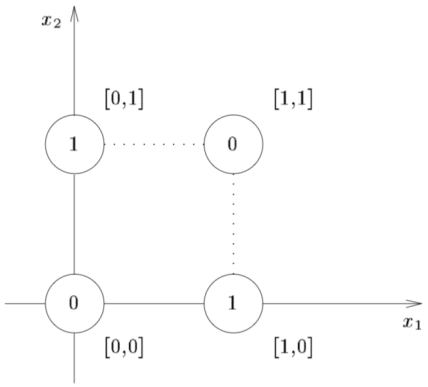
Hlavním problémem, proč nebylo možno využít neuronových sítí, byla absence vhodné učící funkce. S příchodem první učící funkce se tedy začaly perceptrony



Obr. 1.2: Lineárně separabilní problém řešený náhodně inicializovaným neuronem (A) a naučeným neuronem (B)

spojovat do sítí. Přidávání perceptronů do sítí je možno si představit jako přidání dělící čáry do roviny. Přidání dělící čáry umožní řešení složitějšího problému. Spojením dvou perceptronů již tedy můžeme řešit problém exkluzivní disjunkce XOR viz. obrázek 1.4.

U neuronových sítí se častěji než skokové aktivační funkce používají funkce spojité sigmoidální. V těchto případech již pak neurony nedělí rovinu čárou, ale spojitou křivkou. Spojitá křivka pak umožňuje zvýšení přesnosti výpočtů. Při obecném pohledu se výstup jednoho perceptronu stane vstupem jednoho nebo více dalších perceptronů.

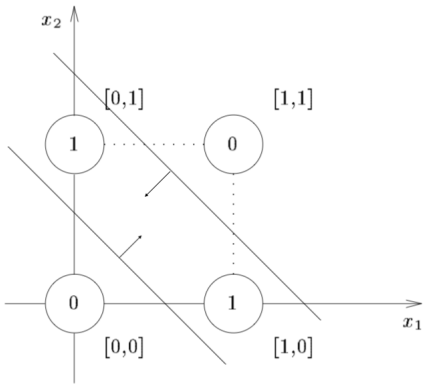


Obr. 1.3: Funkce exkluzivní disjunkce XOR [1]

Perceptrony se v neuronových sítích dělí na vstupní, pracovní a výstupní. Vstupní perceptrony dostávají na svůj vstup normalizovaná vstupní data. Na výstupy vstup-

ních perceptronů jsou připojeny pracovní perceptrony. Ty jsou uspořádány do vrstev, které nazýváme skryté. Každá síť může mít konečný počet skrytých vrstev. Výstupní perceptrony zajišťují reprezentaci výstupních dat. Jejich uspořádání se mění dle požadovaného výstupu. V případě klasifikace dvou tříd by výstupní vrstva měla dva perceptrony.

Způsob zapojení perceptronů určuje architekturu neuronové sítě. Základní dvě architektury jsou dopředná a cyklická. Dalším velice slibným modelem se stal temporální model.



Obr. 1.4: Funkce XOR řešená neuronovou sítí

Dopředná architektura byla navrhnuta tak, aby vzruchy postupovaly stále dopředu. Zjednodušeně řečeno jsou výstupy předchozích vrstev vždy vstupy vrstev následujících. Vzruchy se v žádném případě nevrací sítí zpět.

V případě cyklické architektury nejsou perceptrony zapojeny do vrstev. Příkladem cyklické architektury může být rekurentní topologie. V rekurentní topologii je výstup perceptronu připojen na vstup, a je tak provedena zpětná vazba.

Neuronové sítě se v průběhu učení dělí na tři etapy dle výpočtu - organizační, aktivní a adaptační. Organizační fáze slouží k nastavení topologie sítě. V aktivní fázi dochází k samotnému výpočtu dat. V adaptační fázi pak síť určuje chybu výsledků pro dané vstupy a upravuje váhy jednotlivých synapsí.

Vícevrstvé sítě vyžadují složitější algoritmus pro nastavení vah. V kapitole 1.1 byl zmíněn tento algoritmus zpětné propagace chyby. Backpropagation se nejčastěji používá v kombinaci s algoritmem Stochastic gradient descent. Algoritmus v adaptivní fázi porovná výstup sítě a očekávané výsledky. Dle zjištěné chyby poté upravuje váhy. Cílem je dosáhnout co nejnižší možné chyby. Oba algoritmy jsou nad rámec práce a jsou podrobněji popsány v knize Teoretické otázky neuronových sítí [1].

Učení neuronových sítí má také dva nežádoucí stavy. Prvním problémem je přeučení. Přeučení je nežádoucí stav nastavení vah synapsí. K přeučení dochází v situaci, kdy naučená síť je na testovacích datech až příliš přesná. Při použití takové sítě na jiná než trénovací data jsou výsledky nepoužitelné. Pro přeučení je důležité množství trénovacích dat. Předcházet se mu dá například rozšířením učící množiny.

Lokální minimum je problémem druhým. Během učení je běžné, že se síť ocitne v lokálním minimu. Je důležité, aby učící algoritmus nebyl v lokálním minimu zastaven, ale pokračoval na minimum globální. Je třeba nastavit správně parametr λ, koeficient rychlosti učení. S příliš malým koeficientem rychlosti učení se může stát, že nedojde k přeskočení hranice lokálního minima. Dalším řešením problému lokálního minima je funkce dropout. Přidanou hodnotou funkce dropout je ignorace určitého počtu vstupů pro každou iteraci. Tím dojde k rozšíření počtu kombinací a redukuje se možnost nalezení pouze lokálního minima.

1.2.4 Hluboké učení

Jednou z podnoží počítačového učení je hluboké učení. Výhodou hlubokého učení jsou jeho možnosti. Algoritmy hlubokého učení dokáží hledat a prezentovat data strukturovaná i nestrukturovaná. Vstupní data mohou mít různé podoby - textová data, obrazová data, časové řady a jiné. [10]

Hluboké učení je v dnešní době velmi populární. Jeho publicitě pomohl především rozvoj výpočetního výkonu. Dalším faktorem úspěchu je vývoj teoretických studií algoritmů Deep neuron network (DNN). [10]

Označení DNN lze použít pro jakoukoliv síť o jedné a více skrytých vrstvách. Ve většině případů se však jedná o desítky či stovky vrstev. Aplikační část práce bude věnována Convolution neuron network (CNN). Této problematice se věnuje kapitola 3.2.

Učení DNN lze dělit na dvě skupiny. První je učení s učitelem (anglicky supervised). V případě učení s učitelem jsou síti předána data a předem známý správný výsledek. Síť se pak snaží naučit, dle čeho je daný výsledek správný. Druhou skupinou je učení bez učitele (anglicky unsupervised). V případě učení bez učitele se síť sama naučí hledat podobnosti mezi daty. Pracuje na principu shlukování. [11]

Přenesené učení

Podstata přeneseného učení spočívá v myšlence přenesení předtrénovaného modelu na nový, avšak příbuzný problém. Touto technikou lze docílit rychlých výsledků nové sítě s poměrně malým množstvím trénovacích dat. To je přínosné hlavně v oblastech, kde není množství dat dostatečné. [12]

Přenesené učení má tři hlavní problémy. Prvním problémem je definice znalosti, jež má být přenesena. Druhým problémem je “jak přenést znalost“ a posledním pak “kdy přenést znalost“.

Na definování přenášené vědomosti je třeba pohlížet z pohledu domény a požadavku. Některé požadavky mohou být v jiné doménové oblasti, a přesto přinést výkonnostní zlepšení trénování. Po definování přenášení znalosti je třeba vyvinout novou síť tak, aby bylo možno znalost přenést. Posledním problémem je definování, kdy znalost přenést. V některých situacích může přenesení znalosti cílovou síť poškodit z pohledu výkonosti. Komplexněji je problematika přeneseného učení popsána v článku A Survey on Transfer Learning [13].

2 ÚPRAVY OBRAZOVÝCH DAT

Kapitola popisuje čtyři vybrané kategorie práce s obrazovými daty. Popisuje základní mechanismy komprese obrazu, ztrátové i bezztrátové. Navazující podkapitola popisuje klasifikaci obrazu. Vybráno je několik možností klasifikací. Součástí je také stručný popis použití neuronových sítí. Dále je popsána oprava obrazových dat a některé její mechanismy. Na závěr kapitoly je popsána problematika nadvzorkování obrazu, která zároveň představuje hlavní předmět a cíl praktické části této práce.

2.1 Komprese obrazu

Komprese obrazu se z pohledu zpětné rekonstrukce rozděluje na bezztrátovou a ztrátovou. U bezztrátové komprese lze data zpětně rekonstruovat bez ztráty jakékoliv informace. Z tohoto důvodu je používána v oblastech, kde nesmí dojít ke ztrátě žádné informace. Ztrátová komprese je metodou, kdy při rekonstrukci není možno úplně zrekonstruovat původní informaci. Odstraňuje především data, jež mají v v daném kontextu malý význam.

Kompresní poměr Veličina vyjadřující podíl velikosti dat před kompresí a komprimovaných, se nazývá kompresní poměr. Ztrátová komprese má typicky vyšší kompresní poměr než bezztrátová komprese.

2.1.1 Bezztrátová komprese

Bezztrátová komprese je metoda, která reprezentuje vstupní obraz tak málo bity, jak je jen možné bez ztráty jakékoliv informace. Má využití například v oblastech medicíny, archivace obrazu či analýza obrazu.

Obraz je často vnímám jako dvourozměrné pole hodnot intensity. Ve většině případů jsou tyto hodnoty reprezentovány 8 bity. V případě černobílého obrazu je to pak hodnota mezi 0 a 255. Některé oblasti používají větší rozsahy. Například radiologie uchovává informace ve 12 bitech. [14]

Bezztrátovou kompresi lze demonstrovat na umělém příkladu. Pokud by vstupní data měla hodnoty: $p(n) = (1, 1, 1, 5, 5, 5, 7)$. Je možno bezztrátově vyjádřit zkráceně: $p(n) = (3, 1)(3, 5)(1, 7)$. Z nového vyjádření je možno zpětně rekonstruovat informaci do původního tvaru.

Komplexněji téma popisuje kniha Lossless Compression Handbook [14].

2.1.2 Ztrátová komprese

Ztrátová komprese je metodou, která využívá nedokonalosti lidského zraku (obraz) nebo například sluchu (zvuk). Cílem ztrátové komprese je informaci ořezat tak, aby člověk rozeznal známý objekt. Většina metod se snaží komprimovat data tak, aby lidské oko téměř nebo vůbec nepoznalo rozdíl. Lidské vidění je citlivé na lokální jasové změny, které odpovídají hranám. Ty se kompresní algoritmy snaží zachovávat. Absolutní jas však není vizuálně vnímán, proto bývá odstraněn. [15]

Důležitou složkou obrázku je barva. Ta bývá často ovlivněna jako první. Informace barvy se mohou komprimovat průměrováním. Při průměrování se používají hodnoty několika okolních bitů, kde algoritmus zprůměruje společné hodnoty barev. Další metodou je snížení počtu barev. Oba přístupy je také možno kombinovat.

Příkladem průměrování může být následující postup. Pokud by byl vstup $p(n) = (1, 2, 1, 3, 2, 3)$, průměrováním byl bylo dosaženo hodnot $p(n) = (2, 2, 2, 2, 2, 2)$. Ze získaného výsledku je možno vidět, že nelze původní hodnoty rekonstruovat. Informace je ztracena.

Detailně se problematice věnují knihy Digital Image Processing Techniques [16] a The Image Processing Handbook [15].

2.2 Klasifikace obrazu

Detekci a klasifikaci obrazu je v posledních letech věnována spousta pozornosti. Úkolem klasifikace je zařazení všech nalezených objektů do předem známých tříd. Klasifikace probíhá na základě předem naučeného klasifikátoru z trénovací množiny. Níže jsou krátce popsány vybrané metody Histograms of Oriented Gradients (HOG), Suport vector machines (SVM), Speeded-Up Robust features (SURF) a neuronové sítě.

HOG Metoda HOG je založena na histogramech orientovaných gradientů. V roce 2005 byla vyvinuta za účelem detekce lidských postav v obraze. Vychází z myšlenky popisu objektů pomocí gradientů, kde jednotlivé gradienty jsou reprezentovány vlastní velikostí a směrem. Obraz se rozdělí na malé výřezy zvané buňky. Pro každou buňku je spočítán histogram. Histogram je počítán ze všech oblastí buňky.

Celý obraz je rozdělen do několika výřezů stejné velikosti. Každá buňka reprezentuje určitou oblast (např. 8x8). Každé buňce je určen směr gradientu, přičemž výsledné gradienty následně vymezují hledaný objekt. [17]

SVM Metoda SVM je novější metodou detekce objektů v obraze. Úkolem metody je využití efektivních algoritmů k nalezení lineární hranice. Zároveň je metoda na-

vržena tak aby dokázala reprezentovat složité nelineární funkce. Podstatou metody je převod obrazu do vícedimenzionálního prostoru, kde se pak třídy dělí lineárně. [18]

Metoda SVM se pro robustnější detekci může kombinovat s metodou HOG. Příkladem takové kombinace může být práce A Two-Stage Approach to People and Vehicle Detection With HOG-Based SVM [19].

SURF SURF je metodou, jež umí detekovat významné body nezávisle na změně měřítka. Metoda byla vyvinuta se záměrem rychlého deskriptoru, který pracuje v reálném čase. Detekce bodů probíhá přímo pomocí determinantu Hessianovy matice. Výpočet determinantu je umožněn aplikací integrálního obrazu. Integrální obraz je struktura, která se buduje nad vstupním obrazem. Tato struktura slouží k rychlému výpočtu součtu hodnot uvnitř libovolného výřezu obrazu. [20]

2.2.1 Neuronové sítě

Neuronové sítě v posledních letech dominují svým řešením v řadě doménových úkolů. Klasifikace a porozumění obrazu není výjimkou. Od roku 2012 jsou prezentovány stále lepší a přesnější řešení. V roce 2012 byla vydána klasifikační síť DCNN AlexNet. O dva roky později představil Girshuck et al. řešení RCNN. V roce 2015 byla představena síť VGGNet. Mezi nejnovější řešení patří DenseNet. [21]

DCNN AlexNet Síť představená v roce 2012 se zakládá na pěti konvolučních plně propojených vrstvách. Představením architektury sítě bylo přivedeno několik nových mechanismů. Jedním z nich je použití aktivační funkce ReLU. Dalším je překrývající se spojování (overlapping pooling), tedy použití menšího skoku konvoluce než velikosti jádra. [22]

VGG Hluboká konvoluční síť Visual Geometry Group (VGG) je jednu z nejpopulárnějších sítí. Její popularita je vysoká především díky přenesenému učení. VGG má šest různých voleb konfigurací. Každá konfigurace je rozdílná v počtu vrstev, velikosti konvolučního jádra nebo kombinací těchto dvou změn. Velikosti konfigurací jsou v rozsahu jedenácti až devatenácti vrstev. Velikosti konvolučních jader jsou 1x1 a 3x3. Konkrétní konfigurace a podrobný popis sítě jsou k nalezení v originálním článku Very Deep Convolutional Networks [23].

DenseNet DenseNet je první klasifikační sítí, která se namísto optimalizace parametru pokusila o nový přístup k řešení problému. Architektura je velice podobná síti

ResNet [1]. Prvním rozdílem je použití řetězení funkcí namísto sčítání. Tedy namísto napojení vrstev jedné za druhou $[l-1]$ jsou tedy vrstvy napojeny od současné pozice až po poslední dostupnou $[1, ...(l-1)]$, kde l je vrstva. Kompletní popis architektury je obsažen v originálním článku Densely Connected Convolutional Networks [21].

2.3 Oprava obrazových dat

Pořídit digitální snímky v požadované kvality může být velice časově náročné, například pokud není možnost kontrolovat zdroje světla. V takových případech je třeba využít jiných technik. Dvě nejzákladnější popisuje tato podkapitola. Jsou jimi rozšíření kontrastní funkce a prostorové průměrovaní.

2.3.1 Rozšíření kontrastu

Pro úpravu kontrastu je využívám histogram jasové funkce. Digitální obrázky ukládají hodnoty jasu mezi 0 (černá) až 255 (bílá). Barevné obrázky mají pro každou barevnou složku jeden bit navíc. Některé profesionálnější kamery jsou citlivější, a dokáží tak ukládat větší rozsahy. Může se jednat například o 10, 12 nebo více bitů.

Pokud je inherentní rozsah variability jasu obrazu mnohem menší než dynamický rozsah kamery, následné elektroniky a digitizéru, pak je skutečný rozsah čísel mnohem menší než plný rozsah 0 až 255. Příkladem může být snímek tkáně v mikroskopu. Osvětlení v mikroskopu a lehké zabarvení řezu poskytují velmi malý celkový kontrast. Ty jsou reprezentovány úzkými vrcholy a prázdnými oblastmi na koncích histogramu. Viditelnost přítomných struktur může být zlepšena natažením kontrastu tak, aby hodnoty obrazových bodů byly přiřazeny. Ideálně tak, aby pokryly celý dostupný rozsah. Mnohé hodnoty však mohou stále vykazovat nulový počet pixelů v histogramu, což znamená, že bylo dosaženo vyššího vizuální kontrastu. Nebyla však zvýšena schopnost rozlišovat odchylky, které nejsou obsaženy v původním obraze. Touto úpravou je také zvýšen šum obrazu.

Problematika a další metody korekce jsou blíže popsány v knize The Image Processing Handbook [15].

2.3.2 Prostorové průměrování

Lineární expanze kontrastu popsaná v předchozí podkapitole je často doprovázena zvýšenou viditelností šumu (náhodné fluktuace hodnot pixelů). Hluk je důležitou vadou v obrazech, které mohou mít mnoho různých forem a vznikají z různých zdrojů.

[1] Hluboká konvoluční síť s využitím reziduálních bloků. [24]

Nejjednodušší formou prostorového zprůměrování je sčítat hodnoty jasu pixelů v každé malé oblasti obrazu, dělit počtem pixelů v okolí a výslednou hodnotu. Výsledek je použit k vytvoření nového obrazu.

Běžnějším způsobem, jak dosáhnout průměrování okolí, je nahradit každý pixel průměrem jeho hodnoty a jeho sousedů. Toto je často popisováno jako operace jádra, protože implementace může být zobecněna jako součet hodnot pixelů v oblasti násobených množinou celočíselných vah. Proces je také nazýván konvoluce a může být ekvivalentně prováděn ve Fourierově prostoru, kde konvence spočívá v tom, že tyto koeficienty mají být vynásobeny hodnotami pixelů, které jsou kolem centrálního pixelu vynásobeny součtem vah. [15].

2.4 Nadvzorkování obrazu

Pro nadvzorkování obrazu se v dnešní době používá mnoho rozdílných metod. Jedná se o základní metody, od matematických až po neuronové sítě. Metody dosahují různých výsledků. V některých případech může být využití složitějších metod zbytečně namáhavé. V případě velkých zvětšení se však interpolační metody nedají použít pro získání vysokého rozlišení.

2.4.1 Význam nadvzorkování obrazu

Motivace k nadvzorkování rozlišení obrazu může být velice různá. Cílem může být získání více detailů, čitelnost textu nebo třeba zobrazení na větším zařízení, než je rozlišení fotografie.

Prvním příkladem může být zobrazení fotografie na velkém monitoru. Může se jednat o situace, kdy byla fotografie pořízena na nedostatečně kvalitní fotoaparát nebo byla případně pořízena starším telefonem a uživatel se snaží tuto fotografii prezentovat na větším monitoru, třeba televizoru s vysokým rozlišením. Ve všech zmíněných případech dojde k nekvalitnímu zobrazení.

Jedním z dalších případů může být bezpečnostní záznam, kdy je pořízen záznam kamery, ale není dostatečně čitelná například poznávací značka nebo jiný text, který by mohl vést k dopadení pachatele.

2.4.2 Klasické metody nadvzorkování obrazu

V této podkapitole jsou vybrány některé metody nadvzorkování obrazu. Cílem je přiblížit řešení problematiky jednoduššími metodami pro porovnání se složitějšími.

Nejbližší soused

Nejjednodušší metodou je nejbližší soused. Při nadvzorkování obrazu je pro nově vzniklý bod použita hodnota jasu z nejbližšího známého bodu. Pro velkou jednoduchost metody je při nadvzorkování ztracena přesnost obrazu.

Bilineární interpolace

Pokročilejší metodou je bilineární interpolace. Při jejím výpočtu jsou oproti nejbližšímu sousedu použity čtyři sousední body. Hodnota nového bodu se tedy vypočítá vztahem:

$$\begin{aligned} f(i+\Delta i, j+\Delta j) = [f(i+1,j) - f(i,j)]\Delta i + [f(i,j+1) - f(i,j)]\Delta j + \\ [f(i+1,j+1) + f(i,j) - f(i+1,j) - f(i,j+1)]\Delta i \Delta j + f(i,j) \end{aligned} \tag{2.1}$$

Kde (i, j) jsou souřadnice bodu a f je hodnota jasu bodu. $\Delta i, \Delta j$ vyjadřují neceločíselnou vzdálenost souřadnic bodu. [25]

2.4.3 Využití neuronových sítí v oblasti SR

Nadvzorkování, neboli Super resolution (SR), je sada metod pro nadvzorkování obrazu. Může se jednat o jedno snímkové obrázky nebo sekvence snímků (videa). Jedná se o metody sloužící k rekonstruování obrazu nízkého rozlišení do obrazu vysokého rozlišení.

V souvislosti se SR jsou využívány také metody počítačového učení, konkrétněji hlubokého učení. Ty pak slouží k vytvoření chybějících hodnot bodů s maximálním potlačením artefaktů.

Při použití klasických metod dochází ve zvetšených snímcích ke ztrátě vysokých frekvencí. Ztrátou těchto frekvencí je obraz po nadvzorkování rozmazaný nebo jinak deformovaný. Cílem využití neuronových sítí je získání ostrého obrazu ve vysokém rozlišením. Zajímavějších výsledků dosahují Super resolution convolution neuron network (SRCNN), Super resolution generative adversary network (SRGAN) a Enhanced super resolution generative adversary network (ESRGAN).

3 APLIKACE HLUBOKÝCH SÍTÍ

Vlastnosti hlubokých sítí jsou určeny především třemi hlavními parametry. Architekturou, tedy způsobem, jakým jsou jednotlivé perceptrony propojeny, metodou výpočtu zpětné chyby a použitou aktivační funkcí. Kapitola popisuje a přibližuje tuto problematiku. Popisuje základní prvky použitých sítí. Popsána je teorie s možnými přístupy k tvorbě SR sítí. Současně jsou také popsány možnosti vývojových prostředků a hardware.

3.1 Optimalizační funkce

V matematice se optimalizací rozumí výběr nejlepšího elementu z dané množiny elementů dle výběrového kritéria. V oblasti neuronových sítí se pojmem optimalizace rozumí hledání správného nastavení vah pro vazby jednotlivých neuronů tak, aby v dalším kroku bylo dosaženo nižších hodnot chybové funkce nebo ideálně nalezeno globální minimum.

Optimalizační funkce v hlubokých konvolučních sítích vylepšují funkci zpětné propagace chyby (backpropagation). V současnosti jsou neuronové sítě používány pro mnoho různých problémů. Při samotném modelování sítě není nutno základní funkce implementovat. Všechny frameworky nabízejí již hotová řešení. Programátor se empirickou metodou snaží najít jejich optimální nastavení.

Adagrad

Adagrad je optimalizační funkce, která mění rychlost učení dle četnosti výskytu parametru. U parametrů s nižším výskytem je uplatněna vyšší hodnota rychlosti učení, zatímco u parametrů s vyšším výskytem je uplatněna nižší hodnota rychlosti učení. Ideální použití funkce je pro data s nízkou hustotou. Vzorky v těchto databázích jsou různé a mají nižší podobnost. Výhodou této funkce je také odstranění nutnosti sledovat a nastavovat rychlost v čase. [26]

Adadelta

Funkce Adadelta vychází z funkce Adagrad, avšak omezuje součet předchozích gradientů na určitou maximální hodnotu. Tím zajišťuje, aby hodnota parametru rychlosti učení neklesala k nule. Tím předchází předčasnému ukončení procesu učení. [27]

RMSprop

Root Mean Square Propogation, neboli RMSprop, je postaven na myšlence ekvivalence znamének předchozího a současného gradientu. Pokud jsou znaménka shodná, zvětší krok. Pokud jsou znaménka rozdílná, krok sníží. Tato optimalizační funkce je používána především pro rekurentní neuronové sítě. [28]

Adam

Optimalizační funkce Adam, celým názvem Adaptive moment estimation, je velmi oblíbenou a používanou metodou. Vychází z optimalizační funkce Adadelta a RMSprop. Adam si ukládá předchozí hodnoty gradientu a také používá moment vycházející z průměru předchozích gradientů. [29]

3.2 Konvoluční sítě

Konvoluční neuronové sítě jsou aplikovány na řadu různých úloh. První skupinou je počítačové vidění. V oblasti počítačového vidění jsou sítě využívány k rozpoznávání obličejů, rozpoznávání druhu scény obrazu, klasifikaci obrazu a dalších. Zpracování přirozeného jazyka je oblastí druhou. V této oblasti se sítě využívají k rozpoznávání řeči a klasifikaci textu. [30]

Konvoluční metoda je inspirovaná zrakem. U zrakového nervu reagují neurony na vstup aktivací dalších okolních neuronů. K aktivaci dochází podle velikosti konvolučního jádra. V konvoluci je využívána operace matematické diskrétní konvoluce. Příklad dvojrozměrné konvoluce je znázorněn na obrázku 3.1. [31]

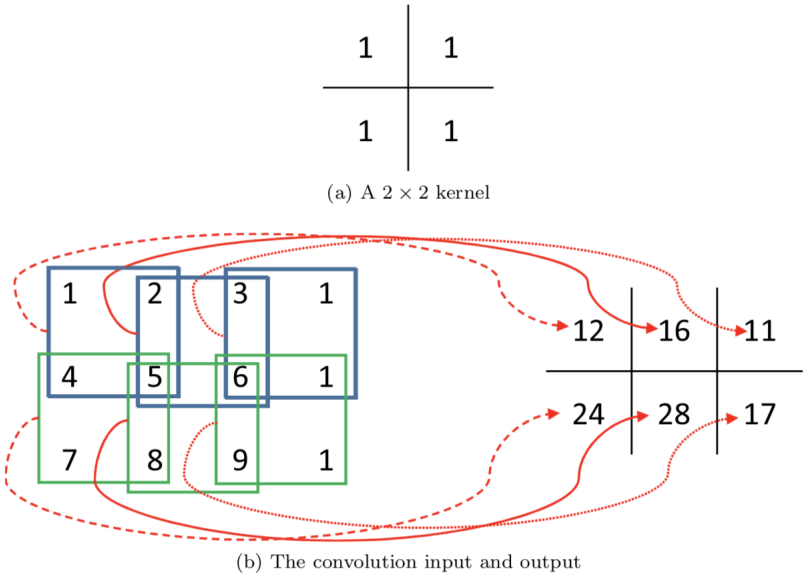
Při aplikaci konvolučních sítí prochází data několika různými vrstvami. Jednotlivé vrstvy jsou popsány níže. Vstupní data jsou dvourozměrnými poli pro obrázek, třírozměrnými poli pro video a jednorozměrnými poli pro zvuk. [30]

Konvoluční vrstva

Konvoluční vrstvy jsou složeny z trénovatelných jader nebo filtrů. Konvoluční filtry procházejí celou hloubkou vstupu. Každá jednotka přijímá vstupy ze sady jednotek umístěných v blízkosti v předchozí vrstvě. Velikost sady je určena filtrem. Taková sada je nazývána vnímavým polem neuronů. Aplikací několika takových filtrů jsou získány vstupní vrstvy konvoluce. Takováto vrstva je základem každé CNN. [30]

Nelineární vrstva

Vrstva využívající kombinace několika různých aktivačních funkcí se nazývá nelineární. Typickým příkladem jsou funkce sigmoid, tanh a ReLU. Tyto funkce jsou



(a) A 2 × 2 kernel

(b) The convolution input and output

Obr. 3.1: Dvourozměrná konvoluce [2]

upřednostňovány, protože zrychlují proces trénování. [30]

Sjednocovací vrstva

Po konvoluční vrstvě může následovat sjednocovací vrstva (anglicky pooling layer). Tato vrstva sjednocuje malé bloky z konvoluční vrstvy a vytvoří z něj jednu hodnotu. Například maximální hodnotu z daného obdélníku. Sjednocovací vrstva postupně snižuje objem vstupních dat, čímž se snižuje počet vstupních parametrů. Sjednocovací funkce mohou být různé. Příkladem takové funkce je hledání maxima, průměrování nebo minimum. [30]

Plně propojená vrstva

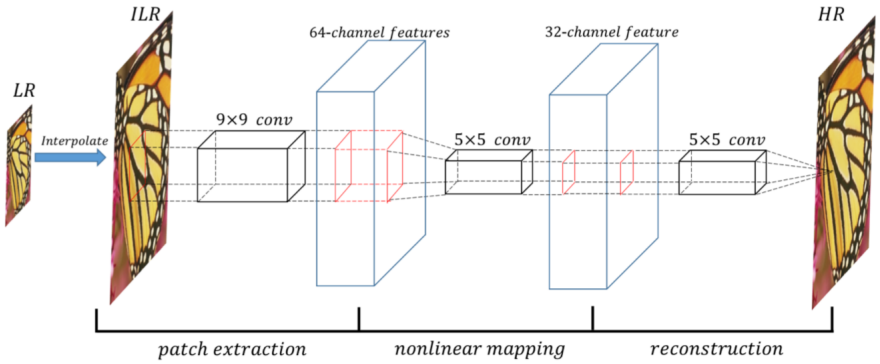
V modelu konvolučních sítí existuje jedna nebo více plně propojených vrstev. Tyto vrstvy provádí uvažování na vysoké úrovni. Toho je dosaženo propojením neuronů předchozí vrstvy ke všem neuronům současné vrstvy. [30]

Normalizace dávek

Často používanou vrstvou v moderních architekturách je normalizace dávek, Batch normalization (BN). BN která řeší problém interního kovariantního posunu. [32] Výhodou užití BN je rychlejší proces učení, jelikož neuronová síť nemusí reagovat na změnu distribučních hodnot mezi vrstvami.

3.2.1 SRCNN

Metoda SR s využitím konvolučních sítí znázorněna na obrázku 3.2 je referenčním příkladem SRCNN. Jde o tří vrstvou architekturu. Vrstvy SRCNN jsou extrakce patchů, nelineární mapování a rekonstrukce obrazu. [3]



Obr. 3.2: Model SRCNN [3]

Model SRCNN je obdobný běžné CNN. Obecně lze o SRCNN uvažovat jako o běžné konvoluční síti. Jednotlivé vrstvy sítě slouží ke komplexnímu mapování mezi Low resolution (LR) a High resolution (HR) obrázkem. Ztrátovou optimalizační funkcí SRCNN je Střední kvadratická chyba (MSE). [3]

Při přechodu z LR do HR je vstup modifikován zmíněnými třemi vrstvami. Obrázek nízkého rozlišení se nadvzorkuje bikubickou interpolací a vstoupí do první vrstvy - extrakce patchů. Jelikož je vstup již upravený, nese s sebou jisté nevýhody. Těmito nevýhodami jsou ztráta detailu, náročnost získání interpolovaných vstupů a nutnost znát jádro převzorkování.

3.3 Generativní kontradiktorní síť

GAN, česky generativní kontradiktorní síť, se využívá v různých oblastech generování obsahu. Příkladem může být CycleGAN, který dokáže změnit scénu obrázku, zimní krajinu na letní, koně na zebru nebo stylovat fotografii dle předlohy. [33] AgecGAN umí simulovat podobu tváře s věkem [34] nebo detekovat anomálie v orgánech. [35]

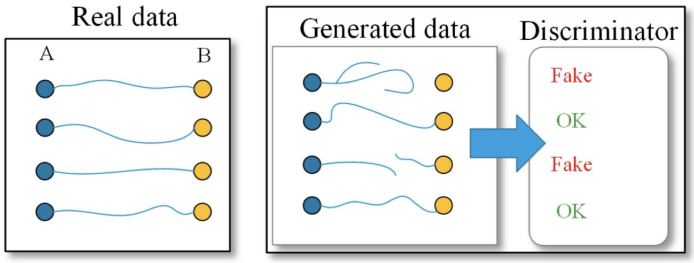
Model GAN

GAN je složený ze dvou různých vícevrstvých sítí. První je generátor a druhou sítí je pak diskriminátor (kap. 3.3). Generátor je sítí produkující výsledky. Diskriminátor

slouží k určení pravosti výsledku produkovaného generátorem. [36]

Matematicky lze popsat GAN rovnicí 3.1. Pro zjištění distribuce generátoru p_g na datech x, je třeba definovat vstupní šum proměnných $p_z(z)$. Následně se provede mapování datového prostoru jako $G(z;\theta_g)$. G je diferencovatelná funkce reprezentovaná sítí θ_g. Definuje se také druhá síť $D(x,\theta_d)$. $D(x)$ reprezentuje pravděpodobnost, že x pochází z dat namísto p_g (generátoru). D se trénuje k maximalizaci pravděpodobnosti přiřazení správného štítku. To platí u trénovacích příkladů stejně jako u vzorků. Současně se trénuje G k minimalizaci $log(1 - D(G(z)))$. [36]

$$\min_G \max_D V(D,G) = \mathbb{E}_{z\sim p_z(z)}[log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{3.1}$$



Obr. 3.3: Koncepční model GAN, na příkladu hledání cesty [4]

Úlohou generátoru je generování obsahu, v případě této práce nadvzorkovaného obrazu. Většinou se jedná o vícevrstvou konvoluční síť.

Diskriminátor slouží k poskytnutí zpětné vazby generátoru. Zjednodušeně klasifikuje výstup generátoru na pravý nebo nepravý. Pokud je výstup označen jako nepravý, vrátí se data zpět generátoru a ten provede vylepšení. Tento proces se opakuje, dokud diskriminátor neoznačí výsledný obrázek za reálný.

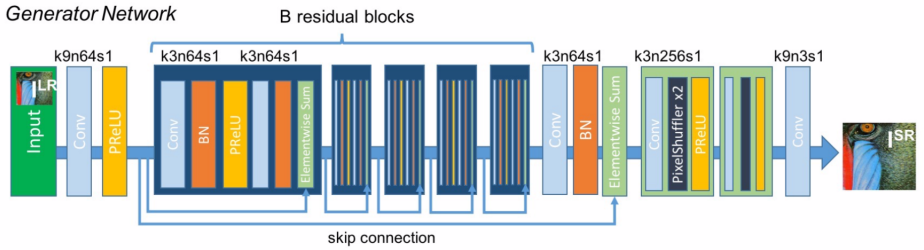
3.4 GAN pro nadvzorkování obrazu

Využití modelu GAN je výhodné i v oblasti SR. Jelikož generátor dostává zpětnou vazbu z diskriminátoru, jsou výsledky sítí GAN v porovnání s jinými metodami lepší. Záleží však na metodě hodnocení. Z pohledu MSE jsou výsledky lepší, avšak Peak signal-to-noise ratio (PSNR) dosahuje horších hodnot. [37]

3.4.1 SRGAN

Základní myšlenkou modelu SRGAN je naučit generátor (obr. 3.4) generovat takové obrázky, aby oklamal diskriminátor (obr. 3.5). V obou případech je využito vlastností CNN k dosažení maximálních výsledků.

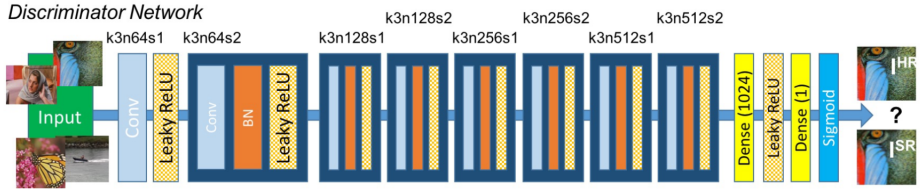
Základem konvoluční sítě generátoru je reziduální blok B, který je tvořen opakujícími se bloky. Také je využito 64 konvolučních map s jádrem 3x3, jež jsou optimalizovány dávkovou normalizací (angl. batch-normalization) [32] s aktivační funkcí parametrické ReLU (angl. ParametricReLU) [38].



Obr. 3.4: Model sítě generátoru pro SRGAN [5]

Diskriminátor (obr. 3.5) je konvoluční síť specializovaná na rozpoznání generovaných obrázku od pravých. Díky tomu je generátor (obr. 3.4) nucen generovat obrázky s vysokou podobností obrázkům pravým. [5]

Diskriminátor využívá vlastností konvolučních sítí. K dosažení (cíle) oddělení HR od generovaných SR je použita aktivační funkce LeakyReLU. V celé síti není použita sjednocovací vrstva hledající maxima (anglicky max-pooling). Síť obsahuje osm konvolučních vrstev. Konvoluční jádro se zvětšuje dle násobku dvou v rozsahu velikosti 64 až 512. [5]



Obr. 3.5: Model sítě diskriminátoru pro SRGAN [5]

Ztrátová funkce

Ztrátová funkce se skládá ze tří funkcí, kde první definuje generátor, druhá diskriminátor a třetí pak hodnotí celou GAN. Ztrátové funkce jsou kritické pro dosažení potřebného výkonu u takto komplexních sítí.

$$l^{SR} = \underbrace{l_X^{SR}}_{\text{content loss}} + \underbrace{10^{-3} l_{Gen}^{SR}}_{\text{adversial loss}} \tag{3.2}$$

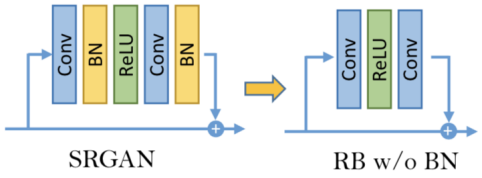
Kde l^{SR} reprezentuje celkovou ztrátovou funkci. l_X^{SR} je ztrátovou funkcí diskriminátoru a $10^{-3}l_{Gen}^{SR}$ generátoru. Podrobný popis ztrátové funkce je nad rámec práce a je k nalezení v originální publikaci Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network [5].

3.4.2 ESRGAN

Rozšířený model ESRGAN je silně založený na architektuře SRGAN. Hlavním rozdílem jsou reziduální bloky generátoru a rozšíření diskriminátoru. Cílem ERGANu je nalezení lepší rovnováhy mezi detailem obrázku a zlepšení PSNR. [6]

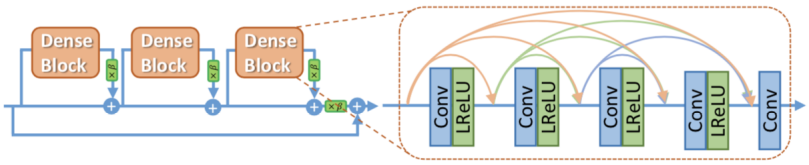
Generátor

V modelu generátoru jsou dvě změny. První změnou je odebrání BN (obr. 3.6). Druhou změnou je nahrazení reziduálních bloků za reziduální husté bloky (obr. 3.7).



Obr. 3.6: Reziduální blok ESRGAN [6]

Odebráním BN vrstvy bylo docíleno lepších výkonnostních výsledků a bylo docíleno lepšího hodnocení PSNR. Empirickou metodou bylo zjištěno, že vrstva BN vytváří v případě hlubších sítí artefakty.

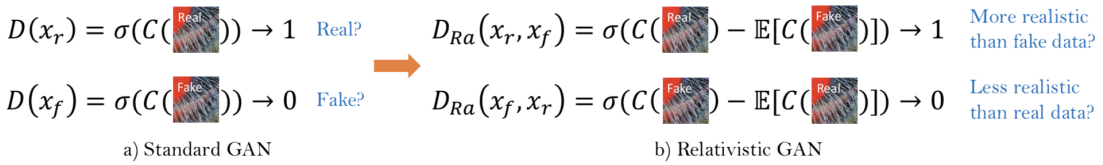


Obr. 3.7: Model nahrazení bloků. [6]

Diskriminátor

S vylepšením generátoru byl zároveň vylepšen i diskriminátor. Rozdíl mezi původním a novým diskriminátorem je ve výsledném štítku. Původní D označoval pravděpodobností, zda jde o reálný obrázek nebo generovaný. Nový D se snaží předpovědět

relativní realističnost skutečného obrazu v porovnání s generovaným. Teorie diskriminátoru je postavena na článku Realistic GAN [39]. [6]



Obr. 3.8: Diskriminátory SRGAN a) v porovnání s ESRGAN b) [6]

Realistic GAN je modifikací klasického GAN modelu, kde je diskriminátor trénovaný k rozpoznávání obrázků tak, aby označil výstup jako reálnější než reálný. Tím je dosaženo ostřejších a detailnějších výsledných obrázků. [39]

Ztrátová funkce

V ESRGAN je použita rozvinutější ztrátová funkce, která je efektivnější. Lepší efektivity je dosaženo přesunutím některých vlastností před aktivaci. Tím je dosaženo lepšího procenta aktivovaných neuronů a snížena deformace jasové funkce. [6]

$$L_G = L_{percep} + \lambda L_G^{Ra} + \eta L_1 \tag{3.3}$$

Kde L_G je celkovou ztrátou generátoru. L_{percep} je podobnost obrázků, L_1 je ztrátová funkce "content loss". λ, η jsou koeficienty balancování rozdílů.

3.5 Vývojové technologie

Počítačové učení je velice populární obor, a proto v oblastech vývojových prostředků existuje možnost volby prostředků. Níže v této kapitole jsou popsány nejpopulárnější dostupné frameworky a použité technologie.

Frameworky počítačového učení

Oblast počítačového učení vývojáři nabízí mnoho možností. Nikdo si nepíše vlastní neurony a neskládá z nich sítě. Tuto nízkou úroveň obstarávají frameworky počítačového učení. Na programátora pak zbývá důležitější část, tedy návrh sítě a optimalizace parametrů. Při optimalizaci parametrů je často využíváno technologie “grid search“. Ta funguje jako kombinátor zadaných parametrů a je schopna na základě zadaných parametrů vytvořit kombinace, provést učení a určit nejúspěšnější kombinaci.

Tensorflow

Knihovnou z dílny Google je Tensorflow, který je open source projektem pro počítačové učení. Jde o vyšší abstrakci než u nízko úrovňových knihoven, kde může být příkladem Theano. Tensorflow, který má silnou uživatelskou základnu, tvoří ve spolupráci s Googlem jeden z nejpoužívanějších frameworku počítačového učení.

PyTorch

Knihovnou počítačového učení od společnosti Facebook je PyTorch. Jedná se o hybridní knihovnu, jež je určena jak pro nízkoúrovňové návrhy, tak pro robustní modely. Knihovna komfortně pracuje na GPU. Její nevýhodou je menší zkušenost a otestovanost v čase.

Keras

Pro vyšší míru abstrakce se často používá Keras. Ten je navržen nad frameworky nižších úrovní, jako je například Tensorflow, Theano nebo CNTK. Důvod použiti Kerasu je také jeho jednoduchost pro použití na experimentech.

3.5.1 Hardware

Důležitým, až zásadním aspektem pro úspěch počítačového učení je hardware. Ve většině případů se již namísto Central processing unit (CPU) používá Graphic processing unit (GPU) nebo Tensor processing unit (TPU). Volba hardwaru je nutná podle daných problémů. V nízkém počtu parametrů procesory vykazují mnohem lepší výsledky než grafické karty. Pokud se však řeší problémy s velkým počtem parametrů, jako například textová analýza nebo obrazová data, nelze již procesory použít. Grafické karty díky nadstavbě CUDA dovolují paralelní výpočty s velkým množstvím dat.

3.5.2 Použité technologie

Pro trénování byly využity externí knihovny a hardware dostupný přes Google Cloud Compute Engine. Použitou sestavu tvořila konfigurace desíti virtuálních procesorů, 50GB RAM paměti a dvě grafické karty NVIDIA Tesla K80.

CUDA

Použitá výpočetní platforma je Compute Unified Device Architecture (CUDA). Ta byla vyvinuta společností NVIDIA se záměrem umožnit výpočty na GPU. CUDA

přidává možnost paralelizace přes veliké množství jader grafické karty. Tím přináší mnohem vyšší výpočetní výkon než CPU. [40]

Keras

Knihovna Keras byla použita s backendem Tensorflow kapitola 3.5. Pro potřeby projektu jsou možnosti nastavení základních ztrátových funkcí a vyšší míra abstrakce celého modelu sítě dostačující.

Tensorboard

Tensorflow, framework popsaný v kapitole 3.5, s sebou automaticky instaluje i nástroj pro grafickou podporu učících modelů Tensorboard. Ten umožňuje graficky znázornit křivku učení a výsledky ztrátové funkce.

Conda

Je projektem s otevřeným kódem, který usnadňuje správu závislosti pomocí linuxových environmentů. Umožňuje jednotnou správu environmentů a balíčků, čímž dovoluje vývojáři spravovat závislosti odděleně v potřebných verzích.

Kex

Experimentální nastavení umožnila a usnadnila modifikovaná knihovna paloha/kex (dále jen kex) využívající `fit_generator` knihovny Keras. Knihovna byla upravena pro současnou verzi knihovny Keras, doplněna o zaznamenávání historie pro generování grafu přes Tensorboard popsaný výše.

Nástroj kex sebou přináší možnost paralelního testování experimentů. Jde o paralelizaci experimentů, nikoliv výpočtů jednoho experimentu. Využívá možnost pustit jednotlivé experimenty na jednotlivých grafických kartách. Pro každý experiment generuje samostatný report, grafy a export testovaných parametrů.

Google cloud

Nejvýhodnější nabídku má v současnosti Google Cloud. Ten oproti Amazonu nabízí dostupnější ceny. Alternativou může být také například služba Scaleway. Ta však účtuje každý dedikovaný stroj i v čase, kdy není aktivně využíván nebo je vypnut. Tím se může Scaleway výrazně prodražit. Volba hardware byla pro úspěšnost experimentů hlubokého učení na daném objemu dat kritická.

Ve službě Google Cloud byly testovány čtyři alternativy. Těmi jsou Colaboratory, Compute Engine a ML Engine ve variantách “notebooks“ a “jobs“. Všechny varianty jsou popsány níže.

Colaboratory Služba Colaboratory je dostupná zdarma. Nabízené výpočetní prostředky jsou CPU, GPU a TPU. Využívá Python notebooku. Podporuje připojení Google disku a instalaci dalších potřebných balíčků. Její nevýhodou jsou výpočetní prostředky pro složitější trénovací úlohy. V případě této práce nestačila dostupná paměť RAM.

ML Engine (Notebooks) Notebooks na Google cloudu jsou placenou alternativou ke Colaboratory. Navíc oproti Colaboratory nabízí možnost sestavit si vlastní konfiguraci hardware s příslušnou cenou. Nevýhodou jsou pak omezení, která přináší samotný Python notebook. Těmi jsou omezení v práci jako samostatný proces a paměťová náročnost.

ML Engine (Jobs) Jobs spustitelné na LM Engine jsou účtované za výpočetní čas a zvolenou konfiguraci jako u ostatních služeb. Rozdílem je práce s balíčkem neboli trenérem. Trenér je jeden spustitelný skript dle dokumentace. Neumožňuje však použití nástroje kex pro paralelní trénování a všechna data musí být vždy před spuštěním trénování nakopírována z bucket (Google Cloud úložiště) do lokálního úložiště. Výhodou je však automatická správa modelů a jejich verzování.

Compute Engine Použitou volbou se stal Google Compute Engine dostupný na Google cloudu. Ten nabízí možnost vytvoření vlastní konfigurace hardware účtované za výpočetní čas. Pokud je instance vypnutá, není za ni nic účtováno, na rozdíl od Scaleway. Instance byla zvolena v konfiguraci 10 virtuálních procesorů, 50GB RAM paměti, 100GB SSD disku a dvou grafických karet NVIDIA Tesla K80. Výhodou Compute Engine je přístup jako k reálnému stroji. Přes SSH je možno na serveru spouštět skripty. Není limitováno žádným technologickým omezením jako v případě Python notebooku nebo strukturálními limity jako u LM Engine.

3.6 Databáze obrázků

Pro trénování, validaci i testování byla použita soutěžní obrazová sada IDV2K [41]. Obrazová sada obsahuje 900 obrázků. Z toho je 800 určených pro trénování a 100 pro testování. Obrázky jsou ve vysokém rozlišení. Při trénování probíhala transformace dat do nízkého rozlišení.

Výsledky práce popsané v kapitole 4 jsou spočítány na sadě čtrnácti obrázků "Set14" [42] [43]. Sada obsahuje čtrnáct obrázků, které znázorňují několik různých scén. Jedná se o jednu z často používaných sad pro porovnávání výsledků v oblasti SR.

4 VÝSLEDKY

V kapitole jsou popsány všechny testované architektury a jejich srovnání v kategorii. Sítě byly testovány z hlediska různých rychlostí učení a architektur. Ke srovnání výsledků slouží hodnotící funkce PSNR popsaná blíže v sekci 4.1. Porovnáno bylo několik modelů konvolučních sítí a následně několik generativních kontradiktorních sítí.

V kapitole 4.5.4 jsou shrnuty výsledky nejúspěšnějších sítí z kategorie konvolučních sítí a generativních kontradiktorních sítí. Výsledná tabulka obsahuje dosažené výsledky PSNR na testovací sadě čtrnácti obrázků "Set14".

4.1 Metody hodnocení obrazu

Sekce se věnuje hodnocení obrazových dat. Popisuje ve zkratce metody použité pro porovnání výsledků. Metody hodnotí kvalitu z různých vlastní obrazu. Výsledky hodnocení také závisí na testovaném obraze. Všechny metody jsou zvoleny v černobílé variantě, aby bylo dosaženo jednoduššího modelu.

Mean squared error

Střední kvadratická odchylka, MSE, dvou obrazů. Jedná se o nejjednodušší a nejpoužívanější metodu hodnocení v oblasti digitálního obrazu. Metoda vyjadřuje součet kvadrátů odchylek v jednotlivých bodech obrazu. [44]

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4.1}$$

Rovnice 4.1 vyjadřuje výpočet střední kvadratické odchylky, kde MSE je hodnotou chyby, n je počet prvků souboru, $\hat{Y}$ je střední hodnota veličiny a Y jsou hodnoty.

Peak signal-to-noise ratio

Funkce PSNR hodnotí poměr mezi signálem a šumem v obraze. Vychází z metody MSE. Tato metoda je ideální pro hodnocení veličiny komprese obrazu. [45]

$$PSNR[n] = 10\log_{10}\frac{255^2}{MSE[n]} \tag{4.2}$$

Rovnice 4.2 popisuje vztah, kde neznámá MSE je střední kvadratickou chybou (rov. 4.1).

Strukturální simularita

Výsledkem funkce Structural similarity (SSIM) je hodnota udávající podobnost obrazů v rozmezí `<0,1>`. Funkce vychází z fungování lidského oka. [46]

$$SSIM[x,y] = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4.3}$$

Rovnice funkce SSIM reprezentovaná 4.3, kde μ je vážený průměr, σ váženou variací a $C_i = (K_i L)$. Kde K_i je konstanta menší než 1 a L je rozsah hodnot pixelu. [46]

4.2 Přehled testovaných sítí

Sekce shrnuje dosažené výsledky trénovaných sítí, přípravu vstupních dat a postupy trénování. Testováno bylo několik variací SRCNN a SRGAN. Poslední popsaný koncept sítě ESRGAN nebyl implementován a ani porovnán z důvodu náročnosti na výpočetní čas všech testů.

NN1000 Průměr PSNR 30, 1461 db.

Jedná se o architekturu inspirovanou sítí z článku [47]. Obsahuje tři konvoluční vrstvy s velikostmi jader 9-3-5. Stanovená rychlost učení je 0, 001. Použitá optimalizační funkce je Adam, není-li uvedeno jinak. Uvedené průměry jsou měřeny na testovacích datech "Set14".

NN1001 Průměr PSNR 30, 0324 db.

Je parametrickou modifikací sítě NN1000 s rychlostí učení je 0, 003. Velikosti konvolučních jader jsou totožné s původní sítí.

NN1002 Průměr PSNR 30, 1886 db.

Je sítí s velikostí konvolučních jader 9-5-5 a rychlostí učení 0, 001.

NN1003 Průměr PSNR 30, 0390 db.

Je parametrickou modifikací sítě NN1002 s rychlostí učení 0, 003.

NN1004 Průměr PSNR 24, 9772 db.

Generátor této sítě je trénován na nižším vzorku než kompletní síť. Trénování proběhlo na deseti epochách, při 800 obrázcích se dvěma náhodnými výřezy. Dohromady byl vytvořen počet 1600 vzorků.

NN1005 Průměr PSNR 26, 1629 db.

Vychází z vlastností sítě NN1004. Síť využívá vlastností optimalizační funkce RMSprop s rychlostí učení 0.001 na vrstvě diskriminátoru.

NN1006 Průměr PSNR 27, 9129 db.

Je základní architekturou GAN sítí. Podrobný popis generátoru a diskriminátoru je obsažen v kapitole 3.4.1. Generátor byl natrénován na stejném vzorku jako konvoluční sítě.

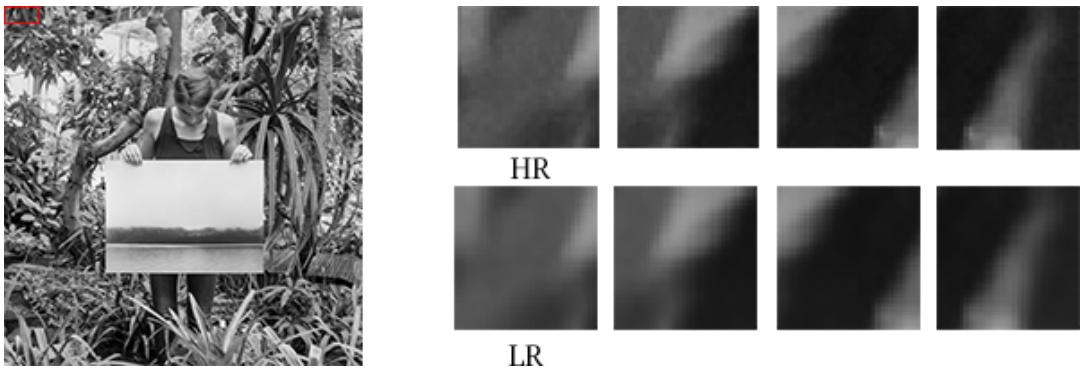
NN1007 Průměr PSNR 27, 5834 db.

Vychází z vlastností sítě NN1006, kde diskriminátor využívá vlastností optimalizační funkce RMSprop s rychlostí učení 0.001.

4.3 Porovnání konvolučních sítí

V sekci je popsán generátor dat, model a výsledky testovaných architektur. Nejlepší testovaná konvoluční síť dosahuje mediánu 29.1694 db a průměru 29.5620 db pro funkci PSNR.

Generátor dat



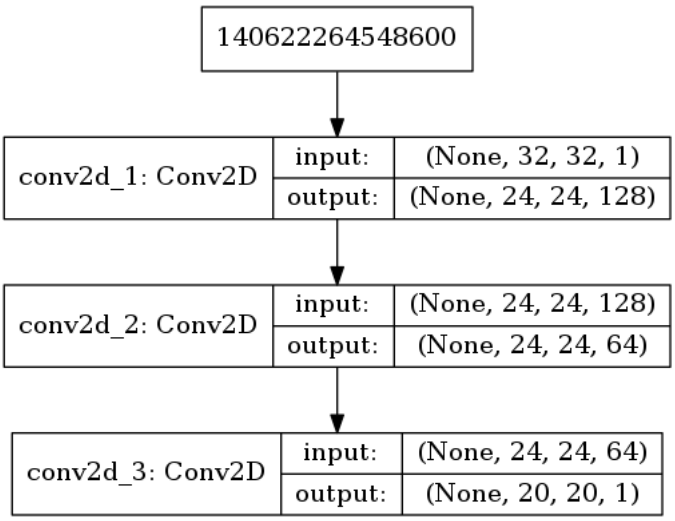
Obr. 4.1: Ukázkový obrázek z trénovací sady (0044.png, 2040x1140)

Generátor dat byl upraven na dávkovou přípravu vstupních dat. Generátor využil vlastností nástroje kex a data předpřipravoval po třech obrázcích. Každý obrázek pak byl rozsekán na výseky velikosti 32x32 pixelů s jedním černobílým kanálem. Překrytí jednotlivých výseků bylo 50%. Z trénovací sady bylo použito 50 procent obrázků. Původní kód generátoru je k nalezení na Githubu MarkPrecursor/SRCNN-keras. Použitá verze je upravena pro současnou verzi Pythonu a využívá dávkové

zpracování vstupních dat. Velikost jedné dávky je nastavena na jeden soubor. To je průměrně 10836 LR vzorků a 10836 HR vzorků.

Model

SRCNN model obr. 4.2 byl použit z Githubu MarkPrecursor/SRCNN-keras. Model je modifikací původního modelu, který byl navržen v článku v Image Super-Resolution Using Deep Convolutional Networks [47]. Oproti původnímu návrhu je model zjednodušen. Používá optimalizační funkci Adam s jednotnou hodnotou rychlosti učení pro všechny vrstvy.



Obr. 4.2: Model trénované SRCNN

Zdrojový kód 1 je implementací modelu ve frameworku Keras. Jde o výstup, který generuje nástroj kex při generování experimentu. Pod zdrojový kód vždy kex umístí parametry použité při experimentu. Těmi se dají sítě různě upravovat, a to od úpravy hyperparametů trénování až po modifikace samotného modelu, jako v příkladu použití různých rychlostí učení.

Konvoluční vrstvy mají velikost 9x9-3x3-5x5. Vycházejí z původního návrhu sítě. Velikost dávky není definována. Síť používá jeden černobílí kanál.

Trénování

Trénované sítě byly dvě. Mezi sebou se lišily v parametru rychlosti učení. Cílem experimentu bylo zjistit, zda trénované sítě dosáhnou stejných výsledků a jak se budou lišit průběhy ztrátové funkce a hodnotící funkce PSNR.

```
model = models.Sequential()

model.add(Conv2D(filters=128, use_bias=True, kernel_size=(9, 9),
    input_shape=(32, 32, 1), activation="relu",
    kernel_initializer="glorot_uniform", padding="valid"))
model.add(Conv2D(filters=64, use_bias=True, kernel_size=(3, 3),
    kernel_initializer="glorot_uniform", activation="relu",
    padding="same"))
model.add(Conv2D(filters=1, use_bias=True, kernel_size=(5, 5),
    kernel_initializer="glorot_uniform", activation="linear",
    padding="valid"))

model.compile(optimizer=(Adam(lr=LR)), loss='mean_squared_error',
    metrics=[
        'mse',
        'acc',
        PSNR,
    ])

# DEBUG
model.summary()

######################################################################
###   Used parameters: {'NBE': 30, 'LR': 0.0003}
######################################################################
```

Zdrojový kód 1: Sekvenční model SRCNN napsaný v knihovně Keras

Naměřené výsledky ukazují, že neexistuje podstatný rozdíl mezi rychlostí učení $0,001$ a $0,003$. Výsledná přesnost sítě je totožná. Rozdíl nastává v průběhu chybové funkce, kde se rychleji učící síť lépe adaptuje.

Výpočetní čas strávený jednotlivými sítěmi je obsažen v tabulce 4.1. Nejlepšího výsledku na trénovací množině dosáhla síť NN1002 s výsledkem 34,85 db, následovaná sítí NN1000 dosahující hodnoty 34,81 db.

Výsledky sítě pro kompletní testovací sadu Set14 jsou shrnuty v tabulce 4.3. Síť dosahuje zajímavých výsledků při faktoru nadvzorkování 2x z pohledu hodnocení PSNR. Medián sítě je $30,0944$ db a průměr je $30,1886$ db.

Síť	Doba trénování	PSNR
NN1000	26,65 H	34,81 db
NN1001	27,52 H	34,64 db
NN1002	23,13 H	**34,85 db**
NN1003	22,19 H	34,65 db

Tab. 4.1: Doba trénování konvolučních sítí

4.4 SRGAN

V sekci jsou popsány jednotlivé části testovaných GAN sítí. Nejlepších výsledků bylo dosaženo u sítě NN1006. Ta dosahuje mediánu 28, 3415 db a průměru 27, 9129 pro funkci PSNR a při dvojnásobném nadvzorkování. Komplexnost architektury GAN sítí není vhodná pro nástroj kex, který byl využit při trénování SRCNN. Experimenty nad sítí GAN byly však tímto nástrojem inspirovány.

Generátor dat

Původní generátor dat byl přepsán z výkonnostních důvodů. Původní implementace generovala dávku a vnitřně si generátor držel ukazatel na index souboru. Tento přístup byl obecně přínosný v jednoduchém škálování počtu procesů. Doba generování však byla mnohonásobně horší než u nové implementace používající `yield`.

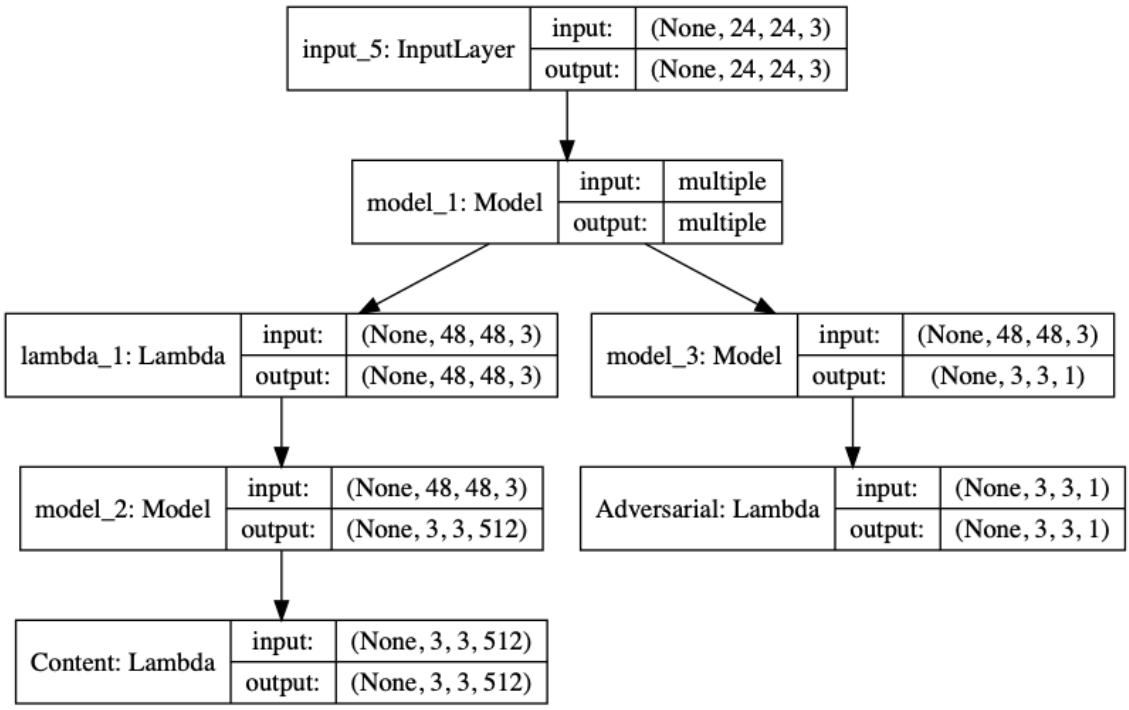
4.4.1 Model

Implementace modelu SRGAN byla použita z github repositáře MathiasGruber / SRGAN-Keras [48]. Jedná se o implementaci totožnou s původním návrhem dle článku Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network [5].

Generátor Implementace generátoru je totožná s původním návrhem. Obsahuje tedy vstupní vrstvu s libovolnými rozměry vstupu. Dále pre-residual blok s konvoluční vrstvou, která má velikost jádra devět, a PReLU funkcí. Následně šestnáct reziduálních bloků popsaných v kapitole 3.4.1. Dále post-residual blok s konvoluční vrstvou, s velikostí jádra tři, a dávkovou normalizací. Následuje blok pro nadvzorkování. Model ukončuje výstupní konvoluční vrstva s velikostí jádra devět.

Kompletní model generátoru je k nalezení v přílohách A.1.

Diskriminátor Diskriminátor je implementovaný totožně k návrhu z původního článku. [5] Model obsahuje vstupní vrstvu s rozměry odpovídajícími HR obrázku



Obr. 4.3: Model GAN

a osm konvolučních vrstev s velikostí jádra tři. Následuje plně propojená vrstva s počtem filtrů 1024. Dále je použita aktivační funkce LeakyReLU následovaná další plně propojenou vrstvou s jednou jednotkou.

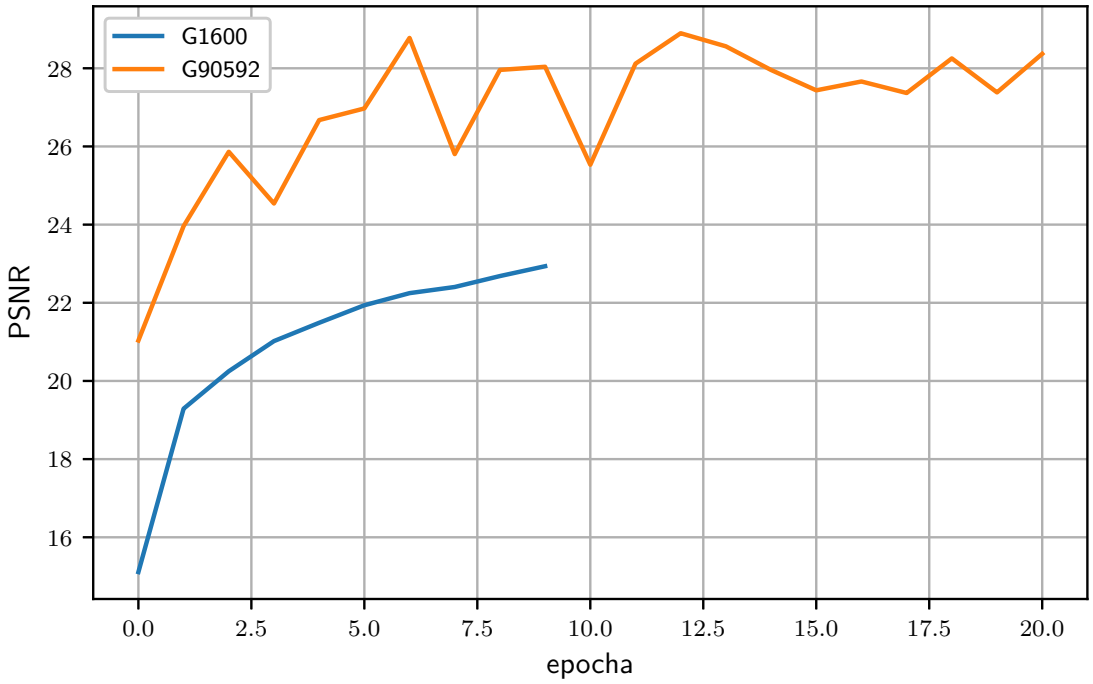
Trénování

Komplexnost modelu GAN v některých případech komplikuje trénování sítě. Postup trénování je více krokový. Nejprve se trénuje generátor (obr. 3.4). Při dosažení potřebné úrovně se přejde k trénování diskriminátoru. V průběhu trénování diskriminátoru se průběžně také trénuje generátor tak, aby jeho výstupy byly přesnější.

Generátor byl před trénován ve dvou verzích. Každá verze je rozdílná pouze rozsahem dat. Jednotlivé názvy se skládají z písmene "G" a počtu vzorků, které byly generátoru poskytnuty jako trénovací data.

G1600 Generátor s označením G1600 je verzí, která zpracovala při jedné epoše 1600 vzorků dat. Jedná se o velice nízké množství dat, které má za cíl vizualizovat závislost modelu SRGAN a optimalizační funkce na velikosti trénovacích dat. Generátor funguje na náhodném výběru dvou výseků zdrojového obrázku. V průběhu učení se tak může stát, že se data při každé epoše mění, nebo naopak síť generátoru dostává stále dokola pouze dva konkrétní výřezy z každého jednotlivého obrázku.

G90592 Označení G90592 nese generátor, jenž byl inspirovaný generátorem použitým pro konvoluční sítě. Data jsou poskytována sítí generátoru v dávkách po šestnácti vzorcích. Každý obrázek je rozřezán na výřezy. Tyto výřezy jsou čtvercového tvaru a vzájemně se překrývají v 50 procentech plochy. Generátor tedy dostal při počtu 800 obrázků přesně 90592 výřezů.



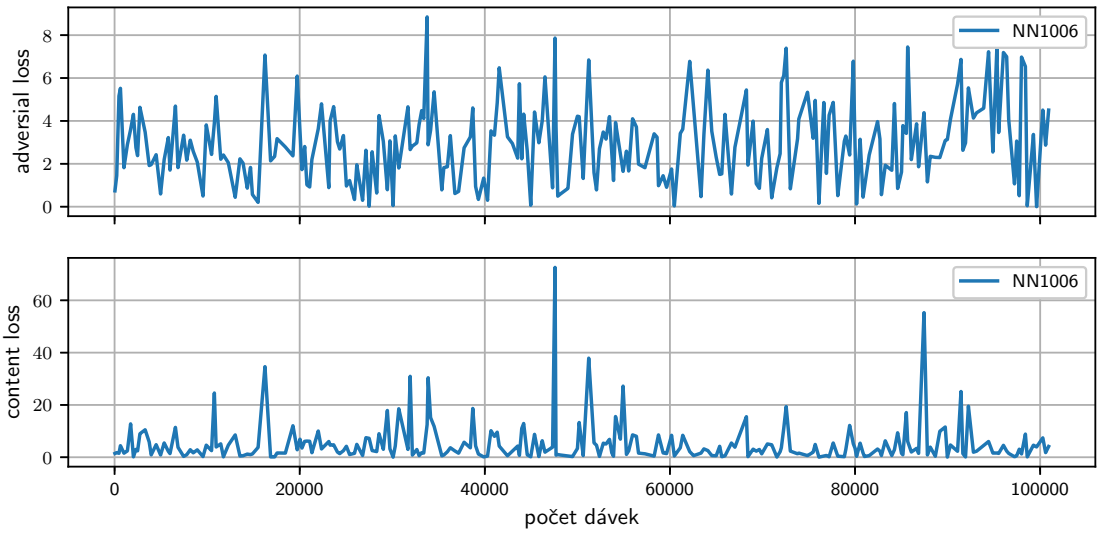
Obr. 4.4: PSNR v průběhu učení generátoru před přidáním do modelu GAN

VGG Net Inicializační váhy byly použity z předtrénované klasifikační sítě VGG Net. VGG pochází z dílny společnosti Google ve spolupráci s University of Oxford. Jejím účelem je klasifikace objektů na obrázku. [23] Vzhledem k příbuznosti úloh je možno využít již předučených vah a vyhnout se učení od nuly. Problematiku blíže popisuje podkapitola přeneseného učení 1.2.4.

Trénování diskriminátoru je jednodušší typ úlohy, kdy se posílají generovaná a skutečná data se správnými štítky do sítě v dávkách. Dávky jsou rozděleny vždy na dvě po šestnácti vzorcích. První dávka obsahuje pouze generované obrázky. Druhá dávka obsahuje pouze originální HR obrázky.

Průběh trénování diskriminátoru je součástí trénování celé SRGAN sítě. Během průběhu trénování jsou měřeny dvě ztrátové funkce, jež ovlivňují výsledné váhy neuronů v poměru $[0,001\ 0,006]$ pro ztrátovou funkci diskriminátoru a generátoru. Ztrátová funkce diskriminátoru je křížová entropie. Generátor používá MSE.

Celkový průběh učení SRGAN sítě je reprezentován pomocí “content“ a “adversarial“ ztrátových funkcí. Jejich popis je v podkapitole 3.4.1. Průběh těchto ztrátových funkcí nejlepší sítě je možno sledovat na grafu 4.5



Obr. 4.5: Průběh ztrátových funkcí sítě NN1006

První změnou oproti původní sítí je použitá trénovací sada. Použitá trénovací sada byla IDV2K [41]. Původní síť byla autorem trénována na datech ImageNet. [5]

Druhou změnou je testování sítě diskriminátoru na optimalizační funkci RMSprop (kap. 3.1). Ta se v průběhu experimentů ukázala jako zajímavá volba ve stabilizaci “adversarial“ ztrátové funkce. Blíže jsou tato specifika popsána v kapitole 4.5.1.

4.5 Porovnání testovaných sítí

Kapitola je věnována porovnání naučených sítí dle různých parametrů. Z každé kategorie byly vždy vybrány nejúspěšnější variace. Cílem srovnání je nalezení optimální nastavení architektury a hyperparametrů.

4.5.1 Porovnání dle optimalizační funkce

Porovnání v této podkapitole se věnuje vlivu optimalizační funkce na proces učení a výsledky SR sítí. Experiment optimalizační funkce měl zjistit, zda je možno urychlit proces učení. Tato problematika je aktuální zejména u SRGAN sítí.

model	PSNR (db)	SSIM
NN1004	24.9772	0.7172
NN1005	26.1629	0.8004
NN1006	**27.9129**	**0.8438**
NN1007	27.5834	0.8360

(a) Průměr výsledků SRGAN sítí

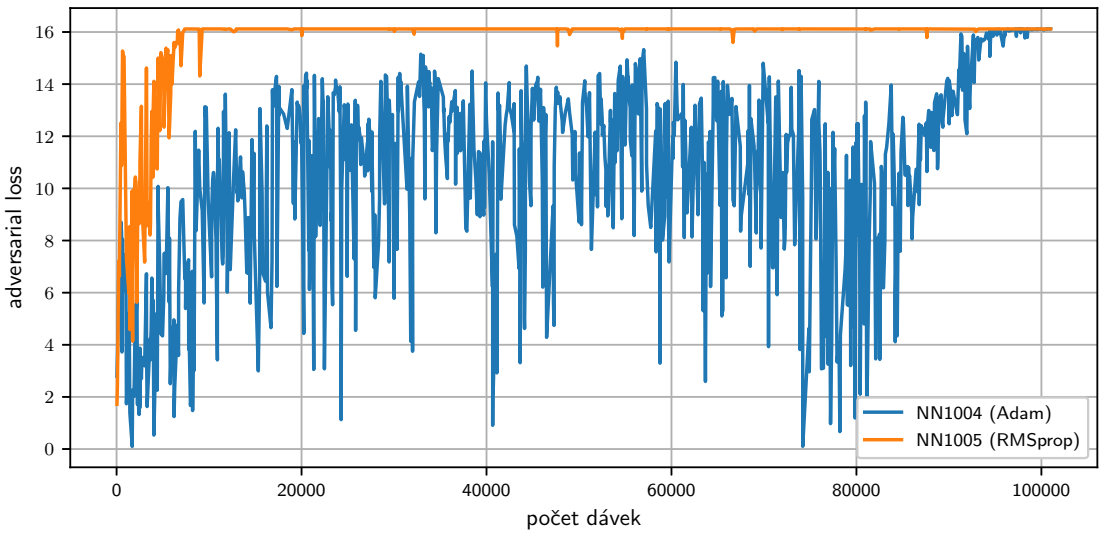
model	PSNR (db)	SSIM
NN1004	25.4944	0.7229
NN1005	26.9638	0.7979
NN1006	**28.3415**	**0.8303**
NN1007	27.9748	0.8227

(b) Medián výsledků SRGAN sítí

Tab. 4.2: Výsledky naměřené na testovací sadě “Set14“

Z tabulek (tab. 4.2a a 4.2b) je možno vyčíst, že nejlepších výsledků v porovnání dle optimalizační funkce dosahuje síť NN1006. Nejlepších výsledků tedy stále dosahuje divergentní optimalizační funkce Adam s velký množstvím dat.

Pokud však porovnáváme síť s nízkým množstvím prvků, jmenovitě NN1004 a NN1005, má síť s optimalizační funkcí RMSprop o $5,76$ procent vyšší medián. Experiment tedy prokazuje lepší výsledky optimalizační funkce RMSprop na malém množství dat.

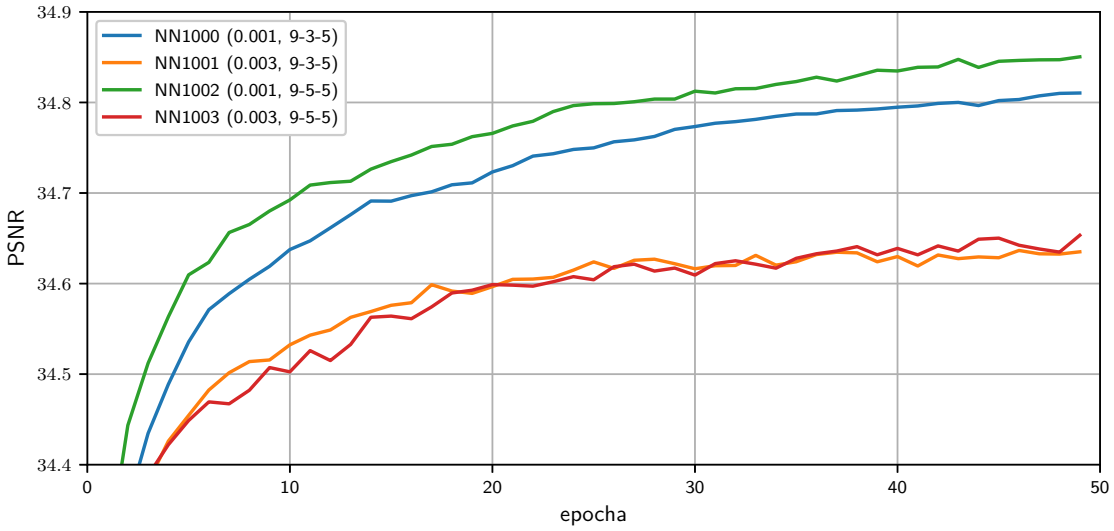


Obr. 4.6: “Adversarial“ ztrátová funkce sítí NN1004 a NN1005

RMSprop také ovlivňuje hodnotu tzv. “adversarial“ ztrátovou funkci SRGAN sítě. Na obrázku 4.6 jsou znázorněny hodnoty ztrátových funkcí sítí NN1004 a NN1004. Již po několika tisících vzorků se stabilizuje kolem hodnoty $16,12$, zatímco Adam je velice divergentní.

4.5.2 Porovnání dle rychlosti učení

Rychlost učení je veličina spojená s optimalizační funkcí. Hodnota rychlosti učení určuje velikost kroku, o který se mění nastavení hodnoty jednotlivých vah. K jejich úpravě dochází během zpětné propagace. Velikost tohoto hyperparametru určuje rychlost učení, fluktuaci konvergence průběhu učícího procesu a pravděpodobnost nalezení globálního minima chyby. Pokud je velikost rychlosti učení příliš malá, učení uvázne v lokálním minimu. Pokud se nastaví hodnota příliš velká, bude síť chaoticky hledat minimum po směru gradientu.



Obr. 4.7: PSNR v průběhu učení sítí NN1000, NN1001, NN1002 a NN1003

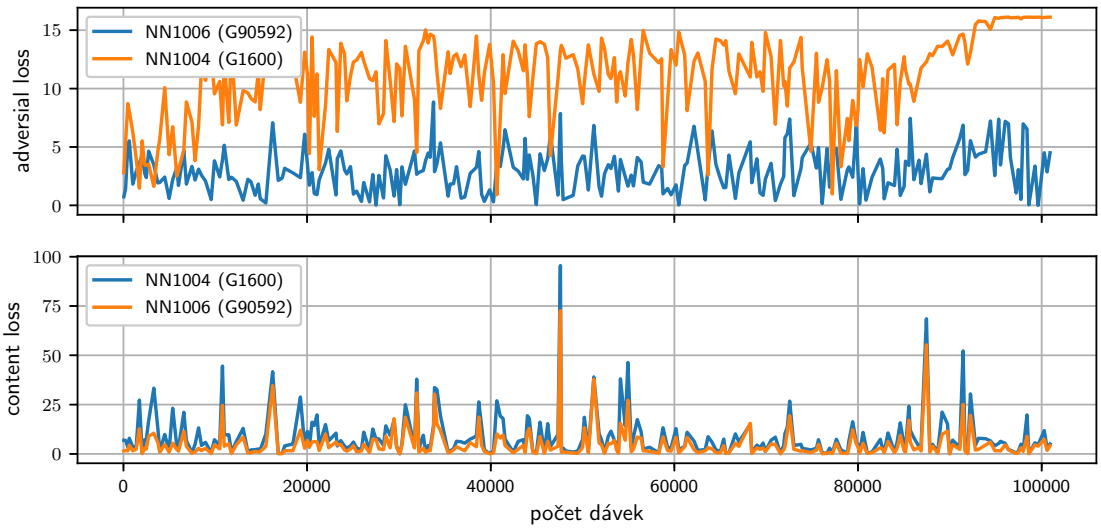
Rychlost učení byla porovnávána na konvolučních sítích v různých kombinacích. Trénované sítě se v experimentu dostaly k podobným výsledkům. Na obrázku 4.7 jsou vykresleny průběhy učení. Lepších výsledků dosahují sítě s hodnotou hyperparametru $0,001$, jmenovitě NN1000 a NN1002. Rozdílem těchto sítí jsou velikosti konvolučního jádra vnitřní vrstvy. NN1000 obsahuje konvoluční vrstvy s velikostí jader 9-3-5. Síť NN1002 obsahuje konvoluce s velikostí jader 9-5-5.

Vzhledem k výsledkům tohoto experimentu nebyla dále testována rychlost učení na SRGAN sítích.

4.5.3 Porovnání dle generátoru

V kapitole trénování (4.4.1) SRGAN sítě, jsou popsány dva generátory (G1600 a G90592). Jejich rozdílnost je pouze v počtu poskytnutých dat. Cílem porovnání je sledovat závislost SRGAN sítí na předtrénovaném generátoru. Trénování GAN

modelů se skládá z vícero částí. První částí je trénování samotného generátoru. Ve druhé a třetí fázi se pak střídavě trénují generátor a diskriminátor. Rozdíl mezi druhou a třetí fází je především v trénování diskriminátoru. Zatímco druhá fáze je krátká a jejím cílem je především trénování diskriminátoru, poslední fáze se zaměřuje na sladění spolupráce těchto dvou částí.



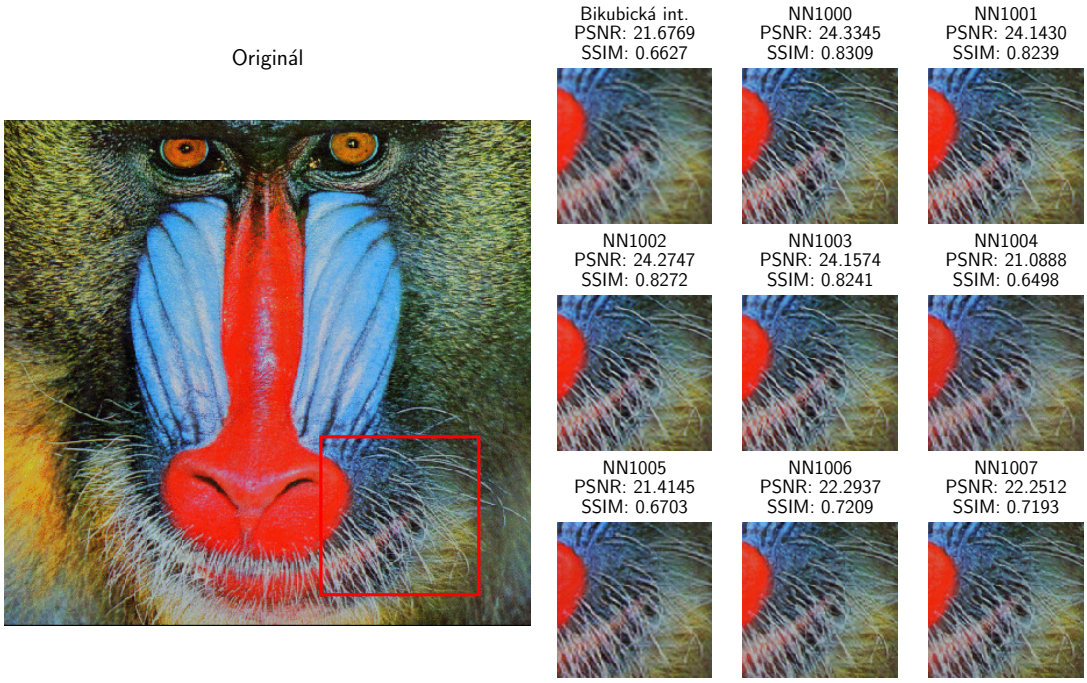
Obr. 4.8: Ztrátové funkce sítí NN1004 a NN1006

Experiment ukázal (obr. 4.8) jasnou závislost mezi předtrénovaným generátorem a úspěchem sítě. Diskriminátor vykazuje u obou generátorů totožný průběh "content" ztrátové funkce. Generátor v případě nízkých vzorků vykazuje průběh obdobný jako na obrázku 4.6 při použití RMSprop, avšak ke konvergenci dochází při mnohem pozdějším kroku. Generátor ke konci trénování začne konvergovat kolem hodnoty $16,1$.

4.5.4 Porovnání výstupů na testovací sadě

Pro porovnání jednotlivých sítí byla použita sada "Set14" popsaná v kapitole 3.6. Z testovací sady byl vybrán obrázek "baboon.png". Jednotlivé výstupy sítí jsou prezentovány v pravé části obrázku 4.9. Výstupy jsou porovnávány jako výřezy. Je tomu tak z toho důvodu, že celkové rozdíly obrázků nejsou dostatečné, aby prezentovaly vlastnosti jednotlivých sítí. Mezi výřezy byla také přidána metoda bikubické interpolace pro další srovnání.

Z obrázku 4.9 je možno vyčíst, že nejlepších výsledků dosahuje síť NN1000 následovaná NN1002. Obě sítě jsou konvoluční a liší se velikostí jádra. Tabulka průměrů



Obr. 4.9: Vizualizace výstupů obrázku baboon.png

4.3a však ukazuje opačné pořadí. Zároveň ukazuje podstatně lepší výsledky pro sítě typu GAN. Z toho plyne jasná závislost dat na nadvzorkovávaném obraze.

Závislost nadvzorkovávaného obrazu a sítě by mohla být směrem dalšího bádání v oblasti nadvzorkování obrazu. Jedním ze směrů může být experimentování s možností obohatit síť o další informaci o nadvzorkovávaném obraze. Pokud by byl obraz nejprve klasifikován a až poté s touto přidanou informací zaslán do sítě, mohlo by to zvýšit kvalitu výstupu.

Druhou možností je trénování sítí na konkrétní objekty. Pokud takto obecně trénované sítě vykazují lepší výsledky na určitém typu vstupů, mohou ukazovat na vhodnost různých řešení pro různé situace. Příkladem může být obrázek paviána, pro který vykazují lepší výsledky konvoluční sítě. Naopak na obrázku obličeje jsou přesnější GAN sítě.

Třetí možností pak může být využití klasifikačního kritéria v kombinaci s výřezem obrazu. Pokud by síť dostala pouze výsek, který má nadvzorkovat, s informací, co se na něm nachází, mohla by pak lépe přizpůsobit generování chybějících částí (v případě GAN). Spojením takto nadvzorkovaných výřezů by se získal kompletní generovaný obraz.

model	PSNR (db)	SSIM	doba gen. (s)
bik. int.	28.3155	0.8566	0.0322
NN1000	30.1461	0.9043	0.8109
NN1001	30.0324	0.9023	0.7486
NN1002	**30.1886**	**0.9046**	1.1731
NN1003	30.0390	0.9030	1.1320
NN1004	24.9772	0.7172	4.4367
NN1005	26.1629	0.8004	4.0475
NN1006	27.9129	0.8438	4.0683
NN1007	27.5834	0.8360	4.2082

(a) Průměry

model	PSNR (db)	SSIM	doba gen. (s)
bik. int.	28.2543	0.8550	0.0293
NN1000	30.0903	**0.9115**	0.8804
NN1001	29.9593	0.9084	0.8645
NN1002	**30.0944**	**0.9115**	1.2950
NN1003	29.9063	0.9088	1.2643
NN1004	25.4944	0.7229	4.9891
NN1005	26.9638	0.7979	4.3858
NN1006	28.3415	0.8303	4.3482
NN1007	27.9748	0.8227	4.4687

(b) Medián

Tab. 4.3: Naměřené výsledky sítí na sadě dat “Set14“

4.6 Naměřené výsledky

Nejlepší síť měla na testovaných datech medián úspěšnosti přibližně $30,0944$ db při metodě PSNR a dvojnásobném zvětšení vstupu. Vzhledem k náročnosti na výpočetní čas nebylo možno v rozsahu práce provést trénování na větším počtu dat, epoch a ani rozsáhlejší hyperparametrickou optimalizaci.

Trénování sítí proběhlo na pronajatém hardwaru u společnosti Google. Konkrétně byla použita platforma Google cloud compute engine, blíže popsaný v kapitole 3.5.2. Konfigurovaný hardware měl 10 virtuálních procesorů, 50 GB paměti RAM a dvě grafické karty Tesla NVIDIA K80 s 11GB paměti. Díky takovéto konfiguraci byl čas trénování v celkovém součtu osm dní, tři hodiny a čtyřicet minut.

5 ZÁVĚR

Práce se zabývá problémem rychlosti učení konkrétních implementací neuronových sítí v závislosti na velikosti trénovací množiny. Aplikační oblastí práce je nadvzorkování digitálního obrazu. Vývoj výpočetního výkonu umožňuje urychlit takové výpočty, které používají masivní paralelismus.

Část práce se věnuje tradičním metodám zpracování obrazu. Jmenovitě se jedná o kompresi obrazu, klasifikaci, opravy a největší část je věnována hlavnímu cíli práce, tedy oblasti nadvzorkování. Dále je popsána teorie hlubokých neuronových sítí. Konkrétní řešení se soustředí na využití konvolučních sítí a generativních kontradiktorních sítí. Hlavním problémem metod DNN je rychlost učení.

V práci bylo navrženo použití jiných optimalizačních funkcí pro nízkou populaci trénovacích množin.

Hlavním přínosem práce je použití odlišné optimalizační funkce pro nízkopopulační problémy, která byla experimentálně ověřena na vybraném případě nadvzorkování obrazu. Použití této funkce v případě SRGAN sítí vykazuje o $5,76$ procent vyšší medián.

Navázat na práci je možno několika různými směry. Jedním ze směrů může být experimentování s možností obohatit síť o další informaci o nadvzorkovávaném obrazu. Pokud by vstup byl obohacen o informaci o obsahu, mohla by být zvýšena kvalitu výstupu. Druhou možností je trénování sítí na konkrétní objekty. Výsledky ukazují vhodnost použití sítí na různé typy obsahu obrazu. Třetí možností je využití klasifikačního kritéria v kombinaci s výřezem obrazu. Pokud by síť dostala pouze výsek s informací, která se na něm nachází, mohla by pak lépe přizpůsobit generování chybějících částí (v případě GAN). Výsledný obraz by byl dosažen spojením SR snímků. Dalším experimentem by bylo použití vyššího faktoru zvětšení. Na práci lze také navázat použitím optimalizační funkce pro jiné oblasti využití generativních kontradiktorních sítí.

LITERATURA

[1] Šíma, J.; Neruda, R.: *Teoretické otázky neuronových sítí*. Praha: Matfyzpress, 1996, ISBN 978-80-85863-18-5, oCLC: 38152197.

[2] Wu, J.: Convolutional neural networks. Červen 2018.
URL <https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf>

[3] Yang, W.; Zhang, X.; Tian, Y.; aj.: Deep Learning for Single Image Super-Resolution: A Brief Review. *arXiv:1808.03344 [cs]*, Srpen 2018, arXiv: 1808.03344.
URL <http://arxiv.org/abs/1808.03344>

[4] Mohammadi, M.; Al-Fuqaha, A.; Oh, J.-S.: Path Planning in Support of Smart Mobility Applications using Generative Adversarial Networks. *arXiv:1804.08396 [cs, stat]*, Duben 2018, arXiv: 1804.08396.
URL <http://arxiv.org/abs/1804.08396>

[5] Ledig, C.; Theis, L.; Huszar, F.; aj.: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *arXiv:1609.04802 [cs, stat]*, Září 2016, arXiv: 1609.04802.
URL <http://arxiv.org/abs/1609.04802>

[6] Wang, X.; Yu, K.; Wu, S.; aj.: ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *arXiv:1809.00219 [cs]*, Leden 2018, arXiv: 1809.00219.
URL <http://arxiv.org/abs/1809.00219>

[7] Mcculloch, W. S.; Pitts, W.: A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY. 1990.

[8] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, ročník 65, č. 6, 1958: s. 386–408, ISSN 1939-1471(Electronic),0033-295X(Print), doi:10.1037/h0042519.

[9] Stumpe, M.; Mermel, C.: Improved Grading of Prostate Cancer Using Deep Learning. Listopad 2018.
URL <http://ai.googleblog.com/2018/11/improved-grading-of-prostate-cancer.html>

[10] Schmidhuber, J.: Deep learning in neural networks: An overview. *Neural Networks*, ročník 61, Leden 2015, ISSN 08936080, doi: 10.1016/j.neunet.2014.09.003.

URL <https://linkinghub.elsevier.com/retrieve/pii/S0893608014002135>

[11] Epstein, B.; Meir, R.: Generalization Bounds For Unsupervised and Semi-Supervised Learning With Autoencoders. *arXiv:1902.01449 [cs, stat]*, Duben 2019, arXiv: 1902.01449.
URL <http://arxiv.org/abs/1902.01449>

[12] Iqbal, M. S.; Kotthoff, L.; Jamshidi, P.: Transfer Learning for Performance Modeling of Deep Neural Network Systems. *arXiv:1904.02838 [cs]*, Duben 2019, arXiv: 1904.02838.
URL <http://arxiv.org/abs/1904.02838>

[13] Pan, S. J.; Yang, Q.: A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, ročník 22, č. 10, Říjen 2010: s. 1345–1359, ISSN 1041-4347, doi:10.1109/TKDE.2009.191.
URL <http://ieeexplore.ieee.org/document/5288526/>

[14] Sayood, K. (editor): *Lossless compression handbook*. Academic Press series in communications, networking and multimedia, Amsterdam ; Boston: Academic Press, 2003, ISBN 978-0-12-620861-0.

[15] Russ, J. C.: *The Image Processing Handbook.* Hoboken: CRC Press, 2011, ISBN 978-1-4398-4063-4, oCLC: 954497165.

[16] Bovik, A. C. (editor): *Handbook of image and video processing*. Amsterdam ; Boston, MA: Elsevier Academic Press, druhé vydání, 2005, ISBN 978-0-12-119792-6.

[17] Hrúz, M.: LBP, HoG. 2015.
URL <http://www.kky.zcu.cz/uploads/courses/mpv/05/materialy05.pdf>

[18] Robust outlier detection using SVM regression. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, ročník 3, Červenec 2004, s. 2017–2022 vol.3, doi:10.1109/IJCNN.2004.1380925.

[19] Pang, Y.; Yuan, Y.; Li, X.; aj.: Efficient HOG human detection. *Signal Processing*, ročník 91, č. 4, Duben 2011: s. 773–781, ISSN 0165-1684, doi:10.1016/j.sigpro.2010.08.010.
URL <http://www.sciencedirect.com/science/article/pii/S0165168410003476>

[20] Bay, H.; Ess, A.; Tuytelaars, T.; aj.: Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, ročník 110, č. 3, Červen 2008: s. 346–359, ISSN 1077-3142, doi:10.1016/j.cviu.2007.09.014.
URL <http://www.sciencedirect.com/science/article/pii/S1077314207001555>

[21] Liu, L.; Ouyang, W.; Wang, X.; aj.: Deep Learning for Generic Object Detection: A Survey. *arXiv:1809.02165 [cs]*, Září 2018, arXiv: 1809.02165.
URL <http://arxiv.org/abs/1809.02165>

[22] Krizhevsky, A.; Sutskever, I.; E. Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, ročník 25, Leden 2012, doi:10.1145/3065386.

[23] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Září 2014, arXiv: 1409.1556.
URL <http://arxiv.org/abs/1409.1556>

[24] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. Prosinec 2015.
URL <https://arxiv.org/abs/1512.03385v1>

[25] Dobeš, M.: *Zpracování obrazu a algoritmy v C#*. Praha: BEN - technická literatura, 2008, ISBN 978-80-7300-233-6, oCLC: 271611311.

[26] Duchi, J.; Hazan, E.; Singer, Y.: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. 2011.

[27] Zeiler, M. D.: ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, Prosinec 2012, arXiv: 1212.5701.
URL <http://arxiv.org/abs/1212.5701>

[28] Mukkamala, M. C.; Hein, M.: Variants of RMSProp and Adagrad with Logarithmic Regret Bounds. *arXiv:1706.05507 [cs, stat]*, Červen 2017, arXiv: 1706.05507.
URL <http://arxiv.org/abs/1706.05507>

[29] Kingma, D. P.; Ba, J.: Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Prosinec 2014, arXiv: 1412.6980.
URL <http://arxiv.org/abs/1412.6980>

[30] Bhandare, A.; Bhide, M.; Gokhale, P.; aj.: Applications of Convolutional Neural Networks. ročník 7, 2016: str. 10.

[31] Povoda, L.: *Dolovanie znalostí z textových dát použitím metód umelej inteligencie*. Dizertační práce, BRNO UNIVERSITY OF TECHNOLOGY, 2018.

[32] Ioffe, S.; Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, Únor 2015, arXiv: 1502.03167.
URL <http://arxiv.org/abs/1502.03167>

[33] Zhu, J.-Y.; Park, T.; Isola, P.; aj.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv:1703.10593 [cs]*, Březen 2017, arXiv: 1703.10593.
URL <http://arxiv.org/abs/1703.10593>

[34] Antipov, G.; Baccouche, M.; Dugelay, J.-L.: Face Aging With Conditional Generative Adversarial Networks. *arXiv:1702.01983 [cs]*, Únor 2017, arXiv: 1702.01983.
URL <http://arxiv.org/abs/1702.01983>

[35] Schlegl, T.; Seeböck, P.; Waldstein, S. M.; aj.: Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. *arXiv:1703.05921 [cs]*, Březen 2017, arXiv: 1703.05921.
URL <http://arxiv.org/abs/1703.05921>

[36] Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; aj.: Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, Červen 2014, arXiv: 1406.2661.
URL <http://arxiv.org/abs/1406.2661>

[37] Yang, C.-Y.; Ma, C.; Yang, M.-H.: Single-Image Super-Resolution: A Benchmark. In *ECCV*, 2014, doi:10.1007/978-3-319-10593-2_25.

[38] He, K.; Zhang, X.; Ren, S.; aj.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*, Únor 2015, arXiv: 1502.01852.
URL <http://arxiv.org/abs/1502.01852>

[39] Jolicoeur-Martineau, A.: The relativistic discriminator: a key element missing from standard GAN. *arXiv:1807.00734 [cs, stat]*, Červenec 2018, arXiv: 1807.00734.
URL <http://arxiv.org/abs/1807.00734>

[40] van Meel, J. A.; Arnold, A.; Frenkel, D.; aj.: Harvesting graphics power for MD simulations. *Molecular Simulation*, ročník 34, č. 3, Březen 2008: s. 259–266,

ISSN 0892-7022, 1029-0435, doi:10.1080/08927020701744295, arXiv: 0709.3225.
URL <http://arxiv.org/abs/0709.3225>

[41] Agustsson, E.; Timofte, R.: NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Honolulu, HI, USA: IEEE, Červenec 2017, ISBN 978-1-5386-0733-6, doi:10.1109/CVPRW.2017.150.
URL <http://ieeexplore.ieee.org/document/8014884/>

[42] Lai, J.: LapSRN. 2017.
URL <http://vllab.ucmerced.edu/wlai24/LapSRN/results/LapSRN_results.zip>

[43] Lai, W.-S.; Huang, J.-B.; Ahuja, N.; aj.: Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks. *arXiv:1710.01992 [cs]*, Říjen 2017, arXiv: 1710.01992.
URL <http://arxiv.org/abs/1710.01992>

[44] Lehmann, E. L.; Casella, G.: *Theory of point estimation*. Springer texts in statistics, New York: Springer, druhé vydání, 1998, ISBN 978-0-387-98502-2.

[45] Nemethova, O.; Ries, M.; Zavodsky, M.; aj.: PSNR-Based Estimation of Subjective Time-Variant Video Quality for Mobiles. 2019: str. 5.

[46] Wang, Z.; Bovik, A.; Sheikh, H.; aj.: Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, ročník 13, č. 4, Duben 2004: s. 600–612, ISSN 1057-7149, doi:10.1109/TIP.2003.819861.
URL <http://ieeexplore.ieee.org/document/1284395/>

[47] Dong, C.; Loy, C. C.; He, K.; aj.: Image Super-Resolution Using Deep Convolutional Networks. *arXiv:1501.00092 [cs]*, Prosinec 2014, arXiv: 1501.00092.
URL <http://arxiv.org/abs/1501.00092>

[48] Gruber, M.: Implementation of SRGAN in Keras. Try at: www.fixmyphoto.ai: MathiasGruber/SRGAN-Keras. Březen 2019, original-date: 2018-04-08T13:19:38Z.
URL <https://github.com/MathiasGruber/SRGAN-Keras>

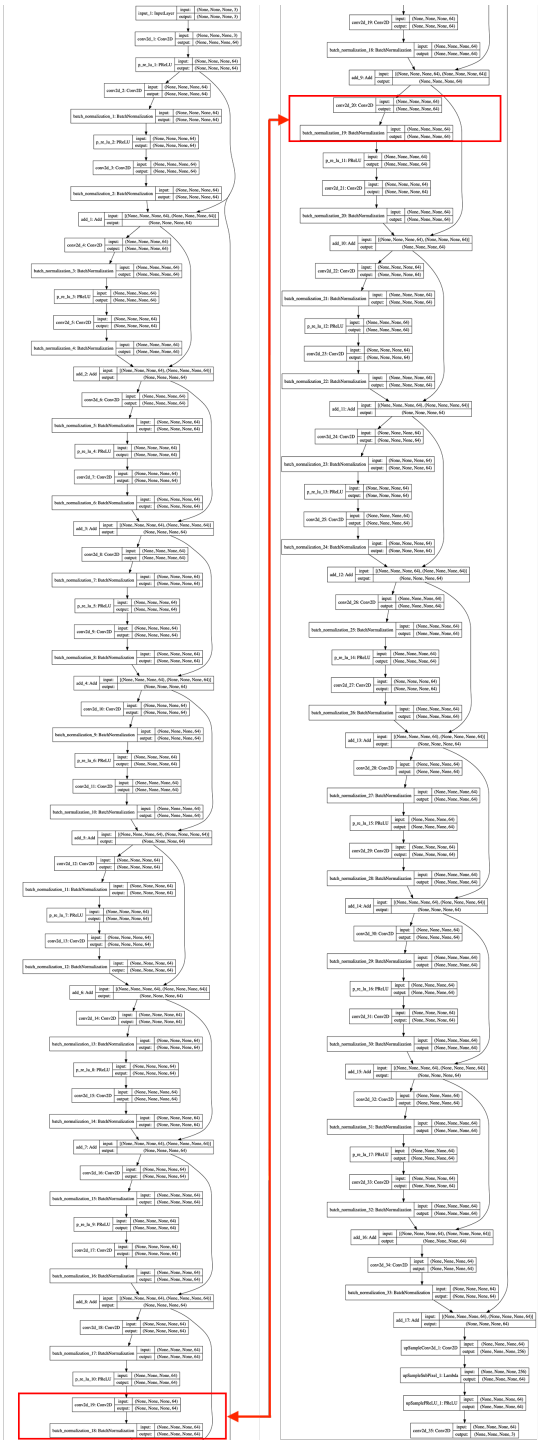
SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

DNN	Deep neuron network
CNN	Convolution neuron network
SRCNN	Super resolution convolution neuron network
GAN	Generativní kontradiktorní síť
SRGAN	Super resolution generative adversary network
ESRGAN	Enhanced super resolution generative adversary network
LR	Low resolution
HR	High resolution
SR	Super resolution
MSE	Střední kvadratická chyba
PSNR	Špičkový odstup signálu k šumu
BN	Batch normalization
CPU	Central processing unit
GPU	Graphic processing unit
TPU	Tensor processing unit
CUDA	Compute Unified Device Architecture
PSNR	Peak signal-to-noise ratio
SSIM	Structural similarity
VGG	Visual Geometry Group
HOG	Histograms of Oriented Gradients
SVM	Suport vector machines
SURF	Speeded-Up Robust features

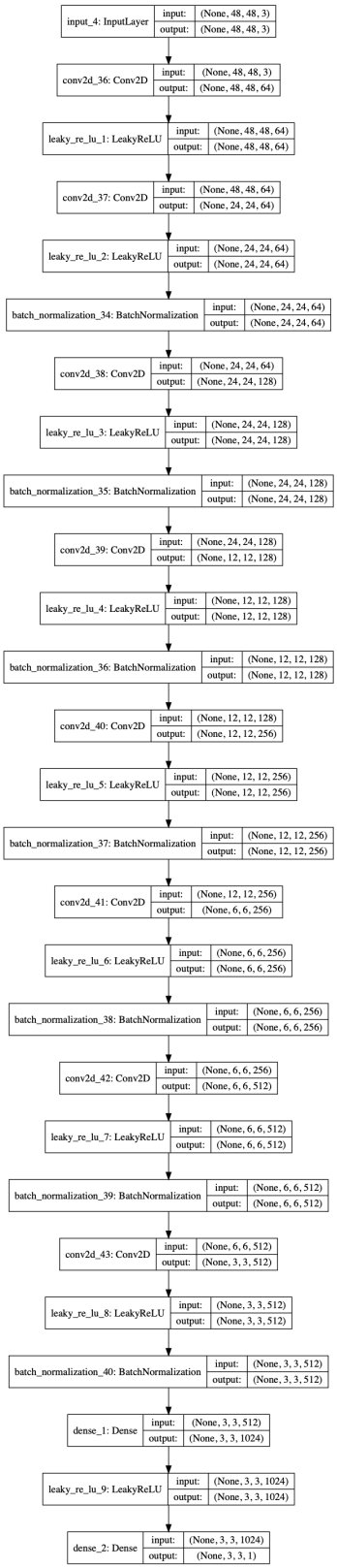
SEZNAM PŘÍLOH

A Přílohy **51**
A.1 Datové médium 53

A PŘÍLOHY



Obr. A.1: Model generátoru sítě GAN



Obr. A.2: Model diskriminátoru sítě GAN

A.1 Datové médium

- složka `kex-trainer` obsahuje zdrojový kód použitý pro trénování konvolučních sítí,
  - `datagenerator.py` je datový generátor výseků,
  - `experiment.py` je zdrojový kód knihovny kex,
  - `model.py` je zdrojový kód pro kex experiment,
  - `run.py` zdrojový kód spouštění kex experimentů,
  - `tf_device_test.py` je soubor pro testování podpory grafické karty,
- složka `NNs` obsahuje data trénovaných sítí,
  - `NN1000-NN1003` jsou složky konvolučních sítí,
    - složka `tensorboard-epoch` obsahuje log dat pro Tensorboard,
    - `*_exp_code.py` je zdrojový kód modelu konkrétní sítě,
    - `*_mod_summ.txt` je textový souhrn modelu,
    - `*-mod_*.h5` je uložený model s váhami v poslední epoše,
    - `*-mod_best.h5` je uložený model s váhami v nejlepší epoše,
    - `*-mod_plot.png` je grafické zobrazení modelu,
  - `NN1004-NN1007` jsou složky generativních kontradiktorních sítí,
    - složka `datalogs` obsahuje data pro Tensorboard,
    - složka `dataweights` obsahuje uložené váhy modelů,
      - `SRGAN_imagenet_discriminator_2X.h5` obsahuje váhy diskriminátoru,
      - `SRGAN_imagenet_generator_2X.h5` obsahuje váhy generátoru,
      - `SRResNetimagenet_2X.h5` obsahuje váhy pro přenesené učení,
    - složka `libs` obsahuje zdrojový kód sítě,
      - `srgan.py` je zdrojový kód sítě,
      - `util.py` je zdrojový kód datového generátoru,
    - `Example_Usage.ipynb` je ukázka zdrojového kód pro generování výsledků,
    - `log.log` obsahuje záznam průběhu trénování sítě,
    - `test_generator.py` je zdrojový kód pro test funkčnosti přepsaného generátoru,
    - `train.py` je spouštěcí skript trénování,
- složka `sr_output` obsahuje vygenerované SR a rozdílové obrázky,
  - `out_*` je složka jednotlivých sítí,
    - `diff/*.png` jsou soubory rozdílů bikubického nadvzorkování a SR v šedi,
    - `ssim/*.png` jsou soubory rozdílu SR a originálního obrázku funkcí SSIM,

        * `*.png` jsou soubory SR,
- složka `srgan-model*.png` jsou grafické modely sítě SRGAN,
- `srgan-trainertrain.py` je skript pro paralelní trénování SRGAN,
- složka `tesing` obsahuje soubory pro testování,
  - `Set14*.png` jsou obrázky testovací množiny,
  - `generate_output.py` je skript pro vygenerování SR obrázků.

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma: Kombinovaná
Obor/komb.: Aplikovaná informatika (ai2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Dobrovolný Michal	Strahcotice 138, Strachotice	I1700318

TÉMA ČESKY:

Metody pro zlepšování obrazu s pomocí neuronových sítí

TÉMA ANGLICKY:

Methods for image improvements with the use of neural networks

VEDOUCÍ PRÁCE:

Ing. Karel Mls, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Možnosti optimalizace různých typů umělých neuronových sítí s omezeným množstvím trénovacích prvků při nadvzorkování digitálního obrazu.

Osnova:
1. Úvod
2. Umělé neuronové sítě
3. Úpravy obrazových dat
4. Aplikace hlubokých neuronových sítí
5. Výsledky a závěr

SEZNAM DOPORUČENÉ LITERATURY:

Yang, W.; Zhang, X.; Tian, Y.; aj.: Deep Learning for Single Image Super-Resolution, 2018
Russ, J. C.:The Image Processing Handbook. 2011

Podpis studenta: ..... **Datum:** .....

Podpis vedoucího práce: ..... **Datum:** .....