Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Využití metod hlubokého učení k vytváření realistických medicínských obrazových dat

Petr Vorlíček

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Petr Vorlíček

Informatika

Název práce

Využití metod hlubokého učení k vytváření realistických medicínských obrazových dat

Název anglicky

Usage of deep learning methods to generate realistic medical image data

---

Cíle práce

Zobrazovací metody jsou jednou z klíčových diagnostických metod v medicíně a jejich význam roste díky stále větší dostupnosti. V reakci na tento trend nabízí některé společnosti systémy pro podporu rozhodování založený na metodách strojového učení, který lékaře upozorní na potenciálně problematický snímek, či dokonce na možnou lokalizaci nálezu na RTG snímku hrudníku, což mimo jiné umožňuje standardizaci diagnostiky napříč různými odděleními.

Cílem této práce je využít generativní modely hlubokých neuronových sítí, jako jsou Variational Autoencoder, Generative Adversarial Networks nebo jiné, k vygenerování realistických rentgenových snímků různých distribucí, které by mohly být později použity pro zlepšení výsledného produktu dané společnosti. Výsledkem práce bude generativní model nebo více modelů, které budou vyhodnoceny na základě řady experimentů zahrnujících například kosinovou podobnost tréninkových a vygenerovaných snímků.

Metodika

V práci budou popsány současné metody použití hlubokých neuronových sítí pro zpracování a generování obrazové informace. Bude popsán opensource software určený k jejich relizaci, především frameworky TensorFlow, Keras a PyTorch. Dále budou popsány generativní modely hlubokých neuronových sítí, jako jsou Variational Autoencoder, Generative Adversarial Networks nebo jiné.

Pro úlohu generování RTG snímků hrudníku bude demonstrováno použití jedné nebo více popsaných metod. Model či modely vzniklé touto demonstrací budou vyhodnoceny na základě řady experimentů zahrnujících například kosinovou podobnost tréninkových a vygenerovaných snímků. Pro realizaci budou použity frameworky TensorFlow a PyTorch.

Doporučený rozsah práce

60

Klíčová slova

zpracování obrazové informace, hluboké učení, Generative Adversarial Networks, RTG snímky hrudníku

---

Doporučené zdroje informací

BISHOP, Christopher M. *Pattern recognition and machine learning.* [New York]: Springer, 2006. ISBN 0-387-31073-8.

GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-4920-3264-9.

GOODFELLOW, Ian J., Jean POUGET-ABADIE, Mehdi MIRZA, Bing XU, David WARDE-FARLEY, Sherjil OZAIR, Aaron COURVILLE a Yoshua BENGIO. Generative Adversarial Networks [online]. Montréal, 2014. Dostupné z: arXiv:1406.2661. Universite de Montréal.

CHOLLET, François. *Deep learning with Python.* Shelter Island: Manning, 2021. ISBN 978-1-61729-686-4.

KINGMA, Diederik P. a Max WELLING. Auto-Encoding Variational Bayes [online]. 2013. Dostupné z: arXiv:1312.6114. Universiteit van Amsterdam.

Předběžný termín obhajoby

2023/24 ZS – PEF

Vedoucí práce

doc. Ing. Arnošt Veselý, CSc.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 30. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 07. 01. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Využití metod hlubokého učení k vytváření realistických medicínských obrazových dat " jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2024 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce doc. Ing. Arnoštu Veselému, CSc., za jeho odborné vedení a rady, které mi byly během tvorby této práce poskytnuty.

Využití metod hlubokého učení k vytváření realistických medicínských obrazových dat

Abstrakt

Tato práce je zaměřena na možný způsob řešení rozšíření datových souborů používaných ve vývoji systémů na podporu rozhodování v lékařské diagnostice vývojem generativního modelu založeném na hlubokém učení pro vytváření umělých rentgenových snímků hrudníku. Využití generativních modelů, jako jsou generativní soupeřící sítě, variační autoenkodéry a difuzní modely, představuje nový způsob vytváření realistických medicínských obrazových dat, které mají potenciál zlepšit přesnost a vysvětlitelnost systémů na podporu rozhodování v lékařské diagnostice.

Cílem práce je vytvoření generativního modelu vytvářejícího realistické snímky hrudníku. Model je vytvořen za pomoci moderní knihovny strojového učení PyTorch. V teoretické části práce jsou uvedeny možná využití a výzvy umělé inteligence v lékařské praxi. Dále jsou přiblíženy knihovny pro tvorbu neuronových sítí a výše zmíněné modely používané pro generování obrazové informace. V praktické části je výsledný model podroben zkoumání pomocí řady metrik, jako je například kosinová podobnost.

Klíčová slova: hluboké učení, zpracování obrazové informace, generativní umělá inteligence, computer aided diagnosis, generativní soupeřící sítě, RTG snímky hrudníku, TensorFlow, PyTorch, rozšíření datového souboru

Usage of deep learning methods to generate realistic medical image data

Abstract

This work focuses on a possible way to address the extension of datasets used in the development of decision support systems in medical diagnosis by developing a deep learning based generative model for generating artificial chest X-ray images. The use of generative models, such as generative adversarial networks, variational autoencoders and diffusion models, represents a new way of generating realistic medical image data that has the potential to improve the accuracy and explainability of decision support systems in medical diagnosis.

The goal of this work is to create a generative model that produces realistic chest images. The model is created using the machine learning library PyTorch. In the theoretical part of the thesis, possible applications, and challenges of artificial intelligence in medical practice are presented. Furthermore, libraries for creating neural networks and the models used for generating image information are presented. In the practical part, the resulting model is examined using several metrics such as cosine similarity.

Keywords: Deep Learning, Image Information Processing, Generative Artificial Intelligence, Computer Aided Diagnosis, Generative Adversarial Nets, Chest X-Ray, TensorFlow, PyTorch, Data Augmentation

Obsah

1 Úvod

Využití zobrazovacích metod v medicíně je v dnešní době nedílnou součástí lékařské diagnostiky a jejich využití díky vyšší dostupnosti roste. Dalším rychle rostoucím odvětvím je strojové učení, které se s medicínskou diagnostikou prolíná v praxi prostřednictvím systémů Computer Aided Diagnosis.

Tyto systémy na podporu rozhodování a jiné způsoby využití AI v medicíně však v realitě naráží na řadu výzev. Jednou z nejvýraznějších výzev je přístup ke kvalitním a robustním datovým souborům. Tyto datové soubory obsahují citlivá data a jejich anotování musí být prováděno lékařskými specialisty. To zvyšuje jejich pořizovací cenu a snižuje jejich dostupnost.

Tato práce se zabývá jedním z možných řešení tohoto problému. Cílem této práce je vytvoření generativního modelu schopného vytvářet umělé rentgenové snímky hrudníku. Takové modely mohou pomoci vytvářet robustní a dostupné datové sady, které následně mohou být použity pro vytvoření přesných, vysvětlitelných a spravedlivých systémů na podporu rozhodování, které sníží cenu a zvýší dostupnost a kvalitu lékařské péče.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je vytvoření generativního modelu využívajícího hluboké neuronové sítě, schopného vytváření rentgenových snímků hrudníku. Tento model je vytvářen za účelem možnosti rozšíření a diverzifikování sady rentgenových snímků hrudníku, které potenciálně zlepší robustnost a efektivitu systémů pro podporu rozhodování založených na strojovém učení v medicínské diagnostice.

Podobné modely mají potenciál rozšířit existující datasety pro výzkum a vývoj, čímž se zvýší schopnost modelů strojového učení generalizovat a zlepší se jejich diagnostická přesnost a vysvětlitelnost těchto modelů.

2.2 Metodika

V práci jsou popsány současné metody použití hlubokých neuronových sítí pro generování obrazové informace. Je popsán opensource software určený k jejich relizaci, především frameworky TensorFlow, Keras a PyTorch. Dále jsou popsány generativní modely hlubokých neuronových sítí, jako jsou Variational Autoencoder, Generative Adversarial Networks nebo jiné.

Pro úlohu generování RTG snímků hrudníku je demonstrováno použití modelu Generative Adversial Network. Modely vzniklé touto demonstrací jsou vyhodnoceny na základě běžných metrik Inception Score a Fréchet Inception Distance. Vybrané generované snímky jsou dále zkoumány pomocí kosinové podobnosti tréninkovým snímkům a pomocí lineární interpolace mezi těmito snímky. Pro realizaci je použit framework PyTorch.

2.2.1 NIH Chest X rays dataset

Pro tuto práci je jako trénovací soubor zvolen datový soubor NIH Chest X-ray. Soubor dat NIH Chest X-ray je jeden z největších veřejně dostupných souborů lékařských snímků. Skládá se z více než 100000 anonymizovaných rentgenových snímků hrudníku pocházejících od více než 30000 pacientů. Tato datová sada, kterou spravují Národní ústavy

zdraví (National Institutes of Health) a je poskytnuta NIH Clinical center1, má zásadní význam pro rozvoj výzkumu v oblasti lékařského zobrazování, zejména při vývoji a hodnocení systémů počítačové podpory diagnostiky. Každý snímek v souboru dat je opatřen jedním nebo více štítky ze sady osmi standardních hrudních patologických kategorií, což z něj činí cenný zdroj pro trénování a testování modelů strojového učení, včetně generativních soupeřících sítí (GAN). Snímky jsou ve formátu PNG s rozlišením 1024 x 1024 pixelů. Anotace byly získány kombinací zpracování přirozeného jazyka (natural language processing) souvisejících radiologických zpráv a odborného posouzení. Samotný objem a rozmanitost datového souboru představují jedinečnou příležitost pro přístupy hlubokého učení, které mohou potenciálně zlepšit diagnostickou přesnost, automatizovat radiologická hodnocení a usnadnit vývoj nových zobrazovacích technologií [1].

2.2.2 Fréchet Inception Distance

Cílem generativních modelů je tvorba dat, která odpovídají pravděpodobnostnímu rozložení trénovacích dat. Rozdíl mezi těmito rozloženími může sloužit k hodnocení kvality generativního modelu. Fréchet inception distance (FID) je metrika používaná k hodnocení kvality snímků generovaných pomocí GAN. Byla zavedena Heusel et. al. a poprvé diskutována v článku „GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium“. FID měří podobnost mezi distribucí generovaných a skutečných snímků pomocí fréchetovy vzdálenosti latentních prostorů předem natrénovaného modelu. Výhodou FID je, že měří nejen vizuální podobnost dvou souborů dat, ale také hodnotí zachycení rozmanitosti souboru dat [2]. Nižší FID značí nižší rozdíly mezi jednotlivými rozděleními.

Význam FID při posuzování kvality generovaných snímků, zejména v oblasti lékařského zobrazování, jako jsou rentgenové snímky hrudníku, spočívá v jeho schopnosti zachytit jak věrnost, tak rozmanitost generovaných snímků. Na rozdíl od jiných metrik, které mohou hodnotit pouze jeden aspekt kvality obrazu nebo se spoléhat na přímé porovnání jednotlivých pixelů, poskytuje FID komplexnější měřítko tím, že hodnotí, jak dobře generované obrazy napodobují rozložení reálného světa v abstraktním prostoru rysů. To je důležité zejména v lékařském zobrazování, kde je přesné zobrazení anatomických struktur a

[1] Dataset je dostupný z: https://nihcc.app.box.com/v/ChestXray-NIHCC

patologií klíčové pro klinickou využitelnost snímků. Pomocí FID může být kvantitativně posouzena výkonnost modelů GAN a může být zajištěno, aby generované obrazy byly nejen vizuálně přesvědčivé, ale také dostatečně rozmanité, aby reprezentovaly variabilitu, která se vyskytuje v reálných souborech lékařských dat. V této práci proto FID slouží jako metrika pro ověření účinnosti GAN při generování realistických rentgenových snímků hrudníku, které by mohly potenciálně pomoci v lékařském výzkumu, vzdělávání a diagnostických procesech, protože poskytují robustní kvantitativní měřítko kvality a rozmanitosti snímků.

Pro PyTorch existuje implementace FID [3]. Pro hodnocení je potřeba nejméně 10000 snímků [4].

2.2.3 Kosinová podobnost

Metrika kosinové podobnosti je metoda, která měří podobnost mezi dvěma vektory. Měří se kosinus úhlu mezi těmito dvěma vektory, čímž je určeno, zda tyto vektory ukazují přibližně ve stejném směru. Předpokladem této metody je stejná délka vektorů. Platí

$$S_{cos}(a, b) = \frac{a \cdot b}{\|a\|\|b\|} \tag{1}$$

kde $\|a\|$ je norma vektoru $a = (a_1, a_2, a_3, \dots, a_n)$ dle vztahu:

$$\|a\| = \sqrt{\sum_{i=1}^{n} a_i^2}. \tag{2}$$

Norma vektoru je tedy jednotková a míra podobnosti je rovna kosinu úhlu mezi těmito vektory. Obor hodnot fukce $S_{cos}(a, b)$ nabývá $\langle -1; 1 \rangle$, kde 1 značí totožnost vektorů [5].

Tato metoda je tedy vhodná k vyhledání podobných obrázků v různých datových souborech, což poskytuje jiný úhel pohledu na hodnocení generativních soupeřících sítí pro vytváření realistických snímků. Tyto dvojice lze vizuálně prezentovat a porovnat jejich vizuální kvalitu a rozmanitost generovaných dat. Tím je zajištěno kvalitativní hodnocení generovaných snímků, které doplňuje kvantitativní hodnocení poskytované metrikou FID.

2.2.4 Inception Score

Jelikož modely GAN neobsahují objektivní funkci, je obvykle složité porovnávat různé modely. Jednou z možností, kterou popsal Salimans et. al. [6] je možnost využití lidí jako anotátorů reálných a generovaných snímků. Hodnocení snímků lidmi je intuitivní a v případě použití výstupů generativních modelů pro použití lidmi je to i konečný cíl trénování, jedná

se však o metodu nepraktickou. Tento proces je náročný časově i pracovně, a hodnocení anotátorů je subjektivní. Proto Salimans et. al. hledali objektivnější postup, který by odpovídal potřebě účinnosti a škálovatelnosti hodnocení generativních modelů.

Pro jejich hodnocení využili Inception model [7], který je původně určen pro klasifikaci obrazu. IS měří rozmanitost generovaných obrazů. Vyšší hodnota IS naznačuje, že generované obrazy jsou různorodé a odlišné, což jsou vlastnosti, které jsou při generování realistických snímků žádoucí. Salimans et. al. podotýkají, že toto skóre samostatně lze použít pro trénink jen s obtížemi, přesto však dobře koreluje s lidským úsudkem. Pro tuto metriku je doporučováno použití alespoň 50000 vzorků.

3 Teoretická východiska

3.1 Umělá inteligence a strojové učení

3.1.1 Umělá inteligence

Alan Turing ve svém článku Výpočetní technika a inteligence [8] rozvíjí otázku, zda mají digitální počítače schopnost inteligentního myšlení stejně jako lidé. Přichází zde se slavným konceptem imitační hry. Umělou inteligenci v širším kontextu lze tedy jako chování stroje, které imituje chování člověka; v užším kontextu je to vědní disciplína, studující právě tento fenomén [9].

Počátky AI jako vědního oboru se datují do roku 1956 [10]. Dnes s příchodem systémů jako je ChatGPT [11] či Dall-e [12] je umělá inteligence jedním z největších celospolečenských témat, podněcující veřejnou debatu jak směrem k možnostem a příležitostem využití AI, tak možných rizik a výzev, jako jsou etika či možnost tvorby dezinformací [13,14].

3.1.2 Strojové učení

Aurélien Géron definuje strojové učení jako vědu, zabývající se programováním instrukcí tak, že se počítače mohou učit z dat. Strojové učení je podoborem umělé inteligence. Strojové učení nalézá nejlepší uplatnění v programování úloh, které jsou obtížné na vyjádření exaktními pravidly, jsou moc složité na konvenční přístup či nemají žádný známý algoritmus. Příkladem může být spamový filtr, kde hledání kombinace slov a vyjádření těchto kombinací pravidly je pro lidského programátora časově náročný úkol, nebo strojové rozpoznání řeči, kde neexistuje žádný známý algoritmus [15].

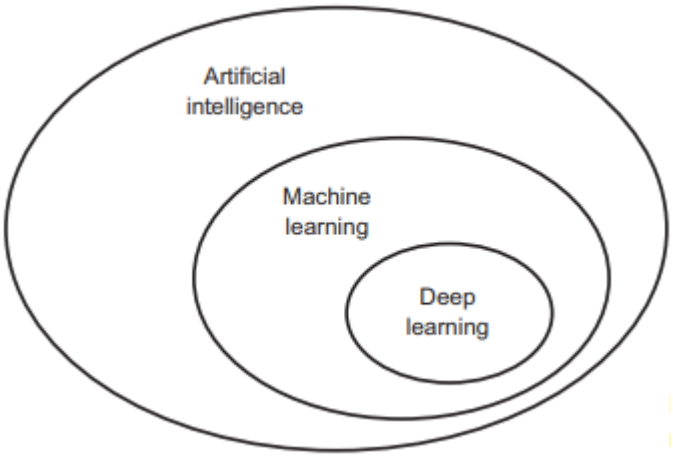
Modely strojového učení mohou také pomáhat lidem vysvětlit úlohy, na které byly konstruovány. Ze zmíněného spamového filtru lze například zpětně získat seznam zakázaných kombinací slov. Tento obrácený proces, kdy na velká data aplikujeme metody strojového učení s cílem vysvětlení či hledání vztahů, se obvykle nazývá data mining [15].

3.1.3 Deep learning

Dle François Chollet je deep learning podkategorií strojového učení. Toto je znázorněné na obrázku 1. Cílem algoritmů strojového učení je vytvoření modelu, který reprezentuje systém původního datového souboru. Metody deep learningu dosahují této reprezentace pomocí

vrstev reprezentací s rostoucí smysluplností. Hloubka – depth – modelu je tedy počtem vrstev modelu. Jednotlivé vrstvy modelu jsou téměř vždy neuronovými sítěmi. Operace s těmito modely lze vyjádřit jako tenzorové operace. Moderní modely hlubokého učení mohou mít desítky či stovky vrstev, na rozdíl od tradičních metod strojového učení, které se někdy nazývá jako shallow learning [16].

Obrázek 1 *Zařazení deep learningu do oboru umělé inteligence*



Zdroj: [16]

Přestože deep learning byl zkoumán již v padesátých letech, jeho rozšíření přišlo až v roce 2012. Chollet [16] vysvětluje nástup deep learningu jako syntézu tří faktorů.

1. Vhodný harware s dostatečným výkonem: v prvních letech dvacátého století bylo kvůli zájmu o fotorealistické 3D hry silně investováno do vývoje GPU – rychlých a velmi paralelizovaných procesorů. Tento vývoj vyústil vytvořením CUDA API společností Nvidia, které bylo použito pro implementaci neuronových sítí.

2. Dostupnost datových souborů: S nástupem internetu a postupným zlevňováním hardwarových úložišť bylo vytvořeno ideální prostředí nejen pro sběr, ale i sdílení různých datových souborů, jako ImageNet.

3. Nové algoritmy: Do začátku desátých let 20. století neexistovala spolehlivá metoda trénování hlubokých neuronových sítí. Bylo tedy možné trénovat jen mělké neuronové sítě, které se nemohly poměřovat s ostatními metodami strojového učení. Průlom přišel s vylepšeními jako optimalizační schéma Adaptive movement estimation, lepšími aktivačními funkcemi neuronů nebo dávkové normalizace.

3.2 Využití AI v medicíně

3.2.1 Konkrétní využití AI v medicíně

Metody umělé inteligence mohou nachází široké uplatnění v různých odvětvích medicíny. Odvětvím, které je díky silné integraci výpočetní techniky do procesu diagnostiky zvláště otevřené implementaci AI systémů, je radiodiagnostika [17]. Radiodiagnostika je součástí téměř všech diagnostických postupů a poptávka po radiologické diagnostice roste každým rokem. Trénink nových radiologů je však nákladný a časově náročný. Tím se zvedá profesní tlak na současné radiology, což vede k vyšším počtům chybných diagnóz. Proto má velký praktický význam najít jiné způsoby, například umělou inteligenci, jak naléhavou situaci zmírnit. Systémy CAD (Computer aided diagnosis) s použitím AI mají potenciál využití při diagnóze rentgenových snímků hrudníku, mamografii, magnetické rezonanci, výpočetní tomografii, screeningu onemocnění sítnice či diagnostiky rakoviny plic [18].

Přehled využití umělé inteligence v endoskopii vytvořil tým Okagawa et al. [19]. Z jejich závěrů vyplývá, že některé CAD systémy jsou již aplikovány v klinické praxi. AI může asistovat při vyšetřeních jícnu, žaludku, tenkého střeva nebo kolonoskopii. Dokáže odhalovat polypy, časná stádia rakoviny, infekci nebo zánět. Umělá inteligence pro asistenci při gastrointestinální endoskopii se v blízké budoucnosti objeví jako součást klinické praxe. Okagawa et al. však nejsou spokojeni se současným výzkumem, jelikož jeho kvalita je nízká. Kritizují použití statických snímků a retrospektivitu studií, což může podle jejich názoru vést ke zkreslenému výběru. V současném stavu je AI v endoskopii použitelná jako kontrola diagnostika či pro školení nových diagnostiků. Potenciál Okagawa et al. vidí v multicentrických prospektivních studiích za použití videa, a zvýšení specificity modelů tréningem za použití velkého počtu snímků s nenádorovými lézemi.

Dalšími diagnostickými metodami, ve kterých AI může nalézt uplatnění, je ultrasonografie a biochemická vyšetření. Použití AI může podpořit tradiční ultrazvuková vyšetření nádorů štítné žlázy, prsu, plicních sklípků nebo gynekologických lézí, a to s vysokou přesností a účinností. AI se také může podílet na analýze genů, které jsou mutovány u myeloidní leukémie a tím se podílet na včasném záchytu a monitorování této choroby. Systémy hlubokého učení se také mohou podílet na včasné diagnostice syndromu Noonanové a astmatu [18].

Algoritmy AI mají také dobré vyhlídky v patologii. Na proces segmentace obrazu, identifikace nádoru a určení metastáz se zaměřili Wang et al. [20]. Jejich práce nalezla postup s použitím deep learningu, který je rychlejší a přináší kvalitnější výsledky než metody bez použití deep learningu. Komura a Ishikawa [21] poukazují na fakt, že v klasifikačních úlohách v oblasti patologie mohou algoritmy AI předčit lidské lékaře v přesnosti, avšak kooperace CAD a lékařů je preferovaná, jelikož systémy s použitím AI a lékaři selhávají různými způsoby.

Uplatnění systémů využívajících AI je nejenom v diagnostice. Trendem chirurgie v posledních 20 letech je robotizace operací a operačního sálu pomocí robotických systémů jako jsou Senhance, MiroSurge nebo Versius [22,23,24]. Samotná operace je prováděna pomocí robota, který napodobuje pohyby lidských rukou v těle pacienta. Robot je ovládán chirurgickým expertem například pomocí joysticku a operace je sledována stetoskopickou kamerou poskytovanou daným robotem. Jedním z nejznámějších a nejpokročilejších robotických systémů je systém da Vinci, instalovaný například v centru robotické chirurgie Krajské zdravotní [25]. Vyspělost robotických chirurgických systémů jako da Vinci nabízí potenciál využití AI nejen jako pomocníka pro zmírnění rizik operace, ale teoreticky i převzetí celého procesu operace a samostatného výkonu chirurgických zákroků [18, 26]. Dnešní robotické chirurgické systémy již dokáží samostatně vykonávat jednodušší části zákroku, jako je šití [27].

AI také nachází uplatnění v perioperační fázi čili fázi před a po operaci, a ve výzkumu a tvorbě léčiv. Může se podílet na identifikaci rizik a pooperačních komplikací pacientů, kontrole anestezie, či sběru a vyhodnocování dat ze senzorů na jednotkách intenzivní péče, ale i administrativních činnostech jako je predikce doby pobytu pacienta v pooperační péči [18, 28].

V odvětví výzkumu a výroby léčiv mohou metody umělé inteligence nahradit staré metody pokusu a omylu při začleňováních nových léčiv do vhodných lékových forem s požadovanými vlastnostmi, konkrétně řešit problémy se stabilitou, rozpustností a porézností konkrétních léčiv. Lze je využít i v klinických studiích pro výběr vhodných pacientů, kdy nábor vhodných pacientů v současnosti zabírá až třetinu času konkrétního klinického testování. V procesu výroby se pak mohou systémy AI podílet přímo na optimálním nastavení výroby a kontrole kvality [29].

Neposledním využitím AI v lékařství je využití v managementu nemocnic a studiu nových lékařů. Plánování a rozdělování zdrojů v nemocničním prostředí je pro managery nemocnic často těžký úkol. Studie Nas a Koyuncu ukazuje, že použití algoritmů deep learningu pro predikci příchodu pacientů na oddělení urgentního příjmu je přesnější než použití například Poissonova rozdělení, což vede ke snížení čekací doby a přeplněnosti oddělení a zároveň zvyšuje spokojenost pacientů [30]. Rekonstrukce 3D lékařských modelů pro výuku anatomických struktur, simulace pro studium chirurgických technik nebo sledování duševního zdraví studentů medicíny jsou jen některé věci, ve kterých je využito umělé inteligence pro podporu vzdělávání nových lékařů [18].

Medicína je tedy odvětví, ve kterém má umělá inteligence potenciál širokého uplatnění. Lze ji uplatnit od diagnostiky přes léčebný proces, výrobu a výzkum léčiv, po zdravotnický management a výuku. Metody AI mohou zejména zefektivnit velké množství procesů v lékařské a nemocniční praxi. Tato efektivita může pomoci snížit náklady na léčbu a výkonnost doktorů, čímž pomůže uspokojit poptávku po lékařské péči a rozšířit její dostupnost.

3.2.2 Výzvy pro AI v klinickém prostředí

V posledních letech strmě vzrostl počet studií umělé inteligence v medicíně [31]. Přestože se systémy umělé inteligence opakovaně osvědčují v různých retrospektivních lékařských studiích, do medicínské praxe bylo převedeno relativně málo nástrojů umělé inteligence. Rajpurkar et al. [32] identifikují výzvy, kterým musí AI v medicíně čelit.

Jako první typ problému byly určeny problémy s datovými soubory. Existuje zde několik konkrétních problémů, jako je cena pořízení datového souboru, velké velikosti snímků jejichž zpracování je náročné na paměť počítače, zkreslení datového souboru v důsledku použití dat vytvořených jedním přístrojem, či anotování datového souboru, které je prováděno specialisty, tudíž nákladné, nebo pomocí crowdsourcingu, tudíž méně přesné. K obdobnému závěru dospěli i Li et al. [33], kteří podotýkají, že výzkumníci aplikující metody AI do medicíny mají většinou zázemí v oblasti informatiky, a nemají přístup k přístrojům na pořizování medicínských snímků ani k spolehlivé cestě, jak snímky anotovat.

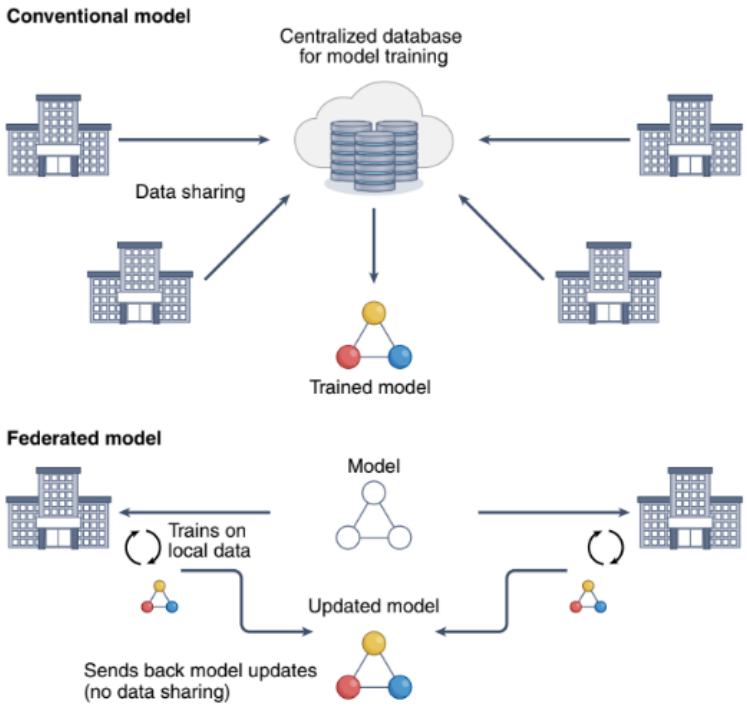
Druhým problémem identifikovaným Rajpurkar et al. [32] je budování důvěry v model strojového učení. Modely musí být přesné, jejich výsledky vysvětlitelné a musí být dostatečně jednoduché k použití, aby je bylo možné integrovat do klinické praxe. Reprodukovatelnost a opakovatelnost studií bohužel představuje výzvu, jelikož datové sady, kód a natrénované modely často nejsou zveřejňovány, což znesnadňuje nezávislé ověření a navazování na předchozí výsledky.

Dalším problémem identifikovaným je odpovědnost. Sem řadí jak změny v odpovědnosti, přecházející z lékařů na systémy na podporu rozhodování, ale i problémy v zákonných procesech či normách, které nejsou připraveny na iterativní povahu průběžného zdokonalování zmíněných systémů založených na AI.

Helen Smith [34] poznamenává, že pokud lékař nemůže vysvětlit výstupy z AI systému, který používá, nemůže plně zodpovídat za své jednání, pokud se rozhodne tyto výstupy použít. Habli et al. [35] toto tvrzení rozšiřují v dilema. Lékaři za použití nevysvětleného výstupu AI systému buď musí věnovat čas vytvoření vlastního expertního názoru, což činí přidanou hodnotu takového systému nízkou, či musí slepě těmto výstupům věřit, čímž trpí důvěra mezi pacientem a daným lékařem.

Posledním identifikovaným problémem spravedlivost AI systémů. Medicínské datové soubory obsahují citlivá pacientská data, kterých mohou zneužít pochybní aktéři. Kvůli tomu vznikají instituce schraňující tyto datové soubory, které poskytují služby trénování AI modelů na těchto datech, bez přístupu autora AI modelu k těmto datům, znázorněné na obrázku 2. V některých případech je však teoreticky možné rekonstruovat data, na kterých byl model trénován. Vyvstává zde i etická otázka, zdali je vlastníkem medicínských dat pacient, či organizace, která data pořizuje. V každém případě by medicínská data měla být použita pro dobro pacientů. Bohužel zde může vznikat konflikt mezi zájmy pacientů a ochranou jejich dat – implementace ochrany dat vyžaduje značné prostředky a úsilí, a instituce které nemohou tyto požadavky splnit, nemusí mít přístup k určitým datovým souborům, i kdyby jejich výzkum prospěl daným pacientům [32].

Obrázek 2 Model poskytování dat pro trénování AI modelů. Tento model řeší citlivost medicínských dat, ale neřeší problém nákladnosti přístupu k těmto datům



Zdroj: [32]

K poslednímu problému se také váže problém rovnosti a předpojatosti, kdy některé systémy mohou špatně reprezentovat marginalizované skupiny, což může vést k poškození těchto skupin. Tento problém je možné částečně řešit vysvětlitelností výsledků modelu, jak je zmíněno výše. Pokud je známo, jaké faktory se podílely na výsledném rozhodnutí systému, lze zvážit jeho předpojatost [32]. Původ a genetická variabilita jsou proměnné, které mají vliv na výskyt některých chorob v různých populacích, například protein EGFR, který je rizikovým faktorem rakoviny, u populací s asijským původem. Může se tak zdát, že zahrnout parametry jako je etnicita do vstupních parametrů modelu je vhodné. Tyto parametry jsou však také zástupným ukazatelem vlivu jiných socioekonomických faktorů. Je tedy třeba zajistit spravedlivé a různorodé trénovací soubory dat [35].

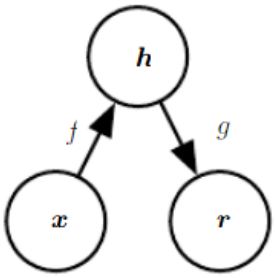
Z uvedených poznatků vyplývá, že přístup k robustním, anotovaným datovým souborům je jednou z největších překážek, kterým výzkumníci při tvorbě modelů AI čelí. Poskytování dat je však problematické, kvůli času strávenému anotováním datových sad či starostem o bezpečí citlivých informací o pacientech.

3.3 Využití metod hlubokého učení pro generování obrazové informace

3.3.1 Variační autoenkodéry

Autoenkodér je hlubokou neuronovou sítí vycvičenou pro kopírování svého vstupu na svůj výstup. Základní model autoenkodéru je popsán na obrázku 3 [36].

Obrázek 3 Grafické znázornění modelu autoenkodéru. Autoenkodér mapuje vstup x na výstup (rekonstrukci) r pomocí vnitřní reprezentace h. Autoenkodér je složen ze dvou částí: funkce f(x), která se nazývá enkodér, a funkce g(h), která se nazývá dekodér.



Zdroj: [36]

Historické využití autoenkodérů bylo využití možnosti změny rozměrů prostoru vstupu x na prostor vnitřní reprezentace s menšími rozměry (h). Této vlastnosti bylo dále využíváno pro kompresi obrazových dat či rozpoznání řeči [37].

Variační autoenkodéry naproti tomu řeší problém generování dat se stejnou pravděpodobnostní distribucí parametrů jako původní soubor pomocí prostoru latentních proměnných z, (který je alegorický k vnitřní reprezentaci h popsané v obrázku (3)). Tento latentní prostor obvykle obsahuje méně prvků než vstup x.

Princip, na kterém variační autoenkodéry pracují, je variační odvozování. Variační odvozování se zabývá odhadem pravděpodobnostního rozdělení, které se obtížně vypočítává. Variační odvozování je metoda, která převádí latentní veličiny původního pravděpodobnostního rozdělení na podmíněná rozdělení. Tato metoda aproximuje tato podmíněná rozdělení prostřednictvím optimalizace [38].

Problém řešený variačními autoenkodéry je následující: Předpokládá se, že množina dat $X = \{x^{(i)}\}_{i=1}^{N}$ s nezávislým a identickým rozdělení prvků je vygenerována náhodným procesem z latentní spojité množiny proměnných z. Tento proces má dva kroky:

1. Hodnota $z^{(i)}$ je vybrána z rozdělení $P_{\Theta^*}(z)$.
2. Hodnota $x^{(i)}$ je následně vygenerována z podmíněného rozdělení $P_{\Theta^*}(x|z)$.

Předpoklady pro tento proces jsou neřešitelnost (nemožnost vyhodnocení mezní pravděpodobnosti) tohoto procesu a velká velikost množiny prvků. Proto je nemožné použití EM algoritmu či obvyklých variačních bayesovských metod. Zároveň se jedná o úlohu, kde chceme parametry neuronové sítě upravovat pro jednotlivé datové body či malé dávky.

Řešení problému nastínili Kingma a Welling ve své práci Auto-Encoding Variational Bayes. Jejich řešení se pokoušejí o efektivní definici přibližných parametrů Θ, efektivní přibližné odvozování latentního prostoru z, a efektivní přibližné odvození prvku x. Kvůli předpokladům uvedeným v předchozím odstavci navrhují využití SGVB (Stochastic Gradient Variational Bayes) jako estimátoru pro optimalizaci procesu autoenkodéru, a algoritmus AEVB (Auto-encoding Variational Bayes) [39].

Autoenkodéry tedy obsahují proces kódování dat a jejich dekódování. Obvyklým příkladem použití je komprese obrazu, kdy hlavními parametry je snížení velikosti vnitřní reprezentace h oproti původnímu obrazu x, a následná podobnost rekonstrukce r taktéž k původnímu obrazu x, ideálně identické zobrazení. Oproti tomu variační autoenkodéry mají za cíl transformaci rozdělení množiny dat X s prvky $x^{(1)}$, …, $x^{(N)}$ do latentního prostoru z, ze kterého lze následně konstruovat generovaná data g, která ideálně patří do původního rozdělení množiny dat X.

3.3.2 Rekurentní soupeřící sítě

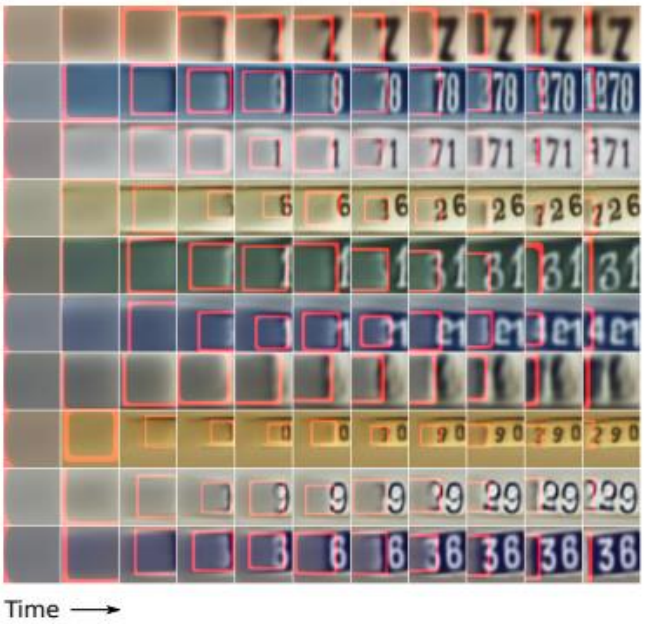
Typické variační autoenkodéry kódují, respektive dekódují vstup během jednoho průchodu neuronovou sítí, obdobně jako fotoaparáty ukládají celý obraz po stisknutí spouště. Pro rekurentní soupeřící sítě je však využíván jiný princip. Stejně jako obraz je malován malířem od hrubých obrysů postupně k detailu, rekurentní soupeřící sítě jako DRAW (Deep recurrent attentive writer), navržená Gregorem et al. [40] generují obraz postupně. Tyto sítě patří do skupiny variačních autoenkodérů, ale jejich enkodéry a dekodéry jsou rekurentními sítěmi.

Existují tři hlavní rozdíly mezi variačními enkodéry a rekurentními soupeřícími sítěmi jako DRAW. Zaprvé, enkodér i dekodér jsou rekurentními sítěmi, takže si mezi sebou vyměňují sekvenci vzorků z latentního prostoru. Enkodér také zná předchozí výstupy dekodéru, dle čehož přizpůsobuje vybrané vzorky dle dosavadního chování dekodéru.

Druhým rozdílem je, že výstupy dekodéru jsou postupně přidávány do distribuce, která nakonec generuje data, na rozdíl od vysílání této distribuce v jednom kroku. Třetím rozdílem je využití dynamicky aktualizovaného mechanismu, který určuje pozornost čili omezuje jak vstupní oblast sledovanou enkodérem, tak výstupní oblast modifikovanou dekodérem.

Rekurentní soupeřící sítě jako DRAW tedy nejen že negenerují obraz v jednom kroku, ale jsou schopné vybrat oblast právě generovaného obrazu, na kterou se zaměří v dalším kroku generování. Tímto způsobem tedy doplňují detaily do generovaného obrazu, jak ej patrné z obrázku 4[40].

Obrázek 4 Postupné generování vzorků pro dataset Street view house numbers. Červený rámeček zobrazuje místo ve snímku, na které je zaměřeno generování sítě. Síť tedy postupně doplňuje do snímku detaily



Zdroj: [40]

3.3.3 Difuzní modely

Difuzní modely, na rozdíl od variačních autoenkodérů, založených na Bayesovksé statistice, využívají princip Markovova řetězce k přeměně daného rozdělení na jiné rozdělení. V těchto generativních modelech je používán difuzní proces k transformaci vstupního (například Gaussova) rozdělení na cílové rozdělení. Inspirují se v nerovnovážné statistické fyzice a metodách sekvenčního Monte Carlo [41].

Difuzní modely jsou složeny ze dvou částí. První částí je dopředný difuzní proces, který převádí libovolné složité rozdělení na jednoduché rozdělení. Druhou částí je převrácení

tohoto procesu tak, aby byl spočetný v konečném čase. Tím je získán generativní proces. Tyto procesy se nedějí naráz, ale mají vysoký počet kroků čili vysoký počet vrstev neuronové sítě. Rychlost procesu je dána difuzní mírou β_t. Optimální míra β_t závisí na typu difuze [41].

Pro zvýšení rozlišení generovaných vzorků lze použít například kaskádové modely. Kaskádování je jednoduchá technika modelování dat pomocí naučené řady samostatně natrénovaných modelů se zvyšujícím se rozlišením. První model v řadě generuje vzorky s nízkým rozlišením, které si přebírá model následující na svém vstupu, a na výstupu zvyšuje jejich rozlišení. Tato řada se opakuje, dokud není dosaženo cíleného rozlišení. Tato technika není závislá na difuzních modelech a lze ji teoreticky použít i pro VAE a GAN modely [42].

3.3.4 Generativní soupeřící sítě

První dva zmíněné přístupy jsou založené na Bayesovksé statistice, respektive Markovových řetězcích. Generativní soupeřící sítě podobně využívají principy zakořeněné v teorii her, matematického oboru, který je úzce spojen s ekonomií. Konkrétně používají dvě sítě, zvané generátor a diskriminátor, které spolu soutěží v rámci odpovídajícímu minimaxové hře dvou hráčů [43,44].

Logickou otázkou, která vyvstává při hře dvou hráčů je existence Nashovy rovnováhy. Farnia a Ozdaglar se ve své práci [45] věnují právě tomuto tématu. Na empirických příkladech i teoreticky ukazují, že v případě generativních soupeřících sítí nemusí vždy existovat Nashova rovnováha. To vzniká ze sekvenčního trénování generativních soupeřících sítí, které je popsáno níže. Tím je praktické trénování těchto druhů sítí s konečným počtem kroků zkomplikováno. Farnia a Ozdaglar zavádí termín přibližná rovnováha, který lépe odpovídá sekvenčnímu charakteru trénovaní těchto sítí. Zbytek této práce se také bude vyhýbat termínu Nashova rovnováha v kontextu generativních soupeřících sítí.

Principem funkce GAN je minimaxové hra dvou hráčů, kde spolu soutěží typicky dva vícevrstvé perceptrony. Prvním vstupem do GAN je pole šumu $p_z(z)$, které reprezentuje mapovací prostor reálné proměnné x na rozdělení generátoru p_g, pro generátor $G(z;\theta_g)$, kterým je v tomto případě vícevrstvý perceptron, kde θ_g reprezentuje parametry tohoto perceptronu. Další částí je diskriminátor $D(x;\theta_d)$, který je v tomto případě také vícevrstvý

perceptron s parametry θ_d. Výstupem diskriminátoru je skalár $D(x)$, který vyjadřuje pravděpodobnost, že vstup x patří do množiny reálné proměnné [43].

Trénování D je maximalizace pravděpodobnosti přiřazení správného labelu pro reálná i generovaná data. Tréning G pak lze vyjádřit jako minimalizaci funkce:

$$\log\Big(1 - D\big(G(z)\big)\Big). \tag{3}$$

V praxi však při začátku trénování je D schopen rychle rozlišit špatné výstupy z G a výše uvedená funkce pak neposkytuje požadované gradienty pro kvalitní trénování. Generátor G však nemusí minimalizovat pravděpodobnost odhalení diskriminátorem D, ale snažit se maximalizovat jeho chybu. Trénovací proces pro G pak lze vyjádřit jako maximalizaci funkce:

$$\log\Big(D\big(G(z)\big)\Big). \tag{4}$$

Výsledkem této účelové funkce je zachování dynamiky G a D, ale poskytuje mnohem silnější gradienty na počátku učení [43].

Původní implementace GAN je založena na Jensen-Shannon metodě určení vzdálenosti reálného a generovaného rozdělení [43, 46]. Tato metoda však dává pro určité soubory dat mizející gradienty, což vede k divergenci modelu, respektive k nízké schopnosti naučit se dané rozdělení. Proto Arjovsky et. al. [46] navrhují optimalizaci Wassersteinovy vzdálenosti, která poskytuje spojitou funkci, a tudíž použitelné gradienty, které vedou k nalezení optima a lepší konvergenci modelu. Navrhují novou architekturu přizpůsobenou použití právě Wassersteinovy vzdálenosti, kterou pojmenovávají Wasserstein GAN neboli WGAN a demonstrují vyšší stabilitu učení pro WGAN oproti původní GAN.

Wassersteinova vzdálenost, původně nazývána Earth Mover's Distance, byla navrhnuta jako metrika pro nepodobnost dvou (či více) rozdělení, původně pro obrazové databáze. Vychází z řešení speciálního případu dopravní úlohy. Pokud jsou dána dvě rozdělení, Ruber et. al. tvrdí, že na jedno lze pohlížet jako na haldu hlíny rozloženou v prostoru, a na druhé jako soubor děr skrze ten samý prostor. Wassersteinova vzdálenost je pak minimální množství práce, které je potřeba vynaložit pro zaplnění daných děr hlínou, kde prací se rozumí suma přemístěných množství hlíny krát vzdálenost přemístění [47]. Tato

vzdálenost W ve WGAN, narozdíl od vzdálenosti D v GAN, neurčuje přímo pravděpodobnost správnosti labelu, ale jedná se o nějaké libovolné číslo, které se diskriminátor pokouší maximalizovat pro reálné vzorky, a naopak minimalizovat pro vzorky generované . V případě WGAN proto nehovoříme o diskriminátorech, ale o kriticích [46].

Podmínkou pro kritika ve WGAN je, že se jedná o funkci s lipschitzovským zobrazením. Tuto podmínku je možné zajistit pomocí tzv. ořezávání vah, který je v původním WGAN algoritmu nastaven pomocí ořezávacího parametru c [46]. Samotné využití tohoto přístupu však může vést k problémům s optimalizací kritika, jelikož ořezávání vah k zajištění lipschitzovkého zobrazení vede kritika k hledání jednoduchých funkcí. Dalším problémem mohou být velké nebo naopak mizející gradienty při nesprávném nastavení ořezávacího parametru c [47].

S jiným způsobem vynucení pravidla Lipschitzova zobrazení pro kritika ve WGAN přichází Gulrajani et. al. [48]. Jedná se o přímé omezení normy gradientu výstupu kritika vzhledem k jeho vstupu. Je vyjádřeno jako:

$$L = E_{\tilde{x}\sim P_g}\left[D(\tilde{x})\right] - E_{x\sim P_r}\left[D(x)\right] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right], \qquad (5)$$

kde:

$$E_{\tilde{x}\sim P_g}\left[D(\tilde{x})\right] - E_{x\sim P_r}\left[D(x)\right] \qquad (6)$$

je původní chyba kritika, a

$$\lambda E_{\hat{x}\sim P_{\hat{x}}}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right] \qquad (7)$$

je nově navržená gradientní sankce.

Cílem této metody je tedy eliminace některých možných problémů původně navrhovaného způsobu WGAN, které jsou uvedeny výše. Jelikož se používá gradientní sankce, tj. gradient penalty, nazývá se tato implementace jako WGAN-GP. Další výhodou této metody je, že nemusí využívat dávkové normalizace, která se běžně využívá v GAN k stabilizaci tréninku, ale mění původní problém kritika z mapování jednoho vstupu na jeden výstup na mapování celé dávky vstupů na dávku výstupů [48].

3.3.5 Latentní prostor

Latentní prostor v kontextu generativních neuronových sítí je abstraktní zachycení struktury původního trénovacího souboru. Vzorek latentního prostoru je vstupem do generativní části modelu (dekodér, generátor). Cíl trénování generativního modelu se dá tedy vyjádřit jako naučení se mapovat vzorky z latentního prostoru do rozdělení původního datového souboru neboli zachytit vlastnosti původního prostoru do svého latentního prostoru [49].

Latentní prostor generativního modelu jako GAN nebo VAE má tedy několik zajímavých vlastností. Nejdůležitější z nich je extrakce vlastností původního trénovacího datového souboru do smysluplné hierarchie prvků. Z této vlastnosti latentního prostoru vychází aritmetika vlastností, demonstrovaná na následujícím příkladu:

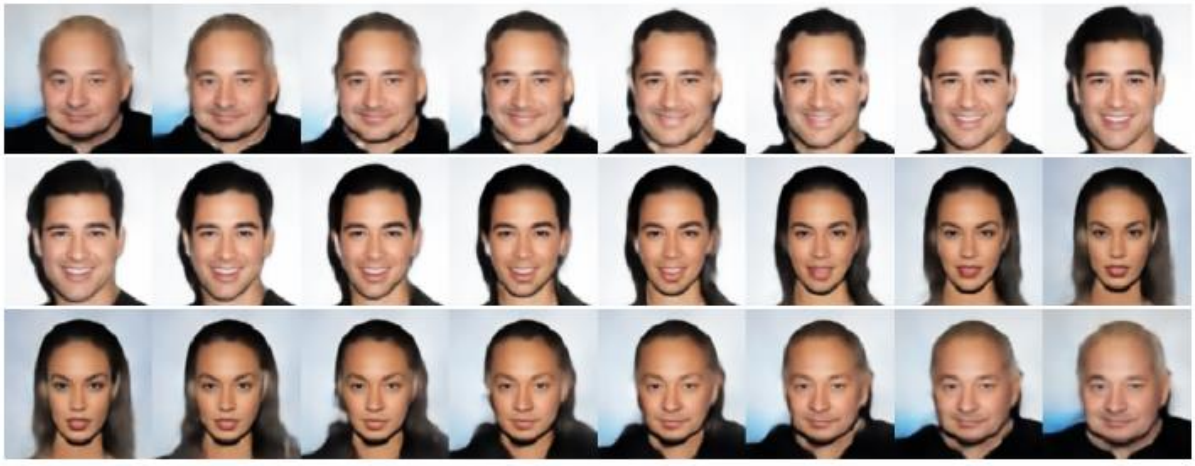
Mějme generativní síť generující snímky obličejů natrénovanou například na datovém souboru CelebA [50] z latentního prostoru Z. Byla získána množina snímků A, zobrazující ženy, ze vzorku latentního prostoru Z_A. Dále máme k dispozici množiny snímků B a C, zobrazující muže s brýlemi a muže bez brýlí, a jejich vzorky latentního prostoru Z_B a Z_C. Pokud získáme průměrný vzorek latentního prostoru muže s brýlemi $\bar{z}_B$, odečteme od něj průměrný vzorek muže $\bar{z}_C$ a přičteme průměrný vzorek ženy $\bar{z}_A$, získáme vzorek z_d, jehož vygenerovaný snímek d bude zobrazovat ženu s brýlemi. Odečtením reprezentace muže ze snímku muže s brýlemi byla extrahována vlastnost brýlí, v jejímž směru byla posunuta reprezentace ženy, za výsledku reprezentace ženy s brýlemi [51] Toto je patrné na obrázku 5.

Obrázek 5 Aritmetika vlastností latentního prostoru generativních modelů

Zdroj: [51]

Další zajímavou vlastností latentního prostoru je možnost interpolace. Lineární interpolace v latentním prostoru je generativní funkcí převáděna na sémantickou interpolaci v prostoru výstupních snímků. Tyto vlastnosti značí, že generativní neuronové sítě umí generalizovat původní datové soubory, a jsou demonstrovány na obrázku 6 [51].

Obrázek 6 Interpolace prostoru výstupních snímků. Vytvořené snímky nespadají k zřejmému průměrnému snímku, ale obsahují vlastnosti obou interpolovaných snímků



Zdroj: [51]

3.4 Tensorflow

Tensorflow je open-source knihovna, používaná pro potřeby strojového učení. Některými případy použití jsou číselné výpočty založené na vícerozměrných polích, automatická diferenciace, konstrukce, trénování a export modelů, distribuované zpracování a zpracování na GPU [52]. Tensorflow nabízí možnost vyhodnocování modelů ve dvou módech, tzv. dychtivém (eager) zpracování a zpracování grafem [53].

Tensorflow byl původně vyvinut společností Google a v roce 2015 byl vydán pod licencí Apache 2.0. Výpočetní kernel této knihovny je napsán v jazyce C++ s použitím rozhraní CUDA. Samotné uživatelské rozhraní je implementováno v jazyce Python. Knihovna je založena na výpočetních blocích spojených datovými toky do výpočetního grafu. Data se zpracovávají přechodem z jednoho bloku do druhého. To usnadňuje použití paralelních výpočtů na procesorech s více jádry, na GPU, TPU či distribuovaných clusterech, což je vhodné pro výpočty používané v neuronových sítích [54].

3.4.1 Základní prvky Tensorflow

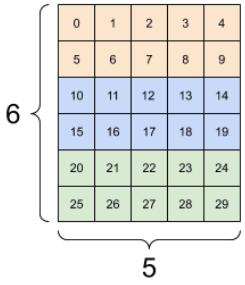
3.4.1.1 Tensory

Základním datovým typem, který je pomocí Tensorflow zpracováván, je tensor. Tensor je vícerozměrné pole obsahující jednotný typ. Tensory jsou neměnné. Obsah tensoru nelze změnit, lze pouze vytvořit nový tensor. Implementace tensorů (tensorflow.Tensor), zobrazena na obrázku 7, předpokládá, že podél každé osy je každý prvek stejně velký. Pro tensory s různými velikostmi existují speciální typy tensorů jako Ragged tensor či Sparse tensor [55].

Tensory jsou definovány zejména následujícími vlastnostmi:

- Tvar: délka (počet prvků) každé z os tensoru,
- pořadí: počet os tensoru,
- dimenze: dimenze tensoru,
- velikost: celkový počet prvků tensoru [55].

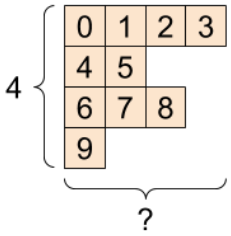
Obrázek 7 *tf.Tensor*



Zdroj: [55]

Pro tensor s proměnným počtem prvků podél některé osy lze použít Ragged tensor, znázorněný na obrázku 8 [55].

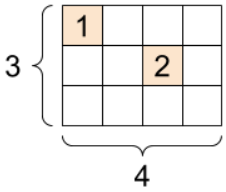
Obrázek 8 *tf.ragged.RaggedTensor*



Zdroj: [55]

Pro tensory s řídkými daty je vhodné použití Sparse tensor, který je vyobrazen na obrázku 9 [55].

Obrázek 9 *tf.sparse.SparseTensor*



Zdroj: [55]

3.4.1.2 Proměnné

Pro reprezentaci sdíleného, trvalého stavu modelu je doporučeno používat proměnné (tensorflow.Variable). Proměnné představují tensory, kterým lze měnit hodnotu operacemi, které jsou na nich spouštěny. Proměnné jsou využívány například knihovnou Keras k ukládání parametrů modelu [56].

Implementace proměnných je založena na tensorech. Lze s nimi tedy provádět stejné operace a mají stejné vlastnosti [56]. Některé operace se ale pro tensory a proměnné chovají

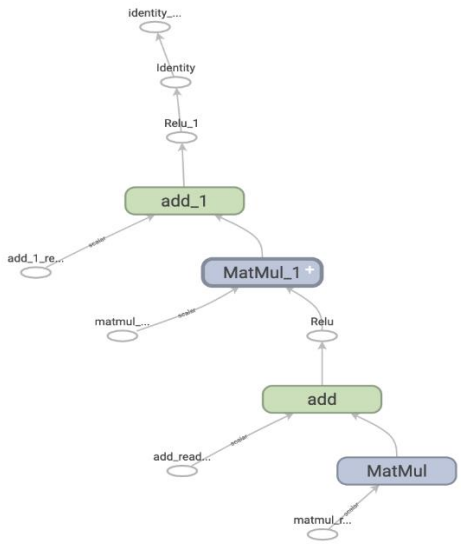
různě. Například zpětná propagace chyby pomocí tensorflow.GradientTape může pracovat jak s tensory a proměnnými, ale s proměnnými pracuje ve výchozím nastavení, zatímco tensory pro práci musí být přímo specifikovány [57].

3.4.1.3 Grafy, dychtivé zpracování

Pro tvorbu modelů v Tensorflow jsou dostupné dvě základní možnosti. První možností je definice a trénování modelu přímo v Pythonu. Postup je procedurálně vyjádřen jako operace v kódu, a je tak také vykonáván. Tato možnost se nazývá dychtivé (eager) zpracování. Tato metoda je obecně pro uživatele jednodušší, ale je vázána přímo na jazyk Python.

Druhou metodou je definice postupu pomocí grafu. Graf je v tomto kontextu datová struktura, která obsahuje jednotlivé objekty operace (tensorflow.Operation) a tensory, jimiž jsou představována data proudící mezi operacemi. Graf je zobrazen na obrázku 10. Protože grafy jsou datové struktury, lze je ukládat a spouštět nezávisle na jazyku [58]. Definice a spouštění pomocí grafu je možná i pomocí dekorátoru @tensorflow.function [59].

Obrázek 10 Vizualizace grafu



Zdroj: [58]

3.4.1.4 Vrstvy a modely

Pro provádění strojového učení (ML) pomocí TensorFlow se definují, uchovávají a načítají modely. Model je množina proměnných, které se aktualizují trénováním. Modely jsou typicky složeny z vrstev (tensorflow.Module). Vrstvy mají vnitřní stav a metody, které používají tento stav, jsou proto ekvivalentem pro objekty [60].

3.4.2 Keras

Keras je vysokoúrovňové rozhraní knihovny TensorFlow. Jsou jím pokryty všechny kroky v běžném procesu ML, jako zpracování dat, ladění parametrů či nasazení modelu. Keras usnadňuje práci s knihovnou TensorFlow například pomocí jednoduššího a konzistentního rozhraní, snížení počtu kroků nutných pro dosažení častých cílů či jasných chybových hlášení [61].

Jelikož Keras vychází z knihovny TensorFlow, pro běžné potřeby ML jsou používány vrstvy (tensorflow.keras.layers.Layer), které se seskupují v modely (tensorflow.keras.Model). Základním typem modelu je sekvenční model, který je řadou vrstev s právě jedním vstupním, a právě jedním výstupním tensorem. [61, 62].

Pro jiné typy modelů lze využít funkční API. Je tudíž možné definovat složitější typy modelů, například modely s nelineární topologií, sdílenými vrstvami, nebo s různým množstvím vstupů a výstupů. Modely jsou definovány zadáním vstupů a výstupů v grafu vrstev. To znamená, že grafy vrstev lze využít při definici více modelů. Použití funkční API umožnuje oproti přímému použití modelu z knihovny TensorFlow ověření modelu při definování, plánovatelnost, kontrolovatelnost, a jednodušší serializaci a klonování modelu [63].

TensorFlow 2 ve výchozím nastavení spouští modely dychtivě [59]. Toto spouštění tedy ve výchozím nastavení používá i Keras.

3.5 Pytorch

Pytorch je framework pro vytváření modelů hlubokého učení. Je vytvářen vývojáři z Facebook AI Research a dalších laboratoří. Pytorch kombinuje framework Torch [64], vědecký výpočetní framework s širokou podporou algoritmů strojového učení zaměřený na výpočty na GPU, s rozhraním v jazyce Python. Pytorch byl roku 2017 vydán jako open source, což pomohlo jeho rozšíření mezi veřejnost [65].

Pytorch je obdobně jako Tensorflow postaven na výpočetních grafech. Rozdílem mezi grafy v Tensorflow a Pytorch je, že Pytorch využívá dynamické výpočetní grafy, na rozdíl od knihovny Tensorflow, kde je výpočetní graf statický. Python je do Pytorch těsně integrován, což prakticky znamená snadnější použití oproti knihovně Tensorflow. Obecně

lze ale Tensorflow považovat za dominantní, jelikož byl vydán dříve a získal si tedy dříve širší obecenstvo [54].

3.5.1 Základní prvky Pytorch

3.5.1.1 Práce s daty ve frameworku Pytorch

Pro práci se vstupními daty využívá Pytorch dvou tříd, které se vzájemně doplňují. První třídou je Dataset, která ukládá data a jejich labely [66]. Pytorch podporuje dva různé typy pro třídu Dataset. Tyto styly jsou iterativní styl a mapový styl. Iterativní styl má implementován protokol iterable. Tím reprezentují iterátor přes vzorky dat či proud dat.

Druhým stylem je mapový styl, který dovoluje přístup k datům nejen sekvenčně, ale i pomocí indexu. Je vhodnější pro vzorky dat o známé délce. Nahrávání dávek dat je v iterativním stylu závislé na konkrétní uživatelově implementaci, pro mapový styl se pak využívá třídy Sampler [67].

Druhou třídou pro práci se vstupními daty je třída DataLoader. Tato třída kombinuje Dataset a Sampler, a poskytuje iterable přes daný Dataset. Rozdělení práce se vstupními daty do těchto dvou tříd představuje oddělení zájmů, kde třída Dataset je abstrakcí pro reálné vzorky dat a třída DataLoader poskytuje vzorky dat pro trénování daného modelu [68].

Z toho vyplývá, že transformace reálných vzorků dat na vzorky dat vhodné k trénování modelů probíhá mezi těmito dvěma třídami. Konkrétně se transformace vzorků dat i jejich labelů děje při vytváření objektu Dataset. Pro transformaci dat se používá parametr *transform*, pro transformaci labelů se používá parametr *target_transform* [69].

Pro trénování modelu jsou obdobně jako v Tensorflow používány tensory. Tensor je v Pytorch specializovaná datová struktura, která se používá jak pro vstupy a výstupy modelů, tak pro parametry modelu. Je podobná poli (array) z knihovny NumPy [70].

3.5.1.2 Definice modelů neuronových sítí

Pro definici neuronových sítí se ve frameworku Pytorch používá především torch.nn. Torch.nn obsahuje základní typy vrstev, jako jsou konvoluční, transformační, lineární, normalizační či rekurentní, abychom jmenovali alespoň některé z nich [71].

Torch.nn také poskytuje ostatní nástroje pro definici neuronové sítě v jazyce Python. Základní třídou pro všechny části či kompletní neuronové sítě je nn.Module. Moduly mohou také obsahovat další moduly jako atributy podtřídy, umožňující vnořování modulů do stromové struktury. Třída Module také obsahuje metodu forward, která musí být přepsána každou podtřídou. Tato metoda definuje výpočet či událost, která je implementací operací na vstupních datech [72].

Definice struktury neuronové sítě probíhá v konstruktoru podtřídy nn.Module. Pro zabalení vrstev dohromady se typicky využívá třídy nn.Sequential. Data se pak předávají mezi vrstvami v pořadí, v jakém jsou definovány v konstruktoru třídy Sequential.

Některé vrstvy mají parametry, které se během trénování optimalizují. Podtřídy třídy nn.Module automaticky registrují tyto parametry, což umožňuje jejich inspekci a zjednodušuje trénování modelů [73].

3.5.2 Trénování modelů

Jedním z nejpoužívanějších postupů pro trénování neuronových sítí, a postup využívaný v této práci, je použití algoritmu zpětné propagace chyby. V tomto algoritmu se parametry (váhy modelu) upravují podle gradientu ztrátové funkce vzhledem k danému parametru. Pro výpočet těchto gradientů se ve frameworku Pytorch nachází torch.autograd, jímž je podporován automatický výpočet gradientu pro libovolný graf neuronové sítě. Výpočet gradientu lze provádět pro každý tensor pomocí metody backward. Pytorch nabízí různé ztrátové funkce jako například nn.CrossEntropyLoss, či dovoluje použít vlastní ztrátové funkce [74].

Po výpočtu gradientu následuje optimalizace parametrů. Optimalizační metody jsou implementovány v torch.optim. Pro použití optimalizace je potřeba vytvořit instanci některého optimalizátoru, která uchovává aktuální stav a upravuje parametry na základě vypočtených gradientů [75].

3.6 Porovnání frameworků Tensorflow/Keras a Pytorch

Z uvedených skutečností vyplývá, že Tensorflow a Pytorch mají velkou míru alegorických prvků. V obou frameworcích lze řešit širokou paletu úloh strojového učení, strojového vidění či deep learningu. Frameworky lze porovnávat objektivně a dojít k určitému závěru,

je však nutné podotknout, že subjektivní názor, komunita a efektivita v daném prostředí jsou faktory, které mají na výsledné rozhodnutí o použití daného frameworku uživatelem zásadní vliv.

Porovnání výkonu tréninku v prostředích Tensorflow a Pytorch se věnují například Dai et. al. [76]. Jejich práce se zabývá výkonem tréninku v prostředích s jedním GPU. Identifikují Tensorflow a Pytorch jako dva nejpopulárnější frameworky; Pytorch kvůli jednoduchosti jeho programování a ladění chyb, Tensorflow kvůli dobré vizualizaci a komunitě.

Dai et. al. také vysvětlují rozdíl mezi Pytorch a Tensorflow v přístupu ke spouštění výpočetních grafů. Tensorflow vytváří výpočetní graf před spuštěním, což umožňuje grafovou optimalizaci a vyšší efektivitu provedení. Pytorch naopak preferuje dynamičtější přístup, kdy uživatel může definovat a provádět uzly grafu v běhu programu. To frameworku Pytorch znemožňuje optimalizaci před provedením, což v kombinaci se závislostí na interpretu jazyka Python ukazuje na nevýhodu frameworku Pytorch oproti Tensorflow.

Srovnání počtu funkcí Stančin a Jović [77] provedli v roce 2019. Tensorflow mělo v kategorii vrstev 14 ze 17 sledovaných, v aktivačních funkcích 12 z 18, v ztrátových funkcích 11 ze 17 a v optimalizačních funkcích 10 ze 17. Pytorch měl 13, 13, 11 a 10 respektive. Celkem mají oba shodně 47 ze 69 všech sledovaných funkcí. Tato informace je sice zastaralá, například Pytorch již ve verzi 2.1 poskytuje vrstvu flatten [78], ale slouží jako ilustrace podobného rozsahu funkčností obou frameworků.

Dai et. al. [76] shrnují některé poznatky o rozdílech mezi Tensorflow a Pytorch a podotýkají, že je obtížné systematicky vysvětlit příčiny rozdílných výkonů různých grafů v obou frameworcích, což je dáno rozdílnou konstrukcí výpočetního grafu a plánování běhu programu.

Jejich práce definuje 5 poznatků:

1. Většinu času trénování modelu spotřebovává výpočetní čas GPU, což ukazuje, že při trénování hraje zásadní roli implementace prvků, které, jako jsou konvoluční vrstvy či LSTM.

2. Konvoluční vrstvy mohou mít velké množství implementací s různou rychlostí. Rychlejší algoritmy obvykle používají více paměti. Dai et. al. tvrdí, že Tensorflow má lepší správu paměti než Pytorch, což znamená, že pro velké modely či velké trénovací dávky je vhodnější Tensorflow.
3. Při použití konvolučních vrstev v modelu mají uživatelé možnost ovlivnit výběr algoritmu, který bude použit pro výpočet. Pokud je cílem pouze trénování malého počtu epoch, je výhodné použít rozhraní heuristického volení konvolučního algoritmu. Pro vyšší počet epoch je výhodné nastavit režim profilování.
4. Pro implementace LSTM je nejvýkonnější implementace cuDNNLSTM. Tato implementace však neposkytuje dostatečně široké rozhraní, které by pokrývalo všechny uživatelské požadavky.
5. Optimalizace grafu prováděná frameworkem Tensorflow nevykazuje výrazně vyšší rychlost trénování a neměla by tedy být rozhodujícím faktorem při volbě mezi Pytorch a Tensorflow.

Novac et. al. [79] se zabývají výběrem mezi frameworky Pytorch a Tensorflow, konkrétně pro použití s konvolučními neuronovými sítěmi. Zdůrazňují, že s počtem dostupných knihoven neuronových sítí a podobností jejich funkcionalit roste obtížnost rozhodnutí, která knihovna je pro konkrétní úlohy vhodnější.

Dle jejich výsledků shrnují, že Pytorch je více zaměřený na fázi návrhu, díky přívětivějšímu rozhraní, jednodušší instalaci, jednodušší integraci v řešeních, bez externích závislostí, a celkově vychází jako lepší volba pro začátečníky či při potřebě získání rychlých výsledků, jelikož z jejich měření vychází Pytorch o 25,5 % rychlejší při trénování a 77.7 % rychlejší při exekuci neuronové sítě.

Tensorflow je vhodnější pro pokročilé uživatele, jelikož je přesnější, a nabízí širokou podporu CUDA, na rozdíl od úzké podpory frameworku Pytorch. Framework Tensorflow nabízí širší možnosti při tvorbě a trénování neuronových sítí, a dovoluje větší množství kontroly průběhu tréninku [79].

4 Vlastní práce

4.1 Příprava trénovacího souboru

Velikost celého datového souboru představuje výzvy pro praktické účely trénování modelu GAN. Dávka snímků v plném rozlišení vyžaduje značnou alokaci paměti, a změna jejich dimenzí zvyšuje výpočetní zátěž. Kromě toho iterativní povaha trénování GAN, která se vyznačuje potřebou průběžně upravovat a optimalizovat generátor a kritika, vede k prodloužení doby trénování. Trénování sítí generujících rentgenové snímky tudíž vyžaduje značný výpočetní výkon a čas. Pro úspěšnou tvorbu reálného modelu je tedy potřeba tyto nároky snížit.

Pro účely práce bylo trénováno na podmnožině 100000 snímků. Tyto snímky bylo nutné převést do výsledné velikosti 256*256 pixelů, která je na výstupu generátoru a na vstupu kritika. Pro převod obrázků do požadované velikosti byl vytvořen následující python script, využívající knihovny Pillow:

```
import os
from PIL import Image


init_path = "/mnt/d/Files/AI/NIH chest Xrays/"
save_path = "./NIH/images/"
prefix = "images"
suffixes = [
      "_002","_003","_004","_005","_006",
      "_007","_008","_009","_010","_011",]
def resize_save(path, save_path, size=256):
    sizes = size, size # set the size of output image
    for data in os.listdir(path):
        image = os.path.join(path, data)
        fullsize = Image.open(image).convert("L") # Convert to
grayscale
        newsize = fullsize.resize(sizes,
resample=Image.Resampling.LANCZOS) # Resize the image
        newsize.save(os.path.join(save_path, data)) # Save image

for suffix in suffixes:
    path = os.path.join(init_path, f"{prefix}{suffix}/{prefix}/")
    print(f"Starting {suffix} suffix")
    resize_save(path, save_path)
    print(f"Suffix {suffix} finished")
```

4.2 Použité knihovny

Trénink GAN byl proveden s využitím souboru různých knihoven. Jádrem tohoto souboru je knihovna PyTorch, popsaná výše, nabízející nástroje pro tenzorové operace na GPU, dynamický výpočetní graf a mnoha modulů pro tvorbu hlubokých neuronových sítí.

Modul *torch.nn*, který je součástí knihovny PyTorch, obsahuje hlavní komponenty neuronových sítí použitých v této práci, jako je *nn.ConvTranspose2d, nn.Sequential* či *nn.LeakyReLU*. Tyto komponenty reprezentují vrstvy neuronové sítě či výstupní funkce, a je možné je optimalizovat pomocí algoritmu zpětné propagace chyby.

Pro využití vlastní trénovací sady dat je využito tříd *torch.utils.data.Dataset* a *torch.utils.data.DataLoader*. Dataset poskytuje načítání a převod snímků do matic za pomoci knihovny *numpy*, zatímco třída DataLoader poskytuje zamíchané dávky reálných dat pro účely trénování.

Pro sledování tréninku jsou použity metody *make_grid*, která z výstupů generátoru vytvoří matici snímků ve stupních šedi, tqdm.auto která poskytuje vizuální přehled o průběhu trénování, a knihovny matplotlib, která vykresluje generované snímky v průběhu trénování.

Souběžně s výše uvedenými knihovnami byla pro vývoj použita Jupyter notebook, open-source webová aplikace, umožňující vytváření a sdílení dokumentů, které obsahují živý kód, rovnice, vizualizace a popisný text. Použití tohoto rozhraní v rámci této práce nabízí možnost spouštění kódu po interaktivních blocích, což je velmi výhodné pro průzkumovou analýzu trénovacího souboru. Tento způsob spouštění kódu také nabízí rychlejší zpětnou vazbu, která podporuje experimentování s neuronovými sítěmi a trénovacím procesem a jejich iterativní zlepšování.

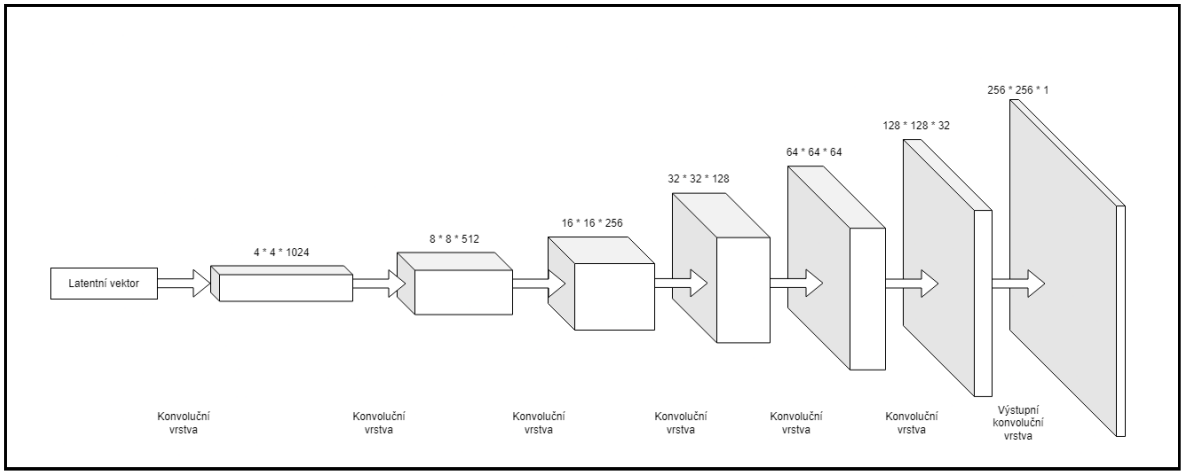
4.3 Architektura modelu

Model WGAN-GP se skládá ze dvou neuronových sítí: generátoru, který vytváří snímky transformací vzorků latentního prostoru, a kritika, který hodnotí reálnost snímků. Obě sítě použité v této práci jsou složeny z na sebe navazujících konvolučních vrstev.

Generátor začíná vstupní konvoluční vrstvou, která přetváří šum na tenzor 4*4*1024, postupně převzorkuje data v několika stupních (8 * 8, 16 * 16, ..., 256 * 256), přičemž v

každém kroku zdvojnásobí šířku a výšku a zároveň sníží počet kanálů, dokud nedosáhne konečné velikosti obrazu a požadované hloubky kanálu (pro obrazy ve stupních šedi 1 kanál).Vstupem generátoru je latentní vektor. Tento latentní vektor je v průběhu trénování náhodný vektor dané velikosti. Graf generátoru je k nahlédní na obrázku 11.

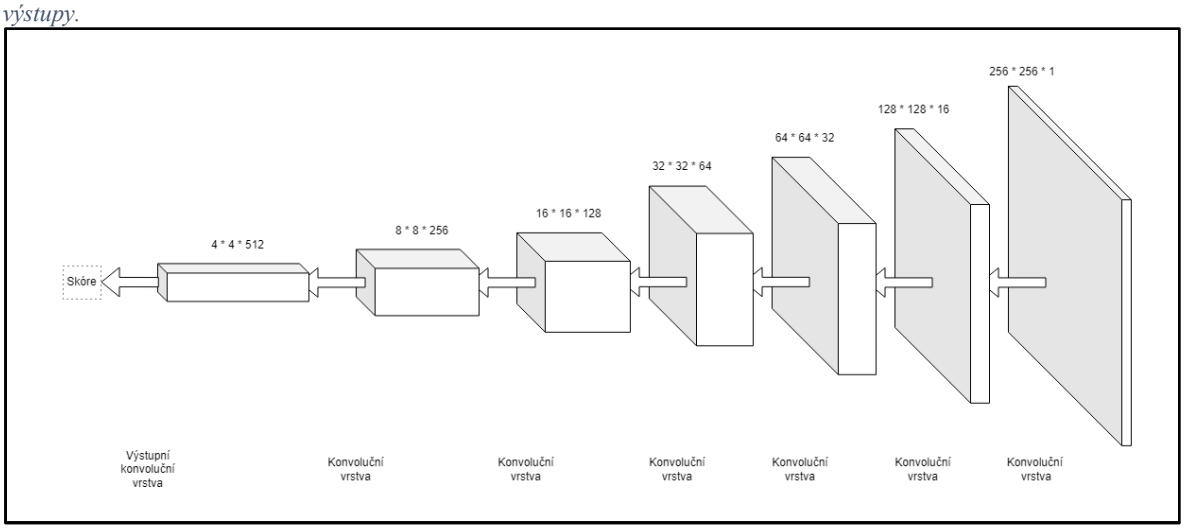
Obrázek 11 Graf architektury generátoru. Šipky reprezentují konvoluční vrstvy a kvádry reprezentují vstupy, resp. výstupy



Zdroj: Vlastní zpracování

Kritik (viz obrázek 12) provádí extrakci rysů snímku pomocí redukce vstupního snímku při průchodu konvolučními vrstvami. Výstupem kritika je, na rozdíl od vstupu generátoru, pouze skalární veličina hodnota udávající "realističnost" vstupního obrázku, přičemž vyšší hodnoty označují obrázky, které jsou kritikem považovány za realističtější.

Obrázek 12 Graf architektury kritika. Šipky reprezentují konvoluční vrstvy a kvádry reprezentují vstupy, resp. výstupy.



Zdroj: Vlastní zpracování

Výstupem kritika je, na rozdíl od vstupu generátoru, pouze skalární veličina hodnota udávající "realističnost" vstupního obrázku, přičemž vyšší hodnoty označují obrázky, které jsou kritikem považovány za realističtější.

4.4 Implementace modelu

Dle návrhu architektury jsou implementovány dva modely; generátor a kritik. Jedná se o konvoluční neuronové sítě, které jsou pro modely GAN typické. Implementace GAN není však striktně závislá na konvolučních sítích a lze použít i jiné typy sítí, jako jsou lineární sítě či transformátory.

Jako optimalizátor je pro obě sítě použit Adam s mírou učení 0.0001 a betami 0.5, respektive 0.9.

4.4.1 Generátor

První implementovanou neuronovou sítí je generátor. Třída Generator je určena k vytváření obrazových snímků v rozlišení 256*256 pixelů v černobílém spektru. V podstatě převádí náhodný šum na strukturovaná data podobná realistickým snímkům hrudníku. Tato třída je implementována pomocí třídy nn.Module, základní třídy pro všechny moduly neuronových sítí ve frameworku PyTorch. Tato třída implementuje tři následující metody.

Statická metoda *gen_block* skládá dohromady konvoluční vrstvu, dávkovou normalizaci, a aktivační vrstvou. Pro konvoluční vrstvu je použit nn.ConvTranspose2d, která aplikuje transponovanou konvoluci na vstupní obraz. Pro normalizaci a aktivační vrstvu jsou použity běžné vrstvy nn.BatchNorm2d, respektive nn.ReLU.

Konstruktor je inicializován s parametry pro dimenzionalitu latentního prostoru (z_dim). Základní dimenzionalita konvolučních vrstev je 16. Architektura této sítě, vytvořená pomocí nn.Sequential, začíná od vektoru šumu a přes řadu vrstev se postupně vytváří větší zobrazení 2D obrazu, až dosáhne velikosti 256*256 pixelů. Poslední aktivační funkcí je nn.Sigmoid.

Neuronová síť je logicky rozvržena do tří částí. Vstupní část transformuje latentní vektor do malé prostorové dimenze, čímž připravuje tenzor pro následné převzorkování. Následují upsampling bloky, definované pomocí *gen_block*, které postupně zvětšují generovaný snímek. Poslední částí je převod převzorkovaného tenzoru do výsledné velikosti

256*256 a hloubky 1 dimenze. Pomocí sigmoidní aktivační funkce je pak výstup normalizován tak, aby představoval hodnoty intenzity bílé barvy, čímž se dokončí generování syntetického obrazu.

Metoda forward definuje zpracování vektoru šumu na generovaný obraz jako průchod latentního vektoru skrze definovanou konvoluční neuronovou síť. Tato metoda musí být implementována pro každou podtřídu třídy nn.Module.

```
class Generator(nn.Module):


    @staticmethod
    def gen_block(input: int, output: int) -> nn.Sequential:
        return nn.Sequential(
            nn.ConvTranspose2d(input, output, 4, 2, 1),
            nn.BatchNorm2d(output),
            nn.ReLU(True),)


    def __init__(self, z_dim:int, d_dim:int=16):
        super(Generator, self).__init__()
        self.z_dim=z_dim

        self.gen = nn.Sequential(

            # Initial block - size 4x4, channels 1024
            nn.ConvTranspose2d(z_dim, d_dim * 64, 4, 1, 0),
            nn.BatchNorm2d(d_dim * 64),
            nn.ReLU(True),

            self.gen_block(d_dim * 64, d_dim * 32), # s 8, ch 512
            self.gen_block(d_dim * 32, d_dim * 16), # s 16, ch 256
            self.gen_block(d_dim * 16, d_dim * 8), # s 32, ch 128
            self.gen_block(d_dim * 8, d_dim * 4), # s 64, ch 64
            self.gen_block(d_dim * 4, d_dim * 2), # s 128, ch 32

            # Final block - size 256x256, channels 1
            nn.ConvTranspose2d(d_dim*2, 1, 4, 2, 1),
            nn.Sigmoid(),
        )


    def forward(self, noise):
        x = noise.view(len(noise), self.z_dim, 1,1)
        return self.gen(x)
```

4.4.2 Kritik

Třída Critic také dědí z třídy nn.Module, základní třídy frameworku PyTorch pro všechny moduly neuronových sítí. Funkcí kritika je vyhodnocení přesvědčivosti snímku. Výstupem kritika je průměrné skóre pro generované a pro reálné snímky. Vzdálenost těchto průměrů lze vnímat jako přesvědčivost generovaných snímků; čím je bližší, tím věrohodnější jsou pro kritika tyto snímky. Třída kritika definuje tři následující metody:

Konstruktor inicializuje model kritika pomocí sekvenčního kontejneru nn.Sequential. Přebírá nepovinný parametr d_dim s výchozí hodnotou 16, který nastavuje základní dimenzionalitu pro konvoluční vrstvy. Konstruktor obsahuje konvoluční neuronovou síť, která postupně redukuje snímek na výsledný skalár.

Statická metoda *crit_block* definuje základní blok neuronové sítě kritika. Každý blok se skládá z 2D konvoluční vrstvy nn.Conv2d, vrstvy normalizace nn.InstanceNorm2d a aktivace leaky ReLU nn.LeakyReLU. Tento návrh vystihuje běžný vzorec v CNN, kde konvoluční vrstvy extrahují rysy z obrázků, normalizační vrstvy stabilizují učení a leaky ReLU zavádí nelinearitu a zároveň umožňuje malý gradient pro záporné hodnoty. Jelikož se jedná o model WGAN-GP, jako normalizační vrstva je použit nn.InstanceNorm2d místo běžného nn.BatchNorm2d. Hodnota 0.2 záporného sklonu pro leaky ReLU byla vybrána jako obvyklá pro tento typ sítě.

Metoda forward definuje výpočet prováděný při každém volání této třídy. Tato metoda musí být definována pro každou podtřídu třídy nn.Module. Tato metoda určuje, jak vstupní obraz prochází modelem. Prochází obrazem přes sekvenční kontejner (který provádí vrstvy v pořadí) a nakonec přetvoří výstup na dvourozměrný tenzor, kde každý řádek odpovídá skalární hodnotě pro každý obraz v dávce.

Tato implementace kritika v modelu WGAN tedy používá konvoluční vrstvy, instanční normalizaci a aktivační funkci leaky ReLU. Tento kritik je nezávislý na délce latentního vektoru, lze ho tudíž použít ve všech experimentech se snímky v rozlišení 256*256 pixelů.

```
class Critic(nn.Module):

    @staticmethod
    def crit_block(input: int, output: int) -> nn.Sequential:
        return nn.Sequential(
                nn.Conv2d(input, output, 4, 2, 1),
                nn.InstanceNorm2d(output),
                nn.LeakyReLU(0.2),
        )

    def __init__(self, d_dim:int=16):
        super(Critic, self).__init__()

        self.crit = nn.Sequential(


            self.crit_block(1, d_dim),
            self.crit_block(d_dim, d_dim*2),
            self.crit_block(d_dim*2, d_dim*4),
            self.crit_block(d_dim*4, d_dim*8),
            self.crit_block(d_dim*8, d_dim*16),
            self.crit_block(d_dim*16, d_dim*32),


            # Final layer
            nn.Conv2d(d_dim*32, 1, 4, 1, 0), # 1*1 (ch: 512 -> 1)

        )

    def forward(self, image):
        crit_pred = self.crit(image)
        return crit_pred.view(len(crit_pred), -1)
```

4.5 Implementace trénovacího procesu

4.5.1 Gradientní sankce

Důležitou části modelu WGAN-GP je gradientní sankce, která zajišťuje 1 – Lipschitzovo pravidlo penalizací normy gradientu výstupu kritika vzhledem k jeho vstupu. Je implementována pomocí funkce *get_gp*. Tato funkce nejprve vytvoří interpolované snímky z reálných a generovaných snímků, které jsou vyhodnoceny kritikem. Následně je vypočítán součet gradientů výstupů kritika vzhledem k interpolovaným snímkům. K tomuto gradientu je získán kolmý vektor, který je následně použit při samotném výpočtu gradientní sankce dle literatury.

```
def get_gp(real, fake, crit, alpha, gamma=10):
    # Creation of interpolated images
    mix_images = real * alpha + fake * (1-alpha)
    mix_scores = crit(mix_images)

    # Computation of the critic gradient
    gradient = torch.autograd.grad(
        inputs = mix_images,
        outputs = mix_scores,
        grad_outputs=torch.ones_like(mix_scores),
        retain_graph=True,
        create_graph=True,
    )[0]

    # Computation of the gradient penalty
    gradient = gradient.view(len(gradient), -1)
    gradient_norm = gradient.norm(2, dim=1)
    gp = gamma * ((gradient_norm -1) ** 2).mean()

    return gp
```

4.5.2 Třída DataLoader

Další důležitou částí implementace modelů neuronových sítí ve frameworku PyTorch je třída DataLoader. DataLoader je vytvořen tak, aby zvládl předpřipravení a následné odeslání dávky trénovacích dat. Vlastní třída Dataset, podtřída třídy Dataset systému PyTorch, je přizpůsobena specifické struktuře naší sady dat. Inicializuje se přijetím cesty k adresáři a velikosti obrázku, která je ve výchozím nastavení 256 pixelů, čímž se nastaví rozměry načítaných obrázků. V rámci inicializace třída iteruje zadaný adresář a sestaví seznam cest k souborům snímků (položek) a jejich příslušných štítků. Tato konstrukce tvoří hlavní část procesu zpracování dat. Labelem snímku je název souboru; jelikož všechna data pocházející z DataLoader jsou data reálná, není nutné k nim explicitně přiřazovat štítek potvrzující jejich reálnost.

4.5.3 Třída Dataset

Třída Dataset je rozšířena o metodu *__getitem__*, která slouží jako pracovní nástroj pro načítání, zpracování a normalizaci dat. Pro každou dávku dat načtenou třídou DataLoader se pro každý index zavolá metoda *__getitem__*, která načte obrázek ve stupních šedi prostřednictvím knihovny PIL a převede jej na pole knihovny numpy typu float32. Pole je rozšířeno o rozměr kanálu (v případě snímku se stupni šedi se jedná o jeden kanál), a následně je převedeno na tenzor knihovny PyTorch. Tenzor je normalizován z hodnot pixelů

0–255 do rozmezí 0–1. Prostřednictvím těchto procesů třída Dataset nejen zvyšuje efektivitu trénování optimalizací využití paměti a výpočetních zdrojů, ale také zajišťuje standardizaci vstupních dat, což je rozhodující pro stabilitu a konvergenci modelu WGAN-GP během trénování.

```
class Dataset(Dataset):
    def __init__(self, path, size=256):
        self.sizes = (size, size)
        items, labels = [], []
        for data in os.listdir(path):
            item = os.path.join(path, data)
            items.append(item)
            labels.append(data)
        self.items = items
        self.labels = labels

    def __len__(self):
        return len(self.items)

    def __getitem__(self, idx):
        # Load the image as grayscale
        data = PIL.Image.open(self.items[idx]).convert("L")
        # Convert the PIL image to a numpy array
        data = np.array(data, dtype=np.float32)
        # Add an extra dimension to represent the single channel
        data = np.expand_dims(data, axis=0)
         # Normalize the pixel values from [0, 255] to [0, 1]
        data = torch.from_numpy(data).div(255)
        return data, self.labels[idx]
```

4.5.4 Trénovací smyčka

Trénování probíhá ve smyčce v definovaném počtu epoch (50), a v každé epoše se iteruje trénovací datovou sadou poskytovanou třídou DataLoader. Pro sledování průběhu tréninku je použita knihovna tqdm, která poskytuje ukazatel průběhu pro vizuální zpětnou vazbu. Jelikož velikost trénovacího datového souboru není dělitelná velikostí dávky, je během každé iterace velikost dávky dynamicky určována podle počtu skutečných obrazů v aktuální dávce. Velikost dávky je pak použita pro generování šumu, aby bylo vygenerováno stejné množství generovaných snímků, jako je snímků reálných.

```
for epoch in range(n_epoch):
    for real, _ in tqdm(dataloader):
        cur_bs= len(real)
        real=real.to(device)

        ### CRITIC TRAINING
        mean_crit_loss = 0
        for _ in range(crit_cycles):
            crit_opt.zero_grad()

            noise=generate_noise(cur_bs, z_dim)
            fake = gen(noise)
            crit_fake_pred = crit(fake.detach())
            crit_real_pred = crit(real)

            alpha=torch.rand(len(real),1,1,1,device=device,
requires_grad=True)
            gp = get_gp(real, fake.detach(), crit, alpha)

            crit_loss = crit_fake_pred.mean() -
crit_real_pred.mean() + gp

            mean_crit_loss+=crit_loss.item() / crit_cycles

            crit_loss.backward(retain_graph=True)
            crit_opt.step()

        crit_losses+=[mean_crit_loss]

        ### GENERATOR TRAINING
        gen_opt.zero_grad()
        noise = generate_noise(cur_bs, z_dim)
        fake = gen(noise)
        crit_fake_pred = crit(fake)

        gen_loss = -crit_fake_pred.mean()
        gen_loss.backward()
        gen_opt.step()
```

Trénink kritika je pro jedno trénování generátoru prováděn vícekrát. Toto opakování je jedním z hyperparametrů a je reprezentováno proměnnou *crit_cycles*. Pro trénink tohoto modelu bylo zvoleno pět trénování kritika na jedno trénování generátoru. Tento poměr je pro trénink WGAN-GP běžný, a slouží k zajištění dostatečné rychlosti učení kritika ve srovnání s generátorem, kterému tak může poskytovat smysluplné gradienty. V každém cyklu aktualizace kritika se gradienty optimalizátoru vynulují, což je nezbytný krok, aby se

zabránilo hromadění gradientů z předchozích iterací. Ztráta kritika se vypočítá pomocí rozdílu mezi jeho předpověďmi na skutečných a falešných datech spolu s gradientní sankcí (gp), která vynucuje 1 – Lipschitzovo omezení. Průměrná ztráta kritika je sledována napříč cykly kritika pro pozdější analýzu, což poskytuje přehled o výkonnosti kritika v čase.

Jakmile je kritik dotrénován, je trénována síť generátoru. Trénink generátoru spočívá v generování falešných snímků za použití vektoru šumu. Tyto snímky jsou následně vyhodnocovány kritikem. Ztráta generátoru je záporný průměr předpovědí kritika na dávce generovaných snímků, což generátor vede k vytváření obrazů, které kritik vyhodnotí jako skutečné. Generátor je pak optimalizován pomocí algoritmu zpětného šíření chyby a optimalizátor provede úpravu vah generátoru. Tréninková smyčka implementuje soupeření mezi sítěmi, kde se kritik a generátor iterativně zlepšují v reakci na vzájemné úpravy.

V průběhu tréninkového procesu se ukládají ztráty pro kritika i generátoru pro sledování a analýzu výsledků tréninku. Sledování těchto ztrát je velmi důležité, protože poskytují přehled o konvergenci modelu. V případě WGAN-GP se očekává, že ztráta kritika bude konvergovat kolem nuly, zatímco ztráta generátoru se bude postupně snižovat, což naznačuje úspěšné učení a syntézu nových snímků, které mají stejné pravděpodobnostní rozložení jako snímky reálné.

4.5.5 Implementace metrik

Jak metrika IS, tak metrika FID jsou v principu použití předem natrénovaného modelu neuronové sítě k zjištění kvality trénované sítě, konkrétně generátoru. Proto je potřeba připravit prostředí na použití této techniky. Oba předem trénované modely pracují se standardními obrázky ve třech dimenzích, proto je potřeba pro ně vytvořit třídimenzionální snímek z jednodimenzionálního snímku, který je generován generátorem.

```
from ignite.metrics import FID, InceptionScore
import torchvision.transforms as transforms
from ignite.engine import Engine

def convert_three_chan(it):
    tensor_three_chan = torch.cat((it, it, it), dim=0)
    return tensor_three_chan

def interpolate(batch):
    arr = []
    for img in batch:
        img_pil = transforms.ToPILImage()(img)
        img_tensor = transforms.ToTensor()(img_pil)
        three_chan_img = convert_three_chan(img_tensor)
        arr.append(three_chan_img )
    return torch.stack(arr)

def evaluation_step(engine, batch):
    with torch.no_grad():
        noise = generate_noise(batch_size, z_dim)
        gen.eval()
        fake_batch = gen(noise)
        fake = interpolate(fake_batch)
        real = interpolate(batch[0])
        return fake, real

fid_metric = FID(device=device)
is_metric = InceptionScore(device=device, output_transform=lambda x:
x[0])

evaluator = Engine(evaluation_step)
fid_metric.attach(evaluator, "fid")
is_metric.attach(evaluator, "is")
```

4.6 Trénování modelu

4.6.1 Vývojové prostředí

Trénování modelů bylo uskutečněno pomocí frameworku Pytorch, verze 2.1.2. Popis systému lze nalézt v následující tabulce.

Tabulka 1 *Hardware a operační systém použitý při trénování modelů*

CPU	AMD Ryzen 9 5900X
GPU	Nvidia RTX 3080 Ti
Systémová paměť (Instalovaná)	64 GB
Systém	Windows 10, WSL2 Ubuntu 20.04

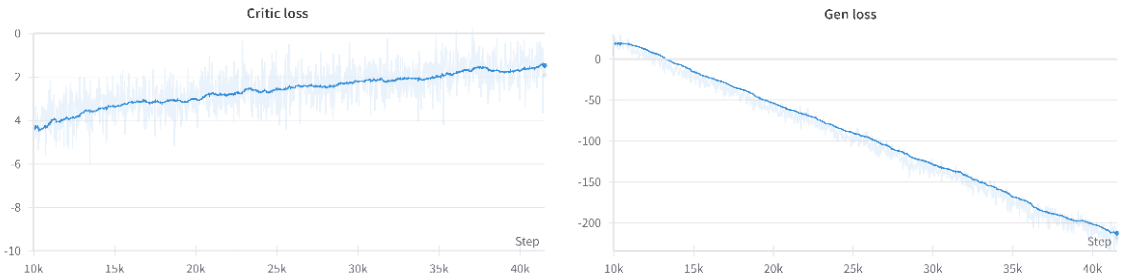
Zdroj: Vlastní zpracování

Trénovací prostředí je realizováno pomocí jupyter, s jádrem využívajícím python verze 3.8.10. Pro sledování výsledků je využito nástroje Weights & Biases.

4.6.2 Trénink tří modelů pro následné vyhodnocení

4.6.2.1 Model 128

Graf 1 *Graf ztráty pro kritika a generátor modelu 128 od kroku 10000*



Zdroj: Vlastní zpracování

Pro popis průběhu tréninku jsou typicky používány tyto dva grafy, které představují ztráty generátoru a kritika v jednotlivých krocích tréninku. Grafy začínají na kroku 10000, jelikož před touto hranicí vykazuje ztráta kritika velké hodnoty, které zastiňují následující trendy.

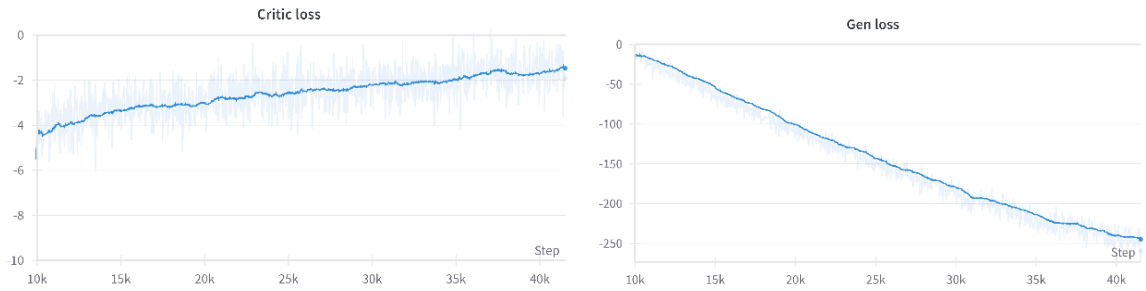
První graf zobrazuje ztrátu kritika, která v průběhu trénování osciluje, jelikož se kritik přizpůsobuje zlepšujícímu se generátoru. Ztráta kritika pomalu nabývá nižší absolutní hodnoty, což značí, že se citlivost kritika ke generovaným snímkům snižuje. Toto chování je typické pro modely WGAN-GP, protože úkolem kritika je poskytnout generátoru robustní gradient, který může použít při svých aktualizacích. Vzestupný trend nemusí nutně znamenat selhání učení, ale spíše odráží probíhající soupeření mezi generátorem a kritikem, což je základní charakteristikou modelů GAN.

Ztráta generátoru je pouze opačné číslo k průměrnému číslu přiřazovanému vygenerovaným snímkům. Kritik se snaží přiřazovat reálným snímkům číslo nižší než

generovaným snímkům. Přiřazené číslo ale záleží jak na schopnostech kritika, tak na schopnostech generátoru čili z těchto hodnot nelze činit definitivní závěry.

4.6.2.2 Model 64

Graf 2 *Graf ztráty pro kritika a generátor modelu 64 od kroku 10000.*

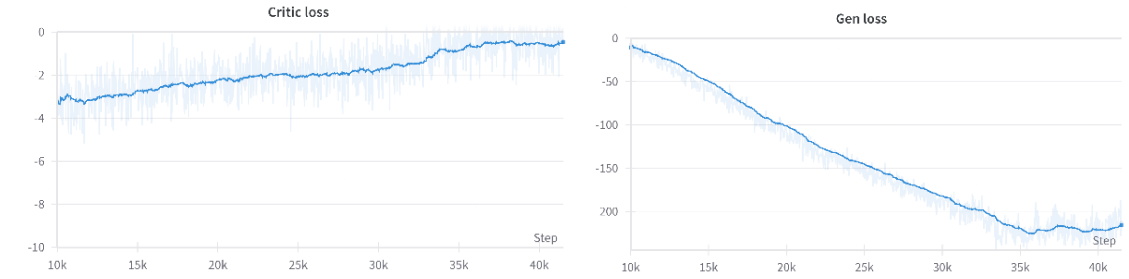


Zdroj: Vlastní zpracování

Graf ztráty kritika je pro model 64 velmi podobný grafu modelu 128. To svědčí, že modely se efektivně učí a kritik je schopen efektivně řídit trénink generátoru. Vzhledem k podobnosti těchto grafů lze předpokládat, že i výsledky těchto modelů budou podobné.

4.6.2.3 Model 32

Graf 3 *Graf ztráty pro kritika a generátor modelu 64 od kroku 10000*



Zdroj: Vlastní zpracování

Model 32 se liší od ostatních modelů tím, že ztráta kritika dosahuje mnohem menších hodnot. To znamená, že kritik má problém rozlišit generované snímky od snímků skutečných. To je cílem tréninku GAN, což zní jako pozitivní zjištění, avšak po takto nízkém počtu generací to spíše značí problém v trénování. Kritik pak není schopný poskytovat dostatečný gradient, což se negativně promítne ve schopnosti učení se pro generátor. Lze tedy očekávat horší výsledky než u předchozích dvou modelů.

4.6.3 Volba modelu pro další trénování

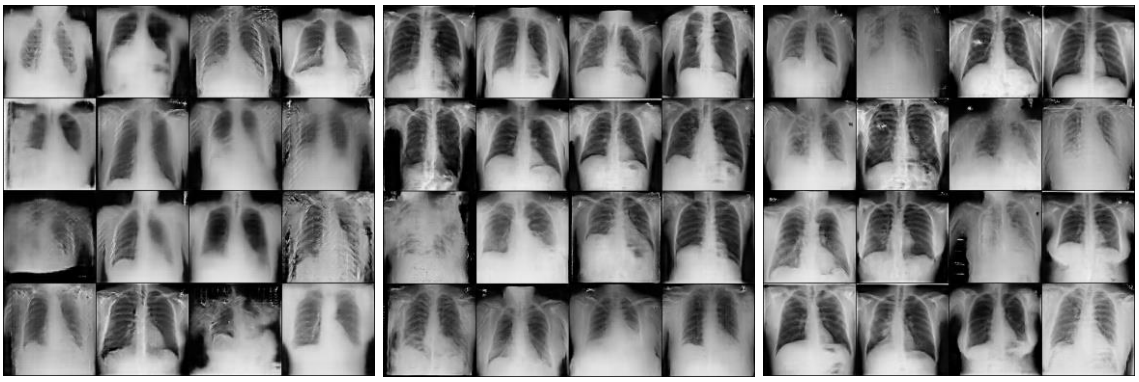
Tabulka 2 Skóre modelů po 50 trénovacích epochách. Nejlepší výsledek je zvýrazněn kurzívou.

MODEL	FID	IS
128	0,11172	3,40974
64	*0,08267*	3,55129
32	0,12306	*4,28219*

Zdroj: Vlastní zpracování

Hodnoty FID jsou velice nízké, a i hodnoty IS jsou nižší, což je očekávaný stav, jelikož reálný datový soubor nemá vysokou rozmanitost. Nejlepšího výsledku FID dosáhl model s velikostí latentního prostoru 64. Nejlepšího skóre IS dosáhl model s velikostí latentního prostoru 32. Model s velikostí latentního prostoru 128 se zřejmě učí pomaleji, jak lze pozorovat z hodnoty ztráty generátoru, a jak potvrzuje jeho FID skóre.

Obrázek 13 Vygenerované snímky po 50 epochách učení. Zleva: model 32, model 64, model 128.



Zdroj: Vlastní zpracování

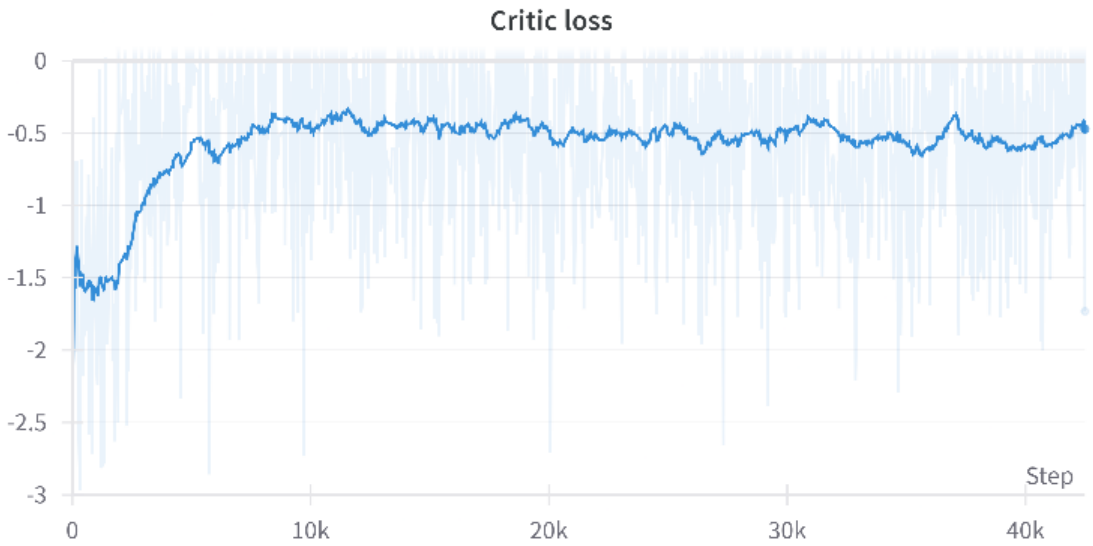
Z porovnání vybraných snímků, viditelných na obrázku 13, lze odhadnout, že model s velikostí latentního prostoru 32 má problémy se zachycením rozdělení trénovacích snímků. Modely 64 a 128 produkují snímky, které ve zmenšené podobě vypadají relativně věrohodně, avšak při přiblížení do plného rozlišení lze jednoduše poznat, že snímky postrádají detaily. Je však nutné podotknout, že vzorek 16 snímků je sám o sobě moc malý, proto je nutné pro volbu nejlepšího modelu využít objektivní metriky. Dle nejlepšího FID skóre byl pro pokračování vybrán model s velikostí latentního vektoru 64.

4.6.4 Trénování vybraného modelu

Zvolený model s velikostí latentního prostoru byl za účelem zlepšení generovaných snímků trénován po dalších 150 epoch. Pro toto trénování bylo přidáno i průběžné měření FID a IS

jednou za 4300 kroků. Trénink byl kvůli dlouhé době trénování, přibližně 12 hodin na 50 epoch, prováděn postupně po 50 epochách.

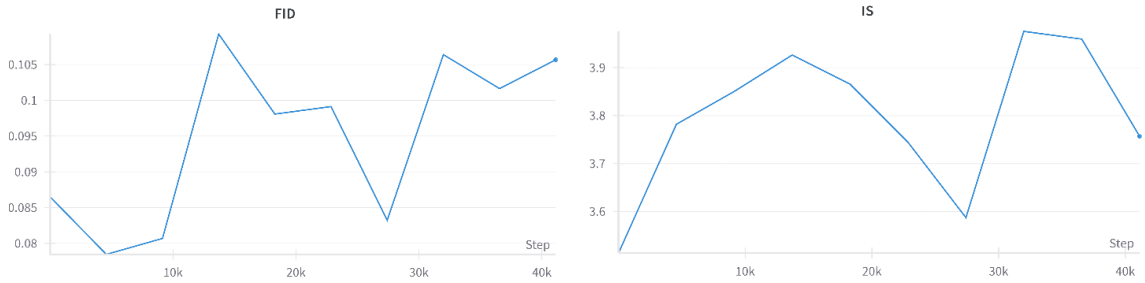
Graf 4 *Graf ztráty kritika pro epochy 51–100*



Zdroj: Vlastní zpracování

Z grafu je patrný strmý pokles schopnosti kritika rozlišovat mezi reálnými a generovanými snímky. Kritik tak pravděpodobně neposkytuje užitečný gradient pro další trénování, což se může projevit stagnací, či zhoršením výkonnosti modelu.

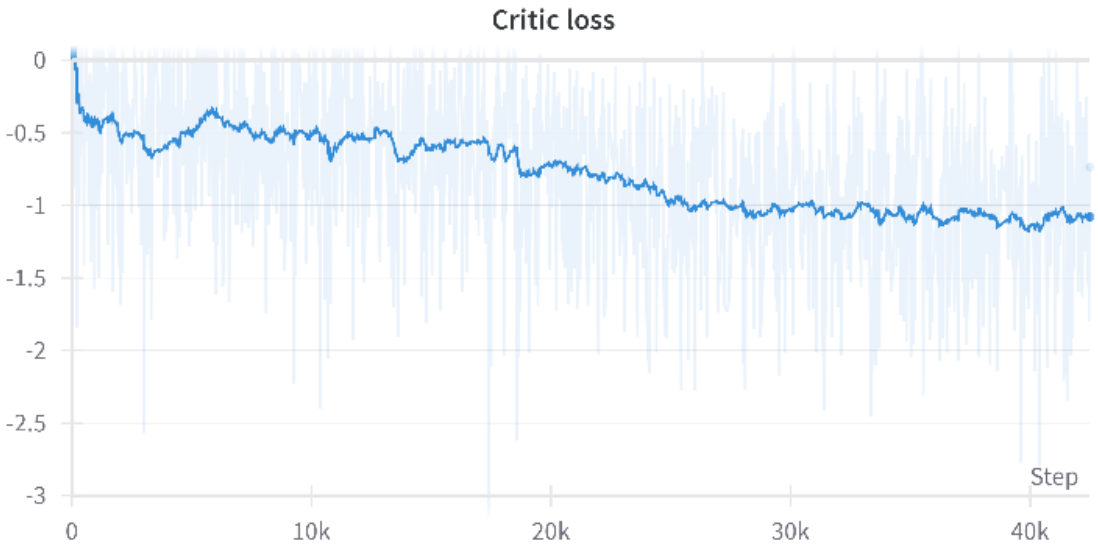
Graf 5 *Graf IS a FID skóre pro epochy 51–100*



Zdroj: Vlastní zpracování

Skóre FID pak částečně potvrzuje tento trend, kdy na počátku 51. epochy je výsledek lepší než na konci tréninku. IS skóre fluktuuje mezi 3,5 a 4, což neznačí žádné potíže jako například kolaps módu. Po průzkumu náhodných generovaných snímků lze tvrdit, že snímky vykazují patrnou ztrátu detailu.

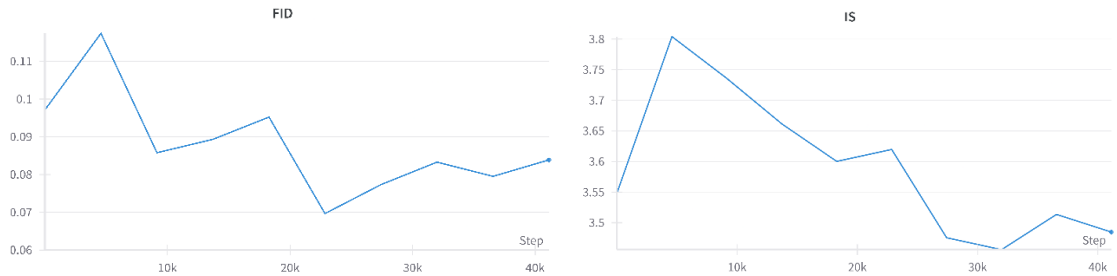
Graf 6 *Graf ztráty kritika pro epochy 101–150*



Zdroj: Vlastní zpracování

Z grafu ztráty kritika pro epoch 101–150 je patrný postupný pokles, tj. zvětšení rozdílu mezi hodnotami přiřazovanými reálným a generovaným snímkům. To značí lepší schopnost kritika rozlišovat mezi reálnými a generovanými snímky.

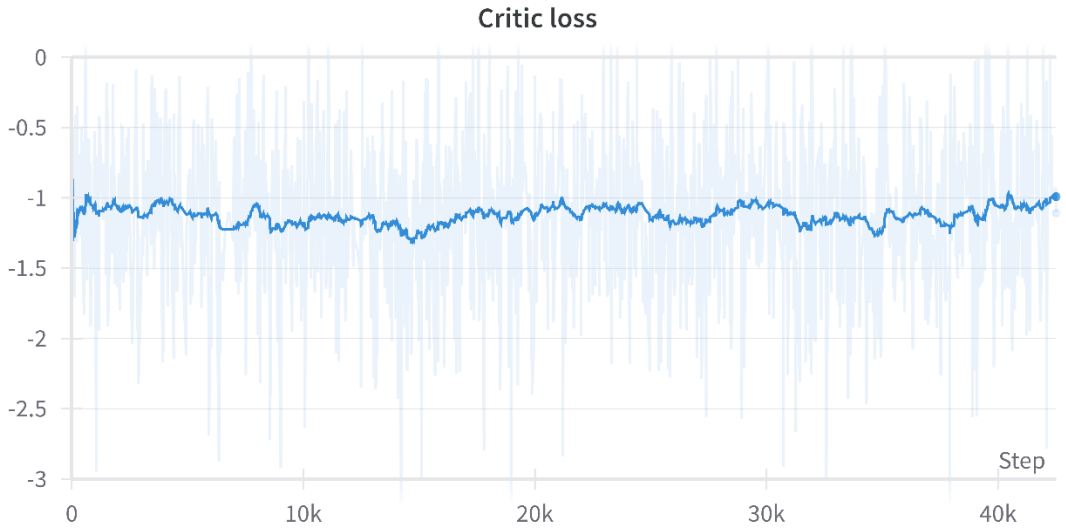
Graf 7 *Graf ztráty kritika pro epochy 101–150*



Zdroj: Vlastní zpracování

FID tento trend potvrzuje postupným zlepšováním. Skóre FID začíná výše a vykazuje výkyvy, ale v průběhu času je zřejmý klesající trend, což naznačuje postupné a konzistentní zlepšování kvality generovaných snímků. S rostoucím počtem kroků se trend stabilizuje, což odráží učení generátoru a jeho přizpůsobení se generování snímků, které se více podobají trénovací sadě dat. Hodnota IS ale klesá, což značí menší rozmanitost generovaných snímků.

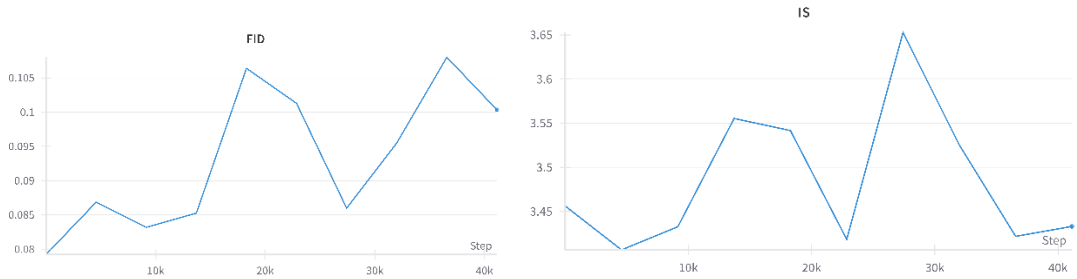
Graf 8 *Graf ztráty kritika pro epochy 151–200*



Zdroj: Vlastní zpracování

Graf ztráty kritika se v průběhu tréninku epoch 151–200 jeví jako relativně konstantní. Stabilní horizontální trend vyhlazené přímky naznačuje, že změna výkonnosti kritika a generátoru v jejich hře postupuje pro obě sítě stejným tempem. Nevyhlazené údaje o ztrátách, znázorněné světlejší, proměnlivější linkou pozadí, vykazují očekávanou variabilitu v jednotlivých hodnoceních ztrát. Tato konzistence by mohla znamenat, že kritik a generátor jsou ve stavu rovnováhy, kdy se kritik nezlepšuje ani nezhoršuje ve své schopnosti rozlišovat mezi skutečnými a falešnými obrázky. Absence výrazného klesajícího trendu v této fázi by mohla naznačovat, že schopnost kritika rozlišovat vyspěla, za předpokladu, že se ustálila i kvalita generátoru při tvorbě obrázků. Budoucí epochy by zvýšení počtů tréninků kritika na trénink generátoru, aby se urychlilo jeho další zlepšení rozlišovací schopnosti.

Graf 9 *Graf IS a FID skóre pro epochy 101-150*

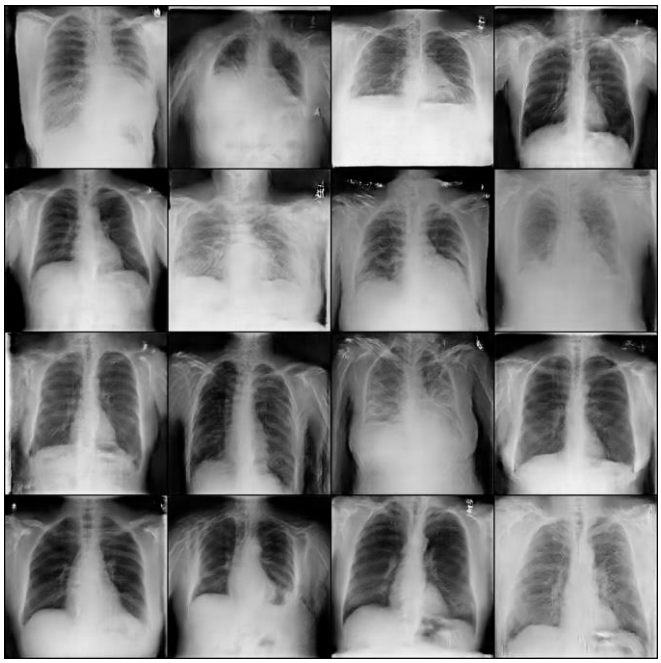


Zdroj: Vlastní zpracování

Graf skóre FID ukazuje postupný nárůst, který naznačuje zhoršení schopnosti generátoru generovat snímky podobné trénovacímu rozdělení. Trend grafu ale není striktně lineární a skóre FID nemusí nutně značit horší kvalitu snímků.

Pro IS lze sledovat vrchol kolem 30 tisíc kroků, který může odpovídat obzvláště rozmanité nebo kvalitní dávce snímků. Poté však dochází k prudkému poklesu, což naznačuje následný pokles rozmanitosti nebo kvality. Podobně jako u grafu FID končí IS na 40 tis. krocích níže než v některých dřívějších bodech, což může naznačovat, že model v vytváří méně rozmanité snímky.

Obrázek 14 *Generované snímky po 200 epochách tréninku*



Zdroj: Vlastní zpracování

První náhled na vygenerované snímky na obrázku 14 vykazuje výrazné zlepšení detailu oproti schopnostem modelu po 50 generacích. Toto kvalitativní zlepšení je patrné zejména v posledních epochách, kdy generované obrazy vykazují zvýšenou jasnost, ostřejší rysy a definovanější zobrazení anatomických struktur. Přestože grafy FID a IS vykazují určitou míru fluktuace, která zpočátku naznačovala variabilitu výkonu generativního modelu, vizuální kontrola odhaluje postupné zpřesňování obrazů. Jemnější detaily a lepší texturní nuance v obrazech odrážejí zrání generativního procesu. Tato zlepšení v detailech naznačují, že model se účinně učí zachycovat a replikovat složitosti cílového rozložení dat, což zvyšuje realističnost a užitečnost generovaných obrazů pro potenciální aplikace v analýze lékařských obrazů i mimo ni.

5 Výsledky a diskuse

5.1 Kvalitativní hodnocení generovaných snímků

5.1.1 Ukázky vygenerovaných snímků

Účelem modelů GAN je vytváření realistických syntetických dat, které odpovídají pravděpodobnostnímu rozdělení dat trénovacích. Tato kapitola obsahuje prezentaci vybraných snímků syntetizovaných výsledným natrénovaným modelem za účelem kvalitativního posouzení výkonu modelu, přítomnosti anatomických struktur a úrovně detailu výsledných generovaných snímků.

Obrázek 15 Generovaný snímek hrudníku

Zdroj: Vlastní zpracování

Na prvním snímku je patrné, že model má schopnost vygenerovat snímek podobný skutečnému radiografickému vzhledu lidského hrudníku. Lze pozorovat úspěšné zachycení základních anatomických struktur rentgenového snímku hrudníku. Na snímku lze rozeznat plíce, klíční kosti, kosti pažní, žebra, či aortu. Plíce jsou zřetelně symetrické a kontrastní vůči zbytku snímku. Dále je možné sledovat náznaky srdečního stínu či rozeznat krční páteř, i některé zajímavé detaily jako je přítomnost plynů ve střevě pacienta.

Tento snímek ovšem postrádá rozlišení v oblasti mezihrudí, jako například identifikovatelnou hrudní páteř. Zobrazení struktury plicní tkáně není na úrovni kvalitního rentgenového snímku, kdy generovaný snímek postrádá například rozlišitelné cévy. V některých místech lze nalézt artefakty typicky vznikající v konvolučních sítích, jako je

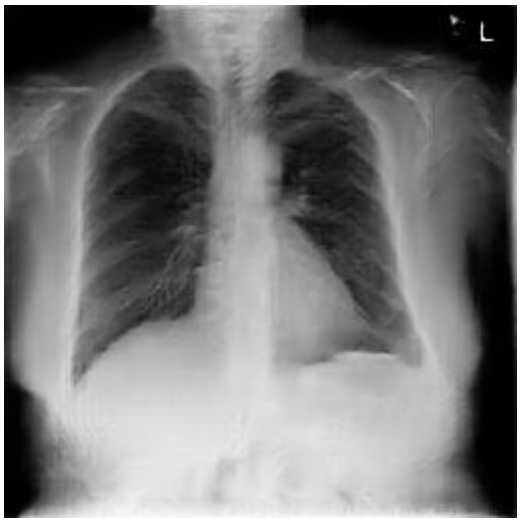
například šachovnicový vzor v podpaží. Celkově lze tvrdit, že snímek postrádá detaily, které lze pozorovat na kvalitním rentgenovém snímku.

Obrázek 16 *Generovaný snímek hrudníku*

Zdroj: Vlastní zpracování

Na druhém snímku lze pozorovat snímek s odlišnou mírou prostupnosti rentgenového záření, což snižuje kontrast v oblasti plic. Nacházejí se zde jasné anatomické struktury jako na předchozím snímku: dva symetrické plicní laloky, viditelná žebra a klíční kosti, srdeční stín. Jsou zde patrné i jasné náznaky plicních cév či páteře. Snímek obsahuje i artefakty, které se nacházejí v trénovacím souboru. Jsou jimi pruhy na stranách snímku (černý pruh v dolní části a bílý pruh v levé části), které v původním souboru vznikly pravděpodobně při digitalizaci analogových snímků. Tento snímek ukazuje, že síť generátoru je schopna alespoň částečně zachytit variabilitu původního souboru. Nicméně, stejně jako u předchozího snímku chybí detailní zachycení plicního parenchymu či jasnější ohraničení kostí.

Obrázek 17 Generovaný snímek hrudníku



Zdroj: Vlastní zpracování

Na tomto snímku se nachází podobně jako na předchozích snímcích zřetelně ohraničené plíce a bránice. Snímek má relativně dobře zachycen srdeční stín a vysoký kontrast v oblasti plic. Generovaný snímek však má také chyby, například chybějící žebra a klíční kosti.

Zajímavostí tohoto snímku je zejména orientační značka L, která se velmi často objevuje v trénovacím souboru dat a značí levou stranu pořízeného snímku. Tyto detaily se v generovaných snímcích v prvních 150 epochách nevyskytovaly, což ukazuje, že přestože hrubý obrys daného objektu je GAN schopná zachytit velmi rychle (po několika jednotkách epoch), učení detailů probíhá pomalu a postupně.

Obrázek 18 Generovaný snímek hrudníku

Zdroj: Vlastní zpracování

Příkladem snímku, na kterém je patrný nález, je výše uvedený snímek. Je zde zřetelný rozdíl v prostupnosti rentgenového záření na obou stranách hrudníku. Pravá strana snímku (levá plíce generovaného pacienta) je tmavá, což u reálných snímků značí prostupnost rentgenového záření, typické pro struktury vyplněné vzduchem. Levá strana snímku (pravá plíce pacienta) je jasná, což naopak svědčí o nízké prostupnosti rentgenového záření, která je atypická pro zdravé plíce. Také zde dochází ke ztrátě normálního obrysu bránice.

Takový nález může být podobný patologiím jako je fluidothorax nebo hemothorax, kdy se nahromadí – v případě fluidothoraxu tekutina, v případě hemothoraxu krev – v prostoru mezi pohrudincí a poplicnicí. Dalšími možnostmi je například rakovina plic, rozedma plic nebo atelaktáza.

Srdeční stín a aorta jsou na snímku patrné, avšak neostře ohraničené. Dobře viditelná jsou žebra, klíční kosti, pažní kosti či páteř. Jedná se tedy, vzhledem k ostatním generovaným snímkům, o kvalitní snímek, který ukazuje, že se síť naučila zobrazovat i vysoce patologické snímky, a nedochází ke kolapsu módu.

Obrázek 19 Generovaný snímek hrudníku

Zdroj: Vlastní zpracování

Poslední diskutovaný snímek má zřetelně viditelné plicní laloky. V obou plicních polích jsou jasně viditelné větvící se vzory, které jsou obdobou průdušek a plicních cév. Tmavší oblasti označují prostory vyplněné vzduchem. Světlejší místa odpovídají cévám a jiným tkáním.

Obě plíce jsou ohraničeny hrudním košem. Jednotlivá žebra jsou dobře viditelné zakřivené útvary. Pod plícemi je viditelná bránice jako horizontální linie s kopulovitým tvarem na každé straně, výraznější na pravé straně vytvořeného pacienta, kde se nachází játra. Dále jsou patrné klíční kosti, pažní kost na levé straně snímku a horní část páteře. Snímek postrádá detail v oblasti břišní dutiny.

Prostředek hrudní dutiny, tzv. mediastinum, je světlejší prostor mezi plícemi. Nachází se v něm srdce, průdušnice a hlavní cévy jako aorta. Srdeční stín a aorta jsou patrné, avšak méně výrazné než na některých předchozích snímcích. Srdeční trn zasahuje do levé strany pacientovy hrudní dutiny, což je normální umístění.

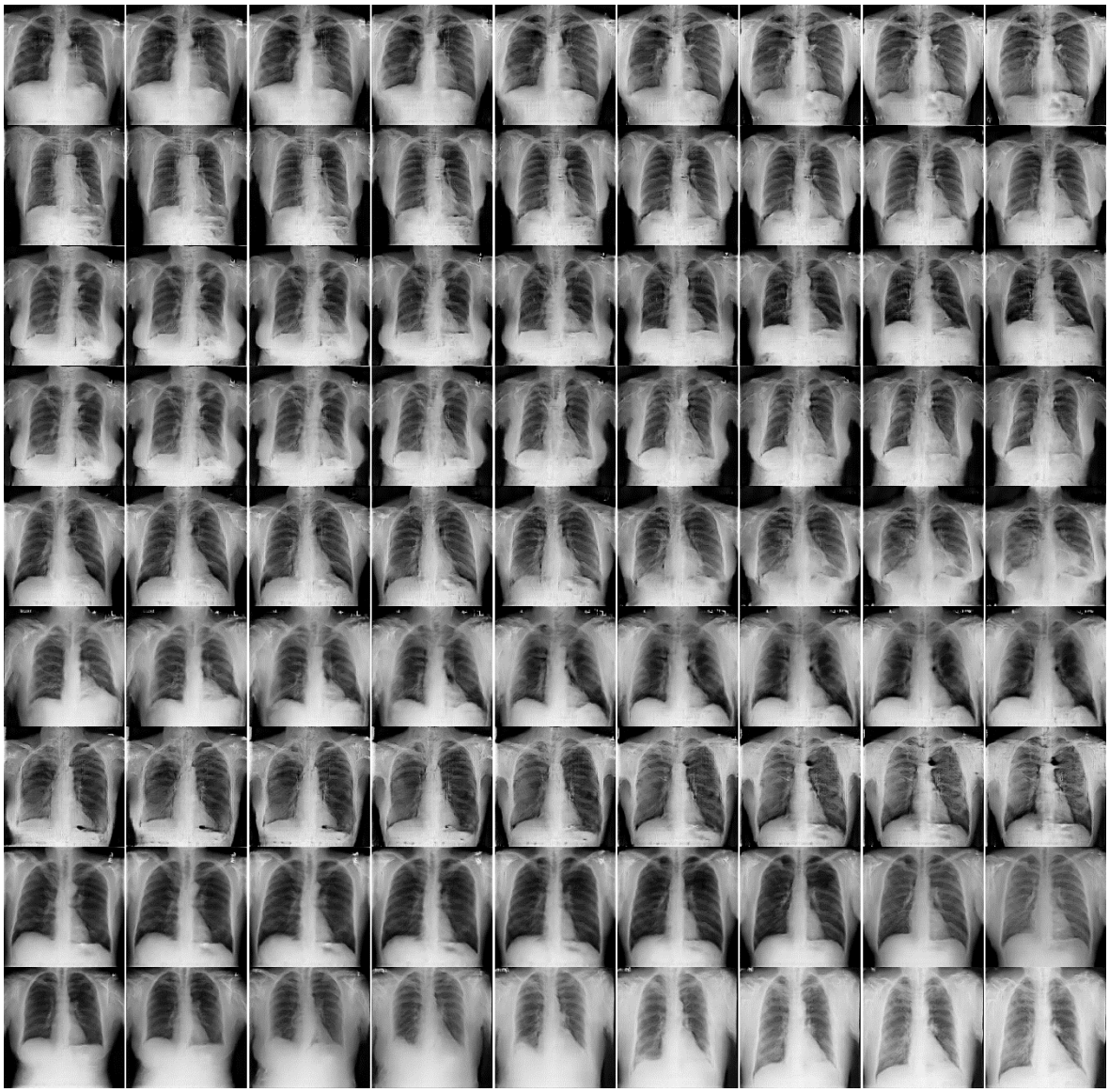
Celková jasnost a detaily obrazu jsou v porovnání s ostatními generovanými snímky poměrně vysoké, umožňující jednoduchou identifikaci výše popsaných anatomických struktur. Míra obrazových artefaktů je ve srovnání s ostatními vytvořenými rentgeny také nízká. Tento snímek také neobsahuje zjevné patologie a reprezentuje tak normativní anatomický základ. Subjektivně byl tento snímek vybrán jako reprezentace nejlepších výsledků této soupeřící generativní sítě.

5.1.2 Lineární interpolace snímků

Generativní modely GAN využívají pro reprezentaci reálných rozdělení svůj latentní prostor. Vstupem do generátoru je vzorek latentního prostoru, který se po průchodu generátorem promění na generovaný snímek. Pokud jsou dány dva různé latentní vektory, měla by síť generátoru poskytovat dvě různé reprezentace – v opačném případě se může jednat o kolaps módu. Platí tedy, že pro vzorky vzniklé lineární interpolací daných dvou latentních vektorů poskytuje síť generátoru smysluplné generované reprezentace.

Jelikož prvky latentního prostoru kódují prvky snímků, postupná lineární interpolace kvalitního modelu zobrazuje postupné morfování mezi danými snímky. Pro kvalitativní posouzení schopností modelu a jeho latentního prostoru je užitečné posouzení schopnosti interpolace snímků. Takové zkoumání má zásadní význam pro vymezení generativních schopností modelu a umožňuje nahlédnout do vnitřních charakteristik dat reprezentovaných v latentních dimenzích. Použití lineární interpolace pro posouzení schopností modelu navíc může sloužit ke zdůraznění nuancí při hodnocení atributů a variací snímků.

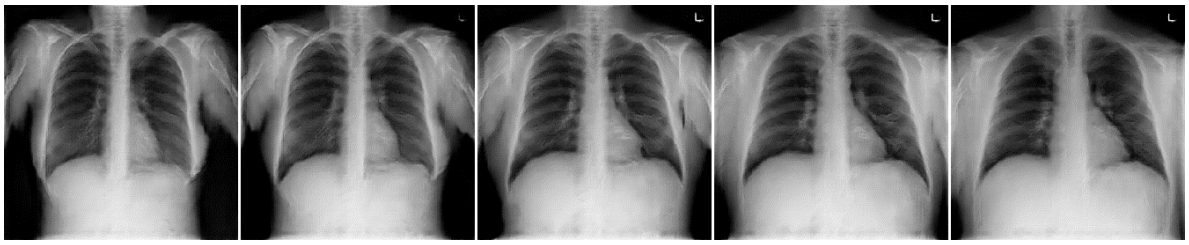
Obrázek 20 Lineární interpolace generovanými snímky hrudníku



Zdroj: Vlastní zpracování

Na obrázku 20 je uvedena praktická demonstrace lineární interpolace. Obrázek zobrazuje sérii snímků generovaných modelem, s plynulým přechodem od jednoho snímku ke snímku druhému prostřednictvím lineární interpolace jejich latentních vektorů. Každý snímek vlevo představuje výstup generátoru generovaný z prvního vzorku latentního prostoru, zatímco obrázek vpravo odpovídá snímku generovanému pomocí druhého latentního vektoru. Přechod mezi všemi zobrazenými snímky je plynulý, což vizualizuje schopnost modelu generovat snímky se smysluplnými reprezentacemi vlastností mezi dvěma danými snímky.

Obrázek 21Detail lineární interpolace generovanými snímky hrudníku



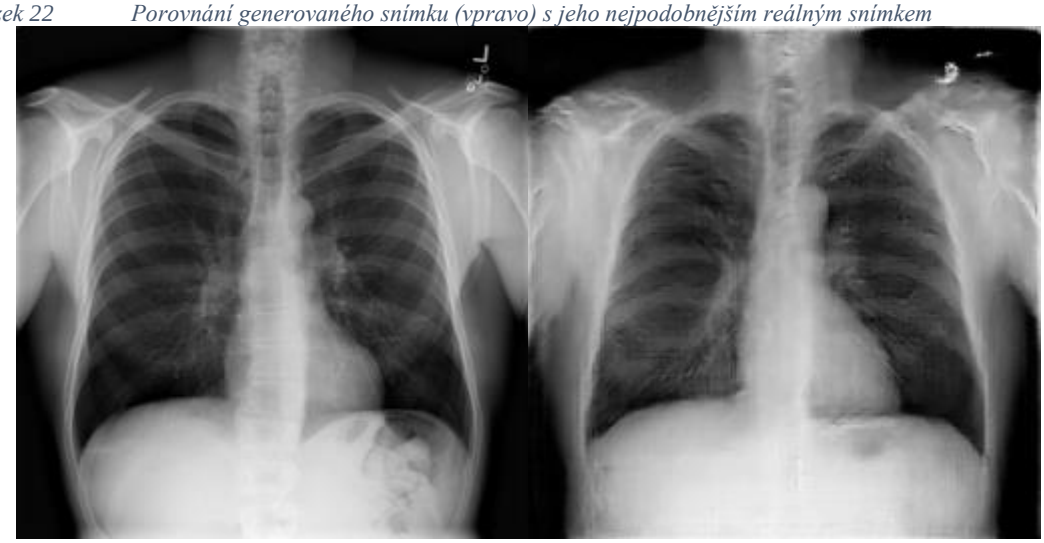
Zdroj: Vlastní zpracování

Pro zobrazení větší diference mezi interpolovanými snímky byl pro obrázek 21 snížen počet kroků na pět. Na mezilehlých snímcích jsou patrné nuance, které ukazují postupný přechod od anatomických rysů přítomných na jednom snímku k těm na druhém. Na všech snímcích jsou dobře patrné anatomické struktury jako plíce, srdeční stín, žebra, páteř či bránice.

5.1.3 Porovnání s reálnými snímky

Tato kapitola obsahuje porovnání reálných snímků se snímky generovanými. Kvantifikace podobnosti různých snímků je složitý úkol, který přináší kompromisy. Jako metrika podobnosti snímků zde byla použita nejvyšší kosinová podobnost reálného snímku z trénovací sady dat k danému generovanému snímku. Tato metoda umožňuje přímé kvantitativní srovnání dvou snímků, ale její interpretace nemusí být jednoduchá, jak je zřejmé ze snímků níže. Metoda je implementována pomocí modulu knihovny PyTorch *nn.CosineSimilarity*.

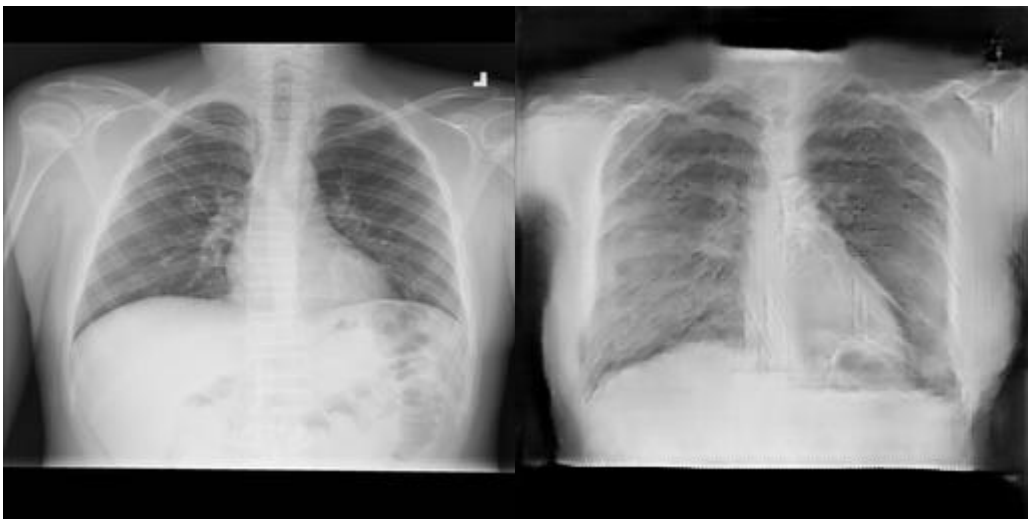
Obrázek 22 Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem



Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Generovaný snímek na pravé straně lze v kontextu trénovaného modelu považovat za kvalitní. Má jasně rozpoznatelné základní anatomické struktury, a obsahuje i některé hrubší detaily jako jsou průdušky a plicní cévy. Kosinová podobnost s reálným snímkem dosahuje 0,98343. Toto lze považovat za dobrý snímek relativně ke schopnostem natrénovaného modelu generátoru.

Obrázek 23 *Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem*



Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Tento představený generovaný snímek je na rozdíl od předchozího falešného snímku výrazně méně kvalitní. Míra detailu je viditelně nižší, a některé hrubší detaily jako je bránice, žebra či tlusté střevo jsou velmi špatně rozlišitelné. Přesto tento snímek dosahuje v hodnocení kosinové podobnosti skóre 0,97627, což není výrazně nižší skóre než skóre předešlého kvalitního generovaného snímku. Pravděpodobně tomu tak je proto, že oba výše uvedené snímky obsahují neanatomické artefakty v podobě černých pruhů v horní a spodní části snímku, čímž se snižuje plocha, ve které se mohou snímky odlišovat.

Obrázek 24 Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem

Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Generovaný snímek v této sade je velmi dobrou ukázkou toho, jakým způsobem vytváří generativní soupeřící sítě tzv. nové snímky. Generovaný snímek sám o sobě má nízkou kvalitu, srovnatelnou s předešlým generovaným snímkem. Jeho kosinová podobnost je 0,93771.

Vzorek latentního prostoru sítě je reprezentací vlastností, které se generátor naučil na generovaných snímcích vytvářet. Artefakty v podobě černých pruhů po stranách snímku jsou jednou z těchto možných vlastností, které se generátor naučil. Sestavením možných vlastností pak vzniká konkrétní snímek, v tomto případě snímek hrudníku s podlouhlými plícemi a černými artefakty po stranách. V trénovacím souboru tedy existují snímky s podlouhlými plícemi a snímky s černými pruhy po stranách, avšak ne snímky s černými pruhy a zároveň podlouhlými plícemi. V reálném datovém souboru tedy pravděpodobně neexistuje takový snímek, jinak by byl vybrán jako nejpodobnější snímek.

Obrázek 25 Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem

Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Zde je snímek s viditelnou abnormalitou v pravé plíci pacienta. Oproti ostatním snímkům mají oba snímky nižší kontrast v části plic. Generovaný snímek má poměrně nízkou kvalitu, ale dosáhl zatím nejvyšší podobnosti 0,98597.

Obrázek 26 Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem

Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Falešný snímek na pravé straně má viditelně velmi málo detailu, dosahuje však také velmi vysokého skóre 0,99204. Spolu s předchozím snímkem jsou ukázkami případů, pro které má trénovaný model zvláště velký problém naučit se zachycovat detaily. Je tomu tak pravděpodobně právě kvůli menšímu kontrastu anatomických struktur skutečného snímku, díky čemuž je pro kritika složitější zhodnotit reálnost těchto snímků. Generátor má pak

menší šanci se naučit tvořit detailní struktury, a učí se pomaleji. Problém vysokého skóre, avšak horší sledované kvality snímku také ukazuje, že kosinovou podobnost není vhodné používat jako jedinou metriku pro posouzení kvality snímku.

Obrázek 27 Porovnání generovaného snímku (vpravo) s jeho nejpodobnějším reálným snímkem

Zdroj: Vlastní zpracování, NIH Chest X-Ray dataset [1]

Poslední prezentovaný generovaný snímek má dobrý kontrast a je velmi podobný snímku reálnému, obsahujíc dobře definované plíce, srdeční stín či klíční kosti. Jeho kosinová podobnost dosahuje 0,98393.

5.2 Diskuse

5.2.1 Trénovací prostředí

Pro účely této práce bylo zvoleno mé standardní pracovní prostředí, sestávající z operačního systému Windows s použitím Windows subsystem for Linux s distribucí Ubuntu. Toto prostředí obvykle kombinuje jak výhody systému Windows, tak standardní pracovní postupy používané ve vývoji software v prostředí GNU/Linux. Právě v tomto hybridním prostředí byl zahájen prvotní tvorba s použitím knihovny TensorFlow.

Byla očekávána jednoduchá integrace knihovny TensorFlow s prostředím WSL a rychlé zprovoznění vývojového prostředí. Toto očekávání se však nenaplnilo. Samotná knihovna byla schopná základní funkce v prostředí WSL, avšak tato schopnost se omezovala pouze na použití CPU pro trénování neuronových sítí.

Pro vyřešení problému bylo provedeno důkladné přečtení dokumentace, několikanásobná reinstalace ovladačů grafického procesoru, konzultace s internetovými fóry či kontejnerizace vývojového prostředí. Žádná z těchto možností však neumožnila použití grafického procesoru pro virtualizované Linuxové prostředí. V důsledku toho bylo rozhodnuto přejít na nativní prostředí systému Windows pro implementaci TensorFlow.

Použití nativního prostředí Windows pro vývoj s knihovnou TensorFlow sice přineslo možnost využít pro trénink GPU, avšak také přineslo problémy s kompatibilitou různých knihoven, i z repositáře TensorFlow. Například knihovna Datasets, používaná pro správu trénovacích datových souborů, vykazovala nekompatibilitu s operačním systémem Windows[2].

Zjištěná omezení vyústila v přehodnocení zvoleného frameworku. Práce s knihovnou TensorFlow se bohužel sestávala zejména z řešení problémů, které nebyly přímo související se zaměřením práce. Tato zkušenost je v souladu se zjištěními z teoretické části této práce, kdy knihovna TensorFlow je doporučován pro zkušené uživatele. V této souvislosti bylo přehodnoceno rozhodnutí použití této knihovny, která dle literární rešerše poskytuje mírně lepší výkon, pro alternativní open-source knihovnu strojového učení PyTorch.

PyTorch s použitím WSL nabídl bezproblémové prostředí s podporou výpočtů na GPU, které je kritické pro minimalizaci času trénování velkých neuronových sítí. To umožnilo soustředění se na stavbu, vývoj a trénování modelu GAN namísto řešení problémů týkajících se trénovacího prostředí. Pro knihovnu PyTorch existuje dostatečná komunitní podpora a dostatek modulů a funkcí, tudíž trénovací proces nemusel podléhat kompromisu a nebylo třeba vytvářet potřebné vrstvy neuronových sítí. Celkově lze říci, že z pohledu vývojáře bylo používání knihovny PyTorch bezproblémové a jednoduché.

5.2.2 Výsledky generativního modelu

Je očividné, že současné výsledky modelu neobsahují jemné detaily, například kvůli nízkému výstupnímu rozlišení 256*256 pixelů. I když to omezuje přímou použitelnost generovaných snímků pro úlohy, které vyžadují tyto jemné detaily, jako je například diagnostika, jsou výsledky přesto slibné. Svědčí o tom nízké skóre FID a podobnost

[2] Pro daný problém byl otevřen ticket, dostupný na: https://github.com/tensorflow/datasets/issues/5075

generovaných snímků snímkům reálným, kvantifikována jako kosinová podobnost snímků. Tyto metriky naznačují silnou vizuální a strukturální shodu se skutečnými snímky CXR a zdůrazňují schopnost modelu vytvářet realistické snímky.

Za zmínku stojí schopnost modelu replikovat skutečné anatomické struktury, jako jsou plíce nebo mezihrudí. To naznačuje, že přestože generované snímky pravděpodobně postrádají detail pro klasifikační úlohu v medicínské oblasti, mohly by sloužit jako trénovací sada pro segmentační úlohu. Tato hypotéza je však nepotvrzená a k jejímu potvrzení či vyvrácení by byl třeba další výzkum.

Navzdory omezením předčila výkonnost modelu očekávání, zejména s ohledem na dobu tréninku. Doba tréninku výsledného modelu v součtu činí 47 hodin 19 minut a 12 sekund. Tato doba neobsahuje dobu trénovaní ostatních modelů. Je zřejmé, že jiné modely, jako například StyleGAN [80], dosahují mnohem lepších výsledků, avšak za výrazně vyšší čas – 14 dní a 22 hodin pro trénink StyleGAN v rozlišení 256*256 pixelů s použitím 1 GPU Tesla V100 [81]. Proto lze model v rámci zamýšleného použití a jeho omezení považovat za úspěšný.

5.2.3 Limitace práce a budoucí výzkum

Tato práce potvrzuje schopnosti generativních soupeřících sítí generovat realistické syntetické snímky. Trénování modelu však probíhalo s jistými omezeními, která nejen určují použitelnost natrénovaného modelu v praxi, ale jsou důležitá vzhledem k zjištěním v teoretické části práce.

Prvním a největším pozorovaným omezením je výpočetní náročnost trénování těchto generativních modelů. Jak bylo popsáno Rajpurkar et al. [32], jedním ze současných problémů AI v klinickém prostředí je náročné zpracování velkých datových souborů. Generativní sítě bohužel v současnosti potřebují pro kvalitní odhadnutí distribuce snímků také relativně velký počet těchto snímků. Trénování generativních sítí je navíc ještě složitější v porovnání s jinými sítěmi tím, že se zároveň trénují dvě sítě. Je tedy pravděpodobné, že trénování generativního modelu pro každého vědce či skupinu vědců zabývajících se umělou inteligencí zvlášť by nebylo časově ani prostorově výhodnější.

Řešením může být nahrazení poskytování datových souborů institucemi za poskytování generátoru. Kvalitní generátor poskytuje snímky ze stejného

pravděpodobnostního rozdělení a se stejnými znaky jako původní soubor. Síť generátoru je ale menší – v této práci má trénovací soubor obsahující snímky v rozlišení 256*256 2,72 GB, generátor má pouhých 140 MB. V případě opravdu velkých datových souborů může nahrazení specializovaným generátorem ušetřit netriviální objem paměti.

Druhý problém je dostupnost trénovacích dat. Generativní sítě potřebují k trénovaní relativně velký počet dat. Pokud není dostatek těchto trénovacích dat, ale existují jiná data z podobného rozdělení – například vzácná diagnóza mezi relativně dostupnými rentgenovými snímky – je použití generativního modelu pro vylepšení datového souboru možné. Pokud by však trénovacích dat nebyl dostatek, je pravděpodobnost vytvoření kvalitního modelu patrně velmi nízká.

Posledním zjištěným problémem je samotná kvalita generovaných snímků. Je zřejmé, že model vytvořený v rámci této práce neposkytuje snímky s dostatečnými detaily a v dostatečném rozlišení pro diagnostické úlohy; navíc se zdá, že vyšší rozlišení trénovacích snímků, než je v původní datové sadě, poskytuje přesnější modely [82]. Ale i nejmodernější generativní modely mají problémy konzistentně a jednoduše generovat kvalitní bezchybné snímky [83].

Tato práce pokládá základ pro několik možných pokračování výzkumu užitečnosti generativních modelů v lékařském prostředí. Budoucí práce by měly systematicky ohodnotit schopnosti modelů cvičených výhradně nebo částečně na syntetických datech oproti modelům trénovaným na datech reálných. To zahrnuje hodnocení věrnosti syntetických dat při replikaci složitosti skutečných lékařských snímků a určení, zda trénink na těchto datech může vést k modelům s praktickou použitelností v reálném světě. Klíčové metriky pro hodnocení by měly zahrnovat přesnost, citlivost a specifičnost v klinických diagnostických úlohách.

Jedním z přetrvávajících problémů v oblasti použití umělé inteligence v oblasti klinické praxe je nedostatek anotovaných dat. Výzkum by měl zkoumat, do jaké míry mohou generativní modely tento problém zmírnit generováním vysoce kvalitních syntetických dat, která by mohla nahradit nebo doplnit skutečné datové soubory. Zajímavým výzkumem by bezpochyby bylo nahrazení reálného datového souboru generativní sítí, zejména z hlediska doby tréninku, velikosti úložiště a potřebných výpočetních zdrojů.

Současné modely patrně dokáží generovat věrohodné snímky, avšak otázkou zůstává, zda jsou současné generativní modely spolehlivě a konzistentně vytvářet dostatečně kvalitní data pro úlohy vyžadující pečlivé detaily, jako je například lékařská diagnostika. Pro potřeby těchto úloh je nutné vytvoření kvalitních a robustních metrik, které budou objektivně hodnotit schopnost modelů vytvářet věrohodné detaily.

6 Závěr

V rámci této práce byly popsány současné metody použití hlubokých neuronových sítí pro generování realistických obrazových snímků, konkrétně modely VAE, GAN a difuzní modely. Byly uvedeny základní principy jejich funkce, kterým je odhad reálného pravděpodobnostního rozdělení za použití variačního odvozování, teorie her, respektive principu Markovových řetězců. Pro modely GAN byly uvedeny pokročilé techniky stabilizace tréninku, jako je využití Wasssersteinovy vzdálenosti či gradientní sankce. Dále byly rozebírány vlastnosti latentního prostoru a bylo předvedeno, že modely GAN se učí zobrazovat smysluplné vlastnosti snímků z reálného datového souboru. Byl popsán i opensource software určený k tvorbě modelů hlubokého učení, zejména knihovny TensorFlow a PyTorch.

Teoretická část práce se také věnovala možnostem využití AI v medicíně, například v diagnostice, při tvorbě léčiv nebo v managementu nemocnic. Byly představeny i výzvy pro AI v tomto odvětví. Hlavní výzvou je dostupnost datových souborů, která je omezena potřebou anotace lékařských specialistů, citlivou povahou těchto dat a nepřístupnosti medicínských zařízení pořizujících tyto snímky pro výzkumníky v oblasti umělé inteligence. Tato práce reaguje na tuto výzvu a prozkoumává možnosti vytvoření generativních modelů vytvářející realistických medicínských obrazových dat.

V praktické části práce byl vytvořen generativní model WGAN-GP pro tvorbu rentgenových snímků hrudníku v rozlišení 256*256 pixelů. Trénování bylo provedeno na 100000 snímků ze souboru NIH Chest X-ray. Výsledný model dosahuje skóre FID 0,1003 a IS 3,433. Na výsledcích modelu byly popsány základní anatomické struktury a byly ukázány nedostatky modelu. Dále bylo demonstrována lineární interpolace snímků, dokazující že model byl schopný se naučit smysluplné reprezentace i pro interpolované vzorky latentního prostoru. K vybraným výsledkům modelu byl pomocí kosinové podobnosti nalezen nejbližší reálný snímek z trénovacího souboru a prezentován v této práci.

Tato práce otevírá otázku využití generativních modelů nejen k rozšíření stávajících reálných datových souborů, ale i kompletního nahrazení reálného trénovacího souboru generativním modelem. Tyto modely mohou mít výrazně menší velikost než reálné datové soubory, a výstupy mohou být anotované snímky s vysokou spolehlivostí a spravedlivým rozdělením.

7 Seznam použitých zdrojů

[1] WANG, Xiaosong; PENG, Yifan; BAGHERI, Mohammadhadi; SUMMERS, Ronald; LU, Le a LU, Zhiyong. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. Online. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, USA, 2017, s. 3462-3471. Dostupné z: https://doi.org/10.1109/CVPR.2017.369. [cit. 2023-09-27].

[2] HEUSEL, Martin; RAMSAUER, Hubert; UNTERTHINER, Thomas a HOCHREITER, Sepp. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. Online. 2017. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1706.08500. [cit. 2023-09-28].

[3] SEITZER, Maximilian. Pytorch-fid: FID Score for PyTorch. Online. 2020. Dostupné z: https://github.com/mseitzer/pytorch-fid [cit. 2023-09-28].

[4] SEITZER, Maximilian. Minimum Images for Evaluation, question about dimensions. Online. 2020. Dostupné z: https://github.com/mseitzer/pytorch-fid/issues/78 [cit. 2023-09-28].

[5] KOMENDA, Martin. Online. In: Podobnosti a vzdálenosti ve vícerozměrném prostoru. Dostupné z: https://is.muni.cz/www/98951/41610771/43823411/43823458/Analyza_a_hodnoc/44563155/56049312/Vicerozmerky_-_kap4_-_metriky_final.docx?info. [cit. 2024-10-01].

[6] SALIMANS, Tim; GOODFELLOW, Ian; ZAREMBA, Wojciech; CHEUNG, Vicki; RADFORD, Alec et al. Improved Techniques for Training GANs. Online. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1606.03498. [cit. 2023-10-05].

[7] SZEGEDY, Christian; VANHOUCKE, Vincent; IOFFE, Sergey; SHLENS, Jonathon a WOJNA, Zbigniew. Rethinking the Inception Architecture for Computer Vision. Online. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1512.00567. [cit. 2023-10-10].

[8] TURING, Alan M. Computing Machinery and Intelligence. Mind. 1950, roč. LIX, č. 236, s. 433-460. Dostupné z: Doi: https://doi.org/10.1093/mind/LIX.236.433 [cit. 2023-10-15].

[9] IBM DATA AND AI TEAM. AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference? Online. 2023. Dostupné z: https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/. [cit. 2023-10-15].

[10] COPELAND, Jack. Artificial Intelligence. In: The Essential Turing. Oxford University Press, 2004, s. 353–361. ISBN 0–19–825079–7.

[11] OPENAI. GPT-4: Research. Online. 2023. Dostupné z: https://openai.com/research/gpt-4. [cit. 2023-10-15].

[12] OPENAI. DALL·E 3: Research. Online. 2023. Dostupné z: https://openai.com/dall-e-3. [cit. 2023-10-15].

[13] PACOVSKÝ, Ondřej. Vývoj umělé inteligence postupuje příliš rychle a bude potřeba ho regulovat, tvrdí vědec. Online. In: ČT24. ČT24. 2023. Dostupné z: https://ct24.ceskatelevize.cz/clanek/veda/vyvoj-umele-inteligence-postupuje-prilis-rychle-a-bude-potreba-ho-regulovat-tvrdi-vedec-563. [cit. 2023-12-07].

[14] LUCOVIČ, Matěj. Británie pořádá summit o umělé inteligenci. V budoucnu chce být světovou špičkou. Online. In: ČT24. ČT24. 2023. Dostupné z: https://ct24.ceskatelevize.cz/clanek/svet/britanie-porada-summit-o-umele-inteligenci-v-budoucnu-chce-byt-svetovou-spickou-745. [cit. 2023-12-07].

[15] GÉRON, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems. Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-4920-3264-9.

[16] CHOLLET, François. Deep learning with Python. Shelter Island: Manning, 2021. ISBN 978-1-61729-686-4.

[17] PESAPANE, Filippo; CODARI, Marina a SARDANELLI, Francesco. Artificial intelligence in medical imaging: threat or opportunity? Radiologists again at the forefront of innovation in medicine. European Radiology Experimental. 2018, roč. 35, č. 2.

[18] LIU, Peng-ran; LU, Lin; ZHANG, Jia-yao; HUO, Tong-tong; LIU, Song-xiang et al. Application of Artificial Intelligence in Medicine: An Overview. Online. Current Medical Science. 2021, č. 41, s. 1105-1115. Dostupné z:

https://doi.org/https://doi.org/10.1007/s11596-021-2474-3. [cit. 2023-11-08].

[19] OKAGAWA, Yutaka; ABE, Seiichiro; YAMADA, Masayoshi; ODA, Ichiro a SAITO, Yutaka. Artificial Intelligence in Endoscopy. Online. Digestive Diseases and Sciences. 2022, č. 67, s. 1553 - 1572. Dostupné z: https://doi.org/https://doi.org/10.1007/s10620-021-07086-z. [cit. 2023-11-08].

[20] WANG, Shidan; YANG, Donghan M.; RONG, Ruichen; ZHAN, Xiaowei a XIAO, Guanghua. Pathology Image Analysis Using Segmentation Deep Learning Algorithms. Online. The American Journal of Pathology. 2019, č. 9, s. 1686 - 1698. Dostupné z: https://doi.org/https://doi.org/10.1016/j.ajpath.2019.05.007. [cit. 2023-11-11].

[21] KOMURA, Daisuke a ISHIKAWA, Shumpei. Machine learning approaches for pathologic diagnosis. Online. Virchows Archiv. 2019, č. 475(2), s. 131 - 138. Dostupné z: https://doi.org/10.1007/s00428-019-02594-w. [cit. 2023-11-12].

[22] KAŠTELAN, Željko; KNEŽEVIĆ, Nikola; HUDOLIN, Tvrtko; KULIŠ, Tomislav; PENEZIĆ, Luka et al. Extraperitoneal radical prostatectomy with the Senhance Surgical System robotic platform. Online. Croatian medical journal. 2019, č. 60(6), s. 556 - 559. Dostupné z: https://doi.org/10.3325/cmj.2019.60.556. [cit. 2023-11-15].

[23] DLR. MiroSurge. Online. Dostupné z: https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-11674/. [cit. 2023-11-15].

[24] KELKAR, Dhananjay; BORSE, Mahindra A.; GODBOLE, Girish P. a SLACK, Mark. Interim safety analysis of the first-in-human clinical trial of the Versius surgical system, a new robot-assisted device for use in minimal access surgery. Online. Surgical Endoscopy. 2021, č. 35, s. 5193 - 5202. Dostupné z: https://doi.org/https://doi.org/10.1007/s00464-020-08014-4. [cit. 2023-11-08].

[25] CHRÁSTENSKÝ, Ivo. Robotická chirurgie v Ústí patří k nejpilnějším. Online. In: KRAJSKÁ ZDRAVOTNÍ, A. S. Krajská zdravotní, a. s. 2011. Dostupné z: https://www.kzcr.eu/ROP/centrum-roboticke-chirurgie-7/roboticka-chirurgie-v-usti-petri-k-nejpilnejsim-66/zprava.aspx. [cit. 2023-11-14].

[26] RUPP, Scott. The Da Vinci Medical Robot and AI. Online. ELECTRONIC HEALTH REPORTER. 2018. Dostupné z: https://electronichealthreporter.com/the-da-vinci-medical-robot-and-ai/. [cit. 2023-11-14].

[27] ZHOU, Xiao-Yun; GUO, Yao a YANG, Guang-Zhong. Application of artificial intelligence in surgery. Online. Frontiers of Medicine. 2020, č. 14, s. 417 - 430. Dostupné z: https://doi.org/https://doi.org/10.1007/s11684-020-0770-0. [cit. 2023-11-15].

[28] BELLINI, Valentina; VALENTE, Marina; BERTORELLI, Giorgia; PIFFERI, Barbara; CRACA, Michelangelo et al. Machine learning in perioperative medicine: a systematic review. Online. Journal of Anesthesia, Analgesia and Critical Care. 2022, č. 2, article 2. Dostupné z: /https://doi.org/10.1186/s44158-022-00033-y. [cit. 2023-11-18].

[29] PAUL, Debleena; SANAP, Gaurav; SHENOY, Snehal; KALYANE, Dnyaneshwar; KALIA, Kiran et al. Artificial intelligence in drug discovery and development. Online. Drug Discov Today. 2021, č. 26(1), s. 80–93. Dostupné z

https://doi.org/10.1016/j.drudis.2020.10.010. [cit. 2023-11-21].

[30] NAS, Serkan a KOYUNCU, Melik. Emergency Department Capacity Planning: A Recurrent Neural Network and Simulation Approach. Online. Computational and Mathematical Methods in Medicine. 2019, roč. 2019, article 4359719. Dostupné z: https://doi.org/https://doi.org/10.1155/2019/4359719. [cit. 2023-11-21].

[31] VAN DE SANDE, Davy; VAN GENDEREN, Michael E.; SMIT, Jim M.; HUISKENS, Joost; VISSER, Jacob J. et al. Developing, implementing and governing artificial intelligence in medicine: a step-by-step approach to prevent an artificial intelligence winter. Online. BMJ Health & Care Informatics. 2022, č. 29(1). Dostupné z: https://doi.org/10.1136/bmjhci-2021-100495. [cit. 2023-11-18].

[32] RAJPURKAR, Pranav; CHEN, Emma; BANERJEE, Oishi a TOPOL, Eric J. AI in health and medicine. Online. Nature Medicine. 2022, roč. 2022, č. 28, s. 31 - 38. Dostupné z: https://doi.org/https://doi.org/10.1038/s41591-021-01614-0. [cit. 2023-11-21].

[33] LI, Johann; ZHU, Guangming; HUA, Cong; FENG, Mingtao; BENNAMOUN, Basheer et al. A Systematic Collection of Medical Image Datasets for Deep Learning. Online. ACM Computing Surveys. 2023, roč. 56, č. 5, s. 1 - 51. Dostupné z: https://doi.org/https://dl.acm.org/doi/10.1145/3615862. [cit. 2023-12-01].

[34] SMITH, Helen. Clinical AI: opacity, accountability, responsibility and liability. Online. AI & Society. 2021, roč. 2021, č. 36, s. 535 - 545. Dostupné z: https://doi.org/https://doi.org/10.1007/s00146-020-01019-6. [cit. 2023-12-01].

[35] CHEN, Richard J.; CHEN, Tiffany Y.; LIPKOVA, Jana; WANG, Judy J.; WILLIAMSON, Drew F. K. et al. Algorithmic fairness in artificial intelligence for medicine and healthcare. Online. Nature Biomedical Engineering. 2023, roč. 2023, č. 7, s. 719 - 742. Dostupné z: https://doi.org/https://doi.org/10.1038/s41551-023-01056-8. [cit. 2023-12-01].

[36] GOODFELLOW, Ian; BENGIO, Yoshua a COURVILLE, Aaron. In: Deep Learning. MIT Press, 2016, s. 499 - 523.

[37] BOURLARD, Herve a KAMP, Y. Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. Online. Biological cybernetics. 1988, č. 59(4-5). Dostupné z: https://doi.org/10.1007/BF00332918. [cit. 2023-12-05].

[38] BISHOP, Christopher M. Pattern recognition and machine learning. [New York]: Springer, 2006. ISBN 0-387-31073-8.

[39] KINGMA, Diederik P. a Max WELLING. Auto-Encoding Variational Bayes [online]. 2013. Dostupné z: arXiv:1312.6114. Universiteit van Amsterdam.

[40] GREGOR, Karol; DANIHELKA, Ivo; GRAVES, Alex; REZENDE, Danilo Jimerez a WIESTRA, Daan. DRAW: A Recurrent Neural Network For Image Generation. Online. International conference on machine learning. 2015, s. 1462 - 1471. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1502.04623. [cit. 2023-12-06].

[41] SOHL-DICKSTEIN, Jascha; WEISS, Eric A.; MAHESWARANATHAN, Niru a GANGULI, Surya. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. Online. International conference on machine learning. 2015, s. 2256 - 2265. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1503.03585. [cit. 2023-12-10].

[42] HO, Jonathan; SAHARIA, Chitwan; CHAN, William; FLEET, David J.; NOROUZI, Mohammad et al. Cascaded Diffusion Models for High Fidelity Image Generation. Online. The Journal of Machine Learning Research. 2022, č. 23(1), s. 2249 - 2281. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.2106.15282. [cit. 2023-12-12].

[43] GOODFELLOW, Ian J., Jean POUGET-ABADIE, Mehdi MIRZA, Bing XU, David WARDE-FARLEY, Sherjil OZAIR, Aaron COURVILLE a Yoshua BENGIO. Generative Adversarial Networks [online]. Montréal, 2014. Dostupné z: arXiv:1406.2661. Universite de Montréal.

[44] OSBORNE, Martin J. An Introduction to Game Theory. Oxford University Press, 2004. ISBN 978-0198086109.

[45] FARNIA, Farzan a OZDAGLAR, Asuman, DAUMÉ, Hal a SINGH, Aarti (ed.). Do GANs always have Nash equilibria? Online. Proceedings of Machine Learning Research. 2020, č. 119, s. 3029 - 3039. Dostupné z:

https://proceedings.mlr.press/v119/farnia20a/farnia20a.pdf. [cit. 2023-12-12].

[46] ARJOVSKY, Martin; CHINTALA, Soumith a BOTTOU, Léon. Wasserstein GAN. Online. International conference on machine learning. 2017, s. 214 - 223. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1701.07875. [cit. 2023-12-14].

[47] Rubner, Y., Tomasi, C., & Guibas, L. J. (n.d.). A metric for distributions with applications to image databases. Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). doi:10.1109/iccv.1998.710701

[48] GULRAJANI, Ishaan; AHMED, Faruk; ARJOVSKY Martin; DUMOULIN, Vincent a COURVILLE, Aaron. Improved Training of Wasserstein GANs. Online. Advances in neural information processing systems. 2017, č. 30. Dostupné z:

https://doi.org/https://doi.org/10.48550/arXiv.1704.00028. [cit. 2023-12-15].

[49] ASPERTI, Andrea. Generative models and their latent space. Online. The Academic. 2023. Dostupné z: https://theacademic.com/generative-models-and-their-latent-space/. [cit. 2024-12-15].

[50] LIU, Ziwei; LUO, Ping; WANG, Xiaogang a TANG, Xiaoou. Large-scale CelebFaces Attributes (CelebA) Dataset. Online. 2015, 2021-09-10. Dostupné z: https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. [cit. 2023-12-15].

[51] BOJANOWSKI, Piotr; JOULIN, Armand; LOPEZ-PAZ, David a SZLAM, Arthur. Optimizing the Latent Space of Generative Networks. Online. 2017. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1707.05776. [cit. 2023-12-15].

[52] TensorFlow basics. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/basics. [cit. 2023-12-25].

[53] Introduction to graphs and tf.function. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/intro_to_graphs. [cit. 2023-12-25].

[54] GEVORKYAN, Migran Nelsonovich; DEMIDOVA, Anastasia; DEMIDOVA, Tatiana S. a SOBOLEV, Anton A. Review and comparative analysis of machine learning libraries for machine learning. Online. Discrete and Continuous Models and Applied Computational Science. 2019, č. 27(4), s. 305 - 315. Dostupné z: https://doi.org/10.22363/2658-4670-2019-27-4-305-315. [cit. 2023-12-27].

[55] Introduction to Tensors. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/tensor. [cit. 2023-12-25].

[56] Introduction to Variables. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/variable. [cit. 2023-12-25].

[57] Introduction to gradients and automatic differentiation. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/autodiff. [cit. 2023-12-25].

[58] Introduction to graphs and tf.function. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/intro_to_graphs. [cit. 2023-12-25].

[59] Writing a training loop from scratch. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/keras/writing_a_training_loop_from_scratch. [cit. 2023-12-25].

[60] Introduction to modules, layers, and models. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/intro_to_modules. [cit. 2023-12-25].

[61] Keras: The high-level API for TensorFlow. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/keras. [cit. 2023-12-25].

[62] The Sequential model. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/keras/sequential_model. [cit. 2023-12-25]. The Functional [63] API. Online. TensorFlow. Dostupné z: https://www.tensorflow.org/guide/keras/functional_api. [cit. 2023-12-25].

[64] Torch. Online. Dostupné z: http://torch.ch/. [cit. 2023-12-28].

[65] PyTorch, a year in.... Online. PyTorch. 2018. Dostupné z: https://pytorch.org/blog/a-year-in/. [cit. 2023-12-28].

[66] Quickstart. Online. PyTorch. Dostupné z: https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html. [cit. 2023-12-28].

[67] Torch.utils.data. Online. PyTorch. Dostupné z: https://pytorch.org/docs/stable/data.html. [cit. 2023-12-28].

[68] Datasets & dataloaders. Online. PyTorch. Dostupné z: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html. [cit. 2023-12-28].

[69] Transforms. Online. PyTorch. Dostupné z: https://pytorch.org/tutorials/beginner/basics/transforms_tutorial.html. [cit. 2023-12-28].

[70] Tensors. Online. PyTorch. Dostupné z: https://Pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html. [cit. 2023-12-28].

[71] Torch.nn. Online. PyTorch. Dostupné z: https://Pytorch.org/docs/stable/nn.html. [cit. 2023-12-28].

[72] Module. Online. PyTorch. Dostupné z: https://Pytorch.org/docs/stable/generated/torch.nn.Module.html. [cit. 2023-12-28].

[73] Build the neural network. Online. PyTorch. Dostupné z:

https://Pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html. [cit. 2023-12-28].

[74] Automated differentiation with torch.autograd. Online. PyTorch. Dostupné z: https://Pytorch.org/tutorials/beginner/basics/autogradqs_tutorial.html. [cit. 2023-12-28].

[75] Optimizing model parameters. Online. PyTorch. Dostupné z: https://Pytorch.org/tutorials/beginner/basics/optimization_tutorial.html. [cit. 2023-12-28].

[76] DAI, Hulin; PENG, Xuan; SHI, Xuanhua; HE, Ligang; XIONG, Qian et al. Reveal training performance mystery between TensorFlow and PyTorch in the single GPU environment. Online. Science China - Information Sciences. 2022, č. 65, article 112103. Dostupné z: https://doi.org/https://doi.org/10.1007/s11432-020-3182-1. [cit. 2023-12-27].

[77] STANČIN, I. a JOVIĆ, A. An overview and comparison of free Python libraries for data mining and big data analysis. Online. 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics. 2019, s. 977 - 982. Dostupné z: https://doi.org/doi: 10.23919/MIPRO.2019.8757088. [cit. 2023-12-27].

[78] Torch.flatten. Online. PyTorch. Dostupné z: https://pytorch.org/docs/stable/generated/torch.flatten.html. [cit. 2023-12-28].

[79] NOVAC, Ovidiu Constantin; CHIRODEA, Mihai Cristian; NOVAC, Cornelia Mihaela a BIZON, Nicu. Analysis of the Application Efficiency of TensorFlow and PyTorch in Convolutional Neural Network. Online. Sensors. 2022, č. 22(22). Dostupné z: https://doi.org/10.3390/s22228872. [cit. 2024-01-04].

[80] KARRAS, Tero; LAINE, Samuli a AILA, Timo. A Style-Based Generator Architecture for Generative Adversarial Networks. Online. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1812.04948. [cit. 2024-03-25].

[81] KARRAS, Tero a HELLSTEN, Janne. StyleGAN — Official TensorFlow Implementation. Online. 2019. Dostupné z: https://github.com/NVlabs/stylegan?tab=readme-ov-file. [cit. 2024-03-27].

[82] UL HAQUE, Inzamam a HINKLE, Jacob D. The Effect of Image Resolution on Automated Classification of Chest X-rays. Online. Journal of Medical Imaging. 2021. Dostupné z: https://doi.org/https://doi.org/10.1101/2021.07.30.21261225. [cit. 2024-03-27].

[83] ZHANG, Tianyi; WANG, Zheng; HUANG, Jing; TASNIM, Mohiuddin Muhammad a SHI, Wei. A Survey of Diffusion Based Image Generation Models: Issues and Their Solutions. Online. 2023. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.2308.13142. [cit. 2024-03-27].

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů

8.4 Seznam rovnic

8.5 Seznam použitých zkratek

1. GAN Generative adversial network, generativní soupeřící síť
2. WGAN-GP Wasserstein GAN with gradient penalty
3. AI Artificial intelligence, umělá inteligence
4. ML Machine learning, strojové učení
5. API Application programmable interface, aplikační programovatelné rozhraní
6. GPU Graphic processing unit, grafický procesor
7. LSTM Long short-term memory, dlouhodobá krátkodobá paměť
8. CAD Computer aided diagnosis, počítačem podporovaná diagnostika
9. NIH National Institutes of Health
10. FID Fréchet Inception Distance
11. IS Inception Score
12. WSL Windows subsystem for Linux

Přílohy

Příloha 1: Natrénovaný generativní model. Příloha je dostupná na nosiči, který je součástí této práce.
Příloha 2: Implementace generativního modelu. Příloha je dostupná na nosiči, který je součástí této práce.