Zastosowanie autoenkoderów wariacyjnych do rozpoznawania zmian na obrazach medycznych

(The usage of variational autoencoders
to recognize changes in medical images)

Tomasz Nanowski

Praca inżynierska

Promotor: dr Jan Chorowski

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

5 lutego 2019

Tomasz Nanowski

.....

.....
(adres zameldowania)

.....

.....
(adres korespondencyjny)

PESEL: .....

e-mail: .....

Wydział Matematyki i Informatyki
stacjonarne studia I stopnia
kierunek: informatyka
nr albumu: 279076

Oświadczenie
o autorskim wykonaniu pracy dyplomowej

Niniejszym oświadczam, że złożoną do oceny pracę zatytułowaną *Zastosowanie autoenkoderów wariacyjnych do rozpoznawania zmian na obrazach medycznych* wykonałem/am samodzielnie pod kierunkiem promotora, dr. Jana Chorowskiego. Oświadczam, że powyższe dane są zgodne ze stanem faktycznym i znane mi są przepisy ustawy z dn. 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (tekst jednolity: Dz. U. z 2006 r. nr 90, poz. 637, z późniejszymi zmianami) oraz że treść pracy dyplomowej przedstawionej do obrony, zawarta na przekazanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

Wrocław, 5 lutego 2019

(czytelny podpis)

Streszczenie

W tej pracy sprawdzam skuteczność wykorzystania metody uczenia bez nadzoru w problemie lokalizowania zmian nowotworowych na zdjęciach wykonanych metodą rezonansu magnetycznego. Bazuje na obserwacji, że zmiany patologiczne są rzadkie i są pewnym odstępstwem od normy. Wykorzystałem w tym celu autoenkoder wariacyjny, który pozwala oszacować prawdopodobieństwo wystąpienia danej próbki, co pozwoliło mi na ich klasyfikowanie. Pomysł przetestowałem na danych syntetycznych, wykorzystując do tego zbiór MNIST. Otrzymane wyniki okazały się być zadowalające i zachęcały do przeprowadzenia dalszych eksperymentów już na danych medycznych. Niestety w tym przypadku nie można było uznać tego za sukces, a według mojej analizy model nauczył się jedynie zwracać uwagę na prostą własność, jaką jest jasność próbki skorelowana z ilością kontrastu użytego podczas obrazowania.

Spis treści

Rozdział 1.

Wstęp

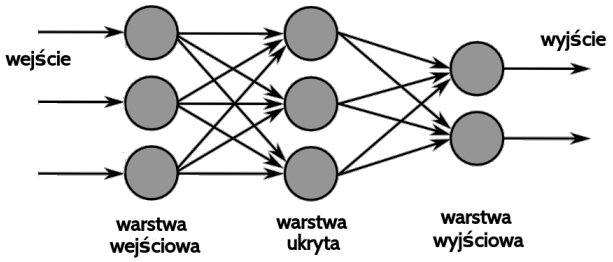
1.1. Przedstawienie problemu

Problem, który rozważam to lokalizowanie zmian nowotworowych na zdjęciach wykonanych metodą rezonansu magnetycznego (MRI, ang. magnetic resonance imaging). Można podejść do niego od strony nadzorowanej, czyli bazować na danych przeanalizowanych wcześniej przez specjalistów, gdzie każdy przypadek został ręcznie obejrzany i oznaczony. O ile taka metoda ma wiele zalet, jak chociażby korzystanie z rzeczywistej wiedzy eksperckiej, to wykorzystywany zbiór jest bardzo kosztowny w przygotowaniu i dalszym rozwoju, a ponadto może być ograniczony jedynie do pojedynczej partii ciała. Zauważając te wady oraz łącząc je z obserwacją, że zmiany patologiczne tak na prawdę są rzadkie i są pewnym odstępstwem od normy, chce spróbować skorzystać z metody nienadzorowanej, w której to model nauczyłby się oszacowywać prawdopodobieństwo występowania pojedynczej próbki w pewnym kontekście. Przy takim podejściu mogę oznaczać obserwacje mało prawdopodobne jako właśnie te nowotworowe. Dodatkowo sama tak informacja może również pomóc specjalistom przy wyłanianiu obszarów podejrzanych. Model, na którego wykorzystanie się zdecydowałem to autoenkoder wariacyjny (VAE, ang. Variational Autoencoder), łączący sztuczne sieci neuronowe z modelowaniem probabilistycznym. Pomysł ten przetestuje początkowo na danych syntetycznych z wykorzystaniem zbioru MNIST.

1.2. Sztuczne sieci neuronowe

Sztuczne sieci neuronowe mają obecnie bardzo mocno ugruntowaną pozycję szczególnie w dziedzinie problemów związanych z analizą i przetwarzaniem obrazów. Pomimo, iż nie jest to nowy pomysł, dopiero ostatni wzrost w wydajności komputerów pozwolił na ich praktyczne zastosowanie. Z matematycznego punktu widzenia są to sparametryzowane nieliniowe funkcje o pewnej ustalonej strukturze. Składają

się z prostych elementów zwanych neuronami, a one natomiast są pogrupowane w warstwy. Połączenia między warstwami definiują przepływ danych. 'Nauka' sieci neuronowych polega na optymalizacji pewnej funkcji straty, czyli wyznaczeniu takich parametrów, żeby osiągnąć minimalny koszt. Do tego celu często korzysta się z metod opartych na Stochastic Gradient Descent, a przy wybranej strukturze można w efektywny sposób zastosować algorytm propagacji wstecznej. W dalszej części pracy będę używał prostszej nazwy - sieci neuronowe. Przykładowa architektura sieci neuronowych jest zaprezentowana na rysunku 1.1.



Rysunek 1.1: Przykładowy model sieci neuronowej

1.3. Uczenie nadzorowane i bez nadzoru

Są to dwa różne rodzaje problemów z dziedziny uczenia maszynowego. W przypadku uczenia nadzorowanego wiemy jaką chcemy uzyskać odpowiedź dla danego wejścia i próbujemy tak dobrać parametry modelu, żeby odpowiadał on z jak największą poprawnością. Przykładowymi zadaniami tego rodzaju są regresja oraz klasyfikacja, która nawiasem mówiąc może zostać uznana za problem regresji, ale na dyskretnym, skończonym zbiorze.

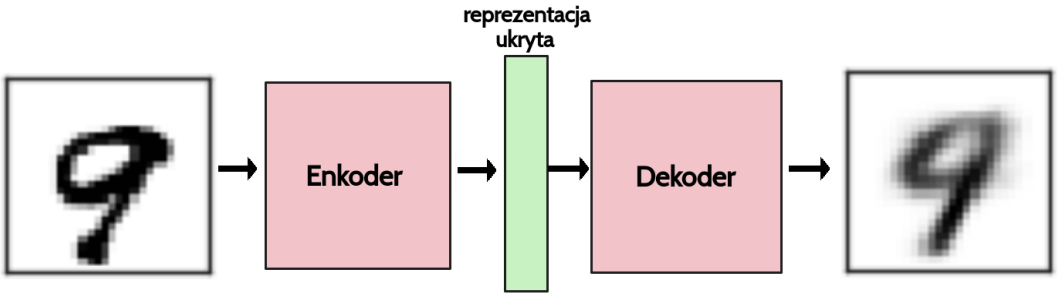
Inaczej jest w sytuacji bez nadzoru. Tam dla danych wejściowych nie znamy interesującej nas odpowiedzi i staramy się tak zbudować model, żeby był on w stanie sam ją wydobyć. Zazwyczaj interesuje nas rozwiązanie takich problemów jak estymacja rozkładu prawdopodobieństwa z którego pochodzą dane, klasteryzacja, redukcja wymiarowości czy wydajne kodowanie danych. Jednym z modeli wykorzystywanym w tej klasie zadań jest właśnie autoenkoder.

W przypadku identyfikacji próbek odstających można zwracać uwagę na ich estymowane prawdopodobieństwo oraz efektywność kodowania i dekodowania, które oba powinny być niskie.

1.4. Autoenkoder

Jest to jeden z rodzai sieci neuronowych, służący do znajdowania wydajnej reprezentacji danych, co jak wspomniałem wcześniej jest przykładem nauki bez nad-

zoru. Myślę, że mogę go określić mianem nieliniowej alternatywy dla klasycznej statystycznej metody analizy głównych składowych (PCA, ang. principal component analysis). W autoenkoderach można wyróżnić dwie połączone ze sobą części zwane enkoderem i dekoderem. Zadaniem enkodera jest wyprodukowanie właśnie tej reprezentacji, podczas gdy dekoder służy do odtworzenia z niej oryginalnej postaci. Zależy nam na tym, żeby wyjście było w jakimś sensie jak najbardziej podobne do wejścia przy zachowaniu odpowiednio małej wymiarowości kodowania. Jest to pewnego rodzaju balansowanie pomiędzy ilością informacji, na których przepływ się zgadzamy, a ich jakością. W szczególności można pomyśleć o takiej patologicznej sytuacji jak ustalenie rozmiaru reprezentacji równej rozmiarowi danych wejściowych, co prawdopodobnie będzie skutkować idealnymi rekonstrukcjami, ale przy okazji zerową wiedzą płynącą z takiego modelu. Analogicznie można rozważyć przypadek kiedy obszar kodowania jest za mały. Model w takiej sytuacji skupi się jedynie na przekazaniu wyłącznie trywialnych cech, żeby mimo wszystko jakkolwiek odtworzyć dane. Poglądowy schemat budowy znajduje się na rysunku 1.2.



Rysunek 1.2: Architektura autoenkodera

W przypadku obrazów na funkcję straty często wybierany jest błąd średniokwadratowy (MSE, ang. Mean Squared Error).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{1.1}$$

gdzie:

Y_i – oryginalne dane
$\hat{Y}_i$ – zrekonstruowane dane

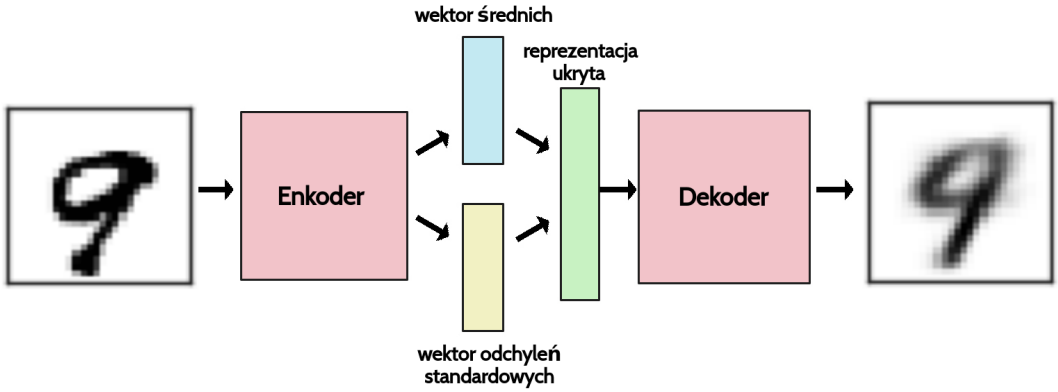
Wymagającym sprostowania może być fakt, iż mimo mówimy jakie oczekujemy wyjście z modelu, w sensie rekonstrukcji, to dalej jest to problem typu nauki bez

nadzoru, ponieważ rzeczywistą szukaną przez nas wiedzą jest umiejętność kodowania danych i osobno ich dekodowania. Chciałbym w tym miejscu zastanowić się co dzięki temu zyskujemy. Już teraz zaletą jest oczywiście redukcja wymiarowości, która ułatwia późniejszą analizę danych oraz pozwala na ich kompresję i ewentualne odszumianie. Dodatkowo wydaje się, że mając wytrenowany dekoder, na konkretnej rodzinie danych, moglibyśmy zaaplikować do niego jakieś nowe kodowanie i otrzymać w ten sposób wynik pochodzący z oryginalnej dziedziny, czyli wykorzystać go jako generator nowych próbek. Niestety ten pomysł ma taki problem, że podczas nauki nie ma żadnej kontroli nad tym jak dane zostaną rozmieszczone w przestrzeni, która notabene jest nieskończona (z ograniczeniem do pojemności liczb zmiennoprzecinkowych). W takim razie nie wiemy jaka podprzestrzeń odpowiada tym kodowaniom, które są sensownie dekodowane. Jest to potrzebne przy generowaniu. Dodatkowo jeśli przestrzeń byłaby nieciągła, to niemożliwe byłoby interpolowanie pomiędzy próbkami. Znaczącym problemem jest natomiast to, że kodowanie jako wektor liczb jest bardzo precyzyjną informacją, co może doprowadzić do przeuczenia modelu i błędnego działania na danych pochodzących z poza zbioru treningowego.

1.5. Autoenkoder wariacyjny

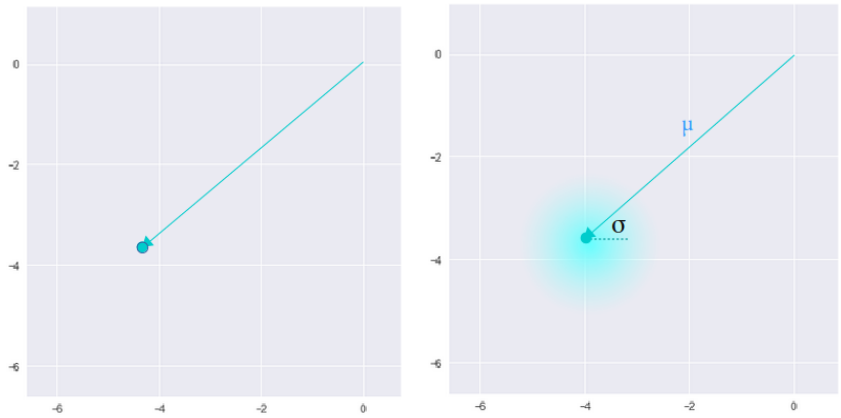
Autoenkoder wariacyjny (VAE, ang. Variational Autoencoder) rozszerza założenia podstawowej architektury o wprowadzenie modelowania rozkładu prawdopodobieństwa dla reprezentacji ukrytej. W odróżnieniu od podstawowej wersji enkoder nie produkuje pojedynczego wektora, ale dwa wektory interpretowane odpowiednio jako średnie μ oraz wariancje σ dla rozkładu Gaussa. Dopiero na podstawie tych parametrów formowane jest kodowanie, gdzie i-ta wartość losowana jest z $\mathcal{N}(\mu_i, \sigma_i^2)$, które aplikowane zostaje do dekodera i rekonstruowana jest na podstawie niego pierwotna próbka. Takie podejście znaczy, że nawet dla tego samego wejścia i podczas gdy średnie oraz odchylenia pozostaną niezmienne, to i tak reprezentacja będzie się różnić właśnie ze względu na występującą losowość. Zmodyfikowany schemat został przedstawiony na rysunku 1.3.

Intuicyjnie wektor średnich oznacza miejsce, gdzie zakodowana zmienna powinna być wycentrowana, natomiast odchylenie kontroluje obszar, wokół którego kodowanie może się różnić. Sytuację tę zaprezentowałem na wykresie 1.4, przy czym porównałem z podstawową wersją. W jej przypadku informacja wychodząca z enkodera jest bardzo precyzyjna, natomiast w obecnie rozważanym modelu, dzięki dodaniu szumu, zostaje ona rozmyta. Taki zabieg ma za zadanie ograniczyć zjawisko przeuczenia oraz wprowadzić pewną lokalną ciągłość w przestrzeni. Jest to wynikiem tego, że gdy kodowanie losowane jest z wnętrza kuli, notabene o nieskończonym promieniu, ale ustalonej gęstości, to dekoder zmuszony jest nauczyć się, że nie tylko pojedynczy punkt odnosi się do danej próbki, ale i również jego całe lokalne sąsiedztwo. To pozwala dekodować nie tylko specyficzne reprezentacje (pozostawiając przestrzeń nieciągłą), ale też te trochę różniące się, ponieważ model narażony



Rysunek 1.3: Architektura autoenkodera wariacyjnego

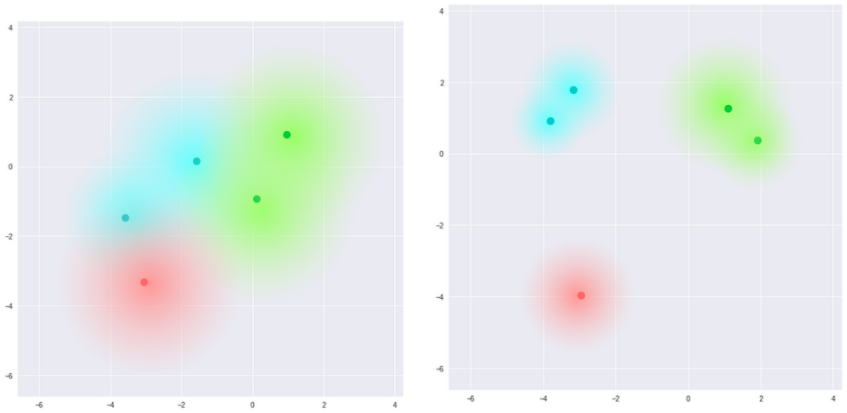
jest na szereg zmian kodowania tego samego wejścia podczas treningu.



(a) Standardowy autoenkoder (b) Autoenkoder wariacyjny

Rysunek 1.4: Porównanie precyzji informacji dostarczanych przez enkodery

Model jest teraz wystawiony na pewien stopień lokalnej zmienności przez wprowadzenie szumu do kodowania pojedynczej próbki, co skutkuje ciągłą przestrzenią w reprezentacji ukrytej dla lokalnego sąsiedztwa. Jednak na razie nie ma ograniczeń co do wartości jakie mogą być przyjmowane przez wektory μ i σ, a w rezultacie enkoder może nauczyć się generować bardzo różne μ dla różnych klas i grupować je minimalizując σ, upewniając się przy tym, że samo kodowanie nie różni się zbytnio dla tej samej próbki. W szczególności prawdopodobnym jest wystąpienie tak patologicznej sytuacji, gdzie σ zawsze wynosi 0, przez co nowy model nie będzie różnić się w działaniu od jego podstawowej wersji, włącznie z uwzględnieniem wad, które starał się wyeliminować. Omawiany przeze mnie przykład przedstawiony jest na rysunku 1.5. Oczekiwaną sytuacją byłoby rozłożenie danych blisko siebie, ale z zachowaniem ich separowalności. Pozwoliłoby to na gładką interpolacje pomiędzy próbkami i generowanie nowych danych.



(a) Oczekiwany układ (b) Niezamierzony układ

Rysunek 1.5: Porównanie rozkładów danych ze względu na brak ograniczeń dla reprezentacji. Przy czym ten pierwszy jest lepszy ponieważ obszar z którego generować można nowe dane jest spójny

W celu osiągnięcia zamierzonych rezultatów, należy wzbogacić funkcję straty o koszt dywergencji Kullbacka-Leiblera, która mierzy różnicę pomiędzy dwoma rozkładami prawdopodobieństwa. Minimalizowanie go oznacza optymalizowanie parametrów rozkładu (μ i σ), tak żeby jak najbardziej przypominał docelowy rozkład, który w przypadku autoenkodera wariacyjnego będzie standardowym rozkładem normalnym ($\mu = 0$, $\sigma = 1$). Przy takim wyborze dane powinny zostać rozrzucone dookoła $\vec{0}$ i nie pozostawiać pustych miejsc w przestrzeni ukrytej.

$$\text{KLD} = \sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1 \tag{1.2}$$

1.5.1. Model probabilistyczny

To co do tej pory napisałem odnośnie autoenkodera wariacyjnego raczej trzeba uznać za zbiór intuicji i oczekiwań. W tym rozdziale chciałbym pokazać jego funkcjonowanie od strony probabilistycznej.

Myślimy, że autoenkoder wariacyjny modeluje wspólny rozkład prawdopodobieństwa $p(x, z)$, gdzie x to obserwowane dane, a z to zmienna ukryta. Możemy to rozpisać w następujący sposób $p(x, z) = p(x|z)p(z)$. Wróćmy teraz do założeń, gdzie zależało nam na produkowaniu dobrych reprezentacji dla określonej próbki, czyli wyznaczeniu prawdopodobieństwa a posteriori $p(z|x)$. Zgodnie z twierdzeniem Bayesa

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Problem jest jednak z mianownikiem $p(x)$. Można go obliczyć marginalizując zmienne ukryte, ale wtedy $p(x) = \int p(x|z)p(z)dz$ i wymaga to przejrzenia wszystkich

kombinacji reprezentacji ukrytych. W takim razie w jakiś inny sposób trzeba policzyć to prawdopodobieństwo $p(z|x)$.

Korzystając z wnioskowania wariacyjnego (ang. variational inference) przybliżymy tę wartość na podstawie rodziny rozkładów $q_\lambda(z|x)$, gdzie jeśli q byłoby rozkładem Gaussa to $\lambda_{x_i} = \left(\mu_{x_i}, \sigma^2_{x_i}\right)$

Do sprawdzenia jak dobrze przybliżamy te rozkłady możemy skorzystać z dywergencji Kullbacka-Leiblera, która mierzy ilość straconych informacji, kiedy używamy q do zaokrąglenia p (w natach)

$$\begin{aligned} \mathbb{KL}\left(q_\lambda(z|x) \| p(z|x)\right) = \mathbf{E}_q\left[\log \frac{q_\lambda(z|x)}{p(z|x)}\right] = \\ \mathbf{E}_q\left[\log p(z|x)\right] - \mathbf{E}_q\left[\log \frac{q_\lambda(x,z)}{p(x)}\right] = \\ \mathbf{E}_q\left[\log q_\lambda(z|x)\right] - \mathbf{E}_q[\log p(x,z)] + \log p(x) \end{aligned} \tag{1.3}$$

Celem jest znalezienie parametru λ, który minimalizuje tę sumę.

$$q^*_\lambda(z|x) = \arg\min_\lambda \mathbb{KL}\left(q_\lambda(z|x) \| p(z|x)\right) \tag{1.4}$$

Niestety nie jest to obliczalne ze względu an ponownie występujący element $p(x)$, tak jak w dyskusji powyżej. Możemy jednak rozważyć funkcję

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x,z)] - \mathbf{E}_q\left[\log q_\lambda(z|x)\right] \tag{1.5}$$

Zauważmy, że łączy się to z powyższą równością co przepiszmy jako

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}\left(q_\lambda(z|x) \| p(z|x)\right) \tag{1.6}$$

Skorzystajmy teraz z faktu, że dywergencja Kullbacka-Leiblera jest nieujemna, czyli zamiast ją minimalizować możemy maksymalizować $ELBO$. Wniosek jest taki, że $ELBO$ pozwala oszacowywać dolne prawdopodobieństwo na wystąpienie danej próbki, a dodatkowo jest to wartość obliczalna.

Przy założeniu, że dane są niezależne, które wykonujemy w przypadku autoenkodera, możemy rozłożyć $ELBO$ na sumę składników, gdzie każdy dotyczy tylko

pojedynczej próbki. Wtedy dla pojedynczego elementu otrzymujemy

$$\begin{aligned} ELBO_i(\lambda) = \mathbf{E}_q[\log p(x_i, z)] - \mathbf{E}_q\left[\log q_\lambda(z|x_i)\right] = \\ \mathbf{E}_q[\log p(x_i|z)p(z)] - \mathbf{E}_q\left[\log q_\lambda(z|x_i)\right] = \\ \mathbf{E}_q[\log p(x_i|z)] + \mathbf{E}_q[\log p(z)] - \mathbf{E}_q\left[\log q_\lambda(z|x_i)\right] = \\ \mathbf{E}_q[\log p(x_i|z)] - (\mathbf{E}_q\left[\log q_\lambda(z|x_i)\right] - \mathbf{E}_q[\log p(z)]) = \\ \mathbf{E}_q[\log p(x_i|z)] - \mathbb{KL}\left(q_\lambda(z|x_i)\|p(z)\right) = \\ \mathbb{E}q_\lambda(z|x_i)\left[\log p\left(x_i|z\right)\right] - \mathbb{KL}\left(q_\lambda(z|x_i)\|p(z)\right) \end{aligned} \tag{1.7}$$

Teraz już możemy połączyć to z sieciami neuronowymi. Do znalezienia parametrów dla $q_\theta(z|x,\lambda)$ posłuży enkoder, który z x produkuje λ. Natomiast dekoder bierze reprezentację ukrytą i zwraca parametry dla rozkładu danych $p_\phi(x|z)$. Wartości θ i ϕ możemy też traktować jako parametry dla sieci neuronowej. Wtedy funkcja straty, którą będziemy minimalizować prezentuje się następująco

$$l_i(\theta,\phi) = -ELBO_i(\theta,\phi) = -\mathbb{E}q_\theta(z|x_i)\left[\log p_\phi\left(x_i|z\right)\right] + \mathbb{KL}\left(q_\theta(z|x_i)\|p(z)\right) \tag{1.8}$$

Interpretacja:

- $-\mathbb{E}q_\theta(z|x_i)\left[\log p_\phi\left(x_i|z\right)\right]$ można utożsamiać z błędem rekonstrukcji, gdyż odpowiada temu składnik $\log p(x|z)$, a w trakcie treningu ta liczb powinna być jak najmniejsza. Dodatkowo jest to potęgowane przez $q(z|x)$, czyli pewność co do reprezentacji ukrytej.
- $\mathbb{KL}\left(q_\theta(z|x_i)\|p(z)\right)$ można interpretować jako ilość przesyłanych informacji, ponieważ jeżeli model będzie chciał odejść od rozkładu a priori $p(z)$ to zapłaci za to karę, ale będzie w stanie przekazać wiadomość.

1.6. Ocena jakości modelu

1.6.1. Krzywa ROC i AUC

Krzywa ROC (Receiver Operating Characteristic) jest narzędziem do oceny poprawności klasyfikatora binarnego. Bazuje ona na wyliczaniu charakterystyki jakościowej modelu predykcji w wielu różnych punktach odcięcia. Działa to na takiej zasadzie, że model dla próbek przewiduje z jaką pewnością pochodzą z klasy 1. Następnie badane są różne progi i klasyfikowane przy ich użyciu obiekty. Dla uzyskanych klasyfikacji liczymy odsetek prawdziwych pozytywnych (TPR) oraz odsetek fałszywych pozytywnych (FPR) i nanosimy te wartości na wykres. Warto zauważyć, że dla losowego modelu jego wykres to prosta przechodząca przez (0, 0) i (1, 1).

Dzieje się tak, ponieważ dla każdego progu połowa klasyfikacji będzie nad i połowa pod. Idealny model znajduje się w punkcie (0, 1).

Przydatne jest również obliczenie pola powierzchni pod krzywą AUROC (ang. Area Under the ROC). Pozwala ona ocenić ogólną skuteczność modelu bez wyboru konkretnego progu. Interpretuje się ją jako prawdopodobieństwo, że badany model predykcyjny oceni wyżej losowy element klasy pozytywnej od losowego elementu klasy negatywnej.

Rozdział 2.

Eksperyment na danych syntetycznych (MNIST)

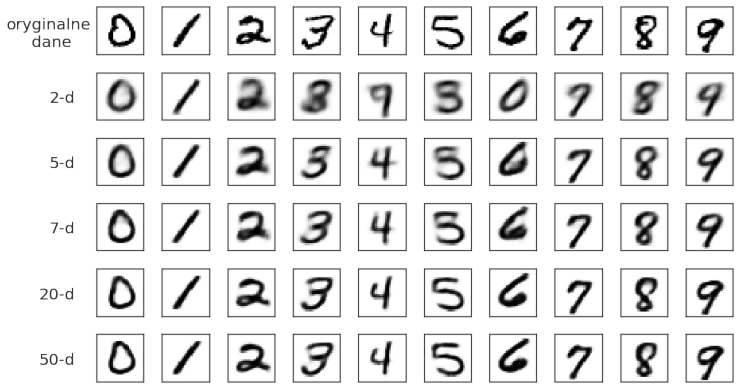
Do przetestowania pomysłu posłuży mi zbiór MNIST. Jest to zestaw odręcznie napisanych cyfr, które następnie zostały znormalizowane względem rozmiaru i wycentrowane. Każdy obrazek ma rozmiar 28x28 pikseli, mających wartości całkowite z przedziału [0, 255]. Na zbiór składa się 60 tys. danych treningowych i 10 tys. testowych. Przykładowe obrazki znajdują się na rysunku 2.1.

Rysunek 2.1: Przykładowe dane ze zbioru MNIST

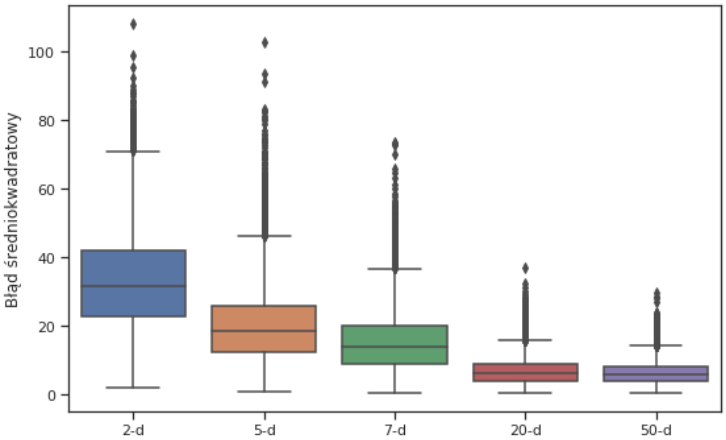
2.1. Mozliwości autoenkodera wariacyjnego

Zacząłem od sprawdzenia podstawowych własności autoenkodera wariacyjnego, przedstawionych w rozdziale 1.5. W tym celu wytrenowałem modele o różnych rozmiarach reprezentacji ukrytej: 2, 5, 7, 20, 50. Następnie sprawdziłem jak radzą sobie

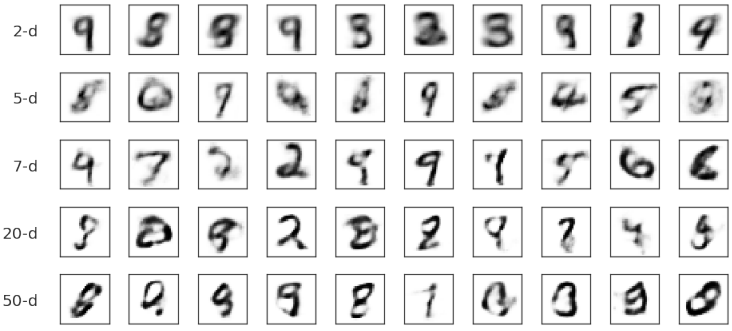
z rekonstrukcją danych ze zbioru testowego oraz generowaniem nowych próbek. Wyniki prezentują się na rysunku 2.2.



(a) Przedstawienie rekonstrukcji w zależności od rozmiaru reprezentacji ukrytej



(b) Porównanie błędów średniokwadratowych na rekonstrukcjach dla zbioru testowego

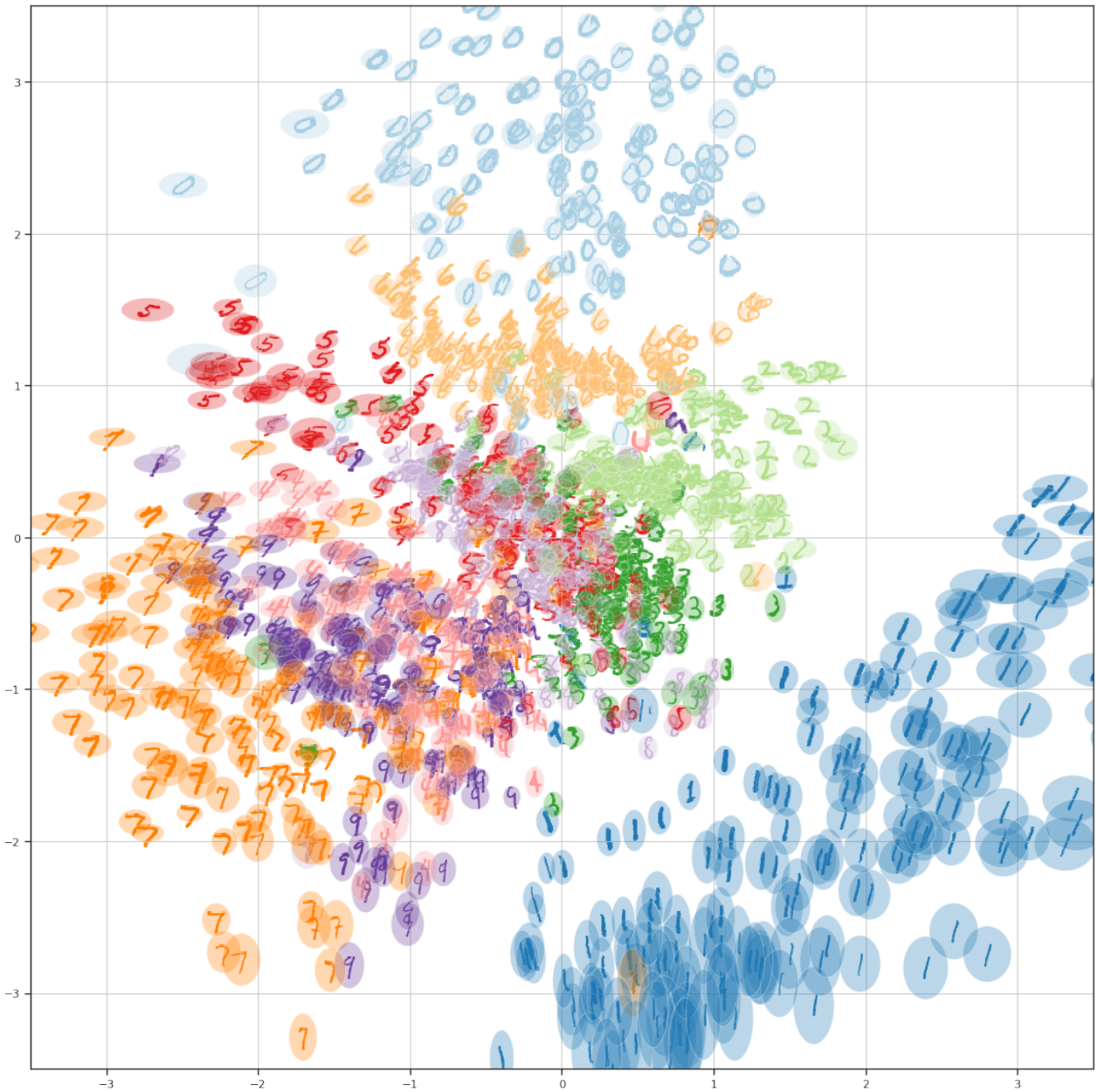


(c) Przykładowe zdekodowane próbki wylosowane z $\mathcal{N}(0, 1)$

Rysunek 2.2

Ze względu na łatwość wizualizacji, do sprawdzenia jak dane reprezentowane są w przestrzeni ukrytej wybrałem model bazujący na wymiarze rozmiaru 2. Zdaję sobie sprawę, że rekonstrukcje w tym przypadku nie są najlepszej jakości, ze względu na zbyt ograniczoną ilość informacji możliwych do przekazania, ale zależy mi też

na przedstawieniu podstawowych założeń. W tym celu na wykresie dla losowych próbek zaznaczyłem odpowiadające im średnie oraz odchylenia. Otrzymany rezultat widać na obrazku 2.3, gdzie można zauważyć opisane wcześniej własności wybranego modelu takie jak skoncentrowanie reprezentacji wokół $\vec{0}$ czy ciągłość przestrzeni.



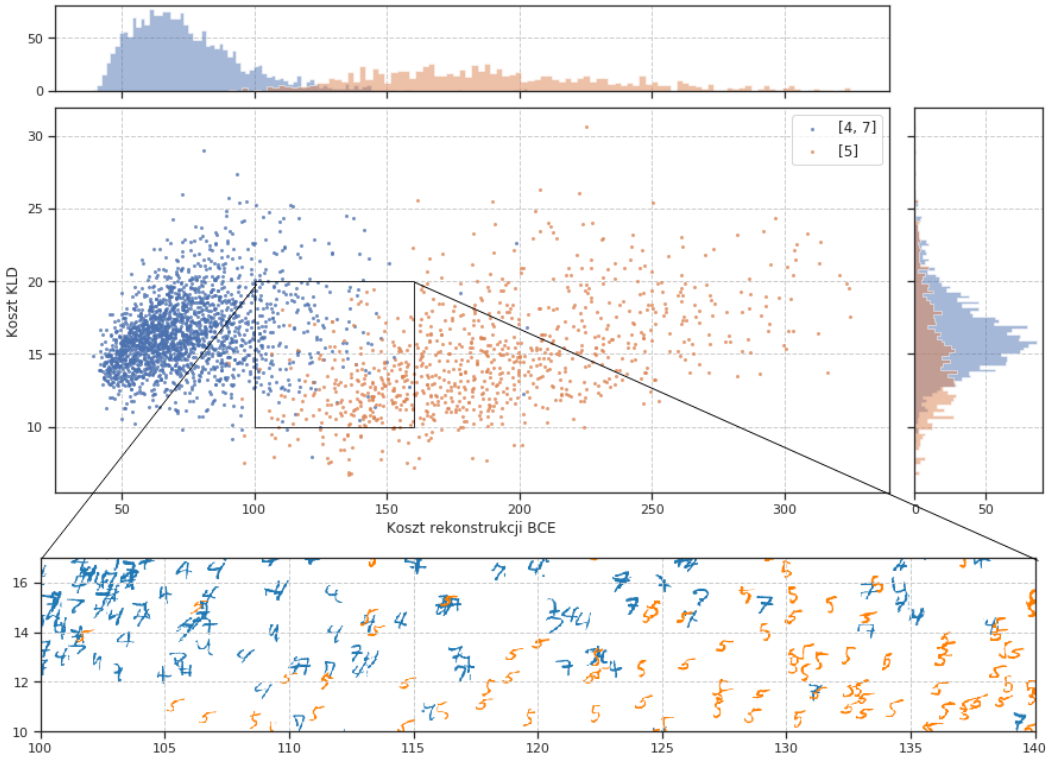
Rysunek 2.3: Rozkład konkretnych próbek w przestrzeni ukrytej z zaznaczonym odchyleniem standardowym

2.2. Symulacja docelowego problemu

W tym miejscu interesuje mnie odtworzenie oryginalnego problemu na prostszych danych, jakim jest właśnie zbiór MNIST. Pozwoli mi to stwierdzić sensowność skorzystania z wybranego modelu. Zakładam, że jeżeli eksperyment nie powiódłby się na łatwiejszych danych, to raczej nie powiedzie się również na tych skomplikowanych. Dodatkowo zyskam pewnego rodzaju doświadczenie z wybraną architekturą.

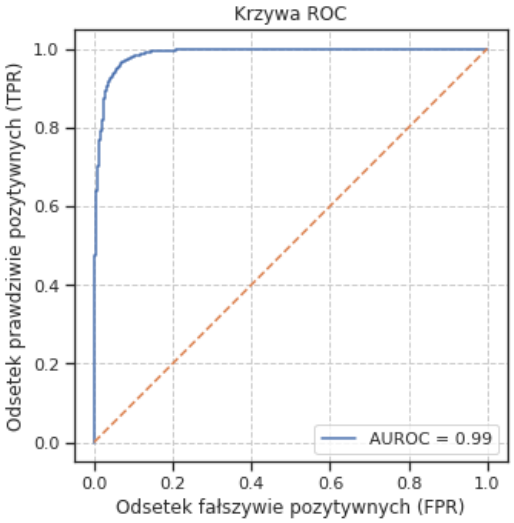
Przygotowana przeze mnie symulacja prezentuje się następująco. Zbiór danych składa się z trzech cyfr: 4, 5, 7, z czego 5 stanowi jedynie 1% wszystkich próbek. Ma to odpowiadać rzadkości sytuacji patologicznych. Natomiast zadaniem jest stwierdzenie czy dana obserwacja jest cyfrą 5 czy nie.

Klasyfikacja odbywać się będzie na podstawie wartości funkcji straty dla autoenkodera wariacyjnego, którą można interpretować jako ograniczenie prawdopodobieństwa. Takie podejście wymaga ustalenia pewnego progu kategoryzacji, dlatego w tym miejscu będę prezentował krzywą ROC, która rozważa wszystkie sensowne wartości. Dodatkowo zaprezentuję dwa składniki błędu oddzielnie, przez co będzie można zaobserwować, który z nich ma największy wpływ.



Rysunek 2.4: Rozkład punktów wraz z przykładami ze względu na poszczególne składniki funkcji straty

Na podstawie wniosków z rozdziału 2.1. wybrałem rozmiar warstwy ukrytej równy 10. Wyniki zaprezentowane są na rysunku 2.4 oraz 2.5. Poza tym, że widać dobrą separację danych, na przybliżonym wycinku można zauważyć obok ogólnie mało prawdopodobnych cyfr 5 cyfry 7 z poprzeczną kreską, które same w sobie są rzadkie.



Rysunek 2.5: Krzywa ROC dla modelu wyuczonego na danych syntetycznych z zachowaniem odpowiedniej dysproporcji w danych

2.3. Wnioski

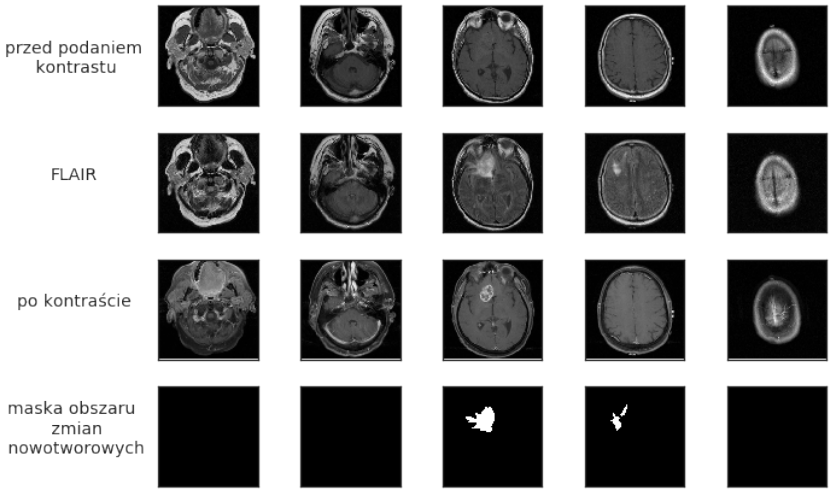
Na podstawie otrzymanych wyników mogę stwierdzić, że eksperyment się powiódł. W przypadku danych MNIST udało się sklasyfikować dane, które są rzadkie i stanowią jedynie odsetek wszystkich. Jest sens w takim razie w użyciu tego modelu dla już prawdziwych danych.

Rozdział 3.

Eksperyment na danych właściwych (MRI FLAIR)

3.1. Opis danych

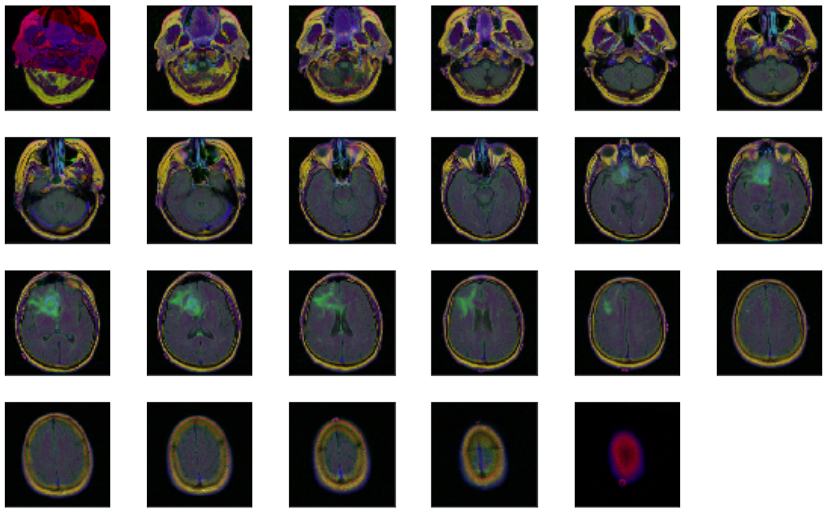
Dane pochodzą z Uniwersytetu Duke'a (ang. Duke University). Są to zdjęcia głowy wykonane metodą obrazowania rezonansu magnetycznego w technice FLAIR (ang. fluid-attenuated inversion recovery) wraz z zaznaczonymi obszarami zmian nowotworowych. Obrazy mają rozmiar 256x256x3 pikseli, gdzie pierwszy kanał odpowiada momentowi przed wprowadzeniem kontrastu, trzeci po, a drugi jest właściwym zdjęciem. Maski mają rozmiar 256x256 pikseli o wartościach odpowiednio 255 dla komórek nowotworowych i 0 w przeciwnym wypadku. Na rysunku 3.1 pokazane są zdjęcia z podziałem na kanały i odpowiadające im maski.



Rysunek 3.1: Przedstawienie konkretnych kanałów w próbce wraz z jego maską zmian

Dane pogrupowane są dla 110 pacjentów ze zdiagnozowanym nowotworem. Na rysunku 3.2 zaprezentowałem zdjęcia pojedynczej osoby w trzech kanałach. Łącznie

w zbiorze danych znajduje się 3929 par obrazów, przy czym jedynie $\sim$ 1.02988% pikseli zostało zidentyfikowanych jako komórki nowotworowe, co potwierdza obserwację odnośnie ich rzadkości.



Rysunek 3.2: Przykładowy zestaw obrazków dla pojedynczego pacjenta

3.2. Wstępna obróbka

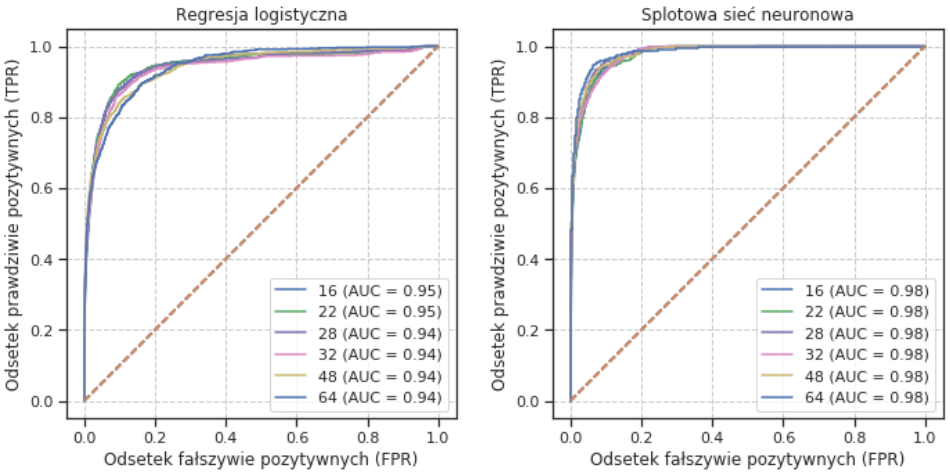
Dane podzieliłem na 2 rozłączne zbiory ze względu na pacjentów w stosunku 70% dla zestawu treningowego i 30% dla testowego.

3.2.1. Podział na mniejsze kawałki

Danymi, które rzeczywiście mnie interesują są poszczególne komórki w mózgu. To co chciałbym umieć oceniać to ich patologiczność. Mam zamiar to robić na podstawie kontekstu w jakim się znajdują, czyli lokalnego sąsiedztwa na zdjęciu. W praktyce sprowadzi się to do utożsamiania wycinka obrazu rozmiaru n x n z wartością komórki leżącą w jego środku. W takim sensie chce podzielić zbiór danych na mniejsze kawałki, żeby każdej komórce odpowiadał jej kontekst. Jednak parametrem wymagającym ustalenia jest rozmiar takiego sąsiedztwa. Musze wiedzieć czy kawałek o danych wymiarach będzie zawierał wystarczającą ilość informacji potrzebnych do poprawnej klasyfikacji. Teoretycznie za kontekst mógłbym uznać cały obrazek, ale zależy mi też na ograniczeniu rozmiaru modelu i potrzebnej w związku z tym mocy obliczeniowej. Do wyznaczenia tej wartości skorzystam z podejścia nauki nadzorowanej, co może wydawać się sprzeczne z moją początkową deklaracją, ale ten krok służy jedynie we wsparciu wyboru optymalnego rozmiaru i nie jest wymagany.

Decyzję podejmę na podstawie rezultatów osiąganych przez modele takie jak regresja liniowa oraz splotowa sieć neuronowa dla następujących rozmiarów: 16, 22,

28, 32, 48, 64. Zakładam, że jeśli w jakimś przypadku dokładność będzie zadowalająca, to ilość obecnych informacji wystarczy do rekonstrukcji. Warto w tym miejscu wspomnieć, że przy takim podejściu muszę zrównoważyć dane w zbiorze treningowym, ponieważ aktualnie jednych jest około 100 razy mniej niż drugich. Rozwiążę to na takiej zasadzie, że w trakcie uczenia porcję danych wejściowych będę komponował losując połowę próbek z jednej klasy i połowę z drugiej. Może to wpłynąć negatywnie na obciążenie modelu, ale i tak ostateczna klasyfikacja będzie odbywać się z wykorzystaniem progu, co powinno zredukować ten błąd. Wyniki zaprezentowałem przy pomocy krzywych ROC na wykresie 3.3. Jak można zauważyć wszystkie rozważane rozmiary prezentują się równie dobrze, w związku z czym zdecydowałem się wybrać jako parametr liczbę 22.



Rysunek 3.3: Porównanie dokładności ze względu na różne rozmiary wycinków dla modeli uczonych metodą nadzorowaną

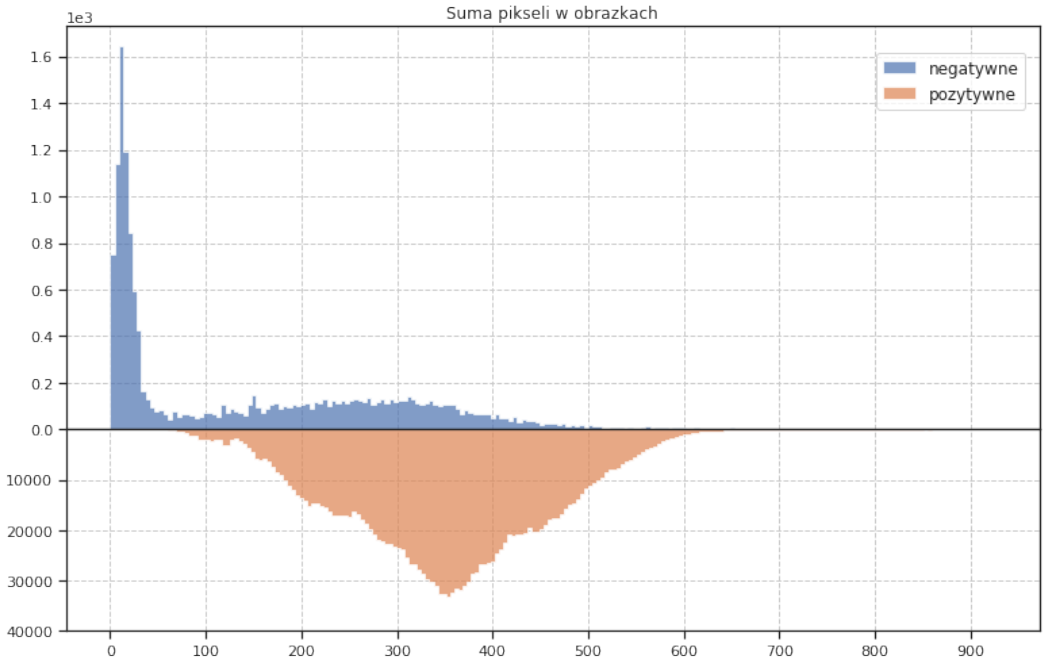
Nie jest to oczywiście najlepsza metoda dla podejścia nadzorowanego, czym mogłoby być wykorzystanie architektury U-Net, ale powinno wystarczyć jako punkt odniesienia dla wymaganej ilości informacji w wycinku.

Dodatkowo w tym miejscu chciałbym wprowadzić słownictwo jakie będę używał do określania konkretnych próbek. Te odnoszące się da zmian nowotworowych będę nazywał pozytywnymi, a pozostałe negatywnymi.

3.2.2. Dodatkowe analiza zbioru

Po zdecydowaniu się na łatki rozmiaru 22 i przygotowaniu takiego zbioru postanowiłem dodatkowo przeanalizować jego zawartość w celu znalezienia trywialnych przypadków. Jednym z nich były kawałki znajdujące się poza czaszkę, które nigdy nie są nowotworowe. Będą to obrazki o niskiej łącznej sumie pikseli, co zaprezentowane jest na histogramie 3.4 (zwracam uwagę na różne skale na osiach). Na początku można zauważyć sporą grupę przypadków niepatologicznych. Sprawdziłem, że minimalna wartość sumy dla klasy odpowiadającej nowotworom wynosi 56.5. Przy tym

założeniu usunąłem wszystkie obrazki u sumie mniejszej, niż 42.0 co zmniejszyło ilość danych o 45.5%, pozbywając się trywialnych próbek. Dany próg będę oczywiście uwzględniał w późniejszej klasyfikacji.



Rysunek 3.4: Rozkład sum pikseli dla poszczególnych rodzajów próbek (zwracam uwagę na różne skale na osiach)

3.3. Wyniki

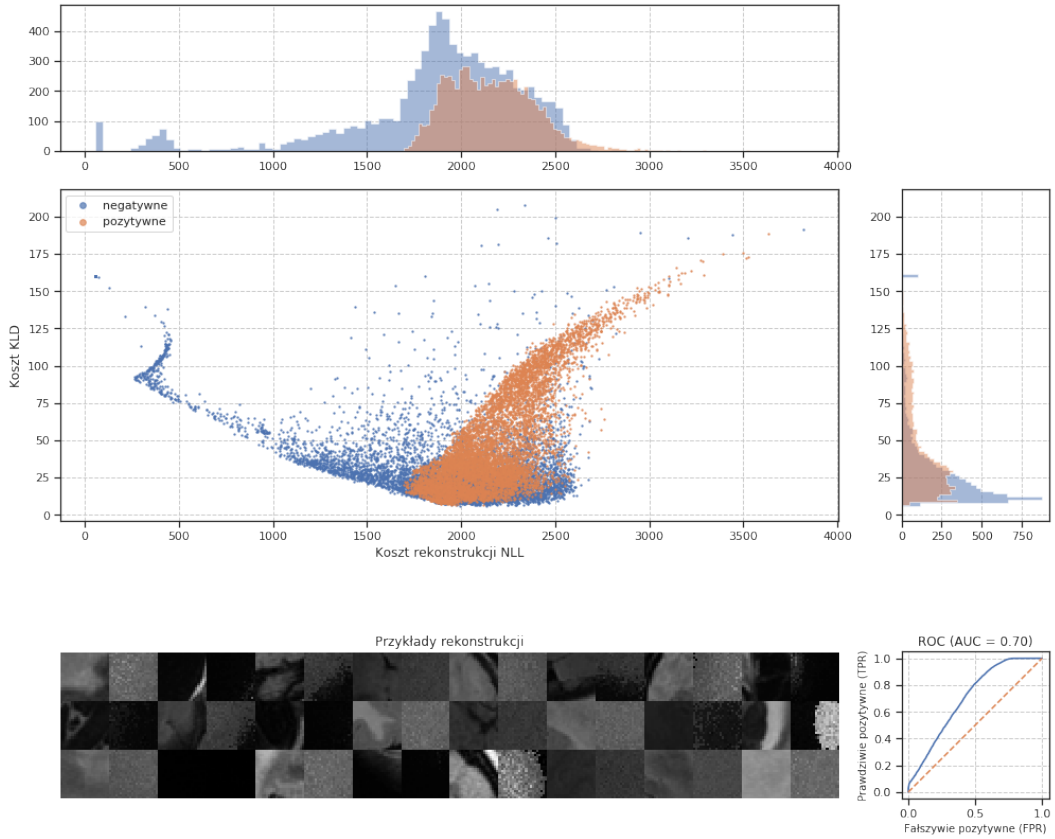
W tabeli 3.1 przedstawiłem wyniki dla modeli z różnymi rozmiarami reprezentacji ukrytej oraz w różnych etapach nauki. Tak samo jak w poprzednich eksperymentach miarą pomiarową, którą obserwowałem było AUC na podstawie wartości funkcji straty. Można zauważyć szybkie przeuczenie się modeli, ponieważ w większości przypadków już od 3 epoki wyniki pogarszają się. Dodatkowo wraz z czasem pomiary stają się prawie jednakowe dla wszystkich rozmiarów.

3.4. Analiza

Zajmę się analizą modelu, który osiągnął najlepszy rezultat, czyli autoenkodera wariacyjnego bazującego na reprezentacji ukrytej rozmiaru 100 po 3 epokach nauki. Na rysunku 3.5 widać wykres z zaznaczonymi poszczególnymi składnikami kosztu, czyli koszt KLD i rekonstrukcji NLL. Przypomnę, że na bazie ich sumy liczona jest krzywa ROC. Można zauważyć specyficzny rozkład punktów. Ponadto na rysunku pokazane są przykładowe rekonstrukcje, ale prawdopodobnie ze względu na zbyt krótki czas nauki nie prezentują się one najlepiej.

Model	Epoka					
	1	2	3	20	60	80
VAE 20-d	0.494	0.693	0.655	0.612	0.597	0.590
VAE 50-d	0.552	0.686	0.697	0.616	0.606	0.592
VAE 100-d	0.580	0.685	**0.701**	0.614	0.595	0.594
VAE 200-d	0.661	0.675	0.668	0.620	0.608	0.597
VAE 300-d	0.704	0.671	0.657	0.620	0.600	0.593

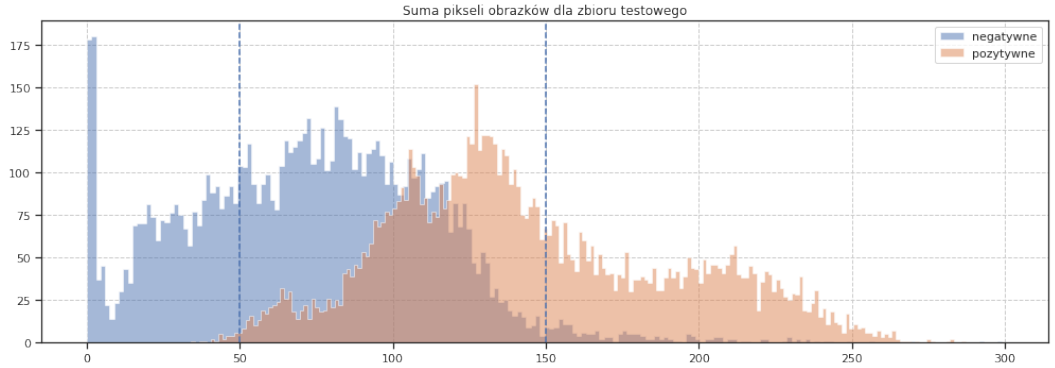
Tablica 3.1: AUC ze względu an rozmiar reprezentacji ukrytej i w różnych stadiach nauki



Rysunek 3.5: Poszczególne składniki kosztu dla modelu z najlepszą separowalnością

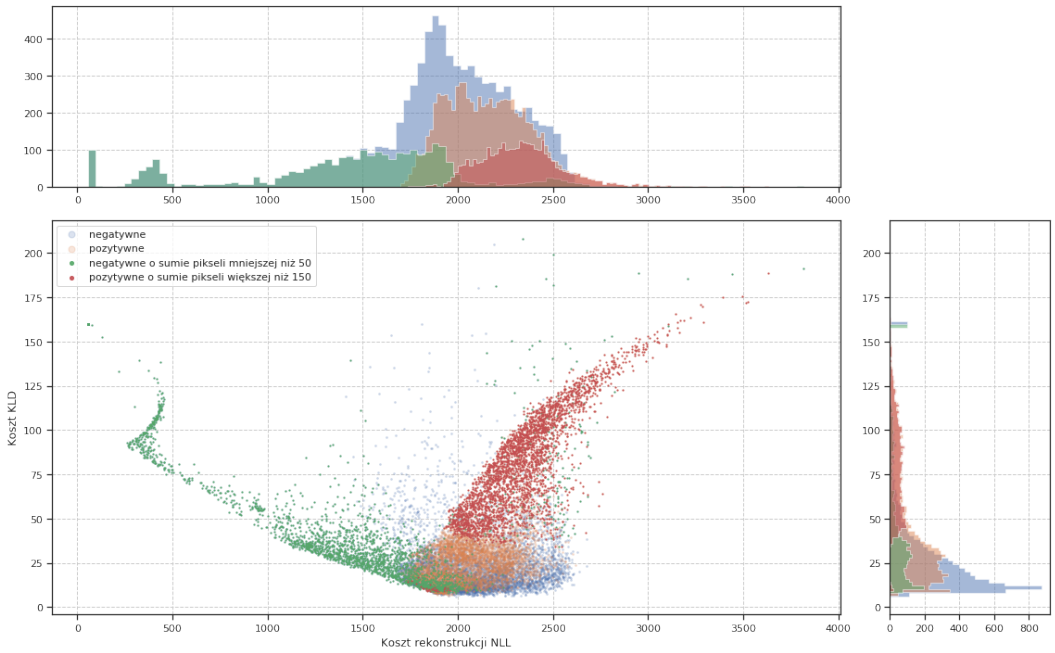
Chciałbym w tym miejscu wrócić to dwóch grup punktów, które się dobrze separowały i sprawdzić jaka ich własność mogła na to wpłynąć. Prostą obserwacją z poprzedniej analizy danych jest to, że obrazki dotyczące komórek nowotworowych są z reguły jaśniejsze od tych dotyczących zdrowych. Jasność obrazka będę utożsamiał z sumą pikseli. Niska wartość będzie oznaczać dla mnie ciemne obrazki, a wysoka jasne. Na wykresie 3.6 znajduje się histogram dla sumy piksli zdjęć z zestawu testowego.

Można zauważyć, że występują dwa progi, które w łatwy sposób klasyfikują sporą część danych z dużą dokładnością. Jest to odpowiednio 50.0 dla obrazków ne-



Rysunek 3.6: Rozkład sum pikseli dla obrazków ze zbioru testowego

gatywnych i 150.0 dla pozytywnych. Znalazłem obrazki, które można zidentyfikować w ten sposób i zaznaczyłem ich położenie na oryginalnym wykresie. W ten sposób otrzymałem wykres 3.7.



Rysunek 3.7: Zaznaczenie obrazków o konkretnych sumach pikseli

Widać, że są to dokładnie te same grupy danych, które dobrze się separowały. Wniosek z tego jest taki, że w wypadku modelu uczonego prawie jedynie na obrazkach bez zmian nowotworowych, ciemne próbki są bardzo prawdopodobne, a jasne już nie. Model nie dał rady znaleźć innych cech, na podstawie których można byłoby rozróżniać te dwie grupy.

Rozdział 4.

Podsumowanie

4.1. Wnioski

Autoenkoder wariacyjny jest bardzo interesującym modelem bazującym na rachunku prawdopodobieństwa, który stara się je oddolnie oszacowywać dla wystąpienia danego zjawiska. Jest to przydatne w problemie znajdywania odchyleń w danych, co pokazałem na bazie prostego zbioru jakim jest MNIST. Niestety przy przejściu do bardziej skomplikowanych próbek, model ten nie był w stanie znaleźć na tyle znaczących cech, które byłyby wystarczające do wykrywania zaburzeń z zadowalającą precyzją. Może być kilka przyczyn wystąpienia tego zjawiska, zaczynając od zbyt słabego modelu jakim jest sam autoenkoder po niewystarczającą obróbkę danych.

4.2. Usprawnienia

Skoro wyszło, że model zwraca przede wszystkim uwagę na jasność obrazków, to można byłoby zastosować normalizację w postaci wyrównywania histogramu (ang. histogram equalization). Metoda ta pozwala na zwiększenie kontrastu próbki, a dodatkowo rozciąga występowanie piksli na całą przestrzeń przez co zmienia się ich łączna suma wartości. Można wyobrazić sobie, że jasne obrazki zrobią się ciemniejsze i odwrotnie w przeciwnym przypadku. Jest szansa, że zmusiło by to model to położenia nacisku na inne cechy.

Innym pomysłem mogłoby być usprawnienie obróbki danych poprzez ograniczenie się jedynie do najbardziej znaczącej zawartości czaszki jakim jest mózg. W danych na jednych obrazkach znajdują się dodatkowo oczy, a na innych nie. Są to jasne obiekty, co mogło wpłynąć na reprezentację danych. Dodatkowo sama czaszka w formie kości też jest rzadka, więc usunięcie takich nieistotnych danych pozwoliłoby na sensowne ograniczenie danych.

Problemem może być sam rozmiar wycinka i brak jakichkolwiek dodatkowych

informacji. W tym momencie model tylko na bazie samego obrazka musi go dobrze zrekonstruować, co wydaje się bardzo trudne szczególnie w tych partiach przy krawędziach, o których ma najmniej danych. Rozwiązaniem mogłoby być dorzucenie dodatkowych informacji do dekodera o sąsiedztwie takiego wycinka, co mogłoby pozytywnie wpłynąć na koszt rekonstrukcji.

Bibliografia

[1] Diederik P. Kingma, Max Welling, *Auto-Encoding Variational Bayes*, ArXiv: 1312.6114, 2014.

[2] Carl Doersch, *Tutorial on Variational Autoencoders*, ArXiv: 1606.05908, 2016.

[3] Xianxu Hou, Linlin Shen, Ke Sun, Guoping Qiu, *Deep Feature Consistent Variational Autoencoder*, ArXiv: 1610.00291, 2016.

[4] David M. Blei, Alp Kucukelbir, Jon D. McAuliffe *Variational Inference: A Review for Statisticians*, ArXiv: 1601.00670, 2018.

[5] Jaan Altosaar, *Tutorial - What is a variational autoencoder?*, `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`, 01.02.2019.

[6] Volodymyr Kuleshov, Stefano Ermon *The variational auto-encoder*, `https://ermongroup.github.io/cs228-notes/extras/vae/`, 01.02.2019.

[7] Xitong Yang *Understanding the Variational Lower Bound*, `https://xyang35.github.io/2017/04/14/variational-lower-bound/`, 01.02.2019.

[8] *Classification: ROC Curve and AUC* , `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc`, 01.02.2019.