

Novelty Detection in Mars Rover Curiosity Images

Piotr Durniat, Tomasz Hałas, Patryk Marciniaak

Czerwiec 2024

Spis treści

1	Opis projektu	3
2	BiGAN	4
2.1	Architektura BiGAN	4
2.2	BiGAN Komponenty	5
2.3	Dobór hiperparametrów	5
2.4	Detekcja anomalii	5
2.5	Struktura modelu	6
2.6	Wyniki	6
2.7	Wnioski	8
3	Autokoder Wariacyjny (VAE)	9
3.1	Architektura VAE	9
3.1.1	Enkoder	9
3.1.2	Dekoder	9
3.1.3	Klasa Variational Autoencoder (VAE)	10
3.2	Dobór hiperparametrów	10
3.3	Trenowanie modelu	11
3.4	Metoda klasyfikacji nowości	12
3.5	Wyniki klasyfikacji	12
4	Normalizing Flows (NF)	13
4.1	Przykłady Normalizing Flows	14
4.2	Masked Autoregressive Flow (MAF)	14
4.3	Wyniki trenowania	15
4.4	Metoda klasyfikacji nowości	17
5	Wnioski	17

1 Opis projektu

Celem niniejszej pracy jest wykrywanie anomalii na zdjęciach oraz identyfikacja wzorców w danych, które wcześniej nie były obserwowane na zdjęciach powierzchni Marsa wykonanych przez kamerę Mastcam zamontowaną na maszcie łazika Curiosity. Szczegółowe zestawienie liczby typowych zdjęć oraz anomalii w naszym zbiorze danych przedstawiono w tabeli 1.

Zbiór danych	Liczba typowych zdjęć	Liczba anomalii
Treningowy	9302	0
Walidacyjny	1386	0
Testowy	426	430

Tabela 1: Liczba typowych zdjęć i anomalii w zbiorze danych [2]

Powodem stworzenia takich zestawów danych jest fakt, że zespoły naukowe zajmujące się misjami eksploracyjnymi opartymi na łazikach planetarnych, takimi jak łazik Mars Science Laboratory Curiosity, mają ograniczony czas na analizę nowych danych przed podjęciem decyzji o dalszych obserwacjach. Dlatego istnieje potrzeba systemów, które mogą szybko i inteligentnie wyodrębniać informacje z danych pochodzących z instrumentów planetarnych i skupiać uwagę na najbardziej obiecujących lub nowatorskich obserwacjach. Aby udoskonalić proces uczenia, postanowiono stworzyć zestaw danych, który będzie składał się z danych multispektralnych. W naszym zbiorze testowym odnotowano następujące typy anomalii, jak przedstawiono w tabeli 3:

Podklasa	Liczba zdjęć
Miejsce DRT	111
Stos odpadów	93
Złamany kamień	76
Otwór wiertniczy	62
Meteoryt	34
Żyla	30
Pływający kamień	18
Skała macierzysta	11

Tabela 2: Liczba zdjęć dla różnych podklas anomalii [2].

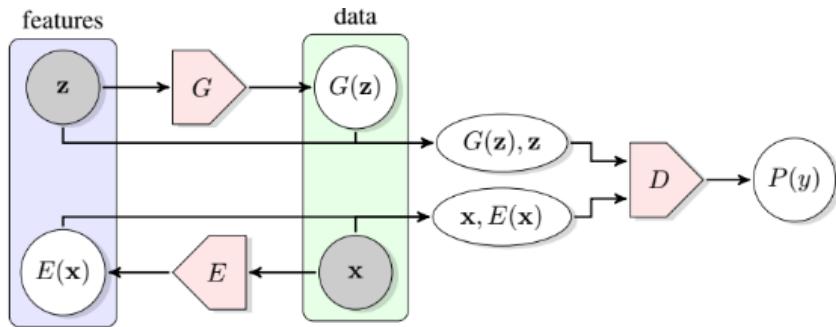


Rysunek 1: Zdjęcia anomalii z powierzchni Marsa. Źródło: [2]

Aby zwiększyć liczbę dostępnych obrazów typowych do treningu i walidacji, twórcy zbioru danych próbkowali zdjęcia o rozmiarze $64 \times 64 \times 6$ pikseli z obrazów w zestawach treningowych i walidacyjnych, stosując przesuwne okno z krokiem co 16 pikseli. Dało to łącznie 9302 typowe zdjęcia treningowe i 1386 typowe zdjęcia walidacyjne. Aby przygotować model, obrazy zostały przeskalowane w każdym kanale do wartości od -1 do 1, co pozwoliło modelom na lepsze uczenie się.

2 BiGAN

2.1 Architektura BiGAN



Rysunek 2: BiGAN struktura

W tradycyjnym modelu GAN mamy dwie sieci neuronowe: generator (G) i dyskryminator (D). Generator tworzy próbki danych na podstawie losowych szumów, a dyskryminator stara się odróżnić próbki wygenerowane od rzeczywistych.

W BiGAN mamy dodatkowy komponent, którym jest enkoder (E). Enkoder mapuje rzeczywiste dane do przestrzeni ukrytej, umożliwiając rekonstrukcję danych i tworzenie bardziej spójnych reprezentacji. Sposób działania jest następujący:

- **Generator (G):** Generuje próbki danych na podstawie losowych wektorów ze przestrzeni ukrytej z .
- **Dyskryminator (D):** Odróżnia rzeczywiste dane od wygenerowanych przez generator, przyjmując jako wejście zarówno próbki danych, jak i wektory ukryte.
- **Enkoder (E):** Mapuje rzeczywiste próbki danych do przestrzeni ukrytej z .

2.2 BiGAN Komponenty

Trening BiGAN odbywa się poprzez naprzemienne aktualizowanie generatora, dyskryminatora i enkodera, tak aby:

- **Generator:** Stara się wytworzyć próbki, które dyskryminator uzna za rzeczywiste.
- **Dyskryminator:** Uczy się odróżniać rzeczywiste próbki danych i ich wektory ukryte od wygenerowanych próbek i losowych wektorów ukrytych.
- **Enkoder:** Uczy się odwzorowywać rzeczywiste dane do przestrzeni ukrytej w sposób, który maksymalizuje podobieństwo między wygenerowanymi i rzeczywistymi danymi.

2.3 Dobór hiperparametrów

Hiperparametry dobrano według artykułu [1]. Jednak podczas pierwszych testów okazały się one nieskuteczne. W związku z tym przeprowadzono wiele testów hiperparametrów, jednak żaden nie przyniósł znaczących zmian w wynikach końcowych.

- $lr = 2 \times 10^{-3}$ - model był niestabilny
- $lr = 2 \times 10^{-4}$ - model się uczył, jednak wyniki nie były zadowalające
- $lr = 2 \times 10^{-5}$ - model uczył się wolniej, ale bardziej stabilnie

Najlepszy wynik uzyskano dla współczynnika uczenia $lr = 2 \times 10^{-5}$. Dodatkowo testowano różne wartości parametrów beta oraz weight decay w optymalizatorach i uznano za odpowiednie wartości 0.5 i 0.99.

Ponadto zwiększano latent size z 200 do 800, jednak wyniki były porównywalne, bez większego znaczenia. Dlatego wybrano wartość 200, jak w oryginalnym dokumencie [1].

2.4 Detekcja anomalii

Wykorzystamy ten model aby wskazać, czy dany obrazek jest anomalią. Posłużymy się tutaj równaniem z [1]:

$$A(x) = \alpha L_G(x) + (1 - \alpha)L_D(x) \quad (1)$$

gdzie $L_G(x) = \|x - G(E(x))\|_1$ oraz $L_D(x)$ zdefiniowane za pomocą funkcji strat krzyżowej entropii σ z dyskryminatora dla x będącego rzeczywistym: $L_D(x) = \sigma(D(x, E(x)), 1)$, co odzwierciedla pewność dyskryminatora, że próbka pochodzi z rzeczywistego rozkładu danych. Natomiast α jest liczbą z przedziału od 0 – 1. Do testowania wybrano 0.5

2.5 Struktura modelu

Model został stworzony na podstawie [1]

Operation	Kernel	Strides	Features Maps/Units	BN?	Non Linearity
<i>E(x)</i>					
Convolution	3×3	1×1	32	×	Linear
Convolution	3×3	2×2	64	✓	Leaky ReLU
Convolution	3×3	2×2	128	✓	Leaky ReLU
Dense			200	×	Linear
<i>G(z)</i>					
Dense			1024	✓	ReLU
Dense			$7*7*128$	✓	ReLU
Transposed Convolution	4×4	2×2	64	✓	ReLU
Transposed Convolution	4×4	2×2	1	×	Tanh
<i>D(x)</i>					
Convolution	4×4	2×2	64	×	Leaky ReLU
Convolution	4×4	2×2	64	✓	Leaky ReLU
<i>D(z)</i>					
Dense			512	×	Leaky ReLU
<i>Concatenate D(x) and D(z)</i>					
<i>D(x,z)</i>					
Dense*			1024	×	Leaky ReLU
Dense			1	×	Sigmoid
Optimizer	Adam($\alpha = 10^{-5}$, $\beta_1 = 0.5$)				
Batch size	100				
Latent dimension	200				
Epochs	100				
Leaky ReLU slope	0.1				
Weight, bias initialization	Isotropic gaussian ($\mu = 0$, $\sigma = 0.02$), Constant(0)				

Table 4: MNIST BiGAN Architecture and hyperparameters

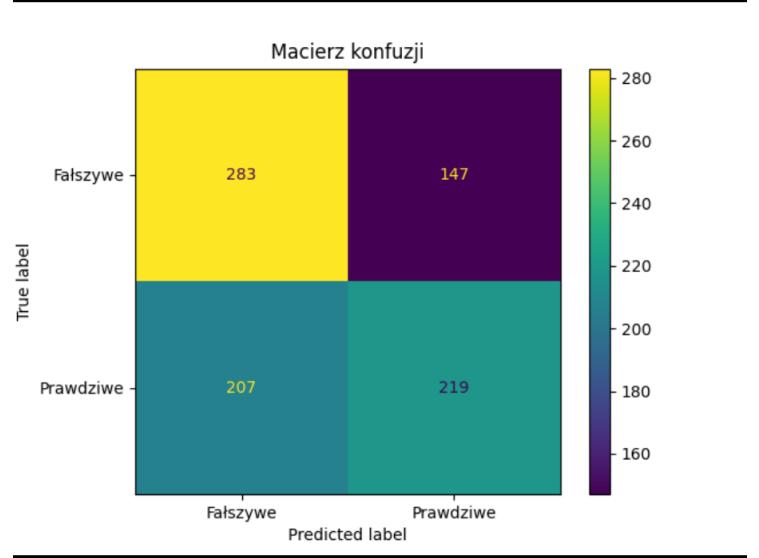
Rysunek 3: Struktura modelu. Źródło [1]

2.6 Wyniki

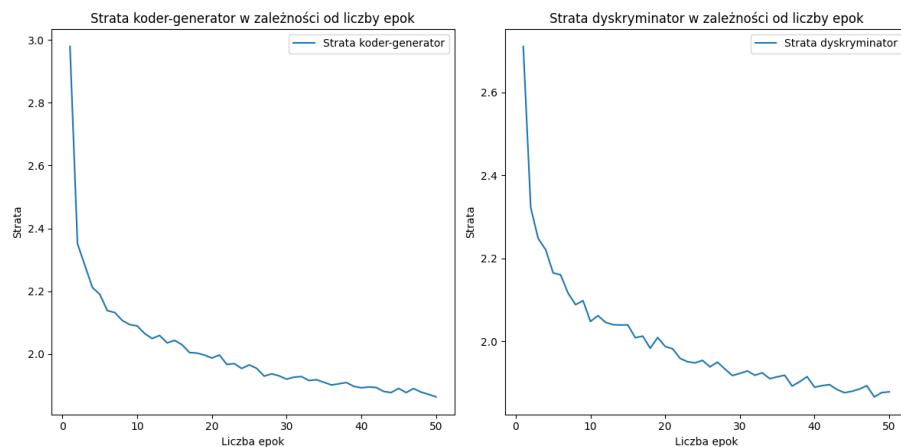
Oto wyniki modelu na zbiorze testowym, przedstawione w tabeli 3 oraz na rysunkach 4, 5 6, 7.

F1 Score	Precyza
0.55	59.9

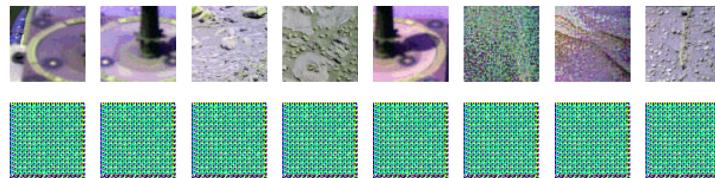
Tabela 3: F1 Score i Precyza dla zbioru testowego.



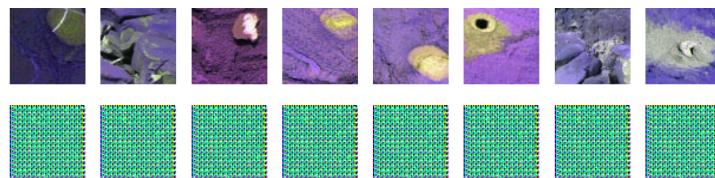
Rysunek 4: BiGAN Macierz Pomyłek



Rysunek 5: BiGAN Funkcja straty podczas treningu



Rysunek 6: BiGAN odwzorowanie obrazków typowych

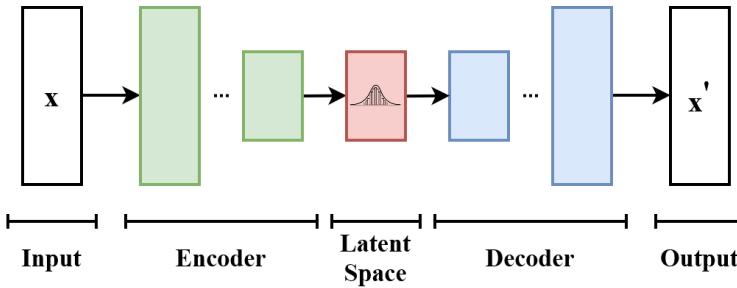


Rysunek 7: BiGAN odwzorowanie obrazków z nowościami

2.7 Wnioski

Model osiągnął wynik lepszy niż rzut monetą w detekcji anomalii. Jednak patrząc na wynik generatora, okazuje się, że jest on fatalny i model w ogóle nie potrafi wygenerować obrazków Marsa. Przetestowano bardzo wiele konfiguracji hiperparametrów, łącznie z liczbą epok, i model nie potrafił się w żaden sposób nauczyć. Najprawdopodobniej winna jest architektura modelu. Wydaje się, że kluczowe byłoby zmodyfikowanie generatora oraz dyskryminatora, aby lepiej adaptowały się do problemu.

3 Autokoder Wariacyjny (VAE)



Rysunek 8: Budowa model VAE

3.1 Architektura VAE

Variational Autoencoder (VAE) zaimplementowany w tym projekcie składa się z dwóch głównych komponentów: enkodera i dekodera. Każdy komponent jest zbudowany z warstw w pełni połączonych (gęstych), które przekształcają dane wejściowe w reprezentację latentną, a następnie rekonstruują ją.

3.1.1 Enkoder

Sieć enkodera mapuje dane wejściowe na przestrzeń latentną. Składa się ona z następujących warstw:

- **Warstwa wejściowa:** Przyjmuje dane wejściowe, które są spłaszczone obrazami o rozmiarze $64 \times 64 \times 3$.
- **Warstwa ukryta:** Warstwa w pełni połączona z 1024 neuronami, po której następuje funkcja aktywacji ReLU.
- **Warstwy latentne:** Dwie oddzielne warstwy w pełni połączone:
 - *Warstwa średnia:* Zwraca średnią (μ) rozkładu latentnego.
 - *Warstwa log-odchylenia:* Zwraca log-odchylenie ($\log \sigma^2$) rozkładu latentnego.

3.1.2 Dekoder

Sieć dekodera rekonstruuje dane wejściowe z reprezentacji latentnej. Składa się ona z następujących warstw:

- **Warstwa latentna wejściowa:** Przyjmuje wektor latentny próbnego z rozkładu zdefiniowanego przez enkoder.

- **Warstwa ukryta:** Warstwa w pełni połączona z 1024 neuronami, po której następuje funkcja aktywacji ReLU.
- **Warstwa wyjściowa:** Warstwa w pełni połączona, która zwraca zrekonstruowane dane, po której następuje funkcja aktywacji sigmoidalnej, aby zapewnić, że wartości wyjściowe są pomiędzy 0 a 1.

3.1.3 Klasa Variational Autoencoder (VAE)

Klasa `VariationalAutoencoder` rozszerza klasę implementującą specyficzne przetwarzanie w przód:

- **Przetwarzanie w przód przez enkoder:** Przekształca dane wejściowe na przestrzeń latentną, przechodząc przez warstwy enkodera.
- **Sztuczka reparametryzacyjna:** Próbkuje z rozkładu latentnego za pomocą średniej i log-odchylenia dostarczonych przez enkoder.
- **Przetwarzanie w przód przez dekoder:** Rekonstruuje dane wejściowe z próbki wektora latentnego, przechodząc przez warstwy dekodera.

Architektura VAE została zaprojektowana tak, aby zrównoważyć złożoność i pojemność modelu, zapewniając efektywne uczenie i rekonstrukcję danych wejściowych. Schematyczny model architektury został zaprezentowany na rysunku 8.

3.2 Dobór hiperparametrów

W celu znalezienia optymalnych hiperparametrów dla modelu VAE, przeprowadzono serię testów z różnymi wartościami współczynnika uczenia (learning rate) oraz rozmiarami warstw ukrytych.

Najpierw przetestowano model z różnymi współczynnikami uczenia, przy stałym rozmiarze warstw ukrytych. Przeprowadzono testy dla wartości:

- $lr = 1 \times 10^{-4}$
- $lr = 1 \times 10^{-5}$
- $lr = 1 \times 10^{-6}$

Najlepszy wynik uzyskano dla współczynnika uczenia $lr = 1 \times 10^{-6}$.

Następnie przetestowano różne rozmiary warstw ukrytych:

- $n_{hidden_features} = 512, n_{latent_features} = 64$
- $n_{hidden_features} = 1024, n_{latent_features} = 128$
- $n_{hidden_features} = 4096, n_{latent_features} = 256$

Najlepszy wynik uzyskano dla trzeciej kombinacji: $n_{hidden_features} = 4096$ oraz $n_{latent_features} = 256$.

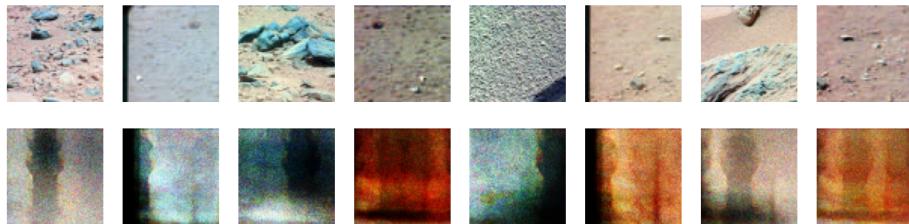
Dodatkowo przetestowano inną architekturę VAE, która posiadała dodatkową (czwartą) warstwę ukrytą. Warstwy miały odpowiednio rozmiary: 4096, 2048 oraz 256. Ta architektura uzyskała gorsze wyniki.

Każdy test przeprowadzano, trenując model przez 50 epok. Skuteczność treningu porównywano, obliczając wartość błędu średnio-kwadratowego (MSE) na zbiorze testowym.

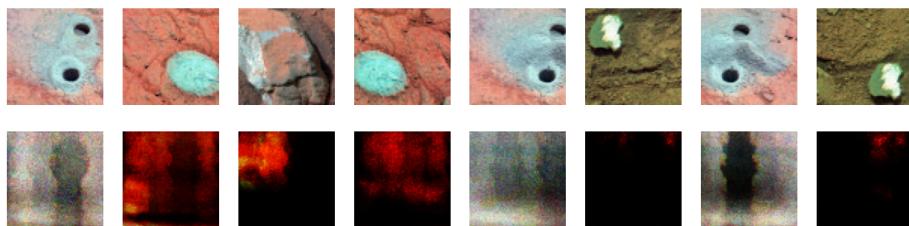
3.3 Trenowanie modelu

Model VAE został wytrenowany przy użyciu algorytmu optymalizacji Adam oraz funkcji straty ELBO (Evidence Lower Bound), która składa się z błędu rekonstrukcji i dywergencji Kullbacka-Leiblera (KL). Proces trenowania polegał na minimalizacji funkcji ELBO, a model trenowano przez 250 epok z współczynnikiem uczenia 10^{-6} i rozmiarem batcha 64, wykorzystując zbiór danych zawierający typowe obrazy. Trenowanie miało na celu zoptymalizowanie parametrów modelu, aby uzyskać jak najdokładniejszą rekonstrukcję danych wejściowych przy jednoczesnym zachowaniu właściwości probabilistycznych przestrzeni latentnej.

Na poniższych rysunkach (10 oraz 9) przedstawiono wyniki reprodukcji obrazów wejściowych przez wytrenowany model.



Rysunek 9: Reprodukcje obrazów typowych przez sieć VAE



Rysunek 10: Reprodukcje obrazów z nowością przez sieć VAE

3.4 Metoda klasyfikacji nowości

W celu klasyfikacji nowości zaimplementowana została metryka **euklidesowej odległości między średnimi rozkładów**. Metoda ta korzysta wyłącznie z informacji o średnich aproksymowanych posteriorów w przestrzeni latentnej zarówno dla normalnych (zbiór treningowy), jak i testowych punktów danych. W tym podejściu wartość nowości obliczana jest jako odległość między średnią rozkładu w przestrzeni latentnej punktu testowego a najbliższą średnią rozkładu w przestrzeni latentnej normalnej próbki: [3]

$$N_{\bar{q} - \bar{q}_y}(\mathbf{x}_{\text{test}}) = \min_{y \in Y} \|E[q_\phi(z | \mathbf{x}_{\text{test}})] - E[q_\phi(z | y)]\|_2^2 \quad (2)$$

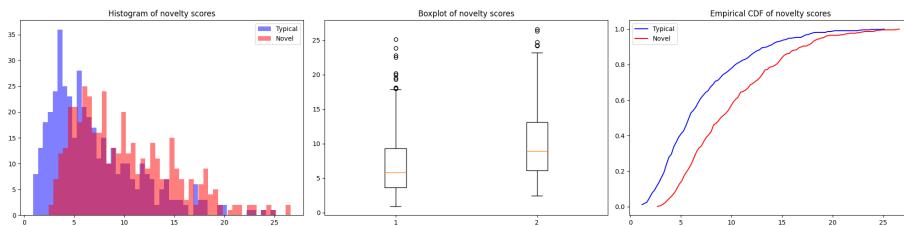
gdzie minimum jest wyznaczane spośród wszystkich normalnych próbek y w zbiorze danych Y .

3.5 Wyniki klasyfikacji

W celu oceny skuteczności modelu VAE w detekcji nowości obliczono metryki klasyfikacji nowości. Na podstawie wyznaczonego progu dokonano klasyfikacji próbek testowych na typowe i nowości. Następnie obliczono metryki takie jak dokładność (Precision), czułość (Recall) oraz miarę F1, które przedstawiono w tabeli 5.

Wartości tych metryk są całkowicie zależne od wybranego progu (threshold). Można dostosować ten parametr do własnych potrzeb, w zależności od tego, czy chcemy uzyskać wyższą precyzję (Precision), czy czułość (Recall).

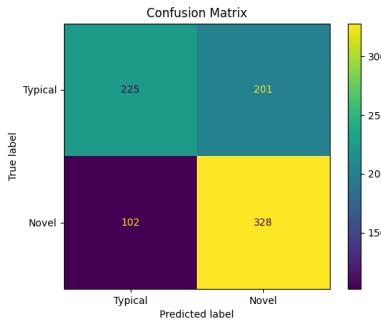
Na rysunku 11 przedstawiono histogramy, wykres pudelkowy oraz dystrybuantę empiryczną wyników klasyfikacji nowości. Na rysunku 12 przedstawiono macierz pomyłek. Wyraźnie widać, że dystrybucja novelty dla danych testowych zawierających nowości jest przesunięta w stronę wyższych wartości względem dystrybucji dla danych testowych typowych.



Rysunek 11: Wyniki klasyfikacji nowości

Metryka	Wartość
Precision	0.6200
Recall	0.7628
F1 score	0.7627

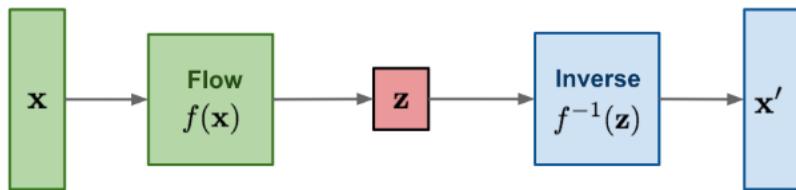
Tabela 4: Metryki wydajności klasyfikatora nowości



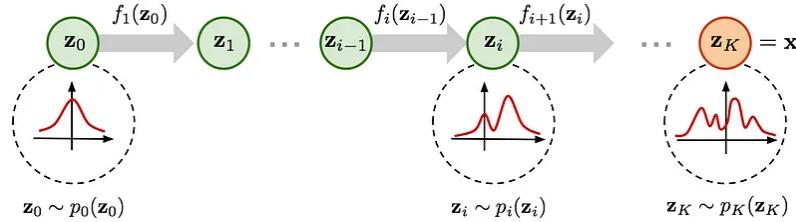
Rysunek 12: Macierz pomylek

4 Normalizing Flows (NF)

Normalizing Flows (NF) to rodzaj modeli probabilistycznych, które transformują proste rozkłady bazowe w bardziej złożone rozkłady za pomocą ciągu odwracalnych przekształceń. Dzięki temu, są w stanie modelować skomplikowane rozkłady danych, zachowując jednocześnie możliwość obliczenia dokładnej gęstości prawdopodobieństwa, także generowania nowych próbek. Jedną z ważniejszych zalet NF jest ich zdolność do dokładnego odwzorowania złożonych struktur danych.



Rysunek 13: Budowa modelu Flow



Rysunek 14: Zasada działania modelu Flow

4.1 Przykłady Normalizing Flows

- Planar Flows - używają prostych, parametrycznych przekształceń do modelowania rozkładów, co pozwala na elastyczne odwzorowanie struktury danych przy ograniczonej złożoności obliczeniowej.
- Radial Flows - korzystają z przekształceń, które rozciągają lub kurczą przestrzeń wzdłuż promieni, umożliwiając modelowanie bardziej skomplikowanych struktur.
- Autoregressive Flows - wykorzystują autoregresję do modelowania skomplikowanych zależności między zmiennymi, np. Masked Autoregressive Flow (MAF) i Inverse Autoregressive Flow (IAF).
- RealNVP (Real-valued Non-Volume Preserving) - wykorzystują specyficzne przekształcenia, tzw. warstwy sprzęgające (coupling layer), które pozwalają na łatwe obliczanie odwrotności i jacobianów, umożliwiając efektywne modelowanie złożonych rozkładów.

4.2 Masked Autoregressive Flow (MAF)

Do niniejszych eksperymentów wybrano model Masked Autoregressive Flow (MAF). Jest to szczególny przypadek Autoregressive Flows, wyróżnia się następującymi cechami:

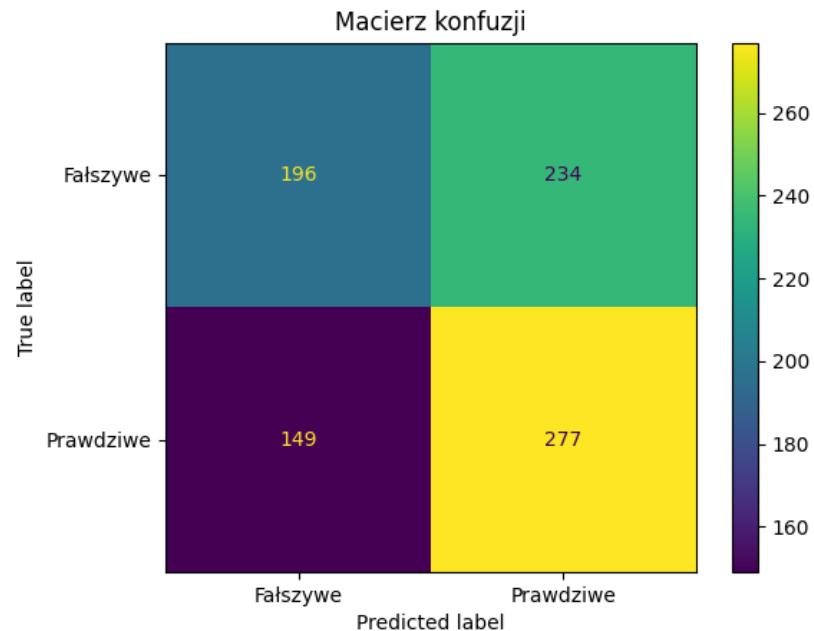
- Maskowanie - w celu zapewnienia autoregresji, MAF wykorzystuje maskowanie. Maski są stosowane na wagach warstw sieci neuronowej, co wymusza, aby każda zmienna była zależna tylko od poprzednich zmiennych w danej kolejności.
- Logarithmic Likelihood - MAF umożliwia dokładne obliczenie funkcji gęstości prawdopodobieństwa (log-likelihood) dla danych wejściowych, co jest korzystne dla wielu zadań probabilistycznych.
- Stabilność uczenia - podejście autoregresywne w MAF skutkuje stabilniejszym procesem uczenia w porównaniu z niektórymi innymi NF, ponieważ nie wymaga inwersji przekształceń podczas treningu.

4.3 Wyniki trenowania

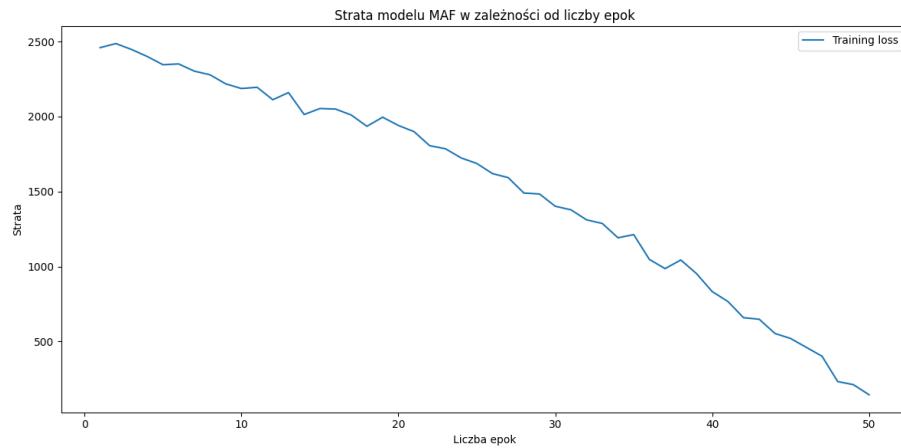
Model trenowany był przez 50 epok.

Metryka	Wartość
Precision	0.5518
Recall	0.5868
F1 score	0.5688

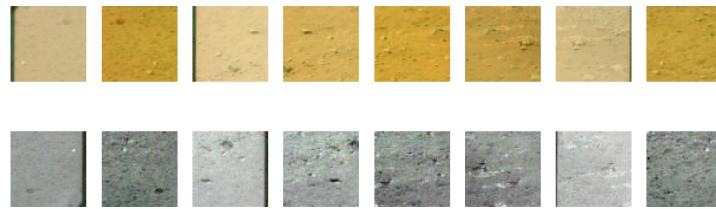
Tabela 5: Metryki wydajności klasyfikatora nowości



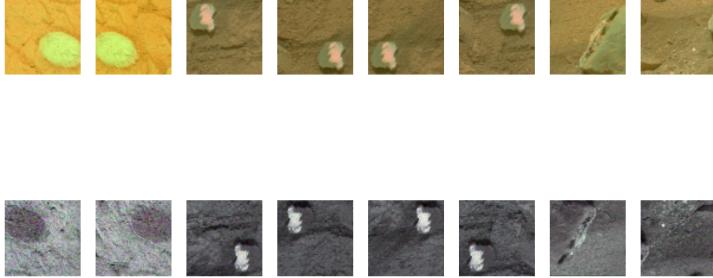
Rysunek 15: MAF Macierz Pomyłek



Rysunek 16: MAF Funkcja straty podczas treningu



Rysunek 17: Reprodukcje obrazów typowych przez model MAF



Rysunek 18: Reprodukcje obrazów z nowościami przez MAF

4.4 Metoda klasyfikacji nowości

Metoda klasyfikacji nowości opiera się na ocenie logarytmicznego prawdopodobieństwa (\log_{prob}) konkretnych próbek przy użyciu modelu MAF (Masked Autoregressive Flow). \log_{prob} jest wykorzystywane jako wskaźnik, gdzie niższe wartości \log_{prob} wskazują na bardziej "nowe" lub nietypowe próbki, podczas gdy wyższe wartości mogą sugerować bardziej typowe lub normalne próbki.

5 Wnioski

Nasze modele zostały przetestowane i wyniki zostały przedstawione. Kod jest reprodukowalny, a instrukcja uruchomienia znajduje się w pliku README.md. Widzimy, że BiGAN nie poradził sobie z zadaniem. Natomiast VAE oraz Flow znacznie lepiej poradziły sobie z zadaniem.

Literatura

- [1] arxiv.org. <https://arxiv.org/pdf/1802.06222.pdf>. [Accessed 21-06-2024].
- [2] Comparison of novelty detection methods for multispectral images in rover-based planetary exploration missions - Data Mining and Knowledge Discovery — link.springer.com. <https://link.springer.com/article/10.1007/s10618-020-00697-6>. [Accessed 21-06-2024].
- [3] Aleksei Vasilev, Vladimir Golkov, Marc Meissner, Ilona Lipp, Eleonora Sgarlata, Valentina Tomassini, Derek K. Jones, and Daniel Cremers. q-Space Novelty Detection with Variational Autoencoders. *arXiv preprint arXiv:1806.02997*, 2018.