



# **Zastosowanie auto-encoderów wariacyjnych do rozpoznawania zmian na obrazach medycznych**

(English title)

Tomasz Nanowski

Praca inżynierska

**Promotor:** dr Jan Chorowski

Uniwersytet Wrocławski  
Wydział Matematyki i Informatyki  
Instytut Informatyki

27 stycznia 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 auto-encoderów 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, 27 stycznia 2019

(czytelny podpis)



## Streszczenie

Celem projektu jest wykorzystanie modelu VAE do wykrywania zmian patologicznych w obrazach medycznych

---

...









# Spis treści

<b>1. Introduction</b>	<b>11</b>
<b>2. Artificial neural networks</b>	<b>13</b>
2.1. Autoencoders . . . . .	13
2.2. Variational autoencoder . . . . .	14
2.2.1. Magia, która za tym stoi . . . . .	16
2.3. Deep feature consistent variational auto-encoder . . . . .	17
<b>3. Experiments on MNIST</b>	<b>19</b>
3.1. MNIST . . . . .	19
3.2. VAE . . . . .	19
3.2.1. Krzywa ROC i AUC . . . . .	22
3.3. Deep feature consistent variational auto-encoder . . . . .	22
<b>4. Experiments on MRI FLAIR images</b>	<b>25</b>
4.1. Data description & preprocessing . . . . .	25
4.1.1. Overview . . . . .	25
4.1.2. Patches . . . . .	26
4.1.3. Preprocessing . . . . .	27
4.1.4. Normalizations . . . . .	28
4.2. Results . . . . .	28
4.3. Analysis . . . . .	28
4.4. Back to basic models . . . . .	30
<b>5. Summary</b>	<b>33</b>





# Notes

	Dokończyć matkę . . . . .	16
	Uzupełnić podpisy . . . . .	26
	Może LIME? . . . . .	28
	Rozważyć normalizacje: histogramowa, tylko środkowy kanał . . . . .	28
	dodać krótki opis modeli i może dla czego softmax . . . . .	28
	Dodać z-dim . . . . .	28



# Rozdział 1.

## Introduction

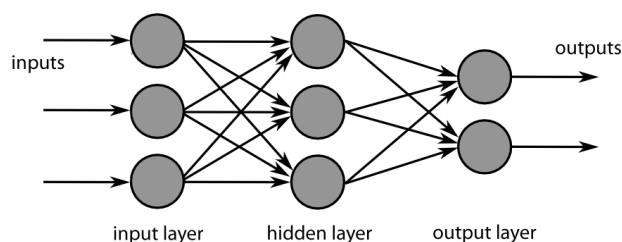
Czy wykorzystanie sztucznych sieci neuronowych może pomóc w diagnozowaniu nowotworów? W tej pracy będę chciał przedstawić podejście do tego problemu z wykorzystaniem wariacyjnego auto-enkodera. Dane na których będę bazował pochodzą z uniwersytetu Duke i przedstawiają zdjęcia rezonansowe głowy pacjentów z wykrytym nowotworem. I tu pojawia się jeden z pierwszych problemów, jakim jest niezbalansowanie danych. W ogólności liczba osób chorych jest zdecydowanie mniejsza od tych zdrowych, a dodatkowo samych komórek nowotworowych też jest mniej w porównaniu do pozostałych. W tym ma właśnie pomóc VAE, który stara się oszacować prawdopodobieństwo wystąpienia danej próbki, a skoro danych zmutowanych jest mniej, to ich prawdopodobieństwo również powinno być niższe. Będzie to jedna z podstaw użyta do klasyfikowania występowania raka.



## Rozdział 2.

# Artificial neural networks

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 SGD, a przy wybranej strukturze można w efektywny sposób zastosować algorytm propagacji wstecznej. W dalszej części pracy będę używał prostszej nazwy (neural nets). Przykładowa architektura sieci neuronowych jest zaprezentowana na wykresie 2.1.



Rysunek 2.1: Exemplary architecture of the basic neural network

### 2.1. Autoencoders

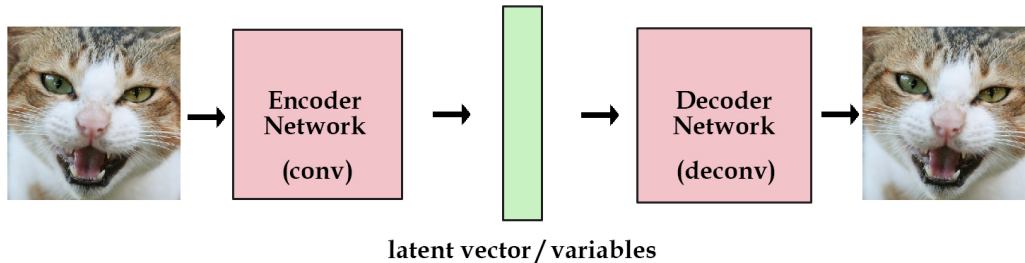
Jest to jeden z rodzajów sieci neuronowych, służący do znajdowania wydajnej reprezentacji danych, co jest przykładem nauki bez nadzoru. W autoencoder'ach można wyróżnić dwie połączone ze sobą części: encoder i decoder. Zadaniem encodera jest wyprodukowanie reprezentacji, natomiast decoder służy do odtworzenia

z niej oryginalnej postaci. Zależy nam na tym, żeby wyjście było w jakimś sensie jak najbardziej podobne do wejścia. W przypadku obrazów jako funkcja straty często stosowany jest błąd średnio kwadratowy (MSE).  $Y_i$  oznacza oryginalne dane, natomiast  $\hat{Y}_i$  zrekonstruowane.

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

Przykładowy schemat znajduje się na wykresie 2.2. Zdjęcie kota zostaje włożone do encodera, a on znajduje jego liczbową reprezentację (czyli wektor o jakiejś ustalonej długości). Następnie tą reprezentacją karmiony jest dekodery, który ma za zadanie jak najlepiej odtworzyć oryginał. Chcemy mieć nadzieję, że taki model nauczy się dobrze układać wysoko wymiarowe dane w mniejszej przestrzeni, gdzie podobne do siebie obiekty będą blisko siebie.

Często autoenkodery po wytrenowaniu nadają się do generowania całkiem nowych, nigdy jeszcze nie widzianych danych, ale podobnych do tych z zestawu uczącego. Można albo całkowicie wylosować wektor reprezentacji, albo delikatnie zmodyfikować dla już istniejącego przykładu. Dzięki temu możemy uzyskać realistycznie wyglądający obiekt o np. innym kolorze, kształcie.



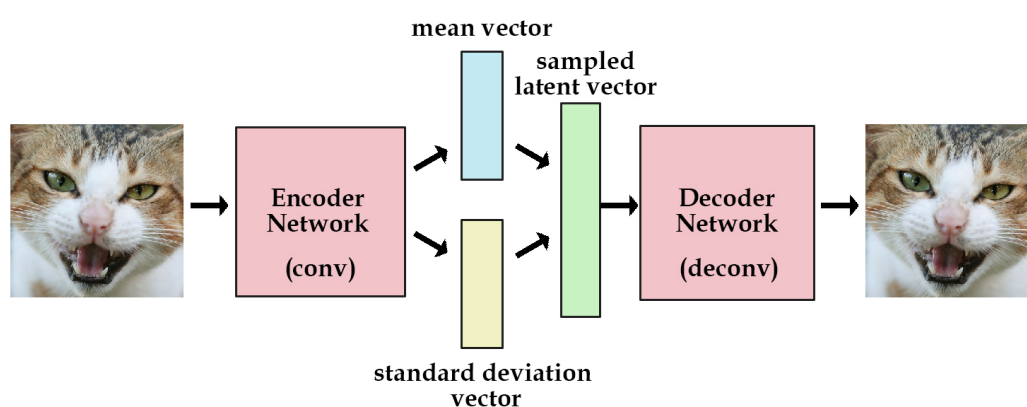
Rysunek 2.2: Architecture of autoencoder

## 2.2. Variational autoencoder

W przypadku podstawowych autoenkoderów jednym z problemów jest brak kontroli nad tym jak dane są rozrzucone w przestrzeni. O ile sam model może bardzo dobrze rekonstruować dane, to same ich reprezentacja w przestrzeni może nie być ciągła, tj. pewne obszary nigdy nie były używane i dekodery nie za bardzo wie co zrobić, jeśli wylosujemy wektor z tego regionu. Jest to problemem przy generowaniu nowych danych, bądź interpolowaniu między nimi. W innych natomiast miejscach dane mogą zostać poukładane zbyt ciasno, co skutkować będzie błędną rekonstrukcją na niewidzianych danych. Próba rozwiązania tych problemów jest właśnie Variational Autoencoder.

Variational autoencoder rozszerza założenia o wprowadzenie modelowania rozkładu prawdopodobieństwa dla reprezentacji ukrytej. Czyli teraz encoder nie produkuje pojedynczego wektora, ale dwa wektory interpretowane odpowiednio jako średnia  $\mu$  oraz wariancja  $\sigma$  dla rozkładu Gaussa. Następnie z tych parametrów formowany jest wektor, gdzie  $i$ -ty element powstał z  $i$ -tej pary  $\mu, \sigma$ . Dopiero taki wektor trafia do decodera, a z niego rekonstruowana jest wejściowa dana. Schemat przedstawiony jest na rysunku 3.2

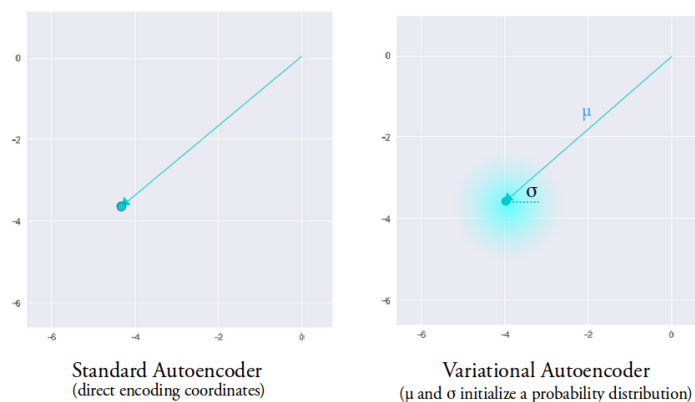
Takie generowanie znaczy, że nawet dla tego samego wejścia i podczas gdy średnie oraz odchylenie zostaną takie same, to rzeczywiste kodowanie będzie się mimo wszystko różnić właśnie ze względu na to próbkowanie.



Rysunek 2.3: Architecture of Variational Autoencoder

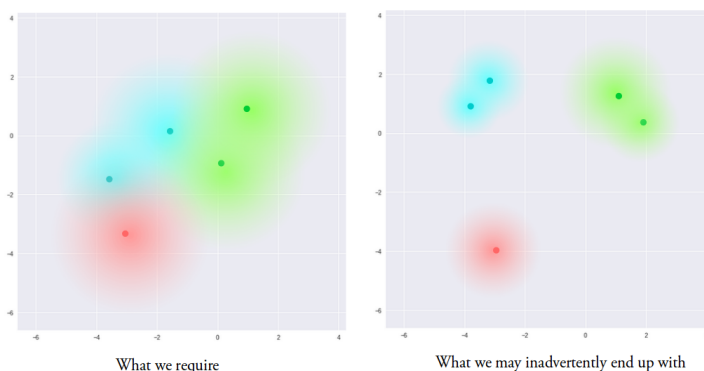
Intuicyjnie wektor średnich oznacza, gdzie zakodowana zmienna powinna być wycentrowana, natomiast odchylenie kontroluje obszar, wokół którego kodowanie może się różnić. Podczas gdy kodowanie jest losowane z wnętrza koła, dekodery uczy się, że nie jedynie pojedynczy punkt z utajonej przestrzeni odnosi się do próbki, ale wszystkie punkty znajdujące się w pobliżu odnoszą się również do tego samego. To pozwala dekodownikowi dekodować nie tylko pojedyncze, specyficzne reprezentacje (zostawiając przestrzeń nieciągłą), ale też te trochę się różniące, ponieważ narażony jest na szereg zmian kodowania tego samego wejścia podczas treningu. Rysunek 2.4

Model jest teraz wystawiony na pewien stopień lokalnej zmienności przez zmianę kodowania pojedynczej próbki, co skutkuje ciągłą przestrzenią w reprezentacji ukrytej dla lokalnego sąsiedztwa. Jednak na razie nie ma ograniczeń dla wartości jakie mogą przyjmować wektory  $\mu$  i  $\sigma$ , przez co enkoder może nauczyć się generować bardzo różne  $\mu$  dla różnych klas i grupować je minimalizując  $\sigma$ , upewniając się, że samo kodowanie nie różni się zbyt wiele dla tej samej próbki. W szczególności  $\sigma$  może wynosić 0, przez co wrócimy do podstawowego modelu. To oczywiście pozwala dekodownikowi na skuteczną rekonstrukcję, ale jedynie danych treningowych. Przykład przedstawiony jest na rysunku 2.5. To co byśmy idealnie chcieli to kodowanie, gdzie



Rysunek 2.4

grupy są blisko siebie, ale mimo wszystko są dobrze separowalne. Pozwoliłoby nam to na gładką interpolację między próbkami i konstruowanie nowych danych.



Rysunek 2.5: Architecture of Variational Autoencoder

Żeby to osiągnąć wzbogacimy funkcję straty o koszt the Kullback–Leibler divergence. KL divergence mierzy po prostu różnicę pomiędzy dwoma rozkładami prawdopodobieństwa. Minimalizowanie go oznacza optymalizowanie parametrów rozkładu ( $\mu$  i  $\sigma$ ), tak żeby jak najbardziej przypominały celowy rozkład. W przypadku VAE będzie to standardowy rozkład normalny ( $\mu = 0$ ,  $\sigma = 1$ ).

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

### 2.2.1. Magia, która za tym stoi

To co do tej pory napisałem to były w większości intuicje i nasze zachcianki. Chciałbym teraz przedstawić jak to wygląda od strony matematycznej.

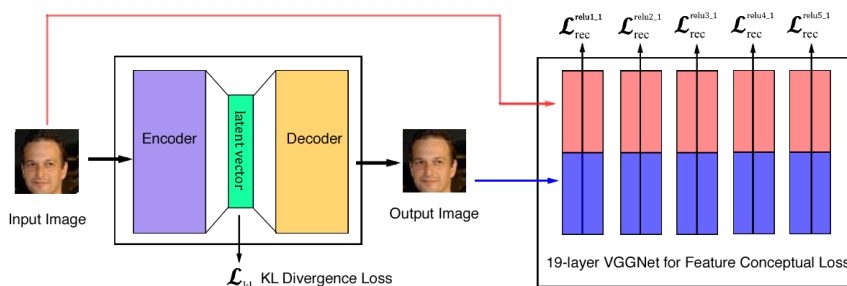
Dokończyć matkę



## 2.3. Deep feature consistent variational auto-encoder

W autoenkoderze wariacyjnym jednym ze składników kosztu jest błąd rekonstrukcji. Standardowo funkcja ta oparta jest na porównywaniu wartości odpowiadającym sobie pikselom z oryginalnego obrazu i tego zrekonstruowanego. Często przyczynia się to do uśredniania wartości pikseli przez co wyjściowy obraz nie jest wyraźny. Przyczynie się do tego między innymi fakt, że taki liczenie błędu jest bardzo czułe na chociażby przesunięcie. Oryginalny obraz i ten lekko przesunięty mimo, iż nie różnią się prawie wogóle perceptualnie dla człowieka, to ich koszt MSE jest bardzo duży. Uśrednianie wartości jest sposobem na optymalizowanie tego problemu. Wynika z tego, że raczej chcielibyśmy patrzeć na to co znajduje się na obrazkach, niż porównywać poszczególne piksele.

I w tym właśnie miejscu pojawia się Deep feature consistent. Opiera się ona na wykorzystaniu wcześniej już wyuczonej sieci splotowej. W takich sieciach coraz głębsze filtry odpowiadają za rozpoznawanie coraz to bardziej abstrakcyjnych cech. Będziemy teraz myśleć, że dwa obrazy są podobne, jeśli mają podobne wartości aktywacji w warstwach ukrytych tej sieci. Dzięki takiemu patrzeniu na różnice dwa obrazy powinny być podobne nawet jeśli zostaną poddane takim operacjom jak m.in przesunięcie, rotacja, co stanowi duży problem dla kosztów typu pixel-to-pixel. Błąd ten jest określany mianem perceptual loss.



Rysunek 2.6: Architecture of Variational Autoencoder



## Rozdział 3.

# Experiments on MNIST

Na początek chciałbym zacząć od odtworzenia docelowego problemu na prostszych danych jakim jest zbiór MNIST. Pozwoli to w ogólności stwierdzić sensowność skorzystania z wybranego modelu. Zakładam, że jeśli eksperyment nie powiódłby się na łatwiejszych danych, to raczej nie powiedzie się również na tych bardziej skomplikowanych. To miejsce traktuję jako poligon testowy. Pozwoli mi to też nabrać pewnego rodzaju doświadczenia z wybranym modelem.

Sytuację, którą będę chciał poddać testowi jest całkowite niebalansowanie danych, a wręcz brak danych drugiej kategorii i obserwacja jak zachowuje się w takim przypadku model. Będę chciał to osiągnąć poprzez uczenie modelu jedynie na danych pochodzących z dwóch klas [4, 7], a następnie obserwacja tego co się stanie z modelem przy zaaplikowania danych z klasy [5]. Ma to odtworzyć sytuację gdzie danych opisujących zmiany nowotworowe jest przytłaczająco mniej od tych zdrowych.

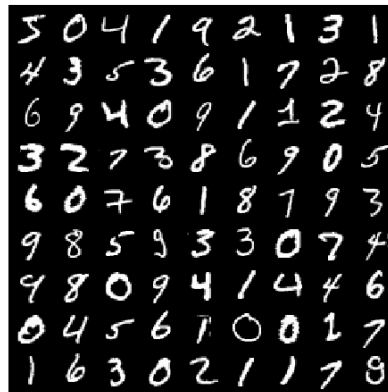
### 3.1. MNIST

Jest to zbiór po kategoryzowanych odręcznie napisanych cyfr. Wszystkie obrazki są czarno-białe, rozmiaru 28x28 i wycentrowane. Każdy piksel ma wartość dualną: 0 albo 1. Zbiór składa się z 60000 danych treningowych i 10000 testowych. Przykładowe obrazki 3.1.

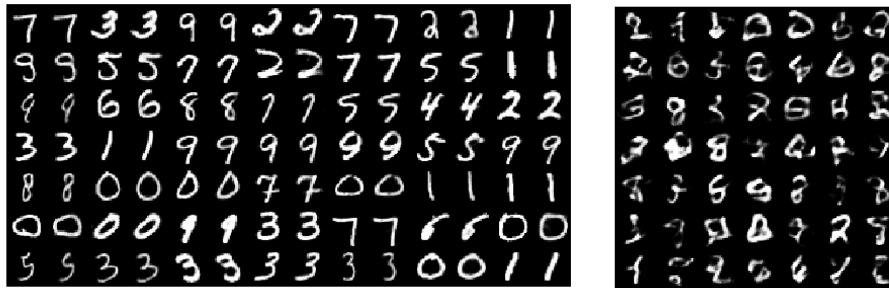
### 3.2. VAE

Na rysunku 3.2 znajduje się efekt wyuczenia modelu VAE z warstwą ukrytą rozmiaru 20. Po lewej widać rekonstrukcje, a po prawej efekty zdekodowania wektora wygenerowanego ze standardowego rozkładu normalnego.

Dodatkowo warto byłoby zobaczyć jak konkretne cyfry rozrzucone są w przestrzeni. 20 wymiarów jest dosyć trudne do zwizualizowania, więc wyuczyłem model



Rysunek 3.1: Samples from MNIST dataset



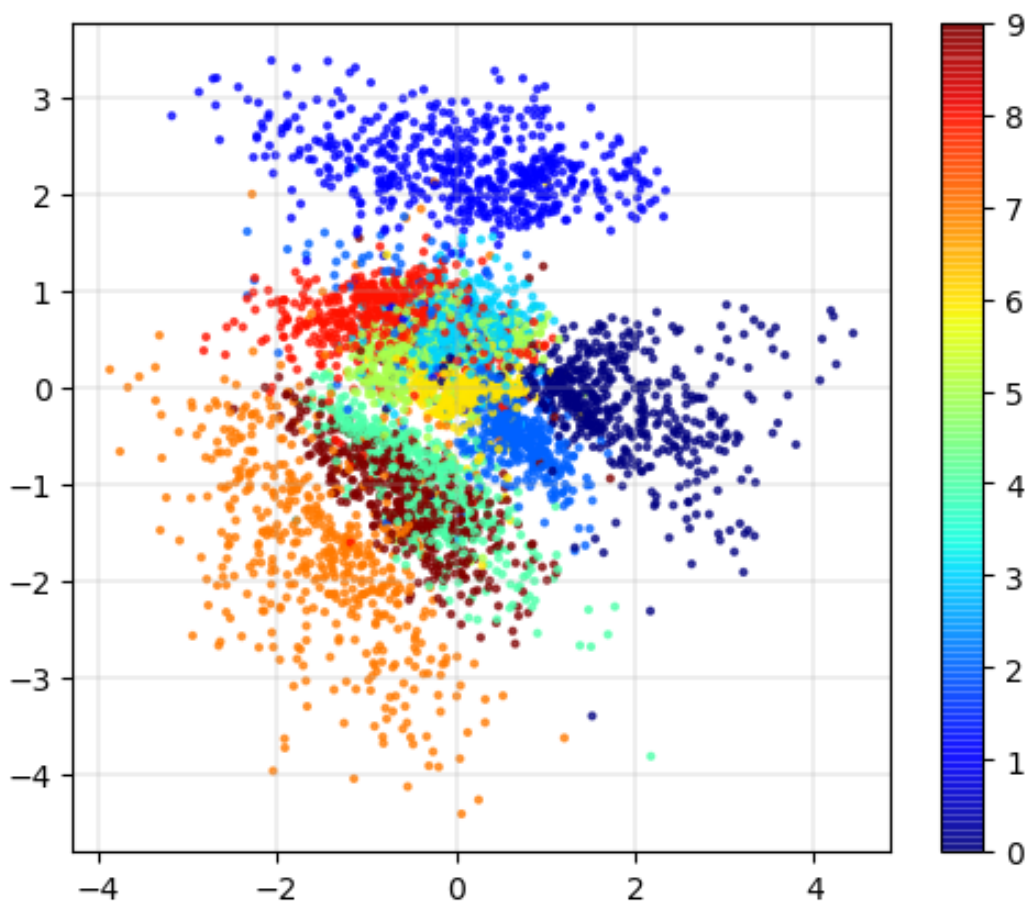
(a) W nieparzystych kolumnach znajdują się oryginalne obrazy, natomiast po prawej ich rekonstrukcje

Rysunek 3.2: Przedstawienie efektów modelu VAE

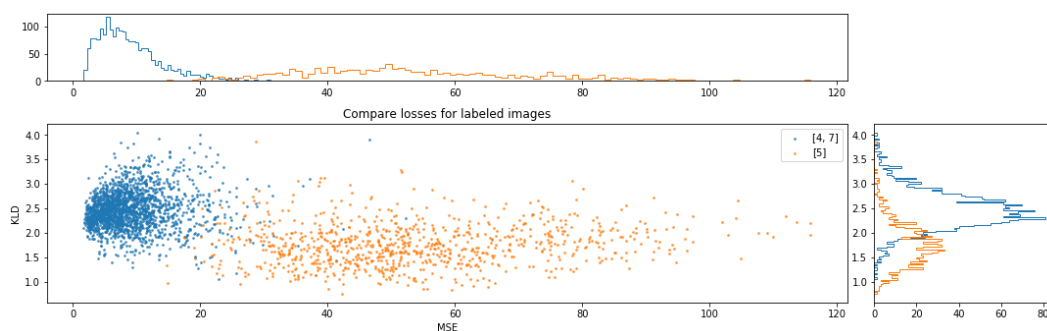
dla reprezentacji ukrytej rozmiaru 2. Na rysunku 3.3 znajdują się wyniki. Warto odnotować fakt, że niektóre klasy są bardzo dobrze separowalne. Widać tu właśnie opisane wcześniej zalety modelu VAE. Te ciekawą własność wyniku reprezentacji będę chciał wykorzystać w późniejszej analizie.

W oryginalnym problemie mamy bardzo duży dysonans pomiędzy ilością przykładów dla każdej z klas. Według statystyk dane z komórkami rakowymi stanowią 1% wszystkich. Ciężko jest więc nawet mówić o jakimś sensownym podejściu supervised. Do zasymulowania tego problemu dla MNIST będę uczył model jedynie na dwóch klasach [4, 7], a następnie testował zachowanie reprezentacji ukrytej dla przykładów z klasy 5.

Rozmiar reprezentacji ukrytej wynosi 10. Analizować natomiast będziemy 2 składowe koszty dla modelu VAE: KLD oraz błąd rekonstrukcji MSE. Na rysunku 3.4 znajduje się przedstawienie ich wraz z histogramami. Można zauważyć, że wyłącznie na podstawie samego błędu rekonstrukcji można z bardzo dużą dokładnością określić dane pochodzące z klasy 5, mimo iż model nie widział żadnych ich przykładów podczas uczenia.

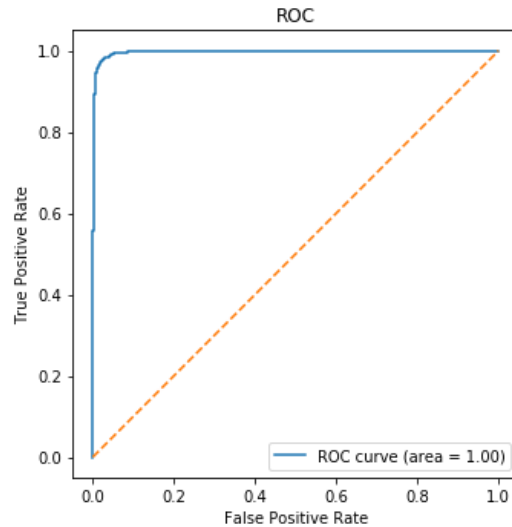


Rysunek 3.3



Rysunek 3.4

Do określenia jak rzeczywiście dobra jest ta separacja można wykorzystać krzywą ROC. Traktujemy to jako problem binarnej klasyfikacji, gdzie dane z klasy 5 będą oznaczone jako 1, a z [4, 7] jako 0. Wartość confidence to suma kosztów KLD i MSE. Jak widać na rysunku 3.5 takie podejście osiąga prawie 100% skuteczność. Podobne podejście będę chciał zastosować do danych medycznych.



Rysunek 3.5

### 3.2.1. Krzywa ROC i AUC

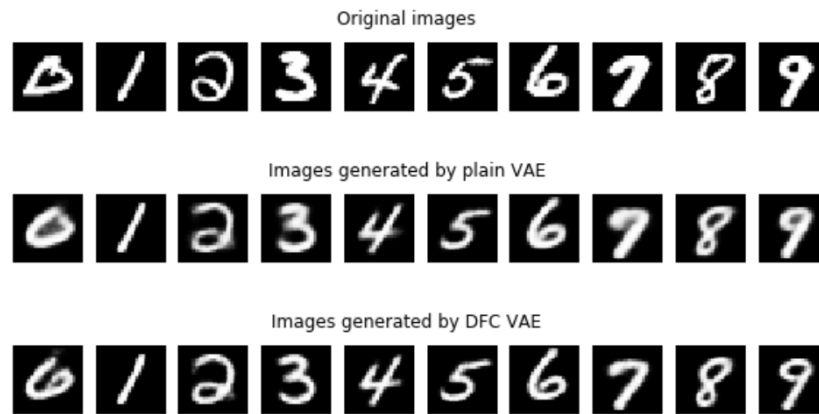
Krzywa ROC (Receiver Operating Characteristic) jest narzędziem do oceny poprawności klasyfikatora binarnego. Bazuje ona na wyliczaniu charakterystyki jakościowej modelu predykcji w wielu różnych punktach odcięcia. Z praktycznego punktu widzenia działa to na takiej zasadzie, że mamy dane pochodzące z dwóch klas. Model odpowiada z jaką pewnością dane należą do klasy 1. Następnie badamy różne progi i klasyfikujemy obiekty (jeśli przewidziana wartość jest powyżej progu to 1 wpp 0). Dla uzyskanych klasyfikacji liczymy TPR oraz 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 w przy każdym progu połowa danych będzie nad i połowa pod. Idealny model znajduje się w punkcie (0, 1).

Przydatne jest również obliczenie pola powierzchni pod krzywą AUC (Area Under Curve). Interpretuje się ją jako prawdopodobieństwo, że badany model predykcyjny oceni wyżej losowy element klasy pozytywnej od losowego elementu klasy negatywnej.

## 3.3. Deep feature consistent variational auto-encoder

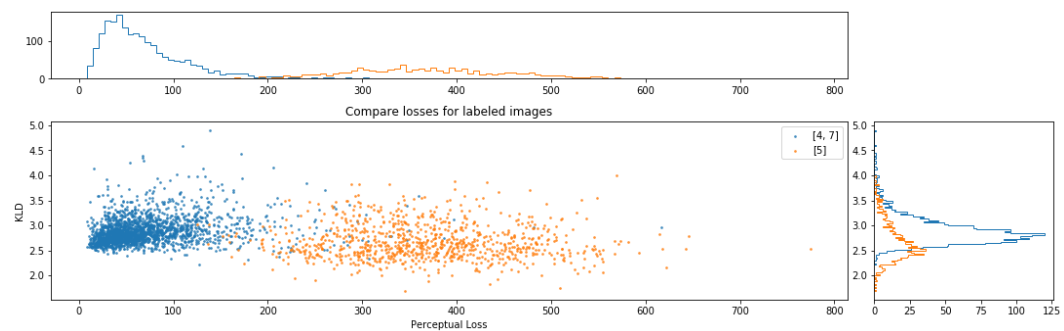
Podobny eksperyment jw. przeprowadziłem również dla modelu DFC. Na początku jednak warto zobaczyć co zyskujemy na zmianie podejścia do kosztu rekonstrukcji. Różnice prezentują się na rysunku 3.6. Widać, że rekonstrukcje są mniej rozmazane niż przy standardowym VAE. Dodatkowo lepiej rekonstruuje takie elementy jak np. pozioma kreska w cyfrze 7.

Na rysunku 3.7 przedstawiony jest efekt przeprowadzenia analogicznego eks-



Rysunek 3.6

perymentu z nauką na jedynie przykładach z klas [4, 7] i sprawdzeniu zachowania dla danych z klasy 5. Zamiast kosztu rekonstrukcji MSE używamy błędu perceptualnego. Dodatkowy model splotowy również nie widział danych pochodzących z 5 klasy. Jak widać separacja jest co najmniej tak dobra jak w przypadku zwykłego VAE.



Rysunek 3.7





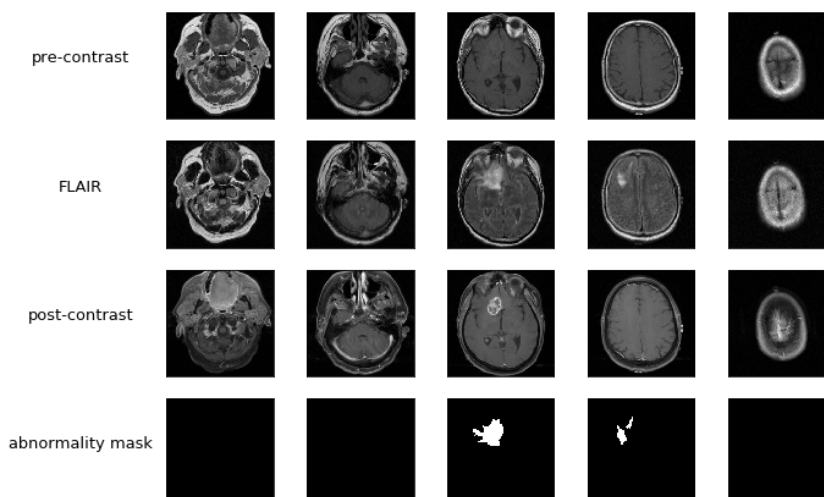
## Rozdział 4.

# Experiments on MRI FLAIR images

### 4.1. Data description & preprocessing

#### 4.1.1. Overview

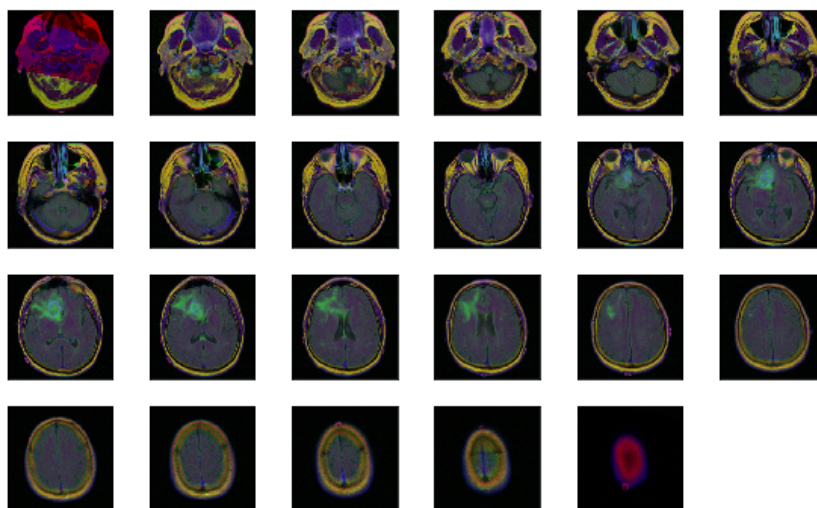
Dane pochodzą z Duke University. Są to zdjęcia z rezonansu magnetycznego głowy wraz z zaznaczonymi obszarami zmian nowotworowych. Obrazy są rozmiaru 256x256x3. Pierwszy kanał jest zdjęciem wykonanym przed wprowadzeniem kontrastu, drugi w trakcie, a trzeci po. Maski mają rozmiar 256x256, a każdy piksel ma wartość 0 albo 255 (255 oznacza komórkę nowotworową). Na rysunku 4.1 pokazane są osobne kanały oraz maska.



Rysunek 4.1

Dane pogrupowane są dla 110 pacjentów. Zestaw dla jednego z nich znajduje się na rysunku 4.2. Łącznie znajduje się 3929 par obrazów, z czego w 1373 jest przynaj-

mniej jedna komórka nowotworowa ( $\sim 34.95\%$ ). Natomiast w całym zbiorze piksele oznaczające zmiany nowotworowe stanowią jedynie  $\sim 1.02988\%$  wszystkich pikseli. Przy tak znaczącej dysproporcji ciężko liczyć na rzetelne podejście supervised, dlatego będę chciał spróbować podejścia unsupervised. Założeniem wykorzystania VAE będzie patrzenie na koszty. Hipotezą jest to, że obrazki z rakiem, czyli takie mało prawdopodobne będą zauważalne w rozkładzie i rekonstrukcji.



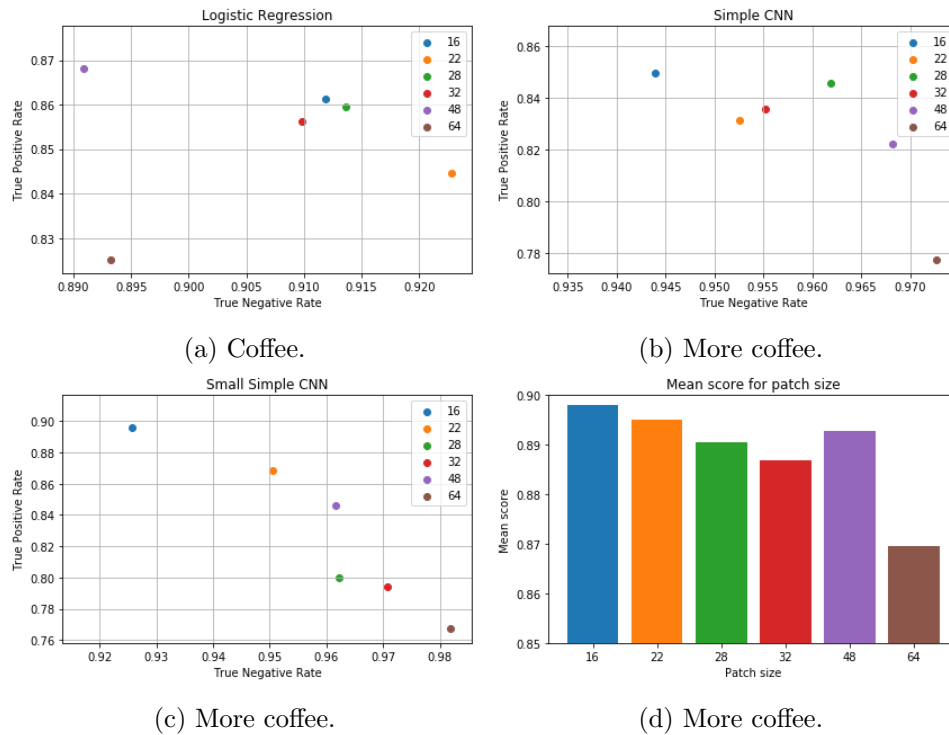
Rysunek 4.2

Pacjenci zostali podzieleni na 2 zbiory: treningowy (70%), testowy (30%). Obrazki oznaczające komórki rakowe będę nazywał pozytywnymi, a te bez negatywnymi.

#### 4.1.2. Patches

Rekonstrukcje nie chcemy robić od razu z całego obrazku, a jedynie z niewielkich wycinków. Rozmiar wycinków ustalę na podstawie wyników osiągniętych przez modele supervised, tj. regresja liniowa, małe cnn (mnist), cnn (cifar10). Będę testował rozmiary 16, 22, 28, 32, 48, 64.

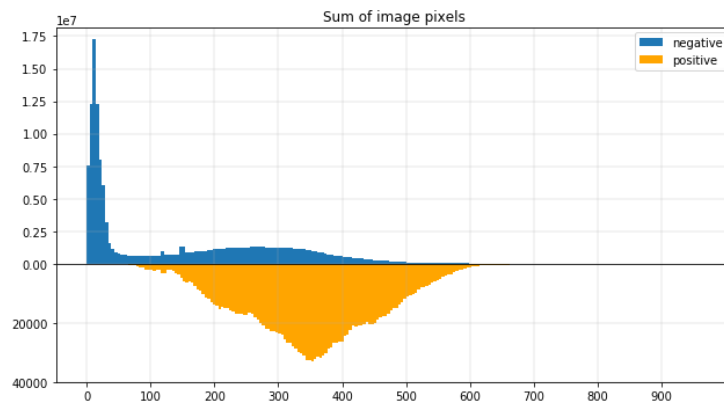
Na obrazku 4.3 widać wyniki dla poszczególnych modeli i rozmiarów. Patrzymy na 2 wartości true positive rate i true negative rate. Najlepszy przypadek jest w takim razie w punkcie (1, 1). Obrazek (d) prezentuje średnie wyniki dla każdego z rozmiarów. Przy takim liczeniu najlepsze wyniki wyszły dla rozmiarów odpowiednio 16 i 22. W dalszych eksperymentach będę korzystał z pacy o rozmiarze 22 ze względu na trochę większe pole obserwacji.



Rysunek 4.3: The same cup of coffee. Two times.

### 4.1.3. Preprocessing

Na obrazku 4.4 przedstawiony jest histogram zsumowanych wartości pikseli dla każdego patchu rozmiaru 22 z podziałem na klasy. Minimalna wartość sumy dla klasy pozytywnej wynosi 56.5. Oznacza to, że ze zbioru treningowego można usunąć obrazki o mniejszej sumie, gdyż automatycznie są to negatywne obrazki. Usunięcie obrazków o wartości mniejszej niż 42 zmniejszy dane o 45.5% i usunięcie trywialnych przypadków czyli prawie całych czarnych obrazków.



Rysunek 4.4

#### 4.1.4. Normalizations

Rozważyć normalizacje: histogramowa, tylko środkowy kanał

### 4.2. Results

Przetestuję 4 kombinacje modeli, ze zmienioną funkcją kosztu

W tabeli 4.1 zostały przedstawione wyniki dla czterech kombinacji modeli oraz parametrów. Wartości liczbowe opisują najlepsze AUC ROC podczas uczenia wraz z odpowiadającą epoką. Jak widać w tym zestawieniu najlepiej wypadły podstawowe modele VAE.

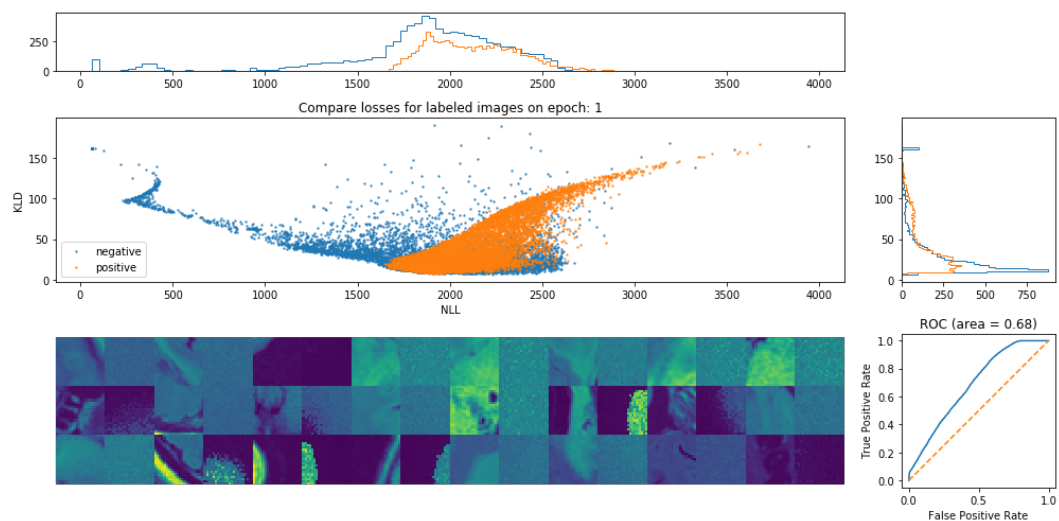
Model	Best ROC AUC
VAE	0.68 (1)
VAE + Softmax	0.68 (1)
C-VAE	0.59 (1)
C-VAE + Softmax	0.67 (1)

Tablica 4.1: My table

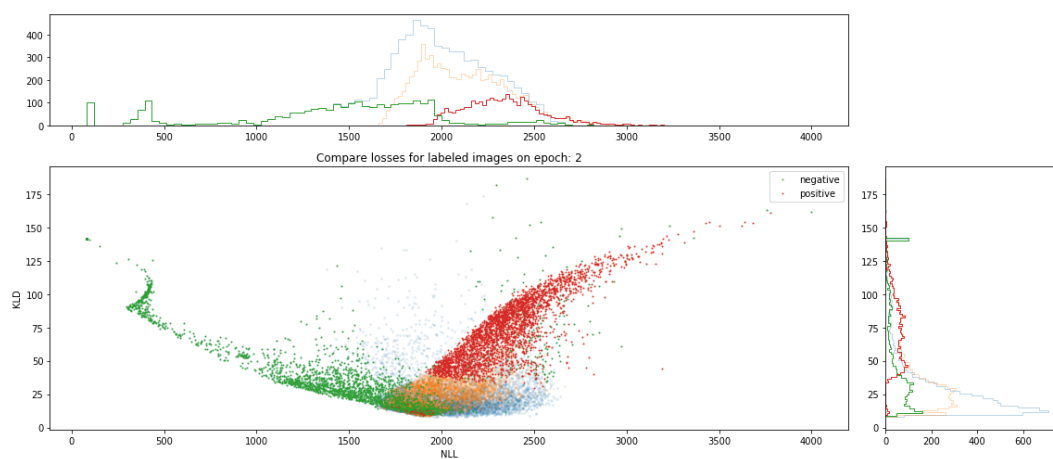
### 4.3. Analysis

Zajmę się analizą modelu w wersji VAE + Softmax. Na wykresie 4.5 przedstawione są koszty, ich rozkłady, przykłady rekonstrukcji oraz krzywa ROC. Ciekawie wyglądają te dwa zauważalne ogony na wykresie. Postaram się lepiej przyjrzeć ich specyfice.

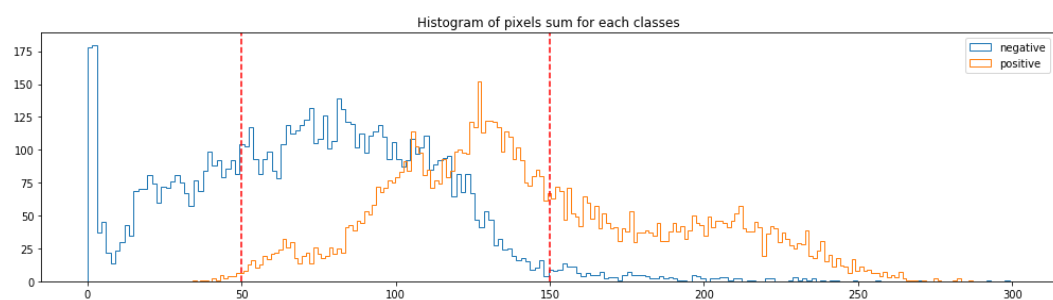
Pierwszym pomysłem jest, żeby sprawdzić rozkład obrazków w zależności od ich jasności, czyli łącznej sumy pikseli. Ciemne obrazki definiuję jako te o niskiej sumie, a jasne o wysokiej. Na rysunku 4.6 zaznaczyłem negatywne obrazki, dla których suma  $\mu = 50$  oraz pozytywne obrazki z sumą co najmniej 150. Zaznaczone są one odpowiednio zielonym i czerwonym kolorem na wykresie. Jak widać po rozkładach składają się one na dokładnie te 2 separowalne ogony. Wniosek z tego jest następujący. Na podstawie tego modelu możemy odesparować jedynie najciemniejsze i najjaśniejsze obrazki z obydwu klas. W takim razie model nie robi nic nadzwyczajnego. Wystarczy przyjrzeć się histogramowi na rysunku 4.7 z zaznaczonymi progami. One już bardzo dobrze separują te klasy.



Rysunek 4.5

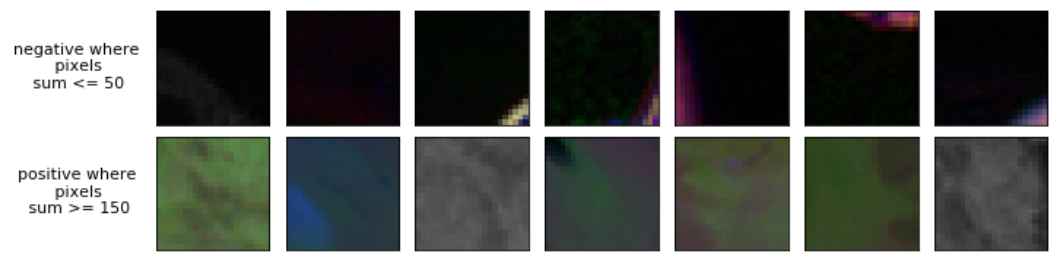


Rysunek 4.6



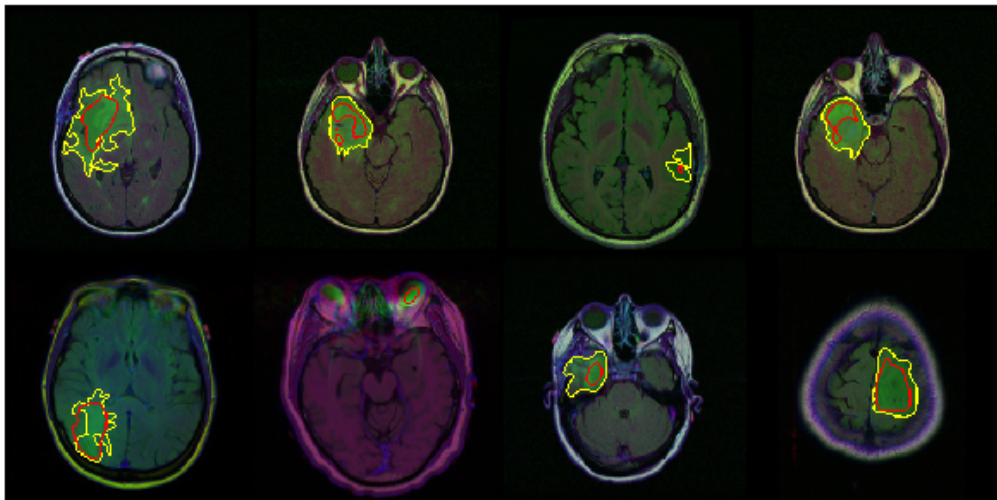
Rysunek 4.7

Na rysunku 4.8 pokazane są przykłady obrazków z zadanymi poziomami sumy pikseli.

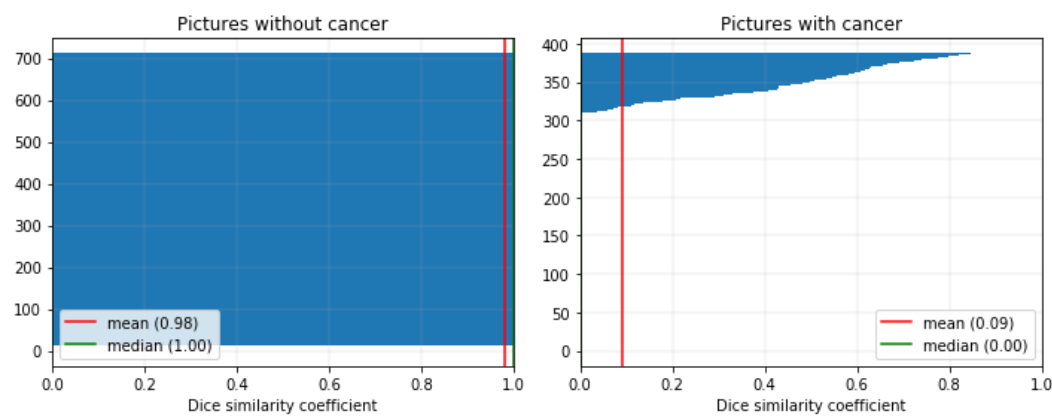


Rysunek 4.8

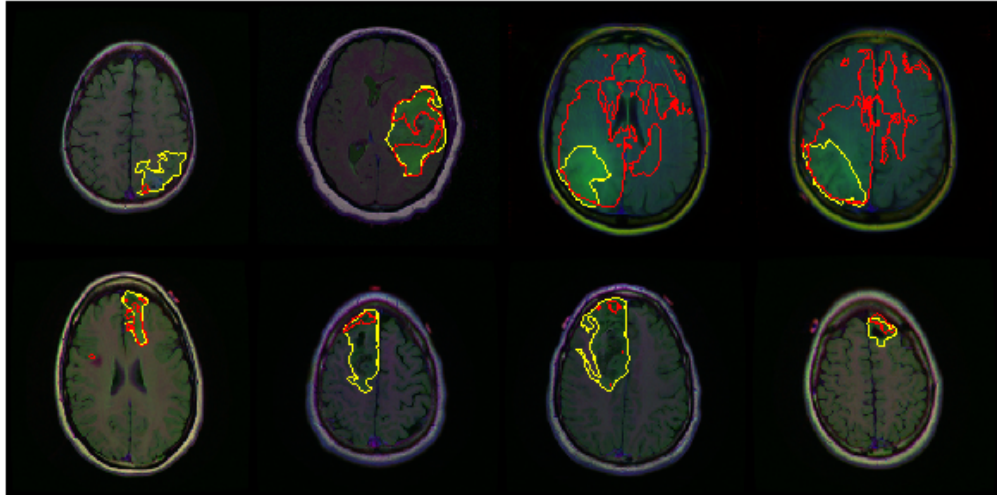
4.4. Back to basic models



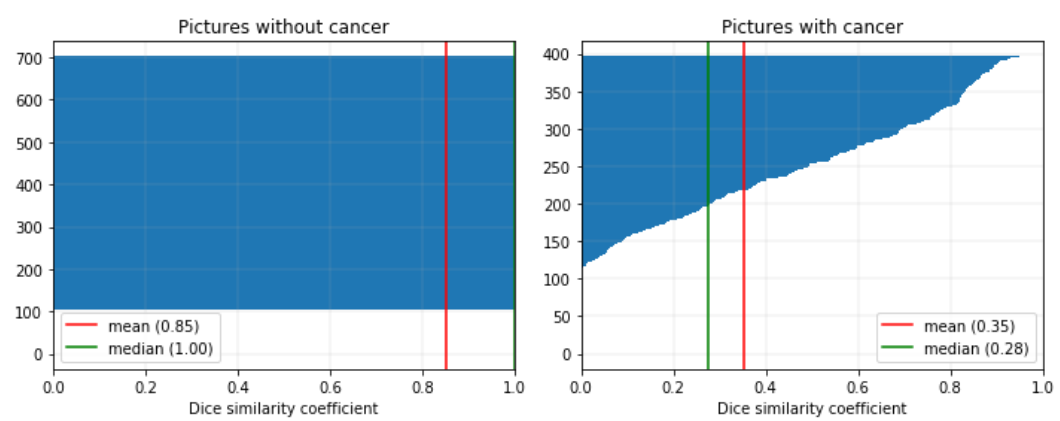
Rysunek 4.9



Rysunek 4.10



Rysunek 4.11



Rysunek 4.12





## Rozdział 5.

## Summary