VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

GENERATIVNÍ NEURONOVÁ SÍŤ PRO TVORBU SYNTETICKÝCH FOTOREALISTICKÝCH OBRAZŮ

GENERATIVE NEURAL NETWORK FOR CREATING SYNTHETIC PHOTOREALISTIC IMAGES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Adam Hora

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Kamil Říha, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Adam Hora ***ID:*** 211479

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Generativní neuronová síť pro tvorbu syntetických fotorealistických obrazů

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte teoretický princip funkce generativních neuronových sítí pro vytváření fotorealistického obsahu digitálních obrazů na základě specifické trénovací databáze. S využitím vhodné softwarové platformy naprogramujte generativní neuronovou síť, která bude vytvářet syntetické snímky tak, aby jejich obsah stylově i tématicky reflektoval podobnost s vybranou trénovací obrazovou databází fotografií či jiných fotorealistických děl. Součástí práce je i nalezení vhodné databáze tak, aby její využití nebylo v rozporu s licenčními podmínkami. V rámci diplomové práce je očekávána implementace pokročilého algoritmu pro generování syntetických obrazů o dostatečné kvalitě.

DOPORUČENÁ LITERATURA:

[1] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. MIT press, 2016.

[2] GULLI, Antonio; PAL, Sujit. Deep learning with Keras. Packt Publishing Ltd, 2017.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 21.5.2024

Vedoucí práce: doc. Ing. Kamil Říha, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Hlavním cílem této práce je vybrat a navrhnout model neuronové sítě, který bude schopen generovat realistické obrázky tématicky zapadající do vybrané datové sady. Pro řešení je použita architektura hluboké konvoluční generativní adverzní sítě. Tato síť je implementována v programovacím jazyce Python pomocí aplikačních programovacích rozhraní Tensorflow a v něm obsaženém rozhraní Keras. Model je natrénován na vybrané datové sadě a jsou zobrazeny výsledné vygenerované snímky. Finální model a jednotlivé snímky jsou nakonec vyhodnoceny pomocí různých metod hodnocení kvality.

KLÍČOVÁ SLOVA

neuronové sítě, umělá inteligence, konvoluční neuronové sítě, generativní adverzní sítě, WGAN, generování obrazu, Python, Tensorflow, Celeba-HQ

ABSTRACT

The main objective of this work is to select and design a neural network model that will be able to generate realistic images thematically fitting the selected dataset. The architecture used for the solution is Deep convolutional generative adversarial network. This network is than implemented in the Python programming language using the Tensorflow application programming interface and its included interface Keras. Finally, the model is trained on the selected dataset and the resulting generated images are presented. The final model and individual images are then evaluated using various quality assessment methods.

KEYWORDS

neural networks, artificial intelligence, convolutional neural networks, generative adversarial networks, WGAN, image generation, Python, Tensorflow, Celeba-HQ

HORA, Adam. *Generativní neuronová síť pro tvorbu syntetických fotorealistických obrazů*. Diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023. Vedoucí práce: doc. Ing. Kamil Říha, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Adam Hora
VUT ID autora:	211479
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Generativní neuronová síť pro tvorbu syntetických fotorealistických obrazů

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

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

Obsah

Úvod **19**

1 Neuronové sítě **21**
1.1 Architektura . . . 22
1.2 Historie . . . 23

2 Hluboké neuronové sítě **25**
2.1 Hluboké učení . . . 25
2.2 Konvoluční neuronové sítě (CNN) . . . 26
2.2.1 Konvoluční vrstva . . . 26
2.2.2 Sdružovací vrstva . . . 28
2.2.3 Aktivační vrstva . . . 29
2.2.4 Normalizace dávek . . . 32
2.2.5 Plně propojená vrstva . . . 32
2.2.6 Dropout vrstva . . . 33
2.3 Učení neuronových sítí . . . 34
2.3.1 Druhy učení . . . 34
2.3.2 Ztrátová funkce . . . 35
2.3.3 Optimalizace učení . . . 36

3 Generativní neuronové sítě **41**
3.1 Generativní adverzní neuronové sítě . . . 41
3.1.1 Použití technik učení bez učitele v GAN . . . 41
3.1.2 Trénování . . . 42
3.1.3 Varianty GAN . . . 45

4 Vlastní řešení **47**
4.1 Implementace . . . 47
4.1.1 Tensorflow . . . 47
4.1.2 Datová sada . . . 48
4.1.3 Předzpracování dat . . . 48
4.1.4 Trénovací hardware . . . 49
4.1.5 Parametry trénování . . . 49
4.1.6 Návrh a tvorba diskriminátoru a generátoru . . . 50
4.2 Trénování modelu . . . 54
4.2.1 Průběh trénování . . . 54
4.3 Výsledky trénování . . . 57
4.3.1 Generované obrazy . . . 57

4.4 Vyhodnocení výsledků 57
4.4.1 Metody hodnocení modelu 57
4.4.2 Metody hodnocení jednotlivých obrazů 60

Závěr **67**

Literatura **69**

Seznam symbolů a zkratek **73**

Seznam obrázků

1.1 Model umělého neuronu 21
1.2 Architektura jednoduché neuronové sítě 23
2.1 Hluboké učení v rámci umělé inteligence 25
2.2 Architektura konvoluční neuronové sítě 26
2.3 Ukázka 3D barevné matice 27
2.4 Funkce konvolučního jádra 27
2.5 Funkce sdružovací vrstvy 29
2.6 Aktivační funkce sigmoid 30
2.7 Aktivační funkce hyperbolický tangens 31
2.8 Aktivační funkce ReLU 31
2.9 Aktivační funkce leaky ReLU 32
2.10 Funkce dropout vrstvy 33
3.1 Princip GAN 43
4.1 Diskriminátor 51
4.2 Generátor 53
4.3 Ztráta diskriminátoru 55
4.4 Penalizace gradientu 56
4.5 Ztráta generátoru 56
4.6 Vygenerované obrazy 58
4.7 Nejbližší obraz kosinusové podobnosti 62
4.8 Nejbližší obraz PSNR 63
4.9 Nejbližší obraz SSIM 64
4.10 Nejbližší obraz kombinovaných metrik 66

Seznam tabulek

4.1 Tabulka incepčního skóre 59
4.2 Tabulka skóre FID 60
4.3 Tabulka rozložení skóre generovaných vzorků 65

Úvod

Během posledních deseti let se umělá inteligence (UI) díky většímu množství dostupných dat, optimalizaci algoritmů a neustálému pokroku ve zvyšování počítačového výkonu stala stále schopnější výkonu lidské činnosti. Jedním z odvětví v rámci UI je generování obrazů, kde jsou počítače schopny vytvářet obsah podobný fotografiím či lidské tvorbě. Tato schopnost otevírá nové možnosti v oblasti tvorby dat, například použití syntetických snímků pro naplnění datových sad sloužícím pro trénování neuronových sítí.[8]

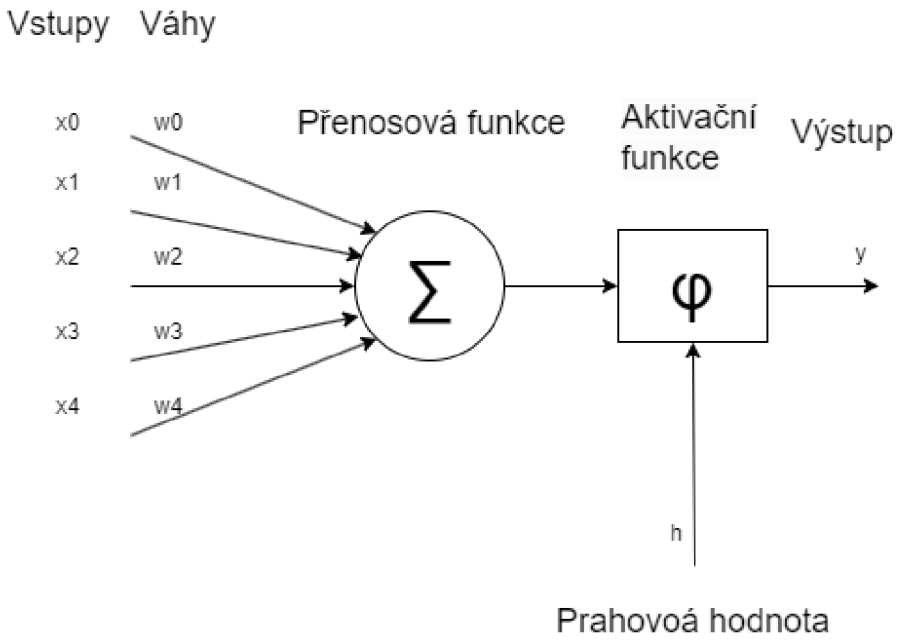
Hlavními cíli této práce bylo implementovat a vytvořit funkční model Wassersteinovi generativní adverzní sítě s penalizací gradientu (WGAN-GP) pro generování obrázků. Implementace zahrnovala přípravu vhodné datové sady, návrh architektury modelu a následné úspěšné trénování GAN. K dosažení tohoto cíle byl zvolen programovací jazyk Python ve spojení s knihovnou Tensorflow a v ní obsažené API Keras, implementace byla provedena v prostředí Google Collab.

První dvě části se zabývají problematikou hloubkových neuronových sítí, se zaměřením na konvoluční neuronové sítě (CNN) a metody učení v rámci této architektury. Třetí část se věnuje generativním adverzním sítím (GAN) s důrazem na proces trénování, optimalizace a různé varianty této architektury. Poslední kapitola popisuje implementaci kódu a trénování na datové sadě spolu s vyhodnocením modelu. Dále jsou ukázány výsledky trénování a vygenerované obrázky.

1 Neuronové sítě

Umělé neuronové sítě jsou algoritmy patřící do kategorie strojového učení. Tyto abstraktní modely propojených uzlů, takzvaných neuronů, umožňuje řešit složité úlohy pomocí počítačů. Tyto umělé neurony napodobují funkci skutečných neuronů v lidském mozku. Neuronové sítě jsou velmi aktivní oblastí výzkumu v rámci umělé inteligence.[18]

Umělá neuronová síť se skládá z neuronů, které dostávají informace buď od jiných neuronů nebo z vnějšku neuronové sítě. Tato data jsou následně modifikována a předána jako výstup. Neurony můžou mít více vstupů, ale jen jediný výstup, který ovšem může být předáván vícekrát.[18]



Obr. 1.1: Model umělého neuronu.

Neuron se skládá z libovolného počtu vstupů, přičemž každý vstup má určenou váhu, která reprezentuje, jak moc velký ohled na něj bude brán v porovnání s ostatními vstupy. Z tohoto hlediska neuron připomíná lineární regresní model. Výsledek transformační funkce se porovná s prahovou hodnotou h (threshold) a pokud je vyšší, tak se „aktivuje“ a pošle na výstup.[18]

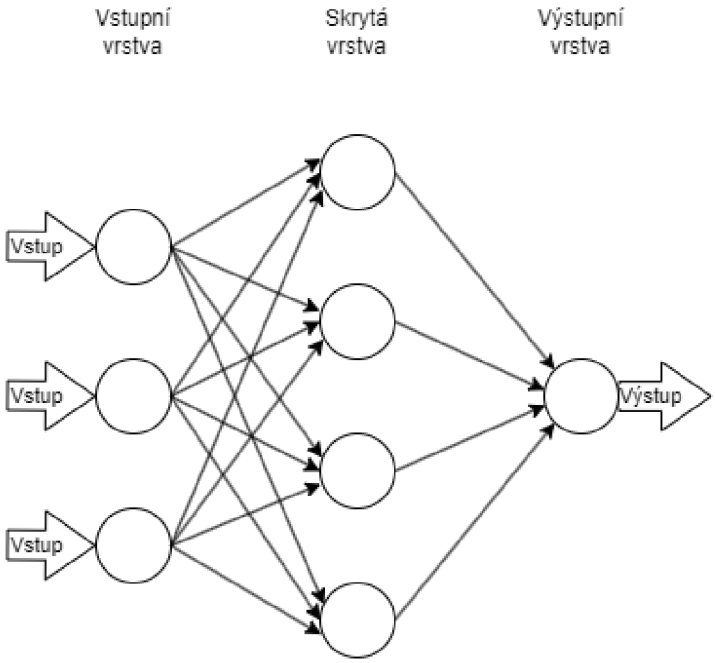
1.1 Architektura

Neurony jsou v neuronové síti uspořádány do vrstev, které se obvykle skládají ze vstupní vrstvy, jedné nebo více skrytých vrstev a výstupní vrstvy. Vstupní vrstva přijímá výchozí data, zatímco skryté vrstvy zpracovávají informace pomocí řady vážených výpočtů a aktivačních funkcí. Výstupní vrstva vytváří konečný výsledek nebo předpověď na základě zpracovaných informací ze skrytých vrstev.

Spojení mezi neurony jsou reprezentovány váhami, které se upravují během procesu trénování, aby se optimalizoval výkon sítě. Této úpravy se dosahuje iteračním procesem známým jako zpětné šíření, kdy se síť učí ze svých chyb aktualizací vah na základě vypočtené chyby. Jemným doladěním vah a zkreslením neuronů se neuronová síť může naučit zobecňovat a vytvářet přesné předpovědi na nových, dosud neviděných datech.

K funkci sítě slouží tři různé vrstvy, z nichž každé lze přiřadit typ neuronu. Pro vstup dat je to vstupní vrstva, pro výstup dat slouží výstupní vrstva a uprostřed se nachází libovolný počet tzv. skrytých vrstev.[1]

- **Vstupní vrstva**: Tato vrstva poskytuje vstupní data. Data jsou zpracována vstupními neurony, je jim přiřazena patřičná váha, a poté jsou odeslána do následující vrstvy.[2][31]
- **Skrytá vrstva**: Tato vrstva se nachází mezi vstupní a výstupní vrstvou. Skrytých vrstev může být libovolný počet na rozdíl od vstupní a výstupní vrstvy, která je právě jedna. Zde se datům opět přiřazuje váha, než jsou odeslána z neuronu na neuron až do výstupní vrstvy, přičemž každá vrstva skryté vrstvy má jinou váhu. Na rozdíl od vstupní a výstupní vrstvy, kde jsou příchozí a odchozí data viditelná, je vnitřek neuronové sítě pro vnějšího pozorovatele neviditelný. Odtud pochází termín „skrytá vrstva“.[2][31]
- **Výstupní vrstva**: Výstupní vrstva je poslední vrstvou a následuje hned za poslední úrovní skryté vrstvy. Výstupní neuron nebo neurony obsahují výsledné rozhodnutí, které se projevuje jako tok informací.[2][31]



Obr. 1.2: Architektura jednoduché neuronové sítě.

1.2 Historie

Historie neuronových sítí započala v první polovině 20. století, kdy se objevil koncept konektivismu jako model pro napodobení fungování mozku. V roce 1943 představili Warren McCulloch a Walter Pitts tuto myšlenku pomocí jednoduchého elektrického obvodu a položili tak základ pro začátek chápání nervových cest, jejichž posilování opakovaným používáním poté navrhl v roce 1949 Donald Hebb.

Mezi klíčové předchůdce neuronových sítí patří „prahová logika“, která převádí spojitý vstup na diskrétní výstup, a „Hebbovo učení“, model založený na nervové plasticitě. V padesátých letech 20. století byla na MIT vytvořena první Hebbova síť, což bylo významným milníkem v dosavadním vývoji.

Významná také byla práce Franka Rosenblatta na perceptronech na konci padesátých let dvacátého století. Perceptron, inspirovaný McCulloch-Pittsovými neurony, zavedl koncept učení prostřednictvím po sobě jdoucích vstupů, i když tento model byl omezený na lineárně oddělitelné třídy.

V šedesátých letech dvacátého století se kolem neuronových sítí objevily první úspěchy, jako jednovrstvá síť ADALINE a pokročilejší trojvrstvá MADALINE, vyvinuté Bernardem Widrowem a Marcianem Hoffem na Stanfordu. Modely měly ovšem problémy, a to nepraktickou dobu běhu a neschopnost naučit se nelineární obvody. Tyto důvody bez dalších významných pokroků znamenaly dočasný úpadek zájmu o umělou inteligenci.

Další pokrok neuronových sítí nastal počátkem osmdesátých let dvacátého století, kdy Jon Hopfield představil Hopfieldovu síť a Japonsko se začalo věnovat výzkumu umělé inteligence páté generace. Klíčovou roli při oživení oboru sehrálo znovuobjevení a zdokonalení metody zpětného šíření, která slouží k úpravě vah v neuronových sítích. První poznatky Paula Werbose spolu s následnými příspěvky Rumelharta, Hintona a Williamse položily základy moderních technik trénování neuronových sítí.

V devadesátých letech dvacátého století se neuronové sítě staly středem zájmu díky pokroku v algoritmech zpětného šíření a gradientního sestupu. Dnes neuronové sítě překonaly očekávání, způsobily revoluci v různých oblastech a podnítily nové diskuse o důsledcích umělé inteligence pro společnost. Navzdory pozoruhodnému pokroku však cesta ke skutečně inteligentním strojům pokračuje a připomíná nám, jak složité je replikovat funkci lidské mysli.[19]

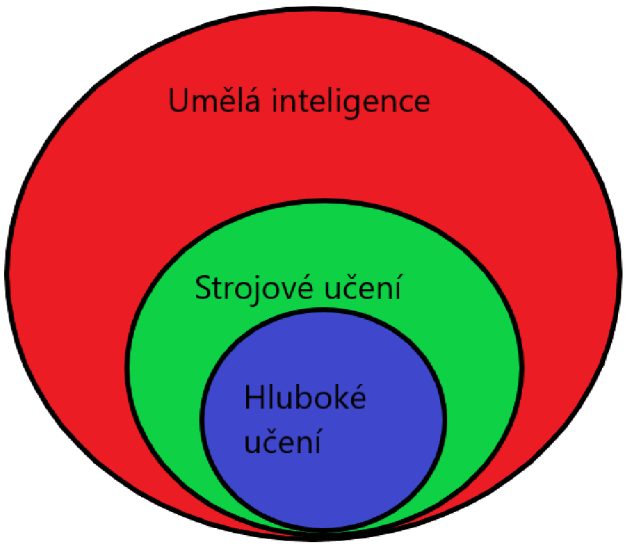
2 Hluboké neuronové sítě

Mnoho prvků moderního života je poháněno technologií strojového učení, včetně vyhledávačů, filtrování obsahu na sociálních sítích a vybraných překladačů. Algoritmy strojového učení se používají k rozpoznávání objektů na fotografiích, převodu hlasu na text, přiřazování produktů, příspěvků a zpráv k zájmům uživatelů a k výběru relevantních výsledků vyhledávání. To vše je možné pomocí předem naučených neuronových sítí, které dokážou vstupní data po průchodu sítí vhodně klasifikovat.[20]

2.1 Hluboké učení

Od strojového učení, jehož je hluboké učení podoblastí, se liší v typu vstupních dat a metodou učení. Pro konvenční metody strojového učení bylo zpracování surových přirozených dat mimo jejich možnosti. Po mnoho let bylo nutno data předzpracovat, většinou ručním vytvořením vektoru příznaků, který daná data vnitřně reprezentoval.[20][16]

Hloubkové učení umožňuje funkci bez některých kroků předzpracování a dokáže pracovat s nestrukturovanými daty, jakými jsou například obrázky. Toto je možné díky automatizaci extrakce příznaků, která probíhá ve vrstvách neuronové sítě spolu s klasifikací bez závislosti na lidech.[20][16]



Obr. 2.1: Hluboké učení v rámci umělé inteligence.

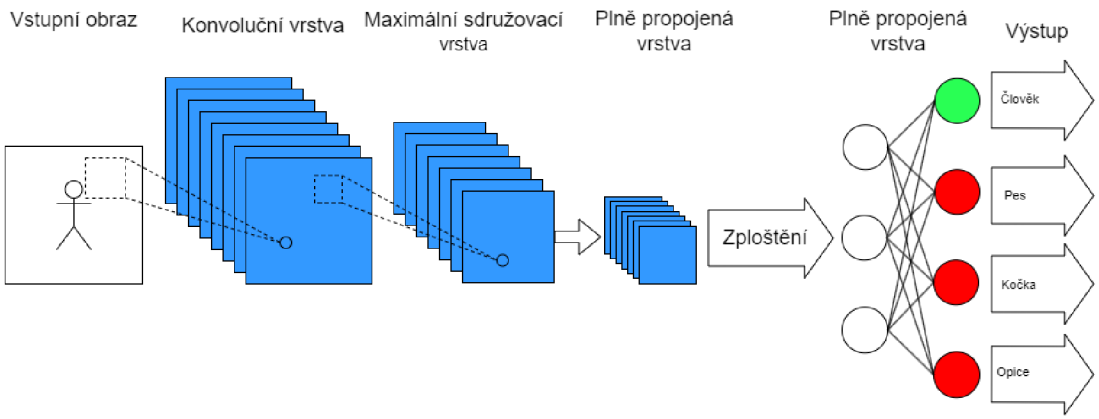
Hluboké učení se používá v neuronových sítích pro vytvoření takzvaných hloubkových neuronových sítí. Takto se nazývá každá neuronová síť, která má kromě vstupní a výstupní vrstvy alespoň dvě skryté vrstvy.

Neuronová síť funguje pomocí dvou opakujících se operací. První operací je dopředná propagace, kde vstupní data projdou všemi vrstvami neuronů až na výstup, kde se vypočítá ztráta sítě. Tato ztráta je pomocí zpětné propagace poslána pozpátku, kde se podle ní mění jednotlivé váhy. Pomocí těchto ztrát a změn se algoritmus učí.[16]

2.2 Konvoluční neuronové sítě (CNN)

Konvoluční neuronová síť je technologie, která napodobuje strukturu lidského optického nervu a je algoritmem, který se dokáže automaticky naučit funkce od zpracování obrazu až po rozpoznávání znaků, obrazů a objektů a jejich prostorové orientace. Konvoluční neuronové sítě mohou plně využívat dvourozměrná obrazová data a ve srovnání s jinými strukturami neuronových sítí pro hluboké učení vykazují dobré výsledky v oblasti obrazu a klasifikace.[29] [17]

CNN používá ve své činnosti tři typy vrstev, a to konvoluční (convolution), sdružovací (pooling) a plně propojenou (fully connected). První vrstvou konvoluční neuronové sítě je vždy konvoluční vrstva. Konvoluční vrstvy mohou být následovány dalšími konvolučními nebo sdružovacími vrstvami zatímco plně propojená vrstva je vždy poslední a žádná vrstva za ní být nemůže.[29] [17]

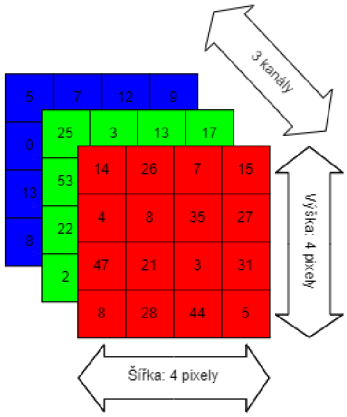


Obr. 2.2: Architektura konvoluční neuronové sítě.

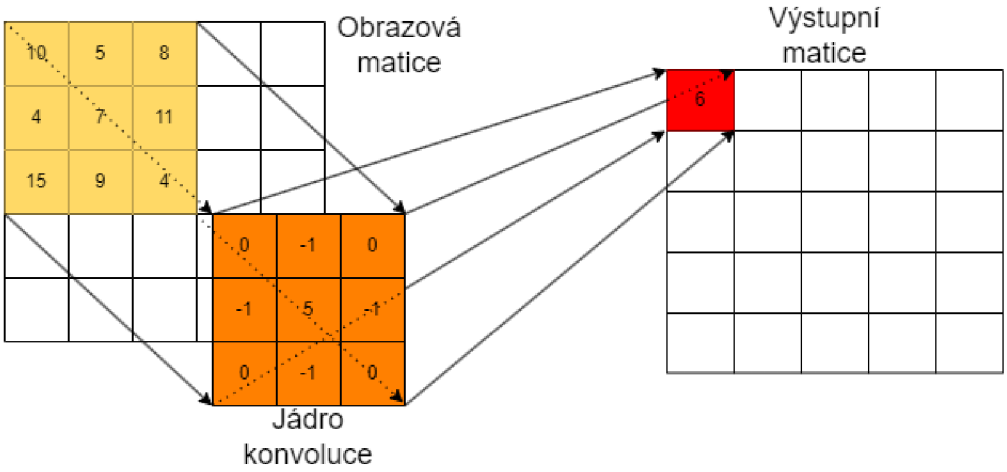
2.2.1 Konvoluční vrstva

Konvoluční vrstva je hlavní složkou konvoluční neuronové sítě a je vrstvou, v níž se provádí většina výpočtů. Zahrnuje vstupní data, filtr také známý jako konvoluční jádro (kernel) a mapu příznaků. Černobílý vstup je reprezentován jako 2-D matice

pixelů (výška, šířka). Přidáním barevných kanálů se do matice přidá hloubka, která je pro typický barevný obrázek RGB rovna třem. [17]



Obr. 2.3: Ukázka 3D barevné matice kde výška a šířka jsou 4 pixely a hloubka je reprezentována třemi barevnými kanály.



Obr. 2.4: Konvoluční jádro projíždí přes vstupní matici a po provedení konvoluční operace zapíše výsledek na výstupní matici.

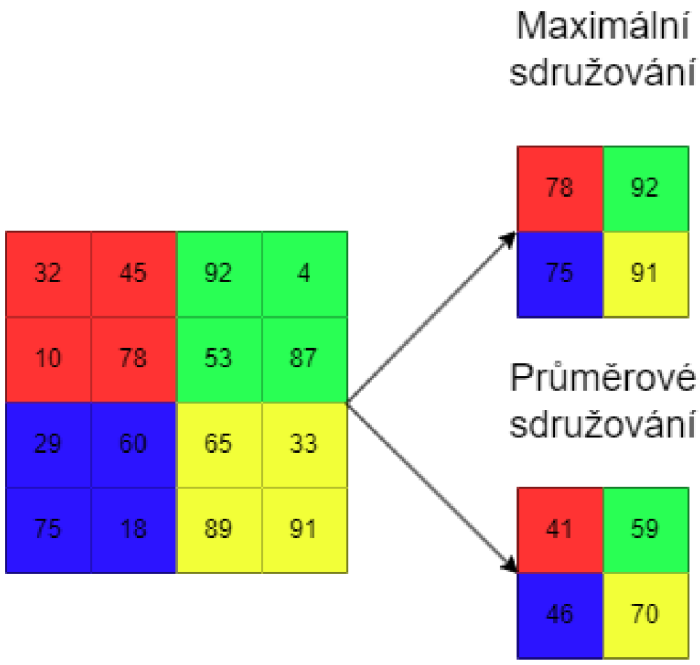
Jak je vidět na obrázku výše, jádro konvoluce se pohybuje po receptivních polích matice obrazu a zpracovává prvky pomocí konvoluce. Pohyb probíhá horizontálně po řadách, přičemž při dosažení konce řady se jádro posune vertikálně dolů. Jádro, kterým obvykle bývá matice 3×3, aplikuje bodový součin na vstupní pixely a hodnoty filtru a vytváří mapu příznaků. Tento proces se opakuje s posunem filtru o zadaný krok, čímž se pokryje celý obraz a vytvoří se mapa prvků.[29] [17]

Váhy detektoru příznaků zůstávají při pohybu po obraze konstantní, což je metoda známá jako sdílení parametrů. Při trénování se ke změně parametrů, jako jsou hodnoty vah, používá gradientní sestup a zpětná propagace. Než se však neuronová síť začne trénovat, je třeba zadat tři hyperparametry, které ovlivňují velikost objemu výstupu. Mezi ně patří následující.[17]

- **Počet filtrů**: Znázorňuje hloubku, barevné kanály, výstupní matice konvoluce. Například pro barevný výstup RGB jsou vytvořeny 3 mapy příznaků.
- **Krok**:Krok jádra je počet pixelů, o které se jádro konvoluce posune při jeho pohybu přes vstupní matici. Větší krok vede ke zmenšení výstupní mapy příznaků, i když hodnoty kroku dva nebo více jsou u běžných konvolučních neuronových sítí neobvyklé. Takový krok se používá v generujících sítích.
- **Ohraničení nulami** (zero padding): Tato technika spočívá v doplnění okraje obrázku nulami před samotným konvolučním procesem. Tento postup s nulovým paddingem pomáhá zachovat informace na okrajích obrázku a je často využíván k vylepšení schopnosti poznání příznaků v datech. Existují tři typy ohraničení.
  - **Platné ohraničení**(Valid padding): V tomto případě k žádnému přidávání nulových okrajů nedochází, a proto je výstup obvykle menší než vstup. Pokud se jádro nemůže dále posunout do určité strany, tak jsou hodnoty vynechány.
  - **Stejné ohraničení** (Same padding): Přidává nulové řady a sloupce na okraje vstupní matice a má za cíl zachovat rozměry původního vstupu.
  - **Plné ohraničení** (Full padding): Přidání více vrstev ohraničení za účelem, aby byl výstup konvoluce větší než její vstup.

2.2.2 Sdružovací vrstva

Sdružovací vrstvy jsou často umístěny za konvolučními vrstvami. Sdružovací vrstva, na rozdíl od konvoluční vrstvy, která extrahuje rysy výpočtem, shromažďuje rysy extrakcí reprezentativních hodnot mezi pixely v daném rozsahu. V sdružovací vrstvě lze reprezentativní hodnoty extrahovat pomocí dvou různých metod, podle maxima (max pooling) a nebo podle průměru (average pooling).[29] [17]



Obr. 2.5: Funkce sdružovací vrstvy.

Například při změně velikosti obrázku 4×4 na obrázek 2×2 se zjednoduší analýza tvaru z jednotlivých pixelů, na úkor kvality obrazu. Výhodou sdružovací vrstvy je tedy její schopnost zvýraznit určité charakteristiky obrazu při zmenšení jeho velikosti.[29] [17]

2.2.3 Aktivační vrstva

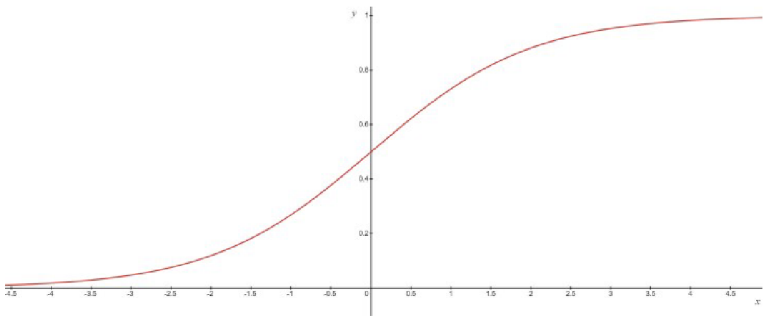
Aktivační funkce rozhodují o aktivaci neuronu na základě jeho vstupu do sítě. Tyto funkce používají matematické operace k posouzení důležitosti vstupu. Pokud je vstup vyšší než hraniční hodnota, tak se neuron aktivuje.

V neuronových sítích neurony zpracovávají vstupní signály a podle toho reagují. Aktivační funkce, odrážející rozhodovací proces mozku, transformují vážený vstup x na výstup y pro další vrstvy nebo jako konečný výsledek v hlubokém učení.

Protože reálná data jsou zřídkakdy lineární, aktivační funkce slouží k vložení nelinearity do sítě. Slouží k výpomoci mapování vstupního proudu neznámého rozložení a měřítka na známý. Nelineární aktivační funkce vnášejí do neuronových sítí složitost, čímž zlepšují výsledky komplikovanějších sítí.[24]

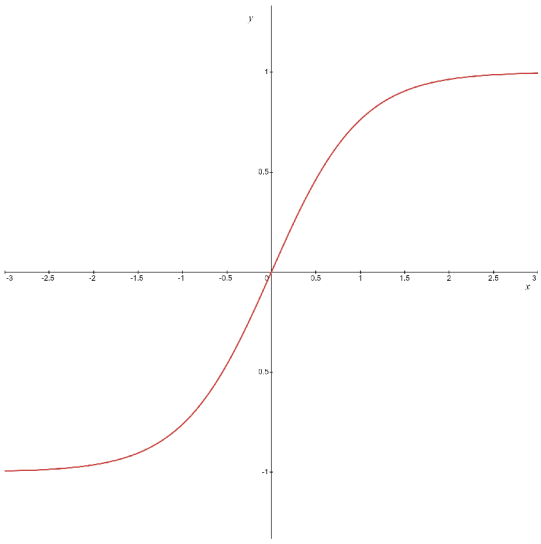
V rámci konvolučních neuronových sítí a této práce jsou nejdůležitější následující aktivační funkce.[24][25]

- **Sigmoid**: Aktivační funkce sigmoid má omezený rozsah hodnot (0,1), což ji činí vhodnou zejména pro výstupy neuronů v rozmezí úrovně pravděpodobnosti. Avšak, při použití sigmoidní aktivační funkce se setkáváme se dvěma klíčovými problémy, které vznikají při zpětné propagaci, a to mizením a explozí gradientů.
  - Mizení gradientů nastává, když gradient postupně klesá s hloubkou konvolučních vrstev. Tento jev může vést k minimálním úpravám vah a v extrémních případech až k selhání schopnosti učení neuronové sítě.
  - Exploze gradientů se na druhou stranu objevuje, když gradienty postupně rostou, což může vést k příliš velkým aktualizacím vah a nestabilitě učení. Tato situace může ovlivnit kvalitu a rychlost trénování sítě.



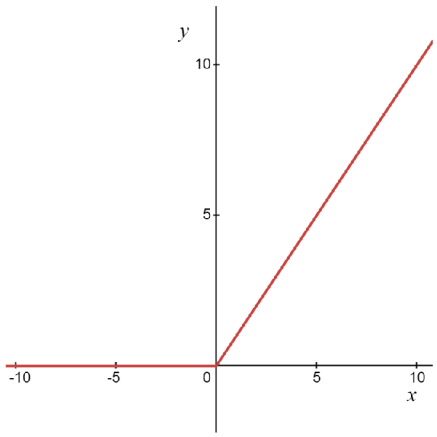
Obr. 2.6: Aktivační funkce sigmoid.

- **Tanh**: Hyperbolický tangens oproti sigmoidu nabývá hodnot v rozmezí (-1, 1), což poskytuje symetričtější výstupy kolem nuly. I přes tuto výhodu má stejné nevýhody při použití příliš malých nebo příliš velkých vstupů.



Obr. 2.7: Aktivační funkce hyperbolický tangens.

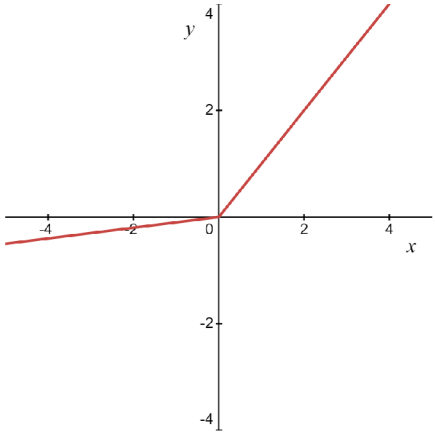
- **ReLU**: Aktivační funkce Rectified Linear Unit (ReLU) nabývá hodnoty nula pro všechny negativní vstupy a nechává kladné vstupy nezměněné. ReLU dokáže efektivně aktualizovat váhy modelu a konverguje šestkrát rychleji než aktivační funkce typu sigmoid.
  Výhodou ReLU je eliminace problému mizení gradientů. Vyskytuje se však jiný problém, a to takzvané „dying ReLU“. Při tomto jevu neuron přestane reagovat na vnější vstup a konstantně generuje nulu tzv. umírání neuronu, a tak brání efektivnímu učení modelu.



Obr. 2.8: Aktivační funkce ReLU.

- **Leaky ReLU**: Tato funkce byla vytvořena modifikací standardní funkce ReLU v reakci na problém umírání neuronu. Toho je dosaženo malým průtokem

hodnot pro negativní vstupy, sloužící k zachování části informace obsažené v záporných vstupech.



Obr. 2.9: Aktivační funkce leakyReLU.

2.2.4 Normalizace dávek

Normalizace je často používána v konvolučních neuronových sítích k normalizaci vstupu každého neuronu tak, aby měl nulový průměr a směrodatná odchylka se blížila jedné. Tento postup pomáhá stabilizovat proces učení a eliminovat problém interního kovariantního posunu, který nastává, když se distribuce vstupů do vrstvy mění během trénování. Normalizace se provádí odděleně pro každou dimenzi (vstupní neuron) v tzv. „dávkách“ a ne současně pro všechny dimenze.[27][25]

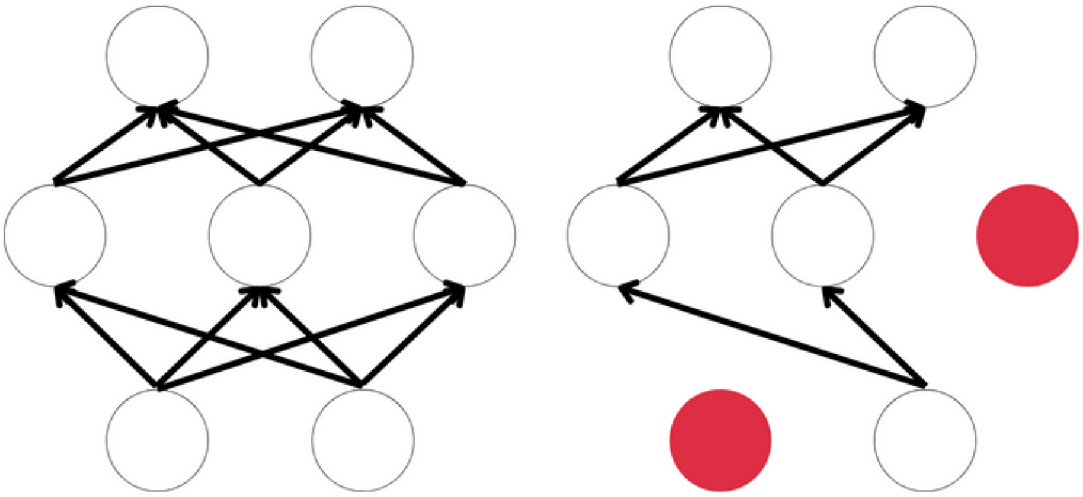
2.2.5 Plně propojená vrstva

Přidání plně propojené vrstvy slouží k učení nelineární kombinace rysů extrahovaných konvoluční vrstvou. Tato vrstva vytváří přímá spojení mezi každým uzlem ve výstupní vrstvě a každým uzlem ve vrstvě předchozí.

Plně propojená vrstva je primárně zodpovědná za klasifikaci, a pro tuto činnost využívá funkce z předchozích vrstev a různorodé filtry. V konvolučních a sdružovacích vrstvách klasifikačních sítí jsou používány funkce ReLU, plně propojené vrstvy obvykle používají aktivační funkci softmax, která klasifikuje vstupy a generuje pravděpodobnosti v rozmezí 0 až 1. Prostřednictvím zpětné propagace v průběhu trénování se model zlepšuje v rozlišování dominantních a specifických příznaků nízké úrovně v obrazech, což zlepšuje přesnost klasifikace.[17][29]

2.2.6 Dropout vrstva

Tato vrstva funguje na principu náhodné deaktivace určitého počtu neuronů během každé iterace trénování. Tato deaktivace pomáhá zabránit neuronům ve specializaci na konkrétní vzory, a tak pomáhá při náhlých změnách vstupních dat. Tato vrstva tedy slouží k snížení rizika přetrénování. Je nutné dát pozor, protože při použití normalizace dávek po dropout vrstvě se model stává nestabilní. Je proto vhodné tuto vrstvu použít až na konci na plně propojených vrstvách.[25][28]



Obr. 2.10: Funkce dropout vrstvy. Obrázek nalevo představuje neuronovou síť bez použití funkce dropout a obrázek vpravo ukazuje síť po požití funkce dropout kde jsou deaktivované neurony zobrazeny červeně.

2.3 Učení neuronových sítí

Jak již bylo zmíněno výše, učení neuronových sítí se provádí úpravou vah, konkrétně v procesu zpětné propagace. Aby se vědělo, o kolik se mají jednotlivé váhy upravit pro zlepšení výsledků neuronové sítě, je nutné spočítat chybu modelu, k čemuž se používá ztrátová funkce.

2.3.1 Druhy učení

Učení s učitelem

Učení s učitelem je základním konceptem strojového učení, kdy je model trénován na označených datech, aby se naučil mapování mezi vstupními a odpovídajícími výstupními znaky. Při učení s učitelem se algoritmus učí ze souboru dat, kde je každý vzorek spárován s cílovou třídou, což modelu umožňuje provádět předpovědi nebo klasifikace na neznámých datech na základě naučených znaků z trénovacího souboru. Tento přístup se široce používá v různých aplikacích, jako je rozpoznávání obrazu, zpracování přirozeného jazyka a regresní úlohy.

Během fáze trénování se model učí základní příznaky a vztahy v datech tím, že upravuje své parametry tak, aby minimalizoval rozdíl mezi předpovězenými výstupy a skutečnými třídami. Mezi běžné algoritmy používané při učení pod dohledem patří lineární regrese, rozhodovací stromy, stroje s podpůrnými vektory, neuronové sítě a další.

Jakmile je model natrénován, je ověřen na samostatné validační nebo testovací sadě dat, aby byla zhodnocena jeho výkonnost a schopnost generalizace. K hodnocení výkonnosti modelu se běžně používají metriky, jako je přesnost, skóre F1 a střední kvadratická chyba.[26]

Typy modelů

- **Klasifikační**: V klasifikačních úlohách model předpovídá diskrétní označení třídy pro každý vstupní vzor. Mezi klasifikační algoritmy patří logistická regrese, stroje s podpůrnými vektory, random forest a neuronové sítě.[26]
- **Regresivní**: Regresní úlohy zahrnují předpovídání spojité číselné hodnoty na základě vstupních znaků. Pro regresní úlohy se běžně používají lineární regrese, polynomiální regrese, rozhodovací stromy a neuronové sítě.[26]
- **Klasifikační s více třídami**: V klasifikaci s více třídami model předpovídá více tříd pro každý vstupní vzor. Toto je běžné používáno v případech, kde každý vzor může patřit do více tříd současně.[26]

Učení bez učitele

Učení bez učitele je druhým z přístupů strojového učení, kdy algoritmy samostatně získávají vzory a struktury v neoznačených a nezatříděných datech bez explicitního vedení. Tento přístup umožňuje modelům nezávisle zkoumat data a odhalovat skryté vztahy, shluky nebo reprezentace bez potřeby předem definovaných tříd. Existují mnohé techniky, které se používají v různých aplikacích k získání informací a vzorů z nezpracovaných dat.

Shlukovací algoritmy, jako je například K-Means, rozdělují data do různých skupin na základě podobnosti a snaží se minimalizovat vzdálenost mezi datovými body a centry shluků pomocí iterativního přiřazování a aktualizace. Tato technika je používána pro seskupování datových bodů do shluků za účelem identifikace základních vzorů či vztahů v souboru dat.

Metody redukce dimenzionality, jako je analýza hlavních komponent, transformují vysokorozměrná data do méně rozměrného prostoru zachycením nejvýznamnějších informací prostřednictvím ortogonálních komponent. Tato metoda zjednodušuje složité soubory dat, čímž je činí lépe zvládnutelnými pro analýzu, přičemž zachovává podstatné rysy a snižuje redundanci.

Autoenkodéry, které patří mezi modely neuronové sítě, se používají v úlohách učení bez učitele k zakódování vstupních dat do komprimované reprezentace a jejich rekonstrukci zpět na původní vstup. Autoenkodéry jsou efektivní pro úlohy, jako je komprese dat, učení příznaků a detekce anomálií, kde se model učí efektivně reprezentovat data v méně rozměrném prostoru.

K identifikaci odlehlých hodnot nebo nepravidelností v datech, které se výrazně odchylují od normy, se využívají techniky detekce anomálií, jako jsou metody Isolation Forest a jedno třídní stroje s podpůrnými vektory. Algoritmy detekce anomálií pomáhají odhalováním neobvyklých vzorů nebo případů a dokážou označit potenciální chyby, anomálie nebo podvodné činnosti v souboru dat.

Tyto techniky mají klíčovou roli v různých aplikacích strojového učení v oblastech jako je kybernetická bezpečnost, detekce podvodů, segmentace zákazníků generace dat a další. Zkoumáním přirozené struktury dat bez explicitních tříd poskytují algoritmy cenné poznatky, napomáhají v rozhodovacích procesech a odhalují skryté vzorce, které nemusí být zřejmé při použití technik učení s učitelem.[26]

2.3.2 Ztrátová funkce

Ztrátová funkce je zásadní pro vyhodnocení rozdílu mezi předpovídanými a skutečnými výstupními hodnotami. Kvantifikuje, jak dobře neuronová síť modeluje trénovací data, a směřuje trénování k co nejmenším ztrátám.[32]

Binary Cross-Entropy/Logaritmická ztráta

Úlohou binární klasifikace je zařazení vstupů do jedné ze dvou předem definovaných tříd. V kontextu neuronových sítí model vytváří vektor udávající pravděpodobnost příslušnosti vstupu k daným třídám. Konečná klasifikace je určena výběrem třídy s nejvyšší pravděpodobností.

Při binární klasifikaci mohou hodnoty y (názvy tříd) být pouze 0 nebo 1. K vyčíslení ztráty porovnává ztrátová funkce skutečnou hodnotu s pravděpodobností, že vstup odpovídá dané třídě *p(i)*), kde *p(i)* představuje pravděpodobnost, že vstup spadá do třídy 1, a $1 - p(i)$ je pravděpodobnost, že naopak spadá do třídy 0.[32] Ztrátová funkce je tedy definována jako

$$CEloss = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)\right). \tag{2.1}$$

Wassersteinova ztráta

Wassersteinova ztrátová funkce ve Wassersteinových generativních adverzních sítích (WGAN) je založena na Kantorovichově-Rubinsteinově dualitě a je definována jako:

$$\mathcal{L}(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)]. \tag{2.2}$$

- $L(P_r, P_\theta)$ představuje Wassersteinovu ztrátovou funkci.
- P_r je rozdělení skutečných dat.
- P_θ je rozdělení generovaných dat.
- f je 1-Lipschitzova funkce.
- $\|f\|_L$ označuje Lipschitzovu konstantu funkce f.
- $\mathbb{E}_{x \sim P_r}[f(x)]$ označuje očekávánou distribuci nad reálnými vzorky dat.
- $\mathbb{E}_{x \sim P_\theta}[f(x)]$ označuje očekávánou distribuci nad generovanými vzorky dat.

Wassersteinova ztrátová funkce ve WGAN je založena na Kantorovichově-Rubinsteinově dualitě, což je matematický koncept umožňující výpočet Wassersteinovy vzdálenosti optimalizací nad 1-Lipschitzovskými funkcemi. Cílem WGAN je minimalizovat Wassersteinovu vzdálenost mezi skutečným a generovaným rozložením dat maximalizací rozdílu očekávání těchto rozložení nad funkcemi s Lipschitzovskou konstantou 1. Díky tomuto způsobu model zajišťuje zlepšení kvality generovaných vzorků.[3]

2.3.3 Optimalizace učení

Výběr vhodného optimalizačního algoritmu je důležitým krokem a má na trénování zásadní dopad. Kvůli jeho skvělým výsledkům při použití v neuronových sítí je v této práci použit algoritmus RMSProp, který je spolu s algoritmem ADAM vhodný pro trénování sítí typu wasserstein gan.[5]

Gradientní sestup

Gradientní sestup je základní optimalizační algoritmus používaný ve strojovém učení a hlubokém učení sloužící k minimalizaci ztrátové funkce. Hlavním cílem gradientního sestupu je najít optimální sadu parametrů modelu, která minimalizuje danou ztrátovou funkci. Toho je dosaženo úpravou parametrů na základě gradientu ztrátové funkce, který udává směr nejstrmějšího poklesu funkce.

Existují různé typy algoritmů gradientního sestupu, z nichž každý má své vlastní charakteristiky. Dávkový gradientní sestup počítá gradienty s použitím celé trénovací sady dat, což zajišťuje konvergenci ke globálnímu minimu, ale může být pro velké sady dat výpočetně nákladné. Na druhé straně stochastický gradientní sestup aktualizuje parametry pomocí jediného trénovacího příkladu najednou, čímž vzniká velký rozptyl v aktualizacích parametrů, ale také vede k výpočetní efektivitě metody. Minidávkový gradientní sestup dosahuje kompromisu mezi předchozími variantami tím, že aktualizuje parametry pomocí podmnožiny (minidávky) trénovacích dat, čímž kombinuje výhody dávkového i stochastického přístupu.

Rychlost učení je v gradientním sestupu, tak jako u ostatních optimalizačních algoritmů, klíčovým hyperparametrem, který řídí velikost kroků prováděných při aktualizaci parametrů. Zvolení vhodné velikosti tohoto parametru je jedním z nejzákladnějších a nejobtížnějších kroků, protože čím menší míra tím pomalejší konvergence, zatímco velká míra učení může způsobit, že algoritmus překročí lokální minimum a nedojde ke konvergenci. Ke konvergenci při gradientním sestupu dochází, když algoritmus aktualizuje parametry, dokud není dosaženo předem definovaného počtu iterací nebo určité úrovně konvergence.

Gradientní sestup má ovšem své problémy, a to zejména při trénování v hlubokých neuronových sítích, kde dochází k problémům jako jsou mizející či explodující gradienty. Tyto problémy pomáhají řešit techniky, jako je ořezávání a penalizace gradientů a normalizovaná inicializace. Různé optimalizační algoritmy, například Adam, RMSprop a Adagrad, což jsou varianty založené na gradientním sestupu, zahrnují hybnost a adaptivní rychlost učení, sloužící k urychlení konvergence a zlepšení optimalizačního výkonu.

Gradientní sestup je široce používán při trénování neuronových sítí, lineární regrese, logistické regrese a dalších modelů strojového učení. Pochopení gradientního sestupu a jeho variant je klíčové pro efektivní trénování modelů a dosažení optimálního výkonu, jeho předpoklady k zmizení a explozi gradientů jsou ovšem u hlubokých generativních modelů s vysokým rozlišením příliš vysoké a proto je vhodné použít již zmíněné algoritmy Adam a RMSprop.[11]

Adam

Adam (Adaptive Moment Estimation) je široce používaný optimalizační algoritmus v oblasti hlubokého učení. Kombinuje výhody adaptivních rychlostí učení a momentu k urychlení procesu trénování neuronových sítí. Adam funguje na principu výpočtu adaptivních rychlostí učení pro každý parametr jednotlivě pomocí odhadu prvního a druhého momentu gradientů, čímž zvyšuje efektivitu optimalizace.

Algoritmus inicializuje dvě proměnné, m a v, na nulu. Tyto proměnné jsou pak v každé iteraci aktualizovány pomocí gradientů parametrů. První moment, m, představuje exponenciálně se zmenšující průměr minulých gradientů, zatímco druhý moment, v, představuje exponenciálně se zmenšující průměr minulých gradientů na druhou.

Navíc zahrnuje korekci zkreslení, zejména v počátečních fázích tréninku. Tato korekce pomáhá stabilizovat optimalizační proces a zlepšit konvergenci algoritmu.

Jednou z výhod algoritmu Adam je jeho schopnost přizpůsobit rychlost učení pro různé parametry na základě velikosti jejich gradientů. Tato adaptivní povaha umožňuje algoritmu Adam rychleji konvergovat a efektivně zpracovávat řídké gradienty, díky čemuž je vhodný pro širokou řadu úloh hlubokého učení.

Hyperparametry, jako je rychlost učení, $\beta 1$ (rychlost zmenšení pro první moment) a $\beta 2$ (rychlost zmenšení pro druhý moment), mají pro výkon algoritmu Adam klíčovou roli. Vyladění těchto hyperparametrů totiž značně ovlivňuje rychlost konvergence a stabilitu optimalizačního procesu.

Celkově je algoritmus Adam efektivní při optimalizaci nekonvexních účelových funkcí, a proto je vhodný pro trénování hlubokých neuronových sítí. Jeho adaptivní rychlost učení, začlenění hybnosti a robustní výkon přispěly k jeho širokému využití.[11]

RMSProp

RMSProp (Root Mean Square Propagation) je optimalizační algoritmus, který řeší některá omezení algoritmu AdaGrad, zejména v kontextu trénování hlubokých neuronových sítí. RMSprop modifikuje algoritmus AdaGrad tím, že mění akumulaci gradientu na exponenciálně vážený klouzavý průměr, což umožňuje efektivněji přizpůsobovat rychlost učení pro různé parametry.

Klíčovou myšlenkou RMSprop je výpočet klouzavého průměru gradientů na druhou pro každý parametr. Tento klouzavý průměr se používá k úpravě míry učení na základě předchozího stavu gradientů. Díky normalizaci míry učení pomocí gradientů na druhou dokáže RMSprop odstranit problém s mizejícími nebo explodujícími gradienty, které představují problémy při trénování hlubokého učení.

Výhodou metody RMSprop je její schopnost stabilizovat rychlost učení během tréninku. Díky tomu, že RMSprop dělí aktuální gradient druhou odmocninou z nahromaděných gradientů na druhou, zabraňuje příliš velkému nebo příliš malému růstu míry učení, což vede ke stabilnější a efektivnější optimalizaci.

Algoritmus zavádí hyperparametr ρ, který řídí exponenciální rychlost útlumu klouzavého průměru. Tento hyperparametr ovlivňuje, jak rychle algoritmus přizpůsobuje rychlost učení na základě historie gradientů, tedy jak rychle se minulé gradienty přestávají používat. Metoda RMSprop se používá při optimalizaci hlubokých neuronových sítí a poskytuje praktickou a efektivní optimalizační metodu pro trénování složitých modelů jako jsou generativní modely s vysokým rozlišením. Díky své jednoduchosti, efektivitě a stabilitě je použitelná pro stejné úkony jako již zmíněný algoritmus Adam.[11]

3 Generativní neuronové sítě

Generativní modelování, které spadá v rámci strojového učení do odvětví učení bez učitele, zahrnuje automatickou identifikaci vzorů ve vstupních datech za účelem generování nových, věrohodných výstupů.[6]

3.1 Generativní adverzní neuronové sítě

Generativní adverzní sítě (GAN) představují revoluční třídu neuronových sítí v oblasti umělé inteligence. Síť GAN, kterou v roce 2014 představil Ian Goodfellow se svými kolegy, je populární díky skvělé schopnosti generovat realistická data jako jsou obrázky, ale také i jiné formy dat.[6] [10]

GAN používají inovativní přístup k trénování generujícího modelu tím, že k úloze přistupuje jako k problému učení bez učitele se dvěma dílčími pod-modely. Generátor vytváří nové data a diskriminátor rozlišuje mezi skutečnými a vygenerovanými daty.

Generátor a diskriminátor spolu soupeří. Generátor vytváří vzorky, které jsou smíchané s autentickými daty a diskriminátor na základě jejich klasifikace zdokonaluje své schopnosti. Tato hra s nulovým součtem, kde zisk jednoho modelu je ztráta druhého, zahrnuje odměny a tresty, úspěšné klamání generátorem nebo přesná identifikace diskriminátorem a vede k aktualizaci jejich parametrů. Ačkoli ideální scénář zahrnuje dokonalou replikaci, kde si diskriminátor není jistý o skutečnosti dat, praktické aplikace těží z vysoce účinného generátoru, aniž by vyžadovaly dokonalost.[6]

3.1.1 Použití technik učení bez učitele v GAN

Učení bez učitele v generativních adverzních sítích zahrnuje trénování modelu GAN na netříděných datech za cílem generace nových vzorků, které se podobají distribuci tréninkových dat. V kontextu tohoto druhu učení nevyžadují sítě GAN k trénování data zařazené do tříd, protože se učí z distribuce nezpracovaných dat proto, aby vytvářely syntetické vzorky.

Použití této techniky v GAN má různé aplikace, včetně generování obrázků, rozšiřování dat a přenosu stylu. Učením distribuce dat z netříděných dat mohou GAN generovat nové obrázky, vylepšovat stávající obrázky nebo přenášet styly mezi různými obrázky bez potřeby explicitních tříd. Tento neřízený přístup umožňuje GAN zachytit komplexní vzory a struktury v datech, což vede k vytváření rozmanitých a realistických syntetických vzorků.

Kromě toho se učení bez učitele v GAN rozšířilo na scénáře hybridního učení, kdy se malé množství označených dat kombinuje s velkým množstvím neoznačených dat s cílem zlepšit výkonnost modelu. Toto umožňuje nové možnosti v oblasti generování

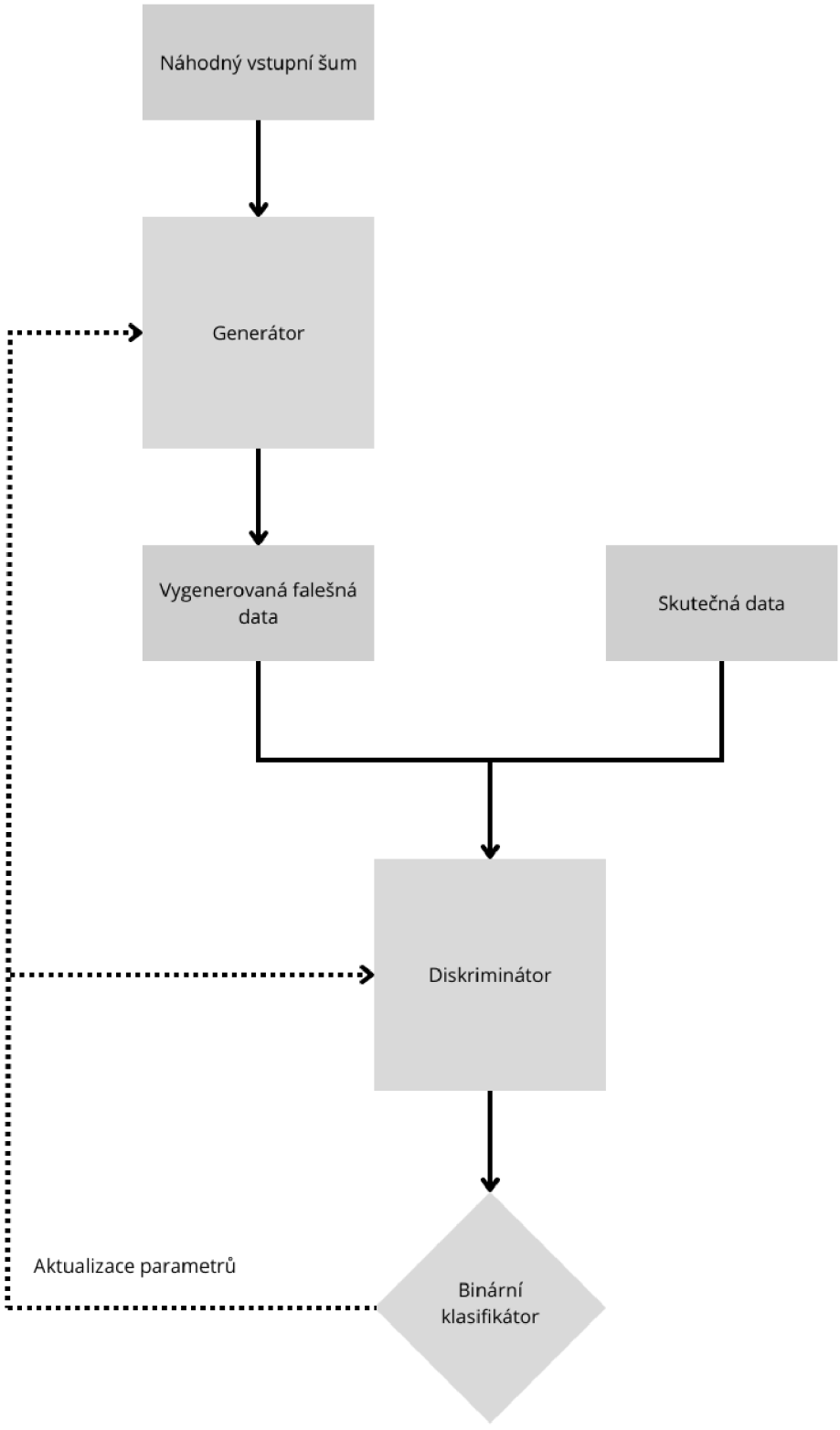
dat, manipulace s nimi a učení reprezentace, což je využitelné v různých oblastech, jako je počítačové vidění, zpracování přirozeného jazyka a další. Tento přístup také umožňuje modely typu text na obraz, kdy se kupříkladu při generování lidských tváří mohou generovat jenom tváře s jednou barvou vlasů při rozřazení trénovacích dat do tříd podle barev vlasů.[26]

3.1.2 Trénování

Trénování funguje jako minmax funkce mezi generátorem a diskriminátorem, kde modely spolu soupeří z cílem najít rovnováhu[10] [9]

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

Cílem je naučit se distribuci generátoru *pg* nad daty x v porovnání s distribucí reálných dat *pdata*. To znamená stanovit prioritu vstupních šumových proměnných *pz(z)* a definovat mapování do datového prostoru *G(z;θg)* kde G je diferencovatelná funkce reprezentovaná vícevrstvým perceptronem s parametry θg. Výstupem druhého perceptronu *D(x; θd)* je skalár představující pravděpodobnost, že x pochází z reálných dat, a není vygenerováno z *pg*. Proces trénování zahrnuje maximalizaci pravděpodobnosti správného označení jak trénovacích příkladů, tak vzorků generovaných G. Současně je G trénován tak, aby minimalizoval log(1 - *D(G(z))*). D a G takzvaně hrají minmax hru s hodnotovou funkcí *V(G, D)*, kde hodnota $\mathbb{E}$ představuje průměrnou hodnotu přes všechny možné hodnoty dané distribuce.[10] [9]



Obr. 3.1: Princip GAN spočívá v tom, že generátor z náhodného šumu vygeneruje data. Následně diskriminátor rozhoduje jestli jsou data skutečná nebo vygenerovaná. Nakonec se podle úspěšnosti diskriminátoru aktualizují parametry obou modelů

Problémy při trénování GAN

Trénování generativních adverzních sítí je z důvodu současného trénování dvou soupeřících modelů náročné. Dosažení rovnováhy je komplikované, protože každá aktualizace jednoho modelu ovlivňuje i druhý model. Nejčastějšími problémy jsou nekonvergence a kolaps modelu, kdy generátor z různých vstupů tvoří ten stejný výstup. Na vzdory nejasné teorii stojící za fungováním GAN existují praktické řešení, které byli empiricky ověřeny.[6]

Tipy pro trénování stabilních GAN

- **Krokovaná konvoluce**: Pro diskriminátor se sdružovací vrstva nahrazuje konvoluční vrstvou s krokem větším než jedna, která umožňuje síti naučit se vlastní prostorové zmenšování. Tento přístup se použije i pro generátor, ale s krokem menším než jedna.[7]
- **Odstranění plně propojených vrstev**: Na rozdíl od ostatních neuronových sítí jsou v GAN plně propojené vrstvy vynechány. U těchto sítí jsou konvoluční vrstvy zploštělé a přímo spojené s výstupní vrstvou. Navíc pro efektivní zpracování je náhodný Gaussův vstupní vektor, který je vstupem pro generátor, přetvořen na vícerozměrný tenzor.[7]
- **Použití normalizace dávek**: Normalizace dávek se doporučuje požívat v všech vrstvách obou modelů, kromě výstupní vrstvy generátoru a vstupní vrstvy diskriminátoru. Na těchto vrstvách se normalizace nepoužívá z důvodu, že použití normalizace dávek na všech vrstvách může vést k oscilaci vzorků a nestabilitě modelu.[7]
- **Použití aktivačních funkcí ReLU, Leaky ReLU a Tanh**: Funkce ReLU je kvůli její vyšší rychlosti použita pro generátor, zatímco pro diskriminátor je použita Leaky ReLU, aby nenastal problém „umírání neuronů“. Tanh se používá ve výstupní vrstvě generátoru. Použití těchto aktivačních funkcí přispívá k řešení problémů při trénování GAN, například problému mizejícího gradientu.[7]
- **Použít optimalizační algoritmus RMSProp nebo Adam**: Pro trénování generátoru i diskriminátoru se používá stochastický gradientní sestup s optimalizací Adam nebo RMSProp. Modely s menším počtem skrytých vrstev využívají hyperparametry jako je rychlost učení, s doporučenou hodnotou 0,0002, a pro algoritmus Adam momentum kde se pro zvýšení stability požívá hodnota 0,5.[7]

3.1.3 Varianty GAN

Vanilla GAN

Takto se nazývá původní GAN, která byla navržena v roce 2014 Ianem Goodfellowem.[4]

DCGAN

Hluboké konvoluční generativní adverzní sítě představují specifický typ GAN, kde generátor a diskriminátor jsou sestaveny výhradně z konvolučních a dekonvolučních vrstev. Tato architektura dosahuje výrazně lepších výsledků ve srovnání s jinými variantami GAN své doby.[4]

Wasserstein GAN

Wassersteinovy generativní adverzní sítě (WGAN) využívají přístup k trénování generativních modelů založený na využití Earth Mover (EM) vzdálenosti, také známé jako Wassersteinova vzdálenost. Vzdálenost EM stanovuje minimální náklady potřebné k transformaci jednoho rozdělení na jiné a nabízí stabilnější a smysluplnější metriku ve srovnání s tradičními divergentními mírami. Zaměřením na optimalizaci Wassersteinovy vzdálenosti sítě WGAN zlepšují stabilitu trénování a poskytují srozumitelnější učební křivku, ze které lze vyčíst informace pro vyladování sítě a jejich hyperparametrů.

Jedním z hlavních aspektů WGAN je prosazení Lipschitzovy kontinuity v modelu kritika nahrazujícího roli diskriminátoru v sítích DCGAN, která hraje klíčovou roli v procesu optimalizace. Na rozdíl od tradičních sítí GAN, které mohou mít problémy s mizejícími gradienty a kolapsem režimu, sítě WGAN používají buď ořezávání nebo penalizaci gradientu, sloužící k slabému omezení Lipschitzovy konstanty v modelu kritika.[3]

- **WGAN s ořezáváním gradientu**: Klasická síť WGAN zavedla ořezávání vah jako mechanismus k vynucení Lipschitzova omezení na síti kritika. Ořezání vah zahrnuje omezení vah kritické sítě na předem definovaný rozsah, obvykle [-c, c], kde c je hyperparametr definující hranice omezení. Ořezávání vah se původně používalo ke stabilizaci tréninku, později se však prokázalo, že vede k nežádoucímu chování a potížím při optimalizaci.

  Jedním z hlavních problémů s ořezáváním vah ve WGAN je možnost explodujících nebo mizejících gradientů během trénování. Tento jev vzniká v důsledku interakce mezi váhovým omezením a ztrátovou funkcí kritika, a z tohoto důvodu je náročné najít optimální práh ořezávání, který by vyvažoval stabilitu a konvergenci sítě.[13]

- **WGAN s penalizací gradientu**: Řešení daných omezení techniky ořezávání vah v síti WGAN bylo koncept gradientové penalty jako alternativní regularizační techniky. Namísto vynucování Lipschitzova omezení prostřednictvím ořezávání vah tato motoda penalizuje normu gradientu kritika vzhledem k jeho vstupu.

  Metoda penalizace gradientů nabízí oproti ořezávání vah také několik výhod. Pomáhá stabilizovat trénink tím, že zabraňuje možnosti explodujících nebo mizejících gradientů, což jsou běžné problémy při ořezávání vah. Penalizace gradientů dále napomáhá síti kritika v učení hladších rozhodovacích hranic bez nutnosti ručního dolaďování prahových hodnot ořezávání. Tento přístup zajišťuje lepší plynulost gradientů, stabilizuje trénování a vylepšuje konvergenční vlastnosti modelu.[13]

Využití vzdálenosti EM v sítích WGAN poskytuje ztrátovou metriku, která lépe vypovídá o kvalitě vygenerovaných vzorků a více vypovídá o stavu trénování. Tato vlastnost WGAN, kdy ztrátová metrika odráží chování při konvergenci, je odlišuje od tradičních variant GAN a ukazuje výhody optimalizace Wassersteinovy vzdálenosti pro účely generativního modelování.

WGAN jsou navíc odolnější a výkonnější ve srovnání s tradičními GAN při různých volbách architektury generátoru. Toto je ovšem spojeno s tím, že čas trénování WGAN bývá oproti DCGAN zhruba o 30% delší. Experimentální výsledky ukazují, že WGAN dokážou efektivně fungovat s různými typy architektur generátorů, včetně konvolučních generátorů DCGAN a generátorů založených na MLP, aniž by docházelo ke kolapsu režimu. Schopnost WGAN generovat různorodé a vysoce kvalitní výsledky napříč různými architekturami prokazuje stabilitu a odolnost při učení modelu. Díky své vysoké stabilitě, která patří mezi nejvyšší v rámci generativních sítí, je WGAN vhodná pro hluboké sítě, kde je kolaps režimu výrazný problém.[3]

4 Vlastní řešení

Cílem této práce bylo vytvoření neuronové sítě vhodné ke generování fotorealistických obrazů tématicky zapadajících do vybrané datové sady. Byla vybrána architektura WGAN-GP, neboť je to varianta GAN, která je k tomuto účelu optimalizovaná díky vysoké stabilitě, jenž je při generování obrazů o vysokém rozlišení nutná. Následují tedy kroky nutné pro úspěšnou implementaci.

4.1 Implementace

Práce je implementována v programovacím jazyce python s pomocí knihovny Tensorflow a v ní obsaženým prostředím Keras. Kroky v implementaci zahrnují předzpracování datové sady, navržení a vytvoření modelů generátoru a diskriminátoru a nakonec trénování kompletního modelu na vybrané datové sadě. Řešení je založeno na tutoriálu pro DCGAN na hlavní stránce Tensorflow[1] a rozšířeno na implementaci WGAN-GP. Tento model byl vybrán z důvodu, že s narůstajícím rozlišením trénovací datové sady, a z toho vyplývajících generovaných obrazů, se zvyšuje pravděpodobnost kolapsu režimu. Z tohoto důvodu je DCGAN, která proti tomuto jevu nemá skoro žádnou ochranu, nevhodná naopak WGAN je velmi stabilní a je skoro nemožné, aby ji tento problém provázel.

4.1.1 Tensorflow

TensorFlow, spolu s rozhraním API Keras, je jedním z předních prostředí pro strojové učení, které je vhodné pro jazyk Python. Rozhraní Keras nástroje TensorFlow zjednodušuje proces pomocí před-připravených vrstev a optimalizátorů, což přispívá k jeho přívětivému uživatelskému prostředí. TensorFlow navíc umožňuje integraci architektur CUDA a cuDNN, čímž využívá paralelní výpočetní schopnosti grafických karet pro rychlejší trénování modelů hlubokého učení.[30]

Verze použité pro tuto práci:

- Python = 3.10.12
- Tensorflow = 2.16.1
- matplotlib = 3.7.1
- skimage = 23.2
- tensorflow-gan = 2.1
- numpy = 1.25.2
- skimage = 0.19.3
- CUDA = 12.2

[1] `https://www.tensorflow.org/tutorials/generative/dcgan`

- cudnn = 9.1.1

4.1.2 Datová sada

Datová sada použitá pro tuto práci zahrnuje celkem 30 000 obrazů lidských tváří z datové sady CelebA-HQ.[21] Tato datová sada byla zvolena z několika klíčových důvodů. Obsahuje dostatečný počet obrazů vysoké kvality, což je jedna zásadních předpokladů pro trénink generativních neuronových sítí, jako jsou WGAN-GP. Původně mají všechny obrazy v této datové sadě jednotné rozlišení 1024×1024 pixelů, neboť by natrénování generativní sítě o takto vysokém rozlišení vyžadovalo ještě větší grafické prostředky, byly obrazy pro účel této práce zmenšeny na velikost 512×512 pixelů. Toto vysoké rozlišení je vybráno kvůli tomu, že umožňuje modelu zachytit jemné detaily a nuance lidských tváří, což je zásadní pro generování realistických obrazů.

obrazy v CelebA-HQ jsou rozmanité a zahrnují širokou škálu výrazů, osvětlení a pozadí, což pomáhá modelu naučit se a reprodukovat široké spektrum rysů lidských obličejů. Kromě rozmanitosti datová sada obsahuje řadu obrazů s drobnými defekty, některé úmyslné sloužící pro zvýšení generalizace modelu a některé neúmyslné způsobené předzpracováním daných obrazů na dané rozlišení 1024×1024.

Datová sada byla použita čistě pro studijní a výzkumné účely a její použití bylo nezaujaté s ekvivalentním využitím všech 30 000 snímků trénovací databáze.

Pro účel trénování modelu program očekává, že složka trénovací datové sady v rozlišení 512×512 bude ve složce s program z názvem 512, případně při kopírování do prostředí Google Collab z Google drivu ve formátu zip s názven 512.zip v hlavním adresáři disku.

4.1.3 Předzpracování dat

Pro načtení a následné předzpracování datové sady je vytvořena funkce `load_images`, která je navržena tak, aby efektivně připravila obrazová data pro trénink generativních neuronových sítí. Tato funkce využívá funkci aplikačního rozhraní TensorFlow, konkrétně z API `tensorflow.keras.utils` `image_dataset_from_directory`,která umožňuje rychlé a paměťově nenákladné načtení obrazů přímo z vybrané složky.

V rámci této funkce jsou obrazy načteny ve specifikovaném rozlišení, které je v tomto případě 512×512 pixelů. Dávky obrazů jsou vytvořeny s velikostí 64, což znamená, že každý tréninkový krok bude zpracovávat 64 obrazů najednou. Důležitý je taky parametr `shuffle`, jehož povolení způsobuje promíchávaní obrazů a tak zabraňuje vytvoření pořadové závislosti v modelu.

Poté je vytvořena normalizační vrstva, která převede pixely z původních hodnot ⟨0,255⟩ na normalizované hodnoty v rozmezí ⟨-1,1⟩. Tato vrstva je poté pomocí funkce `map` aplikována na všechny obrazy v datové sadě.

Nakonec je datová sada předzpracována pomocí metody `prefetch` s parametrem `buffer_size=AUTOTUNE`, což zajišťuje efektivní načtení a minimalizuje dobu čekání během tréninku modelu.

4.1.4 Trénovací hardware

Trénování probíhalo v prostředí Google Collab s 80 gigabajty RAM, a grafickou kartou Tesla A100 (40GB). Díky tomuto bylo možné trénovat model s vysokým rozlišením 512x512 pixelů, ale i přesto je toto rozlišení pro tento typ grafické karty náročné a při použití dostatečného množství filtrů pro dosažení požadované kvality generovaných snímků je možné použít maximální velikost dávky 64.

4.1.5 Parametry trénování

K optimalizaci trénování byl použit optimalizační algoritmus RMSProp s mírou učení nastavenou pro diskriminátor na 5×10^{-5} a pro generátor ze začátku na 1×10^{-5} a po zjištění, že se model okolo epochy 550 přestal konvergovat zvýšena na 2×10^{-5}. Délka dávek byla pro nastavena na 64 z čehož vyplývá, že pro datovou sadu 30 000 obrazů existuje 469 dávek pro každou epochu. Model byl trénován na 950 epochách. Dále je pro stabilní nastavena váha gradientní penalizace λ na hodnotu 10.

Nastavitelné parametry

- `-train`: Počet epoch pro trénování zadaných jako přirozené číslo.
- `-generate`: Počet obrazů k vygenerování zadaných jako přirozené číslo maximálně však 1 000.
- `-batch_size`: Velikost dávky pro trénování. Výchozí hodnota je 64, pro správné trénování je vhodné použít mocniny čísla dva.
- `-grid_size`: Počet obrazů generovaných v každé řadě a sloupci snímku při použití funkce `generate`. Výchozí hodnota je 3.
- `-model_score`: Počet obrazů pro výpočet incepčního skóre a FID. Maximální hodnota je stejná jako velikost trénovací datové sady, což je 30 000.
- `-image_similarity`: Cesta k obrazu pro výpočet SSIM, PSNR a kosinové podobnosti proti všem obrazům trénovací datové sady. Do cesty se automaticky zahrnuje výchozí složka kde je program obsažen.

4.1.6 Návrh a tvorba diskriminátoru a generátoru

Diskriminátor

Funkce `define_discriminator` slouží k tvorbě architektury diskriminátoru v rámci sítě GAN. Architektura je vytvořena požitím modelu `Sequential` z API Keras. Model začíná konvoluční vrstvou `Conv2D` 512 × 512, která má 64 filtrů. Všechny konvoluční vrstvy modelu mají `kernel_size=5`, udávající velikost konvolučního jádra, nastavenou na 5 pixelů a velikost kroku `stride=2` zajišťující zmenšení rozlišení na polovinu tím, že se jádro posouvá krokem o velkosti dvou pixelů. Předposledním sdíleným argumentem těchto vrstev je `padding=same`, který zajišťuje, že prostorové dimenze map příznaků zůstanou po provedení vrstvy stejné. Nakonec je využit inicializátor jádra RandomNormal, který inicializuje váhy z normálního rozdělení se směrodatnou odchylkou 0,02. Tato operace slouží k stabilizování tréninku a dodatečná ochrana proti problémům, jako je mizení nebo exploze gradientů.

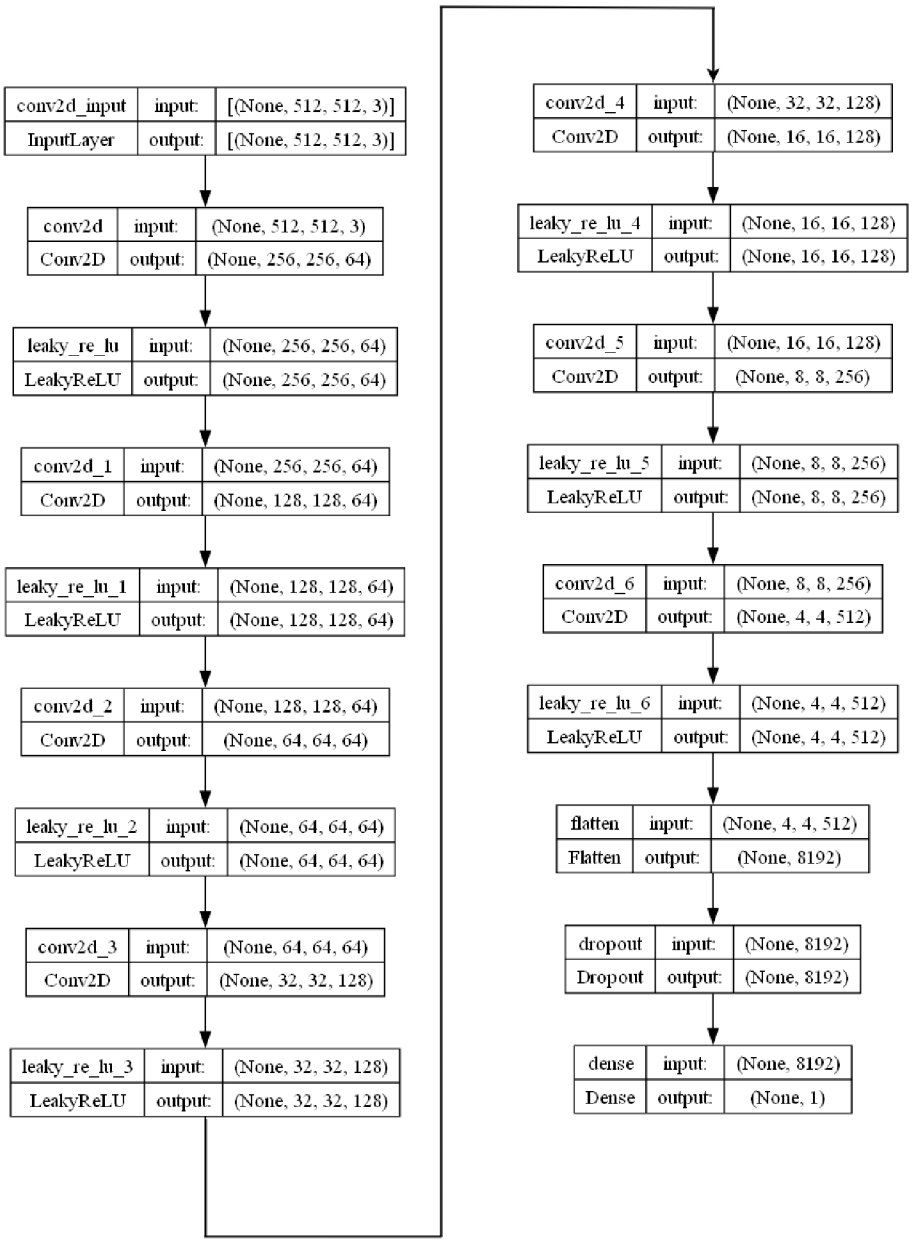
Pro zavedení nelinearity se za každou konvoluční vrstvu přidává aktivační funkce `Leaky ReLU` s parametrem α=0,2, která má stejný počet filtrů jako předcházející konvoluční funkce, v tomto případě 64.

Následujících šest konvolučních vrstev má podobný průběh s jediným rozdílem, a to je počet filtrů, který se postupně zvětšuje od 64 až na 512 filtrů. Každá konvoluční vrstva je, jak bylo řečeno, doprovázena aktivační funkcí `Leaky ReLU`, která zachovává dříve zavedenou nelinearitu.

Nakonec se pomocí funkce `Flatten` 3D matice 4 × 4 převede na vektor a předá plně propojené vrstvě s vysokou hodnotou `dropout=0,5`. Poslední vrstva se skládá z jednoho neuronu a to bez aktivační funkce, neboť sítě WGAN oproti klasickým GAN pracují s lineárním výstupem v intervalu $\langle 0,1 \rangle$, což reprezentuje s jakou pravděpodobností je vstupní obraz vyhodnocen jako skutečný.

Na dalším snímku je konkrétní architektura vygenerovaná pomocí funkce `plot_model` rozšíření graphviz a pydots.[2]

[2] `https://graphviz.org/`



Obr. 4.1: Architektura diskriminátoru vytvořená s pomocí funkce `plot_model`

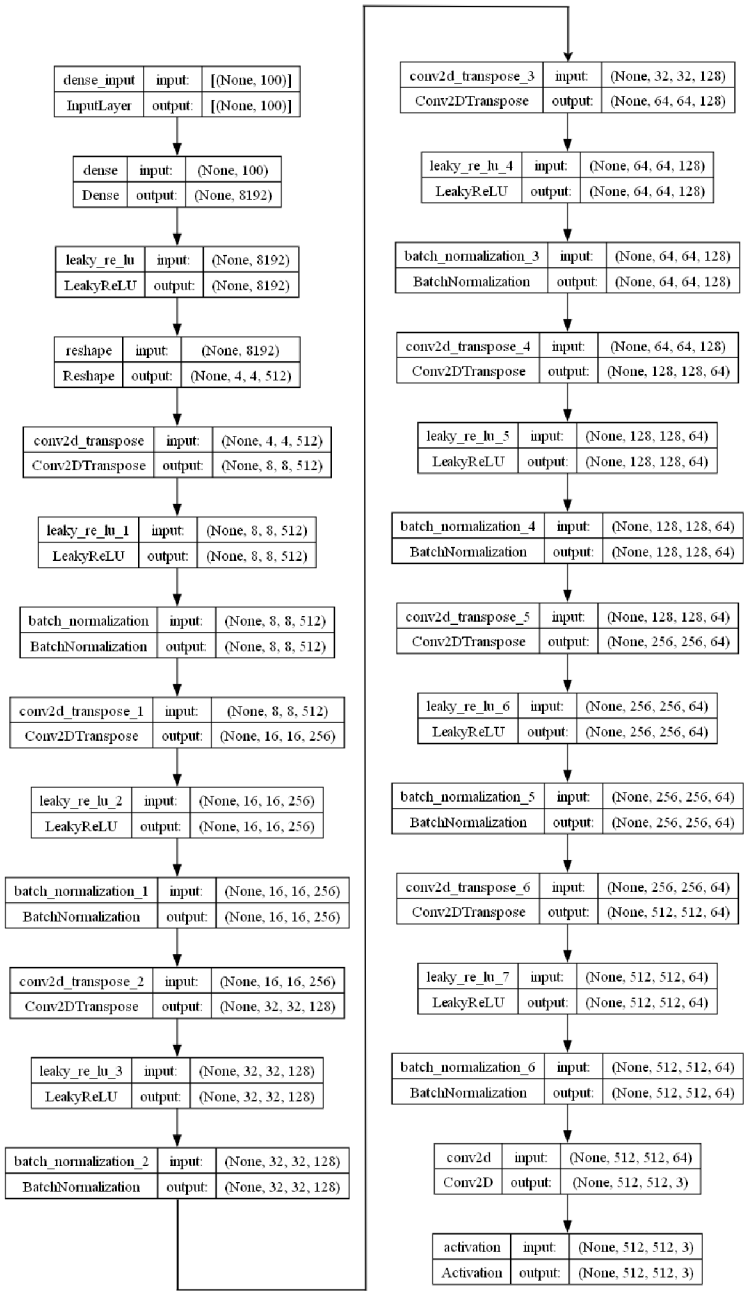
Generátor

Cílem generátoru je transformovat náhodný šum na vstupu a vytvořit natolik realistický obraz, že dokáže oklamat diskriminátor. Funkce `define_generator` funguje podobně jako `define_discriminator`, ale pro architekturu generátoru.

Funkce začíná plně propojenou vrstvou spojenou s aktivační funkcí `Leaky ReLU` s parametrem α=0,2, která je následně pomocí funkce `Reshape` převedena na matici 4×4 s 512 filtry.

Následují dvě vrstvy s funkcemi `Conv2DTranspose` s 512 a 256 filtry, které fungují jako dekonvoluční vrstvy. Tyto vrstvy extrahují příznaky a s krokem `stride` nastaveným na hodnotu dva zdvojnásobují rozlišení matice. První vrstva tedy extrahuje nízkoúrovňové příznaky a zvětší rozlišení z 4×4 na 8×8. Následující dvě vrstvy `Conv2DTranspose` jsou identické s jediným rozdílem, že je použito 128 místo 256 filtrů. Poslední tři dekonvoluční vrstvy mají filtrů 64. Všechny tyto vrstvy jsou opět propojeny s aktivačními funkcemi `Leaky ReLU` a s funkcemi `BatchNormalization`, které stabilizují a zrychlují trénování modelu. V dekonvolučních vrstvách je použito `use_bias=False`, protože vrstvy `BatchNormalization` vykonávají podobnou roli, a tak tuto funkci dekonvoluční vrstvy není nutno použít, aby se předešlo zbytečné redundanci.

Posledním párem vrstev je klasická konvoluční vrstva, která obsahuje tři filtry reprezentující kanály barev RGB, a aktivační funkce `tanh`. Tato funkce je zvolena, protože její výstup reprezentuje normalizované hodnoty pixelů v rozmezí $\langle$-1,1$\rangle$.



Obr. 4.2: Architektura generátoru vytvořená s pomocí funkce `plot_model`

4.2 Trénování modelu

Cílem trénování modelu je vytvořit generátor, který je schopen generovat realistické obrazy podobné těm z trénovací datové sady a tím oklamat diskriminátor. K tomu jsou použity optimalizační algoritmy, které jsou pro oba modely identické, a to algoritmus RMSProp, který vykazuje při trénování WGAN mírně lepší výsledky. Důležitým krokem je definice chybové funkce pro oba modely, kde je využita funkce `Wasserstein loss`. Pro generátor je použita funkce `gen_loss`, zatímco pro diskriminátor je použita funkce `disc_loss`. Navíc je použita penalizace gradientu `gp`, která penalizuje diskriminátor pro zaručení hladšího trénovacího procesu.

Funkce `train` obsahuje dvojitý cyklus `for`, kde první cyklus probíhá pro zadaný počet epoch. V každé epoše tato funkce iteruje přes datovou sadu a zpracovává ji po dávkách. Pro každou dávku je volána funkce `train_step_d` a každou pátou dávku i funkce `train_step_g`, které provedou kroky trénování diskriminátoru a generátoru pro danou dávku.

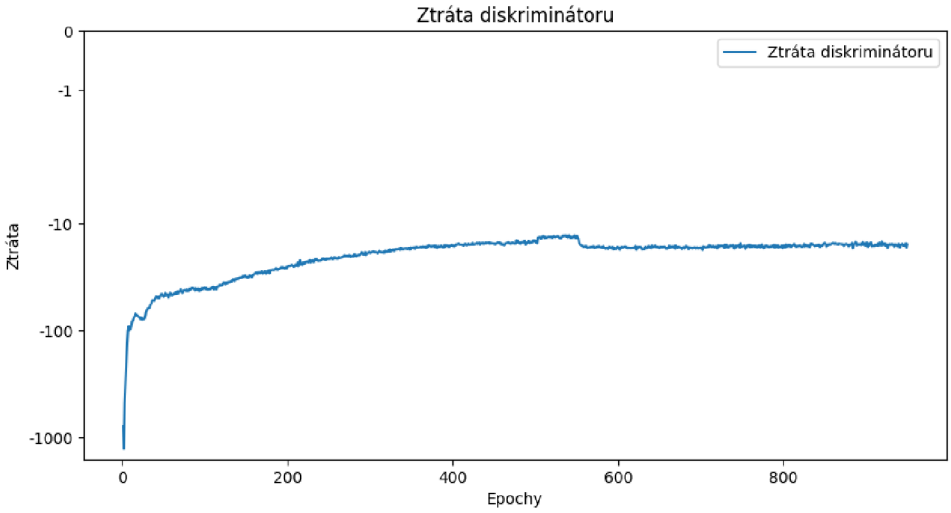
Obě funkce `train_step` generují pro každou dávku obrazů jiný vstupní šum jako vstup do generátoru. Pro obě se používá funkce knihovny Tensorflow `GradientTape`, která uchovává operace provedené v jejím rámci v paměti. Uvnitř těchto funkcí jsou spočítány ztráty generátoru `gen_loss` a diskriminátoru `disc_loss` spolu s penalizací gradientů `gp` pro diskriminátor. Tyto ztráty jsou poté spolu s penalizací a váhou penalizace λ využity k výpočtu gradientů, které jsou následně použity k aktualizaci parametrů generátoru a diskriminátoru.

4.2.1 Průběh trénování

Tato sekce obsahuje grafy zobrazující klíčové metriky trénovacího procesu, čímž jsou ztráta diskriminátoru, penalizace gradientu a ztráta generátoru. Tyto grafy jsou nezbytné pro přehled o stabilitě tréninku a hlubokou analýzu konvergence. Sledování vývoje těchto ztrát za průběhu epoch poskytuje informace o dynamice učení modelů, účinnosti použitých optimalizačních algoritmů a celkové efektivitě trénovacího procesu. Na základě monitorování metrik mohou být identifikovány a adresovány problémy, jako je například nestabilita tréninku nebo neefektivní učení a následně provedeny potřebné úpravy pro dosažení optimálních výsledků. K jedné z takovýchto úprav došlo v epoše 650, kde při původním tréninku diskriminátor překonal generátor a model přestal konvergovat. Z tohoto důvodu bylo trénování vráceno do epochy 550 a míra učení optimalizátoru generátoru zvýšena z 1×10^{-5} na 2×10^{-5}.

Ztráta diskriminátoru

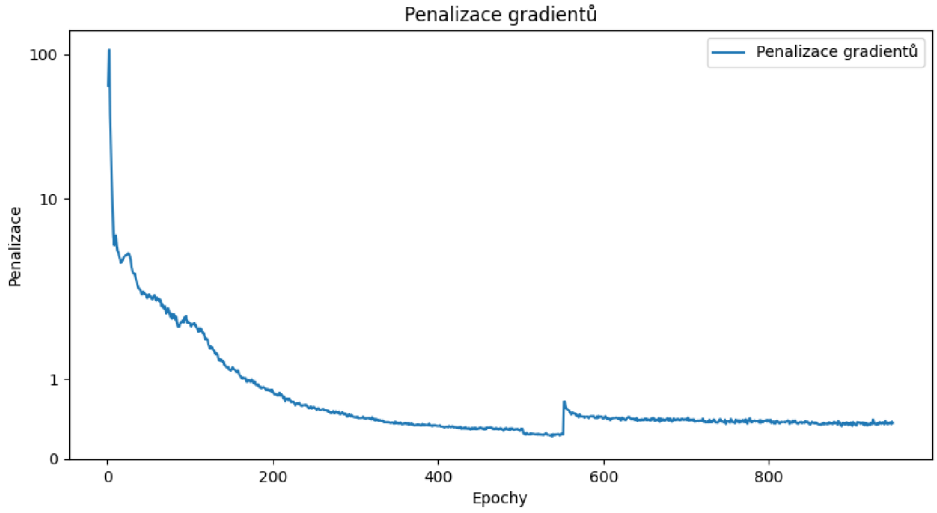
Pro zobrazení grafu je z důvodu velkých počátečních hodnot trénování použita logaritmická osa y, která umožňuje zobrazení smysluplnějšího, a pro hodnoty blížící se nule detailnějšího grafu. Z grafu je vidět, že ztrátová funkce WGAN-GP má hladký průběh pro celou dobu konvergence modelu. Je zde také vidět dopad zvýšení míry učení generátoru v epoše 550, což vedlo k větším ztrátám, ale zároveň i k vyšší kvalitě generovaných snímků. V epoše 950, kdy bylo trénování přerušeno, model stále pomalu konvergoval, ale podle ztrát by došlo k úplnému natrénování modelu okolo epochy 3 000.



Obr. 4.3: Graf ztráty diskriminátoru za uplynulé epochy vytvořený pomocí knihovny `matplotlib`.

Penalizace gradientu

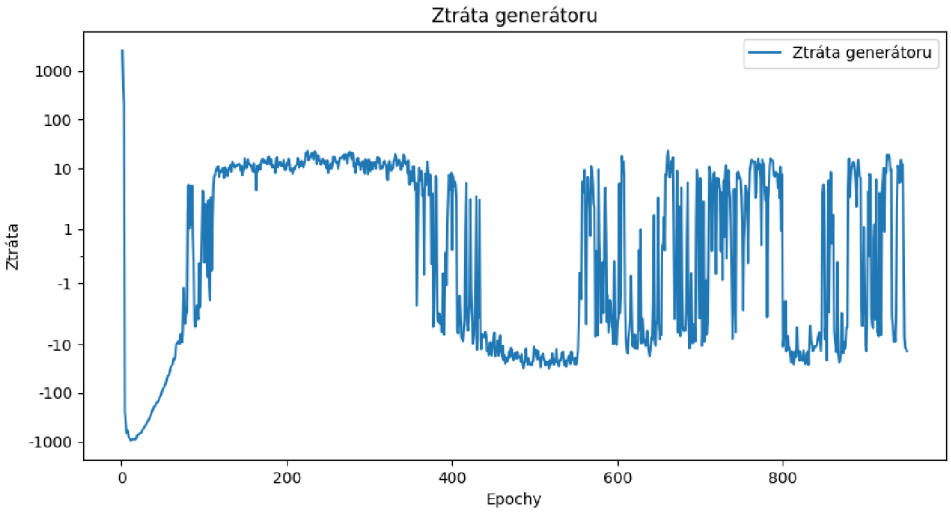
Pro zobrazení grafu je ze stejného důvodu jako u ztráty diskriminátoru použita logaritmická osa y. Průběh vývoje velikosti penalizace gradientu je díky její závislosti na ztrátě diskriminátoru opačný. Je také nutné brát na vědomí, že míra penalizace aplikovaná na gradienty diskriminátoru je vynásobena váhou λ, což znamená, že v tomto případě je penalizace desetkrát větší, než je zobrazeno na následujícím grafu.



Obr. 4.4: Graf penalizace gradientu za uplynulé epochy vytvořený pomocí knihovny `matplotlib`.

Ztráta generátoru

Opět je použito logaritmické zobrazení osy y kvůli vysokým hodnotám při začátku trénování modelu. Na grafu lze vidět, že i přes zmenšující se trend ztrát a zlepšující se kvalitu generovaných obrazů, ztráta generátoru osciluje kolem nuly.



Obr. 4.5: Graf ztráty generátoru za uplynuté epochy vytvořený pomocí knihovny `matplotlib`.

4.3 Výsledky trénování

4.3.1 Generované obrazy

Na snímcích jsou jasně zřetelné obrazy lidských obličejů a vybrané jsou pro danou kvalitu dostatečně realistické. Je ale viditelné, že některé vygenerované obrazy jsou v nižší kvalitě než originály a existují problémy se zobrazením pozadí a uší. Tyto problémy mohou nastávat z nedostatečného trénování modelu a nedostatečného počtu filtrů, který je daný schopností grafické karty. Problém s kvalitou pozadí vyplývá z jeho různorodosti na obrázcích v datové sadě a problém s pomalým trénováním generování uší vzniká s důvodu, že se na většině snímků trénovací datové sady nevyskytují.

4.4 Vyhodnocení výsledků

Dobré systémy sloužící pro hodnocení kvality generovaných obrazů jsou klíčové z několika důvodů.

Jedním aspektem je možnost kvantitativního posouzení věrnosti generovaných obrazů ve srovnání s reálnými daty. Stanovením metriky pro generované obrazy se mohou objektivně měřit aspekty jako percepční podobnost, strukturální koherence a celková vizuální kvalita. Tento kvantitativní přístup slouží jako klíčový nástroj pro pochopení generativních modelů a může sloužit i k jejich zdokonalování.

Dalším bodem je, že systémy hodnocení usnadňují identifikaci anomálií a nedostatků v generovaných datech. Analyzováním rozložení skóre napříč vygenerovanými vzorky se mohou zjišťovat časté nedostatky či nesrovnalosti, jako je rozmazanost, zkreslení nebo nedostatek detailů. Díky této zpětné vazbě je možno upravovat model a dolaďovat hyperparametry.

Pro vyhodnocení kvality vygenerovaných obrazů oproti databází reálných dat se používají různé metody, každá s unikátním způsobem, podle kterého je kvalita zhodnocena.

4.4.1 Metody hodnocení modelu

Při hodnocení generativních modelů je důležité mít k dispozici efektivní metody, které umožní objektivní posouzení jejich výkonnosti a schopnosti reprodukovat reálná data. Metody hodnocení modelu jsou navrženy tak, aby poskytovaly celkový obraz o kvalitě generovaných dat a schopnosti modelu zachytit distribuci trénovacích dat. Příklady takovýchto metod jsou incepční skóre a hodnota FID, která pro získání metriky počítá Fréchetovu vzdálenost mezi distribucemi reálných a generovaných obrazů.

Obr. 4.6: Obrazy vygenerované na základě datové sady CelebA-HQ upravené na velikost 512×512

Incepční skóre

Incepční skóre je základní metrikou používanou k hodnocení kvality a diverzity obrazů generovaných generativními adverzními sítěmi. Tato metrika je zvláště přínosná při posuzování výkonnosti GAN na různých před-trénovaných datových sadách, včetně datové sady ImageNet, která obsahuje obrazy z datové sady ILSVRC2012 s 1 000 kategoriemi. Incepční skóre se vypočítává na základě podmíněného rozdělení tříd generovaných obrazů, kde cílem je nízká entropie v rozdělení významných objektů a vysoká entropie v různorodosti generovaných obrazů.

Vyšší incepční skóre indikuje lepší kvalitu a rozmanitost obrazů, což umožňuje

objektivní srovnání různých modelů GAN a technik trénování. Incepční skóre poskytuje standardizovanou metriku pro hodnocení efektivity generativních modelů při vytváření realistických a různorodých obrazů.

Bylo prokázáno, že incepční skóre se shoduje s lidským hodnocením kvality obrazu, což dokazuje užitečnost incepčního skóre jako spolehlivého měřítka kvality obrazu, které umožňuje hodnotit výkonnost GAN bez nutnosti spoléhat se na subjektivní lidské hodnocení.

Pro tuto generativní síť je incepční skóre ucházejícím indikátorem kvality generovaných obrazů kde problém spočívá v tom, že existuje jenom jedna třída, a to lidské obličeje. Z tohoto důvodu je hodnota incepčního skóre méně senzitivní, a v pozdějších částech trénování klesá vypovídací hodnota tohoto skóre. Incepční skóre je v modelu spočtené pomocí funkce `calculate_is`, která počítá skóre samotné spolu se standardní odchylkou.[12]

Lidské obličeje se v datové sadě ImageNet vyskytují primárně ve čtyřech třídách a to lidské obličeje, člověk, portrét a profil. Dále existují další třídy, které mohou mít částečnou shodu jako třídy oko nebo pusa.

Tab. 4.1: V tabulce jsou znázorněny hodnoty incepčního skóre pro dané epochy trénování modelu při generování 30000 obrazů a použití celé trénovací sady.

Epocha	Incepční skóre
250	8,58
500	10,12
750	10,72
950	11,05

V rámci incepčního skóre se počítá i standardní odchylka, která slouží k určení rozmanitosti vygenerovaných obrazů. Neboť tento model generuje data jenom z jedné třídy tak je standartní odchylka, poměrně malá s hodnotou 0,00097. [12]

Fretchetova incepční vzdálenost

Fréchetova incepční vzdálenost (FID) je široce používaná metrika v oblasti generativních adverzních sítí sloužící k hodnocení kvality generovaných obrazů. FID byla zavedena jako prostředek ke kvantitativnímu hodnocení věrohodnosti a rozmanitosti obrazů vytvořených modely GAN a využívá reprezentaci rysů naučené předtrénovanou sítí `Inceptionv3`, která je trénovaná na modelu ImageNet, k porovnání distribucí skutečných a generovaných obrazů.

Podstatou FID je měření podobnosti mezi reprezentacemi znaků skutečných obrazů a generovaných obrazů pomocí přiřazení vícerozměrných Gaussových rozdělení těmto reprezentacím. Výpočtem Fréchetovy vzdálenosti mezi těmito Gaussovými rozděleními poskytuje skalární hodnotu FID, která vyjadřuje jak moc se rozdělení generovaných obrazů shoduje s rozdělením skutečných obrazů. Nižší skóre FID znamená vyšší úroveň podobnosti mezi těmito rozděleními, což naznačuje, že generované obrazy vykazují charakteristiky více podobné reálným datům.

Jednou z klíčových výhod FID oproti jiným metrikám zabývajícím se hodnocením, jako je například Incepční skóre, je její schopnost brát v úvahu statistiky reálných i generovaných obrazů současně. Tento holistický přístup umožňuje získat podrobnější představu o kvalitě vygenerovaných obrazů a zohlednit nejen jejich vizuální věrnost, ale také jejich distribuční charakteristiku.

S využitím FID je tedy možné porovnávat různé modely GAN, strategie trénování a vhodné hodnoty hyperparametrů. V rámci kódu je FID vypočítané pomocí funkce `calculate_fid` za využití funkce `eval.frechet_inception_distance` z modelu `tensorflow-gan`.[14]

Tab. 4.2: V tabulce jsou znázorněny hodnoty FID pro dané epochy trénování modelu při generování 30000 obrazů a použití celé trénovací sady.

Epocha	Hodnota FID
250	56,65
500	42,25
750	29,53
950	21,45

Jak je vidět tak hodnota skóre FID s postupným trénováním stále, aneb pomaleji, klesá a kvalita generovaných obrazů se tedy zlepšuje. Na epoše 950, kde se trénování zastavilo, je vidět divize mezi kvalitou jednotlivých obrazů. Proto, napříč vysoké kvalitě některých obrazů, je FID poměrně nízké a proto je nutné spočítat kvalitu jednotlivých obrazů.[14]

4.4.2 Metody hodnocení jednotlivých obrazů

Kromě hodnocení celkového výkonu modelu je důležité také analyzovat kvalitu jednotlivých generovaných obrazů. Metody hodnocení jednotlivých obrazů se zaměřují na posouzení vizuální kvality a srovnání s reálnými daty. Mezi tyto metody patří Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) a kosinová podobnost. Tyto metriky poskytují kvantitativní odhad kvality generovaných obrazů

a jsou užitečné při optimalizaci modelů a identifikaci nedostatků v generovaných datech.

Pro ohodnocení modelu byly tyto tři měřítka spočítané pro sto náhodně vygenerovaných obrazů za účelem ohodnocení jejich realističnosti a také za účelem najití obrazu z trénovací sady s nejvyšší shodou.

Tyto výpočty byly provedeny pro sto náhodně vygenerovaných snímků za účelem zjištění konkrétní kvality jednotlivých vzorků.

Kosinová podobnost

Kosinová podobnost slouží jako matematická metrika pro měření podobnosti mezi dvěma soubory a nabízí vhled do jejich vzájemné dynamiky. Na rozdíl od metod, které se zaměřují pouze na jednotlivé hodnoty pixelů, kosinová podobnost zohledňuje směr vektorů a poskytuje tak podrobnější pohled na jejich vzájemnou shodu. Kosinová podobnost se používá v různých oblastech, jako jsou vyhledávače, zpracování přirozeného jazyka a porovnávání obrazů, a usnadňuje efektivní porovnávání a vyhledávání informací z rozsáhlých souborů dat.

Podstata kosinové podobnosti spočívá v kvantifikaci úhlu kosinus mezi dvěma nenulovými vektory ve vícerozměrném prostoru. Matematicky počítá skalární součin vektorů dělený součinem jejich velikostí, což vede k hodnotě spadající do intervalu $\langle -1, 1\rangle$. Hodnota kosinu blížící se 1 znamená větší podobnost, zatímco hodnoty blízké 0 naznačují ortogonalitu. Tento matematický rámec ukazuje jak kosinová podobnost funguje a její účinnost při zachycování vztahů mezi vektory nezávisle na jejich velikostech.

Zásadní výhoda oproti ostatním metodám spočívá v její neměnnosti z hlediska velikosti vstupních dat, díky níž je kosinová podobnost spolehlivá v různých měřítkách. [23][22]

Díky tomu, že je metoda na podobnosti úhlů lépe zachytí rysy obrazů oproti ostatním způsobům, které se zaměřují na podobnost pixelů. Tato skutečnost je pro tuto práci zásadní, protože i když jsou obrazy vygenerované sítí GAN podobné těm z trénovací sady, nejsou stejné, a proto je nutno brát ohled spíše na rysy jednotlivých obrazů, než na ně samotné.

Obr. 4.7: Obraz s nejvyšší hodnotou kosinusové podobnosti, konkrétně 0.922 s obrazem 28285.jpg z trénovací databáze.

PSNR

Poměr špičkového signálu k šumu (PSNR) je objektivní metrika běžně používaná pro hodnocení kvality rekonstruovaných nebo komprimovaných obrazů. Měří kvalitu obrazu porovnáním původního obrazu s komprimovanou nebo rekonstruovanou verzí a poskytuje číselnou hodnotu, která udává úroveň zkreslení přítomného v obraze. V této práci je metoda využita k nalezení nejpodobnějšího obrazu z trénovací sady k danému vygenerovanému vzorku.

PSNR se vypočítá na základě střední kvadratické chyby (MSE) mezi původním a rekonstruovaným obrazem, v tomto případě mezi generovaným obrazem a všemi obrazy trénovací sady. Vzorec pro PSNR zahrnuje logaritmus špičkové hodnoty pixelu na druhou děleno MSE. Vyšší hodnota PSNR znamená menší zkreslení a lepší kvalitu obrazu, protože znamená menší rozdíl mezi generovaným a trénovacím obrazem.

Slabou stránkou PSNR je, že se zaměřuje především na rozdíly v jednotlivých pixelech a nezohledňuje vizuální prvky kvality obrazu. To znamená, že PSNR nemusí vždy odpovídat lidskému vnímání, zejména v případech, kdy jsou pro hodnocení kvality obrazu rozhodující strukturální informace a další percepční faktory jaké nastávají při srovnávání odlišných obrazů.

PSNR se běžně používá v úlohách, jako je komprese videa, restaurování obrazu a v tomto případě pro další pohled při hledání způsobu jakým neuronové sítě generují data. Při posuzování kvality obrazu v kontextech, kde hraje významnou roli lidské vnímání je vhodnější použít metriky jako SSIM, které zohledňují percepční faktory a poskytují přesnější reprezentaci kvality obrazu, a použít metriku PSNR jako pomocnou.[15]

Obr. 4.8: Obraz s nejvyšším skórem PSNR, a to 18.726 s obrazem 22514.jpg z trénovací databáze.

SSIM

Index strukturální podobnosti (SSIM) je široce používaná metrika kvality obrazu, která slouží k zjištění podobnosti mezi dvěma obrazy. Byla vyvinuta Zhou Wangem a jeho kolegy a navržena tak, aby napodobovala vnímání kvality obrazu lidským zrakovým systémem. SSIM zohledňuje tři klíčové složky zkreslení obrazu, a to ztrátu korelace, zkreslení jasu a zkreslení kontrastu. Vyhodnocením těchto faktorů poskytuje SSIM relevantní míru podobnosti mezi porovnávanými obrazy.

Vzorec pro SSIM zahrnuje výpočet funkce porovnání jasu, funkce porovnání kontrastu a funkce porovnání struktury. Zohledněním těchto aspektů dokáže SSIM

účinně kvantifikovat percepční rozdíly mezi obrazy, což z něj činí vhodný nástroj pro hodnocení kvality obrazu.

Použití tedy umožňuje komplexnější přístup ve srovnání s tradičními metrikami, jako je například poměr špičkového signálu k šumu (PSNR). Bere totiž v úvahu nejen rozdíly jednotlivých pixelů, ale také strukturální informace a percepční aspekty, které jsou důležité pro lidské vnímání. Díky tomu je SSIM vhodné při porovnávání podobnosti a hledání nejbližší shody generovaného obrazu s trénovací sadou.[15]

Obr. 4.9: obraz s nejvyšší hodnotou metriky SSIM s hodnotou 0.763 s obrazem 2573.jpg z trénovací databáze.

Vyhodnocení jednotlivých obrazů

Přesné hodnocení obrazů generovaných neuronovými sítěmi je obtížné, protože jednotlivé techniky nedokážou zachytit všechny nuance obrazů tak jako lidský zrak. Z tohoto důvodu je zde pro hodnocení generovaných snímků použito více klasifikačních metod.

Metoda PSNR je z použitých metod nejméně vypovídající a to hlavně z důvodu, že se zaměřuje jenom na hodnoty pixelů což při porovnávání odlišných obrazů není absolutně přesné. I přesto, že u některých z použitých vzorků s vysokou hodnotou

PSNR byla podobnost dána na barvě pozadí, tak metoda PSNR dokázala najít podobnosti. Při hodnocení kvality snímků lze říct, že i přes výjimky je většina snímků s hodnotou PSNR větší než 15 realistická.

Metody jako kosinová podobnost a SSIM poskytují v tomto případě vizuálně výstižnější informace. Kosinová podobnost se zaměřuje na podobnost obrazů podle jejich rysů, což je vhodné pro analýzu generovaných obrazů, zatímco SSIM bere v potaz strukturální informace a percepční aspekty, které jsou důležité pro lidské vnímání. Při vizuálním porovnávání generovaných vzorků s obrazy s nejvyšší shodou lze vidět, že kosinová podobnost najde obraz z nejbližším tvarem obličeje, zatímco SSIM najde obraz s větším důrazem na podobnost barevné a jasové složky. Problémem při hodnocení kosinusovou podobností je, že kvůli zaměření na rysy může dát vysoké skóre částečně rozmazaným snímkům. Generované obrazy, které působí realisticky mají SSIM nad 0,65 a a kosinovou podobnost nad 0,88.

Tab. 4.3: Tabulka rozložení skóre jednotlivých metod provedených na sto vygenerovaných vzorcích

Metoda	Rozložení hodnot
PSNR	14.913 ± 3.813
SSIM	0.645 ± 0.118
Kosinová podobnost	0.813 ± 0.109

Kvůli malé citlivosti kosinusové podobnosti na rozmazání a příliš velké závislosti SSIM na individuálních pixelech je pro přesnější kvalitativní metriku a nalezení nejbližší shody z trénovací sady požita kombinace těchto dvou metod. Tento kompromis dokáže najít obraz, který z nalezených obrazů použitými metrikami vypadá pro lidské smysly nejpodobněji použitému vygenerovanému snímku.

Obr. 4.10: obraz s nejvyšší průměrnou hodnotou metrik kosinové podobnosti a SSIM o podobnosti 0.801 s obrazem 17822.jpg z trénovací databáze.

Závěr

Cílem této diplomové práce bylo vybrat si datovou sadu a algoritmus, který bude schopen generovat realistické obrázky z tématického okruhu vybrané sady. Tato práce navrhuje metodu pro trénování architekturu WGAN-GP, která je speciálně přizpůsobená pro stabilní generování realistických obrázků. Implementace této metody je provedena v prostředí Python s využitím knihoven TensorFlow a Keras.

Výsledkem trénování na datové sadě Celeba-HQ je model schopný generovat realistické obrázky obyčejů v rozlišení 512×512 pixelů. Největším problém u generovaných obrázku je rozdíl kvality mezi jednotlivými generovanými snímky, problém rozlišení pozadí, uší a dalších prvků jako čepice. Toto je zapříčeného různorodostí a malým zastoupením těchto prvků v trénovací sadě, což vede k nedostatku informací pro jeho rekonstrukci.

Dalším bodem je také vyhodnocení celkového modelu a jednotlivých vygenerovaných obrázků, a to jejich porovnáním s trénovací sadou.

Možnost pro rozvoj existuje v několika ohledech, a to hlavně dodatečné trénování sítě a doladění jednotlivých hyperparametrů modelu k dosažení lepších výsledků. Další možnost je použit pokročilejší variantu GAN jako je například progresivně rostoucí generativní adverzní síť, která dokáže na úkor komplexity modelu vytvářet snímky ve vyšším rozlišení než vstupní obrázky.

Literatura

[1] AGGARVAL, C. C.: *Neural Networks and Deep Learning* Springer, 2018. s. 1-21 ISBN: 9783319944623

[2] Amazon web services *What is a Neural Network?* Online. Dostupné z: `https://aws.amazon.com/what-is/neural-network/`. [cit. 2023-11-15].

[3] ARJOVSKY, M. a CHINTALA, S. a BOTTOU, L.: *Wasserstein generative adversarial networks* Online Dostupné z: `https://arxiv.org/pdf/1701.07875`. [cit. 2024-04-08]

[4] BOESCH, G.: *Guide to Generative Adversarial Networks (GANs) in 2024* Online viso.ai 2024 Dostupné z: `https://viso.ai/deep-learning/generative-adversarial-networks-gan/`. [cit. 2023-11-26]

[5] BROWNLEE, J.: *Code Adam Optimization Algorithm From Scratch* Online machine learning mastery 12.10.2021 Dostupné z: `https://machinelearningmastery.com/adam-optimization-from-scratch/`. [cit. 2023-11-22]

[6] BROWNLEE, J.: *A Gentle Introduction to Generative Adversarial Networks (GANs)* Online machine learning mastery 19.7.2019 Dostupné z: `https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/`. [cit. 2023-11-27]

[7] BROWNLEE, J.: *Tips for Training Stable Generative Adversarial Networks* Online machine learning mastery 12.9.2019 Dostupné z: `https://machinelearningmastery.com/how-to-train-stable-generative-adversarial-networks/`. [cit. 2023-11-27]

[8] COMBETTES, S. *A basic intro to GANs (Generative Adversarial Networks)* Online. towardsdatascience 26.10.2020 Dostupné z: `https://towardsdatascience.com/a-basic-intro-to-gans-generative-adversarial-networks-c62acbcefff3`. [cit. 2023-11-11].

[9] COMBETTES, S.: *A basic intro to GANs (Generative Adversarial Networks)* Online towardsdatascience 26.10.2020 Dostupné z: `https://towardsdatascience.com/a-basic-intro-to-gans-generative-adversarial-networks-c62acbcefff3`. [cit. 2023-11-25]

[10] GOODFELLOW, I. a POUGET-ABADIE, J. a MIRZA, M. a XU, B. a WARDE-FARLEY, D. a OZAIR, S. a COURVILLE, A. a BENGIO, Y.: *Generative Adversarial Nets* Online 2014 Dostupné z: https://papers.nips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf. [cit. 2023-11-27]

[11] GOODFELLOW, I a BENGIO, Y a COURVILLE, A.: *Deep learning* MIT press, 2016. s. 275-300 ISBN: 9780262035613

[12] GOODFELLOW, I. a SALIMANS, T. a ZAREMBA, W. a CHEUNG, V. a RADFORD, A. a CHEN, X.: *Improved Techniques for Training GANs* Online Dostupné z: https://arxiv.org/pdf/1606.03498. [cit. 2024-05-10]

[13] GULRAJANI, I. a AHMED, F. a ARJOVSKY, M. a DUMOULIN, V. a COURVILLE, A. C.: *Improved Training of Wasserstein GANs* Online Dostupné z: https://arxiv.org/pdf/1704.00028. [cit. 2024-04-08]

[14] HEUSEL, M. a RAMSAUER, H. a UNTERTHINER, T. a NESSLER, B. a HOCHREITER, S.: *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium* Online Dostupné z: https://arxiv.org/pdf/1706.08500. [cit. 2024-05-10]

[15] HORÉ, A. a ZIOU, D.: *Image Quality Metrics: PSNR vs. SSIM* Online Dostupné z: https://ieeexplore.ieee.org/document/5596999. [cit. 2024-05-02]

[16] IBM *What is deep learning?* Online. Dostupné z: https://www.ibm.com/topics/deep-learning. [cit. 2023-11-12].

[17] IBM *What are convolutional neural networks?* Online. Dostupné z: https://www.ibm.com/topics/convolutional-neural-networks. [cit. 2023-11-16].

[18] IBM *What are neural networks?* Online. Dostupné z: https://www.ibm.com/topics/neural-networks. [cit. 2023-11-14].

[19] JASPREET *A Concise History of Neural Networks* Online. Dostupné z: https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec. [cit. 2024-02-15].

[20] LECUN, Y. a BENGIO, Y. a HINTON, G. *Deep learning* Online. Nature 27.05.2015 Dostupné z: https://www.nature.com/articles/nature14539. [cit. 2023-11-11].

[21] LEE, C.-H. a LIU, Z. a WU, L. a LUO, P.: *CelebAMask-HQ* Online Dostupné z:
https://github.com/switchablenorms/CelebAMask-HQ. [cit. 2024-05-03]

[22] MIESLE, P.: *What is Cosine Similarity?* Online Dostupné z:
https://medium.com/building-the-open-data-stack/what-is-cosine-similarity-ebce79586018. [cit. 2024-04-15]

[23] OKONJI, O.: *Cosine similarity — measuring similarity between multiple images* Online Dostupné z:
https://onyekaokonji.medium.com/cosine-similarity-measuring-similarity-between-multiple-images-f289aaf40c2b. [cit. 2024-04-15]

[24] POTRIMBA, P. *What is an Activation Function? A Complete Guide.* Online. Roboflow 12.1.2023 Dostupné z: https://blog.roboflow.com/activation-function-computer-vision/.
[cit. 2023-11-19].

[25] POTRIMBA, P. *What is a Convolutional Neural Network?* Online. Roboflow 10.2.2023 Dostupné z: https://blog.roboflow.com/what-is-a-convolutional-neural-network/.
[cit. 2023-11-19].

[26] PRINCE, Simon JD. *Understanding Deep Learning.* MIT press, 2023. s. 1-22, 268-297 ISBN: 9780262048644

[27] RAJAGOPAL, I. *Batch Normalization — Speed up Neural Network Training* Online. Medium 29.5.2018 Dostupné z:
https://medium.com/@ilango100/batch-normalization-speed-up-neural-network-training-245e39a62f85. [cit. 2023-11-20].

[28] RAJAGOPAL, I. *BatchNorm: Fine-Tune your Booster* Online. Medium 8.6.2018 Dostupné z:
https://medium.com/@ilango100/batchnorm-fine-tune-your-booster-bef9f9493e22. [cit. 2023-11-20].

[29] SAHA, S.: *A Comprehensive Guide to Convolutional Neural Networks* Online towardsdatascience 15.12.2020 Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
[cit. 2023-11-18]

[30] Tensorflow: *What is TensorFlow? The machine learning library explained* Online 8.6.2023 Dostupné z: `https://www.tensorflow.org/guide/keras`. [cit. 2023-12-01]

[31] WUTTKE, L.: *Artificial neural networks* Online. datasolut Dostupné z: `https://datasolut.com/neuronale-netzwerke-einfuehrung/`. [cit. 2023-11-14].

[32] YATHIS, V.: *Loss Functions and Their Use In Neural Networks* Online towardsdatascience 4.8.2022 Dostupné z: `https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9`. [cit. 2023-11-20]

Seznam symbolů a zkratek

UI	Umělá inteligence
EM	Earth mover
MLP	Vícevrstvý perceptron - Multilayer perceptron
NN	Neuronová síť - Neural Network
CNN	Konvoluční neuronová síť - Convolutional Neural Network
RGB	červená,zelená,modrá - red,green,blue
FID	Fretchetova incepční vzdálenost - Frechet Inception Distance
GAN	Generativní adverzní sítě – Generative adversarial networks
Adam	Adaptive Movement Estimation
RMSProp	Root Mean Square Propagation
DCGAN	Hluboké konvoluční generativní adverzní sítě – Deep convolutional generative adversarial network
API	Aplikační programovací rozhraní –Application programming interface
PSNR	Špičkový poměr signálu k šumu –Peak signal-to-noise ratio
MSE	Střední kvadratická chyba –Mean square error
SSIM	Míra indexu strukturální podobnosti –Structural Similarity Index Measure
GB	Gigabajt
PGGAN	Progresivně rostoucí generativní adverzní sítě – Progresive growing generative adversarial network