

Kompaktowe reprezentacje binarne i modele generatywne

(Compact binary representations and generative models)

Jakub Zadrożny

Praca licencjacka

Promotor: dr Rafał Nowak

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

6 września 2019

Streszczenie

Niniejsza praca przedstawia możliwości zastosowania autoenkodera wariacyjnego (VAE) do znalezienia reprezentacji (w tym kompatkowej reprezentacji binarnej) dla chmur punktów 3D. W pracy opisano teoretyczne podstawy uczenia autoenkoderek wariacyjnych z regularyzacją zmiennej ukrytej rozkładem normalnym. Przedstawiono również metodę wykorzystywaną do wymuszenia bardziej skomplikowanych rozkładów na zmiennej ukrytej (m.in. Gamma, Beta oraz mieszanek gaussowskich). Opisano również, w jaki sposób wykorzystać regularyzację mieszanką gaussowską do podzielenia danych na naturalne klastry. Ponadto zaproponowano sposób użycia VAE z regularizacją rozkładem Beta do znalezienia kompatkowej reprezentacji binarnej dla chmur punktów 3D. Opisane modele zaimplementowano i przeprowadzono szereg eksperymentów sprawdzających ich zdolności rekonstrukcji oraz możliwości generatywne.

Abstract

The following work demonstrates the possibility of using a variational autoencoder (VAE) for obtaining representations (including a compact binary one) for 3D point clouds. The paper includes theoretical baseline study of training variational autoencoders with a normally-distributed latent variable. A method for regularization of the latent variable with a more complex distribution (such as Gamma, Beta or a gaussian mixture) has also been presented. The work includes also a description of how to use a variational autoencoder regularized with a gaussian mixture to divide data into natural clusters. Moreover the paper contains a proposed method of utilising a Beta-regularized VAE in order to obtain a compact binary representation for 3D point clouds. All described models have later been implemented and tested with a variety of experiments focusing on their reconstructive and generative capabilities.

Spis treści

1. Wprowadzenie	7
2. Podstawy opracowania	9
2.1. Chmury punktów 3D	9
2.2. Autoenkodery	10
2.3. Modele generatywne	11
2.4. Estymacja gradientu wartości randomizowanych	11
3. Metody	13
3.1. Autoenkoder wariacyjny z rozkładem normalnym	13
3.1.1. ELBO	13
3.1.2. Zadanie optymalizacyjne	14
3.1.3. Koszt KL	15
3.1.4. Koszt rekonstrukcji	15
3.2. Autoenkoder wariacyjny z mieszaną gaussowską	17
3.2.1. ELBO	18
3.3. Autoenkoder wariacyjny z rozkładem Beta	19
3.3.1. Koszt KL	20
3.3.2. Koszt rekonstrukcji	20
4. Implementacja i architektura modeli	23
4.1. Zbiór danych i trenowanie	24
5. Wyniki eksperymentalne	27
5.1. Zdolność rekonstrukcji	27

5.2. Możliwości generatywne	27
5.3. Dodatkowe eksperymenty	30
5.4. Klasteryzacja	33
5.5. Binaryzacja reprezentacji	34
6. Wnioski	39
Bibliografia	41

Rozdział 1.

Wprowadzenie

Modele generatywne, takie jak autoenkoder wariacyjny (w skrócie **VAE**), stały się obecnie jednym z podstawowych narzędzi do pracy z wielomammi rodzajami danych od zdjęć twarzy do plików muzycznych. Umożliwiają one znajdowanie efektywnych i interpretowalnych reprezentacji, które pozwalają nie tylko na generowanie syntetycznych danych, ale rozwiązuja również problemy takie jak gładka interpolacja pomiędzy dwoma obiektami, czy edycja obiektów poprzez intuicyjne operacje.

Obecnie jednak, dzięki popularyzacji urządzeń reprezentujących głębię obrazu, takich jak LIDAR i kamera RGB-D, coraz większą rolę odgrywają chmury punktów 3D. Wykorzystuje się je w wielu nowoczesnych, rewolucyjnych i kluczowych dla postępu technologiiach, m.in. pojazdach autonomicznych, robotyce oraz wirtualnej i rozszerzonej rzeczywistości. Chmury punktów są dość skomplikowanymi danymi, które zajmują znacznie ilości miejsca i przetwarzane w oryginalnej postaci nakładają spory narzut obliczeniowy. Dlatego istotnym zagadnieniem jest znalezienie odpowiedniej reprezentacji dla takich danych. Taka reprezentacja musi być kompaktowa, aby zmniejszyć narzut pamięciowy i obliczeniowy, a model powinien być generatywny, żeby umożliwić tworzenie i edycję obiektów.

Chmury punktów 3D są aktywnym obszarem badań i na przestrzeni ostatnich lat powstało zarówno wiele problemów (np. zbiory ShapeNet [1] i ModelNet40 [2]), jak i rozwiązań (np. architektura PointNet [3] i metryka EMD [4]). Niektóre z tych osiągnięć pozwalają tworzyć kolejne, bardziej zaawansowane modele, m.in. opisany w tej pracy autoenkoder wariacyjny, który na podstawie znalezionej reprezentacji potrafi rekonstruować i generować nowe chmury punktów.

Jednym ze znanych problemów VAE, jest trudność narzucenia na zmienną ukrytą niektórych mniej typowych rozkładów. W dalszej części pracy opisano znane rozwiązania tego problemu, rozszerzające możliwości enkodera, dzięki którym da się regularyzować zmienną ukrytą np. rozkładem Gamma, Beta oraz mieszaną gaussowską. Wykazano również, w jaki sposób wykorzystać regularyzację reprezentacji mieszaną gaussowską, aby podzielić dane na naturalne klastry, zgodnie z ich ce-

chami charakterystycznymi, w sposób w pełni nienadzorowany. Dodatkowo podano metodę, która pozwala wykorzystać VAE do znalezienia kompaktowej reprezentacji binarnej. Wykorzystanie takiej reprezentacji pozwala rozwiązać kluczowy w dobie wszechobecnych danych problem – można drastycznie ograniczyć ilość przechowywanych i przetwarzanych informacji. Uzyskana reprezentacja zajmuje jedynie 128 bitów na jedną chmurę, co przy prawie 200kb w oryginalnej reprezentacji daje ponad 1500-krotną skalę kompresji. Ponadto reprezentacja binarna zawiera tyle informacji, ile zaledwie czterowymiarowa reprezentacja ciągła, jednak posiada większą zdolność rekonstrukcji, nie poświęcając przy tym możliwości generatywnych.

Podsumowując, praca składa się z czterech głównych części: (1) zawiera opis teoretycznych podstaw autoenkodera wiarygodnego, (2) podaje sposoby wymuszenia różnych rozkładów na zmiennej ukrytej, (3) zawiera eksperymenty demonstrujące zdolności wytrenowanego modelu do rekonstrukcji, generowania syntetycznych danych, interpolacji, edycji obiektów przez intuicyjne operacje oraz klasteryzacji i (4) opisuje sposób, w jaki można wykorzystać VAE z regularizacją rozkładem Beta do otrzymania kompaktowej (128 bitów) reprezentacji binarnej chmur punktów i potwierdza skuteczność tej reprezentacji eksperymentami.

Rozdział 2.

Podstawy opracowania

2.1. Chmury punktów 3D

Chmury punktów 3D reprezentują obiekty, kształty, powierzchnie w przestrzeni za pomocą zbioru trójkę liczb rzeczywistych: współrzędnych x, y, z każdego z punktów. Pojedynczą chmurę punktów reprezentuje zatem macierz wymiaru $3 \times N$, gdzie N – liczba punktów. Ponieważ chmury reprezentowane są przez *zbior* punktów, przyjmujemy, że dwie macierze różniące się tylko kolejnością kolumn przedstawiają w rzeczywistości tę samą chmurę. Stwarza to dwa istotne problemy: (1) cały model powinien być niewrażliwy na permutacje kolumn macierzy wejściowej, (2) dekoder trzeba oceniać metryką niewrażliwą na permutacje punktów.

Architektura PointNet

Silnym narzędziem do pracy z chmurami punktów 3D jest architektura **PointNet** [3], która umożliwia ekstrakcję cech (w pełni sparametryzowanych) z wejściowej chmury punktów. Architektura ta posiada dwie podstawowe zalety istotne dla tej pracy: jest odporna na permutacje punktów chmury wejściowej i przyjmuje jako wejście chmury w naturalnej postaci. Inne metody [5, 2] rozwiązały problem permutacji przetwarzając chmury punktów na inne formaty (np. wykorzystujące voxele), przez co rozmiar przetwarzanych danych i czas obliczeń wzrasta. PointNet osiąga ten sam cel, zachowując względną prostotę modelu, co z uwagi na ograniczone zasoby obliczeniowe jest kluczowe.

Metryki

Powszechnie stosuje się dwie metryki dla chmur punktów 3D; zob. np. [6]. Założymy, że dysponujemy dwoma zbiorami $S_1 \in \mathbb{R}^3$ i $S_2 \in \mathbb{R}^3$.

Metryka *Chamfer* pseudo-distance (**CD**) dla każdego punktu z S_1 oblicza odległość od najbliższego punktu w S_2 i odwrotnie. Dana jest następującym wzorem:

$$CD(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2. \quad (2.1)$$

Metryka Earth-mover distance (**EMD**) [4] znajduje permutację punktów wyjściowego zbioru, dla której suma odległości punkt po punkcie jest najmniejsza. Dana jest wzorem:

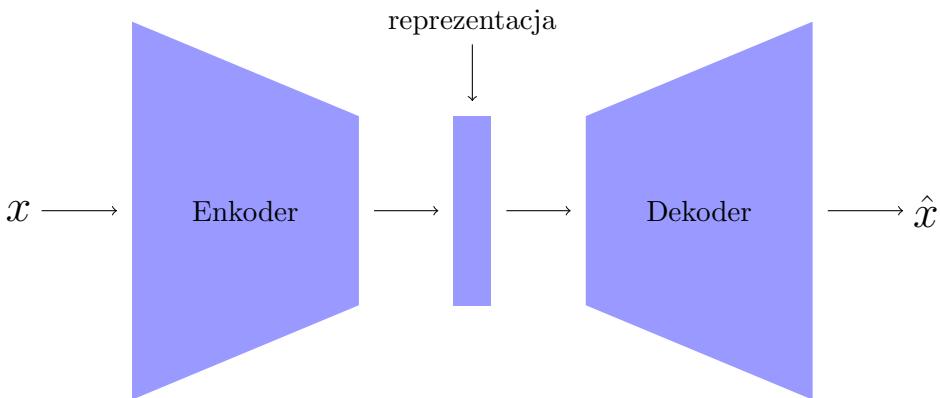
$$EMD(S_1, S_2) = \min_{\phi: S_1 \rightarrow S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2^2, \quad (2.2)$$

gdzie ϕ jest bijekcją.

Jak zaznaczają autorzy [7], użycie metryki EMD pozwala uzyskać lepsze rezultaty w rekonstrukcji obiektów, jednak jej wyliczanie jest znacznie bardziej kosztowne niż CD, dlatego w tej pracy skupiono się na metryce CD.

2.2. Autoenkodery

Podstawowym modelem opisywanym w tej pracy jest autoenkoder [8]. Autoenkoder to model złożony z dwóch sieci neuronowych: **enkodera** i **dekodera**. Sieci trenuje się wspólnie, w taki sposób, aby na podstawie wyniku zwróconego przez enkoder, dekoder był w stanie zrekonstruować oryginalne dane (wejście do enkodera). Autoenkodery stają się praktyczne, gdy nałoży się ograniczenie na dane przekazywane pomiędzy enkoderem i dekoderem (nazywane zmienną ukrytą lub reprezentacją), np. ograniczy się mocno ich wymiar. Schemat działania enkodera przedstawia rys. 2.1.



Rysunek 2.1: Schemat działania autoenkodera. Celem jest minimalizacja odległości pomiędzy \hat{x} i x .

2.3. Modele generatywne

Klasycznym wynikiem, który spopularyzował modele generatywne, jest autoenkoder wariacyjny (**VAE**) [9]. Jego najważniejszą zaletą jest probabilistyczny charakter, dzięki czemu możemy żądać, aby zmienna ukryta (reprezentacja) była zradnomizowana i pochodziła z pewnego zadanego rozkładu prawdopodobieństwa (z pewnymi ograniczeniami). Dzięki randomizacji zmiennych ukrytych uzyskujemy zagęszczenie przestrzeni reprezentacji, przez co niemal każdy punkt może zostać zdekodowany na realistyczne dane. Umożliwia to realizowanie zadań takich jak generowanie syntetycznych, realistycznych danych lub gładkie interpolacje znaczeniowe obiektów.

Modele generatywne są szeroko zbadane i często stosowane do różnych problemów, m.in. analizy obrazów. W przypadku chmur punktów 3D ich możliwości i szczegółowe zastosowania są mniej znane. Jedne z pierwszych opracowań tego tematu można znaleźć w [7] (autorzy wykorzystują zwykły autoenkoder do znalezienia reprezentacji, na której uczą modele generatywne – m.in. model mieszanek gaussowskich oraz GANy [10]) i [11] (praca skupia się na zastosowaniu autoenkoderów generatywnych: VAE oraz autoenkodera przeciwnego wykorzystywanego do wymuszenia ciekawych rozkładów na zmiennej ukrytej).

Klasyfikacja i klasteryzacja

Modele generatywne można również zaadaptować do celów najczęściej rozwiązywanych z użyciem tradycyjnych metod – klasyfikacji i klasteryzacji. W pracy [12] zaproponowano modyfikację autoenkodera wariacyjnego, dzięki której mógłby on pełnić rolę klasyfikatora trenowanego w sposób pół-nadzorowany (model M2). Choć jednak podejście to zawodzi, gdy zastosuje się je w sposób całkowicie nienadzorowany, to w literaturze [13, 14] można znaleźć dodatkowe modyfikacje, które umożliwiają modelowi M2 klasteryzację danych w sposób nienadzorowany. W dalszej części pracy opisano dokładnie model oparty na autoenkoderze wariacyjnym, który potrafi dzielić dane na naturalne podkategorie.

2.4. Estymacja gradientu wartości randomizowanych

Do wytrenowania autoenkodera wariacyjnego używa się najczęściej wariantu metody *stochastic gradient descent* (**SGD**), która wymaga wyliczania pochodnych funkcji kosztu po kolejnych parametrach modelu, co z kolei wymaga wyliczania pochodnych wartości pojawiających się we wszystkich warstwach po parametrach warstw poprzednich (propagacja wstecz). Do wyliczenia funkcji kosztu w VAE używa się szacowania Monte Carlo, które opiera się na próbkowaniu zmiennej losowej. Oznacza to, że do wytrenowania autoenkodera wariacyjnego niezbędne jest

obliczanie pochodnych wylosowanej zmiennej po parametrach rozkładu. W tym celu stosuję się dwie postawowe metody.

Estymator **score function** opisany m.in. w [15] oferuje stosunkowo proste przybliżenie gradientu funkcji kosztu i można go stosować do szerokiej gamy rozkładów, jednak charakteryzuje się dużą wariancją oszacowania, co komplikuje proces trenowania.

Drugim podejściem jest **pathwise gradients**, czyli tzw. trik reparametryzacyjny opisywany m.in. w [9]. Metoda ta opiera się na przedstawieniu zmiennej z pochodzącej z rozkładu $q_\theta(z)$, sparametryzowanego przez θ , jako $z = \mathcal{T}(\epsilon; \theta)$, gdzie ϵ pochodzi z pewnego rozkładu q_0 niezależnego od θ , a \mathcal{T} jest obliczalnym i różniczkowalnym po θ przekształceniem. Trik reparametryzacyjny jest łatwy do zastosowania dla części rozkładów ciągłych (szczególnie dla tych z rodziny kształtu i skali), jednak z uwagi na brak prostego przekształcenia \mathcal{T} , nie da się go wprost zastosować do niektórych istotnych rozkładów, np. Gamma i Beta.

Rozszerzenie tej metody na kolejne rodziny rozkładów zostało zaproponowane w [16]. Wprowadzona metoda, wymaga jedynie istnienia pochodnej dystrybuanty po parametrach rozkładu w postaci zwartej lub znalezienia jej dobrego przybliżenia numerycznego. Mimo niewielkich wymagań, rozwiązanie pozwala w efektywny sposób szacować gradienty funkcji kosztu, nawet gdy zmienne pośrednie pochodzą ze skomplikowanych rozkładów jak Gamma i Beta.

Rozdział 3.

Metody

3.1. Autoenkoder wariacyjny z rozkładem normalnym

Możliwości i metodyka pracy z autoenkoderami wariacyjnymi jest szeroko zbadana pod kątem niektórych zagadnień, m.in. analizy obrazów. Jednak zastosowanie ich do stosunkowo nowych danych, jakimi są chmury punktów 3D, wymaga opracowania dokładniejszej metody.

Dalej zakładamy, że dysponujemy zbiorem

$$\mathcal{X} = \{x_i \in \mathbb{R}^d\}_{i \in I}$$

dla pewnego d – wymiaru danych oraz I – zbioru indeksów. Ponadto zakładamy, że dane z tego zbioru są obserwacjami zmiennej losowej o rozkładzie następującej postaci

$$f_\theta(z, x) = f(z)f_\theta(x|z), \quad (3.1)$$

gdzie $z \in \mathbb{R}^k$ dla pewnego k – zmienna ukryta (*latent*) o wymiarze k , $x \in \mathbb{R}^d$ – zmienna obserwowana o wymiarze d , a θ – parametry rozkładu.

Dodatkowo niech

$$\begin{aligned} z &\sim \mathcal{N}(0, I_k), \\ x|z &\sim \mathcal{N}(\mu_x(z; \theta), \mu_\sigma(z; \theta)I_d), \end{aligned} \quad (3.2)$$

gdzie μ_x , μ_σ są skomplikowanymi obliczeniami wykonywanymi przez sieć neuronową sparametryzowaną przez θ .

3.1.1. ELBO

Naszym celem jest odtworzenie parametrów θ rozkładu generującego oraz rozkładu $f_\theta(z|x)$, który wyznacza reprezentację danych generowanych przez proces opisany w (3.1) i (3.2).

Niestety z powodu zastosowania skomplikowanych, nieliniowych transformacji dokładne odtworzenie rozkładu $f_\theta(z|x)$ jest niemożliwe. W tym celu wprowadzamy pewne przybliżenie tego rozkładu – nazwijmy je $g_\phi(z|x)$. Niech $g_\phi(z|x)$ będzie gęstością rozkładu normalnego ze średnią $\rho_x(x; \phi)$ i wariancją $\rho_\sigma(x; \phi)$, gdzie ρ_x, ρ_σ są reprezentowane przez sieci neuronowe parametryzowane przez ϕ .

Dla ustalonego x mamy

$$\begin{aligned} KL(g_\phi(z|x) || f_\theta(z|x)) &= \mathbb{E}_{z \sim g_\phi(z|x)} \left[-\log \frac{f_\theta(z|x)}{g_\phi(z|x)} \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} \left[-\log \frac{f_\theta(z|x)f_\theta(x)}{g_\phi(z|x)f_\theta(x)} \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} \left[-\log \frac{f_\theta(z|x)f_\theta(x)}{g_\phi(z|x)} + \log f_\theta(x) \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} \left[-\log \frac{f_\theta(z, x)}{g_\phi(z|x)} \right] + \log f_\theta(x), \end{aligned}$$

gdzie $KL(\cdot || \cdot)$ jest odległością Kullbacka-Leiblera [17]. Zatem

$$\log f_\theta(x) = KL(g_\phi(z|x) || f_\theta(z|x)) + \mathbb{E}_{z \sim g_\phi(z|x)} \left[\log \frac{f_\theta(z, x)}{g_\phi(z|x)} \right].$$

Ponieważ $KL(\cdot || \cdot) \geq 0$, więc

$$\begin{aligned} \log f_\theta(x) &\geq \mathbb{E}_{z \sim g_\phi(z|x)} \left[\log \frac{f_\theta(z, x)}{g_\phi(z|x)} \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} \left[\log \frac{f_\theta(x|z)f(z)}{g_\phi(z|x)} \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} [\log f_\theta(x|z)] - \mathbb{E}_{z \sim g_\phi(z|x)} \left[-\log \frac{f(z)}{g_\phi(z|x)} \right] \\ &= \mathbb{E}_{z \sim g_\phi(z|x)} [\log f_\theta(x|z)] - KL(g_\phi(z|x) || f(z)). \end{aligned} \tag{3.3}$$

Zatem dla dowolnego rozkładu aproksymującego $g_\phi(z|x)$ otrzymujemy dolne ograniczenie na prawdopodobieństwo wygenerowania zaobserwowanych danych. Dlatego część wzoru po prawej stronie od ostatniej równości nazywamy *evidence lower bound* (w skrócie **ELBO**).

Możemy zauważyć, że ostateczna postać wzoru (3.3) składa się z dwóch części naturalnie odpowiadającymi dwóm celom, które chcemy zoptymalizować: pierwszy składnik odpowiada jakości rekonstrukcji obserwacji ze zmiennej ukrytej z (dlatego nazywany jest kosztem rekonstrukcji), natomiast drugi to odległość KL rozkładu aproksymującego $f_\theta(z|x)$ od naszego założenia na jego temat.

3.1.2. Zadanie optymalizacyjne

Chcemy znaleźć układ parametrów $\langle \theta, \phi \rangle$, który daje najlepszą gwarancję na prawdopodobieństwo wygenerowania zaobserwowanych danych (ELBO). W tym

celu posłużymy się lekko zmodyfikowanym algorytmem SGD. Niech

$$\mathcal{L}(x, \theta, \phi) = \underbrace{\mathbb{E}_{z \sim g_\phi(z|x)} [\log f_\theta(x|z)] - KL(g_\phi(z|x)||f(z))}_{\mathcal{L}_{rec}(x, \theta, \phi)}. \quad (3.4)$$

Korzystając z (3.3) dla każdej pary θ, ϕ mamy

$$\log \prod_{x \in \mathcal{X}} f_\theta(x) = \sum_{x \in \mathcal{X}} \log f_\theta(x) \geq \sum_{x \in \mathcal{X}} \mathcal{L}(x, \theta, \phi) \stackrel{\text{def}}{=} \hat{\mathcal{L}}(\mathcal{X}, \theta, \phi).$$

Naszym zadaniem jest znalezienie $\hat{\theta}, \hat{\phi}$, takich że

$$\hat{\mathcal{L}}(\mathcal{X}, \hat{\theta}, \hat{\phi}) = \max_{\theta, \phi} \hat{\mathcal{L}}(\mathcal{X}, \theta, \phi).$$

Ponieważ bardziej naturalnym zadaniem jest minimalizowanie funkcji kosztu, więc rozwiążemy równoważne zadanie znalezienia $\hat{\theta}, \hat{\phi}$, takich że

$$-\hat{\mathcal{L}}(\mathcal{X}, \hat{\theta}, \hat{\phi}) = \min_{\theta, \phi} -\hat{\mathcal{L}}(\mathcal{X}, \theta, \phi).$$

Aby posłużyć się algorytmem SGD, musimy umieć wyliczać i różniczkować oba składniki funkcji kosztu \mathcal{L} (zob. 3.4).

3.1.3. Koszt KL

Odległość KL dwóch rozkładów normalnych o następujących parametrach

$$\begin{aligned} \mathcal{N}_0 &\sim \mathcal{N}(\mu_0, \Sigma_0), \\ \mathcal{N}_1 &\sim \mathcal{N}(\mu_1, \Sigma_1) \end{aligned}$$

dla pewnych $\mu_0, \mu_1 \in \mathbb{R}^k, \Sigma_0, \Sigma_1 \in \mathbb{R}^{k \times k}$, wynosi

$$KL(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2} \left(\text{tr}(\Sigma_1^{-1} \Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) - k + \log \frac{\det \Sigma_1}{\det \Sigma_0} \right).$$

Ponieważ zakładamy, że $f(z)$ jest rozkładem $z \sim \mathcal{N}(0, I_k)$, więc

$$KL(g_\phi(z|x)||f(z)) = \frac{1}{2} \sum_{i=1}^k (\rho_x(x; \phi)_i^2 + \rho_\sigma(x; \phi)_i^2 - 2 \log(\rho_\sigma(x; \phi)_i) - 1). \quad (3.5)$$

Wzór (3.5) można wyliczać i różniczkować analitycznie.

3.1.4. Koszt rekonstrukcji

Drugiego składnika funkcji \mathcal{L} , czyli kosztu rekonstrukcji \mathcal{L}_{rec} , nie da się wyznaczyć analitycznie. Aby objeść ten problem, możemy metodą Monte Carlo oszacować wartość oczekiwana przez średnią:

$$-\mathcal{L}_{rec}(x, \theta, \phi) = \mathbb{E}_{z \sim g_\phi(z|x)} [-\log f_\theta(x|z)] \approx \frac{1}{m} \sum_{i=1}^m -\log f_\theta(x|z_i),$$

gdzie z_i są obserwacjami rozkładu $g_\phi(z|x)$. Taką wartość potrafimy już wyliczyć, ale nie potrafimy propagować gradientu do parametrów ϕ przez zaobserwowane wartości z_i .

Wprowadzimy reparametryzację zmiennych z_i – możemy zauważyć, że zmienna $z_i = \rho_x(x; \phi) + \epsilon_i \rho_\sigma(x; \phi)$ gdzie $\epsilon_i \sim \mathcal{N}(0, I_k)$ ma rozkład $g_\phi(z|x)$, a ponadto możemy różniczkować z_i po $\rho_x(x; \phi), \rho_\sigma(x; \phi)$ i propagować gradient wstecz do parametrów ϕ . Otrzymaliśmy zatem następujące przybliżenie:

$$-\mathcal{L}_{rec}(x, \theta, \phi) \approx \frac{1}{m} \sum_{i=1}^m -\log f_\theta(x|z_i = \rho_x(x; \phi) + \epsilon_i \rho_\sigma(x; \phi)),$$

gdzie $\epsilon_i \sim \mathcal{N}(0, I_k)$. Dalej, dla uproszczenia, przyjmujemy $m = 1$.

Niech

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}, \quad \mu_x(z; \phi) = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix}.$$

Ponadto, dla uproszczenia, zakładamy, że

$$\mu_\sigma(z; \theta) = \begin{bmatrix} \alpha \\ \vdots \\ \alpha \end{bmatrix} \stackrel{\text{def}}{=} \hat{\alpha}$$

dla wszystkich z i niezależnie od parametrów θ (tzn. przyjęto stałą wariancję dla danych wyjściowych).

Ponieważ $x|z \sim \mathcal{N}(\mu_x(z; \theta), \alpha I_d)$, więc

$$-\log f_\theta(x|z) = \sum_{i=1}^d \left(\frac{1}{2} \log(2\pi) + \log \alpha + \frac{(x_i - \mu_i)^2}{2\alpha} \right). \quad (3.6)$$

Jednak metryka ta niezbyt dobrze nadaje się do chmur punktów, ponieważ jest zależna od kolejności występowania punktów na wejściu i wyjściu.

Zarówno dane wejściowe x , jak i wyjściowe $\mu_x(z; \theta)$ możemy potraktować jak chmury punktów w \mathbb{R}^3 , tj.

$$x = \begin{bmatrix} x_{1,1} & x_{i,1} & x_{d',1} \\ x_{1,2} & \dots & x_{i,2} & \dots & x_{d',2} \\ x_{1,3} & & x_{i,3} & & x_{d',3} \end{bmatrix}, \quad \mu_x(z; \phi) = \begin{bmatrix} \mu_{1,1} & \mu_{i,1} & \mu_{d',1} \\ \mu_{1,2} & \dots & \mu_{i,2} & \dots & \mu_{d',2} \\ \mu_{1,3} & & \mu_{i,3} & & \mu_{d',3} \end{bmatrix},$$

gdzie $d' = \frac{d}{3}$. Dodatkowo niech

$$x_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,3} \end{bmatrix}, \quad \mu_i = \begin{bmatrix} \mu_{i,1} \\ \mu_{i,2} \\ \mu_{i,3} \end{bmatrix}.$$

Wtedy równanie (3.6) można zapisać jako

$$-\log f_\theta(x|z) = \sum_{i=1}^{d'} \left(\frac{3}{2} \log(2\pi) + 3 \log \alpha + \frac{\|x_i - \mu_i\|_2^2}{2\alpha} \right). \quad (3.7)$$

Aby uzyskać niewrażliwość funkcji kosztu na permutacje punktów wejściowych i wyjściowych za koszt rekonstrukcji przyjmiemy

$$\begin{aligned} -\mathcal{L}'_{rec}(x, \theta, \phi) &= \frac{1}{2\alpha} \left(\sum_{i=1}^{d'} \min_{j=1}^{d'} \|x_i - \mu_j\|_2^2 + \sum_{i=1}^{d'} \min_{j=1}^{d'} \|\mu_i - x_j\|_2^2 \right) \\ &= \frac{1}{2\alpha} CD(S_1, S_2), \end{aligned} \quad (3.8)$$

gdzie $S_1 = \{x_i \mid 1 \leq i \leq d'\}$ i $S_2 = \{\mu_i \mid 1 \leq i \leq d'\}$, a CD dane jest wzorem (2.1).

Możemy zauważyć, że wzór (3.8) odpowiada wzorowi (3.7), jeżeli dla uproszczenia pominiemy stałe wyrazy (które nie wpływają na gradient) i założymy, że każdy punkt oryginalnej chmury generowany jest przez najbliższy odpowiednik w chmurze zdekodowanej. Wymagamy również, żeby chmura oryginalna generowała z wysokim prawdopodobieństwem chmurę zdekodowaną, co zapobiega generowaniu przez dekoder nieprzydatnych punktów.

Ostatecznie zatem będziemy minimalizować funkcję

$$-\hat{\mathcal{L}}'(\mathcal{X}, \theta, \phi) = \sum_{x \in \mathcal{X}} -\mathcal{L}'_{rec}(x, \theta, \phi) + KL(g_\phi(z|x) \parallel f(z)). \quad (3.9)$$

3.2. Autoenkoder wariacyjny z mieszanką gaussowską

Możemy przypuszczać, że obserwowane przez nas dane reprezentują kilka naturalnych podkategorii (klastrów), generowanych przez różne komponenty pewnej mieszanki rozkładów. W tym rozdziale został opisany generatywny model bazujący na VAE, który dopasowuje się do danych podzielonych na podkategorie (na podstawie [12] i [14]).

Tak jak w poprzedniej części zakładamy, że posiadane przez nas dane zostały wygenerowane przez pewien proces losowy. Tym razem zakładamy jednak, że w procesie występuje dodatkowa zmienna dyskretna, która wpływa na rozkład zmiennej ukrytej. Ścisłe, zakładamy, że

$$f_\theta(y, z, x) = f(y)f(z|y)f_\theta(x|z). \quad (3.10)$$

Ponadto

$$\begin{aligned} y &\sim \text{Cat}(1/C), \\ z|y &\sim \mathcal{N}(\mu_y, I_k), \\ x|z &\sim \mathcal{N}(\mu_x(z; \theta), \alpha I_d), \end{aligned} \quad (3.11)$$

gdzie $\mu_y \in \mathbb{R}^k$ (dla $1 \leq y \leq C$) są średnimi poszczególnych komponentów mieszanki gaussowskiej, a μ_x ponownie jest obliczeniem reprezentowanym przez sieć neuronową parametryzowaną przez θ .

3.2.1. ELBO

Tak jak w paragrafie 3.1.1., chcemy odtworzyć parametry θ oraz rozkład wyznaczający reprezentację $f_\theta(y, z|x)$. Z uwagi na dodatkową zmienną dyskretną, która wpływa na rozkład z musimy wprowadzić nowe oszacowanie na prawdopodobieństwo wygenerowania zaobserwowanych danych.

Ponownie wprowadzamy przybliżenie $g_\phi(y, z|x)$ rozkładu $f_\theta(y, z|x)$ i ponadto zakładamy, że jest ono postaci $g_\phi(y, z|x) = g_\phi(y|x)g_\phi(z|y, x)$. Niech $g_\phi(y|x)$ będzie rozkładem dyskretnym o prawdopodobieństwach $\pi(x; \phi)$, a $g_\phi(z|y, x)$ będzie rozkładem normalnym o średniej $\rho_x(y, x; \phi)$ i macierzy kowariancji $\rho_\sigma(y, x; \phi)I_d$. Obliczenia π, ρ_x, ρ_σ są reprezentowane przez sieci neuronowe parametryzowane przez ϕ .

Dla ustalonego x mamy

$$\begin{aligned} KL(g_\phi(y, z|x) || p_\theta(y, z|x)) &= \mathbb{E}_{y, z \sim g_\phi(y, z|x)} \left[-\log \frac{f_\theta(y, z|x)}{g_\phi(y, z|x)} \right] \\ &= \mathbb{E}_{y, z \sim g_\phi(y, z|x)} \left[-\log \frac{f_\theta(y, z|x)f_\theta(x)}{g_\phi(y, z|x)f_\theta(x)} \right] \\ &= \mathbb{E}_{y, z \sim g_\phi(y, z|x)} \left[-\log \frac{f_\theta(y, z, x)}{g_\phi(y, z|x)} \right] + \log f_\theta(x). \end{aligned} \quad (3.12)$$

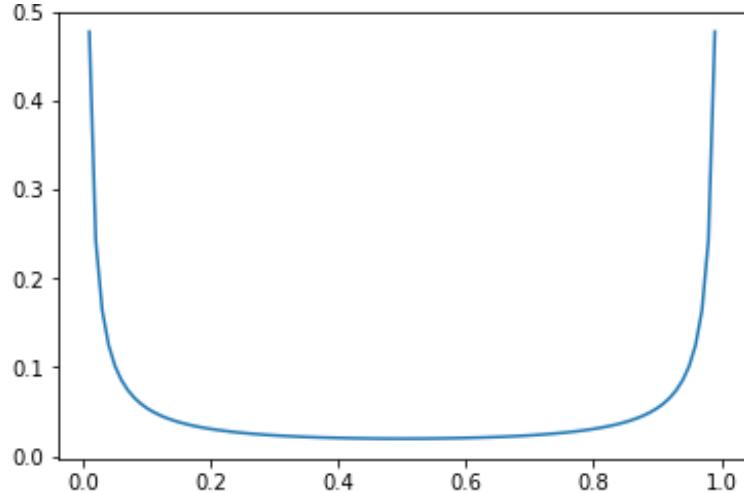
Korzystając z nieujemności $KL(\cdot || \cdot)$, otrzymujemy

$$\begin{aligned} \log f_\theta(x) &\geq \mathbb{E}_{y, z \sim g_\phi(y, z|x)} \left[\log \frac{f(y)}{g_\phi(y|x)} + \log \frac{f(z|y)}{g_\phi(z|x, y)} + \log f_\theta(x|z) \right] \\ &= \sum_{y=1}^C g_\phi(y|x) \mathbb{E}_{z \sim g_\phi(z|y, x)} \left[\log \frac{f(y)}{g_\phi(y|x)} + \log \frac{f(z|y)}{g_\phi(z|y, x)} + \log f_\theta(x|z) \right] \\ &= \sum_{y=1}^C g_\phi(y|x) \underbrace{\left(\mathbb{E}_{z \sim g_\phi(z|y, x)} [\log f_\theta(x|z)] - KL(g_\phi(z|y, x) || f(z|y)) \right)}_{\mathcal{L}(x, y, \theta, \phi)} \\ &\quad + \mathcal{H}(g_\phi(y|x)) - \log C, \end{aligned} \quad (3.13)$$

gdzie $\mathcal{H}(\cdot)$ jest entropią rozkładu dyskretnego.

Otrzymaliśmy nowe dolne ograniczenie na prawdopodobieństwo zaobserwowania danych dla dowolnych parametrów θ, ϕ (ELBO). Ponadto możemy zauważyc, że funkcja $\mathcal{L}(x, y, \theta, \phi)$ posiada tą samą budowę, co funkcja kosztu wprowadzona w paragrafie 3.1.2. (przy znany y wszystkie występujące w niej rozkłady są normalne). Oprócz tego w otrzymanym ograniczeniu występuje entropia rozkładu dyskretnego, którą możemy łatwo wyliczać i różniczkować.

Możemy zatem określić zadanie optymalizacyjne analogicznie do paragrafu 3.1.2. jako zmaksymalizowanie sumy ELBO po wszystkich próbkach (równoważnie zminimalizowanie -ELBO). Nowe ograniczenie będziemy obliczać iterując po wszystkich wartościach y i wyliczając (oraz różniczkując) $\mathcal{L}(x, y, \theta, \phi)$ tak, jak w podrozdziale 3.1..



Rysunek 3.1: Gęstość rozkładu Beta(0.01, 0.01). Istotne dla modelu jest silne skupienie gęstości przy brzegach nośnika.

3.3. Autoenkoder wariacyjny z rozkładem Beta

Podobnie jak w podrozdziale 3.1. założymy, że posiadane przez nas dane są obserwacjami pewnej zmiennej losowej. Tym razem postawmy jednak założenie, że z jest k -wymiarową zmienną, której składowe z_i są niezależne oraz

$$z_i \sim \text{Beta}(0.01, 0.01)$$

dla $1 \leq i \leq k$, podczas gdy rozkład $x|z$ jest taki jak w (3.2).

Rozkład Beta(0.01, 0.01) (zob. rys. 3.1) jest bardzo silnie skupiony przy brzegach nośnika, tj. 0 i 1. Przez regularyzowanie autoenkodera takim rozkładem chcemy, aby otrzymywane zmienne ukryte były również bardzo blisko 0 i 1, dzięki czemu po ich zaokrągleniu (zbinaryzowaniu) i przekazaniu do dekodera błąd rekonstrukcji nie wzrośnie znacząco.

Analogicznie do paragrafu 3.1.1. wprowadzamy przybliżenie $g_\phi(z|x)$ dla nieznanego rozkładu $f_\theta(z|x)$. W tym przypadku niech $g_\phi(z|x)$ będzie iloczynem funkcji $g_\phi(z|x)_i$, z których każda jest gęstością rozkładu Beta(α_i, β_i), gdzie

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_k \end{bmatrix} = \rho_\alpha(x; \phi), \quad \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} = \rho_\beta(x; \phi).$$

Tutaj ρ_α, ρ_β są ponownie reprezentowane przez sieci neuronowe parametryzowane przez ϕ . Innymi słowy, $g_\phi(z|x)$ jest rozkładem łącznym niezależnych zmiennych o rozkładach Beta z różnymi parametrami.

Przy tych założeniach wszystkie obliczenia wykonywane w paragrafie 3.1.1. są dalej poprawne i możemy wykorzystać ELBO (zob. 3.3) do określenia podobnego

do wprowadzonego w paragrafie 3.1.2. zadania optymalizacyjnego i zadania funkcji kosztu wzorem (3.4).

3.3.1. Koszt KL

Odległość KL dwóch jednowymiarowych rozkładów Beta o następujących parametrach

$$\begin{aligned}\beta_0 &\sim \text{Beta}(\alpha_0, \beta_0), \\ \beta_1 &\sim \text{Beta}(\alpha_1, \beta_1)\end{aligned}$$

dla pewnych $\alpha_0, \alpha_1, \beta_0, \beta_1 \in \mathbb{R}_+$ wynosi

$$\begin{aligned}KL(\beta_0 || \beta_1) = \log \left(\frac{B(\alpha_1, \beta_1)}{B(\alpha_0, \beta_0)} \right) + (\alpha_0 - \alpha_1)\psi(\alpha_0) + \\ + (\beta_0 - \beta_1)\psi(\beta_0) + (\alpha_1 - \alpha_0 + \beta_1 - \beta_0)\psi(\alpha_0 + \beta_0),\end{aligned}\tag{3.14}$$

gdzie $B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ to funkcja beta, a $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ to funkcja digamma. Procedury wyliczające obie funkcje (beta i digamma) efektywnie i stabilnie numerycznie są szeroko dostępne w bibliotekach (m.in. używanej w tym projekcie bibliotece PyTorch).

Ponadto odległość KL jest addytywna dla zmiennych niezależnych, zatem

$$KL(g_\phi(z|x) || f(z)) = \sum_{i=1}^k KL(\text{Beta}(\rho_\alpha(x; \phi)_i, \rho_\beta(x; \phi)_i) || \text{Beta}(0.01, 0.01)).$$

Składniki tej sumy możemy wyliczać i różniczkować na podstawie równania (3.14).

3.3.2. Koszt rekonstrukcji

W paragrafie 3.1.4. wprowadzamy szacowanie kosztu rekonstrukcji metodą Monte Carlo

$$\mathbb{E}_{z \sim g_\phi(z|x)} [-\log f_\theta(x|z)] \approx \frac{1}{m} \sum_{i=1}^m -\log f_\theta(x|z_i),$$

gdzie z_i są obserwacjami rozkładu $g_\phi(z|x)$. Aby wytrenować enkoder potrzebujemy wyliczyć

$$\nabla_\phi \log f_\theta(x|z_i),$$

jednak nie możemy tego zrobić wprost, gdy zmienne z_i zależą od parametrów ϕ po przez operację próbkowania. Byłoby to możliwe, gdybyśmy przedstawili zmienne z_i jako obliczalną i różniczkowalną po parametrach ϕ transformację zmiennej wylosowanej z pewnego rozkładu standardowego, niezależnego od parametrów (zachowując odpowiedni rozkład z_i). Podejścia tego użyliśmy już w paragrafie 3.1.4. reparametryzując z o rozkładzie $\mathcal{N}(\mu, \sigma)$ jako $z = \mu + \sigma\epsilon$, gdzie $\epsilon \sim \mathcal{N}(0, 1)$.

Podobny trik można zastosować do innych rozkładów z rodziny położenia i skali, niestety jednak nie działa on z rozkładami takimi jak Gamma i Beta, którego używamy w tym modelu. Autorzy [11] rozwiązują ten problem przy użyciu autoenkodera przeciwnego. Poniżej opisano sposób, w jaki można to osiągnąć dla autoenkodera wariacyjnego.

Bardziej ogólnym kandydatem na reparametryzację zmiennej z jest $z = G_\phi^{-1}(u)$, gdzie G_ϕ^{-1} to odwrotność dystrybuanty rozkładu $g_\phi(z|x)$, a $u \sim \mathcal{U}(0, 1)$. Dla rozkładów Gamma i Beta to podejście również nie jest wystarczające, gdyż ich odwrotna dystrybuanta nie daje się przedstawić zwartym wzorem.

Można rozważyć dwa rozwiązania powyższego problemu. Pierwsze to przybliżenie odwrotności dystrybuanty funkcją obliczalną i różniczkowalną. Drugie podejście korzysta z faktu, że

$$u = G_\phi(z).$$

Obie strony równości różniczkujemy po ϕ i korzystamy z tego, że u nie zależy od parametrów ϕ . Otrzymujemy

$$0 = \frac{dz}{d\phi} g_\phi(z|x) + \frac{\partial G_\phi(z)}{\partial \phi}.$$

Zatem

$$\frac{dz}{d\phi} = -\frac{\frac{\partial G_\phi(z)}{\partial \phi}}{g_\phi(z|x)}. \quad (3.15)$$

Równanie 3.15 pozwala wyliczać pochodne wylosowanej zmiennej z po parametrach rozkładu, jeżeli mamy możliwość wyliczania lub przybliżania pochodnych dystrubuanty G po parametrach rozkładu.

Dla rozkładów Gamma i Beta oba rozwiązania wymagają przybliżania funkcji jej różniczkowalną wersją. Drugie podejście posiada jednak tę zaletę, że nie wpływa na proces losowania zmiennych z_i , dzięki czemu możemy do tego użyć dokładniejszej procedury i zapewnić odpowiedni rozkład z_i . Pierwsze podejście tymczasem wprowadza przybliżenie już na etapie wyciągania próbki, co oznacza, że zmienne z_i mają rozkład $g_\phi(z|x)$ tylko w przybliżeniu.

Najprostszym, ale niezbyt dokładnym, sposobem przybliżenia pochodnej dystrubuanty rozkładu Gamma lub Beta jest policzenie ilorazów różnicowych. Wymaga to jedynie procedur wyliczających dystrybuanty tych rozkładów, które są dostępne w bibliotekach. Autorzy [16] proponują znacznie dokładniejsze (i bardziej skomplikowane) przybliżenie, które również zostało zaimplementowane w bibliotece PyTorch [18].

Uwaga. W powyższym opisie skupiliśmy się na rozkładach Beta oraz Gamma, mimo że w dalszej części wykorzystujemy jedynie rozkład Beta. Rozkład Gamma pojawia się dlatego, że gdybyśmy rozwiązali dla niego rozważany problem, to korzystając z faktu, że $Y = \frac{X_1}{X_1+X_2}$ ma rozkład Beta(α, β), gdy $X_1 \sim \text{Gamma}(\alpha, \theta)$, $X_2 \sim \text{Gamma}(\beta, \theta)$, otrzymujemy rozwiązanie dla rozkładu Beta (ta transformacja jest

różniczkowalna). Autorzy [16] zaznaczają jednak, że ten sposób zwiększa wariancję metody i proponują lepsze rozwiążanie. Mimo tego w implementacji użyto powyższego triku.

Mając do dyspozycji wylosowane zmienne z_i , dla których możemy propagować gradient wstecz, używamy estymatora Monte Carlo i kosztu rekonstrukcji jak w równaniu (3.8).

Rozdział 4.

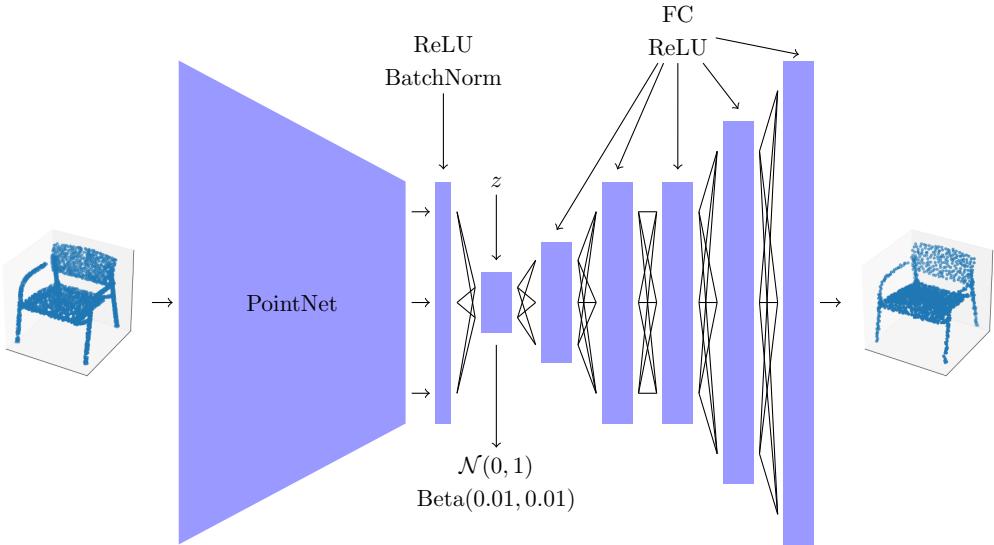
Implementacja i architektura modeli

Metody opisane w rozdziale 3. zawierają abstrakcyjne obliczenia ρ, π (nazywane dalej **enkoderem**) oraz μ (nazywane dalej **dekoderem**) realizowane przez sieci neuronowe. W tym rozdziale przedstawiono konkretną architekturę nadaną tym obliczeniom.

Enkoder (dla wszystkich rozkładów) wykorzystuje opisaną w podrozdziale 2.1. i wprowadzoną w [3] architekturę PointNet do ekstrakcji 1024 sparametryzowanych (wyuczalnych) cech z chmur wejściowych. Dla rozkładów normalnego i Beta, te cechy są dalej traktowane jako wejście do jednej warstwy pełnej o 1024 neuronach wejściowych i 256 neuronach wyjściowych. Pomiędzy siecią PointNet a warstwą pełną znajduje się aktywacja ReLU i warstwa *batch normalization* [19]. Ostatnia warstwa reprezentuje parametry rozkładu zmiennej ukrytej – po 128 liczb dla ρ_x, ρ_σ w przypadku rozkładu normalnego lub po 128 liczb dla ρ_α, ρ_β w przypadku rozkładu Beta.

Dla modelu z mieszką gausowską cechy wyliczone przez sieć PointNet służą jako wejście do jednej warstwy pełnej o C (liczba żądanego klastrów) neuronach wyjściowych, na których wyliczana jest funkcja softmax. Otrzymane wartości to $\pi(x; \phi)$ – parametry rozkładu $g_\phi(y|x)$. Następnie dla każdego $1 \leq y \leq C$, jego kodowanie one-hot doklejane jest do cech wyjściowych z sieci PointNet i traktowane jako wejście do jednej warstwy pełnej o 256 neuronach wyjściowych. Wyliczone wartości to ρ_x, ρ_σ – parametry rozkładu $g_\phi(z|y, x)$. Oznacza to, że wymiar zmiennej ukrytej wynosi 128 (dla wszystkich modeli).

Dekoder posiada znacznie prostszą architekturę, wspólną dla wszystkich modeli. Jest to sieć MLP (*multilayer perceptron*) o 4 ukrytych warstwach (kolejno 512, 1024, 1024 i 2048 neuronów) z aktywacjami ReLU (poza warstwą wyjściową). Ostatnia warstwa składa się z 3×2048 neuronów, co odpowiada wymiarowi wejściowym chmur punktów i reprezentuje parametry μ_x .



Rysunek 4.1: Architektura modelu VAE.

Pomiędzy enkoderem i dekoderem następuje losowanie zmiennej ukrytej z z rozkładu wyznaczonego przez parametry zwrócone przez enkoder. Losowanie wykonywane jest zgodnie z metodami opisanymi w paragrafach 3.1.4. i 3.3.2., aby umożliwić propagowanie gradientu do parametrów enkodera. Aby zmniejszyć wariancję szacowania Monte Carlo, wykonywanego w celu obliczenia kosztu rekonstrukcji, proces losowania i dekodowania można powtórzyć wielokrotnie i obliczyć średni koszt. Wykonane eksperymenty wskazują jednak, że pojedyncze losowanie jest wystarczające.

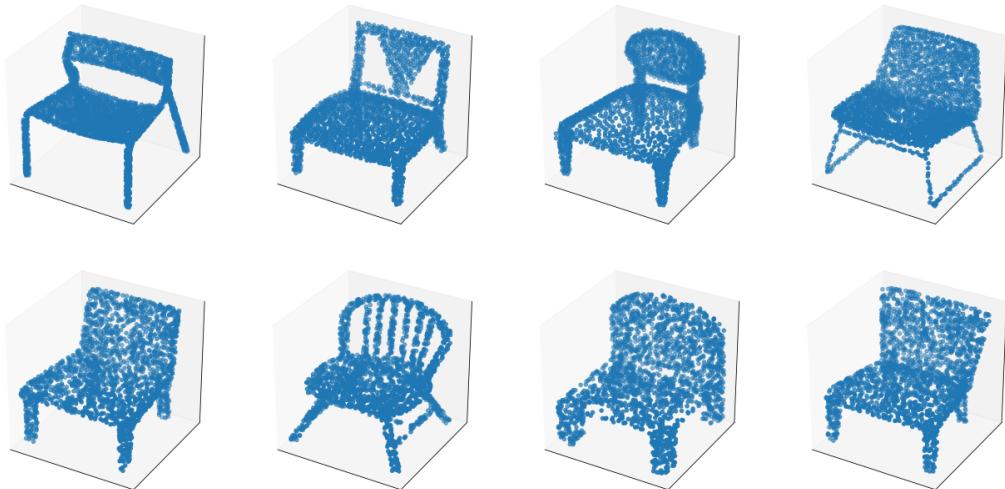
Parametr α pojawiający się w koszcie rekonstrukcji (zob. 3.8) został doświadczalnie ustalony na 5×10^{-4} (dla obu rozkładów).

4.1. Zbiór danych i trenowanie

Model został wytrenowany na jednej klasie obiektów wybranej z połączonych zbiorów ModelNet40 [2] oraz ShapeNet [1] po uprzednim przekształceniu obu zbiorów do chmur punktów wymiaru 2048×3 zgodnie z metodą opisaną w [7]. Do niniejszego opracowania wybrano klasę krzesel. Do przeprowadzenia eksperymentów zbiór podzielono na części treningowe i testowe w proporcjach 80%-20%.

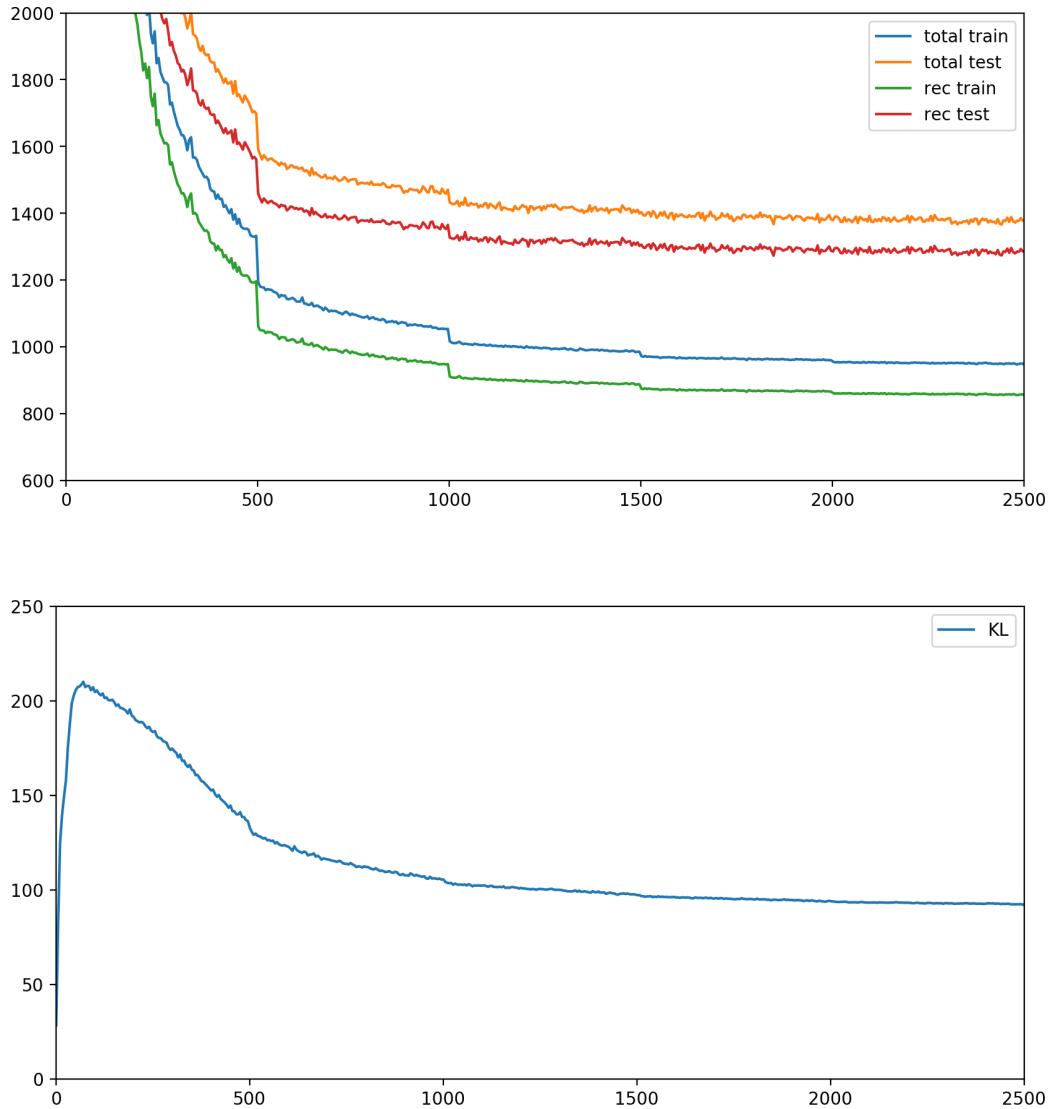
Optymalizację funkcji kosztu danej wzorem (3.4) przeprowadzono z użyciem metody ADAM [20]. Parametry metody wybrano doświadczalnie: początkowe *learning rate* ustalone na 2×10^{-4} , parametr zmniejszał się dwukrotnie co każde 500 epok. Model trenowany był przez 2500 epok (duża liczba epok wynika z niewielkiej ilości próbek).

Na rys. 4.3 przedstawiona została zmiana całkowitej funkcji kosztu, kosztu rekonstrukcji oraz kosztu KL w czasie przy trenowaniu VAE z rozkładem normalnym.



Rysunek 4.2: Przykładowe chmury punktów ze zbioru danych. Górný wiersz pochodzi ze zbioru ModelNet40, dolny ze zbioru ShapeNet.

Możemy zaobserwować charakterystyczną dla autoenkoderów wariancyjnych dynamikę kosztu KL – podczas gdy koszt rekonstrukcji stale maleje, koszt KL na początku szybko wzrasta i powoli spada z biegiem kolejnych epok. Początkowy wzrost kosztu KL odpowiada umieszczaniu przez model wielu informacji w zmiennej ukrytej, żeby uzyskać drastyczny spadek ogromnego kosztu rekonstrukcji. Jednak gdy koszt rekonstrukcji spada, model zaczyna optymalizować informacje przechowywane w zmiennej ukrytej tak, żeby jej rozkład był podobny do $\mathcal{N}(0, I)$, czemu odpowiada powolny spadek kosztu KL.



Rysunek 4.3: Wykresy zmiany funkcji kosztu (równanie (3.4)) wraz z poszczególnymi częściami na zbiorze treningowym oraz testowym podczas uczenia VAE z rozkładem normalnym. Koszt KL jest jednakowy na zbiorze treningowym i testowym.

Rozdział 5.

Wyniki eksperymentalne

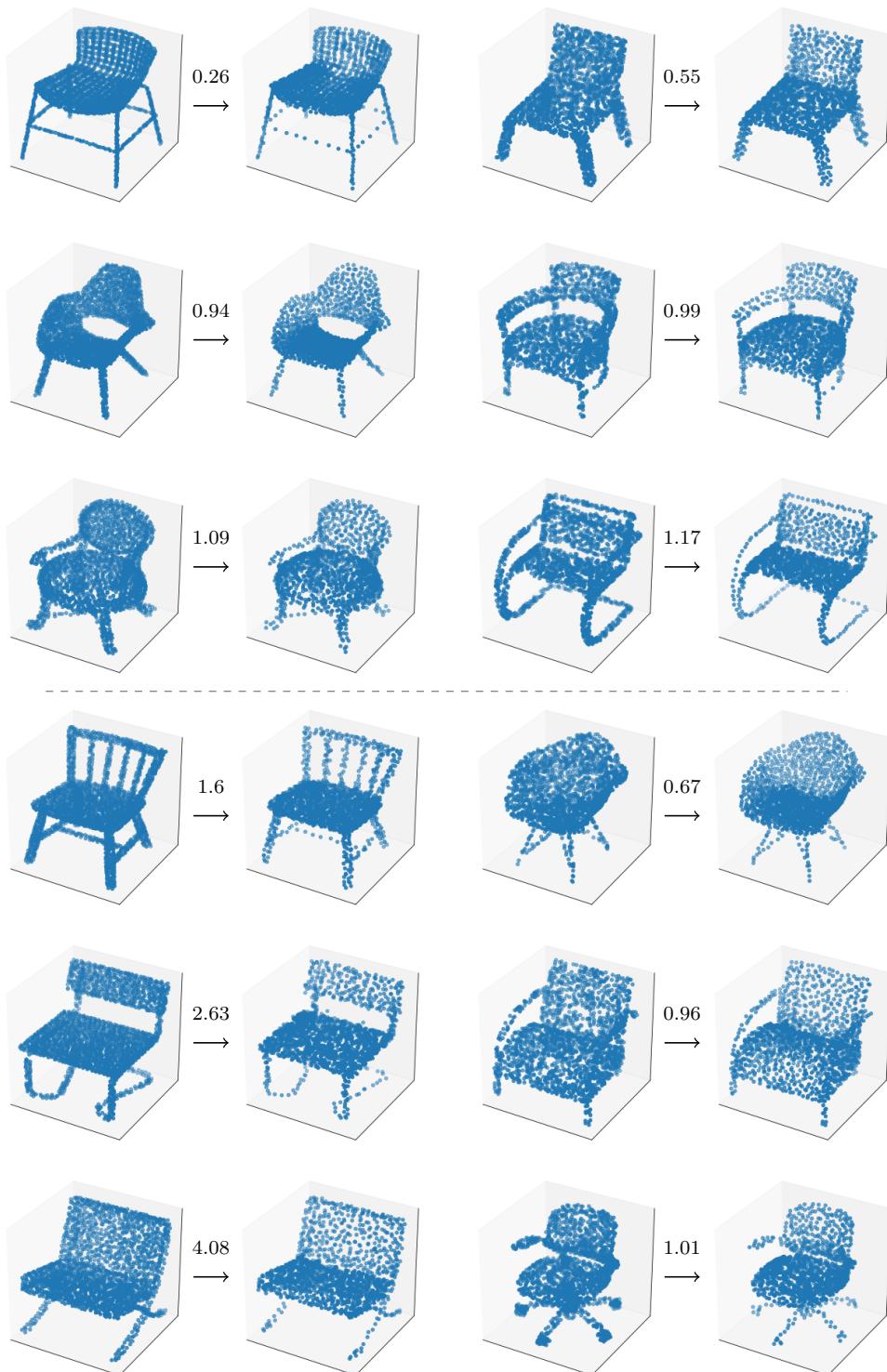
Największym atutem autoenkoderów wariancyjnych jest połączenie zdolności rekonstrukcji (typowych dla autoenkoderów) oraz możliwości generatywnych (charakterystycznych np. dla mieszanek gaussowskich) w jedną spójną całość. Naturalnie zatem sprawdzono możliwości modelu w obu wymienionych powyżej kategoriach oraz w zastosowaniach, które wymagają połączenia tych zdolności.

5.1. Zdolność rekonstrukcji

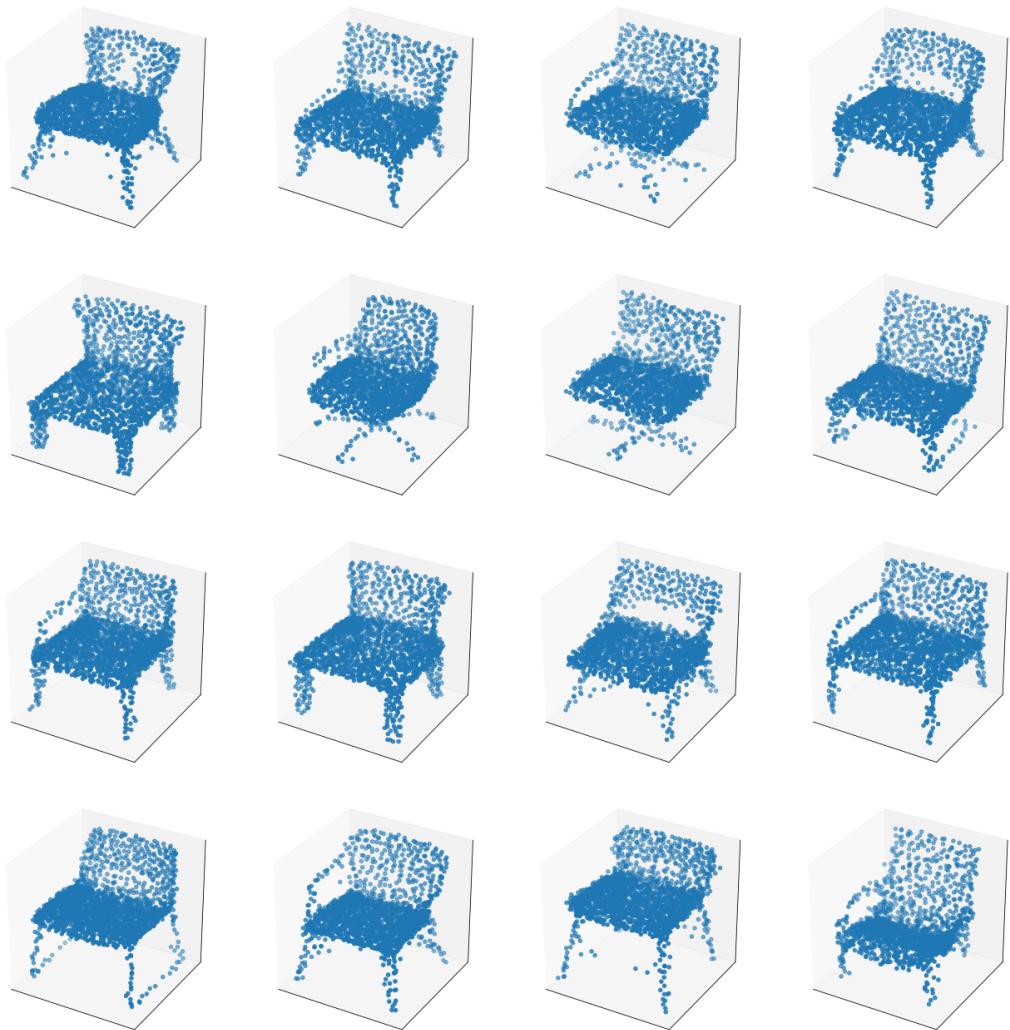
Model wytrenowany zgodnie z opisem w podrozdziale 4.1. posiada dobre zdolności rekonstrukcji, na poziomie zwykłego autoenkodera (bez randomizacji zmiennej ukrytej; nazywanego dalej AE). Na rys. 5.1 znajdują się oryginalne chmury punktów ze zbioru danych (treningowego i testowego) wraz z chmurami zrekonstruowanymi przez dekoder na podstawie 128-wymiarowej zmiennej ukrytej wylosowanej z rozkładu z parametrami zwróconymi przez enkoder. Przedstawione rekonstrukcje ukazują przekrój możliwości modelu, tzn. przykłady poniżej i powyżej średniej jakości, która mierzona *Chamfer distance* wynosi 0.85 na zbiorze treningowym i 1.28 na zbiorze testowym. Dla porównania autoenkoder (AE) osiąga prawie takie same wyniki, $CD = 0.85$ na zbiorze treningowym i $CD = 1.24$ na zbiorze testowym. Rekonstrukcje zostały wykonane przez jeden model wytrenowany na połączonym zbiorze danych (ModelNet40 oraz ShapeNet). Możemy zaobserwować, że model gorzej radzi sobie z częścią testową zbioru ModelNet40, niż zbioru ShapeNet. Jest to spowodowane tym, że ShapeNet jest prawie czterokrotnie większy od ModelNetu40, dzięki czemu model lepiej się na nim generalizuje.

5.2. Możliwości generatywne

Dzięki randomizacji zmiennej ukrytej model powinien zyskać możliwości generowania syntetycznych, nieobecnych w zbiorze danych próbek, które jednocześnie



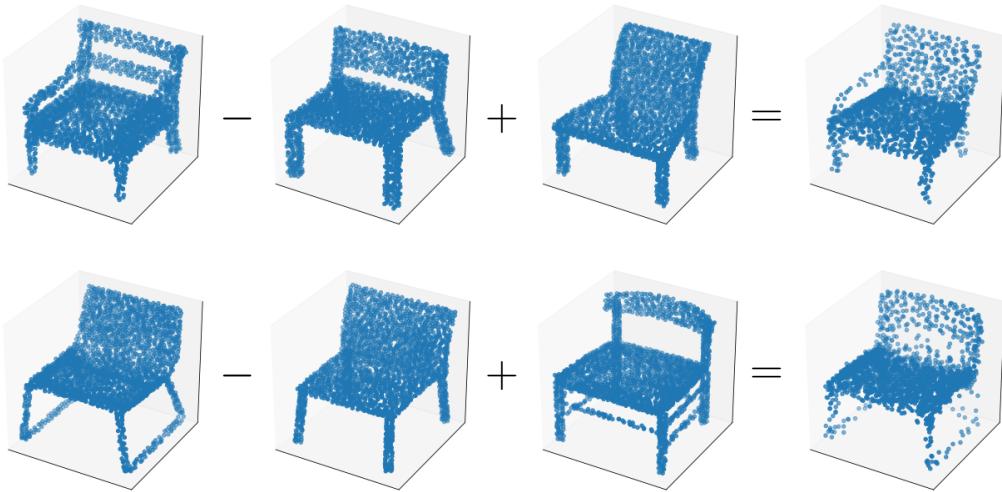
Rysunek 5.1: Oryginalne chmury i otrzymane rekonstrukcje dla zbiorów treningowego (na górze) i testowego (na dole). Chmury z lewej strony pochodzą ze zbioru ModelNet40, z prawej ze zbioru ShapeNet. Nad strzałkami podano odległość CD rekonstrukcji. Średnie CD wynosi 0.85 na zbiorze treningowym i 1.28 na testowym.



Rysunek 5.2: Syntetyczne chmury punktów wygenerowane przez model na podstawie wylosowanych zmiennych ukrytych.

wygląдают realistycznie. Zdolności te potwierdzają eksperymenty wykonane w tym podrozdziale.

Na rozważany model nałożono wymaganie, aby zmienne ukryte posiadały rozkład bliski do standardowego wielowymiarowego normalnego (o średniej w 0 i identycznościowej macierzy kowariancji). Dzięki temu możemy tworzyć syntetyczne próbki danych poprzez wylosowanie zmiennej ukrytej z docelowego rozkładu (standardowego normalnego) i przekazanie wyniku do dekodera. Rysunek 5.2 przedstawia chmury powstałe w opisany powyżej sposób. Możemy zauważyc, że powstałe próbki dobrze pasują do reszty zbioru danych, reprezentują różne podkategorie obiektów i posiadają wykształcone cechy charakterystyczne, które czynią je bardziej realistycznymi. Świadczy to o dobrze zorganizowanej i zregularyzowanej przestrzeni reprezentacji.



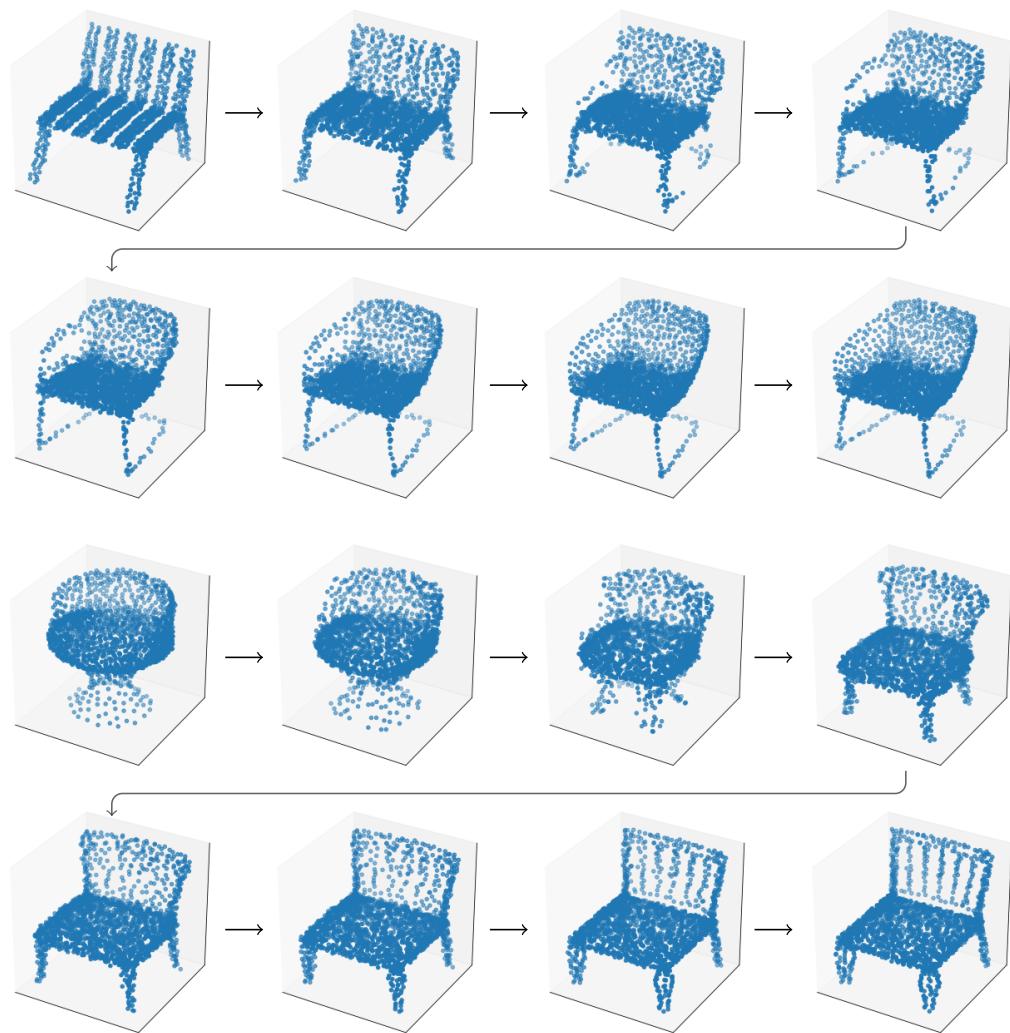
Rysunek 5.3: Wyniki wykonywania prostych operacji arytmetycznych na zmiennych ukrytych. W pierwszym wierszu warto zwrócić uwagę, że w różnicy zmiennych ukrytych została zapamiętana informacja o podłokietnikach, a w drugim wierszu kształt nóg.

5.3. Dodatkowe eksperymenty

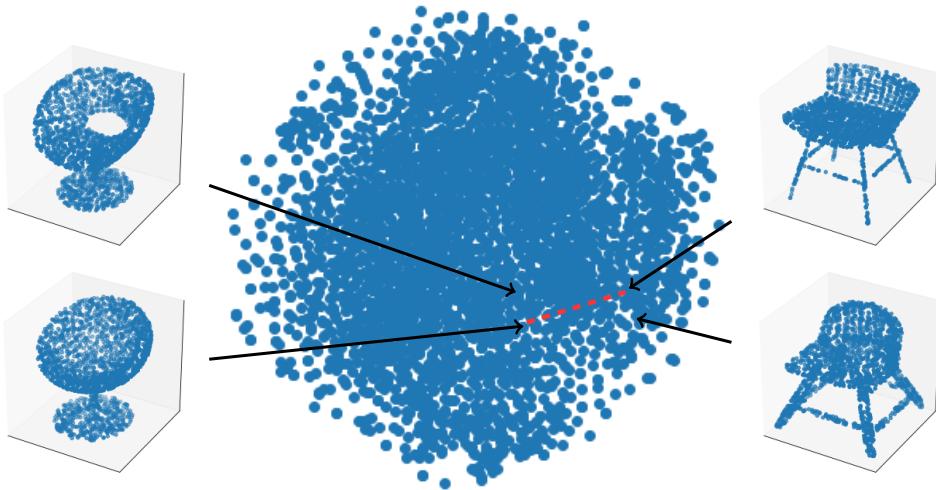
Silnym argumentem przemawiającym na korzyść znalezionej reprezentacji jest możliwość przeprowadzania na niej intuicyjnych operacji, które zakończone powinny być przewidywalnym wynikiem. Autorzy [21] zaznaczają, że dobrze skonstruowane i wyuczone modele znajdują reprezentacje, na których można wykonywać proste działania algebraiczne (dodawanie i odejmowanie) z przewidywalnym wynikiem. Dzięki gęstej i regularnej przestrzeni reprezentacji autoenkoder wariancyjny świetnie radzi sobie z takimi zadaniami.

Jednym z eksperymentów, które potwierdzają dobrą jakość reprezentacji znalezionej przez VAE, jest gładka i naturalna interpolacja pomiędzy dwoma obiektami ze zbioru danych. Zadanie polega na tym, aby wytworzyć obiekty reprezentujące stopniowe przekształcenie jednej próbki ze zbioru danych w drugą. Żądamy przy tym, aby obiekty przejściowe wyglądały naturalnie, podobnie do reszty zbioru danych. Rysunek 5.4 przedstawia interpolacje wykonane przez uzyskany model. Można zaobserwować, że kolejne kroki interpolacji coraz bardziej upodobniają obiekt źródłowy do docelowego, jednocześnie zachowując naturalny wygląd etapów pośrednich.

Co ciekawe, poszczególne części zmiennej ukrytej przechowują ten sam rodzaj informacji dla wszystkich podkategorii obiektów. Dzięki temu możliwe jest przechowywanie częściowej informacji o obiekcie. Informację taką można później wykorzystać do edycji obiektu źródłowego – możemy np. stworzyć reprezentację przechowującą informację o kształcie nóg krzesła i dodać ją do obiektu, zmieniając tym jego



Rysunek 5.4: Gładkie interpolacje pomiędzy obiekty ze zbioru danych. Przedstawione zostały dwa procesy interpolacji – każdy zajmuje dwa wiersze. Chmury oglądane wierszami od lewej do prawej reprezentują kolejne stopnie interpolacji. Pierwsze chmury w wierszach 1. i 3. oraz ostatnie w wierszach 2. i 4. to obiekty docelowe (ze zbioru danych), pozostałe są syntetyczne, wygenerowane przez model. W pierwszej interpolacji warto zwrócić uwagę na początkowe wypełnianie siedzenia i oparcia, dalej połączenie nóg i wytworzenie podłokietników i na koniec – wypełnienie podłokietników. W drugiej możemy zobaczyć stopniowe dzielenie się części jednolitej bryły początkowej na siedzenie, oparci i nogi, które dalej zyskują odpowiedni kształt i liczne szczegóły.

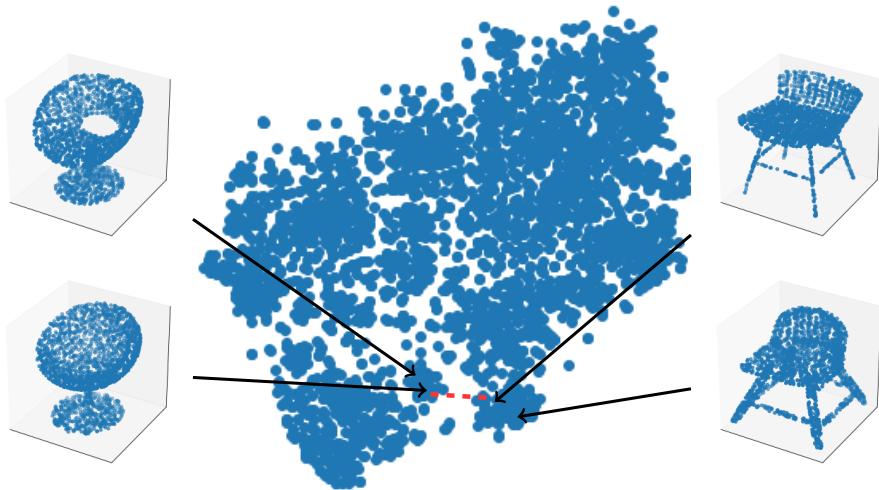


Rysunek 5.5: Wykres zmiennych ukrytych wylosowanych na podstawie parametrów wygenerowanych przez enkoder modelu VAE po zredukowaniu do dwóch wymiarów metodą t-SNE. Warto zwrócić uwagę na duże zagęszczenie przestrzeni i brak widocznych klastrów. Mimo, że osobnych klastrów nie widać na wykresie, to z położonych blisko zmiennych ukrytych model dekoduje bliskie znaczeniowo obiekty, co potwierdzają wybrane przykłady.

oryginalne nogi.

Możliwości te potwierdza następujący eksperyment. Założymy, że dysponujemy trzema obiektami ze zbioru danych – dwoma podobnymi, różniącymi się obecnością jednej cechy (np. podłokietników, lub połączonych nóg) i trzecim różnym od poprzednich, który również nie posiada wybranej cechy. Przypuszczamy, że w różnicy zmiennych ukrytych podobnych obiektów zostanie zakodowana obecność różniącej ich cechy. Aby dodać tę cechę do trzeciego obiektu możemy zatem do jego zmiennej ukrytej dodać otrzymaną różnicę i zdekodować wynik. Rezultaty tego eksperymentu przedstawia rys. 5.3.

Rysunek 5.5 i 5.6 przedstawiają topografię przestrzeni reprezentacji znalezionej odpowiednio przez modele VAE (rys. 5.5) i AE (rys. 5.6). Wszystkie obiekty ze zbioru treningowego zakodowano i dla VAE dodatkowo wylosowano jedną reprezentację. Otrzymane zmienne ukryte zredukowano do dwóch wymiarów metodą t-SNE. Możemy zauważyć, że przestrzeń VAE jest mocno zregularyzowana i upodobniona do obserwacji zmiennej zakładanego rozkładu (normalnego). Widoczne jest duże zagęszczenie wartości i brak wyraźnie odseparowanych klastrów. Mimo braku widocznych klastrów warto zwrócić uwagę na to, że reprezentacje podobnych semantycznie obiektów są położone blisko siebie. Przestrzeń AE natomiast nie jest regularna, posiada większe luki i lekko zarysowane, odseparowane klastry, co negatywnie wpływa na zdolności generatywne modelu.



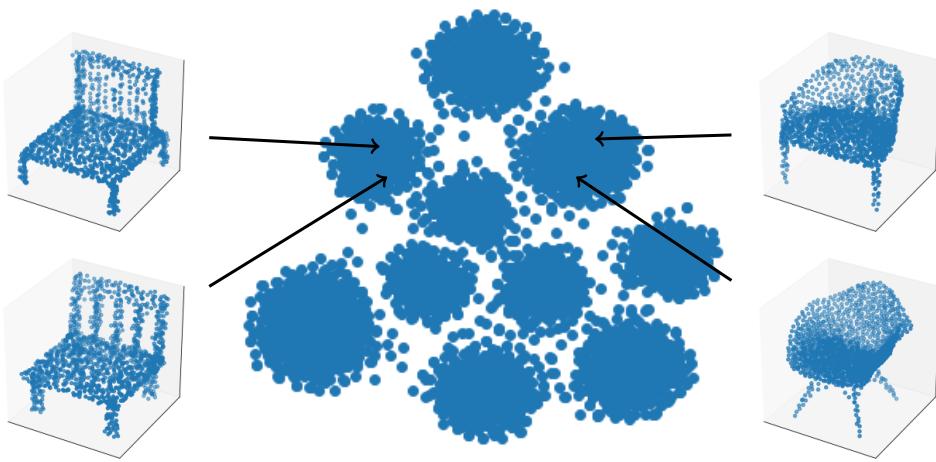
Rysunek 5.6: Wykres zmiennych ukrytych wygenerowanych przez enkoder modelu AE po zredukowaniu do dwóch wymiarów metodą t-SNE. Możemy zaobserwować znacznie mniejsze zagęszczenie przestrzeni niż w przypadku VAE i lekko widoczne odseparowane klastry z przestrzeniami pomiędzy nimi. Taki układ przestrzeni może skutkować gorszymi wynikami na zadaniach generatywnych, np. interpolacji lub generowaniu danych syntetycznych.

5.4. Klasteryzacja

Model opisany w podrozdziale 3.2. posiada dodatkową zmienną dyskretną, która określa prawdopodobieństwo przynależności próbki do każdej z określonych przez model podkategorii. W tym podrozdziale sprawdzono możliwości opisanego modelu do znajdowania wspólnych cech chmur punktów i wykorzystania ich do określenia podkategorii obiektów.

Model wytrenowano tak, aby podzielił dane na 10 klastrów. Następnie wybrano z nich 6 najciekawszych i z każdego wylosowano po 4 próbki spośród tych, dla których prawdopodobieństwo przynależności przekraczało 95%. Wyniki tego eksperymentu przedstawia rysunek 5.8. Możemy zaobserwować, że obiekty zostały podzielone zgodnie z ich cechami charakterystycznymi i widocznymi podobieństwami w budowie. Warto zwrócić uwagę, że klastry zostały zdefiniowane w sposób całkowicie nienadzorowany.

Rysunek 5.7 przedstawia topografię przestrzeni reprezentacji znalezionej przez model VAE z regularyzacją mieszaną gaussowską. Możemy zauważać wyraźnie odseparowane klastry zdefiniowane przez model. Każdy z klastrów jest wciąż podobny do obserwacji standardowej zmiennej normalnej. Z jednej strony taka regularyzacja umożliwia podzielenie obiektów na podkategorie, z drugiej jednak może negatywnie wpływać na niektóre możliwości generatywne, np. interpolację między obiektami z różnych klastrów.



Rysunek 5.7: Wykres zmiennych ukrytych wylosowanych na podstawie parametrów wygenerowanych przez enkoder modelu VAE z mieszanką gaussowską po zredukowaniu do dwóch wymiarów metodą t-SNE. Możemy zaobserwować wyraźnie odseparowane 10 klastrów, z których każdy jest zregularyzowany rozkładem gaussowskim. Obiekty dekodowane z każdego z klastrów wykazują podobne cechy i budowę, co potwierdzają wybrane przykłady.

5.5. Binaryzacja reprezentacji

Aby uzyskać binarną reprezentację chmur punktów wytrenowano autoenkoder wariacyjny z narzuconym rozkładem Beta(0.01, 0.01) dla zmiennej ukrytej, zgodnie z opisem w podrozdziale 3.3..

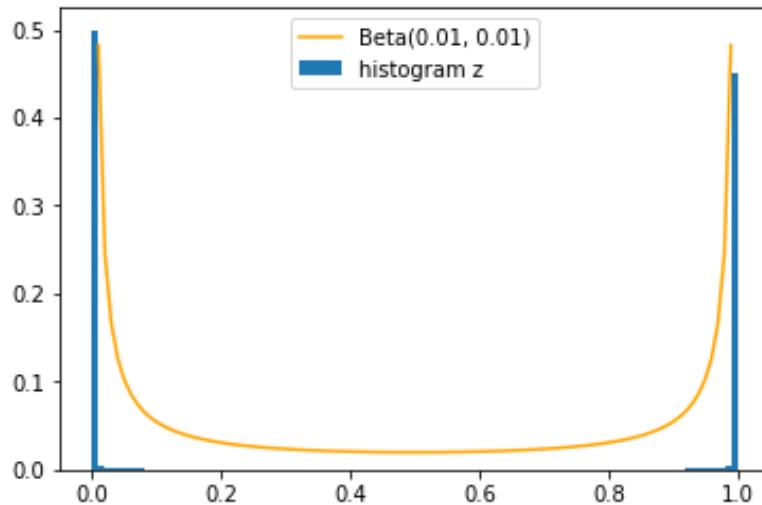
Rysunek 5.9 przedstawia histogram wartości wylosowanych z rozkładów zwróconych przez enkoder dla danych ze zbioru treningowego (po jednej próbce na obiekt, zagregowane do jednego wymiaru). Możemy zaobserwować, że prawie wszystkie wartości znajdują się bardzo blisko brzegów nośnika, tj. 0 i 1, co pozwala przypuszczać, że wzrost kosztu rekonstrukcji (po zaokrągleniu, zbinaryzowaniu) nie będzie znaczący.

Tabela 5.10 przedstawia dokładny koszt rekonstrukcji przed i po zbinaryzowaniu na zbiorach treningowym i testowym. Możemy zauważyć, że wzrost kosztu jest pomijalny i model posiada dobre zdolności rekonstrukcji obiektów na podstawie zaledwie 128 bitów. Koszt rekonstrukcji jest wyższy o niecałe 18% w porównaniu z reprezentacją ciągłą, przy 32-krotnym skompresowaniu zmiennej ukrytej (z 128 32-bitowych liczb do 128 bitów). Rysunek 5.11 przedstawia chmury oryginalne i ich odpowiedniki odtworzone przez dekoder na podstawie 128 bitów wylosowanych z rozkładu wygenerowanego przez enkoder.

Mimo przejścia do dyskretnej przestrzeni zmiennych ukrytych, model pozostał



Rysunek 5.8: Wybrane klastry wydzielone przez model VAE z mieszanką gaussowską. W każdym wierszu znajdują się przykłady wylosowane z jednego klastra (spośród tych, dla których prawdopodobieństwo przynależności przekraczało 95%).

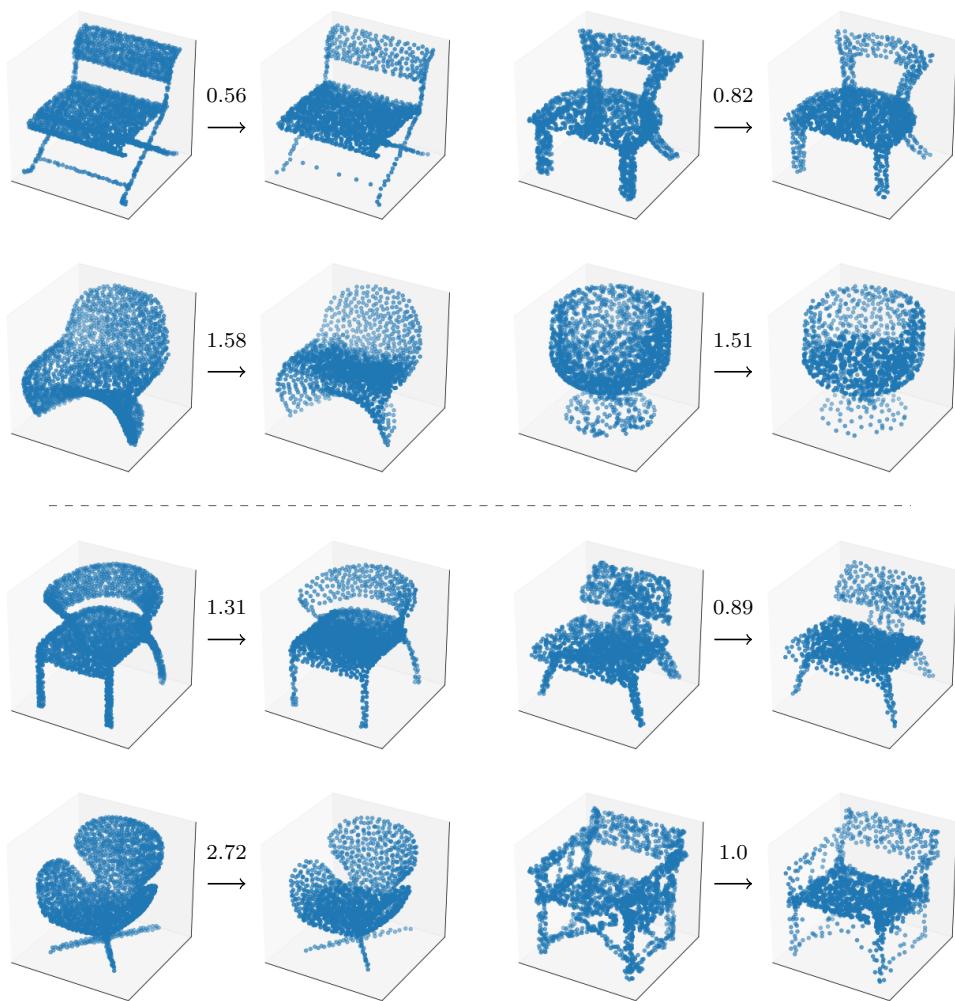


Rysunek 5.9: Histogram wartości zmiennej ukrytej dla danych ze zbioru treningowego przy regularyzacji rozkładem Beta($0.01, 0.01$).

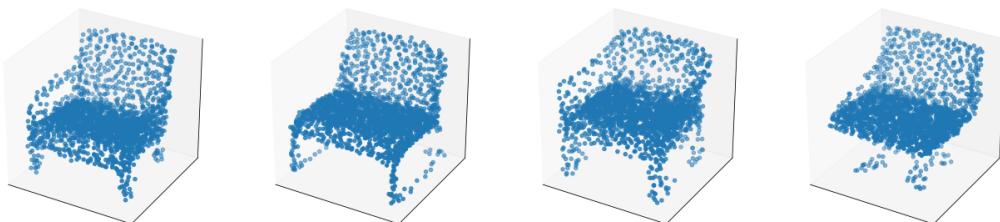
	train	test
AE	0.853	1.247
VAE- \mathcal{N}	0.851	1.287
VAE- β	1.024	1.464
VAE-bin	1.027	1.464

Rysunek 5.10: Średnia odległość *Chamfer distance* pomiędzy oryginalnymi chmurami a ich rekonstrukcjami zwróconymi przed odpowiednie modele na zbiorach treningowym i testowym.

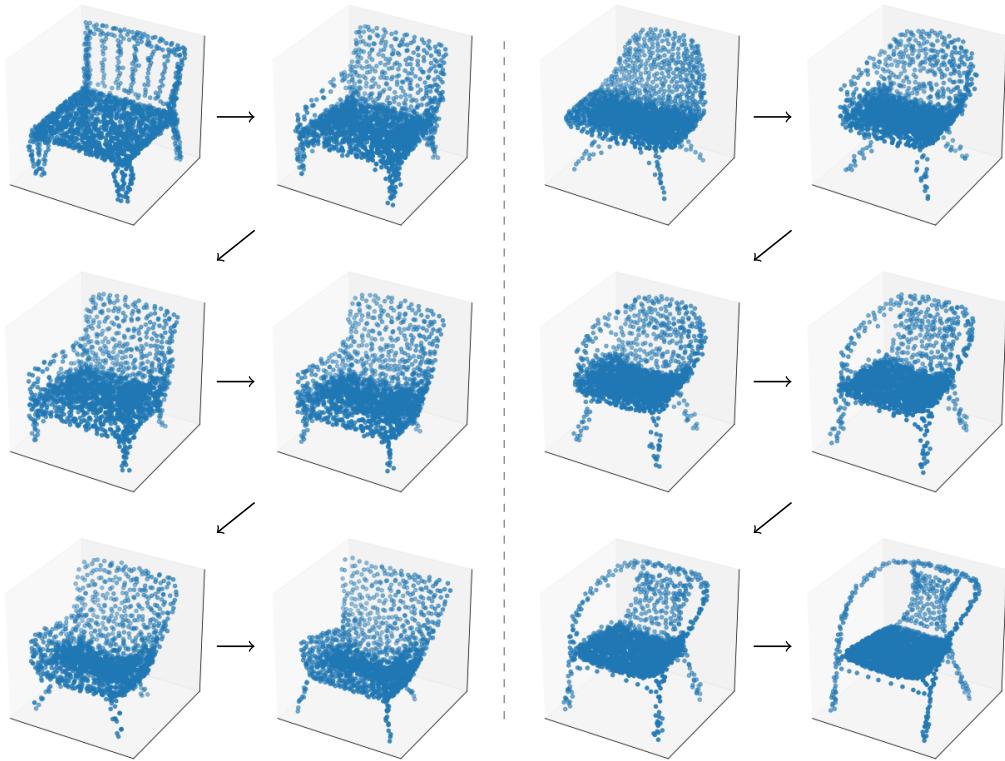
generatywny. Oznacza to, że możemy tworzyć dane syntetyczne na podstawie 128 wylosowanych bitów oraz przeprowadzać interpolacje między obiekty i edytować obiekty za pomocą intuicyjnych operacji. Rysunki 5.12, 5.13 oraz 5.14 prezentują te możliwości.



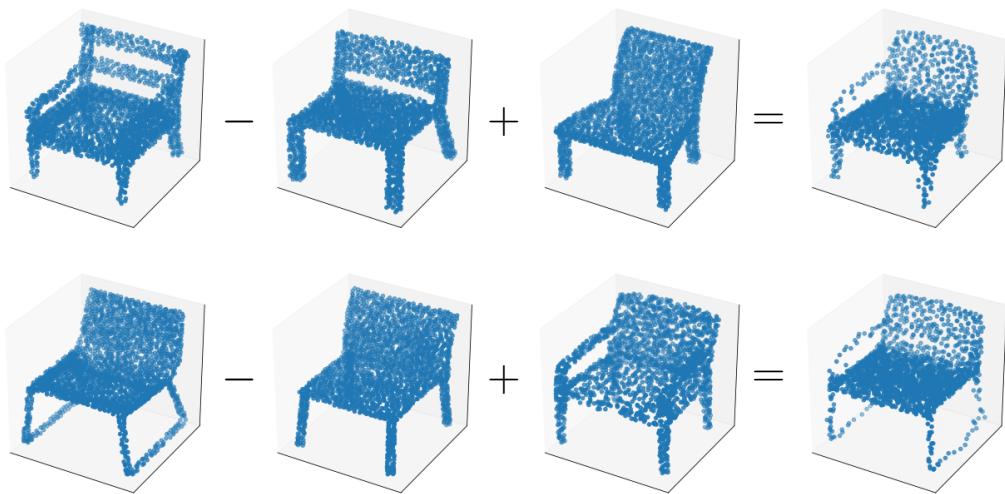
Rysunek 5.11: Oryginalne chmury i rekonstrukcje otrzymane na podstawie 128 bitów wylosowanych z rozkładu wygenerowanego przez enkoder dla zbiorów treningowego (na górze) i testowego (na dole). Chmury z lewej strony pochodzą ze zbioru ModelNet40, z prawej ze zbioru ShapeNet. Nad strzałkami podano odległość CD rekonstrukcji. Średnie CD wynosi 1.02 na zbiorze treningowym i 1.46 na testowym.



Rysunek 5.12: Syntetyczne chmury punktów powstałe ze zdekodowania 128-bitowych zmiennych ukrytych wylosowanych z rozkładu Bernoulliego z $p = 0.5$.



Rysunek 5.13: Dwa procesy interpolacji pomiędzy dwoma obiektami ze zbioru treningowego wykonane na reprezentacjach binarnych. Pośrednie zmienne ukryte zostały utworzone przez negowanie stopniowo wzrastającej liczby bitów różniących zmienne ukryte obiektów krańcowych zgodnie z pewną ustaloną, losową permutacją.



Rysunek 5.14: Wyniki wykonywania operacji arytmetycznych na zmiennych ukrytych podobnie jak na rys. 5.3, ale dla reprezentacji zbinaryzowanej. Wyliczona zmienna ukryta została dodatkowo obcięta do przedziału $[0, 1]$.

Rozdział 6.

Wnioski

W tej pracy przedstawiono kompleksowy opis teoretyczny autoenkodera wariancyjnego (VAE) oraz wskazano, jak można wykorzystać go, do znalezienia efektywnej reprezentacji dla chmur punktów 3D. Co więcej, dzięki zastosowaniu rozszerzeń do metody podstawowej opisano, w jaki sposób można regularyzować zmienną ukrytą różnymi skomplikowanymi rozkładami. Wykorzystując jedno z rozszerzeń uzyskano model generatywny odkrywający naturalny podział danych na podkategorie. Dalej wykorzystano autoenkoder wariancyjny z innym rozszerzeniem do uzyskania dyskretnej reprezentacji binarnej, na której wciąż możliwe było interpolowanie pomiędzy obiektami oraz edycja obiektów za pomocą intuicyjnych operacji na zmiennych ukrytych. Opisane modele zaimplementowano i przetestowano pod kątem zdolności rekonstrukcji i możliwości generatywnych, co pokazało ich dobre wyniki i liczne możliwości.

Przeprowadzona w tej pracy analiza autoenkoderów wariancyjnych potwierdza ich ogromny potencjał. Dzięki zastosowaniu odpowiednio zaawansowanej architektury (PointNet) VAE potrafi dopasować się do trudnych zbiorów danych i osiągnąć satysfakcjonujące wyniki. Ponadto bazując na podstawach VAE i wprowadzając do niego rozszerzenia można uzyskać modele generatywne pełniące dodatkowe funkcje, takie jak produkowanie reprezentacji binarnej, czy klasteryzacja, i osiągające bardzo dobre rezultaty.

Bibliografia

- [1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, “Shapenet: An information-rich 3d model repository,” *arXiv preprint arXiv:1512.03012*, 2015.
- [2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.
- [3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.
- [4] Y. Rubner, C. Tomasi, and L. J. Guibas, “The earth mover’s distance as a metric for image retrieval,” *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.
- [5] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, IEEE, 2015.
- [6] H. Fan, H. Su, and L. J. Guibas, “A point set generation network for 3d object reconstruction from a single image,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.
- [7] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, “Learning representations and generative models for 3d point clouds,” *arXiv preprint arXiv:1707.02392*, 2017.
- [8] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1.
- [9] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

- [11] M. Zamorski, M. Zięba, R. Nowak, W. Stokowiec, and T. Trzciński, “Adversarial autoencoders for generating 3d point clouds,” *arXiv preprint arXiv:1811.07605*, 2018.
- [12] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” in *Advances in neural information processing systems*, pp. 3581–3589, 2014.
- [13] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, “Deep unsupervised clustering with gaussian mixture variational autoencoders,” *arXiv preprint arXiv:1611.02648*, 2016.
- [14] R. Shu, “Gaussian mixture vae: Lessons in variational inference, generative models and deep nets.” <http://ruishu.io/2016/12/25/gmvae/>, December 2016.
- [15] P. W. Glynn, “Likelihood ratio gradient estimation for stochastic systems,” *Commun. ACM*, vol. 33, pp. 75–84, Oct. 1990.
- [16] M. Jankowiak and F. Obermeyer, “Pathwise derivatives beyond the reparameterization trick,” *arXiv preprint arXiv:1806.01851*, 2018.
- [17] S. Kullback and R. A. Leibler, “On information and sufficiency,” *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.
- [19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” pp. 448–456, 2015.
- [20] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, pp. 3111–3119, 2013.