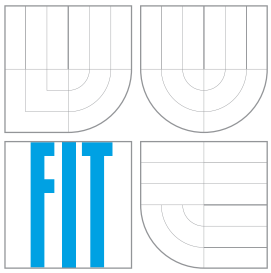
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

GENEROVÁNÍ OBRAZU POMOCÍ KONVOLUČNÍCH NEURONOVÝCH SÍTÍ

LEARNING TO GENERATE IMAGES WITH CONVOLUTIONAL NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN KOHÚT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL HRADIŠ, Ph.D.

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Kohút Jan**
Obor: Informační technologie
Téma: **Generování obrazu pomocí konvolučních neuronových sítí**
Learning to Generate Images with Convolutional Neural Networks
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy neuronových sítí a jejich učení.
2. Vytvořte si přehled o současných metodách pro generování obrázků objektů pomocí konvolučních neuronových sítí.
3. Navrhněte konkrétní způsob generování obrázků pomocí konvolučních neuronových sítí.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dosovitskiy et al.: Learning to Generate Chairs with Convolutional Neural Networks. CVPR 2015.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá navržením a analýzou konvolučních neuronových sítí generujících obrázky znaků na základě jejich parametrů. Parametry znaků jsou typ znaku, font, barva znaku, barva pozadí, translace a rotace. Neuronové sítě si vytvořily mnohorozměrnou reprezentaci jednotlivých parametrů. V rámci těchto reprezentací existují podobné vztahy jako v rámci samotných parametrů. Neuronové sítě generují znaky s novými hodnotami prametrů na základě interpolace mezi naučenými hodnotami parametrů. Neuronové sítě jsou schopné generalizace problému generování obrazu.

Abstract

The aim of this Bachelor's thesis is to design and analyze convolutional neural networks generating images of characters based on their parameters. Parameters of characters are type of char, font, colour of character, background colour, translation and rotation. Neural networks have created multidimensional representation of each parameter. Relations inside these representation are similar to relations inside parameters. Neural networks generate characters with new values of parameters based on interpolation between learned values of parameters. Neural networks are capable to generalize problem of generating images.

Klíčová slova

Neuronové sítě, konvoluční neuronové sítě, generování obrazu, struktury neuronových sítí, embedding, interpolace

Keywords

Neural networks, convolutional neural networks, image generation, neural networks structures, embedding, interpolation

Citace

KOHÚT, Jan. *Generování obrazu pomocí konvolučních neuronových sítí.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Hradiš Michal.

Generování obrazu pomocí konvolučních neuronových sítí

Prohlášení

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

.....
Jan Kohút
17. května 2016

Poděkování

Děkuji svému vedoucímu Ing. Michalu Hradišovi, Ph.D. za nápad tématu této práce a cenné rady při jejím řešení. Děkuji za poskytnutí výpočetních zdrojů projektem CESNET LM2015042.

Obsah

Kapitola 1

Úvod

Neuronové sítě dosahují velmi dobrých výsledků na poli zpracování řeči a obrazu. Z praktického hlediska je většina pozornosti orientována na klasifikaci. Neuronové sítě jsou ovšem schopné aproximovat téměř jakoukoliv funkci. Proces klasifikace lze například otočit. Neuronová síť pak místo popisu obrazu generuje obraz na základě jeho popisu.

Tato bakalářská práce se zabývá generováním obrazu pomocí konvolučních neuronových sítí. Výstupem těchto sítí jsou obrázky znaků s různými parametery. Parametry jsou typ znaku, font, barva znaku, barva pozadí, translace a rotace. Zadáváním těchto parametrů na vstupy sítí se generují příslušné obrázky znaků.

Na začátku této práce se věnuji samotnému problému struktury těchto sítí, zejména pak podobě vstupní vrstvy a konvoluci. Dále se věnuji analýze monohorozměrných reprezentací jednotlivých parametrů, které si síť vytvořila. Závěr práce se zabývá schopností sítě generovat znaky s novými hodnotami parametrů na základě interpolace mezi naučenými hodnotami parametrů.

Kapitola 2

Výchozí poznatky

Neuronová síť je matematický model sloužící pro aproximaci funkcí. Tento model je inspirován biologickými neuronovými sítěmi, které se nachází v mozku (odtud pochází název tohoto modelu). Model se skládá z navzájem propojených částí zvaných neurony. Neuron může mít neomezený počet vstupů a má právě jeden výstup, který může být napojen na více vstupů dalších neuronů. Spoje mezi neurony jsou váhovány. Svůj výstup neuron spočítá sečtením všech vstupů, tato hodnota může být dodatečně zpracována tzv. aktivační funkcí[1]. Tato funkce má práh, což je hodnota, která se přičítá ke vstupu do funkce.

Aby síť dobře aproximovala příslušnou funkci musí být dostatečně velká a její váhy (spoje neuronů a prahy aktivačních funkcí) musí mít správné hodnoty (hodnoty, které zaručí dobrou aproximaci funkce). Nalezení příslušných hodnot vah se provádí zpravidla pomocí tzv. trénování.

Existují různé přístupy k trénování. V této práci je použit „supervised learning“. Na začátku trénování se váhy náhodně inicializují (případně podle vybraného algoritmu). Následně je spočítán dopředný průchod sítí (pro zadané vstupy), tzn. hodnoty výstupních neuronů. Tyto hodnoty jsou zpracovány pomocí chybové funkce (např. střední kvadratická odchylka), která srovnává výstupy sítě s požadovanými výstupy (výstupy, které jsou správné pro zadaný vstup). Chybová funkce je závislá na všech vahách sítě. Minimalizací této funkce je dosaženo trénování. Využívá se přístupu stochastic gradient descent a algoritmu zpětné propagace chyby [6].

Neuronové sítě lze uplatnit například pro klasifikaci obrazu. Aproximovanou funkcí je pak funkce, která dostává na vstup obraz a jejíž výstup je popis obrazu. Při klasifikaci obrazu se navíc používají konvoluční filtry, které jsou z obrazu schopny vytáhnout podstatné informace (např. hrany). Tyto filtry jsou součástí sítě a jejich váhy jsou součástí trénování. Aplikací konvolučních filtrů na obraz vznikají tzv. příznakové mapy. Na tyto mapy se dále mohou aplikovat konvoluční filtry nebo jsou dány na vstup plně propojeným vrstvám (vrstvy, které se skládají pouze z neuronů). Takovéto sítě nazýváme konvolučními neuronovými sítěmi. Tyto sítě dosahují velmi dobrých výsledků při klasifikaci obrazu i při vysokém počtu klasifikovaných objektů [7].

2.1 Generování obrazu

Způsob trénování, zmíněný na začátku této kapitoly, umožňuje vytvořit závilost trénovacích dat na jejich popisech. Pro generování obrazu to znamená možnost vytvořit přímý generátor

[1] Jedná se zpravidla o jednoduché funkce typu sigmoida, arcus tangens, sinus atd.

obrazu. Síť je tedy schopná naučit se generovat obrázky na základě jejich popisu.

Neuronové sítě lze také trénovat s cílem naučit se rozložení, ze kterého byla vygenerována trénovací data. V případě generování obrazu umožňují takové sítě generovat náhodné obrázky z rozložení trénovacích obrázků. Příkladem takových sítí jsou tzv. „Restricted Boltzmann Machines“ a „Deep Boltzmann Machines“ [8]. Tyto sítě jsou schopny naučit se generovat obrázky pouze na základě obrázků v trénovacím datasetu. Výhodou je, že tyto sítě nepotřebují popisy obrázků pro trénování. Takové trénování se nazývá „unsupervised learning“. Jiným příkladem, jak dosáhnout natrénování sítě, která umožňuje náhodné generování obrázků s rozložení, jsou tzv. „Generative Adversarial Networks“ [4]. Tyto sítě se skládají z generativní a diskriminativní části. Úkolem diskriminativní části je odlišit skutečný obrázek (obrázek trénovacího datasetu) od obrázku vygenerovaného generativní částí. Generativní část se snaží „oklamat“ diskriminativní část, tím pádem se učí generovat obrázky podobné obrázkům v datasetu. Pro trénování těchto sítí se využívá přístup popsaný na začátku této kapitoly („supervised learning“).

V této práci se zabývám sítěmi, které generují obrázky na základě jejich popisu. Tato podkapitola popisuje práci Dosovitského a kol. [2], ve které autoři popisují schopnost těchto sítí generalizovat problém generování obrazu. Základní myšlenka této práce je otočit uspořádání konvoluční neuronové sítě, které je popsáno na začátku této kapitoly. To znamená, že síť začíná plně propojenými vrstvami a na ty navazují vrstvy konvoluční. Na vstup takovéto sítě nejsou zadávány obrázky, které má síť rozpoznávat, ale popisy obrázků, které má síť generovat. Síť tedy dostane na vstup např. typ objektu, jeho barvu, pozici a natočení v prostoru. Na výstupu pak generuje obrázek, na kterém lze vidět typ objektu s danými parametry.

Dataset, který byl použit pro trénování sítí, se sestává z dvourozměrných projekcí trojrozměrných objektů několika tříd (židle, stoly a auta). Výsledné dvojrozměrné projekce jsou pak popsány třídou objektu a zorným úhlem (volitelné jsou pak další parametry jako barva, jas, saturace, velikost atd.). Tento popis je pak vkládán na vstup sítě a výstupem jsou požadované dvojrozměrné projekce.

Práce ukazuje schopnost sítě generalizovat problém generování obrazu v mnoha ohledech. Síť je schopná kvalitní interpolace mezi naučeným zornými úhly, to znamená, že je schopná vygenerovat zorné úhly, které nebyly v trénovacím datasetu. Interpolace je také možná mezi objekty stejné i různé třídy, v tomto případě dojde k věrohodnému morfování objektů. Síť je také schopná přenosu informace o zorném úhlu mezi objekty stejné i různé třídy. Například pokud jsou v datasetu stoly s určitým natočením, které židle nemají, síť je ve výsledku schopna generovat židle i v tomto natočení. Generování dosud neviděných natočení, mimo zadaný rozsah (extrapolace), je také možné. Dalším důkazem generalizace je fakt, že trénovaná síť má okolo 32 milionů parametrů, zatímco pixelů v trénovacích datech je okolo 400 milionů. Síť se tedy nemůže naučit všechny obrázky samostatně. Provedena byla také analýza různých částí sítě. Pomocí aktivace samostatných neuronů bylo zjištěno, že tyto neurony mají vliv na určité parametry generovaného obrazu, např. na velikost objektu atd.

Struktura sítě, která dosáhla výše zmíněných výsledků je následující. Síť začíná třemi vstupními vrstvami, každá s těchto vrstev zpracovává různé parametry výsledné dvojrozměrné projekce (třídu, zorný úhel a volitelné parametry). Tyto vrstvy se konkatenují[2] do jedné vrstvy, jejíž výstup je napojen na následující vrstvu. Zbytek sítě tvoří dvě plně propojené vrstvy, RGB část generující RGB obrázek a segmentační část generující segmentační

[2]Konkatenace vrstev nijak neovlivní výstupy jednotlivých vrstev, tyto výstupy se pouze spojí, jsou tak „vedle sebe“ v jedné vrstvě

masku[3]. Každá z těchto dvou částí začíná jednou plně propojenou vrstvou a za ní následují čtyři konvoluční vrstvy. Aktivační funkce u všech vrstev, kromě výstupní (kde není žádná), je LReLU (Leaky Rectified Linear Unit)[4].

Samotný výstup sítě zajišťuje poslední konvoluční vrstva. Počet jader v této vrstvě je roven počtu barevných kanálů ve výsledném obrázku. Každé jádro tedy sestaví příznakovou mapu, která bude představovat jeden barevný kanál výsledného obrázku. Výstupem sítě bude matice o rozměrech $W \times H \times D$, kde W a H jsou šířka a výška obrázku a D je počet barevných kanálů.

Výše je uvedeno rozdělení sítě na dvě části. Každá s těchto části má při trénování svou chybovou funkci. U první části generující RGB obrázky se jedná vždycky o střední kvadratickou odchylku, u druhé části (generující segmentační masku) pak bylo trénování zkoušeno i s funkcí softmax. Výstup chybové funkce druhé části byl navíc různě váhován. Learning rate byl nastaven na 0.0005 po prvních 200000 iterací a poté každých 100000 iterací dvakrát snížen, celkem bylo provedeno 500000 iterací. Při jedné iteraci bylo zpracováno 128 vstupů (popisy obrázků).

Problémy uspořádání sítě. Uspořádání sítě způsobuje dva problémy. Prvním z nich je přechod mezi plně propojenou a konvoluční vrstvou. Zatímco u sítí, které začínají konvolučními vrstvami a pokračují plně propojenými je přechod mezi příznakovými mapami a plně propojenou vrstvou vyřešen tak, že každý pixel příznakové mapy je dán na vstup každého neuronu této vrstvy, při opačném uspořádání sítě je třeba provést konvoluci nad výstupem plně propojené vrstvy. Aby se mohla provést konvoluce, obsahuje RGB i segmentační část jednu plně propojenou vrstvu, jejíž výstupy jsou prezentovány ve tvaru trojrozměrné matice. Vzniknou tak příznakové mapy, nad kterými lze provádět konvoluci. Pokud má plně propojená vrstva 1000 výstupů (neuronů), jde tento výstup reprezentovat jako matice o rozměrech $1000 \times 1 \times 1$. Rozměry této matice lze změnit například na $10 \times 10 \times 10$ (10 příznakových map o rozměrech 10×10) a následně na ní provést konvoluci.

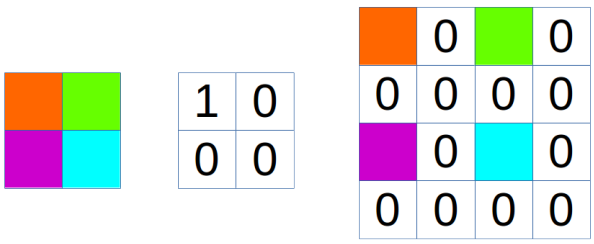
Druhý problém představuje navyšování rozměrů jednotlivých příznakových map.[5] Tento problém je vyřešen použitím tzv. „unpoolingu“. Princip unpoolingu pro jádro o velikosti 2×2 a krok 2 je znázorněn na obrázku 2.1. Čtverec nalevo představuje původní příznakovou mapu, další čtverec pak jádro a poslední čtverec novou příznakovou mapu sestavenou unpoolingem. Jádro má v levém horním rohu váhu 1, ostatní pozice mají váhu 0. Nová příznaková mapa se vytvoří násobením každé hodnoty původní příznakové mapy do tohoto jádra. Výsledný snímek má tedy dvakrát větší rozměry než původní a obsahuje stejné hodnoty s tím, že každá z nich je obalena nulovými. Na těchto nových příznakových mapách se následně provádějí konvoluce. Unpooling se provádí mezi všemi konvolučními vrstvami a také mezi trojrozměrnou reprezentací plně propojené vrstvy a první konvoluční vrstvou.

Výše popsaná práce poukazuje na schopnost konvolučních neuronových sítí generovat obraz a rovněž se naučit vztahy mezi generovanými objekty a jejich parametry. Na základě této práce jsem se rozhodl navrhnout vlastní strukturu konvoluční neuronové sítě generující obraz.

[3] Výstup druhé plně propojené vrstvy je zdvojen, čili do každé části jde totožný vstup

[4] Funkce LReLU je téměř totožná s funkcí ReLU, pouze pro záporné hodnoty není její výstup nulový, ale velmi se jí blíží.

[5] U standardních konvolučních vrstev se tento rozměr snižuje pomocí poolingu. Pooling rozměry snižuje na základě jádra a většího kroku. Například příznakovou mapu o rozměru 4×4 lze zmenšit na rozměr 2×2 použitím jádra o rozměru 2×2 a kroku 2.



Obrázek 2.1: Princip unpoolingu pro jádro o velikosti 2×2 a krok 2. Čtverec nalevo představuje původní příznakovou mapu, další čtverec pak jádro a poslední čtverec novou příznakovou mapu sestavenou unpoolingem. Nová příznaková mapa se vytvoří násobením každé hodnoty původní příznakové mapy do tohoto jádra. Výsledný snímek má tedy dvakrát větší rozměry než původní a obsahuje stejné hodnoty s tím, že každá z nich je obalena nulovými.

Kapitola 3

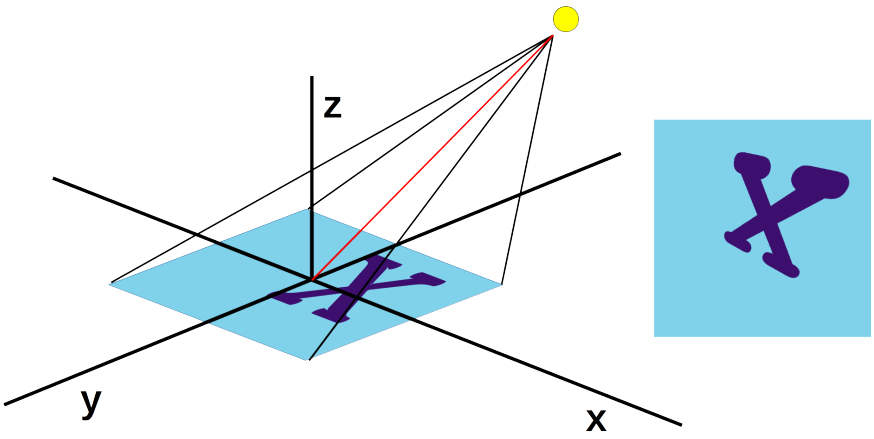
Znaky

Předtím, než se zaměřím na konstrukci mnou navržené sítě popíši dataset, který byl využit pro její trénování. Jako dateset jsem si zvolil obrázky znaků ascii tabulky s různými parametry. Vygenerován byl pomocí skriptu napsaném v jazyce Python 2.7, za použití knihoven OpenCV [1] a PyGame [9].

Parametry znaku jsou typ znaku, font, barva znaku, barva pozadí, translace a rotace. Znaky jsou v rozmezí indexů 33–126 ascii tabulky („!“–„~“). Všechny znaky mají stejnou velikost fontu. Je použito 139 fontů různých typů (tučné, kurzíva, patkové, bezpatkové atd.). Geometrické transformace se provádí v rámci kartézské soustavy souřadnic v prostoru. Samotný obrázek znaku leží v rovině xy. Translace znaku je možná pomocí posunu linky. Všechny znaky bez geometrické transformace leží na stejné lince. Pozice výchozí linky (bez translace) byla zvolena tak, aby malé znaky abecedy, bez vysokých svislých nožiček, ležely ve středu obrázku. Tato linka se může posunovat o $\frac{1}{42}$, $\frac{2}{42}$, $\frac{3}{42}$ nebo $\frac{4}{42}$ velikosti obrázku v rámci osy x a y, kladným i záporným směrem. Existuje tedy 81 různých translací, včetně nulové. Rotace znaku je realizována pomocí virtuální kamery, nacházející se v konstantní vzdálenosti od středu obrázku. Tato kamera je schopna rotace ve vlastní ose a také kolem osy x a y. Rotace jsou v rozmezí -20–$20°$po pěti stupních (729 různých rotací).

Na obrázku 3.1 je znázorněn princip geometrické transformace znaku (vlevo) spolu s jejím výsledkem (vpravo). Je znázorněna translace znaku „X“ v kladném směru osy x a y. Virtuální kamera realizující rotaci (žluté kolečko) je rotována kolem osy x a y o kladný úhel. Červená osa znázorňuje vlastní osu kamery, ve které kamera může rotovat. Rotace se provádí jako první operace, jelikož kamera rotuje celý obrázek podle středu. Znak má v této fázi nulovou translaci. Nad znakem s rotací je následně provedena translace do cílové pozice.

Pro trénování a experimenty se sítěmi byly vytvořeny celkem čtyři typy datasetů. Každý typ datasetu má jinou geometrickou složitost. První typ datasetu obsahuje translace i rotace (T + R, úplná geometrická složitost), druhý typ pouze translace (T), třetí typ pouze rotace (R) a čtvrtý typ neobsahuje žádnou geometrickou transformaci (N). Všechny typy mají parametr typ znaku, font, barvu znaku a barvu pozadí znaku. Každý dataset obsahuje 200000 obrázků. Obrázky mají tři barevné kanály RGB. Hodnoty všech barevných kanálů obrázků jsou normalizovány mezi 0 a 1. Existují dvě velikosti datasetů, a to 16×16 a 32×32. První velikost existuje pouze jako první typ o plné geometrické složitosti. Druhá velikost existuje ve všech čtyřech uvedených typech. Ukázky obrázků ze všech typů datasetů jsou na obrázku 3.2 (před provedením normalizace). Každá řada představuje jeden typ datasetu, tak jak byly vyjmenovány výše. Následující kapitola popisuje struktury sítí, které byly na těchto datasetech trénovány.



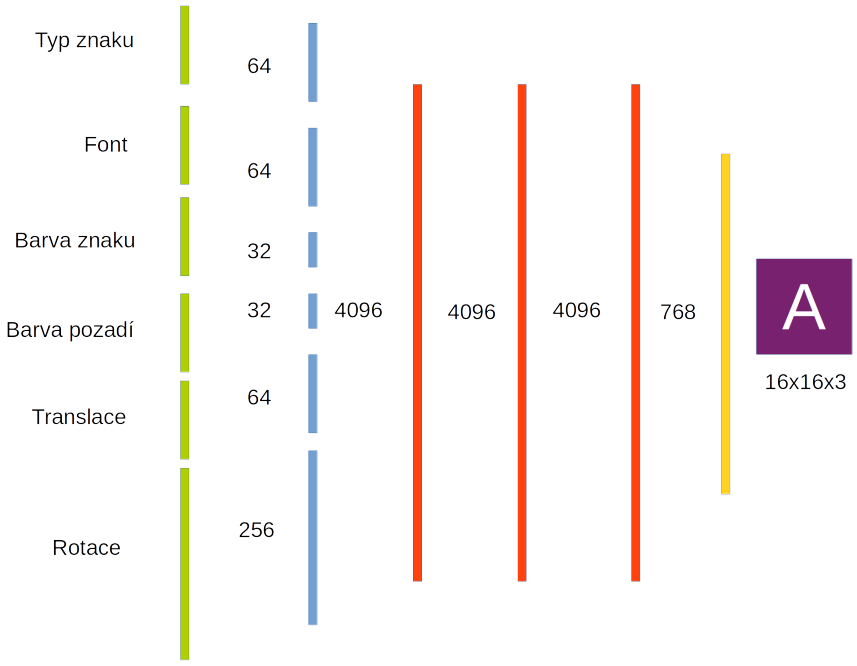
Obrázek 3.1: Princip geometrické transformace znaku (vlevo), spolu s jejím výsledkem (vpravo). Je znázorněna translace znaku „X“ v kladném směru osy x a y. Virtuální kamera realizující rotaci (žluté kolečko) je natočena kolem osy x a y o kladný úhel. Červená osa znázorňuje vlastní osu kamery, ve které kamera může rotovat.

Obrázek 3.2: Ukázka obrázků v datasetech o různé geometrické složitosti. Dataset na prvním řádku obsahuje translace i rotace (T + R), na druhém pouze translace (T), na třetím pouze rotace (R) a na čtvrtém neobsahuje žádnou geometrickou transformaci (N).

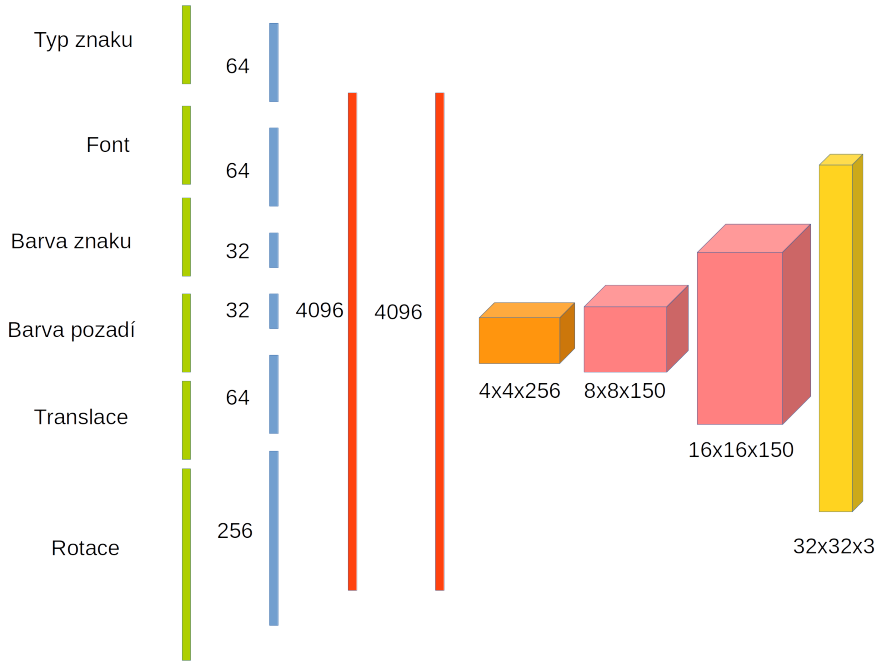
Kapitola 4

Struktury sítí

V této kapitole popisuji tři struktury sítí. Na obrázku 4.1 je zobrazena první ze struktur. Zelená vrstva znázorňuje vstup do sítě. Každý ze vstupů je napojen na příslušnou modrou vstupní vrstvu sítě. Jednotlivé vstupní vrstvy jsou konkatenovány (jak bylo popsáno v podkapitole 2.1) do jedné vrstvy, jejíž výstup je napojen na vstup následující červené vrstvy. Červené vrstvy představují tři plně propojené skryté vrstvy. Výstup žluté výstupní vrstvy je reprezentován jako trojrozměrná matice $16 \times 16 \times 3$, tzn. obrázek. Čísla u jednotlivých vrstev představují počet neuronů dané vrstvy.



Obrázek 4.1: Struktura neuronové sítě. Zelená vrstva znázorňuje vstup do sítě. Každý ze vstupů je napojen na příslušnou modrou vstupní vrstvu sítě. Jednotlivé vstupní vrstvy jsou konkatenovány do jedné vrstvy, jejíž výstup je napojen na vstup následující červené vrstvy. Červené vrstvy představují tři skryté vrstvy. Výstup žluté výstupní vrstvy je reprezentován, jako trojrozměrná matice $16 \times 16 \times 3$, tzn. obrázek. Čísla u jednotlivých vrstev představují počet neuronů dané vrstvy.



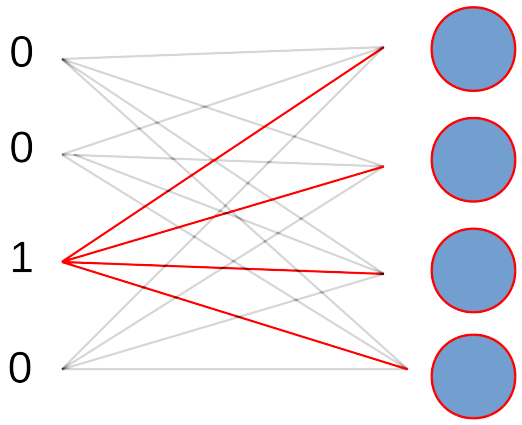
Obrázek 4.2: Struktura neuronové sítě. Zelená vrstva znázorňuje vstup do sítě. Každý ze vstupů je napojen na příslušnou modrou vstupní vrstvu sítě (ta, co je za ním). Jednotlivé vstupní vrstvy jsou konkatenovány do jedné vrstvy, jejíž výstup je napojen na vstup následující červené vrstvy. Červené vrstvy představují dvě skryté vrstvy. Oranžová krychle znázorňuje reprezentaci výstupu druhé skryté vrstvy ve tvaru trojrozměrné matice $4 \times 4 \times 256$. Následují krychle reprezentující konvoluční vrstvy. Poslední žlutá konvoluční vrstva představuje zároveň výstup sítě, tzn. obrázek o rozměrech $32 \times 32 \times 3$. Čísla u jednotlivých vrstev představují počet neuronů dané vrstvy. Čísla u krychlí představují rozměry a počet příznakových map, které jsou vytvořeny jádry dané konvoluční vrstvy.

Na obrázku 4.2 je znázorněna druhá ze struktur sítí. Začátek struktury je totožný jako u první sítě. Tato síť má pouze dvě skryté plně propojené vrstvy. Oranžová krychle znázorňuje reprezentaci výstupu druhé skryté vrstvy ve tvaru trojrozměrné matice $4 \times 4 \times 256$. Následující krychle reprezentují tři konvoluční vrstvy. Poslední žlutá konvoluční vrstva představuje zároveň výstup sítě, tzn. obrázek o rozměrech $32 \times 32 \times 3$. Čísla u jednotlivých vrstev představují počet neuronů dané vrstvy. Čísla u krychlí představují rozměry a počet příznakových map, které jsou vytvořeny jádry dané konvoluční vrstvy.

Třetí struktura sítě je narozdíl od druhé zakončena dvěma konvolučními vrstvami, jinak je totožná. Třetí konvoluční vrstva je pouze odebrána s tím, že druhá konvoluční vrstva představuje výstup sítě, tzn. obrázek o rozměrech $16 \times 16 \times 3$.

4.1 Embedding

V této podkapitole popisuji způsob zadávání vstupu do sítě a dopad tohoto způsobu na jednotlivé vstupní vrstvy. Na vstup sítě se zadávají parametry znaku, který má být zobrazen na výstupním obrázku (parametry jsou popsány v kapitole 3). Každý parametr je zpracován samostatnou vrstvou. Parametr je na vstup zadáván jako vektor o velikosti N, kde N je počet



Obrázek 4.3: Princip embeddingu. Vlevo je vstupní vektor o velikosti 4, představující hodnotu určitého parametru. Červené spoje představují aktivní spoje tzn. váhy těchto spojů jsou násobeny jedničkou, ostatní spoje jsou neaktivní (váhy násobeny nulou). Hodnota parametru je namapována na vektor o velikosti 4, hodnoty tohoto vektoru představují výstupy neuronů. Výstup jednotlivých neuronů je roven váze příslušného aktivního spoje.

hodnot, kterých může daný parametr nabývat. Jedna hodnota vektoru reprezentuje jednu hodnotu parametru. Hodnota parametru, která se má projevit na výstupním obrázku je ve vektoru reprezentována jako 1, ostatní hodnoty pak jako 0.

Jelikož je parametr znaku zadáván tímto způsobem, vstupní vrstva, která zpracovává daný parametr má charakteristickou podobu. Vzniká zde mnohorozměrná reprezentace parametru. Každý parametr je namapován na vektor o počtu neuronů této části, tzv. embedding. Princip embeddingu je znázorněn na obrázku 4.3. Vlevo je vstupní vektor o velikosti 4, představující hodnotu určitého parametru. Červené spoje představují aktivní spoje, tzn. váhy těchto spojů jsou násobeny jedničkou, ostatní spoje jsou neaktivní (váhy násobeny nulou). Hodnota parametru je namapována na vektor o velikosti 4, hodnoty tohoto vektoru představují výstupy neuronů. Výstup jednotlivých neuronů je roven váze příslušného aktivního spoje.

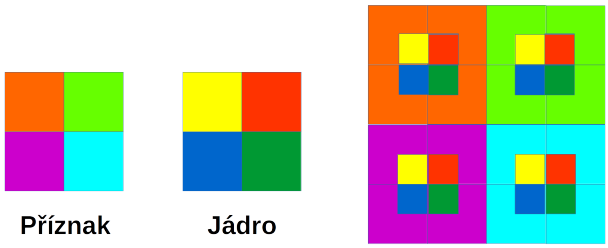
Otázkou je, zda se mezi jednotlivými vektory embeddingu, při trénování sítě, vytvoří vztahy odpovídající vztahům mezi jednotlivými hodnotami parametru. Takové vztahy by naznačovaly schopnost sítě naučit se generalizovat daný parametr. Na základě nepřítomnosti těchto vztahů však nelze soudit opak[1]. Vztahy vektorů embbedingu lze ověřit u parametru translace, rotace a barva, u fontů a znaků se lze domnívat, že podobnější fonty/znaky budou mít podobnější vektory (euklidovská vzdálenost mezi vektory bude menší). Vztahy vektorů embeddingu jsou popsány v kapitole 6.

Aby se zajistilo, že na vstup první skryté vrstvy budou napojeny přímo hodnoty embeddingu, neurony ve vstupních vrstvách nemají žádné aktivační funkce.

4.2 Dekonvoluce

V podkapitole 2.1 je popsán přístup ke zvyšování rozměru příznakových map. V mojí implementaci využívám rozdílného přístupu, a to sice vrstvu implementovanou ve frameworku

[1] Je možné, že se síť naučila generalizovat tento parametr v dalších vrstvách



Obrázek 4.4: Princip dekonvoluce pro jádro o velikosti 2×2 s krokem 2. Vlevo je příznaková mapa o velikosti 2×2, vedle jádro a vpravo výsledná příznaková mapa. Na rozdíl od unpoolingu má jádro ve všech hodnotách určitou váhu. Všechny váhy jádra se vynásobí s první hodnotou příznakové mapy, pak se jádro posune na vedlejší hodnotu a provede se to stejné, přitom výsledek bude ve výsledné příznakové mapě o dva kroky posunut.

Caffe [5] pod názvem „DeconvolutionLayer“ (dekonvoluční vrstva). Název této vrstvy je v mojí práci používán, rovněž operaci kterou tato vrstva provádí označuji jako dekonvoluci. Dekonvoluční vrstva provádí opak konvoluce a tudíž není třeba klasické konvoluce ani unpoolingu.

Princip dekonvoluce pro jádro o velikosti 2×2 s krokem 2 je znázorněn na obrázku 4.4. Vlevo je příznaková mapa o velikosti 2×2, vedle jádro a vpravo výsledná příznaková mapa. Na rozdíl od unpoolingu má jádro ve všech hodnotách určitou váhu. Všechny váhy jádra se vynásobí s první hodnotou příznakové mapy, pak se jádro posune na vedlejší hodnotu a provede se to stejné. Přitom výsledek bude ve výsledné příznakové mapě o dva kroky posunut.

Pokud by jádro mělo velikost 4×4 a krok 2, hodnoty ve výsledné příznakové mapě by se překrývaly. Takovéto hodnoty se sčítají. Konečný rozměr nového příznaku lze dodatečně oříznout (ořezané hodnoty se zahazují a nemají pro další dekonvoluci význam).

Kapitola 5

Trénování sítí

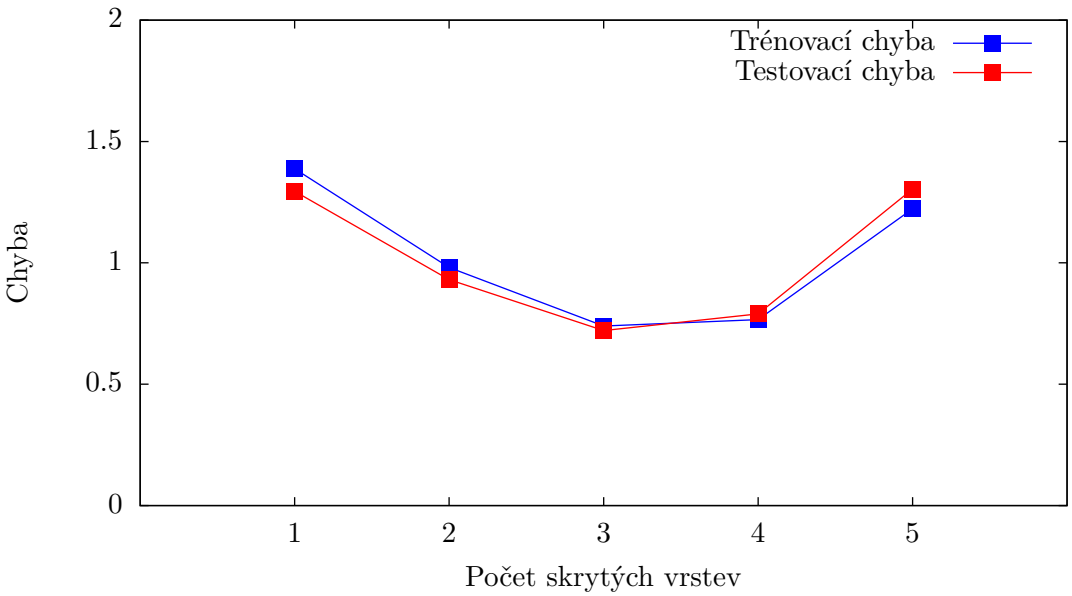
Tato kapitola popisuje experimenty, zabývající se trénováním sítí s různou strukturou tzn. různé počty vrstev, jader atd. Struktury sítí popsané v kapitole 4 dosáhly v experimentech nejlepších výsledků. Na začátku uvádím přístupy, podle kterých jsou sítě o různých strukturách srovnávány. První podkapitola popisuje sítě, které mají pouze plně propojené vrstvy, druhá pak sítě dekonvoluční. Závěrem pak srovnávám obě tyto varianty.

Chybová funkce, která se využívá pro trénování sítí, je čtvercová euklidovská vzdálenost. Hodnota, kterou funkce vrací, určuje, jak kvalitně se síť naučila řešit daný problém a nazývá se chybou. Při trénování se váhy sítě upravují tak, aby tato hodnota byla co nejmenší. V mém případě chyba představuje rozdíl cílového obrázku (obrázek přesně zobrazující znak o zadaných parametrech) a obrázku vygenerovaného sítí. Čím je rozdíl menší, tím jsou obrázky generované sítí kvalitnější. Přesný tvar této funkce používaný ve frameworku Caffe vyjadřuje vztah 5.1. V mém případě t_i značí hodnotu barevného kanálu cílového obrázku a x_i hodnotu barevného kanálu obrázku vygenerovaného sítí. N pak značí počet hodnot barevných kanálů v obrázku ($16 \times 16 \times 3$ nebo $32 \times 32 \times 3$). Při trénování sítě framework Caffe průběžně vypisuje hodnotu chybové funkce, a to jak pro trénovací data (obrázky obsažené v trénovacím datasetu, na kterých se síť trénuje), tak pro testovací data (obrázky, které nejsou při trénování k dispozici). Hodnota chyby tedy průběžně mapuje trénování sítě. Jelikož výstup chybové funkce má tendenci oscilovat, byla trénovací chyba v experimentech této kapitoly vypočítána jako průměr trénovací chyby v posledních 10000 iteracích. Testovací chyba byla vypočítána jako průměr testovacích chyb pro 3200 testovacích obrázků. V průběhu trénování bylo k dispozici 20000 testovacích obrázků.

$$Chyba = \frac{1}{2}\sum_{i=1}^{N}(t_i - x_i)^2 \tag{5.1}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - x_i)^2 \tag{5.2}$$

Při provádění experimentů jsem potřeboval způsob jak změřit kvalitu sítě na různě složitých datech (různá složitost geometrické transformace, okrajové stavy jednotlivých parametrů apod.). Pro tyto účely jsem použil střední kvadratickou odchylku (Mean squared error) 5.2. Hlavním rozdílem, kterého si lze povšimnout v grafech/tabulkách uváděných u experimentů je, že chybová funkce vrací značně menší hodnoty, než mnou použitá střední kvadratická odchylka. To je dáno tím, že střední kvadratická odchylka je počítána na standardních obrázcích (výstup sítě se musí vynásobit 255 a následně je srovnán s původním



Obrázek 5.1: Hodnoty testovací a trénovací chyby v závislosti na počtu skrytých vrstev.

nenormalizovaným obrázkem), zatímco framework Caffe počítá tuto odchylku přímo na výstupu sítě, kde jsou hodnoty normalizovány mezi 0 a 1. Důvodem násobení výstupních obrázků je vizuální interpretace.

Pro různé sítě byly použity různé přístupy k trénování. Inicializace vah podle algoritmu Xavier [3], tak jak je implementován ve frameworku Caffe, však probíhala u všech stejně. Rovněž biasy u všech vrstev, všech sítí, byly nastaveny na 0. Jako aktivační funkce byla u všech vrstev, všech sítí, použita funkce ReLU, kromě vrstev vstupních a výstupních. Nelze zaručit nejlepší možný přístup k trénování daných sítí, stejně jako nelze zaručit nejlepší možný výsledek, při daném přístupu (nedostatečný počet iterací).

5.1 Plně propojené sítě

Následující experimenty jsou prováděny nad první strukturou sítě z kapitoly 4. Dataset pro trénování obsahuje translaci i rotaci (typ T + R) a velikost obrázků je 16×16.

Hloubka sítě. Pro všechny sítě v tomto experimentu byl použit stejný přístup k trénování. Počet vstupů (popisů obrázku) zpracovaných v jedné iteraci byl 32. Počáteční learning rate byl nastaven na 0.002, následně pak každých 150000 iterací dvakrát snižován. Celkem bylo provedeno 1200000 iterací. Momentum bylo nastaveno na 0.9 a weight decay na 0.0005.

Graf na obrázku 5.1 zobrazuje konečné hodnoty testovací a trénovací chyby, po natrénování sítě, v závislosti na počtu skrytých vrstev. Čím více se blíží hodnota testovací chyby k trénovací, tím lépe síť generalizuje. Všech pět sítí generalizuje velmi dobře, výsledky na testovacích datech jsou dokonce lepší než na trénovacích. Síť o třech skrytých vrstvách má nejmenší chybu na generovaných obrázcích.

Kvalita generovaného obrazu. Pro následující experiment je použita síť o třech skrytých vrstvách. Tato síť byla trénována třikrát podle přístupu v předchozím experimentu,

	T + R	T	R	N
Všechny vstupy	101.087	79.099	81.513	67.891
Okrajové vstupy	122.860	90.651	94.853	

Tabulka 5.1: Průměrné střední kvadratické odchylky obrázků vygenerovaných sítí a cílových obrázků v závislosti na složitosti vstupních vektorů. Tyto vektory popisují sedm geometrických složitostí. Translace a rotace (T + R), pouze translace (T), pouze rotace (R), bez translace a rotace (N). První řádek zobrazuje hodnoty pro všechny dostupné hodnoty parametrů jednotlivých geometrických složitostí, druhý řádek pouze pro okrajové hodnoty.

Obrázek 5.2: Dvojice představující cílový obrázek a obrázek vygenerovaný plně propojenou sítí.

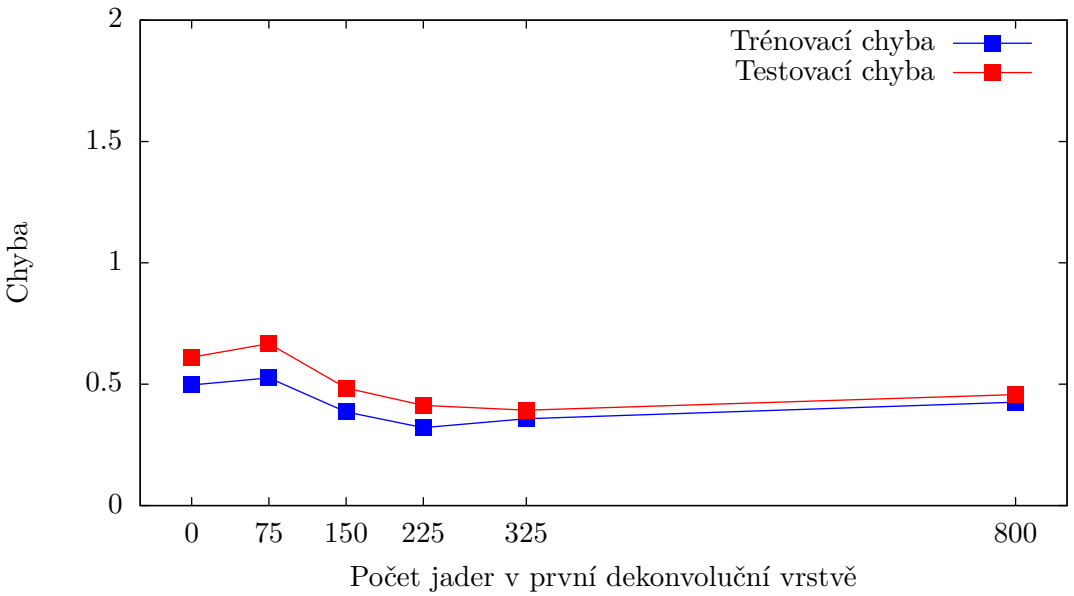
tzn. po každých 1200000 iteracích bylo trénování spuštěno znovu s tím, že síti byly ponechány natrénované váhy (váhy se inicializovaly podle algoritmu Xavier pouze při prvním spuštění). Konečná hodnota testovací a trénovací chyby pro tuto síť je 0.441 a 0.570. Poměr testovací k trénovací chybě (1.293) je tedy horší než v předchozím experimentu, celkově je však chyba menší.

Při standardním testování frameworkem Caffe je síť testována pro různé vstupní vektory, geometrická transformace je většinou zadána[1]. V tomto experimentu jsou síti zadávány vstupní vektory popisující znaky o sedmi různých geometrických složitostech. Translace a rotace, pouze translace, pouze rotace, bez translace a rotace. První tři složitosti mají navíc své okrajové varianty, tzn. je vybráno pouze 16 nejvzdálenějších translací od středu a 64 rotací nejvzdálenějších od nulové rotace. Každá složitost je testována pro 1000 vstupů. Obrázek, který vygeneruje síť na základě vektorů popisujících parametry znaku (násobený 255), je srovnán s cílovým obrázkem podle vztahu 5.2. Toto je provedeno tisíckrát. Střední kvadratická odchylka se sčítá a nakonec je zprůměrována podle počtu obrázků.

Tabulka 5.1 ukazuje výsledky tohoto experimentu. Co se týče prvního řádku je vidět, že síti dělá přibližně stejný problém generovat translace i rotace. Nejtěžší je pak generování translací i rotací zároveň. Druhý řádek poukazuje na problém sítě naučit se okrajové případy. Z tabulky celkově plyne, že pro síť je nejjednodušší naučit se znak bez geometrické transformace a zároveň je pro ni tím těžší naučit se znak, čím je vzdálenější od tohoto stavu (pro rotaci je problém složitější, jak je dále uvedeno v kapitole 6). Tento závěr se může jevit jako poněkud lidský a přirozený, ale s ohledem na trénovací data, která obsahovala stejný počet všech stavů je přinejmenším zajímavý.

Na obrázku 5.2 jsou dvojice představující cílový obrázek a obrázek vygenerovaný sítí. Složitější znaky jako „%“ a „#“ jsou znatelně rozmazané. Výraznější rotace u znaku „N“ způsobuje rovněž rozmazání, byť ne tak značné. Nejlepší výsledky jsou pro jednoduché znaky s nulovou translací a rotací.

[1]To znamená, že se málo kdy stane, že translace a rotace je nulová



Obrázek 5.3: Závislost trénovací a testovací chyby na počtu jader v první dekonvoluční vrstvě. Nulová hodnota představuje úplné vynechání dekonvoluční vrstvy (místo ní je ponechána plně propojená), další hodnoty pak představují počty jader v první dekonvoluční vrstvě.

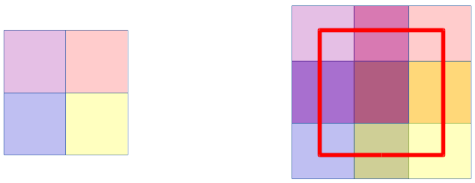
5.2 Dekonvoluční sítě

Podkapitola 5.1 ukazuje, že plně propojené sítě jsou v celku kvalitně schopné generalizovat problém generování obrázků znaků s různými parametry. V této podkapitole se zaměřím zejména na druhou strukturu dekonvoluční sítě popsanou v kapitole 4. Síť má tedy šest vstupních vrstev, dvě plně propojené a dvě dekonvoluční vrstvy. Dataset pro trénování obsahuje translaci i rotaci (typ T + R) a velikost obrázků je 16×16.

Počet jader v první dekonvoluční vrstvě. První experiment se zabývá počtem jader v první dekonvoluční vrsvě, druhá dekonvoluční vrstva je výstupní a tudíž musí mít počet jader roven 3 (počet barevných kanálů obrázků). Velikost jader v obou vrstvách je 4×4, krok 2 a oříznutí výsledných příznakových map o 1 pixel. Na jádra bude zaměřen následující experiment.

Přístup k trénování těchto sítí byl stejný jako v případě plně propojených sítí uvedených v podkapitole 5.1 v části Hloubka sítě. V případě sítí o 325 a 800 jádrech, nebyly z výpočetních důvodů provedeny všechny iterace. Z průběhu trénování je však nepravděpodobné, že by se konečné chyby zásadně změnily.

Graf na obrázku 5.3 znázorňuje závislost trénovací a testovací chyby na počtu jader v první dekonvoluční vrstvě. Ve srovnání s plně propojenými sítěmi 5.1 je vidět horší generalizace. Chyba je však znatelně nižší, což zaručuje lepší kvalitu generovaných obrázků. U většího počtu jader je znatelná mírně lepší generalizace. Jako zvláštní případ je zde uvedena síť, která má pouze jednu dekonvoluční vrstvu. První dekonvoluční vrstva je nahrazena plně propojenou vrstvou a je ponechána pouze druhá (výstupní) dekonvoluční vrstva. Tato síť je označena jako síť s nulovým počtem jader v první dekonvoluční vrstvě. Zajímavou skutečností je, že síť o 800 jádrech je stále schopná velmi dobré konvergence.



Obrázek 5.4: Vlevo je příznaková mapa vytvořená jádrem o velikosti 2×2 s krokem 2, vpravo pak příznaková mapa s překryvy, vytvořená jádrem o velikosti 4 s totožným krokem. Příznaková mapa, ze které se tyto příznaky vytváří, má velikost 2×2. Jednotlivé barvy představují jednotlivé kroky jádra, při vytváření příznakové mapy. Červené ohraničení u druhé příznakové mapy značí ořez o 1 pixel.

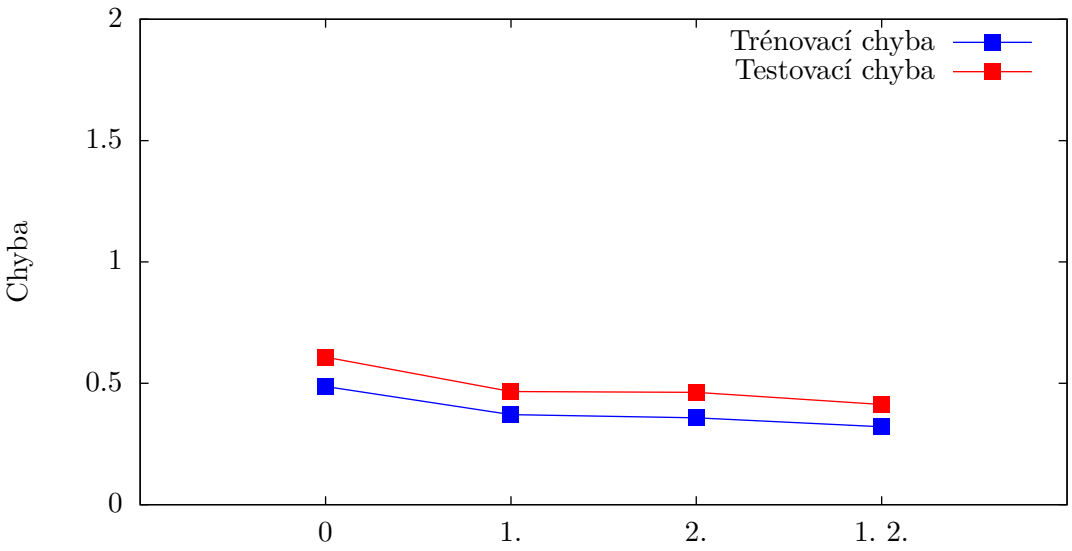
Překryvy hodnot příznakových map. Další experiment zkoumá vliv překryvů hodnot v příznakových mapách. Všechny sítě mají 225 jader v první dekonvoluční vrstvě. Tato hodnota byla zvolena jako nejvýhodnější vzhledem k předcházejícímu experimentu a výpočetnímu výkonu. Přístup k trénování sítí v tomto experimentu byl opět stejný jako v případě plně propojených sítí uvedených v podkapitole 5.1 v části Hloubka sítě.

Na obrázku 5.4 je znázorněn princip překryvů. Jsou zde dvě příznakové mapy, první vytvořená jádrem o velikosti 2×2 s krokem 2 a druhá vytvořená jádrem o velikosti 4×4 s totožným krokem. Příznaková mapa, ze které se tyto příznaky vytváří, má velikost 2×2. Jednotlivé barvy představují krok jádra při vytváření příznakové mapy. U příznakové mapy vpravo vznikají překryvy a tato mapa také není dvakrát větší, což je potřeba pro konstrukci mých sítí. Proto je třeba ji o jeden pixel ořezat. Ořez znázorňuje červená linie. Pro větší zdrojové příznakové mapy se výsledné mapy vytvoří obdobně. Jádro 4×4 s krokem 2 tvoří novou příznakovou mapu, kde hodnoty téměř všech pixelů tvoří suma 4 hodnot (dochází k čtyřnásobnému překryvu). Po ořezání této příznakové mapy o 1 pixel tvoří jedinou výjimku okrajové pixely, kdy všechny kromě rohových jsou tvořeny sumou 2 hodnot. Jádro o velikosti 2×2 s krokem 2 vytvoří dvakrát větší příznakovou mapu bez překryvů.

Graf na obrázku 5.5 ukazuje vztah trénovací a testovací chyby k překryvům hodnot v příznakových mapách. Pořadová čísla značí v jaké vrstvě se vyskytují příznakové mapy s překryvy, nulová hodnota pak představuje síť bez překryvů v příznakových mapách. Všechny sítě generalizují téměř totožně. Největší chybu má síť, která nemá překryvy v žádné vrstvě, sítě s překryvem v jedné vrstvě mají chybu přibližně stejnou a nejmenší chybu má síť s překryvy v obou vrstvách.

Kvalita generovaného obrazu. Následující experiment je totožný s experimentem v podkapitole 5.1 v části Kvalita generovaného obrazu. Tento experiment byl proveden se sítí o 225 jádrech a překryvech v obou dekonvolučních vrstvách. Tato síť byla trénována totožným způsobem jako síť v podkapitole 5.1 v části Kvalita generování obrazu. Trénovací a testovací chyba této sítě je 0.235 a 0.312. Výsledky jsou zobrazeny v tabulce 5.2. Dekonvoluční síť má, stejně jako plně propojená, téměř totožné problémy s translací a rotací, rovněž okrajové stavy jsou problémové.

Na obrázku 5.6 jsou zobrazeny dvojice představující cílový obrázek a obrázek vygenerovaný dekonvoluční sítí. Pro dvojice zobrazující jednoduché znaky bez geometrické transformace je rozdíl téměř nepostřehnutelný. Při složitějších znacích a složitější geometrické transformaci je rozdíl znatelnější.



Obrázek 5.5: Vztah překryvů hodnot příznakových map k trénovací a testovací chybě. Pořadová čísla značí v jaké vrstvě se vyskytují příznakové mapy s překryvy, nulová hodnota pak představuje síť bez překryvů v příznakových mapách.

	T + R	T	R	N
Všechny vstupy	55.893	43.432	43.693	37.373
Okrajové vstupy	69.726	49.262	50.928	

Tabulka 5.2: Průměrné střední kvadratické odchylky obrázků vygenerovaných sítí a cílových obrázků v závislosi na složitosti vstupních vektorů. První řádek zobrazuje hodnoty pro všechny dostupné hodnoty parametrů jednotlivých geometrických složitostí, druhý řádek pouze pro okrajové hodnoty.

Obrázek 5.6: Dvojice představující cílový obrázek a obrázek vygenerovaný dekonvoluční sítí.

	T + R	T	R	N
Plně propojená				
Všechny vstupy	101.087	0.782	0.806	0.671
Okrajové vstupy	1.215	0.896	0.938	
Dekonvoluční				
Všechny vstupy	55.893	0.777	0.782	0.669
Okrajové vstupy	1.247	0.881	0.911	

Tabulka 5.3: Srovnání plně propojené a dekonvoluční sítě pro různě složité vstupní vektory. Místo skutečných průměrných kvadratických odchylek je u jednotlivých složitostí vstupních vektorů zobrazen podíl průměrné střední kvadratické odchylky pro danou složitost a průměrné střední kvadratické odchylky dosažené na vstupních vektorech, popisujících úplnou geometrickou složitost (tato odchylka se přitom sama sebou nedělí a zůstává zachována).

Obrázek 5.7: Trojice představující cílový obrázek, obrázek vygenerovaný plně propojenou sítí a obrázek vygenerovaný dekonvoluční sítí.

Srovnání plně propojené a dekonvoluční sítě. Na závěr této kapitoly uvádím srovnání plně propojené a dekonvoluční sítě, které byly použity pro experimenty s kvalitou generování obrazu. Srovnání je uvedeno v tabulce 5.3, jde v podstatě o srovnání tabulek 5.1 a 5.2. Srovnání je provedeno tak, že místo skutečných průměrných středních kvadratických odchylek je u jednotlivých složitostí vstupních vektorů zobrazen podíl průměrné střední kvadratické odchylky pro danou složitost a průměrné střední kvadratické odchylky dosažené na vstupních vektorech, popisujících úplnou geometrickou složitost (tato odchylka se přitom sama sebou nedělí a zůstává zachována).

Je patrné, že se sítě chovají stejně co se týče problému s různou složitostí vstupních vektorů. Dekonvoluce tedy nijak nepomůže síti v tomto ohledu. Pro dekonvoluční síť je stále stejně obtížnější/jednoduší generovat obrázky s různě složitými znaky, ovšem celkově je kvalita generovaných obrázků lepší.

Na obrázku 5.7 jsou zobrazeny trojice. Cílový obrázek, obrázek vygenerovaný plně propojenou sítí a obrázek vygenerovaný dekonvoluční sítí. První tři trojice představují znaky o nulové geometrikcé transformaci. Zatímco obrázky zobrazující jednoduché znaky jsou téměř totožné s cílovým obrázkem, u znaku „%“ je rozdíl znatelný. Poslední dvě trojice představují okrajovou translaci a rotaci a okrajovou translaci. Obě sítě generují méně kvalitní znaky oproti cílovým obrázkům, obrázky sítě dekonvoluční jsou však kvalitnější.

Tato kapitola ukazuje schopnost neuronových sítí generalizivat problém generování znaků o různých parametrech. Dekonvoluční sítě dosahují značně kvalitnějších výsledků, a to zejména při generování složitějších znaků.

Následující dvě kapitoly popisují experimenty využívající první strukturu dekonvoluční sítě uvedené v kapitole 4. Tato síť má tři dekonvoluční vrstvy a generuje obrázky o rozlišení 32×32. Parametry této sítě jsem zvolil na základě výsledků experimentů této podkapitoly. Počet jader jsem zvětšil na 300 a rozdělil mezi první a druhou dekonvoluční vrstvu. Překryvy hodnot příznakových map se nachází ve všech vrstvách, jádro má tedy velikost 4×4 a krok 2.

Myšlenka je dosáhnout většího rozlišení přidáním další dekonvoluční vrstvy. Počet iterací při trénování byl opět 1200000, learning rate byl po celou dobu konstantní o hodnotě 0.0001. Momentum bylo opět 0.9 a weight decay byl zvýšen na 0.002. Pro dosažení nižší chyby bylo toto trénování opět spuštěno třikrát. Výsledná kvalita generovaného obrazu je přibližně totožná jako v případě druhé dekonvoluční struktury. Tato síť byla natrénována na všech typech datasetu popsaných v kapitole 3.

Kapitola 6

Embedding

Tato kapitola se zabývá embeddingem sítí, jejichž struktura odpovídá první dekonvoluční struktuře popsané v kapitole 4. Tyto sítě mají totožnou strukturu, ale byly natrénovány na různých typech datasetů popsaných v kapitole 3. Trénování těchto sítí bylo popsáno v kapitole 5. Následující podkapitoly rozebírají jednotlivé embeddingy těchto sítí. Pokud je síť natrénována na datasetu, kterému schází nějaká geometrická transformace, pak se pro tuto transformaci embedding nevytvoří[1].

6.1 Translace

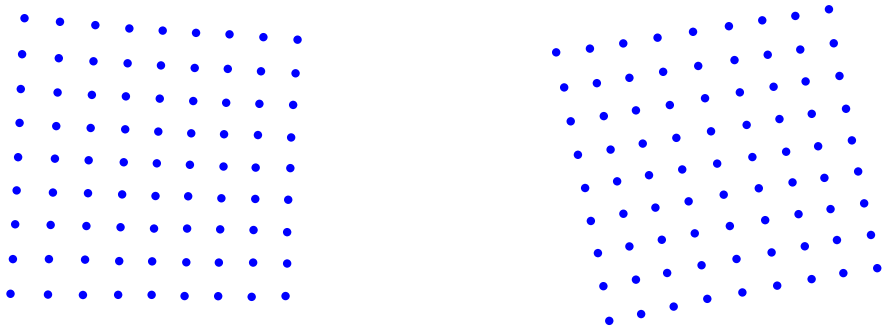
První embedding, na který se zaměřím, představuje reprezentaci translací. Tento embedding reprezentuje každou translaci jako vektor o 64 dimenzích. Redukcí dimenzionality pomocí analýzy hlavních komponent (PCA) lze tento vektor zobrazit jako dvoudimenzionální. Výsledek tohoto zobrazení je na obrázku 6.1. Nalevo je embedding pro síť natrénovanou na datasetu o plné geometrické složitosti, vpravo je pak embedding pro síť natrénovanou na datasetu bez rotací. Výsledkem je téměř přesná mřížka, což naznačuje velmi podobné vztahy jako mezi sebou mají translace. Redukce dimenzionality do trojrozměrného prostoru na obrázku 6.2 už není tak ideální. Embedding pro plnou geometrickou transformaci vlevo má znatelně lepší rozložení jednotlivých translací oproti „zlomenému“ embeddingu pro dataset bez rotací. Dále se zabývám pouze prvním embeddingem, embedding pro jednoduší dataset je zmíněn závěrem této podkapitoly.

Obrázek 6.3 znázorňuje reálné vzdálenosti vektorů v embeddingu translace. Jednotlivé čtverečky představují translace, barva pak značí průměrnou euklidovskou vzdálenost sousedních vektorů, tzn. vektorů představující sousední translace. Výsledek tohoto zobrazení je velmi podobný tomu, co lze vyčíst z redukce dimenzionality do třírozměrného prostoru (obrázek 6.2 vlevo).

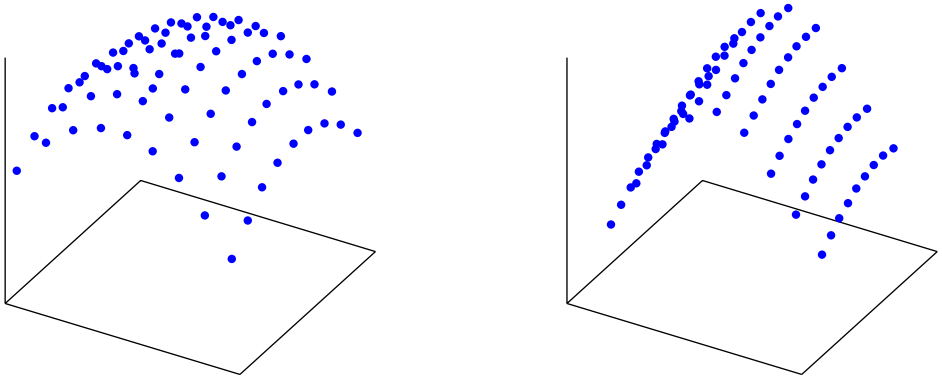
Na základě obrázku 6.1, 6.2 a 6.3 lze předpokládat možnost interpolace mezi jednotlivými vektory. Pokud jsou dva vektory představující translace těsně sousedící s nulovou translací (zleva a zprava nebo shora a zdola) totožně naváhovány (váha 0.5), výsledkem by měl být vektor představující nulovou translaci.

Interpolace mezi dvěma vektory. Následující experiment zkoumá interpolace mezi dvěma vektory na stejné vertikální úrovni, které mezi sebou mají jednu nebo tři translace.

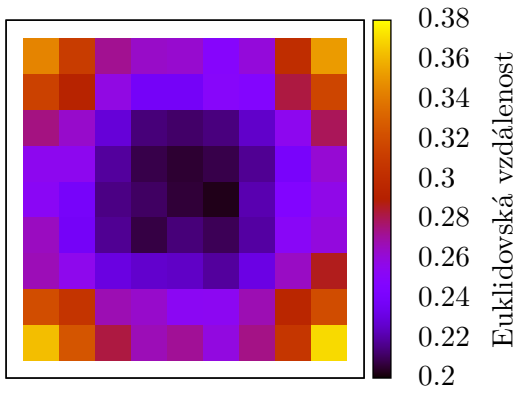
[1] Na vstup příslušné vstupní vrstvy jsou v případě chybějícího parametru zadávány samé 0



Obrázek 6.1: Redukce dimenzionality (PCA) vektorů embeddingu translace do dvojrozměrného prostoru. Vlevo embedding sítě natrénované na datasetu o úplné geometrické složitosti, vpravo pak pro síť natrénovanou na datasetu bez rotace.



Obrázek 6.2: Redukce dimenzionality (PCA) vektorů embeddingu translace do třírozměrného prostoru. Vlevo embedding sítě natrénované na datasetu o úplné geometrické složitosti, vpravo pak pro síť natrénovanou na datasetu bez rotace.



Obrázek 6.3: Euklidovské vzdálenosti vektorů v embeddingu translace pro síť natrénovanou na datasetu o úplné geometrické složitosti. Jednotlivé čtverečky představují translace, barva pak značí průměrnou euklidovskou vzdálenost sousedních vektorů tzn. vektorů představující sousední translace.

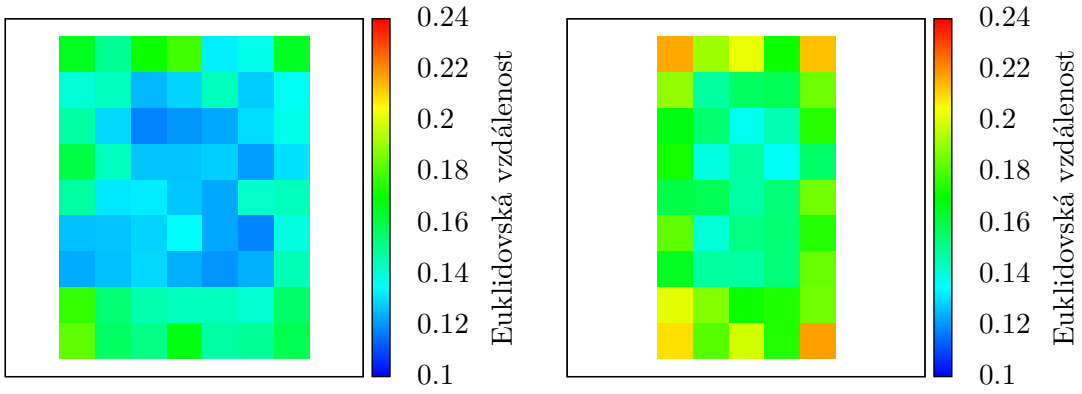
Experiment byl proveden tak, že se vektory vyjadřující okolní translaci naváhovaly váhou 0.5 a poté se sečetly, výsledek byl následně srovnán s vektorem vyjadřujícím translaci mezi těmito dvěma. Výsledné euklidovské vzdálenosti mezi interpolovanými vektory a vektorem, který vyjadřuje předpokládanou translaci, zobrazují teplotní mapy na obrázku 6.4. Teplotní mapa vlevo vyjadřuje jednu vnitřní translaci, teplotní mapa vpravo pak tři. Jednotlivé čtverečky představují translace, barva pak velikost euklidovské vzdálenosti. Bílý prostor znázorňuje translace, pro které nebylo možné tento údaj spočítat. Byly provedeny i experimenty s horizontálně stejnou úrovni, stejně jako s diagonálním rozložením. Výsledky těchto experimentů byly v podstatě totožné.

Interpolace mezi vektory, které představují vzdálenější translace je horší. Výsledek interpolace se průměrně liší o polovinu vzdálenosti mezi vektory představující sousední translace. S rostoucí vzdáleností translací interpolovaných vektorů však tato chyba nijak značně nenarůstá. Chyba narůstá se vzdáleností od nulové translace.

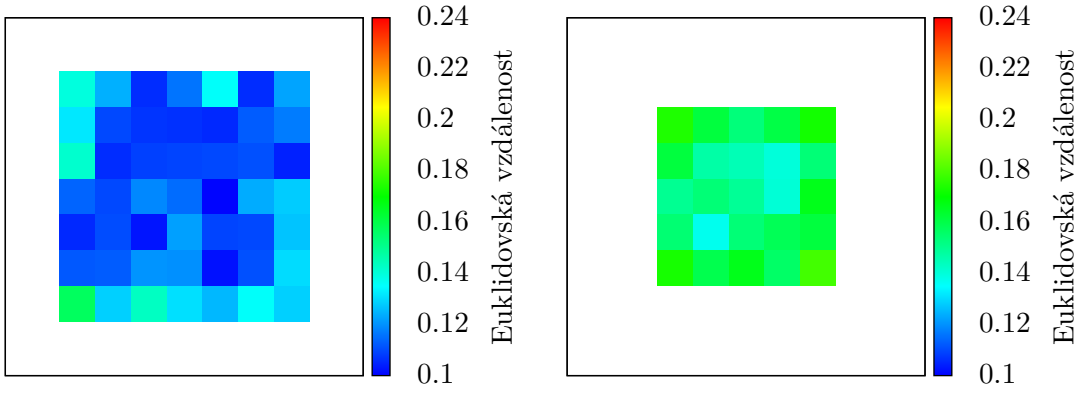
Interpolace mezi osmi vektory. Další experiment popisuje interpolace mezi osmi vektory, které představují translace tvořící čtverec. V prvním případě tyto translace „obkličují“ jednu translaci, v druhém jsou pak o translaci vzdáleny (tvoří tak čtverec o rozměrech 5×5). Osum vektorů se naváhuje hodnotou 0.125 a součet je srovnán s vektorem, který představuje translaci uprostřed čtverce. Výsledky zobrazují teplotní mapy na obrázku 6.5. Vlevo čtverec o velikosti 3×3, vpravo pak 5×5. Interpolace mezi větším množstvím vektorů se zdá být o něco lepší. Interpolace mezi vektory představující vzdálenější translace je opět horší, rovněž chyba narůstá se vzdáleností od nulové translace.

Parametr translace je tedy embeddingem reprezentován nejhůře v okrajových translacích a nejlépe v translacích okolo translace nulové. Na základě tohoto poznatku a tabulky 5.3 (srovnání standardních a okrajových vstupů) předpokládám značný vliv embeddingu na výstup sítě (tato tabulka sice zobrazuje hodnoty pro jiné struktury sítě, ale nepředpokládám velké rozdíly embeddingů těchto sítí).

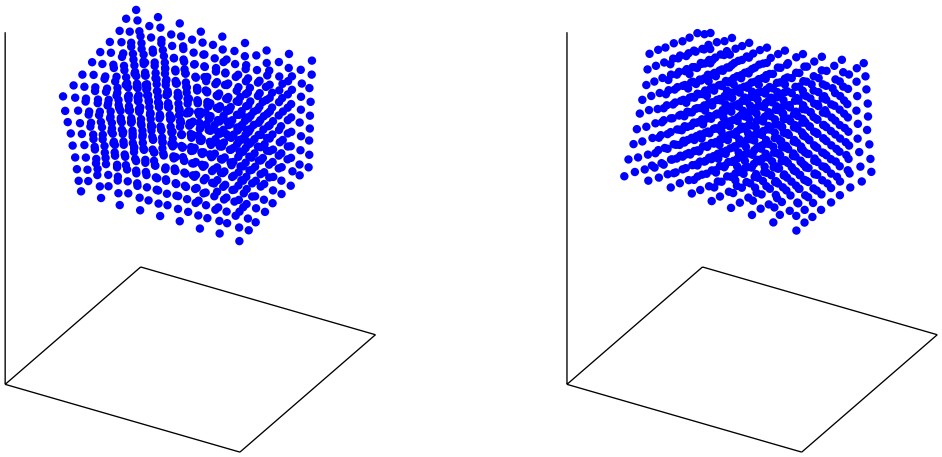
Experimenty s interpolací pro embedding sítě natrénované na datasetu bez rotace se lišily v závislosti na směru interpolace. V horizontálním směru byly interpolace o něco lepší než u výše rozebíraného embeddingu. Ve vertikálním směru byly interpolace znatelně



Obrázek 6.4: Teplotní mapa vlevo vyjadřuje euklidovskou vzdálenost mezi interpolací dvou vektorů představujícími translace, vertikálně na stejné úrovní, mající mezi sebou jednu translaci a vektorem představujícím tuto translaci. Teplotní mapa vpravo vyjadřuje stejné hodnoty, ale dva vektory představující interpolované translace dělí 3 translace, euklidovská vzdálenost je počítána vzhledem k prostřední.



Obrázek 6.5: Teplotní mapa vlevo vyjadřuje euklidovskou vzdálenost interpolace mezi osmi vektory představujícími translace tvořící čtverec 3×3 a vektoru, představujícího translaci ležící ve středu tohoto čtverce. Teplotní mapa vpravo představuje to stejné, ale translace tvoří čtverec 5×5.



Obrázek 6.6: Redukce dimenzionality (PCA) vektorů embeddingu rotace do trojrozměrného prostoru. Síť trénovaná na datasetu o úplné geometrické složitosti vlevo, síť trénovaná na datasetu bez translace vpravo.

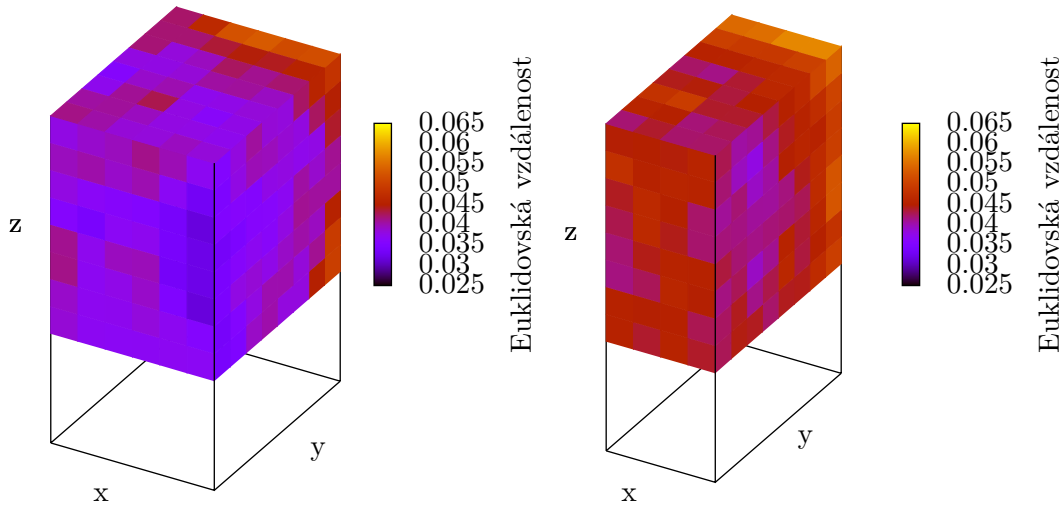
horší, zejména pro vzdálenější translace. Tento výsledek odpovídá obrázku 6.2, kde je vidět „zalomení“ prostoru embeddingu v jednom ze směrů.

Embedding sítě natrénované na datasetu o úplné geometrické složitosti tedy lépe reprezentuje translaci. Rotace pravděpodobně „donutily“ síť naučit se reprezentaci translace odděleně. Zda se síť natrénovaná na datasetu bez rotace naučila kvalitně reprezentovat translaci v dalších částech sítě je rozebráno v kapitole 7.

6.2 Rotace

Embedding rotace reprezentuje každou rotaci jako vektor o 256 dimenzích. Obrázek 6.6 představuje redukci dimenzionality (opět pomocí PCA) embeddingu rotace do třírozměrného prostoru. Síť trénovaná na datasetu o úplné geometrické složitosti vlevo, síť trénovaná na datasetu bez translace vpravo. Stejně jako embedding translace byl znatelně deformován při chybějící rotaci v trénovacích datech, tak i embedding transformace je znatelně deformován při chybějící translaci. Zda jsou tyto parametry na sobě nějak závislé je popsáno v kapitole 7. Embedding první sítě je značně symetrický, což opět naznačuje možnost interpolace. Následující experimenty se zabývají tímto embeddingem.

Několik experimentů s interpolací vektorů představující rozdílné rotace ukázalo, že interpolace dosahuje horších výsledků při extrémnější rotaci kamery kolem osy x o kladný úhel. Obrázek 6.7 zobrazuje teplotní mapy, znázorňující euklidovskou vzdálenost interpolace mezi dvěma vektory představujícími rotace, které mezi sebou mají jednu (teplotní mapa vlevo) nebo tři (teplotní mapa vpravo) rotace a vektoru, který představuje rotaci ležící ve středu mezi nimi. Váhování vektorů při interpolaci je opět 0.5. Každá trojice vektorů má stejnou rotaci kamery kolem osy x. Osa x u teplotních map představuje rotaci kamery kolem osy y, osa y rotaci kamery kolem osy x a osa z rotaci kamery ve vlastní ose. Čelní



Obrázek 6.7: Teplotní mapy, znázorňují euklidovskou vzdálenost interpolace mezi dvěma vektory, které mezi sebou mají jednu (vlevo) nebo tři rotace (vpravo) a vektoru, který představuje rotaci ležící ve středu, mezi nimi. Každá trojice vektorů má stejnou rotaci kamery kolem osy x. Osa x u teplotních map představuje rotaci kamery kolem osy y, osa y rotaci kamery kolem osy x a osa z rotaci kamery ve vlastní ose.

stěna teplotní mapy ukazuje, že rotace kamery ve vlastní ose při rotaci kamery o -20°kolem osy x téměř neovlivňují interpolaci. Boční stěna ukazuje, že rotace kamery ve vlastní ose ovlivňuje interpolaci při rotaci kamery o 15°a 20°kolem osy x. Při zkoumání různých průřezů teplotních map v rovině xy (stejná rotace kamery kolem vlastní osy) je stále patrné zhoršení, při rotaci kamery o 10°, 15°a 20° kolem osy x.

Tabulka 5.3 ukazuje horší kvalitu generovaných obrázků sítí v okrajových případech. Experimenty s embeddingem transformace poukazují na to, že by kvalita generování měla být problematická pouze v rámci kladné rotace kolem osy x. Tabulka 6.1 ukazuje, že je tomu skutečně tak. První údaj značí rozdíl mezi kladnou a zápornou rotací kamery kolem osy x, druhý údaj pouze rozdíl mezi okrajovou rotací kamery kolem osy x a poslední údaj (Rohy) rozdíl mezi okrajovou rotací kamery kolem osy x, y a kolem vlastní osy. Embedding rotace má tedy stejně jako embbeding translace přímý vliv na výstup sítě.

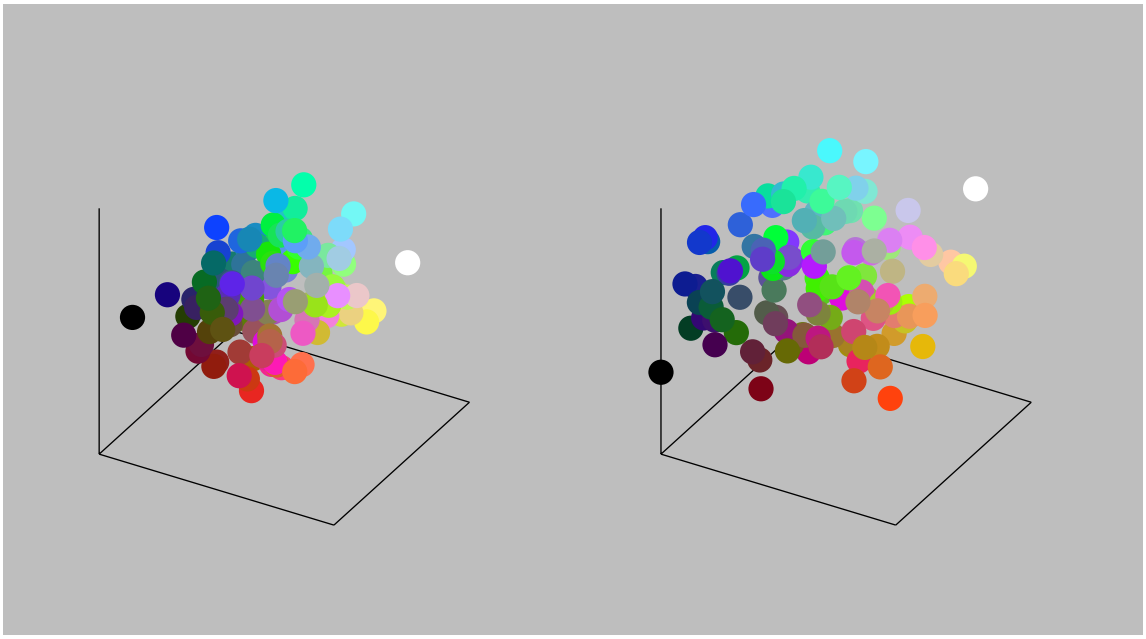
Pro embedding sítě natrénované na datasetu bez translace lze na základě obrázku 6.6 (vpravo) předpokládat horší interpolaci v jistých směrech rotace kamery. Translace pravděpodobně „donutily“ síť reprezentovat rotaci odděleně (podobný vliv, jako měl embedding rotace na embedding translace).

6.3 Ostatní parametry

Embedding barvy reprezentuje každou barvu jako vektor o 32 dimenzích. Na obrázku 6.8 je vlevo embedding barvy pozadí pro síť natrénovanou na datasetu o úplné geometrické složitosti (embedding barvy popředí vypadá podobně), jehož dimenze jsou redukovány na tři pomocí analýzy hlavních komponent (PCA). Každá barevná tečka reprezentuje vektor,

	0 to 20°	20°	Rohy
+x	57.180	64.858	67.403
-x	50.0476	50.350	53.183

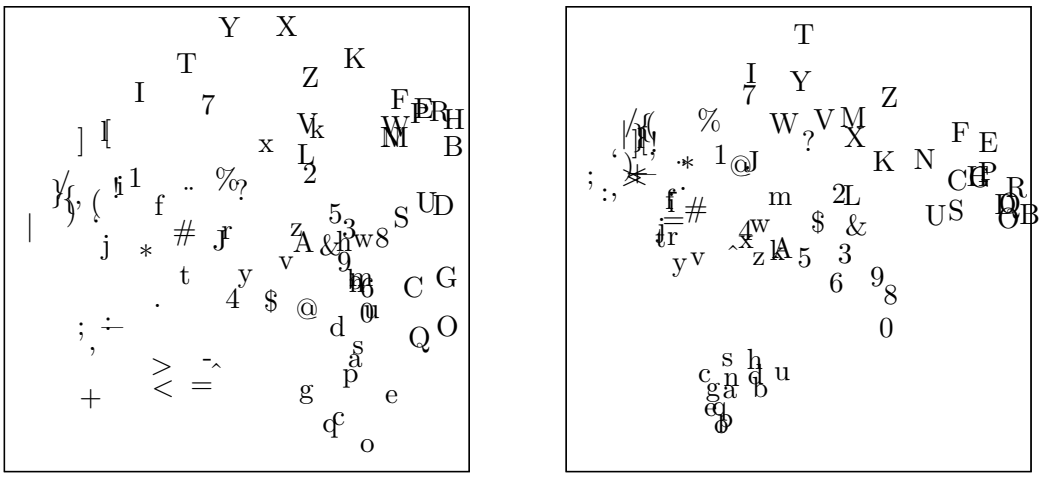
Tabulka 6.1: První údaj značí rozdíl mezi kladnou a zápornou rotací kamery kolem osy x, druhý údaj pouze rozdíl mezi okrajovou rotací kamery kolem osy x a poslední údaj (Rohy) rozdíl mezi okrajovou rotací kamery kolem osy x, y a kolem vlastní osy.



Obrázek 6.8: Vlevo embedding barvy pozadí pro síť natrénovanou na datsetu o úplné geometrické složitosti, jehož dimenze jsou redukovány pomocí analýzy hlavních komponent (PCA). Každá barevná tečka reprezentuje vektor, který reprezentuje danou barvu. Na pravé straně je pak zobrazeno skutečné rozložení barev obsažených v datasetu.

který reprezentuje danou barvu. Na pravé straně je pak zobrazeno skutečné rozložení barev obsažených v datasetu. Vztahy mezi jednotlivými vektory embeddingu jsou velmi podobné reálným vztahům barev. Jelikož byly barvy generovány náhodně, nebyly provedeny experimenty s interpolací (experimenty s interpolací barev na základě výstupu sítě jsou uvedeny v kapitole 7).

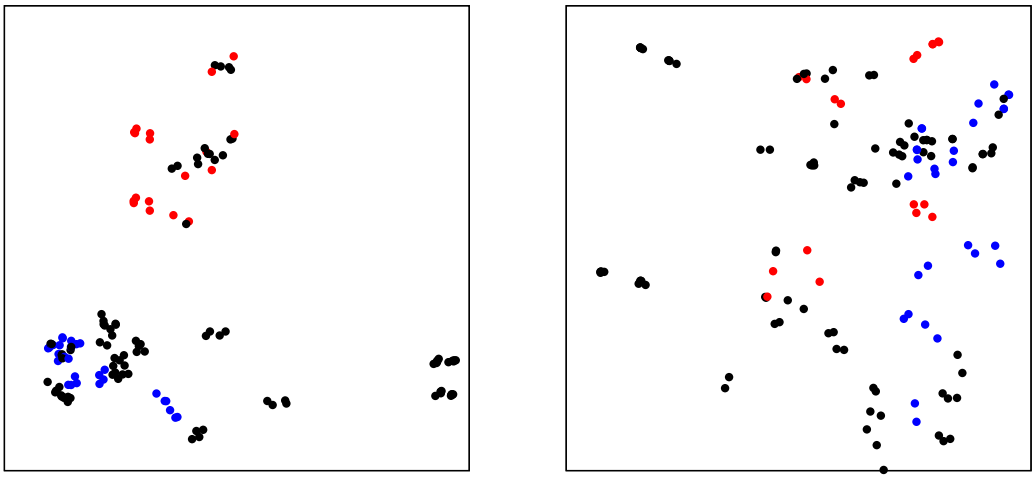
Embedding znaků reprezentuje každý znak jako vektor o 64 dimenzích. Na obrázku 6.9 jsou zobrazeny embeddingy znaků redukované do dvou dimenzí, pomocí analýzy hlavních komponent (PCA). Vlevo je zobrazen embedding znaků pro síť natrénovanou na datasetu o úplné geometrické složitosti a vpravo embedding znaků pro síť natrénovanou na datasetu bez geometrické transformace. Jednotlivé znaky představují pozici vektorů, které je reprezentují. Vznikly čtyři výraznější „ostrůvky“, a to velká písmena, malá písmena, čísla a písmena/znaky připomínající svislou čáru. Písmena velké abecedy, které se podobají znaku „O“, se nacházejí blízko sebe, dále pak znaky „M“ a „W“. V embeddingu znaků se vytvořily smysluplné vztahy, v dalších dimenzích budou s velkou pravděpodobností existovat vztahy další. Oba embeddingy obsahují podobné vztahy (ostrůvky), proto nelze usuzovat, že geometrická transformace znaků má na embedding znaků větší vliv.



Obrázek 6.9: Embeddingy znaků redukované do dvou dimenzí pomocí analýzy hlavních komponent (PCA). Jednotlivé znaky představují pozici vektorů, které je reprezentují. Vlevo je zobrazen embedding znaků pro síť natrénovanou na datasetu o úplné geometrické složitosti a vpravo embedding znaků pro síť natrénovanou na datasetu bez geometrické transformace.

Embedding fontů reprezentuje každý font jako vektor o 64 dimenzích. Na obrázku 6.10 jsou embeddingy fontů opět redukované pomocí analýzy hlavních komponent (PCA) do dvou dimenzí. Vlevo je zobrazen embedding fontů pro síť natrénovanou na datasetu o úplné geometrické složitosti a vpravo embedding fontů pro síť natrénovanou na datasetu bez geometrické transformace. Červené tečky představují patkové fonty („Serif“), modré bezpatkové („Sans-serif“), černé tečky pak ostatní fonty. V dalších dimenzích by s velkou pravděpodobností byly k nalezení další vztahy, např. tučné písmo nebo kurzíva, které ve dvojrozměrné projekci nebyly znatelné. Oba embeddingy opět obsahují podobné vztahy, tudíž geometrická transformace pravděpodobně nemá na tento embedding větší vliv.

První dekonvoluční síť z kapitoly 4, která byla natrénována na datasetu o úplné geometrické složitosti, si vytvořila embeddingy velmi kvalitně reprezentující svůj parametr. U embeddingu translace a rotace se ukazuje, že kvalita interpolace v jednotlivých částech embeddingu těchto parametrů určuje výslednou kvalitu generování obrazu v těchto částech. Sítě, kterým při trénování scházel jeden z těchto parametrů, mají deformovaný embedding, což vede k horší interpolaci v určitých částech embeddingu. Embedding barvy vytváří kvalitní reprezentaci RGB rozložení, embeddingy znaků a fontů si vytvářejí smysluplné vztahy. Embeddingy mají s velkou pravděpodobností přímý vliv na výstup sítě, tzn. ostatní části sítě se pravděpdobně pouze snaží reprezentovat jednotlivé embeddingy v podobě obrázku. Tato kapitola obsahuje experimenty s interpolací pouze v rámci naučených stavů, to znamená, že se interpoluje s cílem „trefit se“ do nějakého naučeného stavu. Následující kapitola obsahuje experimenty s čistě náhodnou interpolací.



Obrázek 6.10: Embeddingy fontů redukované do dvou dimenzí pomocí analýzy hlavních komponent (PCA). Vlevo je zobrazen embedding fontů pro síť natrénovanou na datasetu o úplné geometrické složitosti a vpravo embedding fontů pro síť natrénovanou na datasetu bez geometrické transformace. Červené tečky představují patkové fonty („Serif“), modré bezpatkové („Sans-serif“), černé tečky pak ostatní fonty.

Kapitola 7

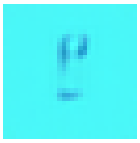
Interpolace

V předchozí kapitole se zabývám experimenty s embeddingem sítě a na základě těchto experimentů odhaduji schopnost sítě generalizovat, tzn. schopnost správně reprezentovat jednotlivé parametry. Problém těchto experimentů spočívá v tom, že se zaměřují pouze na interpolaci v rámci hodnot obsažených v trénovacím datasetu, tzn. ostatní hodnoty se interpolují tak, aby se dosáhlo některé z jiných hodnot. Samotný embedding neumožňuje měřit kvalitu interpolace jejímž výsledkem je úplně nová hodnota. Tato kapitola se zabývá náhodnými interpolacemi[1]. Kvalitu těchto interpolací měřím na základě samotného výstupu sítě. Pro interpolaci mezi translací do pozice (5, 9) a (-4, 6), při váhování 0.7 a 0.3, je předpokládaným výstupem sítě znak s translací do pozice (2.3, 8.1). Příslušné váhování se zadá na vstup sítě a výstup sítě je poté srovnán se znakem s patřičnou translací (pomocí vztahu 5.2). V této kapitole využívám opět sktrukturu první dekonvoluční sítě z kapitoly 4, natrénovanou na datasetech o různé geometrické složitosti. Na úvod této kapitoly uvádím experimenty s nulovými vstupy, tzn. vektory neobsahují 1 indikující hodnotu daného parametru (nulová translace a rotace má také svůj vektor indikující tento stav).

Následující výstupy platí pro síť natrénovanou na datasetu o úplné geometrické složitosti. Pokud je síti zadán na všechny vstupní vrstvy pouze nulový vstup, generované obrázky jsou jednobarevné. Barva je šedá s průměrnou hodnotou barevných kanálů 125.539, což je téměř skutečný průměr všech RGB barev (127.5). Pokud je zadána jedna nebo obě barvy (ostatní vstupy jsou nulové), generované obrázky zobrazují neurčitý symbol, který je zobrazen na obrázku 7.1. Pokud není zadán žádný font ale je zadán znak, na výstupu se objeví standardně vypadající znak. Všechny znaky na výstupu mají nulovou geometrickou transformaci. Pokud jsou navíc zadány různé translace, znak stále nejeví známky rotace, stejně tak pokud jsou zadány různé rotace (bez translace), znak nejeví známky translace. Pokud je síti zadán nulový vstup na některou vstupní vrstvu, vygeneruje průměr všech hodnot představujících parametr této vstupní vrstvy. Sítě jejichž datasety při trénování neobsahovaly translaci nebo rotaci mají problém při nulovém vstupu na části představující barvy (výsledná barva není šedá).

Následující experimenty různým způsobem váhují vstupy sítě. Maximální počet hodnot jednoho parametru, mezi kterými se interpoluje, je 10. Celkem jsou dva typy váhování. Stejné, tzn. váhy jsou rovnoměrně rozděleny, a různé, váhy jsou náhodně vybrány z hodnot 0.1 to 0.9 tak aby suma byla 1. Počet testovacích obrázků pro každou interpolaci je 1000.

[1] Je tedy možná interpolace i v rámci hodnot obsažených v datasetu, ale výskyt těchto variant je vzhledem k náhodnosti malý.



Obrázek 7.1: Výstup sítě, která byla natrénována na datasetu o úplné geometrické složitosti, pro nulové vstupy pro parametr typ znaku, font, translace a rotace.

	1	2	4	7	10
Stejné váhování					
T + R	36.204	0.821	0.735	0.609	0.635
N	16.137	93.000	126.095	159.549	180.689
Různé váhování					
T + R		0.880	0.770	0.641	
N		46.863	98.429	144.468	

Tabulka 7.1: Poměr průměrných středních kvadratických odchylek při různém počtu interpolovaných barev k průměrné střední kvadratické odchylce při jedné barvě tzn. žádná interpolace.

7.1 Barvy

Jako první se zaměřím na schopnost sítě interpolovat v rámci barvy pozadí. Tabulka 7.1 ukazuje poměry průměrných středních kvadratických odchylek při různém počtu interpolovaných barev k průměrné střední kvadratické odchylce při jedné barvě, tzn. žádná interpolace (tato hodnota je vždy uvedena v prvním slupci). Na prvním řádku je síť, která byla trénována na datasetu s úplnou geometrickou složitostí[2], na druhém řádku pak síť, která byla natrénována na datasetu bez geometrické transformace.

Síť trénovaná na datasetu o úplné geometrické složitosti dosahuje při interpolaci lepších výsledků než pro barvy v trénovacím datasetu. S rostoucím počtem interpolovaných barev se výsledky sítě zlepšují. Druhá síť dosahuje velmi špatných výsledků (snaha o interpolaci je však stále patrná) a s rostoucím počtem interpolovaných barev se výsledky zhoršují. Rozdíly mezi stejným a různým váhováním jsou většinou zanedbatelné.

Translace a rotace pravděpodobně donutily síť lépe reprezentovat RGB prostor. Pokud se znak pohybuje po pozadí překrývá tím barvu pozadí na více místech a síť se musí pro stejná místa naučit dvě odlišné barvy. Pokud se znak nijak nepohybuje (případ druhé sítě), síť se může barvy okolí naučit v podstatě nezávisle. Rostoucí počet barev při interpolaci znamená, že se vstup dané vstupní vrstvy sítě stále více blíží vstupu nulovému a barvy se stále více podobají šedé. Druhá síť se tedy nenaučila dobře reprezentovat průměrnou barvu, jak je uvedeno na začátku této kapitoly.

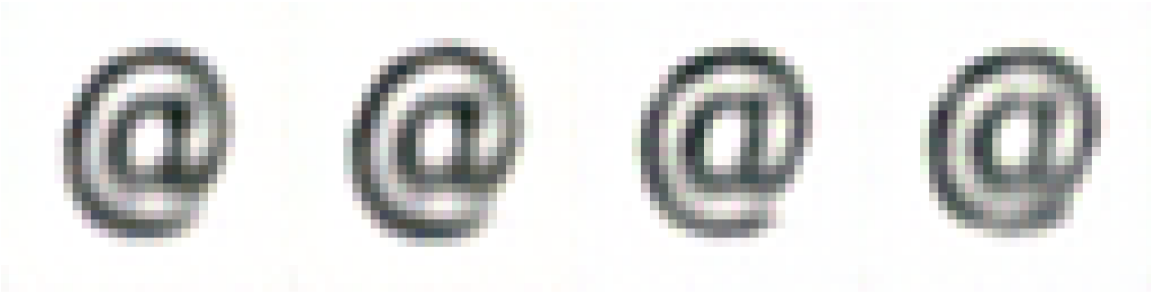
7.2 Translace a rotace

Tabulka 7.2 ukazuje výsledky interpolací translací pro síť natrénovanou na datasetu o úplné geometrické složitosti a pro síť natrénovanou na datasetu bez rotace. Lepší interpolace do-

[2] Při tomto experimentu dostávala tato síť na vstup pouze vstupy popisující znaky s nulovou geometrickou transformací.

	1	2	4	7	10
Stejné váhování					
T + R	44.750	1.082	1.217	1.137	1.150
T	36.477	1.631	1.845	2.113	2.147
Různé váhování					
T + R		1.359	1.267	1.345	
T		1.676	1.968	2.147	

Tabulka 7.2: Poměr průměrných středních kvadratických odchylek při různém počtu interpolovaných translací k průměrné střední kvadratické odchylce při jedné translaci.



Obrázek 7.2: Levá dvojice zavináčů byla vygenerována sítí natrénovanou na datasetu o plné geometrické složitosti. První zavináč při zadané nulové translaci a druhý při nezadané (nulový vstup). Druhá dvojice zavináčů byla vygenerována sítí, natrénovanou na datasetu bez rotace, pro stejné vstupy (se zadanou nulovou translací a bez zadané nulové translace).

sáhla opět síť první[3]. Druhá síť má znatelně horší výsledky. S rostoucím počtem interpolovaných translací se kvalita generovaného výstupů zhoršuje, to značí špatnou reprezentaci nulových vstupů. Na obrázku 7.2 je vlevo zavináč vygenerovaný první sítí při zadané nulové translaci a vedle něj při nezadané (nulový vstup), není zde viditelný žádný rozdíl. Druhá dvojice zavináčů byla vygenerována druhou sítí, při stejných vstupech (se zadanou nulovou translací a bez). Na této dvojici jde vidět znatelný rozdíl a to v neprospěch druhého zavináče, který byl vygenerován pomocí nulového vstupu. Tento zavináč je dokonce horší než zavináč vygenerovaný první sítí. První zavináč je nejkvalitnější ze všech[4]. Vizuální srovnání výstupů sítě naznačuje, že síť nemá problém se správným odhadem nové translace, ale spíše s kvalitní reprezentací znaku v nové translaci (znaky jsou rozmazané).

Tabulka 7.3 ukazuje výsledky interpolací rotací pro síť natrénovanou na datasetu o úplné geometrické složitosti a pro síť natrénovanou na datasetu bez translace. První síť interpoluje opět lépe[5]. Druhá síť ovšem interpoluje velmi dobře a dokonce se její výsledky zlepšují s počtem interpolovaných rotací. To naznačuje, že se kvalitně naučila reprezentovat nulový vstup jako nulovou rotaci. Při nulovém vstupu tato síť skutečně kvalitně generuje znaky bez rotace. Vizuální srovnání výstupů sítě opět naznačuje problémy s kvalitní reprezentací znaku v nové rotaci (znaky jsou rozmazané), výsledná rotace je však správná.

Experimenty s translací a rotací ukazují, že tyto parametry na sobě do jisté míry závisí.

[3]Při tomto experimentu dostávala tato síť na vstup pouze vstupy popisující znaky s nulovou rotací.

[4]Síť, které při trénování chyběly rotace, je díky tomu schopná, lepé se naučit zadávané translace, nikoli však interpolace mezi nimi. Tato skutečnost je patrná s tabulky 7.2.

[5]Při tomto experimentu dostávala tato síť na vstup pouze vstupy popisující znaky s nulovou translací.

	1	2	4	7	10
Stejné váhování					
T + R	43.737	0.933	1.030	0.942	0.961
R	33.934	1.165	1.122	1.087	1.032
Různé váhování					
T + R		0.962	0.937	0.923	
R		1.021	1.074	1.106	

Tabulka 7.3: Poměr průměrných středních kvadratických odchylek při různém počtu interpolovaných rotací k průměrné střední kvadratické odchylce při jedné rotaci.

	1	2	4	7	10
Stejné váhování	49.888	0.935	0.761	0.702	0.676
Náhodné váhování		1.073	0.891	0.786	

Tabulka 7.4: Poměr průměrných středních kvadratických odchylek při různém počtu interpolovaných translací, transformací a barev k průměrné střední kvadratické odchylce při jedné translaci, transformaci a barvě pro síť natrénovanou na datasetu o úplné geometrické složitosti.

Tato závislost se nepromítá pouze do embeddingů jednotlivých parametrů, jak ukazuje kapitola 6, ale také do konečného výstupu sítě. Tato skutečnost zvyšuje pravděpodobnost přímé závislosti výstupu sítě na embeddingu.

Při srovnání sítě, která byla natrénována na datasetu bez rotace a té, která byla natrénována na datasetu bez translace je patrný následující rozdíl. První ze sítí je ve srovnání s výstupem pro jednu zadanou translaci v interpolaci translací dvakrát horší. Druhá ze sítí se naučila generovat interpolaci rotací téměř stejně kvalitně jako vstupy pro jednu zadanou rotaci. Problémy interpolace u obou sítí jsou zejména v nekvalitní reprezentaci znaku s výslednou vlastností, vlastnost sama (umístění znaku do pozadí) se reprezentuje velmi dobře. Otázka zní, co druhou síť naučilo lepší interpolaci v rámci jejího parametru? Rozdíl mezi interpolací translace a rotace (v neprospěch translace) je znatelný i v rámci sítě, která byla natrénována na datasetu o úplné geometrické složitosti.

Důvod může být v překryvech znaků. Zatímco u rotace se znaky z velké části téměř vždy překrývají, u translace existují stavy, kde se můžou znaky překrývat s mnohem menší části. Významem překryvů se zabývám na konci této kapitoly.

Na závěr této podkapitoly uvádím experiment se složitějšími interpolacemi. Tabulka 7.4 ukazuje schopnost sítě, natrénované na datasetu o úplné geometrické složitosti, interpolovat mezi barvou, translací a rotací zároveň. Jednotlivé hodnoty v záhlaví tabulky značí počet barev, translací a rotací mezi kterými se interpoluje (2 značí interpolaci mezi dvěma barvami, translacemi a rotacemi).

Stejně jako v případě 7.1 a 7.3 i pro složitější interpolaci dosahuje síť natrénovaná na datech s veškerými parametry lepších výsledků při interpolování, než pro typy vstupů, na kterých byla trénována. Tato skutečnost znamená schopnost sítě generalizovat do značné míry, tzn. síť zobecnila daný problém do té míry, že jsou KVALITA její výstupy v podstatě nezávislé na vstupu (pokud jsou tyto vstupy v rámci trénovacích dat).

Obrázek 7.3: Interpolace mezi znaky „W“ a „X“ pro síť natrénovanou na datasetu o plné geometrické složitosti.

7.3 Znaky a fonty

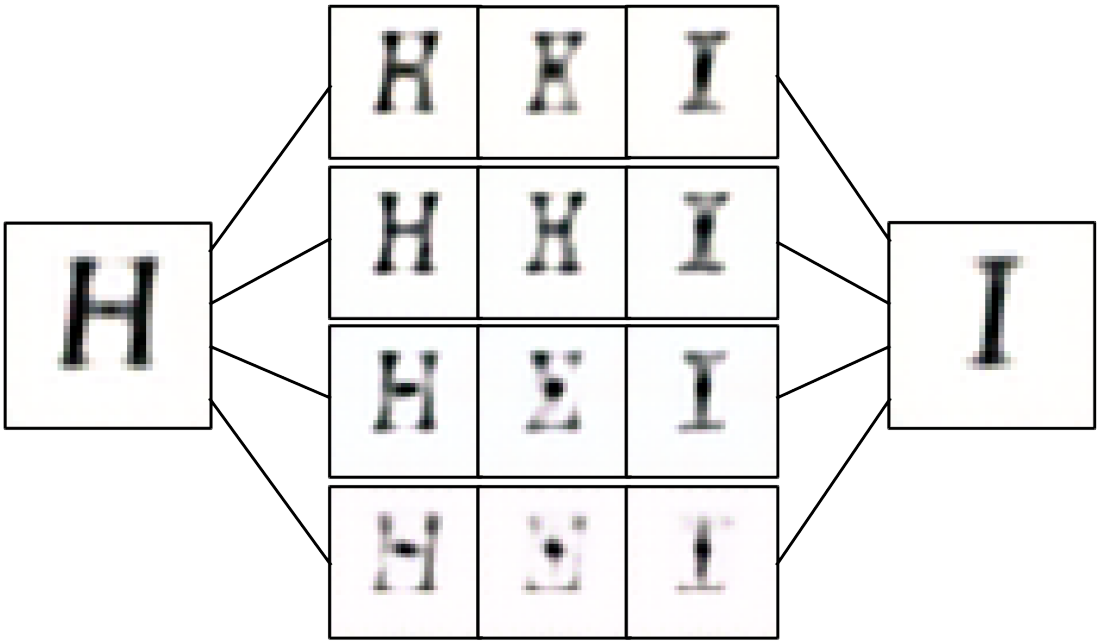
V této podkapitole se zaměřuji na interpolaci znaků a fontů. Kvalitu interpolace zde odhaduji na základě vizuálního posouzení výstupu sítě. Na obrázku 7.3 je znázorněna interpolace mezi znaky „W“ a „X“ pro síť natrénovanou na datasetu o úplné geometrické složitosti. Interpolace vypadá přirozeně, ovšem v částech, kde znaky nemají nic společného, je poněkud nedokonalá. Je zde patrné morfování jednotlivých částí znaku (okrajové nožičky u znaku „W“).

Kvalita interpolace znaků téměř nezávisí na přítomnosti parametru translace a rotace v trénovacím datasetu. Existují zde ale případy, kdy je rozdíl při ztrátě jednoho nebo obou parametrů do značné míry znatelný. Na obrázku 7.4 je znázorněn jeden z takových případů. Obrázek znázorňuje interpolace znaku „H“ a „I“ pro sítě natrénované na datasetech o různé geometrické složitosti. Horní interpolace je pro translaci i rotaci, dále pak popořadě pouze translace, pouze rotace a bez translace a rotace. První dvě sítě interpolují téměř totožně. Problém nastává při ztrátě parametru translace, kdy dochází ke zhoršení, při ztrátě obou parametrů je zhoršení ještě znatelnější.

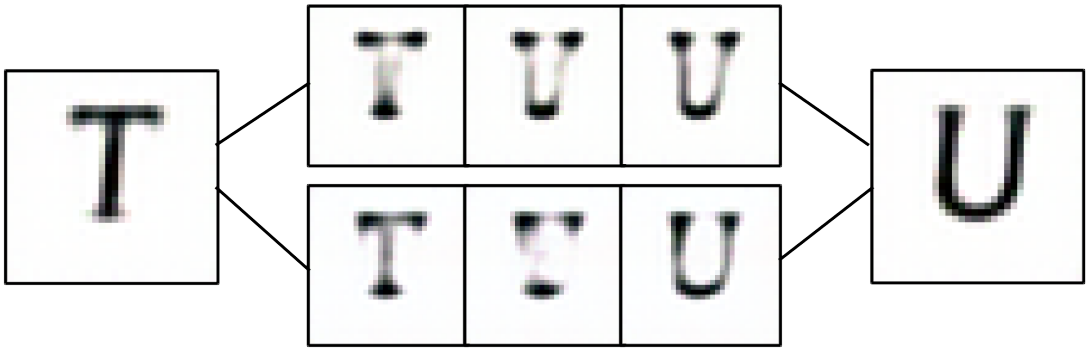
V předcházející podkapitole zmiňuji pravděpodobný význam překryvů pro interpolaci. Rozdíl mezi sítí, která byla natrénována na datasetu bez parametru translace, a tou, která tento parametr v datasetu měla je ten, že první ze sítí nemá k dispozici vhodný překryv znaků „H“ a „I“. Pokud má síť ve svých trénovacích datech rotaci, vzniknou zde překryvy, ale ne tak kvalitní, jako pro různé translace. V případě chybějící translace a rotace je výsledek velmi špatný, neboť znaky „H“ a „I“ nemají v základním stavu téměř nic společného (až na samotný střed, jak lze vidět na prostředním stavu interpolace u poslední sítě na obrázku 7.4). Na obrázku 7.5 je znázorněna interpolace znaku „T“ a „U“. Horní interpolace je pro síť natrénovanou na datasetu obsahující pouze translace, spodní pak pro síť natrénovanou na datasetu obsahujícím pouze rotace. Síť bez translace má opět velmi nekvalitní interpolaci, zatímco síť s translací má interpolaci velmi přirozenou. Při rotaci znaků „T“ a „U“ nedochází v podstatě k žádným překryvům (svislé části znaků se nikdy nepřekryjí), zatímco při různých translacích se znaky můžou překrývat z velké části.

Obrázek 7.6 zobrazuje výsledky interpolací více znaků pro síť natrénovanou na datasetu o úplné geometrické složitosti. Interpolace více znaků nepůsobí síti žádné větší potíže, výsledkem je přirozené morfování všech znaků.

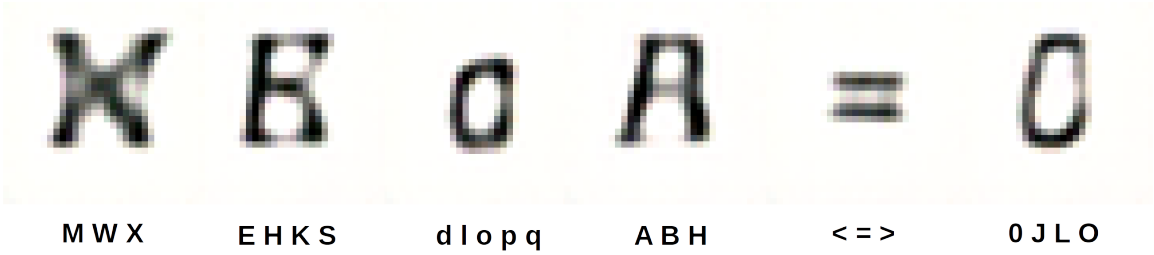
Všechny předchozí experimenty byly provedeny s nulovým vstupem u parametru font. Všechny trénované sítě se pro tento vstup naučily standardně vypadající font, jak je uvedeno na začátku této kapitoly. Obrázek 7.7 znázorňuje interpolaci mezi dvěma fonty, pořadí sítí je stejné jako na obrázku 7.4. Velikost fontu je vždycky stejná, ale i při stejné velikosti fontu se můžou velikosti jednotlivých znaků o něco lišit. Znaky o různých fontech se vždycky



Obrázek 7.4: Interpolace znaku „H“ a „I“ pro sítě natrénované na datasetech o různé geometrické složitosti. Horní interpolace je pro translaci a rotaci, dále pak popořadě pouze translace, pouze rotace a bez geometrické transformace.



Obrázek 7.5: Interpolace znaku „T“ a „U“. Horní interpolace je pro síť natrénovanou na datasetu bez parametru rotace, spodní pak pro síť natrénovanou na datasetu bez parametru translace.



Obrázek 7.6: Interpolace více znaků pro síť natrénovanou na datasetu o úplné geometrické složitosti.

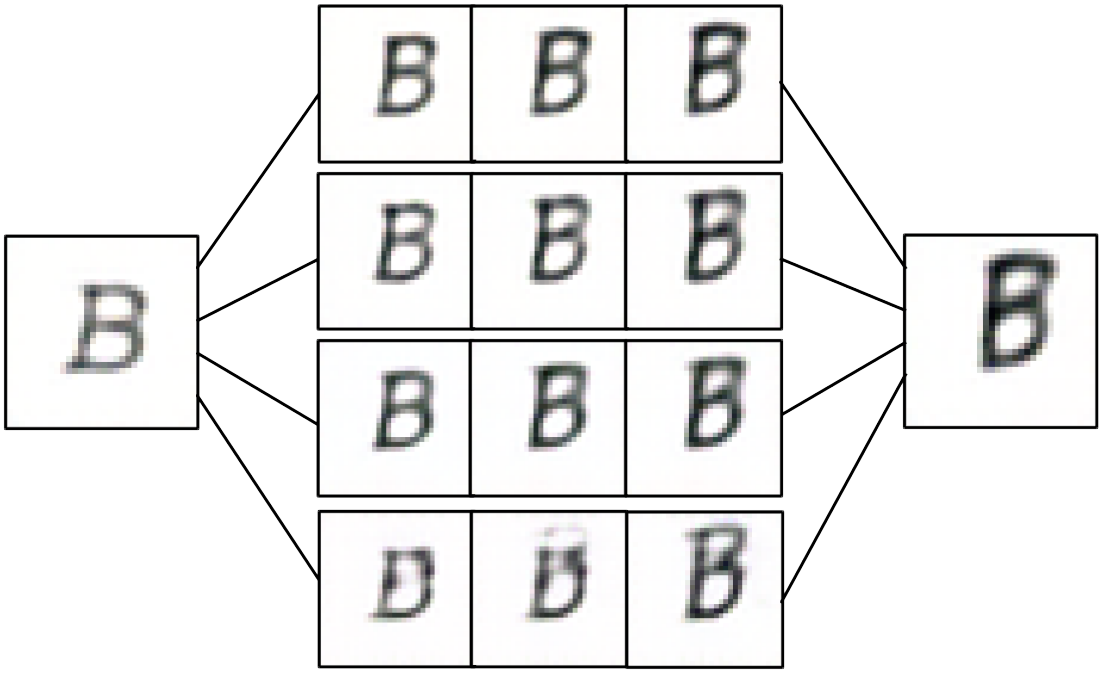
z velké části překrývají, tudíž téměř není zapotřebí geometrické transformace. Přechody ze silnějších fontů na tenké nejsou pro žádnou ze sítí problém. Problém nastane u fontů, které prezentují stejný znak výrazně odlišným způsobem, jako je vidět na obrázku 7.7. Samotná translace nebo rotace dostačuje pro vytvoření kvalitní interpolace. Pokud ovšem v trénovacím datesetu chybí oba tyto parametry je výsledná interpolace značně nekvalitní. Interpolace v posledním případě je však nekvalitní pouze v části, kde dva znaky nemají nic společného (nepřekrývají se), spodní část se interpoluje kvalitně.

Na základě obrázků 7.4, 7.5 a 7.7 lze opět předpokládat důležitost překryvů pro interpolaci. Všechny čtyři sítě, natrénované na datasetech o různých geometrických složitostech, generují kvalitní interpolace mezi znaky a fonty. Problémy s interpolací nastanou pouze tehdy, pokud se části znaků nepřekrývají. Tento problém má zejména síť, která je natrénována na datasetu bez geometrické transformace. Translace a rotace vytvářejí nové podoby znaků, vznikají tak nové překryvy, a to i u znaků, které původně neměly téměř nic společného. Překryvy by také vysvětlovaly tendenci sítě generovat horší výsledky se vzdáleností od nulové translace v případě translace a horší výsledky při kladné rotaci kamery kolem osy x v případě rotace[6].

Množství hodnot, kterých mohou nabývat jednotlivé parametry, pravděpodobně nehraje tak významnou roli jako překryvy. Důvodem je situace na obrázku 7.4 a 7.5, kde síť, natrénovaná na datasetu obsahujícím pouze rotace generuje méně kvalitní interpolace než síť natrénovaná na datasetu obsahujícím pouze translace. Parametr rotace přitom může nabývat devítinásobného počtu hodnot oproti parametru translace.

Pokud se dva stavy nepřekrývají nebo se překrývají málo, síť si nevytvoří potřebné vazby pro kvalitní interpolaci. Pokud by dataset například obsahoval pouze dvě translace, které by se navíc nepřekrývaly, lze předpokládat, že by se síť naučila znaky o těchto translacích reprezentovat jako dva odlišné. Síť v takovém případě nemá žádné „vodítko“, které by naznačovalo jakýkoli vztah mezi těmito translacemi. Přidáním dalších translací a rotací naroste množství podob daného znaku. Síť není schopna reprezentovat všechny podoby znaku odděleně a je nucena se naučit vztahy, které jí umožní kvalitně reprezentovat všechny tyto podoby. Tyto vztahy se podle kapitoly 6 do značné míry podobají reálným vztahům mezi parametry.

[6]Při této rotaci, znaky, které jsou vysoké (zejména velká písmena) zabírají velkou část plochy, zatímco normální znaky ne, tím pádem vysoké znaky v této rotaci nejsou z velké části překryty znaky malými.



Obrázek 7.7: Interpolace fontů pro sítě natrénované na datasetech o různé geometrické složitosti. Horní interpolace je pro translaci a rotaci, dále pak popořadě pouze translace, pouze rotace a bez geometrické transformace.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout a analyzovat konvoluční neuronovou síť, která umožňuje generování obrazu. V první části této práce bylo experimentováno s různými strukturami sítí. Na základě těchto experimentů byly vytvořeny tři sítě, které umožňují generování obrázků znaků s různými parametry na základě těchto parametrů. Parametry jsou typ znaku, font, barva znaku, barva pozadí, translace a rotace. Všechny sítě generovaly méně kvalitní obrázky pro znaky s geometrickou transformací než pro znaky bez geometrické transformace. Konvoluční sítě dosahovaly menší chyby při generování než sítě plně propojené, tudíž byla pro analýzu vybrána síť konvoluční.

Druhá část práce se věnovala analýze vstupních vrstev této sítě. Každý parametr měl vlastní vstupní vrstvu, kde se v průběhu trénování vytvořila jeho mnohorozměrná reprezentace. Toho bylo docíleno zadáváním parametrů na vstup sítě, jako nulového vektoru s jednou 1, kde 1 představuje zadávanou hodnotu parametru a nuly ostatní hodnoty parametru. Bylo zjištěno, že vztahy uvnitř těchto reprezentací jsou do značné míry podobné reálným vztahům v rámci parametrů. Byly provedeny experimenty s různou geometrickou složitostí datasetu. Tyto experimenty prokázaly závislost parametrů translace a rotace. Pokud byly v datasetu přítomny oba tyto parametry, síť si vytvořila kvalitnější reprezentaci obou těchto parametrů, než pro datasety, které obsahovaly pouze translaci nebo rotaci.

Závěrečná část práce se zabývá schopností sítě interpolovat mezi hodnotami parametrů. Výsledky interpolace jsou velmi kvalitní v rámci všech parametrů, a to i pro větší počet interpolovaných hodnot. Síť je schopna naučit se na základě omezeného počtu translací a rotací přítomných v datasetu generovat znaky všech translací a rotací v rozsahu okrajových hodnot. Totéž platí pro barvy. Při interpolaci znaků a fontů vznikne přirozené morfování. Toho lze využít například pro generování efektních přechodů mezi nápisy nebo pro generování nových fontů.

Na základě provedených experimentů s interpolacemi znaků předpokládám velký vliv překryvů znaků na jejich vzájemnou interpolaci. Překryvy by pravděpodobně mohly mít velký dopad i na celkovou kvalitu generování obrazu, tzn. trénování sítí generujících obraz obecně. Toto tvrzení by však bylo potřeba ověřit dalšími experimenty. Dále lze na základě práce Dosovitského a kol. [2] předpokládat schopnost sítí extrapolovat, tzn. generovat translace a rotace mimo rozsah obsažený v datasetu. Zajímavé by také bylo analyzovat konvoluční vrstvy sítí.

Literatura

[1] Bradski, G.: The opencv library. *Doctor Dobbs Journal*, ročník 25, č. 11, 2000: s. 120–126.

[2] Dosovitskiy, A.; Springenberg, J. T.; Brox, T.: Learning to Generate Chairs with Convolutional Neural Networks. *CoRR*, ročník abs/1411.5928, 2014.

[3] Glorot, X.; Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[4] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; aj.: Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, editace Z. Ghahramani; M. Welling; C. Cortes; N. D. Lawrence; K. Q. Weinberger, Curran Associates, Inc., 2014, ISBN 9781510800410, s. 2672–2680.

[5] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[6] LeCun, Y.; Bengio, Y.; Hinton, G.: Deep learning. *Nature*, ročník 521, č. 7553, 2015: s. 436–444.

[7] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi:10.1007/s11263-015-0816-y.

[8] Salakhutdinov, R.; Hinton, G.: Deep Boltzmann machines. In *Artificial Intelligence and Statistics*, 2009.

[9] Shinners, P.: PyGame. Dostupné z: http://pygame.org/ [Online; navštíveno 12.5.2015].

Přílohy

Seznam příloh

Příloha A

Obsah CD

Latex/	Zdrojový tvar bakalářské práce
Nets/	Soubory představující sítě ve frameworku Caffe
Output/	Ukázky výstupů sítí
Videos/	Videa prezentující výsledky bakálářské práce
Fonts/	Fonty vygenerováné sítí
BP.pdf	Bakálářská práce ve formátu PDF
charGenerator.py	Skript umožňující generování datasetů