

Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky



**Variační autoenkodér a úlohy
pozorování v latentním prostoru**

BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná informatika

Autor: Tomáš Faltejsek

Vedoucí práce: Ing. Ondřej Vadinský, Ph.D.

Konzultant práce: full consultant's name (incl. degrees)

Praha, květen 2023

Poděkování

Thanks.

Abstract

Jedním z předních rysů lidské inteligence je intuice a schopnost představovat si nové objekty. Variační autoenkovodér je inovací na poli pravděpodobnostních modelů, umožňující architekturu modelů schopných syntézy zcela nových dat s využitím pozorování atributů v latentním prostoru. Teoretická charakteristika a možnosti využití variačního autoenkovodéru jsou předmětem této bakalářské práce.

Keywords

keyword, important term, another topic, and another one

Obsah

Úvod	12
1 Východiska variačního autoenkodéru	13
1.1 Strojové učení	13
1.1.1 Algoritmus strojového učení	13
1.1.2 Učení s učitelem	13
1.1.3 Učení bez učitele	14
1.1.4 „No free lunch theorem“ pro strojové učení	14
1.1.5 Regularizace	15
1.2 Umělá neuronová síť	16
1.2.1 Architektura umělé neuronové sítě	16
1.2.2 Anatomie umělé neuronové sítě	16
1.2.3 Dopředná umělá neuronová síť	17
1.2.4 Perceptron	17
1.2.5 Vícevrstvý Perceptron a Algoritmus zpětné propagace	18
1.2.6 Univerzální approximační teorém	18
1.3 Redukce dimenziality	19
1.3.1 „The curse of dimensionality“	19
1.3.2 Extrakce vlastností	20
1.3.3 Analýza hlavních komponent (PCA)	20
1.4 Pravděpodobnostní modely	21
1.5 Kullback–Lieblerova divergence	21
1.6 Model využívající latentních proměnných	22
1.6.1 Maximum likelihood	23
1.7 Hluboký model využívající latentních proměnných	23
1.7.1 Latentní proměnné	24
1.8 TODO: Normální rozdělení	24
1.9 Pravděpodobnostní modely a variační inference	24
1.10 Generativní model	24
1.11 Intractabilities	24
2 Autoenkodér	25
2.1 Historický pohled	26
2.2 Mělký autoenkodér	26
2.3 Autoenkodér s neúplnou skrytou vrstvou	27
2.4 Autoenkodér s rozšířenou skrytou vrstvou	28
2.5 Hluboký autoenkodér	29
2.5.1 Stacked autoenkodér	29
2.6 Řídký autoenkodér	30

2.7	Denoising autoenkovdér	32
2.8	Robustní autoenkovdér	33
2.9	Contractive autoenkovdér	34
2.10	Stochastický autoenkovdér	35
2.11	Ostatní typy autoenkodérů	36
2.11.1	Adversiální autoenkovdér	36
2.12	Taxonomie autoenkodérů	37
2.13	Využití Autoenkodéru	37
3	Variační autoenkovdér	38
3.1	Motivace vzniku: variační autoenkovdér jako generativní model	38
3.2	Princip variačního autoenkodéru	40
3.2.1	Notace	41
3.3	Vymezení problémové oblasti	41
3.3.1	Volba latentních proměnných dle typu reprezentace informace	42
3.4	Enkodér modul	44
3.5	Dekodér modul	45
3.6	Evidence Lower Bound	46
3.6.1	Odvození rovnice	47
3.6.2	Optimalizační cíle	48
3.7	Optimalizace pomocí stochastického gradientního sestupu	48
3.7.1	Metoda stochastická gradientní optimalizace ELBO	50
3.8	Reparametrikační trik	51
3.9	Tři různé pohledy na interpretaci naučeného modelu	51
3.9.1	Chyba $\mathcal{D}_{KL}(q_\phi(z x^{(i)}) \parallel p_\theta(z))$	52
3.9.2	Interpretace z pohledu informační teorie	53
3.9.3	VAE a regularizační prvek	53
3.10	Generování nových vzorků	54
3.11	Testování naučeného modelu	56
3.12	Nedostatky a omezení	56
3.12.1	Omezení při optimalizaci	56
3.12.2	Šum v obrázcích vygenerovaných vzorků	57
3.13	Aktuální stav poznání a rozšíření variačního autoenkodéru	57
3.13.1	Wake-sleep algoritmus	57
3.13.2	Regularizované autoenkodéry	58
3.13.3	Generativní stochastické sítě	58
3.13.4	Efektivní učení reprezentací pomocí hlubokých Boltzmann machines .	58
3.13.5	Hluboké autoregresivní sítě	58
3.13.6	Stochastic Backpropagation and Approximate Inference in Deep Generative Models	59
3.13.7	Conditional variační autoenkovdér	59
3.13.8	Deep Recurrent Attention Writer	59
3.14	Pozorování v latentním prostoru	60

4 Úlohy pozorování v latentním prostoru	61
4.1 Generativní modelování obrazových dat	62
4.2 Opaková syntéza obrazových dat	63
4.3 Obarvení černobílých obrázků	64
4.4 Rekonstrukce obrazových dat	65
4.5 Konceptuální komprese	66
4.6 Syntéza přirozeného jazyka	67
4.7 Návrh chemikálií	68
4.8 Astronomie	69
4.9 TODO: Vizualizace high-dimensional dat	69
5 Experiment s modelem variačního autoenkodéru	70
5.1 Vymezení problémové oblasti	70
5.2 Datová sada	71
5.3 Architektura hlubokého modelu variačního autoenkodéru	71
5.3.1 Enkodér	72
5.3.2 Dekodér	74
5.3.3 Reparametrizační vrstva	75
5.3.4 Ztrátová funkce	76
5.3.5 Výsledný model	78
5.4 Trénování hlubokého modelu variačního autoenkodéru	79
5.4.1 Konvergence ztrátové funkce	80
5.4.2 Možné zlepšení trénovací strategie variačního autoenkodéru	81
5.5 Pozorování v latentním prostoru	81
5.5.1 Postupné formování latentního prostoru	83
5.6 Vzorkování z latentního prostoru	85
5.7 Evaluace	86
5.8 Diskuze	86
Závěr	88
Bibliografie	89
A Zdrojové kódy modelů	98
B Hodnoty ztrátové funkce modelu variačního autoenkodéru	99

Seznam obrázků

2.1	Obecná struktura autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h	25
2.2	Jednotlivé moduly architektury umělé neuronové sítě autoenkodéru.	25
2.3	Jednoduchá architektura umělé neuronové sítě Autoenkodéru s neúplnou skrytou vrstvou. Skrytá vrstva představuje <i>bottleneck</i>	27
2.4	Schéma umělé neuronové sítě hlubokého autoenkodéru s neúplnou skrytou vrstvou.	29
2.5	Procedura trénování denoising autoenkodéru. Převzato z (I. Goodfellow et al., 2016).	32
2.6	Obecná struktura Autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h	35
2.7	Autoenkodéry rozděleny dle charakteristik zpracování kódovací vrstvy	37
3.1	Umělá neuronová síť variačního autoenkodéru.	38
3.2	Máme-li náhodnou proměnnou z s nějakým rozdělením pravděpodobnosti, můžeme z ní vytvořit zcela novou náhodnou proměnnou $X = g(z)$ s kompletně jiným rozdělením.	43
3.3	VAE se učí stochastické mapování mezi pozorovaným prostorem x , jehož empirické rozdělení pravděpodobnosti $q_{\mathcal{D}}(x)$ je typicky velmi komplikované, a latentním prostorem z , jehož rozdělení pravděpodobnosti je typicky relativně jednoduché (např. kulovité, jako na tomto obrázku). Generativní model se učí rozdělení pravděpodobnosti $p_{\theta}(x, z)$ (které lze faktorizovat jako $p_{\theta}(x, z) = p_{\theta}(z)p_{\theta}(x z)$), s apriorním rozdělením skrze latentní prostor $p_{\theta}(z)$, a stochastický dekodér $p_{\theta}(x z)$. Stochastický enkodér $q_{\phi}(z x)$ approximuje původní (ale efektivně neřešitelné) posteriorní rozdělení $p_{\theta}(z x)$ generativního modelu. Obrázek včetně interpretace převzat z (Diederik P. Kingma a Max Welling, 2019).	46
3.4	Trénovací fáze dopředné umělé neuronové sítě VAE. Levý obrázek je znázorněn bez reparametizačního triku. Pravý obrázek zahrnuje reparametizační trik. Červené bloky zobrazují vzorkovací operace, které jsou nediferenciovatelné. Modré bloky zobrazují ztrátové vrstvy neuronové sítě. Dopředné chování obou sítí je identické , ale pouze v síti na pravém obrázku lze provést zpětná propagace. Obrázek včetně interpretace převzat z (Doersch, 2021).	49
3.5	Test-time schéma variačního autoenkodéru, které umožňuje generovat nové vzorky. Jelikož k vygenerování nového vzorku není potřeba enkodér, je vazba na něj jednoduše smazána (oproti síti, kterou zachycuje Obrázek 3.4). Schéma a interpretace převzato z (Doersch, 2021).	56
4.1	Aplikace VAE na generování zcela nových obrázků dvou cifer, trénováno na MNIST datasetu. Převzato z Gregor, Danihelka, Graves et al. (2015).	62
4.2	Výsledek vzorkování PixelVAE (Gulrajani et al., 2016), ImageNet dataset.	62

4.3	Využití VAE pro opakovou syntézu vstupního obrázku. V tomto příkladě lze pozorovat posun původního obrázku (vlevo) v modifikovaném latentním prostoru ve směru <i>vektoru úsměvu</i> (tzv. aritmetika v latentním prostoru), což má za výsledek progresivně více usmívající se obličeji. Převzato z White (2016).	63
4.4	Schéma trénovací a vzorkovací procedury rozšířeného CVAE pro predikci barvy pixelů původního obrázku. Převzato z Deshpande et al. (2017).	64
4.5	Vzorek vygenerovaných variací (vlevo)obarvení původního obrázku (vpravo). Vstupem pro generativní model byl pouze černobílý obrázek. Převzato z Deshpande et al. (2017).	64
4.6	Výstup modelů VAE pro rekonstrukci dat. Převzato z Yan et al. (2016).	65
4.7	Konceptuální komprese s použitím VAE aplikovaná na přirozené obrázky z ImageNet datasetu. Poslední řádek představuje originální obrázky. Zbylé řádky představují kompresované (čím menší řádek, tím méně latentních proměnných může model pro kompresi využít) rekonstrukce DRAW modelu. Převzato z Gregor, Besse et al. (2016).	66
4.8	Aplikace VAE pro interpolaci přirozeného jazyka mezi dvojicí vět. Vstupní páry vět jsou vyznačeny tučně, přechodné věty vygenerované modelem VAE jsou kurzívou. Pozoruhodně jsou přechodné věty gramaticky i syntakticky správné a zachovávají tématický kontext vstupních páru vět. Převzato z Bowman et al. (2016).	67
4.9	Aplikace VAE k návrhu chemikálií. Obrázek (a): Proces naučení latentní spojité reprezentace molekul z z velkého vstupního datasetu. Obrázek (b): Takto naučená spojité reprezentace umožňuje hledání nových molekul které maximalizují $f(z)$ (určitou <i>vyžadovanou vlastnost molekul</i>). Převzato z Gómez-Bombarelli et al. (2018).	68
4.10	Aplikace VAE k generování syntetických pseudo dat využitelných pro kalibraci systému k detekci zkreslení pozorování galaxií. K vygenerovanému vzorku pozorování je dodatečně přidán šum. Obrázek vlevo představuje vzorek z generativního modelu, obrázek vpravo představuje reálný snímek pozorování. Převzato z Ravankhsh et al. (2016).	69
5.1	Ukázka vzorků z MNIST.	71
5.2	Rozdíl mezi enkodér modulem autoenkodéru a enkodér modulem variačního autoenkodéru. Umožňuje vzorkování ze spojitého latentního prostoru VAE. Obrázek převzat z Foster (2023).	72
5.3	Diagram enkodér modulu VAE pro generování MNIST číslic.	74
5.4	Reparametizační vrstva modelu variačního autoenkodéru.	75
5.5	Trénovací krok modelu variačního autoenkodéru.	77
5.6	Instanciace navrhnutého modelu VAE pro úlohu generativního modelování obrazových dat MNIST.	78
5.7	Zahájení testovací fáze modelu variačního autoenkodéru pro úlohu generativního modelování obrazových dat MNIST.	79
5.8	Ztrátová funkce po 500 epochách konverguje k hodnotě ~ 135.8 . Osa x značí počet epoch. Osa y značí hodnotu ztrátové funkce.	80

5.9 Hodnoty dílčích prvků ztrátové funkce modelu variačního autoenkodéru po 500 epochách. Osa x značí počet epoch. Osa y značí hodnotu dílčího prvku ztrátové funkce.	80
5.10 Latentní prostor naučeného modelu variačního autoenkodéru. Barvy reprezentují jednotlivé třídy MNIST datasetu. Celkový počet zobrazených latentních proměnných je 10000.	82
5.11 Postupné formování struktur v latentním prostoru naučeného modelu pro generativní modelování MNIST číslic. Dílčí obrázky a, b, c, d zachycují výsledný latentní prostor modelu, jehož trénovací fáze zahrnovala 1, 5, 200, 500 epoch respektive. Při trénování jednotlivých modelů byla využita pouze trénovací množina MNIST datasetu bez štítků . Parametry vizualizace jsou pro každý dílčí obrázek identické a liší se pouze počtem epoch, kterými model v trénovací fázi prošel.	84
5.12 Vzorkování z latentního prostoru naučeného modelu. Červené body značí bod z latentního prostoru, který byl použit jako vstup pro dekódér modul.	85
5.13 Vzorky vygenerované naučeným modelem variačního autoenkodéru.	85
5.14 Náhodné vzorky z MNIST datasetu \times jejich rekonstrukce vygenerované variačním autoenkodérem. Hodnoty nad rekonstruovanou číslicí udávají souřadnice v latentním prostoru modelu, ze kterých byla tato rekonstrukce dekódována.	86
B.1 Hodnota ztrátové funkce modelu variačního autoenkodéru po 200 epochách trénování.	99
B.2 Hodnota ztrátové funkce modelu variačního autoenkodéru po 500 epochách trénování.	99

Note: Add a list of figures if the number of figures in the thesis text exceeds 20. A list of diagrams is applicable only if the author distinguishes between a figure and a diagram. The list of diagrams is included if the number of diagrams exceeds 20. This thesis template does not distinguish between a figure and a diagram.

Seznam tabulek

Note: Add a list of tables if the number of tables used in the thesis exceeds 20.

Seznam použitých zkratek

BCC Blind Carbon Copy

CC Carbon Copy

CERT Computer Emergency Response

Team

CSS Cascading Styleheets

DOI Digital Object Identifier

HTML Hypertext Markup Language

REST Representational State Transfer

SOAP Simple Object Access Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML eXtended Markup Language

Note: Add a list of abbreviations if the number of abbreviations used in the thesis exceeds 20 and the abbreviations used are not common.

Úvod

Introduction is a compulsory part of the bachelor's / diploma thesis. The introduction is an introduction to the topic. It elaborates the chosen topic, briefly puts it into context (there may also be a description of the motivation to write the work) and answers the question why the topic was chosen. It puts the topic into context and justifies its necessity and the topicality of the solution. It contains an explicit goal of the work. The text of the thesis goal is identical with the text that is given in the bachelor's thesis assignment, ie with the text that is given in the InSIS system and which is also given in the Abstract section.

Part of the introduction is also a brief introduction to the process of processing the work (a separate part of the actual text of the work is devoted to the method of processing). The introduction may also include a description of the motivation to write the work.

The introduction to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

Here are some sample chapters that recommend how a bachelor's / master's thesis should be set. They primarily describe the use of the L^AT_EX template, but general advice will also serve users of other systems well.

1. Východiska variačního autoenkodéru

1.1 Strojové učení

Strojové učení je podoblast umělé inteligence zabývající se algoritmy a technikami, které počítačovým systémům umožňují *učit se* z dat.

„Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.“ (Samuel, 1967)¹

1.1.1 Algoritmus strojového učení

Definici algoritmu strojového učení výstižně shrnuje následující definice (Mitchell, 1997, str. 2):

„A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.“

Algoritmy strojového učení lze obecně rozdělit na čtyři třídy – učení s učitelem (*supervised learning*), učení bez učitele (*unsupervised learning*), kombinaci učení s učitelem a učení bez učitele (*semi-supervised learning*) a posilovaného učení (*reinforcement learning*).

Toto dělení vychází ze zkušenosti E , resp. míry, do jaké má algoritmus strojového učení **povoleno** interagovat s datovou sadou (resp. jakou možnost taková datová sada nabízí). (I. Goodfellow et al., 2016)

Pro předmět této práce budou blíže představeny pouze oblasti **učení s učitelem** a **učení bez učitele**.

1.1.2 Učení s učitelem

Zkušenost E algoritmů učení bez učitele vychází z datové sady, která může mít celou řadu vlastností a zároveň je **předem známá klasifikace každého datového bodu do definovaných tříd**. Na základě této asociace je natrénovaný algoritmus schopen provést přiřazení dosud neznámého objektu do jedné z tříd definovaných v trénovací sadě dat.

¹Parafrázováno

Učení s učitelem zahrnuje pozorování několika příkladů náhodného vektoru \mathbf{x} a asociované hodnoty (či vektoru) \mathbf{y} . Následuje učení se predikovat \mathbf{y} z \mathbf{x} , často na základě odhadu $p(\mathbf{y}|\mathbf{x})$.

Samotný termín *učení s učitelem* vychází ze situace, kdy je cílová třída \mathbf{y} poskytnuta jakýmsi instruktorem či učitelem, který systému strojového učení ukazuje očekávané chování. (I. Goodfellow et al., 2016)

1.1.3 Učení bez učitele

Ve vztahu k definici (viz Podsekce 1.1.1) lze říci, že zkušenost E algoritmů učení bez učitele vychází z datové sady, která může mít celou řadu vlastností. Cílem trénování algoritmů učení bez učitele je **naučit se užitečné a charakteristické vlastnosti o struktuře vstupní datové sady**. V kontextu hlubokého učení je pak obvyklým cílem algoritmu naučit se celé rozdělení pravděpodobnosti, které generuje původní datovou sadu (ať už explicitně – např. odhad hustoty, či implicitně – např. úlohy syntézy dat a odstranění šumu). Mezi další techniky strojového učení bez učitele se, mimo jiné, řadí shluková analýza.

Obecně lze říct, že učení bez učitele zahrnuje pozorování několika příkladů náhodného vektoru \mathbf{x} na základě kterého se snaží implicitně či explicitně *naučit* rozdělení pravděpodobnosti $p(\mathbf{x})$, případně *užitečné vlastnosti* tohoto pravděpodobnostního rozdělení.

Na rozdíl od Podsekce 1.1.2, název *učení bez učitele* napovídá, že v procesu učení není zapojen žádný *instruktor* či *učitel*, a tak systém strojového učení sám musí vyvodit smysl a užitečné vlastnosti předložené datové sady. (I. Goodfellow et al., 2016)

1.1.4 „No free lunch theorem“ pro strojové učení

Dle teorie má algoritmus strojového učení schopnost generalizace i z konečné množiny trénovacích dat. Toto tvrzení je ale v rozporu s elementárními principy logiky – indukce obecných pravidel z omezeného vzorku dat je logicky nevalidní. Chceme-li provést indukci obecného pravidla, které popisuje každý prvek množiny, musíme mít k dispozici informaci o každém prvku z množiny. (I. Goodfellow et al., 2016)

Pro logické vyvrácení takto naučené generalizace stačí být jeden vzorek, který je s tímto pravidlem v nesouladu a nebyl součástí trénovací množiny (tzv. *black swan paradox*).

Oblast strojového učení se tomuto paradoxu z části vyhýbá tím, že pracuje pouze s **pravděpodobnostními pravidly** (oproti zcela určitým pravidlům jako v logické indukci). Algoritmy strojového učení hledají pravidla, která jsou tzv. *probably approximately correct*. (Valiant, 1984)

Ani tento trik však kompletně neřeší představený problém. Zjednodušeně, **„No free lunch theorem“** pro strojové učení tvrdí, že každý klasifikační algoritmus má v průměru, skrze

všechny distribuce generující data, **stejnou chybovost** při klasifikování dosud nepozorovaných datových bodů. (Wolpert, 1996) Jinými slovy, **žádný algoritmus strojového učení není univerzálně lepší než kterýkoliv jiný algoritmus strojového učení.**

Toto tvrzení je pravdivé až při zohlednění *všech možných distribucí generujících data* – tedy v teoretické rovině. Lze pozorovat, že v praktických aplikacích je možné navrhnut algoritmus, který si v určitých distribucích vede v průměru lépe než ostatní algoritmy. (I. Goodfellow et al., 2016)

Cílem této práce je představit Variační autoenkovdér – architekturu umělé neuronové sítě, která se těší dobrým výsledkům ve vybraných úlohách učení bez učitele, představených v Kapitola 4, ale v důsledku má i své nedostatky (představené v Kapitola 3) v ostatních třídách úloh.

1.1.5 Regularizace

„No free lunch theorem“ pro strojové učení (představený v Podsekce 1.1.4) implikuje nutnost návrhu algoritmu strojového učení pro konkrétní úlohu, chceme-li aby jeho výkonnost byla vyšší než výkonnost ostatních algoritmů v průměru. Toho lze docílit *zabudováním* určité sady preferencí přímo do algoritmu strojového učení (předpokladem je, že tyto preference jsou v souladu s cílovým problémem, který se algoritmus snaží řešit). (I. Goodfellow et al., 2016)

Chování a výkonnost algoritmu strojového učení lze ovlivnit zvolením velikosti množiny funkcí (a následně konkrétní podoby jejich identity), které jsou v jeho prostoru hypotéz povoleny². V prostoru hypotéz algoritmu lze rovněž vyjádřit preferenci jednoho řešení před druhým. To znamená, že obě takové funkce budou přípustné, ale jedna z nich má preferenci (tedy nepreferovaná funkce bude zvolena právě když je její evaluace vůči trénovací sadě *výrazně* lepší, než preferovaná funkce)³. Případně můžeme funkci (nebo množinu funkcí) z prostoru hypotéz vyřadit kompletně (resp. vyjádřit tak nekonečně velkou míru neupřednostnění takové funkce, či množiny funkcí). (I. Goodfellow et al., 2016)

Obecně můžeme regularizovat model přičtením *trestu* k jeho ztrátové funkci. Tento trest nazýváme **regularizační prvek** (*regularizer*) a značíme jej Ω .

Regularizaci pak definujeme jako:

„Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. Regularization is one of the central concerns of the field of machine learning, rivaled in its importance only by optimization.“ (I. Goodfellow et al., 2016)

²Například prostor hypotéz lineární regrese je složen z množiny lineárních funkcí jejího vstupu. Tedy lineární regrese zřejmě bude mít problém věrohodně predikovat hodnotu $\sin(x)$ z x (a stejně tak řešit další nelineární problémy).

³Například *weight decay*.

Formu regularizace je tedy nutné pečlivě zvolit s ohledem na typ úlohy, který má algoritmus za cíl řešit. I autoenkové (a variační autoenkové) mohou být navrhnuty k řešení celé řady problémů. Jednotlivé metody regularizace autoenkových jsou představeny v Kapitola 2 a jejich následné aplikace v Kapitola 4.

1.2 Umělá neuronová síť

Umělá neuronová síť je model strojového učení inspirovaný přírodou. Zdá se být intuitivní, že chceme-li napodobit lidskou inteligenci, měli bychom se pro inspiraci podívat na architekturu lidského mozku. V průběhu času se však konstrukce umělých neuronových sítí začala jejich přírodnímu protějšku podstatně vzdalovat. Řada architektur umělých neuronových sítí tvoří biologicky nerealistický model, byť z této původní myšlenky vychází (stejně tak jako letadla vycházejí z přírodního vzoru létajících ptáků, pohybem svými křídly se ve skutečnosti ve vzduchu neodrážejí). (Geron, 2019)

Představení kompletních principů umělých neuronových sítí není východiskem variačního autoenkového, nýbrž svým obsahem pokrývají několik monografií – pro úvod například (Chollet, 2017), (Geron, 2019). V této sekci tedy budou představeny pouze stěžejní techniky využívané autoenkové.

1.2.1 Architektura umělé neuronové sítě

Pojem **architektura** rozumíme **celkovou strukturu** umělé neuronové sítě – počet neuronů a způsob jakým jsou tyto neurony mezi sebou propojeny, jejich aktivační funkce a podobně.

Většina umělých neuronových sítí je organizována do skupin neuronů zvaných **vrstvy**. Architektury umělých neuronových sítí poté uspořádávají tyto vrstvy do zřetězené struktury, kde každá vrstva je funkcí předcházející vrstvy. (I. Goodfellow et al., 2016)

V takové struktuře je první vrstva dána následovně:

$$\mathbf{h}^{(1)} = g^{(1)} \left(\mathbf{W}^{(1)\top} \mathbf{x} + \mathbf{b}^{(1)} \right), \quad (1.1)$$

a druhá vrstva je dána následovně:

$$\mathbf{h}^{(2)} = g^{(2)} \left(\mathbf{W}^{(2)\top} \mathbf{h}^{(1)} + \mathbf{b}^{(2)} \right). \quad (1.2)$$

Mezi hlavní architektonická rozhodnutí při návrhu umělé neuronové sítě patří volba **hloubky sítě** a **šířky každé z vrstev**.

1.2.2 Anatomie umělé neuronové sítě

Proces trénování umělé neuronové sítě z pravidla zahrnuje následující objekty (Chollet, 2017):

- *Vrstvy* ze kterých je následně složena *síť* (resp. *model*)
- *Vstupní data* (a případně jejich *cílové třídy*)
- *Ztrátová funkce*, která slouží jako signál zpětné vazby použití pro učení modelu
- *Optimizér*, který modifikuje parametry umělé neuronové sítě (např. váhy)

1.2.3 Dopředná umělá neuronová síť

Dopředná umělá neuronová síť je velmi podstatná pro návrh (hlubokých) modelů strojového učení. Cílem dopředné umělé neuronové sítě je approximovat nějakou funkci f^* .⁴ Dopředná umělá neuronová síť definuje mapovací funkci $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ a učí se hodnotu parametrů $\boldsymbol{\theta}$, jejichž výsledky je nejlepší approximace cílové funkce. (I. Goodfellow et al., 2016)

Dopředné umělé neuronové **sítě** nazýváme:

- *Dopředné*, jelikož v této architektuře neexistují žádné zpětnovazební propojení, které by sloužily jako vstup zpět pro sebe sama⁵.
- *Sítě*, jelikož jsou typicky reprezentovány kompozicí více různých funkcí. Model takové sítě je asociovaný s orientovaným acyklickým grafem, který popisuje, jakým způsobem je tato kompozice tvořena. Mějme tři funkce $f^{(1)}$, $f^{(2)}$, $f^{(3)}$, které jsou zřetězeny následovně: $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Takto zřetězené struktury nazýváme **vrstvy** umělé neuronové sítě. Délku tohoto zřetězení nazýváme **hloubkou** modelu – odtud *hluboké* neuronové sítě (umělé neuronové sítě s 2 a více skrytými vrstvami). (I. Goodfellow et al., 2016)⁶.

1.2.4 Perceptron

Jedná se o jednu z nejjednodušších architektur umělé neuronové sítě (a *model umělého neuronu*) představenou v (Rosenblatt, 1957) inspirovanou principy Hebbovského učení (Hebb, 1949). Perceptron přichází s důležitým principem **numerických hodnot vstupů, výstupů** (oproti pouhým binárním hodnotám) – tzv. *linear threshold unit*, *LTU* a **vah** propojení mezi jednotlivými neurony. Perceptron byl později kritizován (Minsky a Papert, 1969) za jeho neschopnost řešit triviální problémy (např. *XOR* klasifikace), což eventuálně vedlo ke krátkodobé stagnaci konekcionismu.

Jak se ale ukázalo, některé z těchto limitací lze vyřešit uspořádáním více Perceptronů za sebe do vrstev. (Rumelhart a McClelland, 1987)

Takto uspořádaná umělá neuronová síť se nazývá Vícevrstvý Perceptron.

⁴Například pro klasifikátor, $y = f^*(x)$ mapuje vstup x do kategorie y . Převzato z (I. Goodfellow et al., 2016).

⁵Je-li dopředná umělá neuronová sítě obohacena o zpětnovazební propojení se sebou sama, nazýváme ji **Rekurentní neuronová síť**.

⁶V tomto případě je $f^{(1)}$ **první vrstva**, $f^{(2)}$ je **druhá vrstva** a $f^{(3)}$ **třetí vrstva**.

1.2.5 Vícevrstvý Perceptron a Algoritmus zpětné propagace

Vícevrstvý Perceptron je tvořen umělou neuronovou sítí, která se skládá z jedné vstupní vrstvy, **jedné nebo více skrytých vrstev** LTU jednotek, a jedné výstupní vrstvy rovněž složené z LTU jednotek. Součástí každé vrstvy (vyjma výstupní) je tzv. **bias** neuron, který je **plně propojený** s další vrstvou sítě. (Geron, 2019)

Pokud má umělá neuronová síť dvě a více skrytých vrstev, nazýváme ji **hlubokou neuronovou sítí**.

Jak již víme, Vícevrstvý Perceptron adresuje limitace Perceptronu (viz Podsekce 1.2.4). Článek (Rumelhart a McClelland, 1987) ale představil i další revoluční myšlenku – algoritmus **zpětné propagace**⁷ (*backpropagation*).

Algoritmus zpětné propagace lze velmi zjednodušeně interpretovat následovně: Pro každou trénovací instanci je vypočten výstup každého jejího neuronu každé jednotlivé vrstvy. Poté je změřena výstupní chyba celé sítě (například střední kvadratická chyba, MSE) a vypočten podíl, kterým každý neuron poslední skryté vrstvy přispěl k hodnotě každého neuronu výstupní vrstvy. Následně je stejným způsobem měřeno, jak moc byla ovlivněna hodnota jednotlivých neuronů poslední skryté vrstvy hodnotami neuronů předchozí skryté vrstvy – a podobný proces se opakuje než algoritmus dosáhne vstupní vrstvy. (Geron, 2019)

Průchod algoritmu sítí lze intuitivně popsat následovně: Pro každou trénovací instanci algoritmus zpětné propagace nejprve učí predikci cílové hodnoty – tzv. *forward pass*. Poté změří chybu této predikce. Následně zpětně projde každou vrstvu sítě za účelem změření míry přispění k této chybě každým propojením (a jeho vahou) – tzv. *reverse pass*. Závěrem algoritmus *lehce* upraví váhy jednotlivých propojení za účelem snížení chyby – tzv. *Gradient Descent step*. (Geron, 2019)

Pro správný chod algoritmu autoři (Rumelhart a McClelland, 1987) do architektury Vícevrstvého Perceptronu zanesli další zásadní změnu – jako tzv. **aktivační funkci** využili logistickou regresi (zcela běžné je využití i dalších aktivačních funkcí, například ReLU).

1.2.6 Univerzální approximační teorém

Univerzální approximační teorém⁸ (Hornik, M. B. Stinchcombe et al., 1989), (Cybenko, 1989) tvrdí, že dopředná umělá neuronová síť (s alespoň jednou skrytou vrstvou a *potlačující* aktivační funkcí – např. logistická funkce) je schopna approximovat libovolnou (borelovsky měřitelnou) funkci s *libovolnou mírou přesnosti*⁹.

⁷Algoritmus zpětné propagace byl ve skutečnosti nezávisle objeven více výzkumníky, počínaje (Werbos, 1974). Převzato z (Geron, 2019).

⁸Universal approximation theorem

⁹Derivacemi dopředné umělé neuronové sítě lze stejně tak approximovat derivace cílové funkce (Hornik, M. Stinchcombe et al., 1990).

Důsledkem Univerzálního aproximačního teorému je, že dostatečně velký **Vícevrstvý perceptron zvládne reprezentovat libovolnou funkci**. Ale neexistuje zde jistota, že se algoritmus strojového učení bude schopen takovou funkci *vždy* naučit (I. Goodfellow et al., 2016):

- Zvolený optimizér algoritmu strojového učení použitý pro trénování nemusí být schopen najít parametry umělé neuronové sítě, které korespondují s cílovou funkcí.
- Algoritmus nemusí vybrat správnou funkci v důsledku přeúčení.

Dopředné umělé neuronové sítě tak nabízejí univerzální systém pro **reprezentaci** libovolné funkce. **Pro libovolnou funkci existuje dopředná umělá neuronová síť, která ji approximuje**¹⁰.

1.3 Redukce dimenziality

Do oblasti redukce dimenziality patří celá řada technik pro práci s vysokodimenzionálními daty. Cílem této oblasti, na rozdíl od regresních problémů, není predikovat hodnotu cílové proměnné – ale porozumět tvaru dat, se kterými pracuje. Typickou úlohou redukce dimenziality dat je sestrojit **nízkodimenzionální reprezentaci**, která zachytí *většinu významu* původní, vysokodimenzionální, reprezentace. Tento jev nazýváme hledáním **sali-entních vlastností** původní sady dat. (Phillips, 2021)

Redukci dimenziality lze chápat jako úlohu učení bez učitele, ve které se algoritmus strojového učení učí mapování z vysokodimenzionálního prostoru $\mathbf{x} \in \mathbb{R}^D$ do nízkodimenzionálního latentního prostoru $\mathbf{z} \in \mathbb{R}^L$. (Murphy, 2022)

1.3.1 „The curse of dimensionality“

S rostoucím počtem vstupních proměnných exponenciálně roste počet vzorků dat nutný pro *libovolně přesnou* approximaci dané funkce. V důsledku je tedy s rostoucí dimenzí vstupních dat značně degradováno chování většiny algoritmů (strojového učení). Tento problém je známý pod termínem **curse of dimensionality**. (Bellman, 1957)

I proto došlo ke vzniku oblasti zvané *feature engineering*. Feature engineering je oblast, která se, mimo jiné, zabývá disciplínou selekce vlastností dat, které budou použity při trénování modelu. Pro automatizovanou selekci vlastností existuje celá řada technik. Výběr podprostoru, který *nejlépe* reprezentuje vlastnosti původních dat je NP-těžký kombinatorický problém (*exhaustive search through all the subsets of features*). Tyto techniky dokonce často vyhodnocují každou vstupní proměnnou nezávisle, což může vést ke zkresleným závěrům o jejich

¹⁰S ohledem na Podsekce 1.1.4 neexistuje žádný univerzální způsob pro prozkoumání trénovací množiny dat a zvolení funkce, která bude libovolně přesně generalizovat na instance dat, které nejsou součástí této trénovací množiny.

významnosti – naopak je běžné, že proměnné začínají vykazovat určitou míru významnosti až při vzájemném využití. (Stańczyk a Jain, 2015)

Výše stanovené důvody vedly k emergenci další disciplíny, a to **extrakce vlastností**, o níž pojednává Podsekce 1.3.2.

1.3.2 Extrakce vlastností

Cílem extrakce vlastností *feature extraction* je najít reprezentaci vstupních dat, která je vhodná pro algoritmus strojového učení, který se chystáme využít (jelikož původní reprezentace může být z mnoha důvodů nevhodná – například vysokodimenzionální). Typicky tak musí dojít k redukci dimenzionality vstupních dat. (H. Liu a Motoda, 1998)

K extrakci nových vlastností lze dojít mnoha způsoby. Existují techniky založené na hledání lineárních kombinací původních vstupních vlastností, například Analýza hlavních komponent (*PCA Analýza*) nebo Lineární diskriminační analýza (*LDA Analýza*).

Pro nelineární redukci dimenzionality lze využít technik tzv. *manifold learningu*, viz (I. Goodfellow et al., 2016, kap. 5.11.3.).

1.3.3 Analýza hlavních komponent (PCA)

Jednou z nejrozšířenějších metod pro redukci dimenzionality je PCA analýza (*principal component analysis*).

V principu se jedná o nalezení lineární a ortogonální projekce vysokodimenzionálních dat $\mathbf{x} \in \mathbb{R}^D$ do nízkodimenzionálního podprostoru $\mathbf{z} \in \mathbb{R}^L$, tak aby tato nízkodimenzionální reprezentace byla *dobrou approximací* původních dat. Provedeme-li projekci (resp. **kódování**) \mathbf{x} , získáme $\mathbf{z} = \mathbf{W}^\top \mathbf{x}$ (funkci kódování označme e), a následně **dekódujeme** \mathbf{z} abychom získali $\hat{\mathbf{x}} = \mathbf{W}\mathbf{z}$ (funkci dekódování označme d), pak chceme aby $\hat{\mathbf{x}}$ bylo co možná nejblíže \mathbf{x} (měřeno pomocí ℓ_2 vzdálenosti). (Murphy, 2022)

Konkrétně můžeme definovat následující **chybu rekonstrukce** (Murphy, 2022):

$$\mathcal{L}(\mathbf{W}) \triangleq \frac{1}{N} \|\mathbf{x} - d(e(\mathbf{x}_n; \mathbf{W}); \mathbf{W})\|_2^2 \quad (1.3)$$

kde e a d jsou lineární funkce (*mappings*).

Na PCA analýzu tak lze pohlížet jako na algoritmus učení bez učitele, který se učí reprezentaci vstupních dat. Tato reprezentace má dvě kritéria (I. Goodfellow et al., 2016):

- PCA se učí reprezentaci vstupních dat, která má menší dimenze než původní vstupní reprezentace.
- PCA se učí reprezentaci, jejíž elementy mezi sebou nemají žádnou lineární korelacii.

1.4 Pravděpodobnostní modely

Jak napovídá definice z Podsekce 1.1.1, existuje mnoho druhů úloh T , ke kterým je možné využít algoritmy strojového učení. Tato práce se zabývá především úlohami v pravděpodobnostním kontextu. Oblast strojového učení, která tento typ úloh řeší se nazývá **pravděpodobnostní modelování**.

Pravděpodobnostní modely strojového učení pracují s cílovou proměnnou (třídou) jako s **náhodnou proměnnou**, která **podléhá nějakému rozdělení pravděpodobnosti**. Toto rozdělení popisuje váženou množinu hodnot, kterých může náhodná proměnná nabýt. (Murphy, 2022)

Pro nutnost adopce pravděpodobnostních modelů hovoří dva hlavní důvody – jedná se o **optimální přístup pro rozhodování s faktorem neurčitosti** a poté, pravděpodobnostní modelování je *přirozené* pro řadu inženýrských disciplín (například stochastická optimalizace, operační výzkum, ekonometrie, informační teorie a další). (Murphy, 2022)

1.5 Kullback–Lieblerova divergence

Kullback–Lieblerova divergence (Kullback a Leibler, 1951), dále jen KL divergence, vyjadřuje míru *podobnosti* rozdělení pravděpodobnosti p vůči jinému rozdělení pravděpodobnosti q ¹¹.

Nechť $p(x)$ a $q(x)$ jsou rozdělení pravděpodobnosti diskrétní náhodné veličiny pro x ¹². A dále $p(x) > 0$ a $q(x) > 0$ pro každé $x \in X$. Pak definujeme KL divergenci následovně (Murphy, 2022):

$$D_{KL}(p(x)\|q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} \quad (1.4)$$

A spojitou verzi KL divergence následovně (Murphy, 2022):

$$D_{KL}(p(x)\|q(x)) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} dx \quad (1.5)$$

KL divergence je úzce spjata s *relativní entropií* a *teorií informace*. Lze ji tedy interpretovat jako *počet bitů*, které jsou **dodatečně** potřebné pro kódování vzorků dat nevhodnou distribucí q v porovnání s kódováním stejných vzorků dat jejich původní distribucí p .

¹¹Nutno zdůraznit, že ačkoliv lze intuitivně hovořit o *podobnosti* či *vzdálenosti* mezi distribucí p a q , KL divergence **není metrická**. Není symetrická (tedy nesplňuje M3) a neplatí v ní trojúhelníková nerovnost (tedy nesplňuje M4). (Phillips, 2021)

¹²Tedy součet $p(x)$ a $q(x) = 1$

Existuje celá řada dalších (a přesnějších) interpretací KL divergence. Pro formální zavedení a definici vlastností KL divergence odkazují na (Murphy, 2023, kap. 5.1).

Minimalizace KL divergence je využita jako součást trénovacího procesu variačního autoen-kodéru (viz Kapitola 3).

1.6 Model využívající latentních proměnných

Čím komplexnější jsou závislosti mezi dimenzemi generativního modelu, tím složitější je jeho trénování.

Mějme například jednoduchou úlohu generování obrázků ručně psaných číslic 0-9. Snažíme-li se generovat obrázek číslice 5, tak víme, že pravá strana obrázku nemůže obsahovat pravou stranu obrázku číslice 0 (v opačném případě se zřejmě nemůže jednat o obrázek číslice 5).

Intuitivně je tak vhodnou strategií, aby model nejprve učinil rozhodnutí o tom, jakou číslici se chystá generovat, než začne konkrétním pixelům přidělovat hodnoty. Takové rozhodnutí nazýváme *latentní proměnnou*. Tedy, před tím, než model začne generovat obrázek, náhodně vybere vzorek z z množiny $[0, \dots, 9]$ a při generovaní obrázku se ujistí, že všechny vygenerované pixely souhlasí s charakteristikami této číslice. V tomto kontextu *latentní* znamená, že pro danou číslici, kterou model vygeneroval, nemusí být nutně známo jaké nastavení této latentní proměnné bylo modelem použito (to bychom museli odvodit například počítavým viděním). (Doersch, 2021)

Před tím, než lze prohlásit že model je *reprezentací* vstupního datasetu, musíme se ujistit, že pro každý datový bod X v tomto datasetu existuje alespoň jedna konfigurace latentních proměnných modelu, která zajistí že model vygeneruje obrázek *velice podobný* X .

Formálně řekneme, že ve vysokodimenzionálním prostoru \mathcal{Z} existuje vektor latentních proměnných z , ze kterého lze podle *nějaké hustoty pravděpodobnosti* (*probability density function, PDF*) $P(z)$, definované skrze \mathcal{Z} , jednoduše vzorkovat X . Dále, máme-li množinu deterministických funkcí $f(z; \theta)$, parametrisovaných vektorem θ v nějakém prostoru Θ , kde $f : \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$. Byť je f deterministická, je-li z náhodné a θ neměnné, pak $f(z; \theta)$ je náhodná proměnná z prostoru \mathcal{X} . Cílem je optimalizovat θ tak, aby bylo možné vzorkovat z z $P(z)$ a aby, s vysokou pravděpodobností, každý výstup $f(z; \theta)$ byl podobný příslušným X ze vstupního datasetu. (Doersch, 2021)

Tedy, chceme maximalizovat pravděpodobnost každého X z trénovací množiny skrze celý generativní proces dle:

$$P(X) = \int P(X|z; \theta)P(z)dz. \quad (1.6)$$

Kde $f(z; \theta)$ bylo nahrazeno distribucí $P(X|z; \theta)$, která umožnuje explicitně vyjádřit závislost X na z (ze zákonu celkové pravděpodobnosti). (Doersch, 2021)

1.6.1 Maximum likelihood

Intuitivní myšlenka za tímto principem, nazývaným *marginal likelihood*, je, že pokud model s vysokou pravděpodobností vyprodukuje množinu vzorků z trénovací množiny, tak je také **pravděpodobné**, že vyprodukuje **podobné vzorky** které nebyly součástí trénovací množiny (a naopak, je **nepravděpodobné** že vyprodukuje **odlišné vzorky vůči vzorkům v trénovací množině**). (Doersch, 2021)

U variačních autoenkodérů je toto výstupní rozdělení pravděpodobnosti často voleno jako Gaussovo, tedy například $P(X|z\theta) = \mathcal{N}(X|f(z;\theta), \sigma^2 * I)$ Tedy, má střední hodnotu $f(z;\theta)$ a kovarianci rovnu jednotkové matici I krát *nějaká* skalární hodnota σ . (Doersch, 2021)

Obecně (a zejména v rané fázi trénování) model nebude generovat výstupy které jsou *identické* k libovolnému X . V důsledku toho, že u variačního autoenkodéru volíme toto výstupní rozdělení jako Gaussovo, lze provést algoritmus gradientního sestupu (případně jinou optimalizační techniku) za účelem zvýšení $P(x)$ tím, že donutíme $f(z;\theta)$ blížit se k X pro nějaké z (a tím pádem iterativně zvyšovat pravděpodobnost, že model vygeneruje vzorek podobný vzorku z trénovací množiny). (Doersch, 2021)

Důležitou vlastností pro možnost provést tuto optimalizaci tedy je, aby $P(X|z)$ bylo **spočitatelné a spojité v θ** ¹³. (Doersch, 2021)

1.7 Hluboký model využívající latentních proměnných

Model využívající latentních proměnných (*latent variable model*) označíme $p_\theta(\mathbf{x}, \mathbf{z})$.

Jsou-li rozdělení pravděpodobností tohoto modelu parametrisovány neuronovou sítí, nazveme jej **hluboký model využívající latentních proměnných** (*deep latent variable model*, dále jen *DLVM*). DLVM může být podmíněný vůči libovolnému kontextu, tedy $p_\theta(\mathbf{x}, \mathbf{z}|\mathbf{y})$. Důležitou vlastností (a výhodou) DLVM je, že marginální rozdělení p_θ může být velmi komplexní (tedy obsahovat téměř jakékoli závislosti). Tato expresivní vlastnost činí DLVM atraktivní pro **aproximaci složitých skrytých rozdělení pravděpodobnosti** $p^*(\mathbf{x})$. (Diederik P. Kingma a Max Welling, 2019)

Jedním z nejjednodušších (a nejznámějších) DLVM modelů je faktorizace s následující strukturou (Diederik P. Kingma a Max Welling, 2019):

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z}), \quad (1.7)$$

kde $p_\theta(\mathbf{z})$ a $p_\theta(\mathbf{x}|\mathbf{z})$ jsou předem dány.

¹³Výstupní rozdělení nemusí být Gaussovo (ale např. lze využít Alternativní rozdělení)

1.7.1 Latentní proměnné

Latentní proměnné jsou proměnné, které jsou součástí modelu, ale nejsou přímo pozorovatelné (a tím pádem nejsou součástí vstupních dat). Pro označení takových proměnných používáme z . U latentních proměnných se předpokládá jejich vliv na hodnotu cílové proměnné (resp. cílových proměnných). (Diederik P. Kingma a Max Welling, 2019)

1.8 TODO: Normální rozdělení

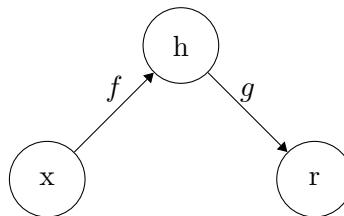
1.9 Pravděpodobnostní modely a variační inference

1.10 Generativní model

1.11 Intractabilities

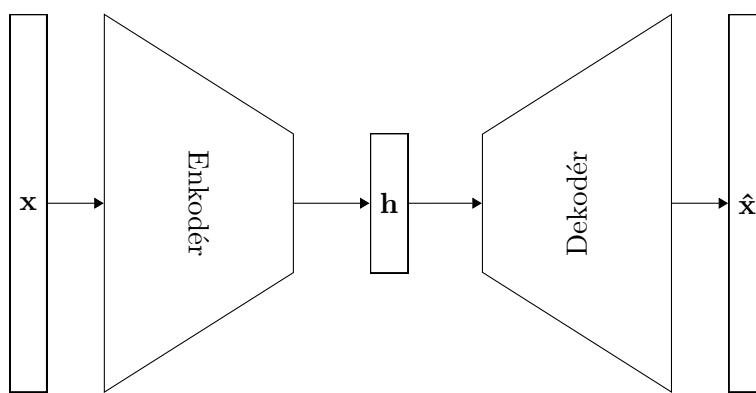
2. Autoenkodér

Autoenkodér je typ umělé neuronové sítě se schopností učit se efektivní reprezentace vstupních dat bez učitele¹. Umělá neuronová síť autoenkodéru má symetrickou strukturu a skrytou vrstvu h , která popisuje kód použitý pro reprezentaci vstupu. Architekturu autoenkodéru (viz Obrázek 2.1) lze principiálně rozdělit na dvě části – kódovací funkci $h = f(x)$, resp. **enkodér** a dekódovací funkci $r = g(h)$, resp. **dekodér**. Hovoríme tedy o typu umělé neuronové sítě s *enkodér-dekodér* moduly. Výstupem enkodéru je **kód** vstupu h . Výstupem dekodéru je **rekonstrukce** vstupu r . (I. Goodfellow et al., 2016)



Obrázek 2.1: Obecná struktura autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h (funkce f). Tento kód je následně dekodérem přetaven na rekonstrukci r (funkce g).

Obecnou strukturu (viz Obrázek 2.1) lze reprezentovat dopřednou umělou neuronovou sítí. Jejím cílem je **rekonstruovat vstupní data na výstupní vrstvě** (tzv. *unsupervised learning objective*). Počet vstupů je tak totožný s počtem neuronů ve výstupní vrstvě umělé neuronové sítě (tedy x a r mají stejnou dimenzi). h může mít *menší* či *větší* dimenzi – volba dimenze h se odvíjí od požadovaných vlastností autoenkodéru. Obecnou architekturu modulů umělé neuronové sítě Autoenkodéru zachycuje Obrázek 2.2. (Charte et al., 2018)



Obrázek 2.2: Jednotlivé moduly architektury umělé neuronové sítě autoenkodéru.

¹V literatuře se můžeme zřídka setkat s zařazením autoenkodérů **obecně** do třídy *semi-supervised* algoritmů strojového učení, například (Chollet, 2017, str. 95). S ohledem na formulaci autoenkodéru představenou v této kapitole je přesnější zařadit **obecnou** architekturu autoenkodérů do třídy algoritmů učení bez učitele. Nutno předeslat, že v architekturách hlubokých autoenkodérů je běžným jevem tzv. *stacked autoenkodér* (viz Podsekce 2.5.1). V instancích tohoto případu pak jednoznačně lze hovořit o zařazení autoenkodéru do třídy semi-supervizovaného učení. (Bengio, Lamblin et al., 2006), (Ranzato et al., 2007), (Erhan et al., 2010)

Autoenkodér je trénován k rekonstrukci jeho vstupů. Pokud by se autoenkodér naučil jednoduše určit $x = g(f(x))$ pro každé x , získali bychom *identitu*, která není patřičně užitečná. Proto je při trénování zavedena řada omezení, jejichž účelem je zabránit možnosti naučení autoenkodéru perfektně kopírovat vstupní data. (I. Goodfellow et al., 2016)

2.1 Historický pohled

Vícevrstvý Perceptron (viz Podsekce 1.2.5) je univerzálním approximátorem – tedy historicky nalézá uplatnění zejména v klasifikačních úlohách učení s učitelem. Sofistikovaný algoritmus se schopností trénování Vícevrstvého Perceptronu s větším počtem skrytých vrstev stále schází, a to zejména v důsledku problému mizejícího gradientu² (Hochreiter, 1998). Až příchod algoritmu gradientního sestupu, který adresuje problém mizejícího gradientu v aplikacích konvolučních sítí (Y. LeCun et al., 1989) (a později i úloh učení bez učitele) značí počátek moderních metod hlubokého učení. V oblasti hlubokého učení tak přirozeně dochází k emergenci a vývoji řady technik pro řešení úloh učení bez učitele. V této kapitole je popsána pouze jedna z nich – architektura umělé neuronové sítě založené na *enkodér-dekodér* modulech: **autoenkodér**. Autoenkodéry byly poprvé představeny jako způsob pro předtrénování umělých neuronových sítí formou automatizované extrakce vlastností (viz Podsekce 1.3.2). Později Autoenkodéry nalézají uplatnění zejména v úlohách redukce dimenzionality (viz Sekce 1.3) či fúzi vlastností (*feature fusion*). (Charte et al., 2018)

Nedávné teoretické propojení autoenkodéru a modelů využívajících latentní proměnných (viz Sekce 1.7) však vedlo ke vzniku zcela nové architektury neuronové sítě kombinující charakter redukce dimenzionality autoenkodéru a pravděpodobnostní modely (viz Sekce 1.4), kterou nazýváme **variační autoenkodér**. To vyneslo autoenkodéry na popředí v oblasti generativního modelování. Představení variačního autoenkodéru je věnována Kapitola 3 a možnostem jeho využití celý zbytek práce.

Byť autoenkodéry vznikly v kontextu hlubokého učení, není pravidlem že všechny modely autoenkodéru obsahují vícero skrytých vrstev. Následuje rozdělení autoenkodérů, mimo jiné, dle struktury jejich umělé neuronové sítě.

2.2 Mělký autoenkodér

Mělký autoenkodér (*shallow autoencoder*), resp. jeho umělá neuronová síť, je sestaven pouze ze tří vrstev – vstupní, kód a výstupní. Je to **nejtriviálnější model autoenkodéru**, jelikož jeho skrytá vrstva (kód) je pouze **jednovrstvá**. (Charte et al., 2018)

²Vanishing gradient problem

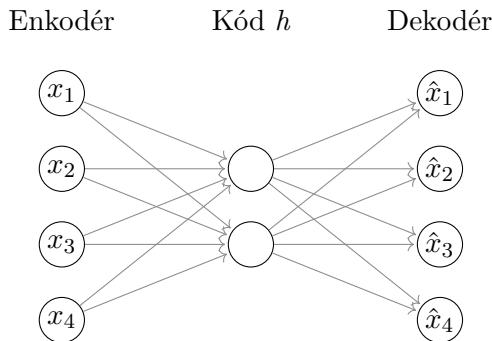
2.3 Autoenkodér s neúplnou skrytou vrstvou

Autoenkodér s neúplnou skrytou vrstvou (*undercomplete autoencoder*) je autoenkodér, jehož dimenze kódu (h) je menší, než dimenze vstupu. Tuto skrytou vrstvu h nazýváme **bottleneck**. Bottleneck je způsob, kterým se autoenkodér s neúplnou skrytou vrstvou učí **kompresované reprezentaci znalostí**. V důsledku bottleneck vrstvy je autoenkodér nucen zachytit pouze salientní vlastnosti trénovacích dat, které následně budou použity pro rekonstrukci. (I. Goodfellow et al., 2016)

Trénovací proces autoenkodéru s neúplnou skrytou vrstvou je popsán jako minimalizace ztrátové funkce:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))), \quad (2.1)$$

kde \mathcal{L} je ztrátová funkce, penalizující $g(f(\mathbf{x}))$ za *rozdílnost* vůči \mathbf{x} (např. *střední kvadratická chyba*). (Charte et al., 2018)



Obrázek 2.3: Jednoduchá architektura umělé neuronové sítě Autoenkodéru s neúplnou skrytou vrstvou. Skrytá vrstva představuje *bottleneck*.

Od Analýzy hlavních komponent po Autoenkodér

Máme-li linéarní dekodér (Autoenkodér používá pouze linéarní aktivační funkce) a jako ztrátová funkce \mathcal{L} je použita *střední kvadratická chyba*, pak se neuplný autoenkodér naučí stejný *vektorový prostor*, který by byl výsledkem Analýzy hlavních komponent (viz Podsekce 1.3.3). V tomto speciálním případě lze ukázat, že autoenkodér trénovaný na úloze kompresované reprezentace znalostí jako vedlejší efekt provedl Analýzu hlavních komponent. (Baldi a Hornik, 1989), (Kamyshanska a Memisevic, 2013)

Důležitým důsledkem tohoto jevu je, že **Autoenkodéry** s nelineární kódovací funkcí f a ne-lineární dekódovací funkcí g jsou schopny učit se obecnější generalizaci než u Analýzy hlavních komponent. (I. Goodfellow et al., 2016)

Na druhou stranu, má-li autoenkodér k dispozici příliš kapacity, může se naučit pouze kopírovat vstup na výstupní vrstvě bez extrakce salientních vlastností (tedy identické zobrazení).

Problém s naučením pouhého identického zobrazení

Extrémním případem je teoretický scénář, ve kterém je Autoenkodér složen z kódu (h) o jedné vrstvě a velmi výkonného enkodéru. Takový Autoenkodér by se mohl naučit reprezentovat každý vstup x_i kódem i . Dekodér by se pak tyto indexy mohl naučit mapovat zpátky na hodnoty konkrétních trénovacích vzorků dat. Tento příklad se v praxi běžně nenaskytne, nicméně jasně ilustruje, jak může Autoenkodér při úloze kopírování vstupu na výstupní vrstvu selhat naučit se užitečné vlastnosti o vstupních datech, jsou-li restrikce při učení příliš nízké. (I. Goodfellow et al., 2016)

Proto je nutné **autoenkodéry regularizovat** (viz Podsekce 1.1.5).

V dalších sekcích (Sekce 2.6 - Sekce 2.9) jsou tedy představeny přístupy k architekturám Autoenkodérů s **využitím regularizace**.

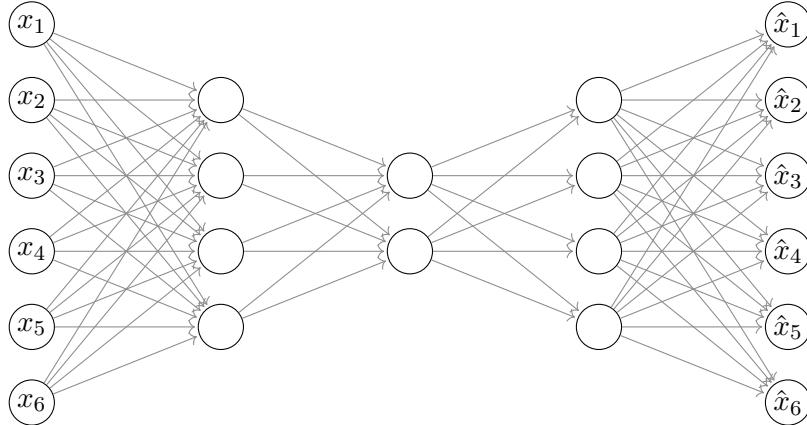
2.4 Autoenkodér s rozšířenou skrytou vrstvou

Autoenkodér s rozšířenou skrytou vrstvou (*overcomplete autoencoder*) je Autoenkodér, jehož počet neuronů ve skrýté vrstvě je větší než počet neuronů vstupní (a výstupní) vrstvy. (Charte et al., 2018)

2.5 Hluboký autoenkodér

V předchozích sekcích (Sekce 2.3, Sekce 2.4) byly představeny autonkodéry s jednovrstvým enkodérem a s jednovrstvým dekodérem. Existují však i autoenkodéry, jejichž enkodér a dekodér moduly jsou vícevrstvé, tedy mají hloubku vyšší než jedna.

Hluboký autoenkodér (*deep autoencoder*), je autoenkodér s netriviální hloubkou skryté vrstvy (*kódu*) \mathbf{h} . (Geron, 2019)



Obrázek 2.4: Schéma umělé neuronové sítě hlubokého autoenkodéru s neúplnou skrytou vrstvou.

Z netriviální hloubky dopředné umělé neuronové sítě plyne Podsekce 1.2.6, který garantuje, že dopředná umělá neuronová síť s alespoň jednou skrytou vrstvou dokáže approximovat (*libovolně přesně*) jakoukoliv funkci (za předpokladu dostatečného počtu neuron skryté vrstvy). (I. Goodfellow et al., 2016)

Nicméně u mělkých enkodérů, které jsou rovněž dopřednou sítí, neexistuje možnost představit libovolná omezení a regularizační prvky (například řídkost *kódu* \mathbf{h} – viz Sekce 2.6). A tedy nelze zamezit problému naučení pouhé identity (viz Sekce 2.3). Na rozdíl od mělkých autoenkoderů (viz Sekce 2.2), však mohou hluboké autoenkodéry (*libovolně přesně*) approximovat jakékoliv mapování vstupu na kód (*opět s předpokladem dostatečného počtu neuronů skryté vrstvy*). (I. Goodfellow et al., 2016), (Charte et al., 2018)

S netriviální hloubkou se rovněž pojí významná redukce výpočetních nákladů spojených s reprezentací některých funkcí. V důsledku možnosti efektivnější reprezentace takových funkcí dochází i k zmenšení nároků na velikost množiny trénovacích dat.

2.5.1 Stacked autoenkodér

Běžnou strategií pro trénování Hlubokého autoenkodéru je hladově předtrénovat (*greedy pre-training*) model individuálním natrénováním většího počtu Mělkých autoenkodérů (předsta-

vených v sekci Sekce 2.2), které jsou následně vloženy za sebe. Takto složený Autoenkodér nazýváme Stacked autoenkovodér. (Geron, 2019)

2.6 Řídký autoenkovodér

Řídká reprezentace dat (*sparsity*) ve strojovém učení znamená, že většina hodnot daného vzorku je nulová. Motivací pro řídkou reprezentaci dat ve strojovém učení je napodobení chování buněk v primární zrakové oblasti ($V1$) mozku savců. Konkrétně schopnosti odhalit a uložit efektivní kódovací strategie pozorovaných vjemů. (Olshausen a Field, 1997)

Pro sestrojení Řídkého autoenkovodérů je tedy nutné představit omezení (regularizační prvek) hodnot aktivací neuronů ve skryté (kódovací) vrstvě \mathbf{h} (resp. počtu aktivních neuronů ve skryté vrstvě).

Řídký autoenkovodér (*sparse autoencoder*) je autoenkovodér, jehož ztrátová funkce je rozšířena o penalizaci řídkosti kódovací vrstvy \mathbf{h} (tzv. *sparsity penalty*) vztahem $\Omega(\mathbf{h})$:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}), \quad (2.2)$$

kde Ω je **regularizační prvek**, jehož cílem je přiblížit hodnoty aktivací neuronů kódovací vrstvy k cílové hodnotě (a zabránit přeúčení). Chceme tak penalizovat neurony kódovací vrstvy, které se aktivují příliš často. (I. Goodfellow et al., 2016)

Běžně lze Ω stanovit následovně. Mějme Bernoulliho náhodnou proměnnou i modelující aktivace neuronů skryté (kódovací) vrstvy – můžou tedy nastat dva stavy: neuron skryté vrstvy je buď aktivován, nebo není aktivován. Pro konkrétní vstup x dostaneme:

$$\hat{p}_i = \frac{1}{|S|} \sum_{x \in S} f_i(x), \quad (2.3)$$

kde $f = (f_1, f_2, \dots, f_c)$, c je počet neuronů skryté (kódovací) vrstvy a \hat{p}_i je průměrná aktivační hodnota neuronu skryté vrstvy (resp. střední hodnota příslušného Bernoulliho schématu). (Charte et al., 2018)

Dále mějme p jako cílové rozdělení aktivací. Kullback-Leibnerova divergence mezi náhodnou proměnnou i a p pak udává rozdíl obou rozdělení:

$$KL(p \parallel \hat{p}_i) = p \log \frac{p}{\hat{p}_i} + (1 - p) \log \frac{1 - p}{1 - \hat{p}_i}. \quad (2.4)$$

Výsledný penalizační prvek Ω pro Řídký autoenkovodér má tedy následující podobu:

$$\Omega_{\text{RAE}}(W, b; S) = \sum_{i=1}^c KL(p \parallel \hat{p}_i), \quad (2.5)$$

kde průměrná hodnota aktivací \hat{p}_i závisí na parametrech enkodéru a množině trénovacích dat S . (Charte et al., 2018)

Přičtením tohoto penalizačního prvku ke ztrátové funkci (a následnou minimalizací celkové ztrátové funkce) je autoenkodér nucen **omezit počet aktivních neuronů v skrýté (kódovací) vrstvě**. V důsledku tohoto omezení pak každý neuron skryté vrstvy reprezentuje nějakou **salientní vlastnost** vstupních dat (a rovněž je zamezeno naučení pouhé identity, viz Sekce 2.3). (I. Goodfellow et al., 2016)

2.7 Denoising autoenkodér

Denoising autoenkodér je autoenkodér, který na vstupu obdrží poškozená vstupní data a při trénování je jeho předpověď originální vstup bez poškození, a ten na výstupní vrstvě vrátit.

Autoenkovéry běžně minimalizují funkci ve tvaru:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))), \quad (2.6)$$

kde L je ztrátová funkce penalizující $g(f(x))$ za odlišnost od \mathbf{x} (např. L^2 norma jejich rozdílů). Jak ale bylo ukázáno v Sekci 2.3, to umožňuje $g \circ f$ naučit se být pouhou identitu.

Z toho důvodu denoising autoenkodér minimalizuje funkci:

$$\mathcal{L}(\mathbf{x}, g(f(\tilde{\mathbf{x}}))), \quad (2.7)$$

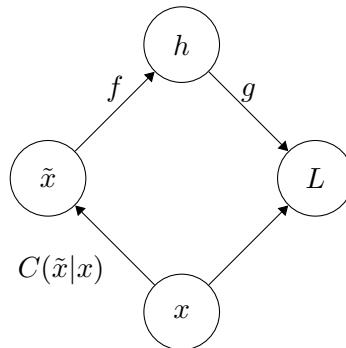
kde $\tilde{\mathbf{x}}$ je kopí \mathbf{x} která byla úmyslně poškozena procesem $C(\tilde{x}|x)$ (*corruption process*), který reprezentuje podmíněné rozdělení pravděpodobnosti poškozených vzorků \tilde{x} v závislosti na vzorku vstupních dat x . (I. Goodfellow et al., 2016)

Denoising autoenkodér se pak učí **rozdělení rekonstrukce** $p_{reconstruct}(\mathbf{x}|\tilde{\mathbf{x}})$, které je odhadnuto z trénovacích dvojic následovně:

1. Zvolit trénovací vzorek \mathbf{x} z množiny trénovacích dat
2. Vygenerovat poškozenou verzi zvoleného vzorku ($\tilde{\mathbf{x}}$) procesem C
3. Použít dvojice $(\mathbf{x}, \tilde{\mathbf{x}})$ jako množinu trénovacích dat pro odhadnutí rozdělení rekonstrukce Denoising autoenkodéru $p_{reconstruct}(\mathbf{x}|\tilde{\mathbf{x}}) = p_{decoder}(\mathbf{x}|\mathbf{h})$, kde $p_{decoder}$ je výstupem funkce dekódéru $g(\mathbf{h})$

Při trénování Denoising autoenkodéru jsou funkce f a g nuceny zachytit implicitní strukturu $p_{data}(\mathbf{x})$. (I. Goodfellow et al., 2016)

Trénovací procedura denoising autoenkodéru schématicky zachycuje Obrázek 2.5:



Obrázek 2.5: Procedura trénování denoising autoenkodéru. Převzato z (I. Goodfellow et al., 2016).

Denoising autoenkovodér se tedy musí naučit toto poškození odstranit a rekonstruovat tak původní vstup (namísto pouhého naučení se identitě).

Denoising Autoenkodéry jsou příkladem hned dvou jevů (Charte et al., 2018):

- Emergence užitečných vlastností o vstupních datech jako výsledek minimalizace chyby rekonstrukce
- Schopnosti modelů s vysokou kapacitou/rozšířenou skrytou vrstvou fungovat jako Autoenkodér, **za předpokladu že je jim zabráněno naučit se identické zobrazení vstupních dat**

2.8 Robustní autoenkovodér

Robustní autoenkovodér (*robust autoencoder*) představuje další způsob, jakým se lze vypořádat s *drobným šumem* ve vstupních datech, která má následně rekonstruovat.

Robustní autoenkodéry, narozdíl od denoising autoenkodéru (viz Sekce 2.7), využívají alternativně definovanou ztrátovou funkci, která je modifikována pro minimalizaci chyby rekonstrukce. Obecně jsou ne-gaussovským šumem ovlivněny **méně než triviální mělké autoenkodéry** (viz Sekce 2.2). Tato alternativní ztrátová funkce je založena na correntropii (*correntropy*, lokalizovaná míra podobnosti), která je v (W. Liu et al., 2006) definována následovně:

$$\mathcal{L}_{MCC}(u, v) = - \sum_{k=1}^d \mathcal{K}_\sigma(u_k - v_k), \quad (2.8)$$

kde

$$\mathcal{K}_\sigma(\alpha) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\alpha^2}{2\sigma^2}\right), \quad (2.9)$$

a σ je parameter kernelu \mathcal{K} .

Correntropie měří hustotu pravdepodobností, že dvě *události* jsou si rovny. Correntropie je **výrazně méně ovlivněna odlehlymi hodnotami**, než např. střední kvadratická chyba. (W. Liu et al., 2006) Robustní autoenkodér se snaží tuto míru maximalizovat, což intuitivně vede k **vyšší odolnosti** Robustního autoenkodéru **na ne-gaussovský šum** přítomný ve vstupních datech. (Charte et al., 2018)

2.9 Contractive autoenkodér

Přehnaná citlivost na *drobné rozdíly* ve vstupních datech by mohla vést k architektuře Autoenkodéru, která pro velmi podobné vstupy generuje odlišné kódy.

Contractive autoenkodér (*CAE*), je Autoenkodér, který je při trénování omezen regularizačním prvkem, který vynucuje aby derivace kódů ve vztahu k jejich vstupu byly co možná nejmenší. Tedy **dva podobné vstupy musí mít vzájemně podobné kódy**. Přesněji je dosaženo lokální invariance na přípustně malé změny vstupních dat. (Rifai et al., 2012)

Citlivost na *drobné rozdíly* ve vstupních datech lze měřit pomocí Frobeniovy maticové normy $\|\cdot\|_F$ Jacobiho matice enkodéru (J_f):

$$\|J_f(x)\|_F^2 = \sum_{j=1}^d \sum_{i=1}^c \left(\frac{\partial f_i}{\partial x_j}(x) \right)^2. \quad (2.10)$$

Čím vyšší je tato hodnota, tím více bude kód nestabilní s ohledem na *drobné rozdíly* ve vstupních datech. Z této metriky je následně sestaven **regularizační prvek** který je připočten k hodnotě ztrátové funkce Contractive Autoenkodéru:

$$\Omega_{CAE}(W, b, S) = \sum_{x \in S} \|J_f(x)\|_F^2. \quad (2.11)$$

Výsledkem je tedy Autoenkodér, jehož dva (lokálně) *podobné* vstupy musejí mít i *podobný* kód. (Charte et al., 2018)

Z contractive autoenkodéru lze rovněž vzorkovat nové výstupy. Jakobián (*Jacobiho determinant*) enkodéru je (jako *drobný šum*) přičten ke kódu vstupu. Takto modifikovaný kód je poté dekodérem přetaven na výstup a dostaváme nový vzorek dat. (I. Goodfellow et al., 2016)

2.10 Stochastický autoenkodér

Struktura stochastického autoenkodéru se vůči obecné struktuře autoenkodéru (Obrázek 2.1) liší reprezentací enkodér a dekodér modulů.

V stochastickém autoenkodéru jsou enkodér a dekodér moduly reprezentovány rozdelením pravděpodobnosti (nejedná se tedy pouze o funkce, ale moduly zahrnují i určitou míru šumu). Výstup těchto modulů tedy obdržíme **výběrem z příslušného rozdělení pravděpodobnosti**. (I. Goodfellow et al., 2016)

Mějme skrytou vrstvu (*kód*) \mathbf{h} . Obecně má pro enkodér toto rozdelení podobu $p_{enkodér}(\mathbf{h}|\mathbf{x})$ a pro dekodér $p_{dekodér}(\mathbf{x}|\mathbf{h})$. Modifikací základní struktury autoenkodéru (Obrázek 2.1) tedy dostáváme:



Obrázek 2.6: Obecná struktura Autoenkodéru. Ze vstupu x je enkodérem vytvořen kód h (funkce f). Tento kód je následně dekodérem přetaven na rekonstrukci r (funkce g).

V tradiční dopředné umělé neuronové síti Podsekce 1.2.3 je běžnou strategií pro návrh výstupní vrstvy definování (*výstupního*) rozdelení pravděpodobnosti $p(\mathbf{y}|\mathbf{x})$. Pro návrh ztrátové funkce pak minimalizace záporného logaritmu věrohodnosti $-\log p(\mathbf{y}|\mathbf{x})$. V takové architektuře je \mathbf{x} vektor vstupních dat a \mathbf{y} vektor cílových proměnných, které se umělá neuronová síť snaží předpovědět (např. štítků jednotlivých tříd). (I. Goodfellow et al., 2016)

S architekturou autoenkodérů se však pojí jeden zásadní rozdíl. V autoenkodéru **je x jak vstupní, tak cílová proměnná**.

Dekodér pak lze interpretovat jako modul poskytující podmíněné rozdelení $p_{dekodér}(\mathbf{x}|\mathbf{h})$. Autoenkodér lze trénovat minimalizací $-\log p_{dekodér}(\mathbf{x}|\mathbf{h})$. Konkrétní podoba ztrátové funkce se odvíjí od požadovaných vlastností autoenkodéru a od přesné podoby modulu dekodéru (pokud hodnoty $\mathbf{x} \in \mathbb{R}$, pak jsou pro parametrizaci normálního rozdělení použity linéarní výstupní jednotky, tedy $-\log p_{dekodér}(\mathbf{x}|\mathbf{h})$ vrací *střední kvadratickou chybu*).

Stochastický autoenkodér tedy **generalizuje kódovací funkci $f(x)$ na kódovací rozdělení pravděpodobnosti $p_{enkodér}(\mathbf{h}|\mathbf{x})$** .

Enkodér a dekodér moduly stochastického autoenkodéru tedy lze považovat za **modely využívající latentní proměnné** (*latent variable models*, viz Sekce 1.7). Model využívající latentní proměnné značíme $p_{model}(\mathbf{h}, \mathbf{x})$. (I. Goodfellow et al., 2016)

Stochastický enkodér je pak definován následovně (I. Goodfellow et al., 2016):

$$p_{enkodér}(\mathbf{h}|\mathbf{x}) = p_{model}(\mathbf{h}, \mathbf{x}) \quad (2.12)$$

a **stochastický dekodér** následovně (I. Goodfellow et al., 2016):

$$p_{dekodér}(\mathbf{x}|\mathbf{h}) = p_{model}(\mathbf{x}, \mathbf{h}) \quad (2.13)$$

Výběr nových dat z rozdělení pravděpodobnosti autoenkodéru je detailněji představuje Kapitola 3.

2.11 Ostatní typy autoenkoderů

Na poli výzkumu strojového učení se autoenkodéry těší velkému úspěchu. Přirozeně tak existuje celá řada architektur a modifikací, které slouží k specifickým účelům. V této kapitole byly představeny pouze ty architektury autoenkodéru, na kterých bude později stavět kapitola Kapitola 4.

V této sekci jsou stručně shrnutý *ostatní* typy autoenkodérů, které nejsou pro předmět nadcházejících kapitol stěžejní, ale obecně se jim dostává vysoké míry využití.

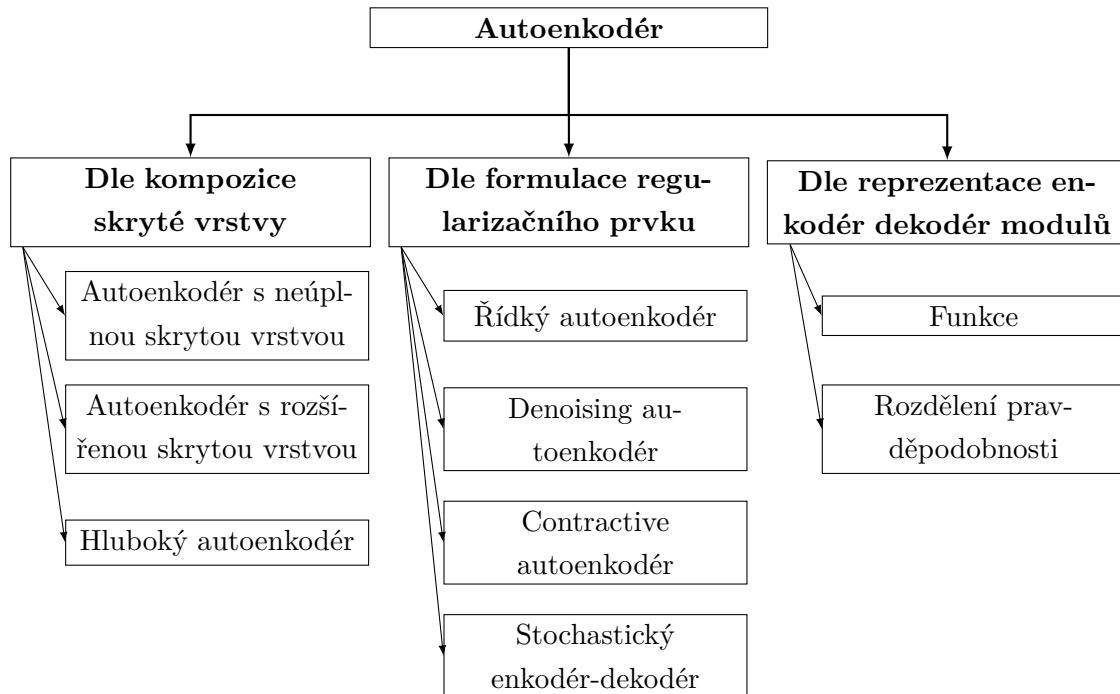
2.11.1 Adversariální autoenkodér

Adversariální autoenkodér (*adversarial autoencoder*) přináší koncept Generativních Adversariálních sítí (I. J. Goodfellow et al., 2014) na pole autoenkodérů. V adversariálním autoenkodéru je kód (\mathbf{h}) modelován uložením předchozího rozdělení pravděpodobnosti (*prior*). Následně je natrénován *běžný* autoenkodér, současně se *diskriminační* síť snaží odlišit kódy modelu od výběru dat z předchozího rozdělení pravděpodobnosti. Jelikož generátor (v tomto případě enkodér) je trénován k *přelstění* diskriminátoru, kódy mají tendenci následovat uložené rozdělení pravděpodobnosti. Tím pádem, z adversariálního autoenkodéru lze **provádět výběr zcela nových dat** (*generovat nové vzorky*). (Charte et al., 2018)

2.12 Taxonomie autoenkodérů

Bylo představeno několik tříd autoenkodérů (rozdělení dle navržení umělé neuronové sítě, rozdělení dle regularizačního prvku) a následně popsáno několik dalších druhů autoenkodérů.

Obrázek 2.7 nabízí jejich (nevyčerpávající) taxonomii. Autoenkodéry jsou zde rozděleny podle představených charakteristik jejich reprezentace a zpracování skryté vrstvy.



Obrázek 2.7: Autoenkovdéry rozděleny dle charakteristik zpracování kódovací vrstvy

2.13 Využití Autoenkovdérů

Představené architektury autoenkovdérů nabízejí řadu využití, mezi které se řadí:

- Mapování vysokorozměrných dat do 2D pro vizualizaci
- Učení se abstraktních vlastností o vstupních datech bez učitele pro následné využití v úlohách učení s učitelem
- Komprese dat

nicméně naráží na zásadní problém nespojitosti naučených manifoldů – tedy (s výjimkou stochastického přístupu, viz Sekce 2.10) nemají příliš dobrý výkon v generativních úlohách. Aplikacím úloh pozorování v latentním prostoru je věnována celá Kapitola 4. Pro jejich realizaci je však nutné představit architekturu **variačního autoenkovdéra** (viz Kapitola 3), která staví na dosud popsaných konceptech.

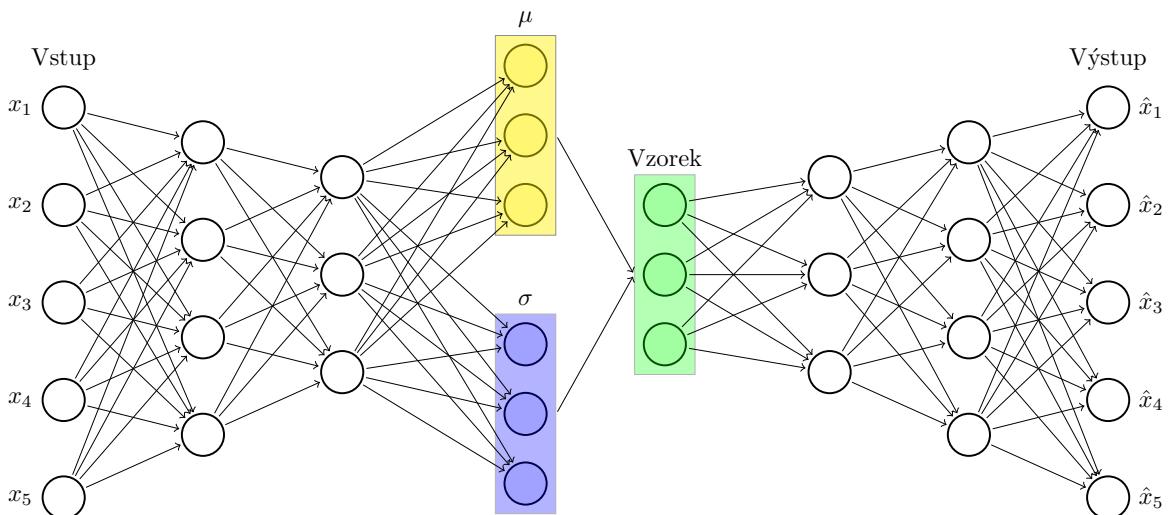
3. Variační autoenkovodér

Generativní modely (Sekce 1.10), učení se reprezentací (Bengio, Courville et al., 2014) a úlohy učení bez učitele (Podsekce 1.1.3) jsou **klíčové oblastí pro vytvoření inteligentních systémů** (Diederik P. Kingma a Max Welling, 2019), (Yann LeCun, 2022). Variační autoenkovodér z těchto principů vychází a propojuje je. Ve snaze o konstrukci takového stroje tak variační autoenkovodér hraje důležitou roli.

Variační autoenkovodér (D. P. Kingma a M. Welling, 2014), (Rezende et al., 2014) (dále jen VAE) je rámec poskytující metodu pro učení víceúčelových hlubokých modelů využívajících latentních proměnných (Sekce 1.7) a příslušných odvozovacích modelů za použití stochastického gradientního sestupu. (Diederik P. Kingma a Max Welling, 2019)

Jednou z hlavních výhod VAE je schopnost transformovat diskrétní prostor po-zorování na spojitý latentní prostor (ze kterého lze následně generovat vzorky, které nebyly součástí trénovací množiny).

VAE tak nalézá širokou škálu aplikací v generativním modelování, učení se reprezentací a úlohách učení se bez učitele (resp. semi-supervizovaných úlohách). Přehled aplikací VAE nabízí Kapitola 4.



Obrázek 3.1: Umělá neuronová síť variačního autoenkovodéru.

3.1 Motivace vzniku: variační autoenkovodér jako generativní model

Velmi aktuálním tématem v odvětví strojového učení je generativní versus diskriminativní modelování. V diskriminativním modelování je cílem naučit se prediktor na základě pozorování. V generativním modelování je cíl poněkud obecnější – naučit se spojité rozdělení pravděpodobnosti skrze všechny proměnné. (I. Goodfellow et al., 2016)

Generativní model simuluje způsob, kterým jsou data generována v reálném světě. *Modelováním* se ve vědních disciplínách rozumí odhalování generujícího procesu stanovením hypotéz a následném testování těchto hypotéz pozorováním. S charakterem generativního modelování je spojena řada užitečných výhod. (I. Goodfellow et al., 2016)

První z nich je možnost zabudování známých fyzikálních zákonů a omezení do samotného generativního procesu – zatímco neznáme (či nepodstatné) *details* můžeme zanedbat formou *šumu*. Výsledný model je pak často vysoce intuitivní a dobře interpretovatelný. (I. Goodfellow et al., 2016)

Dalším důvodem pro snahu o pochopení generativního procesu dat je, že přirozeně zahrnují kauzální vztahy reálného světa. Pozorování a využití těchto kauzálních vztahů nabízí schopnost generalizovat v dosud nepozorovaných situacích (I. Goodfellow et al., 2016)

Zatímco generativní modely se zvládnou učit efektivní reprezentace vstupních dat, mají oproti diskriminativním modelům tendenci činit **silnější předpoklady**, což vede k **vyššímu asymptotickému biasu** pakliže model **chybuje**. (Banerjee, 2007) Pokud nás zajímá pouze naučení se rozlišovat třídy, a náš model chybuje (a každý model *téměř vždy* do určité míry chybuje), pak diskriminativní modely v takové úloze (za předpokladu dostatečného množství dat) často vedou k menší chybovosti. (I. Goodfellow et al., 2016)

I přesto se vyplatí studovat proces generování dat jako způsob, kterým lze trénovací proces diskriminátoru (např. klasifikátoru) zušlechťovat. Typickým scénářem je úloha, kdy máme k dispozici množinu vzorků se štítky a řádově větší množinu vzorků bez štítků. V takové úloze semi-supervizovaného učení lze využít generativního modelu dat k zpřesnění klasifikace. (D. P. Kingma a M. Welling, 2014), (Sønderby et al., 2016)

Generativní modelování může být využito i více obecně. Nad generativním modelováním lze uvažovat jako nad jakousi doprovodnou činností. Například, predikce bezprostřední budoucnosti nám může pomoci při sestavování užitečných abstrakcí o chování světa, které mohou být následně použity pro řadu dalších úloh predikce. Hledání rozmotaných (*disentangled*), sémanticky významných a statisticky nezávislých kauzálních faktorů variací dat je obecně známo pod pojmem učení se reprezentací bez učitele (*unsupervised representation learning*). A **variační autoenkodéry** jsou pro tento účel hojně uplatňovány. (I. Goodfellow et al., 2016)

Na tuto úlohu lze alternativně hledět i jako na implicitní formu regularizace. Nutíme-li vzniklé reprezentace být významnými pro proces generování dat, představujeme bias vůči opačnému procesu (který vstupní data mapuje na neužitečné reprezentace). Doprovodnou činnost predikce bezprostřední budoucnosti tak lze využít k lepšímu porozumění světa v abstraktní úrovni a tím pádem činit přesnější predikce v pozdější fázi. (I. Goodfellow et al., 2016)

3.2 Princip variačního autoenkodéru

Variační autoenkodér lze popsat jako dva provázané, byť **nezávisle parametrizované** modely: **enkodér** (resp. rozpoznávací) model – Sekce 3.4 a **dekodér** (resp. generativní) model – Sekce 3.5. Tyto dva modely se vzájemně podporují. Enkodér model generativnímu modelu dodává aproximaci jeho posteriorních náhodných latentních proměnných, které jsou potřebné pro úpravu jeho parametrů uvnitř iterace *expectation maximization* učení. A opačně, generativní model slouží jako opora pro naučení významných reprezentací vstupních dat enkodér modelem. Tedy, dle Bayesova pravidla, enkodér je *approximate inverse* generativního modelu. (Diederik P. Kingma a Max Welling, 2019)

Jednou z výhod VAE oproti běžné variační inferenci (viz Sekce 1.9, dále jen VI) je, že enkodér model (také nazývaný inferenční model) je nyní stochastickou funkcí vstupních proměnných. Na rozdíl od VI, kde má každý datový bod vlastní variační rozdělení pravděpodobnosti, což je při větším množství dat neefektivní, enkodér používá jednu množinu parametrů pro model vztahů mezi vstupními daty a latentními proměnnými. Tento proces nazýváme amortizovanou inferencí. Takový enkodér model může být libovolně komplexní, ale stále zůstává přiměřeně rychlým, jelikož již z principu může být realizován jedním dopředným průchodem z vstupu skrze latentní proměnné. Nevýhodou ale je, že při vzorkování vzniká v gradientech potřebných pro učení tzv. vzorkovací šum. Možná největším přínosem VAE je řešení tohoto šumu použitím tzv. *reparametizačního triku* – jednoduchou procedurou pro přeorganizování výpočtu gradientů, který snižuje variaci gradientu. (Diederik P. Kingma a Max Welling, 2019)

Variační autoenkodér je inspirován Helmholtz Machine (Dayan et al., 1995), což byl první model který využíval enkodér modelu. Nicméně wake-sleep algoritmus, který byl v návrhu využitý, byl silně neefektivní a neoptimalizoval jednoznačné kritérium. U VAE tedy pravidla pro učení následují jednoznačnou approximaci cíle. (Diederik P. Kingma a Max Welling, 2019)

Variační autoenkodéry jsou spojením pravděpodobnostních grafických modelů a hlubokého učení. Generativní model je tvořen bayesovskou sítí ve tvaru $p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ ¹. Podobně, enkodér model je tvořen podmíněnou bayesovskou sítí ve tvaru $q(\mathbf{z}|\mathbf{x})$ ². Každá podmíněná vrstva může být tvořena komplexní (hlubokou) umělou neuronovou sítí, například: $\mathbf{z}|\mathbf{x} \sim f(\mathbf{x}, \epsilon)$, kde f je mapování umělé neuronové sítě a ϵ je šum (náhodná proměnná). Učící algoritmus VAE je variace klasického *expectation maximization algoritmu*. Ten ale skrze reparametizační trik provádí zpětnou propagaci skrze všechny vrstvy hluboké umělé neuronové sítě (viz Sekce 3.8). (Diederik P. Kingma a Max Welling, 2019)

¹Případně, má-li generativní model více stochastických latentní vrstev, má síť následující hierarchii: $p(\mathbf{x}|\mathbf{z}_L)p(\mathbf{z}_L|\mathbf{z}_{L-1}) \dots p(\mathbf{z}_1|\mathbf{z}_0)$.

²Nebo jako hierarchie, např.: $q(\mathbf{z}_0|\mathbf{z}_1) \dots q(\mathbf{z}_L|X)$.

3.2.1 Notace

- $X = \{x^{(i)}\}_{i=1}^N$: Dataset sestavený z N i. i. d.³ vzorků nějaké spojité či diskrétní proměnné x . Předpokládáme, že tato data byla vygenerována nějakým náhodným procesem, jenž zahrnuje pozorování spojité náhodné proměnné z .
- z : Latentní proměnná (latentní reprezentace), kód.
- $p_\theta(z)$: Apriorní rozdělení s parametry θ . Jeho hustota pravděpodobnosti je diferenciovatelná s ohledem na θ a z .
- $p_\theta(x|z)$: Podmíněné rozdělení s parametry θ . Jeho hustota pravděpodobnosti je diferenciovatelná s ohledem na θ a z .
- Proces náhodného generování dat: Zahrnuje pozorování z . Skládá se ze dvou kroků: (1) hodnota $z^{(i)}$ je vygenerována z $p_\theta(z)$; (2) hodnota $x^{(i)}$ je generována z $P_\theta(x|z)$.
- $q_\phi(z|x)$: *Recognition model* (viz Sekce 3.4). Probabilistický enkodér. Na základě datového bodu x produkuje rozdělení pravděpodobnosti skrze možné hodnoty kódu z , které jej mohly generovat. Aproximace původního *intractable* podmíněného posteriorního rozdělení $p_\theta(z|x)$.
- $p_\theta(x|z)$: Probabilistický dekodér (Sekce 3.5). Na základě kódu z produkuje rozdělení pravděpodobnosti skrze možné hodnoty korespondující s x . Jeho využití je generování nových vzorků dat (viz Sekce 3.10).
- $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$: Generativní model, ze kterého provádíme vzorkování a snažíme se o odhad jeho marginální věrohodnosti.

3.3 Vymezení problémové oblasti

Při řešení Rovnice 1.6 nastávají tři problémy:

1. Jakým způsobem definovat latentní proměnné z . Tedy určit jakou informaci budou reprezentovat.
2. *Intractability*, aneb jak vyřešit integrál skrze z . (D. P. Kingma a M. Welling, 2014, Sekce 2.1.)
3. Velikost datasetu. Provádět dávkovou optimalizaci skrze všechny datové body je příliš drahá operace. Jak dosáhnout parametrických úprav jednotlivých datových bodů⁴ pomocí metod vzorkování.

Na první problém reaguje Podsekce 3.3.1. Na druhý problém reaguje Sekce 3.7. A nákladnost dávkových operací řeší Sekce 3.8. VAE tedy nabízí definitivní všech představených problémů Rovnice 1.6.

³Independent and identically distributed, nezávisle a rovnoměrně rozdělené náhodné veličiny.

⁴Případně *minibatch* dávek.

3.3.1 Volba latentních proměnných dle typu reprezentace informace

Jak zvolit latentní proměnné z které správně zachytí latentní informace obsažené v datech?

Před tím, než model vůbec začne generovat nový vzorek dat (například obrázek ručně psané číslice 0-9) musí provést řadu komplexních rozhodnutí. Volbu samotné číslice, úhel jejího sklonu, tloušťku tuhu a celou řadu dalších stylistických vlastností. Mezi těmito vlastnostmi pochopitelně existují korelace, které model musí rovněž zohlednit⁵. Je tedy zřejmé, že se chceme vyhnout jakémukoliv explicitnímu zahrnutí těchto vlastností do naučeného modelu. To znamená, že nečiníme explicitní rozhodnutí o tom, jaké dimenze z kódují jakou vlastnost dat⁶. Stejně tak se chceme vyhnout jakémukoliv explicitnímu popisu závislostí mezi daty (tzv. popisu jejich latentní struktury) mezi dimenzemi z . (Doersch, 2021)

VAE volí unikátní přístup k řešení výše zmíněných omezení. **VAE předpokládá, že neexistuje žádná triviální interpretace dimenzi z , ale raději usuzuje, že z lze vzorkovat z jednoduchého rozdělení pravděpodobnosti $\mathcal{N}(0, I)$,** kde I je jednotková matice. (Doersch, 2021)

Jak vzorkovat z z rozdělení pravděpodobnosti $\mathcal{N}(0, I)$?

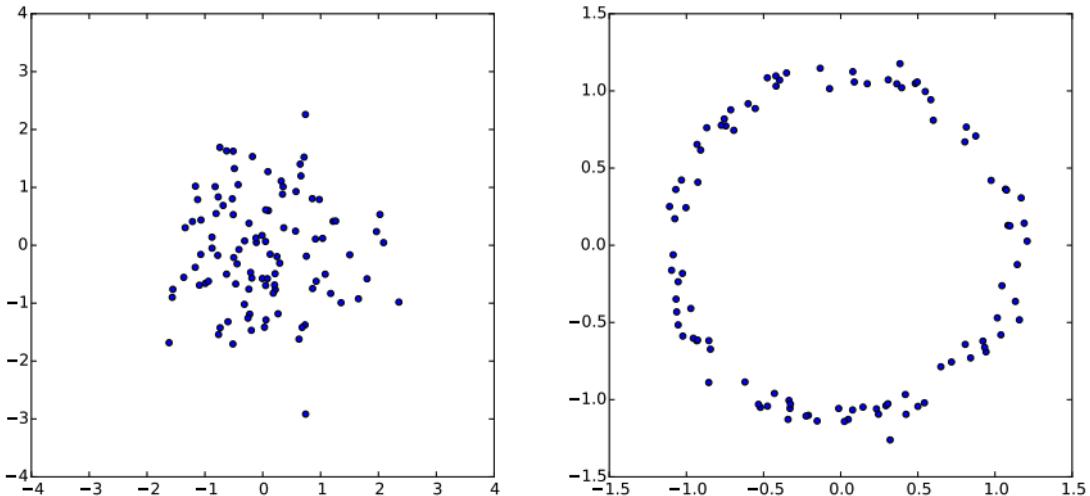
Libovolné rozdělení pravděpodobnosti s d dimenzemi lze generovat množinou o velikosti d proměnných, které mají normální rozdělení a jejich následném mapování skrze dostatečně komplexní funkci⁷.

Řekněme, že chceme sestavit 2D náhodnou proměnou, jejíž hodnoty leží na kruhu. Tedy z je 2D a má normální rozdělení. Pak funkce $g(z) = \frac{z}{\|z\|} + \frac{z}{\|z\|}$ téměř tvoří kruh. Obrázek 3.2 ukazuje z a graf funkce g . Deterministická funkce g naučená ze vstupních dat je naprostě stejná strategie, kterou VAE požívá pro vytvoření distribuce generující nové vzorky dat. (Doersch, 2021)

⁵Např. číslice napsané v rychlosti budou mít typicky vyšší úhel sklonu, ale menší tloušťku tuhu.

⁶Byť určité rozšířené architektury využívají explicitní volby určitých dimenzi z – např. (Kulkarni et al., 2015).

⁷Pro detailní popis generování rozdělení o d dimenzích odkazuji na *inversion method* popsanou v (Devroye, 1986).



Obrázek 3.2: Máme-li náhodnou proměnnou z s nějakým rozdělením pravděpodobnosti, můžeme z ní vytvořit zcela novou náhodnou proměnnou $X = g(z)$ s kompletně jiným rozdělením. Levý obrázek zachycuje vzorky z Gaussova rozdělení. Pravý obrázek zachycuje ty stejné vzorky mapované skrze funkci $g(z) = \frac{z}{10} + \frac{z}{\|z\|}$. Obrázek včetně interpretace převzaty z (Doersch, 2021).

Tedy, má-li VAE k dispozici dostatečně silnou approximační funkci, může se ze vstupních dat jednoduše naučit funkci, která mapuje nezávislé hodnoty z s normálním rozdělením na libovolné latentní proměnné, které jsou pro model potřebné. VAE pak takové latentní proměnné mapuje na X . (Doersch, 2021)

Připomeňme, že (z Podsekce 1.6.1) $P(X|z; \theta) = \mathcal{N}(X|f(z; \theta), \sigma^2 * I)$. Pokud je $f(z; \theta)$ Vícevrstvý Perceptron (viz Podsekce 1.2.5), pak si lze intuitivně představit, že jeho umělá neuronová síť využívá svých prvních pár vrstev k mapování normálně rozdělených z na latentní hodnoty (jako např. identita číslice, tloušťka tahu, sklon číslice apod.). Pozdější vrstvy této sítě mohou být využity k mapování těchto latentních proměnných na obrázek ručně psané číslice (a to sice zcela nově vygenerovaných z pravděpodobnostního rozdělení, viz Sekce 3.10). Důležité je, že (obecně) nemusíme řešit zda-li taková latentní struktura ve vstupních datech vůbec existuje. Pokud nějaká latentní struktura pomáhá modelu s vysokou mírou přesnosti rekonstruovat (tedy optimalizovat ztrátovou funkci, viz Sekce 3.7) data z trénovací množiny, pak se umělá neuronová síť tuto strukturu v *nějaké*⁸ vrstvě naučí. (Doersch, 2021)

⁸Za předpokladu dostatečně vysoko-kapacitní umělé neuronové sítě.

3.4 Enkodér modul

V Sekce 1.7 byly představeny hluboké modely využívající latentních proměnných (DLVM). U těchto modelů je problém provést odhad log-likelihood a rozdelení posteriorní pravděpodobnosti. (Diederik P. Kingma a Max Welling, 2019)

Rámec VAE poskytuje výpočetně efektivní způsob, kterým lze DLVM společně optimalizovat s jejich korespondujícími inferenčními modely pomocí stochastického gradientního sestupu⁹. (Diederik P. Kingma a Max Welling, 2019)

Enkodér modul VAE je pravděpodobnostní enkodér $q_\phi(\mathbf{z}|\mathbf{x})$, který na základě datového bodu \mathbf{x} produkuje rozdelení pravděpodobnosti (typicky Gaussovo) skrze všechny možné hodnoty kódu \mathbf{z} , z něhož tento datový bod teoreticky mohl být vygenerován. (Diederik P. Kingma a Max Welling, 2019)

Posteriorní inference a úlohy učení DLVM jsou efektivně neřešitelné problémy. VAE **představuje parametrický inferenční model** (*parametric inference model*) $q_\phi(\mathbf{z}|\mathbf{x})$ pro přetavení těchto problémů na efektivně řešitelné. Tento model také nazýváme **enkodér** či **rozpoznávací model**. ϕ označuje parametry tohoto inferenčního modelu, tyto parametry nazýváme **variační parametry**. (Diederik P. Kingma a Max Welling, 2019)

Variační parametry ϕ optimalizujeme tak, aby platila approximace (D. P. Kingma a M. Welling, 2014):

$$q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x}) \quad (3.1)$$

Tato approximace posteriorního rozdělení pomáhá k optimalizaci *marginal likelihood*.

Stejně jako u DLVM, inferenční model (enkodér) může být (téměř) libovolný orientovaný pravděpodobnostní grafický model (Diederik P. Kingma a Max Welling, 2019):

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_1, \dots, \mathbf{z}_M|\mathbf{x}) = \prod_{j=1}^M q_\phi(\mathbf{z}_j|Pa(\mathbf{z}_j), \mathbf{x}) \quad (3.2)$$

kde $Pa(\mathbf{z}_j)$ je množina rodičovských proměnných k proměnné \mathbf{z}_j v orientovaném grafu. Podobně jako u DLVM, rozdelení pravděpodobnosti $q_\phi(\mathbf{z}|\mathbf{x})$ může být parametrisováno použitím umělé neuronové sítě (viz Obrázek 3.1). V takovém případě variační parametry ϕ zahrnují váhy a biasy této umělé neuronové sítě (D. P. Kingma a M. Welling, 2014):

⁹Společně optimalizovat (*joint optimization*) zde znamená, že generativní model (který mapuje latentní proměnná na data) a inferenční model (který mapuje data na latentní proměnné) jsou optimalizovány zároveň (raději než optimalizovány nezávisle). Ztrátová funkce, která je při trénování optimalizována, zahrnuje oba modely. Touto společnou optimalizací se VAE mohou učit generovat nové vzorky dat, které mají podobné vlastnosti jako data v trénovací množině a odvozovat skryté latentní vztahy proměnných, která generovala pozorovaná data

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EnkodérSít}_\phi(\mathbf{x}) \quad (3.3)$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \quad (3.4)$$

Pro posteriorní inferenci skrze všechna vstupní data je použita vždy jedna stejná enkodér neuronová síť, tedy **variační parametry jsou sdíleny**¹⁰. Tato strategie, kterou VAE využívá pro **sdílení variačních parametrů skrze všechny datové vstupy** se nazývá **amortizovaná variační inference** (Gershman a Goodman, 2014). Amortizovanou inferencí se VAE vyhýbají optimalizační smyčce pro každý datový bod, a mohou tak plně využít možnosti efektivity stochastického gradientního sestupu. (Diederik P. Kingma a Max Welling, 2019)

3.5 Dekodér modul

Dekodér modul VAE je pravděpodobnostní dekadér $p_\theta(\mathbf{x}|\mathbf{z})$, který na základě *kódu* \mathbf{z} produkuje rozdělení pravděpodobnosti skrze všechny možné hodnoty, kterých může \mathbf{x} nabývat. Aproximace posteriorního rozdělení generativního modelu původních dat $p_\theta(x, z)$. (D. P. Kingma a M. Welling, 2014)

¹⁰V kontrastu s tradičními metodami pro variační inferenci, kde jsou parametry iterativně optimalizovány pro každý datový vstup

3.6 Evidence Lower Bound

Účelovou funkcí VAE je *evidence lower bound*, dále jen ELBO (alternativně se lze setkat s pojmenováním *variační dolní mez*). Typicky je ELBO odvozena pomocí Jensenovy nerovnosti (Wasserman, 2013, Sekce 4.2). Autoři VAE (D. P. Kingma a M. Welling, 2014) však využívají alternativního postupu, jenž se chytře vyhýbá použití Jensenovy nerovnosti a nabízí větší míru *tightness*¹¹. Pro libovolnou konfiguraci inferenčního modelu $q_\phi(z|x)$, včetně volby variačních parametrů ϕ , dostáváme (D. P. Kingma a M. Welling, 2014, Rovnice 3):

$$\mathcal{L}(\theta, \phi, x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)}|z)] - \mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \| p_\theta(z)) \quad (3.5)$$

Rovnice 3.5 je pro **variační autoenkodéry naprosto stěžejní** a slouží jako jádro pro jejich optimalizaci¹². (Doersch, 2021) Schématické zobrazení Rovnice 3.5 nabízí Obrázek 3.3.



Obrázek 3.3: VAE se učí stochastické mapování mezi pozorovaným prostorem x , jehož empirické rozdělení pravděpodobnosti $q_D(x)$ je typicky velmi komplikované, a latentním prostorem z , jehož rozdělení pravděpodobnosti je typicky relativně jednoduché (např. kulovité, jako na tomto obrázku). Generativní model se učí rozdělení pravděpodobnosti $p_\theta(x, z)$ (které lze faktORIZOVAT jako $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$), s apriorním rozdělením skrze latentní prostor $p_\theta(z)$, a stochastický dekodér $p_\theta(x|z)$. Stochastický enkodér $q_\phi(z|x)$ approximuje původní (ale efektivně neřešitelné) posteriorní rozdělení $p_\theta(z|x)$ generativního modelu. Obrázek včetně interpretace převzat z (Diederik P. Kingma a Max Welling, 2019).

¹¹Koncept z matematické teorie míry, který lze intuitivně popsat jako "sadu mér které příliš rychle do nekonečna". (Topsøe, 1974)

¹²Historicky, matematický aparát Rovnice 3.5 byl známý dlouho před formalizací VAE. Například Helmholtz Machines (Dayan et al., 1995) využívají defacto identickou rovnici, ale s jedním zásadním rozdílem. Integrál v středních hodnotách VAE je v (Dayan et al., 1995, Rovnice 5) nahrazen sumou, jelikož Helmholtz Machines usuzují diskrétní rozdělení pravděpodobnosti pro latentní proměnné. A přesně tato volba znemožňuje transformaci, kterou VAE využívají k tomu, aby byl gradientní sestup ELBO efektivně řešitelný (tractable). (Doersch, 2021)

3.6.1 Odvození rovnice

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] \quad (3.6)$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \right] \right] \quad (3.7)$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right] \right] \quad (3.8)$$

$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \right]}_{\Leftrightarrow \mathcal{L}_{\theta, \phi}(\mathbf{x})(ELBO)} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right] \right]}_{\Leftrightarrow D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x}))} \quad (3.9)$$

(3.10)

Druhý prvek Rovnice 3.10 je Kullback-Lieblerova divergence (Sekce 1.5) mezi $q_\phi(\mathbf{z}|\mathbf{x})$ (en-kodérem) a $p_\theta(\mathbf{z}|\mathbf{x})$ (dekodérem), jejíž hodnota je **nezáporná**¹³. Je-li hodnota této KL divergence nulová $\Leftrightarrow q_\phi(\mathbf{z}|\mathbf{x})$ je rovno posteriorní distribuci původních dat (*perfektně je rekonstruuje*). (Diederik P. Kingma a Max Welling, 2019)

První prvek Rovnice 3.10 je variační dolní mez (ELBO):

$$\mathcal{L}_{\theta, \phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \quad (3.11)$$

S ohledem na nenulovost KL divergence je tedy ELBO **dolní mezí**¹⁴ log-likelihood původních dat. (D. P. Kingma a M. Welling, 2014), (I. Goodfellow et al., 2016) Dle Rovnice 3.11 platí:

$$\mathcal{L}_{\theta, \phi}(\mathbf{x}) = \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) \quad (3.12)$$

$$\leq \log p_\theta(\mathbf{x}) \quad (3.13)$$

V tomto případě tedy pozoruhodně tato KL divergence určuje hned dvě *vzdálenosti* (Diederik P. Kingma a Max Welling, 2019):

1. KL divergenci approximace posteriorního rozdělení od původního posteriorního rozdělení (z definice KL divergence)
2. *Mezeru* mezi ELBO $\mathcal{L}_{\theta, \phi}(\mathbf{x})$ a *marginal likelihood* $\log p_\theta(\mathbf{x})$. Tuto *mezeru* také nazýváme mírou *tightness* této meze (\implies vyhnutí se použití Jensenově nerovnosti). Čím lépe $q_\phi(\mathbf{z}|\mathbf{x})$ approximuje $p_\theta(\mathbf{z}|\mathbf{x})$, s ohledem na KL divergenci, tím menší tato *mezera* je.

¹³ $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})) \geq 0$

¹⁴Odtud *evidence lower bound*

3.6.2 Optimalizační cíle

Z Rovnice 3.5 plynou důsledky maximalizace ELBO $\mathcal{L}_{\theta,\phi}$ s ohledem na parametry θ a ϕ . A to sice současná optimalizace dvou kritérií, která jsou pro výkonnost VAE stežejní (Diederik P. Kingma a Max Welling, 2019):

1. *Přibližná* maximalizace marginální věrohodnosti $p_{\theta}(\mathbf{x})$, což implikuje *zlepšení* generativního modelu¹⁵.
2. Minimalizace KL divergence aproximace mezi $q_{\phi}(\mathbf{z}|\mathbf{x})$ a původní posteriorní distribucí $p_{\theta}(\mathbf{z}|\mathbf{x})$. Tedy $q_{\phi}(\mathbf{z}|\mathbf{x})$ (enkodér) se *zlepší*.

3.7 Optimalizace pomocí stochastického gradientního sestupu

Za účelem optimalizace pravé strany Rovnice 3.5 je nutné specifikovat tvar, kterého bude $q_{\phi}(z|x)$ nabývat.

Zvolíme $q_{\phi}(z|x) = \mathcal{N}(z|\mu(X;\theta), \Sigma(X;\theta))$, kde μ a Σ jsou libovolné **deterministické** funkce s parametry θ , které se lze učit z dat. Funkce μ a Σ jsou (typicky) implementovány umělou neuronovou sítí, s tím že Σ musí být reprezentována diagonální maticí (viz Rovnice 3.4). Takto zvolené $Q(z|X)$ má zejména **výpočetní výhodu** – pravá strana rovnice je nyní jednoznačně spočitatelná. (Doersch, 2021)

Nyní je tedy poslední prvek pravé strany, $\mathcal{D}_{KL}[Q(z|X) \parallel P(z|X)]$, KL divergence mezi dvěma vícerozměrnými Gaussovými rozděleními. Takovou KL divergenci lze spočítat v uzavřeném tvaru následovně (Doersch, 2021):

$$\mathcal{D}_{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \parallel \mathcal{N}(0, I)] = \frac{1}{2} \left[\text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log \left(\frac{\det \Sigma_1}{\det \Sigma_0} \right) \right] \quad (3.14)$$

kde k je dimenzionlita výsledného rozdělení pravděpodobnosti. Rovnice 3.14 lze dále zjednodušit následovně (Doersch, 2021):

$$\mathcal{D}_{KL}[\mathcal{N}(\mu(X), \Sigma(X)) \parallel \mathcal{N}(0, I)] = \frac{1}{2} \left[\text{tr}(\Sigma(X)) + (\mu(X))^T (\mu(X)) - k - \log \det(\Sigma(X)) \right]. \quad (3.15)$$

První prvek pravé strany Rovnice 3.5 je o něco komplikovanější.

Bylo by možné pomocí vzorků odhadnout $\mathbb{E}_{q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)}|z)]$, ale pro získání *věrohodného* odhadu je nutné skrze z poslat velké množství vzorků, což je výpočetně náročné. Tedy,

¹⁵Tedy *věrohodností* rekonstrukce původních dat

jak je u stochastického gradientního sestupu běžné, využijeme pouze jeden vzorek z a použijeme $P(X|z)$ pro toto z jako approximaci této střední hodnoty¹⁶. (Doersch, 2021)

Tedy **rovnice kterou chceme optimalizovat** má následující podobu (D. P. Kingma a M. Welling, 2014):

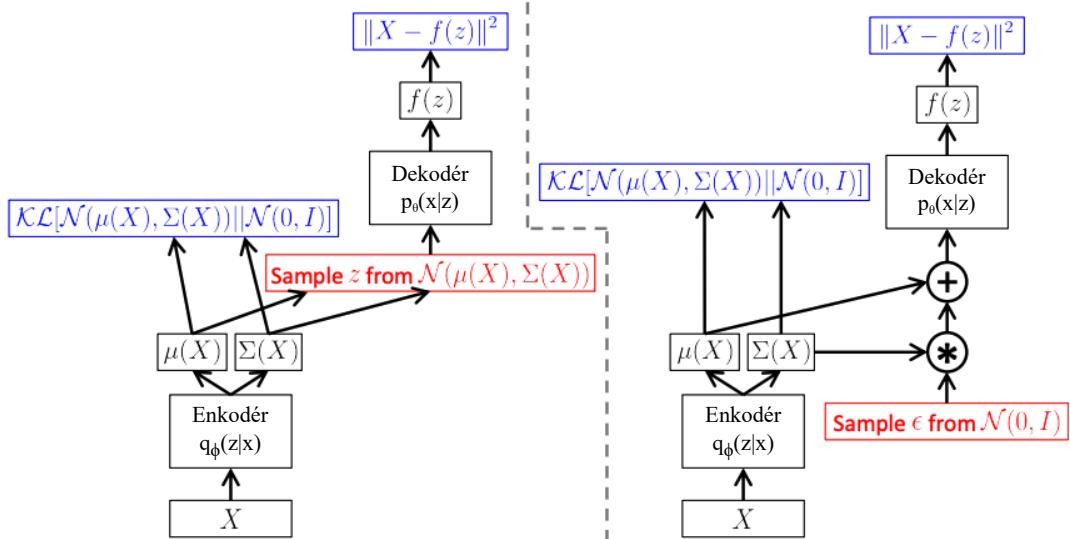
$$\nabla_{\phi} \mathcal{L}_{\theta, \phi}(x) = \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x, z) - \log q_{\phi}(z|x)] \quad (3.16)$$

$$\neq \mathbb{E}_{q_{\phi}(z|x)} [\nabla_{\phi} (\log p_{\theta}(x, z) - \log q_{\phi}(z|x))] \quad (3.17)$$

Symbol gradientu této může být bezpečně přesunut do střední hodnoty. Tedy, můžeme vzorkovat jednu hodnotu z X a jednu hodnotu ze z a následně vypočítat gradient (Doersch, 2021):

$$\log p_{\theta}(x|z) - D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p_{\theta}(\mathbf{z})). \quad (3.18)$$

Průměr gradientu této funkce skrze *libovolně velké* množství vzorků z x a z . Výsledná hodnota tohoto gradientu konverguje ke gradientu Rovnice 3.16. (Doersch, 2021)



Obrázek 3.4: Trénovací fáze dopředné umělé neuronové sítě VAE. Levý obrázek je znázorněn bez reparametizačního triku. Pravý obrázek zahrnuje reparametizační trik. Červené bloky zobrazují vzorkovací operace, které jsou nediferenciovatelné. Modré bloky zobrazují ztrátové vrstvy neuronové sítě. Dopředné chování obou sítí je **identické**, ale pouze v síti na pravém obrázku lze provést zpětná propagace. Obrázek včetně interpretace převzat z (Doersch, 2021).

¹⁶Zde je výhodou, že provést stochastický gradientní sestup skrze všechny různé hodnoty X vzorkované z datové sady D již stochastický gradientní sestup provádíme při trénování.

3.7.1 Metoda stochastická gradientní optimalizace ELBO

Důležitou vlastností ELBO je možnost *joint* optimalizace s ohledem na veškeré její parametry – ϕ a θ – za použití stochastického gradientního sestupu (*stochastic gradient descent, SGD*). Proces lze zahájit náhodnými počátečními hodnotami ϕ a θ a stochasticky optimalizovat jejich hodnoty než dojde ke konvergenci. (Diederik P. Kingma a Max Welling, 2019)

Mějme nezávisle a rovnoměrně rozdělená vstupní data, účelovou funkci ELBO je součet ELBO jednotlivých datových bodů (Diederik P. Kingma a Max Welling, 2019):

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \mathcal{L}_{\theta,\phi}(x) \quad (3.19)$$

ELBO jednotlivých datových bodů a jejich gradienty jsou *obecně* efektivně řešitelné. Dokonce existují estimátory bez biasu, za pomocí kterých lze provést minibatch SGD.

Gradienty ELBO bez biasu s ohledem na parametry generativního modelu θ lze jednoduše získat (viz rovnice 2.14 - 2.17 z (Diederik P. Kingma a Max Welling, 2019)).

Spočítat gradienty ELBO bez biasu s ohledem na variační parametry ϕ je již složitější, jelikož tato ELBO je uvažována s ohledem na rozdělení pravděpodobnosti $q_\phi(z|x)$, tedy je funkci ϕ . Nerovnost gradientů ELBO viz rovnice 2.18 - 2.19 z (Diederik P. Kingma a Max Welling, 2019).

V případě **spojitých** latentních proměnných lze využít tzv. **reparametrizačního triku** pro výpočet odhadů bez biasu $\nabla_{\theta,\phi}\mathcal{L}_{\theta,\phi}(x)$. Tento stochastický odhad umožňuje optimalizovat ELBO za SGD (viz Algoritmus 1)¹⁷. Pseudokód algoritmu převzat z (Diederik P. Kingma a Max Welling, 2019).

Algoritmus 1: Metoda stochastická gradientní optimalizace ELBO

Data:

- \mathcal{D} : Dataset (i. i. d.)
- $q_\phi(z|x)$: Inferenční model (enkodér)
- $p_\theta(x, z)$: Generativní model (dekodér)

Result:

- θ, ϕ : Naučené parametry
 - $\theta, \phi \leftarrow$ Náhodná inicializace parametrů
 - while** SGD nezkonvergoval **do**
 - $\mathcal{M} \sim \mathcal{D}$ (Náhodný minibatch vstupních dat)
 - $\epsilon \sim p(\epsilon)$ (Náhodný šum pro každý datový bod v \mathcal{M})
 - Vypočítat $\tilde{\mathcal{L}}$ and její gradienty $\nabla_{\theta,\phi}(\mathcal{M}, \epsilon)$
 - Upravit θ a ϕ za použití SGD optimizéru - end**
-

¹⁷Existují i metody pro případ **diskrétních** latentních proměnných, které však pro předmět práce nejsou stežejní. Pro jejich představení odkazují na (Diederik P. Kingma a Max Welling, 2019, Sekce 2.9.1.).

Aby VAE fungovaly dle očekávání, $q_\phi(z|x)$ musí být nutno produkovat takové kódy pro x , které bude $p_\theta(x, z)$ schopno **spolehlivě** dekódovat. Na tento problém lze alternativně pohlížet interpretací Rovnice 3.16 jakožto sítě, kterou zobrazuje Obrázek 3.4(a). Dopředný průchod této sítě funguje bez problémů – a je-li její výstup zprůměrován skrze mnoho vzorků x a z , produkuje správnou střední hodnotu. Nicméně, je nutné mít schopnost zpětně propagovat chybu skrze vrstvu, která vzorkuje z z $q_\phi(z|Xx)$, což je **nespojitá operace** a tedy nemá žádný gradient. Stochastický gradientní sestup se zpětnou propagací zvládne stochastické vstupy, ale **neumí pracovat se stochastickými jednotkami (neurony) sítě**.

Řešení tohoto problému nazýváme **reparametrizační trik**.

3.8 Reparametrizační trik

Reparametrizační trik¹⁸ spočívá v přesunutí procesu vzorkování do vstupní vrstvy. (Doersch, 2021)

Mějme $\mu(X)$ a $\Sigma(X)$ – **průměr a kovarianci** $Q(z|X)$. Pak je možné vzorkovat z $\mathcal{N}(\mu(X), \Sigma(X))$ – nejprve provedeme vzorkování z $\epsilon \sim \mathcal{N}(0, I)$ a následně spočteme $z = \mu(X) + \Sigma^{\frac{1}{2}}(X) * \epsilon$. (Doersch, 2021)

Tedy finální rovnice, jejíž gradient chceme spočítat má následující tvar:

$$\mathbb{E}_{X \sim D} \left[\mathbb{E}_{\mathcal{N}(\epsilon; 0, 1)} \left[\log P(X|z = \mu(X) + \Sigma^{\frac{1}{2}}(X) * \epsilon) \right] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})) \right]. \quad (3.20)$$

Kýžená vlastnost Rovnice 3.20 je, že v modelu její **umělé neuronové sítě lze provést zpětnou propagaci**¹⁹. Toto je znázorňuje Obrázek 3.4(b). (D. P. Kingma a M. Welling, 2014)

3.9 Tří různé pohledy na interpretaci naučeného modelu

Jak bylo ukázáno, proces učení variačních autoenkovodérů je efektivně řešitelný problém (*tractable*).²⁰

¹⁸Reparametrizační trik *funguje* pouze pakliže lze vzorkovat z $Q(z|X)$ vyhodnocením funkce $h(\eta, X)$, kde η je šum (z distribuce jejíž parametry nejsou učeny z dat). h také musí být spojitá na X , abychom skrze něj mohli provádět zpětnou propagaci. To, mimo jiné, znamená, že $Q(z|X)$ a tím padem i $P(z)$ **nemohou být diskrétní rozdelení**. V opačném případě by došlo k nespojitosti prostoru vzorků Q – a tedy neschopnosti generativního modelu generovat vzorky v celém rozsahu, včetně vzorků které nebyly součástí dat (běžný problém autoenkovodérů, které uvedla Kapitola 2.)

¹⁹Žádné střední hodnoty (\mathbb{E}) Rovnice 3.20 **nezávisí** na parametrech modelu. Tedy do těchto středních hodnot můžeme bezpečně převést symbol gradientu a zároveň dodržet jejich rovnost. Tedy, mějme neměnné X a ϵ , pak je tato funkce **spojitá a deterministická** v parametrech P a Q , což v důsledku znamená možnost zpětné propagace vypočítat gradient algoritmem stochastického gradientního sestupu.

Při učení je optimalizován $\log P(X)$ skrze celou množinu vstupních dat D . Nicméně, není optimalizován přesně $\log P(X)$, ale pouze jeho odhad.

Tato sekce slouží pro odkrytí mechanismů, které se skutečně na pozadí účelové funkce variacního autoenkodéru dějí. Nabízí tři různé pohledy, kterými může být Rovnice 3.5 interpretována:

1. Velikost chyby, kterou způsobuje současná optimalizace $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ dodatečně vedle optimalizace $\log p_\theta(x^{(i)})$.
2. Interpretace v kontextu informační teorie a propojení s dalšími přístupy založenými na Minimal Description Length.
3. VAE a regularizační prvek. Existuje u variačních autoenkodérů regularizační prvek, obdobně jako u autoenkodérů které uvedla Kapitola 2?

3.9.1 Chyba $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$

Efektivní řešitelnost modelu variačního autoenkodéru závisí na předpokladu, že $Q(z|X)$ lze modelovat jako **Gaussovou funkci se střední hodnotou $\mu(X)$ a rozptylem $\Sigma(X)$** . (Doersch, 2021)

Rozdělení $P(X)$ konverguje k původnímu rozdělení (které generuje vstupní data) pouze když se $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ limitně blíží k nule. Což ale nelze jednoduše zaručit. Ani za předpokladu, že $\mu(X)$ a $\Sigma(X)$ jsou vysoko-kapacitní umělé neuronové sítě²⁰ neplatí, že posteriorní rozdělení $P(z|X)$ musí být vždy Gaussovo (pro libovolnou funkci f která je použita pro definování P). Tedy, pro neměnné P to znamená, že $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ **nikdy není rovno nule** (pokud by tato KL divergence byla rovna nule, viz Rovnice 3.10, znamenalo by to perfektní rekonstrukci původních dat). (Doersch, 2021)

Nicméně, máme-li dostatečně vysoko-kapacitní neuronové sítě, existuje mnoho funkcí f (viz Podsekce 1.2.6), které zaručí, že naučený model generuje libovolné výstupní rozdělení pravděpodobnosti²¹. Libovolná z těchto funkcí maximalizuje $\log P(X)$ stejně dobře. Tedy stačí vybrat jednu z těchto funkcí, která maximalizuje $\log P(X)$ a **zároveň** zaručí, že $P(z|X)$ je Gaussovou funkcí pro všechna $x^{(i)}$. Pokud toto platí, $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$, tlačí model směrem k parametrizaci původní distribuce. (Doersch, 2021)

Zbývá zodpovědět, zda-li taková funkce existuje pro všechny libovolné distribuce, které bychom mohli chtít approximovat. Tento problém (v kontextu variačních autoenkodérů) zatím zůstává nezodpovězen. Nicméně existuje alespoň formální důkaz **nulové chyby approximace** VAE alespoň v triviálním teoretickém scénáři na 1D problému (Doersch, 2021, Příloha A). Autoři rámce VAE se domnívají, že budoucí teoretický výzkum by měl být schopný na tomto důkazu

²⁰Vysoko-kapacitní neuronové sítě, jsou sítě s vysokým počtem parametrů a vah, které jim umožňují učit se komplexním vztahům mezi daty. (I. Goodfellow et al., 2016, Kapitola 5)

²¹Tedy je schopen generovat vzorky *dostatečně podobné* vzorkům množiny trénovacích dat, včetně vzorků, které nebyly při trénování k dispozici.

stavět a rozšířit jej na více (složitějších) scénářů s praktickým využitím. (Doersch, 2021)

3.9.2 Interpretace z pohledu informační teorie

Dalším možným (a velmi důležitým) pohledem na pravou stranu Rovnice 3.5 je v kontextu informační teorie. Konkrétně pomocí principu tzv. *minimum description length*, který stojí za mnoha předchůdci variačního autoenkodéru, jako například Helmholtz Machines, Wake-Sleep Algoritmus, Deep Belief Nets a Boltzmann Machines. (Doersch, 2021)

Na $-\log P(X)$ lze pohlížet jako na celkový počet bitů, které naučený model potřebuje k sestavení daného $x^{(i)}$ za použití **ideálního** kódování. Pravá strana Rovnice 3.5 na toto pohlíží jako na dvoufázový proces pro sestavení $x^{(i)}$. (Doersch, 2021)

V první fázi použijeme *nějaký počet* bitů pro sestavení z ²². Konkrétně $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ je očekávaná informace potřebná pro převedení vzorku z $P(z)$ (bez informace) na vzorek z $Q(z|X)$ (tzv. interpretace KL divergence jako informačního zisku). Tedy měří množství nadbytečné informace které obdržíme o $x^{(i)}$ když obdržíme z ze $Q(z|X)$ namísto z $P(z)$ ²³. (Doersch, 2021)

V druhé fázi $P(X|z)$ měří množství informace potřebné pro rekonstrukci $x^{(i)}$ ze z za použití ideálního kódování.

Tedy celkový počet bitů ($-\log P(X)$) je součtem těchto dvou fází mínus trestu, který platíme za to, že Q suboptimální kódováním $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$. (Doersch, 2021)

3.9.3 VAE a regularizační prvek

Rovnice 3.5 nabízí další pohled na $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ jakožto na jakýsi regularizační prvek. Stejně jako u řídkého autoenkodéru (viz Sekce 2.6), kde regularizační prvek Ω penalizuje aktivaci neuronů. Z tohoto pohledu je zajímavé zamyslet se, zda-li u variačních autoenkodérů vůbec existuje *něco jako* regularizační prvek. (Doersch, 2021)

U variačního autoenkodéru **regularizační prvek** obecně vůbec **neexistuje**. Což je výhodou – jelikož to znamená o jeden prvek, který je nutné optimalizovat, méně. (Doersch, 2021)

Nicméně, určité modely variačního autoenkodéru se jeví, že regularizační prvek *existuje*²⁴. Dobrou volbou výstupního rozdělení pravděpodobnosti pro spojitá data je $P(X|z) \sim \mathcal{N}(f(z), \sigma^2 * I)$ pro libovolné σ které předem dodáme. Tím pádem dostáváme $\log P(X|z) =$

²²Pro připomenut KL divergenci lze uvádět v jednotkách bitů (viz Sekce 1.5).

²³Pro detailněji popsanou interpretaci odkazují na "bits back" argument z (Geoffrey E. Hinton a Camp, 1993, Sekce 5.2).

²⁴Nabízí se zaměnit $z \sim \mathcal{N}(0, I)$ za $z' \sim \mathcal{N}(0, \Sigma * I)$, ale ukázalo se že toto naučený model nijak nezmění a ve skutečnosti vyprodukuje identickou ztrátovou funkci a model pro vzorkování $x^{(i)}$. (Doersch, 2021)

$C - \frac{1}{2}\|X - f(z)\|^2/\sigma^2$, kde C je konstanta, která nezávisí na f a tak ji lze při optimalizaci zanedbat. (Doersch, 2021)

Pokud toto rozdělení pravděpodobnosti použijeme, ve výsledné účelové funkci se σ objeví jako druhý prvek na pravé straně Rovnice 3.5. Tedy v tomto smyslu lze zvolené σ svou rolí považovat za jakési Ω které kontroluje váhy obou prvků pravé strany. Nutno podotknout, že samotná existence tohoto parametru je závislá na zvoleném rozdělení pravděpodobnosti $x^{(i)}$ pro dané z . Tedy pokud je $x^{(i)}$ binární, a jako výstupní model je použito Alternativní rozdělení, tento *regularizační* prvek zmizí²⁵. Což z pohledu informační teorie dává logiku: je-li $x^{(i)}$ binární, lze spočítat počet bitů, které jsou potřebné pro kódování $x^{(i)}$, a tím pádem oba prvky pravé strany Rovnice 3.5 používají stejné jednotky. Nicméně, pokud je $x^{(i)}$ spojité, každý vzorek obsahuje nekonečnou míru informace. Volba σ pak určuje očekávanou míry přesnosti, s jakou je naučený model schopen rekonstruovat $x^{(i)}$, resp. celé X . (tentu kompromis je nutný, aby se míra informace vzorku stala konečná). (Doersch, 2021)

3.10 Generování nových vzorků

Pro vygenerování nového vzorku \mathbf{x} z modelu VAE je nejprve nutné získat vzorek \mathbf{z} z rozdělení pravděpodobnosti kódu $p_{model}(\mathbf{z})$. Poté je tento vzorek vstupem pro síť generátoru $g(\mathbf{z})$. Na konec je \mathbf{x} vzorkováno z rozdělení pravděpodobnosti $p_{model}(\mathbf{x}; g(\mathbf{z})) = p_{model}(\mathbf{x}|\mathbf{z})$. (D. P. Kingma a M. Welling, 2014)

Při trénování je však pro získání \mathbf{z} použit enkodér (*approximate inference network*) $q(\mathbf{z}|\mathbf{x})$ a na $p_{model}(\mathbf{x}|\mathbf{z})$ pak lze pohlížet jako na dekodér. (D. P. Kingma a M. Welling, 2014)

Klíčovým principem VAE je, že může být trénován maximalizací variační dolní meze (*variational lower bound*) $\mathcal{L}(q)$ asociovanou s datovým bodem \mathbf{x} následovně (I. Goodfellow et al., 2016):

$$\mathcal{L}(q) = \underbrace{\mathbb{E}_{q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)}|z)]}_{\text{společná log věrohodnost}} + \overbrace{\mathcal{H}(q(\mathbf{z}|\mathbf{x}))}^{\text{entropie}} \quad (3.21)$$

$$= \mathbb{E}_{q_\phi(z|x^{(i)})} [\log p_\theta(x^{(i)}|z)] + D_{KL}(q(\mathbf{z}|\mathbf{x}) \parallel p_{model}(\mathbf{z})) \quad (3.22)$$

$$\leq \log p_{model}(\mathbf{x}) \quad (3.23)$$

První člen Rovnice 3.21 ukazuje společnou log věrohodnost (*log-likelihood*) viditelných a skrytých proměnných pod *approximate posterior* skrze latentní proměnné (stejně jako u EM, s rozdílem že používáme *approximate* místo *exact* posterior). (I. Goodfellow et al., 2016)

²⁵I přes to existují způsoby, jakým jej lze dostat zpět do rovnice, jako například replikace dimenzí $x^{(i)}$. (Doersch, 2021)

Druhý člen Rovnice 3.21 ukazuje entropii *approximate posterior*. Je-li q Gaussova distribuce, a je-li k predikované střední hodnotě přičten šum, maximalizace této entropie vede k zvyšující se směrodatné odchylce tohoto šumu. Obecně, prvek této entropie zajistí tendenci *variational posterior* přidělit vysokou hodnotu funkce pravdepodobnostní hodnotám \mathbf{z} které by mohly generovat \mathbf{x} (raději než konvergovat pouze k odhadu jedné hodnoty s *nevětší pravděpodobností*). (I. Goodfellow et al., 2016)

První člen Rovnice 3.22 ukazuje log věrohodnost rekonstrukce, který lze nalézt i v ostatních typech autoenkodérů (Kapitola 2).

Druhý člen Rovnice 3.22 zajišťuje, aby se *approximate posterior distribution* $q(\mathbf{z}|\mathbf{x})$ a model prior $p_{model}(\mathbf{z})$ k sobě blížily. (I. Goodfellow et al., 2016)

Tradiční přístupy k variačnímu odvozování a učení odvozují q pomocí optimalizačního algoritmu (typicky iterují skrze soustavy *fixed-point* rovnic). Takové přístupy jsou pomalé a výpočetně neefektivní, jelikož často vyžadují schopnost vypočítat $\mathbb{E}_{\mathbf{z} \sim q}$ v uzavřeném tvaru. (I. Goodfellow et al., 2016)

Hlavní myšlenkou variačních autoenkodérů je natrénování parametrického autoenkodéru (také nazývaného jako *odvozovací síť* či *rozpoznávací model*) **který produkuje parametry \mathbf{q} .** Je-li \mathbf{z} spojitá proměnná, lze vždy provést algoritmus zpětné propagace skrze vzorky \mathbf{z} získané z $q(\mathbf{z}|\mathbf{x}) = q(\mathbf{z}; f(\mathbf{x}; \boldsymbol{\theta}))$ za účelem spočtení gradientu s respektem k $\boldsymbol{\theta}$. (I. Goodfellow et al., 2016)

Učení VAE pak spočívá čistě v maximalizaci \mathcal{L} s ohledem na parametry enkodéru a dekodéru. (D. P. Kingma a M. Welling, 2014)

3.11 Testování naučeného modelu

Pro testování modelu chceme generovat nové vzorky – toho lze dosáhnout použitím hodnot ze $z \sim \mathcal{N}$ jako vstup pro dekodér. Jedná se tedy vlastně o odstranění enkodéru (včetně operací násobení a sčítání, které by jinak měnily rozdělení pravděpodobnosti z). (Doersch, 2021)

Schéma této jednoduché sítě prezentuje Obrázek 3.5.



Obrázek 3.5: Test-time schéma variačního autoenkovéru, které umožňuje generovat nové vzorky. Jelikož k vygenerování nového vzorku není potřeba enkodér, je vazba na něj jednoduše smazána (oproti síti, kterou zachycuje Obrázek 3.4). Schéma a interpretace převzato z (Doersch, 2021).

Chceme-li vyhodnotit pravděpodobnost vygenerování konkrétního vzorku z naučeného modelu, jedná se o efektivně neřešitelný problém (tentu problém adresuje architektura tzv. Conditional variačních autoenkovéru, viz Podsekce 3.13.7).

3.12 Nedostatky a omezení

Variační autoenkovér nabízí elegantní a po teoretické stránce uspokojivý přístup, který je jednoduchý na implementaci. V oblasti generativního modelování dosahuje excelentních výsledků, ale s ohledem na jeho architekturu přichází i několik podstatných nedostatků.

3.12.1 Omezení při optimalizaci

Nálezy v (Bowman et al., 2016), (Sønderby et al., 2016) konzistentně potvrzují, že stochastická optimalizace účelové funkce s neměnnou dolní mezí může uvíznout v nežádoucí rovnováze. Při počátku trénování je pravděpodobnostní prvek $\log p(x|z)$ relativně slabý, tedy je přípustný

stav kdy $q(z|x) \approx p(z)$ – což je přesně bod, kdy tato nežádoucí rovnováha nastává a je složité z ní při optimalizaci uniknout.

Řešení navrhované v (Bowman et al., 2016) a (Sønderby et al., 2016) využívá rozvrhování optimalizace tak, že váhy latentního *nákladového kritéria* $\mathcal{D}_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z))$ jsou při trénování skrze mnoho epoch žíhány v intervalu od 0 do 1. Tento přístup staví na metodě simulovaného žíhání, která představuje způsob pro uniknutí z lokálního extrému při optimizaci (gradientním sestupem). (Kirkpatrick et al., 1983)

Alternativní řešení, navrhované v (Durk P Kingma et al., 2016), je tzv. metoda *free bits*. Jedná se o jakousi modifikaci ELBO (viz Rovnice 3.5) účelové funkce, která zaručí, že v průměru je v každé latentní proměnné (nebo skupině latentních proměnných) zakódováno alespoň určité minimální množství bitů informace.

3.12.2 Šum v obrázcích vygenerovaných vzorků

Jednou z hlavních nevýhod variačního autoenkovéru, zejména v kontextu úloh generativního modelování obrazových dat, je fakt, že výstupní vzorky z VAE natrénovaného na obrazových datech mají tendenci být rozostřeny. Jednou z možných příčin by mohl být důsledek minimizace $D_{KL}(p_{data} \parallel p_{model})$ (zjednodušený zápis pro jednoduchost). Takto natrénovaný model přidělí vysokou pravděpodobnost bodům, které jsou součástí trénovací množiny dat, ale i spoustě dalších bodů (vzorků, které generuje z naučeného Gaussova rozdělení). A právě tyto vzorky, které nebyly součástí trénovací množiny dat, často obsahují šum a jeví se tak jako rozostřené. (I. Goodfellow et al., 2016)

Přesné příčiny tohoto problému však zatím nejsou známy. Přirozeně tak vzniká celá řada nových, rozšířených architektur variačního autoenkovéru, která svými omezeními tento problém to jisté míry eliminují.

3.13 Aktuální stav poznání a rozšíření variačního autoenkovéru

Tato sekce nabízí přehled technik, které se svým principem podobají variačním autoenkovéru a prezentuje jejich odlišnosti.

3.13.1 Wake-sleep algoritmus

Wake-sleep algoritmus (G. E. Hinton, Dayan et al., 1995) je další on-line metodou pro učení, která je aplikovatelná na stejnou třídu modelů využívajících latentních proměnných, jako VAE. Obdobně jako VAE (Sekce 3.4), wake-sleep algoritmus využívá recognition model který approximuje posteriorní rozdělení původních dat. Značnou nevýhodou wake-sleep algoritmu oproti VAE však je nutnost současně optimalizace dvou účelových funkcí, které dohromady

nekorespondují s optimalizací dolní meze marginální věrohodnosti. Výhodou wake-sleep algoritmu oproti VAE je, že je také aplikovatelný na modely s diskrétními latentními proměnnými. Wake-sleep algoritmus rovněž nabízí stejnou výpočetní složitost skrz jeden datový bod jako ELBO. (Diederik P. Kingma a Max Welling, 2019)

3.13.2 Regularizované autoenkodéry

Rozšířením Sekce 2.7 a Podsekce 2.5.1 jsou Stacked Denoising Autoenkodéry (Vincent et al., 2010). Jejich princip spočívá v maximalizaci (s ohledem na parametry) vzájemné informace a je ekvivalentní s maximalizací podmíněné entropie. Což je dolní mezí očekávané log-věrohodnosti dat, které jsou výstupem autoenkodér modelu – tedy tzv. *negative reconstruction* chyba. Nicméně, jak bylo ukázáno (Bengio, Courville et al., 2014), toto kritérium pro rekonstrukci dat **není dostatečné** pro naučení modelu **užitečným** reprezentací vstupních dat.

Přirozeně tedy bylo přistoupeno k regularizačním technikám, které zajistí naučení užitečných reprezentací. Tyto návrhy představují Sekce 2.6, Sekce 2.7, Sekce 2.9 a Sekce 2.10. Možnost regularizačního prvku jako součást účelové funkce variačního autoenkodéru popisuje Podsekce 3.9.3.

3.13.3 Generativní stochastické sítě

(Bengio, Thibodeau-Laufer et al., 2014) představuje generativní stochastickou síť, kde se noisy autoenkodér učí přechodovou funkci a operátory Markovova řetězce, který vzorkuje z rozdělení pravděpodobnosti vstupních dat. Toto vzorkování je potom využito pro sestavení generativního procesu, což je ve vztahu k variačním autoenkodérům přidružená technika. (Diederik P. Kingma a Max Welling, 2019)

3.13.4 Efektivní učení reprezentací pomocí hlubokých Boltzmann machines

V (Salakhutdinov a Larochelle, 2010) byl využit recognition model (podobně jako Sekce 3.4) pro efektivní učení reprezentací pomocí hlubokých Boltzmann machines. Tato metoda je zaměřena pouze na nenormalizované modely (např.: Boltzmann machines) nebo řídké kódovací modely. V kontrastu s variačním autoenkodér, který je schopen učit se problémům v obecné třídě orientovaných pravděpodobnostních modelů. (Diederik P. Kingma a Max Welling, 2019)

3.13.5 Hluboké autoregresivní sítě

Obdobně jako variační autoenkodér, (Gregor, Danihelka, Mnih et al., 2014) představuje metodu pro učení se pravděpodobnostního modelu za využití autoenkodér struktury. Nicméně

navrhovaná metoda je aplikovatelná pouze při uvažování binárních latentních proměnné (v kontrastu s VAE, kde toto omezení neexistuje). (Diederik P. Kingma a Max Welling, 2019)

3.13.6 Stochastic Backpropagation and Approximate Inference in Deep Generative Models

Práce publikována souběžně a nezávisle na autorech rámce VAE (z jejichž publikací Kapitola 3 převážně čerpá). Práce (Rezende et al., 2014) rovněž představuje propojení autoenkodérů, orientovaných pravděpodobnostních modelů a stochastické variační inference pomocí regulařizačního triku, tedy stejných principů, které popisuje Kapitola 3. Představuje alternativní interpretaci a vhled do vnitřní architektury variačního autoenkodéru.

3.13.7 Conditional variační autoenkodér

Conditional variační autoenkodér (CVAE) (Sohn et al., 2015) modifikuje matematický aparát variačního autoenkodéru, který popisuje Kapitola 3, **podmíněním celého generativního procesu na vstupu**. Tedy vytváří vazbu generativního procesu na konkrétní vstup (resp. konkrétní podobu výstupu). CVAE nabízí řešení problému, kde má input-output mapovací funkce kardinalitu 1:n²⁶, **bez nutnosti explicitní specifikace** struktury výstupního rozdělení pravděpodobnosti. Tedy, použijeme-li opět úlohu generování ručně psané číslice: CVAE nabízí možnost pro generování variací vstupu – tedy pokud na vstupu přijde ručně psaná číslice 9, model CVAE je schopen generovat variace ručně psané číslice 9 **které nebyly součástí trénovací množiny**.

CVAE rovněž řeší problém rozostřených výstupů generativního modelu (viz Podsekce 3.12.2). Regresor VAE minimalizuje vzdálenost k množině mnoha různých výstupů, které potenciálně mohly generovat vstup. Naopak CVAE obecně vybere specifický výstup (např. jednu konkrétní číslici, kterou obdrží na vstupu), a tu generuje – tedy výsledkem je více *realistický* vzorek této konkrétní číslice. (Doersch, 2021)

3.13.8 Deep Recurrent Attention Writer

Deep Recurrent Attention Writer (DRAW) (Gregor, Danihelka, Graves et al., 2015) využívá rekurentní enkodér a rekurentní dekodér v kombinací s *attention* mechanismem (jedná se o pokračování práce (Gregor, Danihelka, Mnih et al., 2014)). Generativní proces DRAW modelu se snaží napodobit mechanismus foveace lidského oka za účelem konstrukce (a generování) komplexních obrázků. Evaluace tohoto modelu dosahuje state-of-the art výkonnosti na řadě datasetů a výstupy jsou na první pohled prakticky nerozeznatelné od reálných dat. (Gregor, Danihelka, Graves et al., 2015)

²⁶V textech strojového učení se můžeme setkat s pojmenováním *strukturovaná predikce*.

3.14 Pozorování v latentním prostoru

Kapitola 4 nabízí přehled aplikací variačního autoenkodéru (Kapitola 3), včetně rozšíření které představuje Sekce 3.13, na celé řadě vybraných problémových oblastí.

4. Úlohy pozorování v latentním prostoru

Generativní modely využívající latentních proměnných umožňují transformaci dat do *jednodušších* a lépe interpretovatelných latentních prostorů. A tím pádem umožňují taková data lépe prozkoumat a porozumět jim. (Diederik P. Kingma a Max Welling, 2019)

Rámec VAE nabízí širokou škálu aplikací. Tato kapitola je zaměřena na ty nejprominentnější z nich, které lze rozčlenit do následujících kategorií:

- Generativní modelování obrazových dat: Sekce 4.1, Sekce 4.2, Sekce 4.3 a Sekce 4.4.
- Syntéza přirozeného jazyka: Sekce 4.6.
- Syntéza pseudo-dat: Sekce 4.7 a Sekce 4.8.

Jednotlivé sekce pouze ilustrují využití rámce VAE pro konkrétní úlohu problémové oblasti a prezentují výsledky dosažené v dříve studiích. Pro popis architektury modelu a jeho evaluaci v je v každé sekci odkázáno na původní publikace.

Detailní návrh a vyhodnocení modelů z vybraných problémových oblastí pak nabízí Kapitola 5.

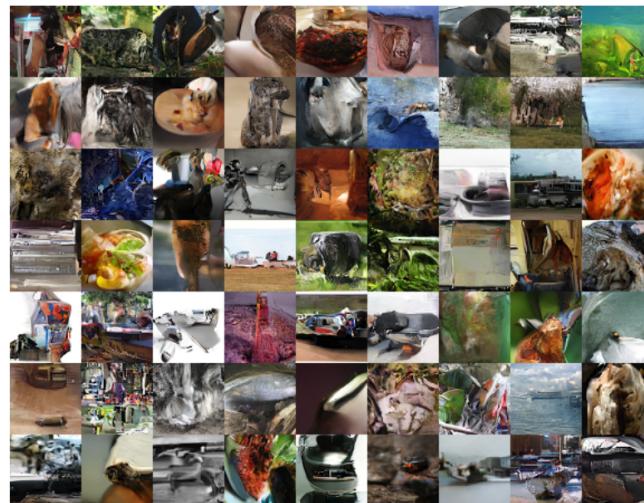
4.1 Generativní modelování obrazových dat

Generativní modelování přirozených obrázků představuje jednu z nejaktuálnějších úloh učení bez učitele vůbec. VAE (a jeho rozšíření) natrénované na datasetech obrazových dat v této oblasti dosahují excelentních výsledků (Gulrajani et al., 2016). DRAW (rozšíření VAE, viz Podsekce 3.13.8) je jedna z vůbec prvních architektur VAE modifikovaná k generování **zcela nových a realistických obrázků**, jenž nebyly součástí trénovací množiny.



Obrázek 4.1: Aplikace VAE na generování zcela nových obrázků dvou cifer, trénováno na MNIST datasetu. Převzato z Gregor, Danihelka, Graves et al. (2015).

Další rozšířenou architekturou je PixelVAE (Gulrajani et al., 2016), která oproti DRAW nabízí větší kvalitu realistických obrázků vzorkovaných z generativního modelu.

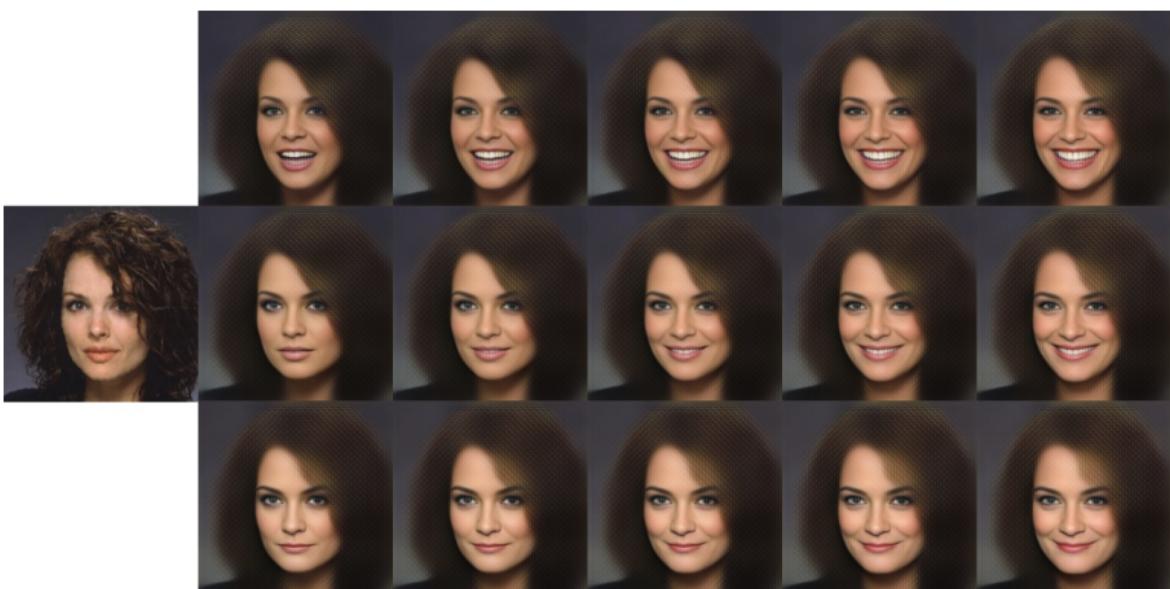


Obrázek 4.2: Výsledek vzorkování PixelVAE (Gulrajani et al., 2016), ImageNet dataset.

4.2 Opakování syntéza obrazových dat

Populární aplikací VAE je (re)syntéza obrazových dat. Využití jednoduchého modelu VAE k opakování syntéze obrazových dat (konkrétně snímků lidských obličejů) představuje White (2016).

VAE lze optimalizovat za účelem zformování generativního modelu skrze vstupní sadu obrazových dat. Z takového generativního modelu pak lze vzorkovat zcela nové obrázky, které nebyly součástí trénovací množiny dat. (Diederik P. Kingma a Max Welling, 2019)



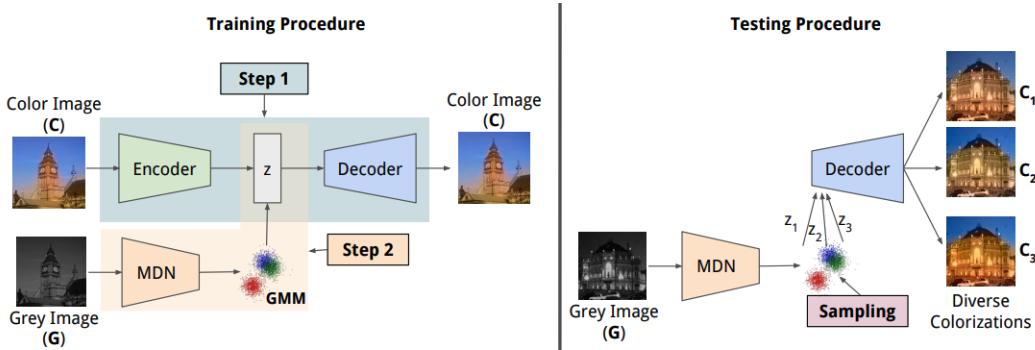
Obrázek 4.3: Využití VAE pro opakovou syntézu vstupního obrázku. V tomto příkladě lze pozorovat posun původního obrázku (vlevo) v modifikovaném latentním prostoru ve směru *vektoru úsměvu* (tzv. aritmetika v latentním prostoru), což má za výsledek progresivně více usmívající se obličeje. Převzato z White (2016).

Takto naučený model ale nabízí ještě jednu zajímavou aplikaci. Inferenční model (enkovér) totiž umožňuje zakódovat reálné obrázky do jeho latentního prostoru. Tento kód pak lze ve vzniklém latentním prostoru modifikovat a následně jej dekódovat zpět do pozorovaného prostoru. Aplikace jednoduchých transformací – například pouhé lineární transformace – na tento prostor má často za důsledek **sémanticky významnou modifikaci původního obrázku**. Obrázek 4.3 demonstruje modifikaci obrázku obličeje v latentním prostoru podél "**vektoru úsměvu**", což činí původní obrázek progresivně více veselým (lze pozorovat i osu otevřených úst při úsměvu). (Diederik P. Kingma a Max Welling, 2019)

4.3 Obarvení černobílých obrázků

Další slibnou aplikací VAE v oblasti generativního modelování obrazových dat je predikce původní barvy jednotlivých pixelů obrázku na základě černobílého (*gray scale*) vstupu.

V Deshpande et al. (2017) autoři představují architekturu VAE, která rozšiřuje CVAE a modifikuje jeho ztrátovou funkci (a to za účelem zpřesnění predikce barvy pixelů a vyhnutí se rozostřeným vygenerovaným vzorkům).



Obrázek 4.4: Schéma trénovací a vzorkovací procedury rozšířeného CVAE pro predikci barvy pixelů původního obrázku. Převzato z Deshpande et al. (2017).

V trénovací fázi se jako první model naučí nízkodimenzionální reprezentaci z pro barevné pole C . Poté je natrénován podmíněný pravděpodobnostní model $P(z|G)$, který generuje nízkodimenzionální reprezentaci na základě černobílého vstupu G . Tento pravděpodobnostní model lze poté vzorkovat a využít dekodér modul vzniklého VAE pro vygenerování rozmanitých barevných polí původního obrázku na výstupu (tj. vygenerování více kombinací *obarvení gray scale* obrázku). **Při vzorkování generativní model jako vstup obdrží pouze gray scale obrázek.** Schéma procedur tohoto modelu zachycuje Obrázek 4.4. (Deshpande et al., 2017)



Obrázek 4.5: Vzorek vygenerovaných variací (vlevo) obarvení původního obrázku (vpravo). Vstupem pro generativní model byl **pouze černobílý** obrázek. Převzato z Deshpande et al. (2017).

Obrázek 4.5 prezentuje vzniklé variace obarvení původního obrázku na základě černobílého vstupu. Takto naučený model tedy lze využít například k obarvení černobílých fotografií.

4.4 Rekonstrukce obrazových dat

Mechanismus VAE je možné využít rovněž k **rekonstrukci** obrazových dat – tedy doplnění chybějící části obrázku generativním modelem (tzv. *inpainting*). Autoři Yan et al. (2016) představují aplikaci VAE (a jeho rozšíření) v kontextu generativního modelování podmíněného atributem (*attribute-conditioned generative modeling*).



Obrázek 4.6: Výstup modelů VAE pro rekonstrukci dat. Převzato z Yan et al. (2016).

Předložený model je naučen na nepoškozených datech. Jako **vstup** pro generativní proces slouží **neúplný obrázek**. Cílem generativního modelu je předpovědět kontext chybějícího regionu obrázku a s ohledem na zbylou část správně doplnit jeho obsah. (Yan et al., 2016)

Obrázek 4.6 prezentuje výstupy modelů VAE aplikovaných na rekonstrukci chybějících regionů **přirozených** obrazových dat. První blok prezentuje výsledky generativního procesu podmíněného atributem obličeje (model je trénován na datasetu LFW¹). Druhý blok pak výsledky které nejsou podmíněny žádným atributem – chybějící region obrázku (16×16 px) je vybrán náhodně (model je trénován na datasetu CUB²).

¹Labeled Faces in the Wild. (Huang et al., 2012)

²Caltech-UCSD Birds-200-2011. (Wah et al., 2011)

4.5 Konceptuální komprese

Další aplikací VAE je konceptuální komprese (Gregor, Besse et al., 2016). Úloha je úzce spjata s informační teorií a interpretací VAE z pohledu informační teorie (viz Podsekce 3.9.2).

Chceme-li komprimovat obrázky psů a koček na jeden bit, jak by měl tento bit vypadat? Dalo by se předpokládat, že by měl ukládat informaci, zda-li se na obrázku nachází pes, či kočka. Na základě této jediné latentní proměnné pak generativní model vygeneruje kompletní obrázek. Uvažujme nyní, že místo jednoho bitu chceme provést kompresi na 10 bitů. Lze tedy uložit ty nejvíce střežejní vlastnosti o vyobrazeném objektu – například barvu, natočení objektu a podobně. Zbývající (detailnější) vlastnosti – jako například vzor srsti – opět *doplňí* generativní model (který na základě těchto informací podmíníme, viz Podsekce 3.13.7). Čím větší počet proměnných v latentním prostoru (čím více bitů má cílová komprese), tím více informací je možné o původním objektu zachovat. (Gregor, Besse et al., 2016)



Obrázek 4.7: Konceptuální komprese s použitím VAE aplikovaná na přirozené obrázky z ImageNet datasetu. Poslední řádek představuje originální obrázky. Zbylé řádky představují **kompresované** (čím menší řádek, tím méně latentních proměnných může model pro kompresi využít) rekonstrukce DRAW modelu. Převzato z Gregor, Besse et al. (2016).

Jelikož značný počet bitů každého obrázku slouží jako nositel těchto *detailnějších* atributů, lze jejich zanedbání využít za účelem ztrátové komprese pomocí VAE.

Model konceptuální komprese založený na rozšíření VAE nabízí **větší kvalitu než kompresní algoritmy JPEG a JPEG 2000**³ ve všech úrovních detekovatelné korupce výsledného obrázku (pro detailnější srovnání viz Gregor, Besse et al. (2016)).

³Aktuálně je stále nejspolehlivějším arbitrem pro posouzení kvality ztrátové komprese člověk. Jiné (dostatečně jednoduché) míry, jako například L2 vzdálenost, jsou nevhodné. (Gregor, Besse et al., 2016)

4.6 Syntéza přirozeného jazyka

Aplikací VAE na interpolaci mezi větami přirozeného jazyka představuje Bowman et al. (2016).

Zde je jednoduchý VAE (se stejnou strukturou, jakou prezentuje Kapitola 3) trénován na korpusu přirozeného anglického jazyka.

“ i want to talk to you . ”
“*i want to be with you . ”*
“*i do n’t want to be with you . ”*
i do n’t want to be with you .
she did n’t want to be with him .

he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

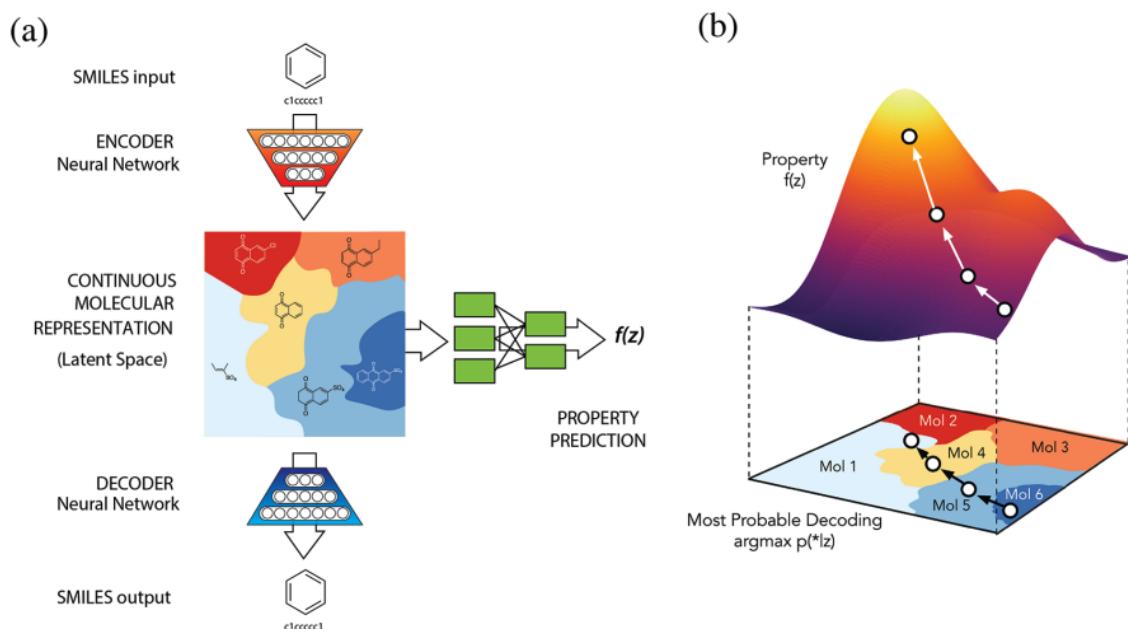
Obrázek 4.8: Aplikace VAE pro interpolaci přirozeného jazyka mezi dvojicí vět. Vstupní páry vět jsou vyznačeny tučně, přechodné věty vygenerované modelem VAE jsou kurzívou. Pozoruhodně jsou přechodné věty gramaticky i syntakticky správné a zachovávají tématický kontext vstupních párů vět. Převzato z Bowman et al. (2016).

Výsledkem je model, který je schopen úspěšně **interpolovat mezi větami a doplnit do vět chybějící slova**. Obrázek 4.8 prezentuje schopnost takto naučeného modelu k interpolaci mezi vstupními páry vět. (Bowman et al., 2016)

4.7 Návrh chemikálií

Doposud byly prezentovány spíše triviální využití VAE. Nicméně rámec VAE nalézá uplatnění i pro vědecké účely.

Gómez-Bombarelli et al. (2018) představují jednoduchý VAE, který je trénován na datové sadě obsahující stovky tisíc existujících chemických struktur. Tento VAE má stejnou strukturu modulů, jako popisuje Kapitola 3. Výsledný latentní prostor (a jeho spojité reprezentace) je posléze použit k gradientní optimalizaci cílového modelu směrem k určitým (chtěným) vlastnostem (viz Gómez-Bombarelli et al. (2018)).

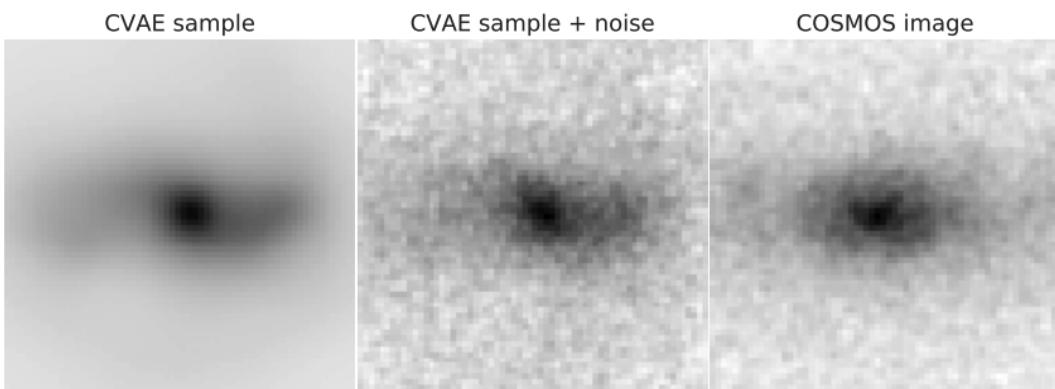


Obrázek 4.9: Aplikace VAE k návrhu chemikálií. Obrázek (a): Proces naučení latentní spojité reprezentace molekul \mathbf{z} z velkého vstupního datasetu. Obrázek (b): Takto naučená spojité reprezentace umožňuje hledání nových molekul které maximalizují $f(z)$ (určitou vyžadovanou vlastnost molekul). Převzato z Gómez-Bombarelli et al. (2018).

Princip této metody k návrhu molekul léků (či alespoň přidružených molekul) demonstrouje Obrázek 4.9. (Diederik P. Kingma a Max Welling, 2019)

4.8 Astronomie

Další aplikací VAE v oblasti vědy prezentuje Ravanbakhsh et al. (2016). Autoři aplikují VAE k simulaci pozorování vzdálených galaxií. Vzorky generované VAE pomáhají kalibrovat systémy, jejichž úkolem je detekovat zkreslení pozorování těchto galaxií (tzv. *shearing*). Toto zkreslení je způsobeno jevem gravitační čočky⁴, který vzniká vlivem přítomnosti temné hmoty mezi Zemí a (pozorovanými) galaxiemi. (Diederik P. Kingma a Max Welling, 2019)



Obrázek 4.10: Aplikace VAE k generování syntetických pseudo dat využitelných pro kalibraci systému k detekci zkreslení pozorování galaxií. K vygenerovanému vzorku pozorování je dodatečně přidán šum. Obrázek vlevo představuje vzorek z generativního modelu, obrázek vpravo představuje reálný snímek pozorování. Převzato z Ravanbakhsh et al. (2016).

Vzhledem k tomu, že jev gravitační čočky je opravdu slabý (a tím pádem složitě detekovatelný), je nutné tyto systémy kalibrovat **realistickými** obrázky s odpovídajícím množstvím zkreslení. A vzhledem k tomu, že dostupné **množství takových reálných dat je aktuálně velmi omezené**, přistoupili autoři studie k využití VAE (respektive CVAE, viz Podsekce 3.13.7) za účelem **syntézy pseudo-dat**, které jsou následně pro kalibraci využity. Jeden takový vzorek prezentuje Obrázek 4.10. (Ravanbakhsh et al., 2016)

4.9 TODO: Vizualizace high-dimensional dat

⁴Pojem gravitační čočky vychází ze spojné (optické) čočky. Užívá se pro popis objektu (např. klastru galaxií), který vlivem svého gravitačního pole způsobuje prohnutí paprsků světla vycházejících ze zdroje směrem k pozorovateli.

5. Experiment s modelem variačního autoenkodéru

Kapitola 3 popsala teoretické aspekty variačního autoenkodéru, princip fungování jeho dílčích modulů a charakter účelové funkce. Kapitola 4 pak ukázala široké spektrum mezioborových aplikací rámce variačního autoenkodéru. Cílem kapitoly je představit praktickou ukázkou sestrojení modelu variačního autoenkodéru na ilustrační úloze v oblasti generativního modelování a představit principy jeho trénovacího procesu z praktického hlediska. Po natrénování modelu následuje analýza vzniklého latentního prostoru.

Princip variačního autoenkodéru (který popisuje Sekce 3.2) se pochopitelně promítá i do návrhu modelu.

5.1 Vymezení problémové oblasti

Cílem této kapitoly je vytvořit **ilustrační implementaci** modelu variačního autoenkodéru.

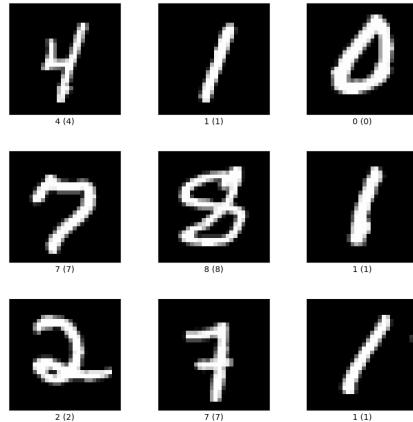
Jak napovídala Sekce 4.1, populární aplikací variačního autoenkodéru je úloha generativního modelování obrazových dat. V rámci této kapitoly je tedy vytvořen **model variačního autoenkodéru pro generativní modelování ručně psaných číslic**. Model je trénován na datové sadě MNIST (viz Sekce 5.2). Implementace jednotlivých částí modelu variačního autoenkodéru je propojena s teoretickými aspekty rámce VAE, které představila Kapitola 3.

Po natrénování modelu jsou prezentovány možnosti **generování zcela nových číslic** (viz Obrázek 5.1) a je prozkoumán vzniklý latentní prostor modelu.

Kompletní zdrojové kódy pro replikaci prezentovaných výsledků a vizualizací modelu jsou k dispozici v Appendix A.

5.2 Datová sada

Datová sada MNIST se skládá z černobílých obrázků **ručně psaných** číslic o velikosti 28×28 px. Každá číslice je vycentrovaná a má konzistentní velikost¹. Vzorky MNIST číslic představuje Obrázek 5.1. MNIST obsahuje 60 000 obrázku pro trénování a 10 000 obrázků pro testování.



Obrázek 5.1: Ukázka vzorků z MNIST.

Datová sada je dostupná z Yann LeCun a Cortes (2010). Datová sada MNIST se díky své jednoduchosti často využívá pro ilustrační implementace modelů strojového učení, a i z tohoto důvodu byl pro tuto kapitolu zvolen.

5.3 Architektura hlubokého modelu variačního autoenkodéru

Pro sestavení hlubokého modelu variačního autoenkodéru² jsou nutné následující moduly, resp. vrstvy:

- Model enkodér modulu
- Model dekodér modulu
- Reparametrikační vrstva
- Ztrátová funkce

V této sekci jsou tyto moduly propojeny s teoretickými aspekty variačního autoenkodéru z předchozích kapitol a postupně implementovány za účelem řešení úlohy generativního mode-

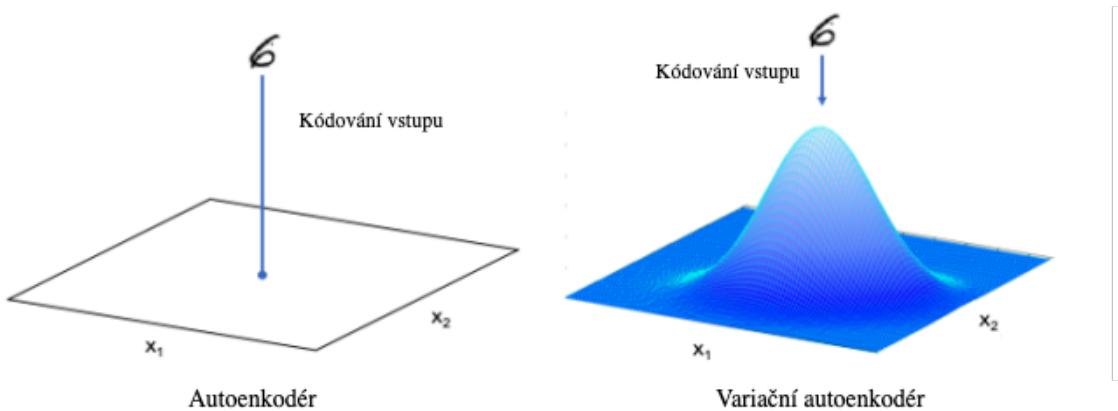
¹Normalizace za účelem lepší rozpoznatelnosti.

²Model variačního autoenkodéru se od prostého autoenkodéru (viz Kapitola 2) liší zejména v **enkodér modulu** a **ztrátové funkci** (viz Sekce 3.6).

lování obrazových dat na datové sadě MNIST. Podsekce 5.3.5 pak prezentuje výsledný model variačního autoenkovodéru, který je připraven pro trénování.

5.3.1 Enkodér

U prostého autoenkovodéru je každý vstup mapován přímo na **pouze jeden bod** v latentním prostoru (tedy výsledný latentní prostor není spojitý). Naopak ve variačním autoenkovodéru je každý vstup mapován na (vícerozměrné) normální rozdělení okolo konkrétního bodu v latentním prostoru, jak ukazuje Obrázek 5.2.



Obrázek 5.2: Rozdíl mezi enkodérem modulem autoenkovodéru a enkodérem modulem variačního autoenkovodéru. Umožnuje vzorkování ze spojitého latentního prostoru VAE. Obrázek převzat z Foster (2023).

Pro připomenutí, jak plyne z Sekce 3.4, enkodér modul variačního autoenkovodéru **zakóduje každý vstup do dvou vektorů** – mu a log_var³, ze kterých je následně v latentním prostoru **sestaveno** (vícerozměrné) **normální rozdělení**, kde: mu: střední hodnota rozdělení a log_var: logaritmus rozptylu (každé z dimenzí).

Pro vytvoření kódu vstupního obrázku do konkrétní latentní proměnné z v latentním prostoru modelu lze z takto vzniklé distribuce vzorkovat pomocí reparametizační vrstvy modelu. Tento proces detailněji popisuje Podsekce 5.3.3.

Model enkodéru modulu variačního autoenkovodéru je pro účely tohoto experimentu definován následující konfigurací vrstev⁴:

³Jak víme z Sekce 3.7 enkodér modul předpokládá že **neexistuje korelace mezi žádnou z dimenzí latentního prostoru** a tím pádem je výsledná kovarianční matici diagonální. V důsledku tak enkodér modulu stačí mapovat každý vstup na vektor středních hodnot a vektor rozptylu.

⁴Počet vrstev vychází z D. P. Kingma a M. Welling (2014) a jejich konfigurace byla následně modifikována a

1. **Vstupní vrstva**⁵:

```
Input(shape=(28, 28, 1), name='encoder_input')
```

2. **Konvoluční vrstva:**

```
Conv2D(filters=32, kernel_size=3, strides=1, padding='same')
```

3. **Konvoluční vrstva:**

```
Conv2D(filters=64, kernel_size=3, strides=2, padding='same')
```

4. **Konvoluční vrstva:**

```
Conv2D(filters=64, kernel_size=3, strides=2, padding='same')
```

5. **Konvoluční vrstva:**

```
Conv2D(filters=64, kernel_size=3, strides=1, padding='same')
```

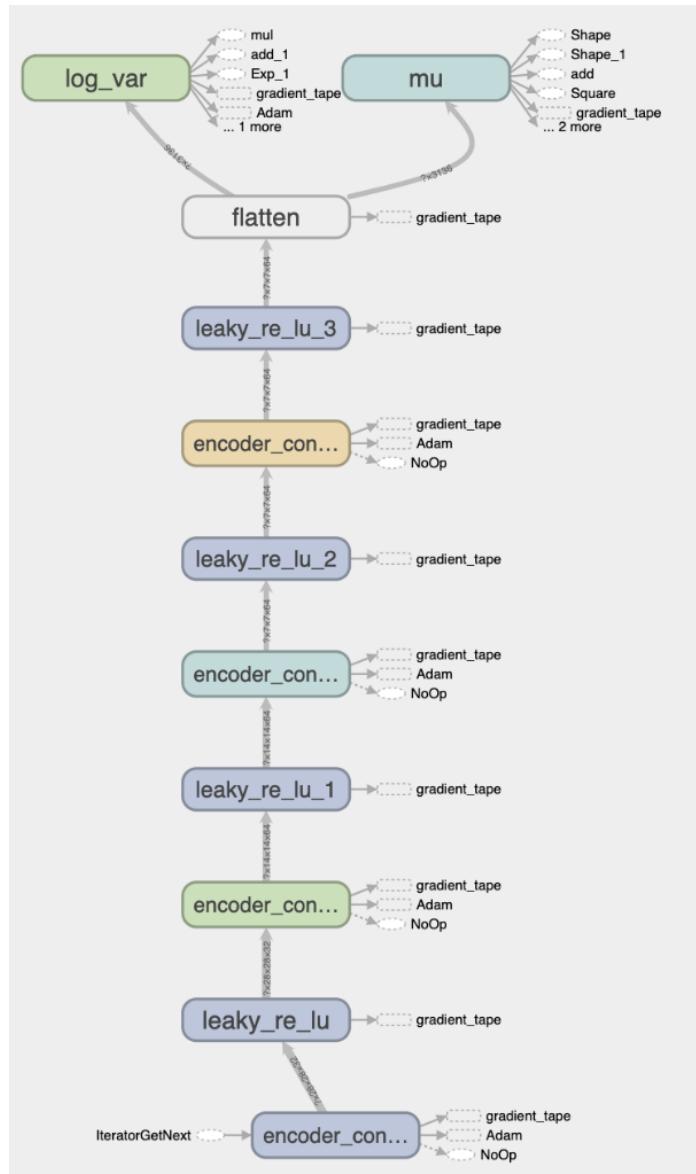
6. **Vyhlažovací vrstva:**

```
Flatten()(x)
```

Schematické zobrazení vrstev enkodér modulu ilustruje Obrázek 5.3.

rozšířena na základě minimalizace ztrátové funkce a prevence přeúčení. Detailnější popis nabízí Podsekce 5.3.4.

⁵input_dim odpovídá jedné číslici MNIST.



Obrázek 5.3: Diagram enkodér modulu VAE pro generování MNIST číslic.

5.3.2 Dekodér

Dekodér modul variačního autoenkovadéru je principem identický k dekovadéru prostého autoenkovadéru. Model denkovadér modulu variačního autoenkovadéru je pro účely tohoto experimentu definován následující konfigurací vrstev⁶:

1. Vstupní vrstva:⁷

```
Input(shape=2, name='encoder_input')
```

2. Dekonvoluční vrstva:

⁶Počet vrstev vychází z D. P. Kingma a M. Welling (2014) a jejich konfigurace byla následně modifikována a rozšířena na základě minimalizace ztrátové funkce a prevence přeúčtení. Detailnější popis nabízí Podsekce 5.3.4.

⁷shape odpovídá dimenze latentního prostoru (pro obrázky MNIST číslic 2D).

```
Conv2DTranspose(filters=64, kernel_size=3, strides=1, padding='same')
```

3. Dekonvoluční vrstva:

```
Conv2DTranspose(filters=64, kernel_size=3, strides=2, padding='same')
```

4. Dekonvoluční vrstva:

```
Conv2DTranspose(filters=32, kernel_size=3, strides=2, padding='same')
```

5. Dekonvoluční vrstva:

```
Conv2DTranspose(filters=1, kernel_size=3, strides=1, padding='same')
```

6. Vyhlažovací vrstva:

```
Flatten()(x)
```

5.3.3 Reparametrizační vrstva

Reparametrizační vrstva modelu variačního autoenkodéru implementuje reparametrizační trik dle Sekce 3.8. Přičemž pro připomenutí, ϵ je bod náhodně vzorkovaný z normovaného normálního rozdělení. Právě ϵ hraje při trénování variačního autoenkodéru důležitou roli, jelikož transformuje vzorkovací proces na diferenciovatelnou operaci (a této vlastnosti je využito při následné optimalizaci modelu, viz Podsekce 5.3.4).

U variačního autoenkodéru je vzorkován náhodný bod z ϵ -okolí mu. Tedy dekodér musí při trénování zajistit, aby všechny body v tomto okolí produkovaly výstup (obrázek) který je *velmi podobný* vstupu, nebo má alespoň podobný sémantický význam. Popsanou operaci v reparametrizační vrstvě modelu zachycuje Obrázek 5.4.

```
1 def reparameterization(self, z_mean, z_log_var):
2     batch = tf.shape(z_mean)[0]
3     dim = tf.shape(z_mean)[1]
4     epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
5     z = z_mean + tf.exp(0.5 * z_log_var) * epsilon
6     return z
```

Obrázek 5.4: Reparametrizační vrstva modelu variačního autoenkodéru.

Toto je velmi elegantní vlastnost variačního autoenkodéru, která zajistí že i v případě, kdy je při vzorkování zvolen bod, který dekodér modul variačního autoenkodéru doposud nikdy *neviděl*, s vysokou pravděpodobností bude výstupem dobře formovaný obrázek. Tedy **variačnímu autoenkodéru je umožněno generovat data, která nebyly součástí trénovací množiny, ale nesou stejné charakteristiky jako vstupní data**. Syntéza takovýchto dat je jedna z hlavních účelů oblasti generativního modelovaní. Tuto vlastnost detailněji zkoumá Sekce 5.5, která se věnuje pozorování latentního prostoru naučeného modelu.

5.3.4 Ztrátová funkce

Ztrátová funkce modelu variačního autoenkodéru implementuje postup dle Rovnice 3.5. Model VAE tedy **minimalizuje součet hodnoty chyby rekonstrukce a hodnoty KL divergence**.

Pro rekapitulaci Sekce 1.5, KL divergence vyjadřuje míru **nepodobnosti** (resp. "*vzdálenost*") mezi pravděpodobnostními distribucemi. U variačního autoenkodéru chceme měřit jak moc se liší normální rozdělení s parametry mu a log_var oproti normovanému normálnímu rozdělení. Prvek KL divergence tedy penalizuje umělou neuronovou síť modelu za takové kódování vstupních dat (jehož výsledkem je mu a log_var), které se **výrazně liší** od parametrů normované normálního rozdělení. Tento prvek KL divergence tak hraje při trénování modelu důležitou roli, jelikož *nutí* všechny distribuce s parametry, které jsou výsledkem kódování, *podobat se* normovanému normálnímu rozdělení. V důsledku existuje **menší šance** že zformovaný **latentní prostor model bude obsahovat značné mezery** mezi klastry latentních proměnných (jak ukazuje Sekce 5.5). Ve snaze o minimalizaci ztrátové funkce modelu se totiž enkoder modul pokouší o efektivní a symetrické využití prostoru v okolí počátku latentního prostoru.

Ztrátová funkce rovněž musí hledat **rovnováhu mezi hodnotou chyby rekonstrukce a hodnotou KL divergence**. Udelení příliš velké váhy jednomu z těchto prvků by mělo za důsledek ztrátu kýzeného regulačního efektu latentního prostoru a vzniklý model variačního autoenkodéru by nabýval stejných problémů, jako prostý autoenkodér. Zajištění konvergence obou prvků je tak důležitou součástí trénovacího procesu modelu variačního autoenkodéru a vede k chtěné vlastnosti spojitého latentního prostoru, která umožňuje variačnímu autoenkodéru generovat i takové vzorky dat, které nebyly součástí trénovací množiny.

Logika trénovacího kroku modelu variačního autoenkodéru zahrnuje dopředný průchod, výpočet hodnoty ztrátové funkce a aktualizaci metrik modelu. Trénovací krok modelu definuje Obrázek 5.5.

```
1 def train_step(self, data):
2     with tf.GradientTape() as tape:
3         z_mean, z_log_var = self.encode(data)
4         z = self.reparameterization(z_mean, z_log_var)
5         reconstruction = self.decode(z)
6
7         reconstruction_loss = tf.reduce_mean(
8             tf.reduce_sum(keras.losses.binary_crossentropy(data, reconstruction), axis=(1, 2))
9         )
10        kl_loss = -0.5 * (1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var))
11        total_loss = reconstruction_loss + kl_loss
12
13        grads = tape.gradient(total_loss, self.trainable_weights)
14        self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
15
16        self.total_loss_tracker.update_state(total_loss)
17        self.reconstruction_loss_tracker.update_state(reconstruction_loss)
18        self.kl_loss_tracker.update_state(kl_loss)
19
20    return {
21        "total_loss": self.total_loss_tracker.result(),
22        "reconstruction_loss": self.reconstruction_loss_tracker.result(),
23        "kl_loss": self.kl_loss_tracker.result(),
24    }
```

Obrázek 5.5: Trénovací krok modelu variačního autoenkodéru.

Kde:

- *Řádek č. 3*: Volá enkodér modul VAE za účelem získání kódu vstupních dat (tedy parametrů pravděpodobnostní distribuce).
- *Řádek č. 4*: Realizuje **reparametizační trik** dle Sekce 3.8 za účelem získání vzorku z latentního prostoru na základě parametrů jeho rozdělení pravděpodobnosti.
- *Řádek č. 5*: Generuje rekonstrukci vstupních dat (v tomto případě obrázku MNIST číslice) na základě získaných latentních proměnných voláním dekodér modulu VAE.
- *Řádek č. 7 - 9*: Realizuje gradientní optimalizaci dle Rovnice 3.16 výpočtem chyby rekonstrukce vygenerovaného vzorku vůči vstupním datům.
- *Řádek č. 10*: Spočte KL divergenci mezi pravděpodobnostní distribucí latentních proměnných (s parametry z_mean , z_log_var) a normovaným normálním rozdělením.
- *Řádek č. 11*: Udává celkovou hodnotu ztrátové funkce dle Rovnice 3.5, respektive součet hodnoty chyby rekonstrukce a hodnoty KL divergence.
- *Řádek č. 13*: Spočte gradienty vah neuronů dle Rovnice 3.16 s ohledem na celkovou hodnotu ztrátové funkce.

- *Řádek č. 14*: Realizuje stochastická gradientní optimalizaci (s ohledem na již spočtené gradienty) dle Algoritmus 1.
- *Řádek č. 16 - 24*: Při každém trénovacím kroku aktualizuje a propaguje metriky ztrátové funkce na instanci VAE modelu.

5.3.5 Výsledný model

Obrázek 5.6 zachycuje konfiguraci instance VAE modelu pro úlohu generativního modelování obrazových dat MNIST. Při instanciaci jsou jako parametry modelu předány konfigurace konvolučních vrstev enkodéru, resp. dekodéru přesně dle Podsekce 5.3.1, resp. Podsekce 5.3.2.

```
1 from model.VAE import VAE
2
3 vae = VAE(
4     input_dim = (28, 28, 1),
5     z_dim = 2,
6     encoder_conv_filters = [32, 64, 64, 64],
7     encoder_conv_kernel_size = [3, 3, 3, 3],
8     encoder_conv_strides = [1, 2, 2, 1],
9     decoder_conv_t_filters = [64, 64, 32, 1],
10    decoder_conv_t_kernel_size = [3, 3, 3, 3],
11    decoder_conv_t_strides = [1, 2, 2, 1]
12 )
```

Obrázek 5.6: Instanciace navrhnutého modelu VAE pro úlohu generativního modelování obrazových dat MNIST.

Kde:

- *Řádek 4*: Definuje rozměry (*dimensions*) vstupních dat, kterou má VAE očekávat. V tomto případě pro MNIST číslice 28×28 px a 1 barevný kanál.
- *Řádek 5*: Definuje požadovanou dimenzi latentního prostoru.
- *Řádek 6 - 8*: Udává konfiguraci konvolučních vrstev modelu enkodér modulu dle Podsekce 5.3.1.
- *Řádek 9 - 11*: Udává konfiguraci konvolučních vrstev modelu dekodér modulu dle Podsekce 5.3.2.

5.4 Trénování hlubokého modelu variačního autoenkodéru

Trénovací fáze modelu variačního autoenkodéru spočívá v minimalizaci ztrátové funkce dle Podsekce 5.3.4. Účelem této ztrátové funkce je současná minimalizace chyby rekonstrukce a regularizace naučené pravděpodobnostní distribuce latentních proměnných za účelem přiblížení k apriorní distribuci vstupních dat. Pro umožnění zpětné propagace byla představena reparametizační vrstva modelu variačního autoenkodéru, která činí vzorkovací proces diferenciovatelným.

Trénování zahrnuje určitý počet **epoch**. Kdy se jednou epochou rozumí **kompletní průchod algoritmu** (specifikovaném v trénovacím kroku modelu, viz Obrázek 5.5) **skrze trénovací data**.

Zahájení trénovacího procesu modelu variačního autoenkodéru ukazuje Obrázek 5.7.

```
1 vae.compile(optimizer = tf.keras.optimizers.Adam())
2 vae.fit(x_train, epochs = 500, batch_size = 128)
```

Obrázek 5.7: Zahájení testovací fáze modelu variačního autoenkodéru pro úlohu generativního modelování obrazových dat MNIST.

Kde:

- *Řádek 1:* Definuje použití optimizéru pro stochastickou gradientní optimalizaci implementující algoritmus Adam (Kingma a Ba, 2017)⁸ dle Podsekce 3.7.1.
- *Řádek 2:* Spouští trénovací fázi modelu variačního autoenkodéru na datové sadě MNIST, specifikuje počet epoch trénovací fáze a velikost dávky. Pro trénovací fázi byla použita **pouze trénovací množina MNIST bez štítků**. Testovací množina MNIST byla ponechána pro evaluaci kvality rekonstrukce⁹. Velikost dávky¹⁰ (*batch size*) je hyperparameter modelu, který udává počet vzorků vstupních dat, který bude modelem zpracován, než dojde k aktualizaci vah jeho neuronové sítě.

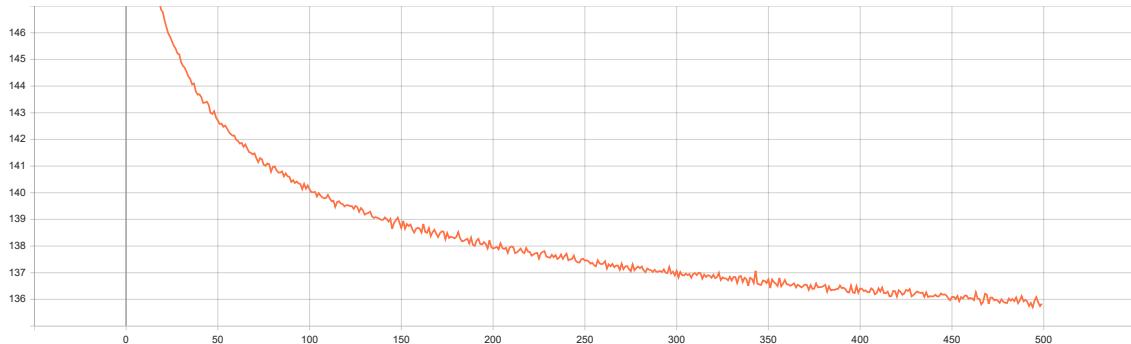
⁸Algoritmus Adam byl využit kvůli jeho výpočetní efektivitě, nízkým paměťovým nárokům a vlastnosti invariance vůči přeskálování gradientů. Autoři Kingma a Ba (2017) jej uvádí jako ideální pro úlohy generativního modelování obrazových dat.

⁹Způsob načtení a předzpracování datové sady uvádí Appendix A.

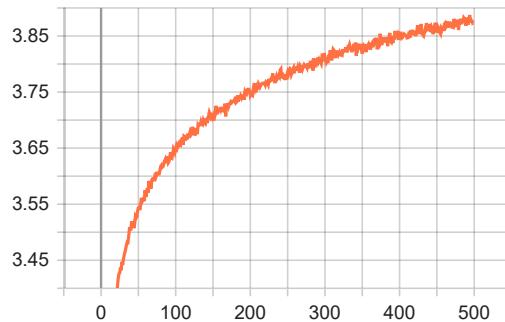
¹⁰Hodnota batch size byla stanovena na základě doporučení v Mishkin et al. (2017), kde autoři pracují s podobnou konfigurací konvolučních vrstev na úloze zahrnující obrazová data.

5.4.1 Konvergence ztrátové funkce

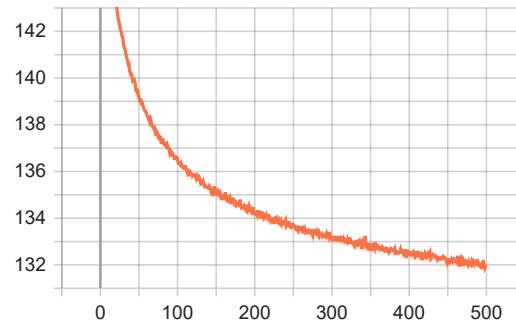
Obrázek 5.8 zachycuje vývoj hodnoty ztrátové funkce natrénovaného modelu variačního autoenkodéru. Obrázek 5.9 pak ukazuje hodnoty dílčích prvků ztrátové funkce (KL divergence a chyba rekonstrukce), které se model variačního autoenkodéru při trénovací fázi snaží balancovat. Výsledná hodnota ztrátové funkce je součtem hodnoty KL divergence a hodnoty chyby rekonstrukce.



Obrázek 5.8: Ztrátová funkce po 500 epochách konverguje k hodnotě ~ 135.8 . Osa x značí počet epoch. Osa y značí hodnotu ztrátové funkce.



(a) KL divergence po 500 epochách konverguje k hodnotě ~ 3.8 .



(b) Chyba rekonstrukce po 500 epochách konverguje k hodnotě ~ 132 .

Obrázek 5.9: Hodnoty dílčích prvků ztrátové funkce modelu variačního autoenkodéru po 500 epochách. Osa x značí počet epoch. Osa y značí hodnotu dílčího prvku ztrátové funkce.

Celkový počet epoch byl stanoven základě doménové znalosti (tedy předem známý počet tříd, jejich sémantika a společné rysy) a postupného formování latentního prostoru, jak zachycuje Obrázek 5.11. Při vizualizaci latentního prostoru modelu, jehož trénovací fáze zahrnovala 500 epoch, lze pozorovat emergenci struktury jeho bodů, která **koresponduje se skutečným významem trénovacích dat** (čísla 0-9, 10 dostatečně izolovaných skupin). Hodnota ztrátové funkce je při takovémto počtu epoch stabilizována a osciluje kolem celkové hodnoty 135.8. Vývoj hodnot ztrátové funkce skrze různý počet epoch v trénovací fázi prezentuje Appendix B. V důsledku těchto jevů byla **trénovací fáze modelu ukončena** a uspokojivý **počet epoch trénovací fáze byl stanoven na 500**.

5.4.2 Možné zlepšení trénovací strategie variačního autoenkodéru

Variační autoenkodér při trénování balancuje dva prvky ztrátové funkce (viz Rovnice 3.5) – chybu rekonstrukce a KL divergenci. Usměrnění těchto dvou prvků lze dosáhnout zahrnutím hyperparametru β do trénovací fáze modelu. Tento hyperparametr má skalární hodnotu a slouží k násobení KL divergence pravděpodobnosti ztrátové funkce, tedy přiděluje KL divergenci váhu (Higgins et al., 2022).

Je-li $\beta = 1$, pak variační autoenkodér optimalizuje efekt chyby rekonstrukce i KL divergence rovnocenně. Je-li $\beta < 1$, dává variační autoenkodér větší důraz na minimalizaci chyby rekonstrukce. Je-li $\beta > 1$, dává variační autoenkodér naopak větší důraz na regularizační efekt KL divergence (Higgins et al., 2022).

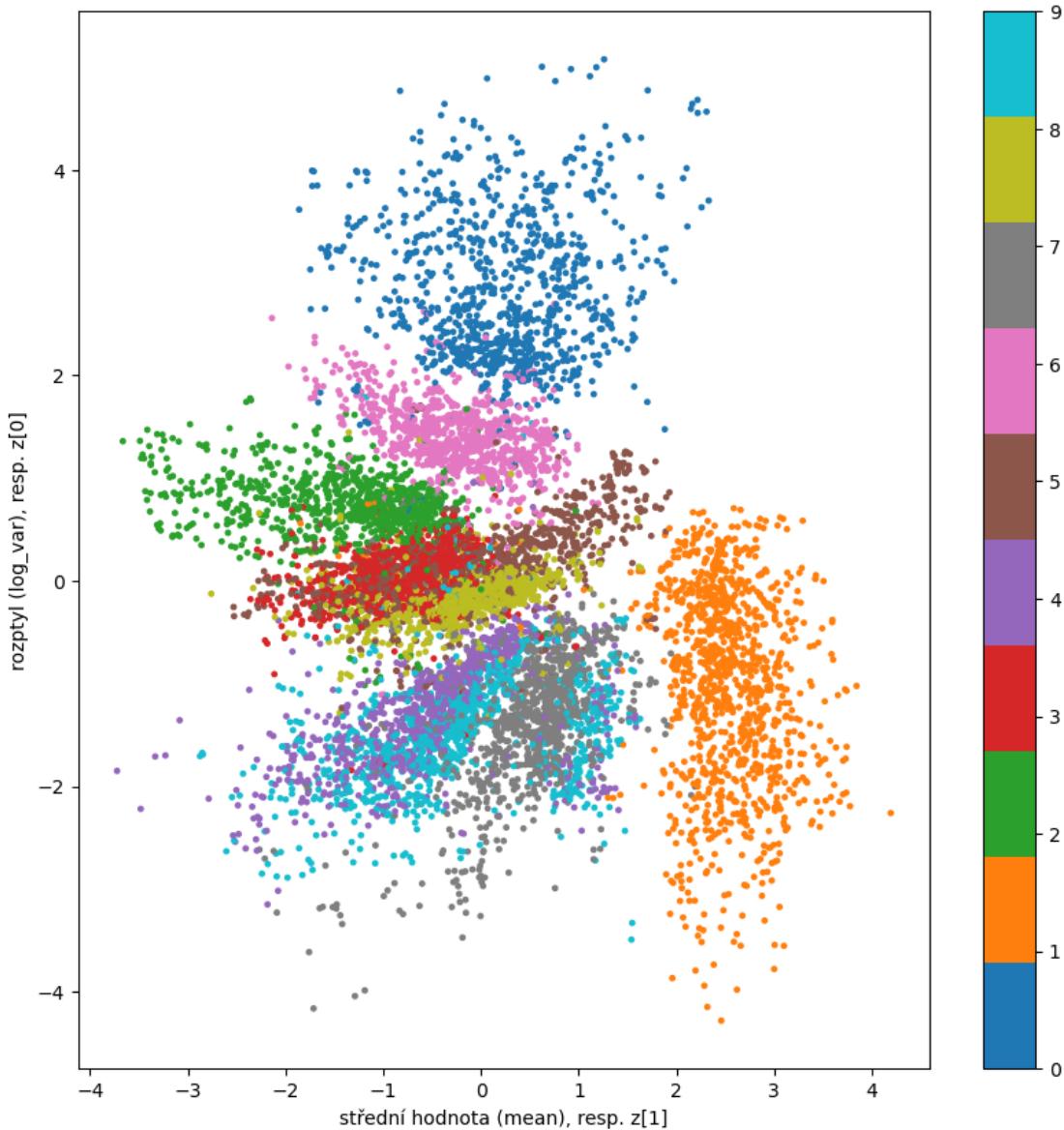
V této ilustrační implementaci generativního modelu MNIST pro jednoduchost **hyperparametr β nebyl součástí trénovacího procesu**. Jeho začlenění by však dle Higgins et al. (2022) vedlo k ještě lépe strukturovanému latentnímu prostoru naučeného modelu, než prezentuje Obrázek 5.11.

U složitějších modelů variačního autoenkodéru nalézá uplatnění β hyperparametru úspěch. Výzkumníci Sankarapandian a Kulis (2021) dokonce přichází se strategií postupného žíhání hodnoty β . Začínají s hodnotou $\beta \ll 1$ a postupně ji zvyšují. To v důsledku umožní modelu se v úvodní fázi trénování zaměřit na minimalizaci chyby rekonstrukce. V pozdější fázi trénovacího procesu je naopak postupně upřednostňován regularizační efekt KL divergence, což ve finále vede k přesnějšímu zachycení struktury dat a zamezuje přeúčtení.

5.5 Pozorování v latentním prostoru

Po natrénování modelu variačního autoenkodéru lze nahlédnout do zformovaného latentního prostoru. Jednotlivé body latentního prostoru jsou tvořeny průchodem vstupních dat (MNIST číslic) enkodér modulem. Obrázek 5.10 ukazuje jejich obarvení, které bylo přiřazeno na základě třídy vstupního vzorku (tedy v tomto případě má každá číslice 0 - 9 přiřazenou jednu barvu)¹¹.

¹¹Pro vizualizaci 2D latentního prostoru MNIST číslic je výsledná vizualizace srovnatelná s výsledky obdrženými využitím t-SNE techniky, které prezentuje G. E. Hinton a Roweis (2002).



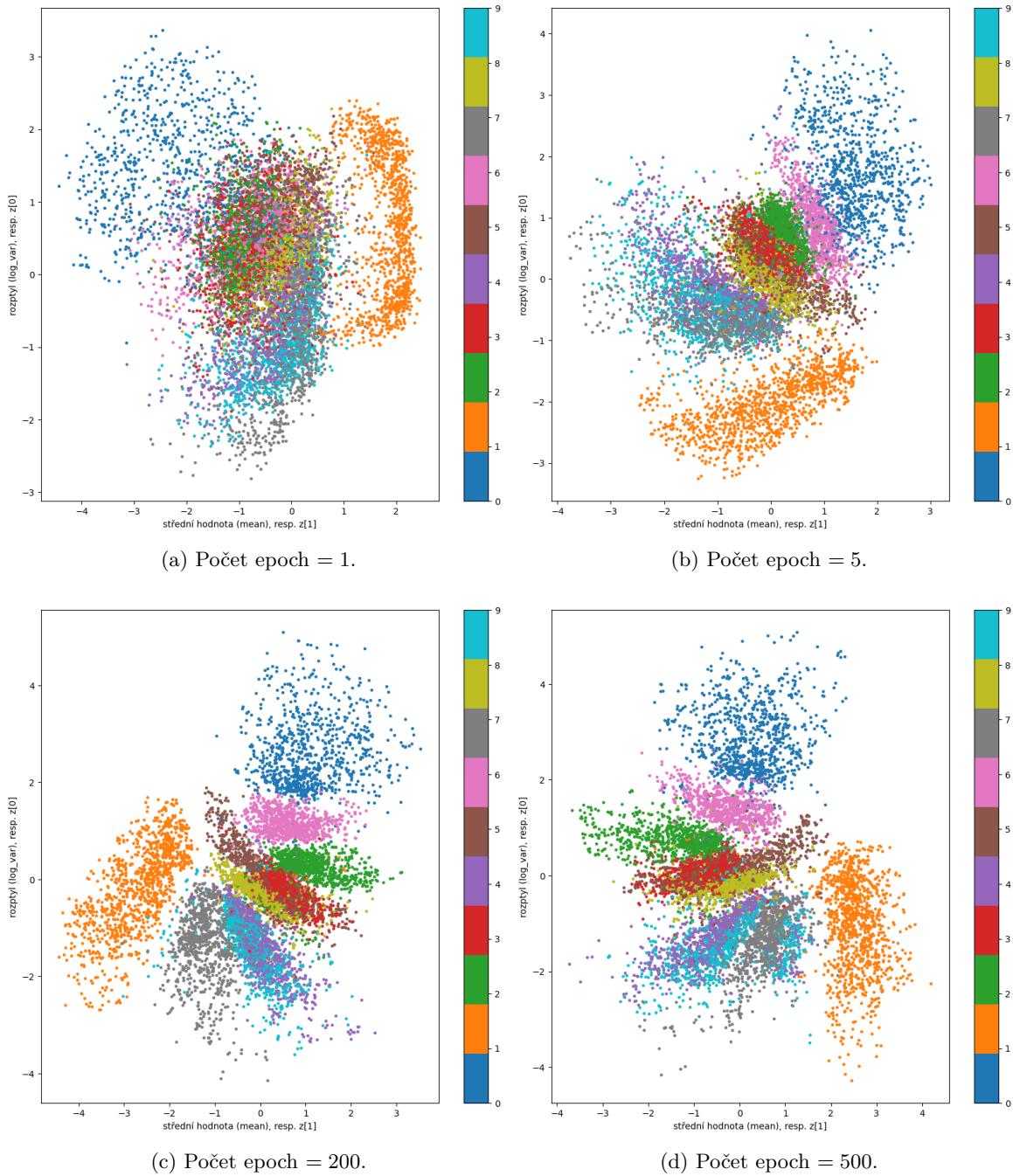
Obrázek 5.10: Latentní prostor naučeného modelu variačního autoenkodéru. Barvy reprezentují jednotlivé třídy MNIST datasetu. Celkový počet zobrazených latentních proměnných je 10000.

Latentní prostor je organizovaný, i přes to, že model variačního autoenkodéru neměl k dispozici štítky trénovacích dat. Tyto uskupení jsou důsledkem ztrátové funkce modelu. Variační autoenkodér se sám naučil tvar a charakteristiky jednotlivých číslic minimalizací chyby rekonstrukce.

Zároveň vzdálenost mezi skupinami číslic, jejichž vzorky jsou si podobné (např. 4 a 9) má v latentním prostoru naučeného modelu tendenci být nízká. I proto lze u naučeného modelu pozorovat určitý sémantický význam aritmetiky v latentním prostoru (např. *plynulý přechod* při posunu mezi 0 a 9).

5.5.1 Postupné formování latentního prostoru

Obrázek 5.11 zachycuje vliv počtu epoch trénovací fáze na strukturu latentního prostoru naučeného modelu. S rostoucím počtem epoch lze pozorovat emergenci oddělených uskupení latentních proměnných, kdy každé takové uskupení zachycuje jednu z tříd MNIST datasetu (tedy číslice 0 - 9).

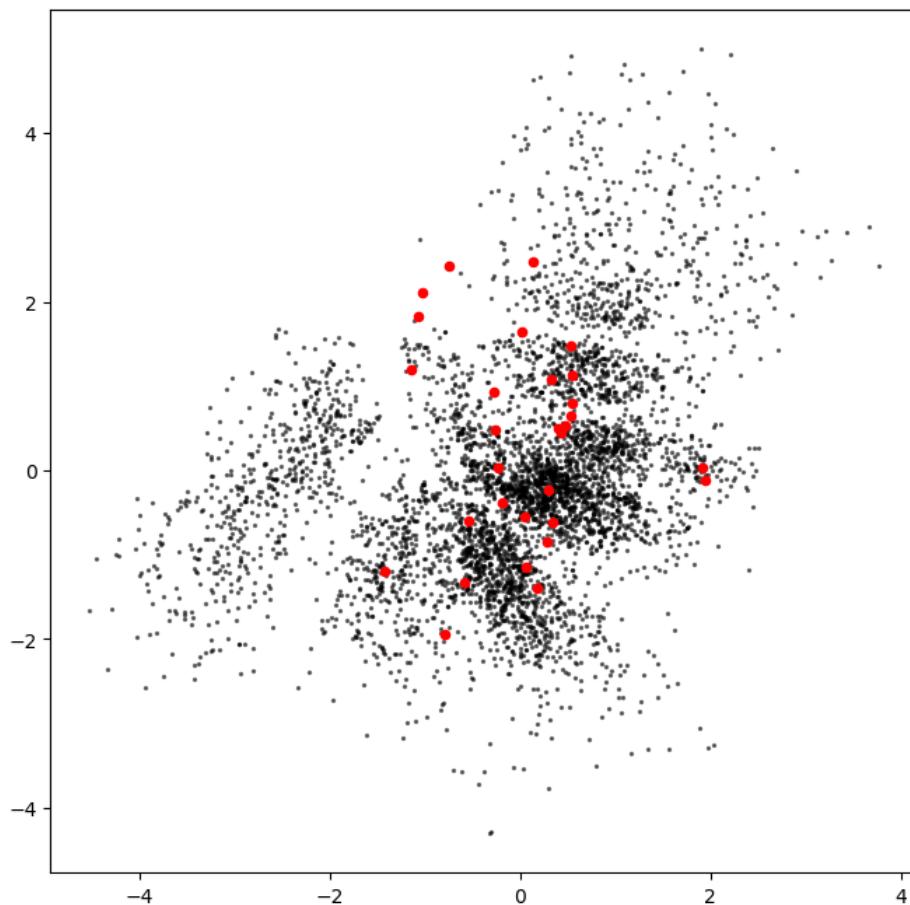


Obrázek 5.11: Postupné formování struktur v latentním prostoru naučeného modelu pro generativní modelování MNIST číslic. Dílkí obrázky a, b, c, d zachycují výsledný latentní prostor modelu, jehož trénovací fáze zahrnovala 1, 5, 200, 500 epoch respektive. Při trénování jednotlivých modelů byla využita pouze trénovací množina MNIST datasetu **bez štítků**. **Parametry vizualizace jsou pro každý dílkí obrázek identické** a liší se pouze počtem epoch, kterými model v trénovací fázi prošel.

Model, jehož trénovací fáze zahrnovala 500 epoch, dospěl k téměř zrcadlovému zobrazení bodů v latentním prostoru oproti modelu, jehož trénovací fáze zahrnovala 200 epoch. To vedlo k **snížení hodnoty ztrátové funkce o 2 body**. Tato zdánlivě zanedbatelná změna

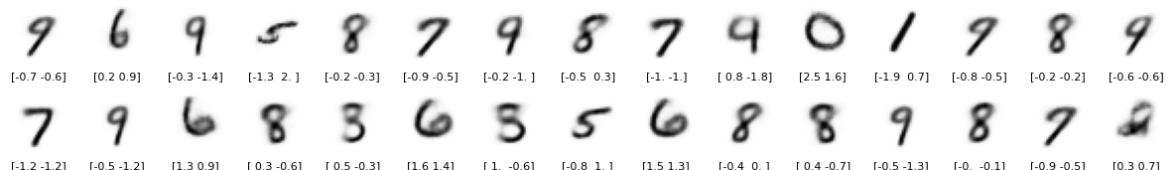
ve formování latentního prostoru se tak pojí s určitým skrytým významem, který vede k zpřesnění rekonstrukce. Vývoj hodnot ztrátové funkce modelu skrze různý počet epoch v trénovací fázi zachycuje Appendix B.

5.6 Vzorkování z latentního prostoru



Obrázek 5.12: Vzorkování z latentního prostoru naučeného modelu. Červené body značí bod z latentního prostoru, který byl použit jako vstup pro dekodér modul.

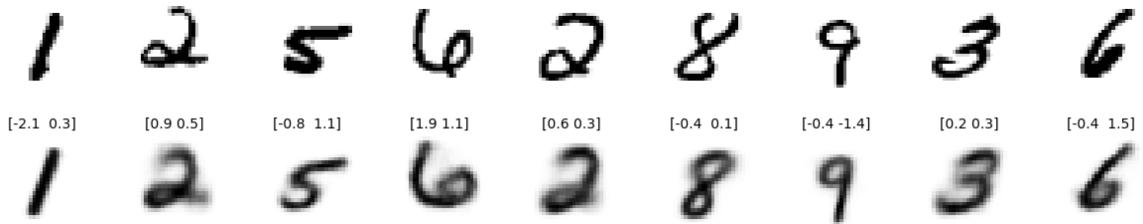
Obrázek 5.13 vzorky vygenerované naučeným modelem. Lze si povšimnout, že většina vygenerovaných čísel **není deformovaná**. To je důsledek **lokální spojistosti latentního prostoru** (v kontrastu s prostým autoenkovdrem, jehož latentní prostor je nespojitý).



Obrázek 5.13: Vzorky vygenerované naučeným modelem variačního autoenkovdru.

5.7 Evaluace

Obrázek 5.14 ukazuje schopnost modelu variačního autoenkodéru **generovat rekonstrukce** které splňují charakteristiky vstupu, ale **nebyly součástí trénovací množiny**.



Obrázek 5.14: Náhodné vzorky z MNIST datasetu \times jejich rekonstrukce vygenerované variacioním autoenkodérem. Hodnoty nad rekonstruovanou číslicí udávají souřadnice v latentním prostoru modelu, ze kterých byla tato rekonstrukce dekódována.

5.8 Diskuze

V této kapitole byl prezentován jednoduchý variační autoenkodér, **jehož latentní prostor má pouze 2 dimenze**. Což je užitečné pro vizualizace sloužící k prohledávání latentního prostoru.

Daleko lepších výsledků ale dosahují variační autoenkodéry, když dojde k zvýšení dimenze jejich latentního prostoru.

Představený model variačního autoenkodéru **je navržen tak, aby uměl pracovat s libovolnou sadou obrazových dat a rovněž uměl reprezentovat latentní prostor s vyšší dimenzí**.

Například, pro sestavení modelu variačního autoenkodéru pro generování realistických obličejů stačí pozměnit argumenty modelu následovně:

```
vae = VAE(
    input_dim=INPUT_DIM,
    encoder_conv_filters=[32, 64, 64, 64],
    encoder_conv_kernel_size=[3, 3, 3, 3],
    encoder_conv_strides=[2, 2, 2, 2],
    decoder_conv_t_filters=[64, 64, 32, 3],
    decoder_conv_t_kernel_size=[3, 3, 3, 3],
    decoder_conv_t_strides=[2, 2, 2, 2],
    z_dim=200,
    use_batch_norm=True,
    use_dropout=True
)
```

Což poukazuje na skvělou flexibilitu rámce variačního autoenkodéru a stojí za detailnější prozkoumání možnosti generalizace.

Závěr

The conclusion is a mandatory part of the bachelor's / diploma thesis. It contains a summary of the work and comments on the degree of fulfillment of the goal, which was set in the work, or summarizes the answers to the questions that were asked in the introduction.

The conclusion to the diploma thesis must be more elaborate - this is stated in more detail in the Requirements of the diploma thesis within the Intranet for FIS students.

The conclusion is perceived as a chapter, which begins on a separate page and is called the conclusion. The name Conclusion is not numbered. The text of the conclusion itself is divided into paragraphs.

Bibliografie

- BALDI, Pierre a HORNIK, Kurt, 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* [online]. Roč. 2, č. 1, s. 53–58 [cit. 2023-03-09]. ISSN 0893-6080. Dostupné z DOI: 10.1016/0893-6080(89)90014-2.
- BANERJEE, Arindam, 2007. An Analysis of Logistic Models: Exponential Family Connections and Online Performance. In: [online]. Society for Industrial and Applied Mathematics, s. 204–215 [cit. 2023-03-25]. Proceedings. ISBN 9780898716306. Dostupné z DOI: 10.1137/1.9781611972771.19.
- BELLMAN, Richard, 1957. *Dynamic Programming*. Princeton University Press. ISBN 9780691079516. Google-Books-ID: wdtoPwAACAAJ.
- BENGIO, Yoshua, COURVILLE, Aaron a VINCENT, Pascal, 2014. *Representation Learning: A Review and New Perspectives* [online]. 2014-04 [cit. 2023-03-25]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1206.5538. arXiv:1206.5538 [cs] type: article.
- BENGIO, Yoshua, LAMBLIN, Pascal, POPOVICI, Dan a LAROCHELLE, Hugo, 2006. Greedy Layer-Wise Training of Deep Networks. In: *Advances in Neural Information Processing Systems* [online]. MIT Press. Sv. 19 [cit. 2023-03-06]. Dostupné z: <https://proceedings.neurips.cc/paper/2006/hash/5da713a690c067105aeb2fae32403405-Abstract.html>.
- BENGIO, Yoshua, THIBODEAU-LAUFER, Éric, ALAIN, Guillaume a YOSINSKI, Jason, 2014. *Deep Generative Stochastic Networks Trainable by Backprop* [online]. 2014-05 [cit. 2023-04-05]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1306.1091. arXiv:1306.1091 [cs] type: article.
- BOWMAN, Samuel R., VILNIS, Luke, VINYALS, Oriol, DAI, Andrew M., JOZEFOWICZ, Rafal a BENGIO, Samy, 2016. *Generating Sentences from a Continuous Space* [online]. 2016-05 [cit. 2023-04-05]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1511.06349. arXiv:1511.06349 [cs] type: article.
- CYBENKO, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* [online]. Roč. 2, č. 4, s. 303–314 [cit. 2023-03-08]. ISSN 1435-568X. Dostupné z DOI: 10.1007/BF02551274.
- DAYAN, P., HINTON, G. E., NEAL, R. M. a ZEMEL, R. S., 1995. The Helmholtz machine. *Neural Computation*. Roč. 7, č. 5, s. 889–904. ISSN 0899-7667. Dostupné z DOI: 10.1162/neco.1995.7.5.889.

- DESHPANDE, Aditya, LU, Jiajun, YEH, Mao-Chuang, CHONG, Min Jin a FORSYTH, David, 2017. *Learning Diverse Image Colorization* [online]. 2017-04 [cit. 2023-04-09]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1612.01958. arXiv:1612.01958 [cs] type: article.
- DEVROYE, Luc, 1986. Sample-Based Non-Uniform Random Variate Generation. In: *Proceedings of the 18th Conference on Winter Simulation*. Washington, D.C., USA: Association for Computing Machinery, s. 260–265. WSC '86. ISBN 0911801111. Dostupné z DOI: 10.1145/318242.318443.
- DOERSCH, Carl, 2021. *Tutorial on Variational Autoencoders* [online]. 2021-01 [cit. 2023-03-31]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1606.05908. arXiv:1606.05908 [cs, stat] type: article.
- ERHAN, Dumitru, BENGIO, Yoshua, COURVILLE, Aaron, MANZAGOL, Pierre-Antoine, VINCENT, Pascal a BENGIO, Samy, 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* [online]. Roč. 11, č. 19, s. 625–660 [cit. 2023-03-06]. ISSN 1533-7928. Dostupné z: <http://jmlr.org/papers/v11/erhan10a.html>.
- FOSTER, David, 2023. *Generative Deep Learning: Teaching Machines To Paint, Write, Compose, and Play*. 2nd edition. O'Reilly Media. ISBN 9781098134181.
- GERON, Aurelien, 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc. ISBN 9781492032649.
- GERSHMAN, Samuel J. a GOODMAN, Noah D., 2014. Amortized Inference in Probabilistic Reasoning. *Cognitive Science*. Roč. 36.
- GÓMEZ-BOMBARELLI, Rafael, WEI, Jennifer N., DUVENAUD, David, HERNÁNDEZ-LOBATO, José Miguel, SÁNCHEZ-LENGELING, Benjamín, SHEBERLA, Dennis, AGUILERA-IPARRAGUIRRE, Jorge, HIRZEL, Timothy D., ADAMS, Ryan P. a ASPURU-GUZIK, Alán, 2018. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*. Roč. 4, č. 2, s. 268–276. Dostupné z DOI: 10.1021/acscentsci.7b00572. PMID: 29532027.
- GOODFELLOW, Ian, BENGIO, Yoshua a COURVILLE, Aaron, 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- GOODFELLOW, Ian J., POUGET-ABADIE, Jean, MIRZA, Mehdi, XU, Bing, WARDE-FARLEY, David, OZAIR, Sherjil, COURVILLE, Aaron a BENGIO, Yoshua, 2014. *Generative Adversarial Networks* [online]. 2014-06 [cit. 2023-03-09]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1406.2661. arXiv:1406.2661 [cs, stat] type: article.

- GREGOR, Karol, BESSE, Frederic, REZENDE, Danilo Jimenez, DANIHELKA, Ivo a WIERSTRA, Daan, 2016. *Towards Conceptual Compression* [online]. 2016-04 [cit. 2023-04-10]. Tech. zpr. arXiv. Dostupné z DOI: [10.48550/arXiv.1604.08772](https://doi.org/10.48550/arXiv.1604.08772). arXiv:1604.08772 [cs, stat] type: article.
- GREGOR, Karol, DANIHELKA, Ivo, GRAVES, Alex, REZENDE, Danilo Jimenez a WIERSTRA, Daan, 2015. *DRAW: A Recurrent Neural Network For Image Generation* [online]. 2015-05 [cit. 2023-04-06]. Tech. zpr. arXiv. Dostupné z DOI: [10.48550/arXiv.1502.04623](https://doi.org/10.48550/arXiv.1502.04623). arXiv:1502.04623 [cs] type: article.
- GREGOR, Karol, DANIHELKA, Ivo, MNIH, Andriy, BLUNDELL, Charles a WIERSTRA, Daan, 2014. Deep AutoRegressive Networks. In: XING, Eric P. a JEBARA, Tony (ed.). *Proceedings of the 31st International Conference on Machine Learning*. Bejing, China: PMLR. Sv. 32, s. 1242–1250. Proceedings of Machine Learning Research, č. 2. Dostupné také z: <https://proceedings.mlr.press/v32/gregor14.html>.
- GULRAJANI, Ishaan, KUMAR, Kundan, AHMED, Faruk, TAIGA, Adrien Ali, VISIN, Francesco, VAZQUEZ, David a COURVILLE, Aaron, 2016. *PixelVAE: A Latent Variable Model for Natural Images* [online]. 2016-11 [cit. 2023-04-09]. Tech. zpr. arXiv. Dostupné z DOI: [10.48550/arXiv.1611.05013](https://doi.org/10.48550/arXiv.1611.05013). arXiv:1611.05013 [cs] type: article.
- HEBB, Donald O., 1949. *The organization of behavior: A neuropsychological theory* [Hardcover]. New York: Wiley. ISBN 0-8058-4300-0.
- HIGGINS, Irina, MATTHEY, Loic, PAL, Arka, BURGESS, Christopher, GLOROT, Xavier, BOTVINICK, Matthew, MOHAMED, Shakir a LERCHNER, Alexander, 2022. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In: [online] [cit. 2023-05-06]. Dostupné z: <https://openreview.net/forum?id=Sy2fzU9gl>.
- HINTON, G. E., DAYAN, P., FREY, B. J. a NEAL, R. M., 1995. The "wake-sleep" algorithm for unsupervised neural networks. *Science (New York, N.Y.)* Roč. 268, č. 5214, s. 1158–1161. ISSN 0036-8075. Dostupné z DOI: [10.1126/science.7761831](https://doi.org/10.1126/science.7761831).
- HINTON, G. E. a ROWEIS, S., 2002. Stochastic Neighbor Embedding. In: BECKER, S., THRUN, S. a OBERMAYER, K. (ed.). *Advances in Neural Information Processing Systems*. MIT Press. Sv. 15. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.
- HINTON, Geoffrey E. a CAMP, Drew van, 1993. Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights. In: *Proceedings of the Sixth Annual Conference on Computational Learning Theory*.

- Santa Cruz, California, USA: Association for Computing Machinery, s. 5–13. COLT '93. ISBN 0897916115. Dostupné z DOI: 10.1145/168304.168306.
- HOCHREITER, Sepp, 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* [online]. Roč. 06, č. 02, s. 107–116 [cit. 2023-03-09]. ISSN 0218-4885. Dostupné z DOI: 10.1142/S0218488598000094.
- HORNIK, Kurt, STINCHCOMBE, Maxwell a WHITE, Halbert, 1990. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks* [online]. Roč. 3, č. 5, s. 551–560 [cit. 2023-03-08]. ISSN 0893-6080. Dostupné z DOI: 10.1016/0893-6080(90)90005-6.
- HORNIK, Kurt, STINCHCOMBE, Maxwell B. a WHITE, Halbert L., 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*. Roč. 2, s. 359–366.
- HUANG, Gary B., MATTAR, Marwan, LEE, Honglak a LEARNED-MILLER, Erik, 2012. Learning to Align from Scratch. In: *NIPS*.
- CHARTE, David, CHARTE, Francisco, GARCÍA, Salvador, JESÚS, María J. del a HERRERA, Francisco, 2018. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion* [online]. Roč. 44, s. 78–96 [cit. 2023-03-04]. ISSN 15662535. Dostupné z DOI: 10.1016/j.inffus.2017.12.007. arXiv:1801.01586 [cs].
- CHOLLET, François, 2017. *Deep Learning with Python*. Manning. ISBN 9781617294433.
- KAMYSHANSKA, Hanna a MEMISEVIC, Roland, 2013. On autoencoder scoring. In: KINGMA a BA, Jimmy, 2017. *Adam: A Method for Stochastic Optimization* [online]. 2017-01 [cit. 2023-05-07]. Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1412.6980. arXiv:1412.6980 [cs] type: article.
- KINGMA, D. P. a WELLING, M., 2014. Auto-Encoding Variational Bayes [online] [cit. 2023-03-24]. Dostupné z: <https://dare.uva.nl/search?identifier=cf65ba0fd88f-4a49-8ebd-3a7fce86edd7>.
- KINGMA, Diederik P. a WELLING, Max, 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* [online]. Roč. 12, č. 4, s. 307–392 [cit. 2023-03-09]. ISSN 1935-8237, ISSN 1935-8245. Dostupné z DOI: 10.1561/2200000056. arXiv:1906.02691 [cs, stat].
- KINGMA, Durk P., SALIMANS, Tim, JOZEFOWICZ, Rafal, CHEN, Xi, SUTSKEVER, Ilya a WELLING, Max, 2016. Improved Variational Inference with Inverse Autoregressive Flow. In: LEE, D., SUGIYAMA, M., LUXBURG, U., GUYON, I. a GARNETT, R. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Sv. 29. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

- KIRKPATRICK, S., GELATT, C. D. a VECCHI, M. P., 1983.
Optimization by Simulated Annealing. *Science* [online].
Roč. 220, č. 4598, s. 671–680 [cit. 2023-04-05].
Dostupné z DOI: [10.1126/science.220.4598.671](https://doi.org/10.1126/science.220.4598.671).
- KULKARNI, Tejas D., WHITNEY, Will, KOHLI, Pushmeet a TENENBAUM, Joshua B., 2015. *Deep Convolutional Inverse Graphics Network* [online]. 2015-06 [cit. 2023-04-04].
Tech. zpr. arXiv. Dostupné z DOI: [10.48550/arXiv.1503.03167](https://doi.org/10.48550/arXiv.1503.03167).
arXiv:1503.03167 [cs] type: article.
- KULLBACK, S. a LEIBLER, R. A., 1951. On Information and Sufficiency.
The Annals of Mathematical Statistics [online]. Roč. 22, č. 1, s. 79–86 [cit. 2023-03-09].
ISSN 0003-4851. Dostupné z: <https://www.jstor.org/stable/2236703>.
- LECUN, Y., BOSEN, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W. a JACKEL, L. D., 1989.
Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*.
Roč. 1, č. 4, s. 541–551. ISSN 0899-7667. Dostupné z DOI: [10.1162/neco.1989.1.4.541](https://doi.org/10.1162/neco.1989.1.4.541).
- LECUN, Yann, 2022. *A Path Towards Autonomous Machine Intelligence* [online]
[cit. 2023-03-25]. Dostupné z: <https://openreview.net/forum?id=BZ5a1r-kVsf>.
- LECUN, Yann a CORTES, Corinna, 2010. MNIST handwritten digit database
[<http://yann.lecun.com/exdb/mnist/>].
Dostupné také z: <http://yann.lecun.com/exdb/mnist/>.
- LIU, Huan a MOTODA, Hiroshi (ed.), 1998.
Feature Extraction, Construction and Selection [online].
Boston, MA: Springer US [cit. 2023-03-05]. ISBN 9781461376224 9781461557258.
Dostupné z DOI: [10.1007/978-1-4615-5725-8](https://doi.org/10.1007/978-1-4615-5725-8).
- LIU, Weifeng, POKHAREL, P.P. a PRINCIPE, J.C., 2006.
Correntropy: A Localized Similarity Measure. In:
The 2006 IEEE International Joint Conference on Neural Network Proceedings, s. 4919–4924. Dostupné z DOI: [10.1109/IJCNN.2006.247192](https://doi.org/10.1109/IJCNN.2006.247192).
- MINSKY, M. a PAPERT, S., 1969. *Perceptrons*. Cambridge, MA: MIT Press.
- MISHKIN, Dmytro, SERGIEVSKIY, Nikolay a MATAS, Jiri, 2017.
Systematic evaluation of CNN advances on the ImageNet.
Computer Vision and Image Understanding [online]. Roč. 161, s. 11–19 [cit. 2023-05-07].
ISSN 10773142. Dostupné z DOI: [10.1016/j.cviu.2017.05.007](https://doi.org/10.1016/j.cviu.2017.05.007). arXiv:1606.02228 [cs].
- MITCHELL, Tom M., 1997. *Machine Learning*. New York: McGraw-Hill.
ISBN 978-0-07-042807-2.
- MURPHY, Kevin P., 2022. *Probabilistic Machine Learning: An introduction*. MIT Press.
Dostupné také z: probml.ai.
- MURPHY, Kevin P., 2023. *Probabilistic Machine Learning: Advanced Topics*. MIT Press.
Dostupné také z: <http://probml.github.io/book2>.

- OLSHAUSEN, Bruno A. a FIELD, David J., 1997.
Sparse coding with an overcomplete basis set: A strategy employed by V1?
Vision Research [online]. Roč. 37, č. 23, s. 3311–3325 [cit. 2023-03-09]. ISSN 0042-6989.
Dostupné z DOI: 10.1016/S0042-6989(97)00169-7.
- PHILLIPS, Jeff M., 2021. *Mathematical Foundations for Data Analysis*.
Springer International Publishing. ISBN 9783030623401.
Google-Books-ID: AUDYzQEACAAJ.
- RANZATO, M.A., HUANG, Fu, BOUREAU, Y-Lan a LECUN, Yann, 2007. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In: s. 1–8. Dostupné z DOI: 10.1109/CVPR.2007.383157.
- RAVANBAKHSH, Siamak, LANUSSE, Francois, MANDELBAUM, Rachel, SCHNEIDER, Jeff a POCZOS, Barnabas, 2016.
Enabling Dark Energy Science with Deep Generative Models of Galaxy Images [online]. 2016-11 [cit. 2023-04-09]. Tech. zpr. arXiv.
Dostupné z DOI: 10.48550/arXiv.1609.05796.
arXiv:1609.05796 [astro-ph, stat] type: article.
- REZENDE, Danilo Jimenez, MOHAMED, Shakir a WIERSTRA, Daan, 2014. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models* [online]. 2014-05 [cit. 2023-03-24]. Tech. zpr. arXiv.
Dostupné z DOI: 10.48550/arXiv.1401.4082. arXiv:1401.4082 [cs, stat] type: article.
- RIFAI, Salah, BENGIO, Yoshua, DAUPHIN, Yann a VINCENT, Pascal, 2012.
A Generative Process for Sampling Contractive Auto-Encoders [online]. 2012-06 [cit. 2023-03-09]. Tech. zpr. arXiv.
Dostupné z DOI: 10.48550/arXiv.1206.6434. arXiv:1206.6434 [cs, stat] type: article.
- ROSENBLATT, F., 1957. *The perceptron - A perceiving and recognizing automaton*. Ithaca, New York, 1957-01. Tech. zpr., 85-460-1. Cornell Aeronautical Laboratory.
- RUMELHART, David E. a MCCLELLAND, James L., 1987.
Learning Internal Representations by Error Propagation. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, s. 318–362.
- SALAKHUTDINOV, Ruslan a LAROCHELLE, Hugo, 2010.
Efficient Learning of Deep Boltzmann Machines.
In: TEH, Yee Whye a TITTERINGTON, Mike (ed.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR. Sv. 9, s. 693–700.
Proceedings of Machine Learning Research.
Dostupné také z: <https://proceedings.mlr.press/v9/salakhutdinov10a.html>.
- SAMUEL, Arthur L., 1967.
Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. Dev.* Roč. 44, s. 206–227.

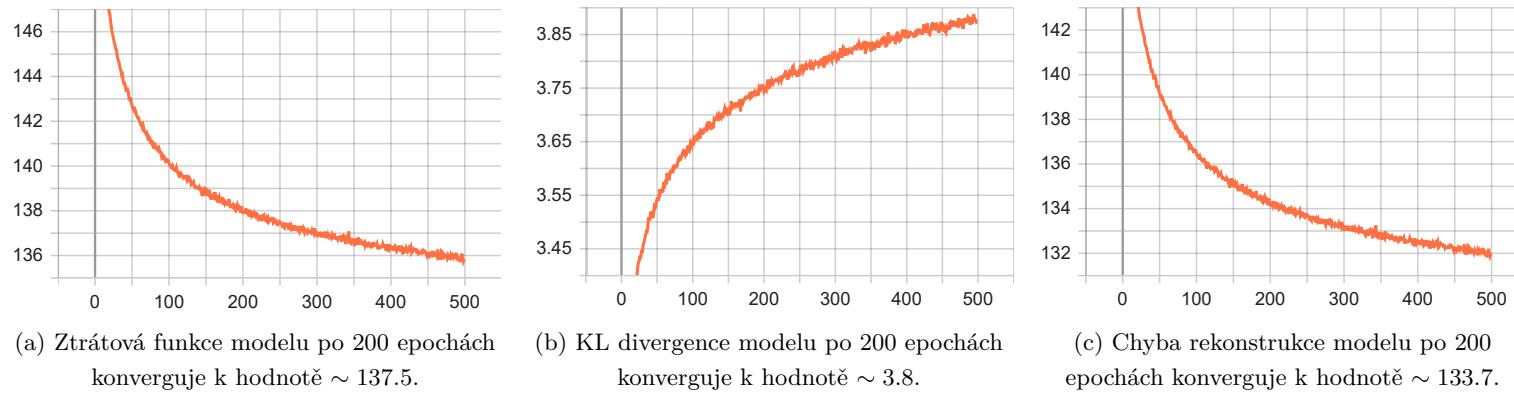
- SANKARAPANDIAN, Sivaramakrishnan a KULIS, Brian, 2021.
\$\beta\$-Annealed Variational Autoencoder for glitches [online]. 2021-07 [cit. 2023-05-06].
Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.2107.10667.
arXiv:2107.10667 [gr-qc] type: article.
- SOHN, Kihyuk, LEE, Honglak a YAN, Xinchen, 2015.
Learning Structured Output Representation using Deep Conditional Generative Models.
In: CORTES, C., LAWRENCE, N., LEE, D., SUGIYAMA, M. a GARNETT, R. (ed.).
Advances in Neural Information Processing Systems. Curran Associates, Inc. Sv. 28.
Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.
- SØNDERBY, Casper, RAIKO, Tapani, MAALØE, Lars, SØNDERBY, Søren a
WINTHER, Ole, 2016.
How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks.
- STAŃCZYK, Urszula a JAIN, Lakhmi C. (ed.), 2015.
Feature Selection for Data and Pattern Recognition [online].
Berlin, Heidelberg: Springer Berlin Heidelberg [cit. 2023-03-05].
Studies in Computational Intelligence. ISBN 9783662456194 9783662456200.
Dostupné z DOI: 10.1007/978-3-662-45620-0.
- TOPSØE, Flemming, 1974. Compactness and Tightness in a Space of Measures with the
Topology of Weak Convergence. *Mathematica Scandinavica* [online].
Roč. 34, č. 2, s. 187–210 [cit. 2023-03-26]. ISSN 0025-5521.
Dostupné z: <https://www.jstor.org/stable/24490646>.
- VALIANT, L. G., 1984. A theory of the learnable. *Communications of the ACM* [online].
Roč. 27, č. 11, s. 1134–1142 [cit. 2023-03-08]. ISSN 0001-0782.
Dostupné z DOI: 10.1145/1968.1972.
- VINCENT, Pascal, LAROCHELLE, Hugo, LAJOIE, Isabelle, BENGIO, Yoshua a
MANZAGOL, Pierre-Antoine, 2010. Stacked Denoising Autoencoders: Learning Useful
Representations in a Deep Network with a Local Denoising Criterion.
J. Mach. Learn. Res. Roč. 11, s. 3371–3408. ISSN 1532-4435.
- WAH, Catherine, BRANSON, Steve, WELINDER, Peter, PERONA, Pietro a
BELONGIE, Serge, 2011. *The Caltech-UCSD Birds-200-2011 Dataset* [online].
Pasadena, CA: California Institute of Technology [cit. 2023-04-10]. Č. 2010-001.
Dostupné z: <https://resolver.caltech.edu/CaltechAUTHORS:20111026-120541847>.
- WASSERMAN, L., 2013. *All of Statistics: A Concise Course in Statistical Inference*.
Springer New York. Springer Texts in Statistics. ISBN 9780387217369.
Dostupné také z: <https://books.google.cz/books?id=qrcuBAAQBAJ>.
- WERBOS, P. J., 1974.
Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.
Dis. pr. Harvard University.

- WHITE, Tom, 2016. *Sampling Generative Networks* [online]. 2016-12 [cit. 2023-04-09].
Tech. zpr. arXiv. Dostupné z DOI: 10.48550/arXiv.1609.04468.
arXiv:1609.04468 [cs, stat] type: article.
- WOLPERT, David H., 1996.
The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*.
Roč. 8, č. 7, s. 1341–1390. ISSN 0899-7667.
Dostupné z DOI: 10.1162/neco.1996.8.7.1341.
- YAN, Xinchen, YANG, Jimei, SOHN, Kihyuk a LEE, Honglak, 2016.
Attribute2Image: Conditional Image Generation from Visual Attributes [online].
2016-10 [cit. 2023-04-10]. Tech. zpr. arXiv.
Dostupné z DOI: 10.48550/arXiv.1512.00570. arXiv:1512.00570 [cs] type: article.

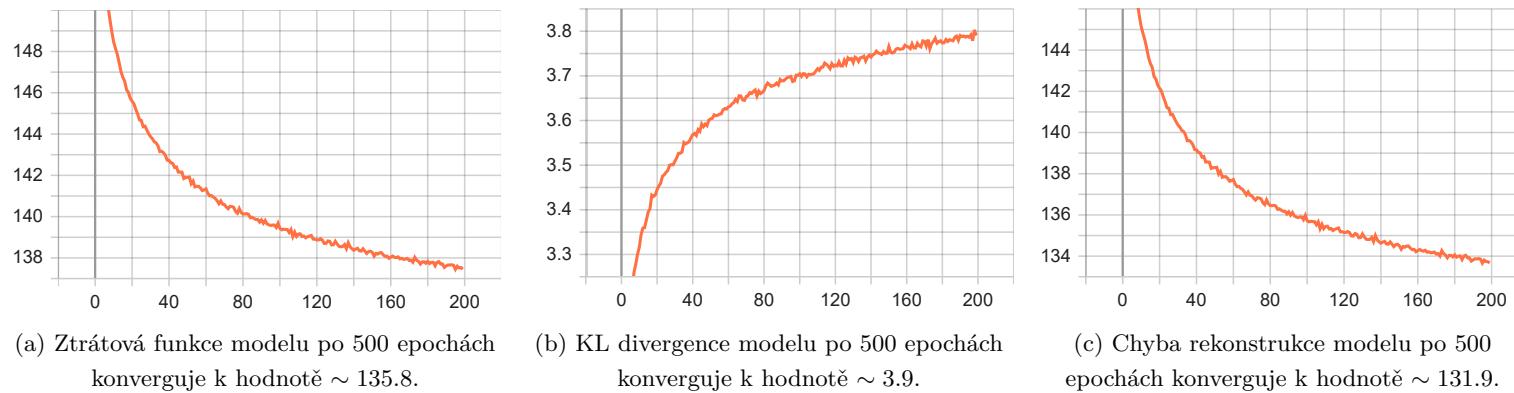
Přílohy

A. Zdrojové kódy modelů

B. Hodnoty ztrátové funkce modelu variačního autoenkodéru



Obrázek B.1: Hodnota ztrátové funkce modelu variačního autoenkodéru po 200 epochách trénování.



Obrázek B.2: Hodnota ztrátové funkce modelu variačního autoenkodéru po 500 epochách trénování.