Detekcja anomalii w zbiorze CREDO

Jan Tyc, Marcin Zub

6 lutego 2024

1 Wstęp

1.1 Projekt CREDO

CREDO (Cosmic Ray Extremely Distributed Observatory) [5] to projekt badawczy, który skupia się na monitorowaniu promieni kosmicznych poprzez stworzenie rozległej sieci detektorów rozmieszczonych na całym świecie. Promienie kosmiczne to wysokoenergetyczne cząstki pochodzące z przestrzeni kosmicznej, takie jak protony czy jądra atomowe, które docierają do Ziemi. Projekt CREDO ma na celu zrozumienie i zarejestrowanie tych promieni kosmicznych poprzez skonstruowanie globalnej infrastruktury detekcyjnej.

Rozproszona natura zbierania danych do CREDO opiera się na wykorzystaniu istniejących technologii, takich jak kamery CMOS smartfonów i wykorzystywaniu ich jako detektory promieniowania kosmicznego. Dzięki temu projekt może zaangażować ogromną liczbę uczestników na całym świecie, co pozwala na zebranie dużej ilości danych i stworzenie bardziej kompleksowego obrazu dotyczącego promieniowania kosmicznego. Wyniki uzyskane przez CREDO mają potencjał przyczynienia się do lepszego zrozumienia procesów kosmicznych oraz do rozwinięcia dziedziny astrofizyki wysokich energii.

1.2 Detekcja anomalii

Detekcja anomalii [7] to proces identyfikowania nieprawidłowości, odstępstw lub nietypowych zachowań w danym systemie lub zbiorze danych. Anomalie te mogą wskazywać na potencjalne problemy, błędy lub nieoczekiwane zdarzenia, co czyni detekcję anomalii kluczowym elementem w różnych dziedzinach, takich jak bezpieczeństwo komputerowe, monitorowanie sieci, diagnostyka medyczna czy analiza danych.

Głównymi metodami detekcji anomalii w danych 2D (obrazach) [10] są:

1. Podejście statystyczne
2. Rekonstrukcja obrazu
3. Jedno-klasowa klasyfikacja
4. Klasteryzacja cech

2 Metody

2.1 Dane

Zbiór danych [2] tworzy ponad 1 milion zdjęć zrobionych przez kamery CMOS zaimplementowanych w smartfonach na całym świecie. Każde zdjęcie jest w formacie JPG i ma wymiary 60x60 pikseli i przedstawia zderzenie naładowanej cząstki z detektorem. Wyróżnia się kilka klas zdjęć między innymi: dots, worms, anomalies.

2.2 Preprocessing

Celem preprocessingu jest ujednolicenie danych i zmniejszenie rozbieżności przy zachowaniu cech pierwotnych obrazu. Żeby to uzyskać, każde zdjęcie zostały poddane następującej procedurze:

1. Zmiana z formatu RGB do skali szarości (eng. grayscale)
2. Segmentacja za pomocą algorytmu OTSU [6].

3. Dylatacja i otwrcie o promieniu 5 w celu rozszerzenia obszaru znaczącego dla obiektu.
4. Nałożenie otrzymanej w ten sposób maski na pierwotny obraz
5. Okrślenie środka ciężkości i przesunięcie obiektu na środek.
6. Dopasowanie prostej do maski obrazu i obrót obiektu na obrazie, żeby najlepiej dopasowana prosta biegła pionowo.

Przetwarzanie danych w ten sposób sprawia, że podczas rotacji tracona jest część informacji. Ilość traconej informacji jest obliczana przez obrót, powrót do pierwotnego stanu i obliczenie średniego błędu procentowego [3] między dwoma obrazami. Poprzez pojedynczy obrót tylko 0,25% informacji jest tracone z obrazu. Preprocessing został częściowo oparty na pracy odnośnie klasyfikacji danych pochodzących z CREDO [4].

2.3 Ekstrakcja cech

Każdy obraz posiada wymiary 60x60 co sprawia, że charakteryzuje go 3600 cech. W celu redukcji wymiarowści danych zdecydowano się na następujące podejścia:

- Rozkład cech przy pomocy PCA i PCA-2D
- Ekstrakcję cech przy pomocy autoencoderów liniowych i konwolucyjnych.

2.3.1 PCA, PCA-2D

"Principal component analysis"(PCA) [8] jest metodą ekstrakcji cech. Każdy obraz posiadał wymiary 60x60 czyli posiadał 3600 rozłącznych cech. Obraz można traktować jako macierz 60x60 lub jako wektor o długości 3600. PCA dokonuje redukcji wymiarowości poprzez tworzenie macierzy kowariancji oraz wartości oraz wektorów własnych tej macierzy. Następnie uzyskane wektory są sortowane i wybierane jest z nich k pierwszych wektorów. Wynikiem działania PCA jest k wektorów o długości 3600, a poprzez dodanie ich do siebie w odpowiednich proporcjach jesteśmy w stanie odwzorować każdy obraz ze zbioru (z pewną dokładnością). Wzajemne proporcje poszczególnych wektorów stają się nowymi cechami naszego obrazu. W ten sposób jesteśmy w stanie przedstawić każdy obraz jako zbiór ok. 60 innych obrazów, oraz wektorów o długości 60, które opisują udział poszczególnych obrazów. W ten sposób z 3600 udało nam się zredukować liczbę cech do 60.

PCA-2D [11] opiera się na podobnej zasadzie, ale zamiast obrazów i wektora, otrzymujemy dwie macierze, macierz bazową oraz macierz cech o wymiarach odpowiednio 60:5 i 5:60. Poprzez pomnożenie ze sobą tych macierzy otrzymujemy zrekonstruowany obraz.

2.3.2 Autoenkodery

Autoenkodery to rodzaj algorytmów w dziedzinie uczenia głębokiego, które są używane do redukcji wymiarów danych lub do ekstrakcji istotnych cech z danych [1]. Są one często stosowane w problemach związanych z przetwarzaniem obrazów, dźwięku i tekstu.

Podstawowym celem autoenkoderów jest uczenie się reprezentacji danych w sposób nienadzorowany. Składa się z dwóch głównych części: enkodera (koder) i dekodera. Enkoder przekształca wejściowe dane na pewną reprezentację w niższym wymiarze, natomiast dekoder rekonstruuje dane z tej reprezentacji.

W naszej pracy używamy autoenkoderów do redukcji wymiarowości obrazów dlatego używamy dwóch rodzajów autoenkoderów:

- Liniowych
- Konwolucyjnych

Autoenkodery liniowe są w stanie znacząco zredukować liczbę cech, jednak ich głównym problemem jest traktowanie obrazu jako zbioru niezależnych parametrów. W ten sposób część informacji może być tracona podczas uczenia.

Autoenkodery konwolucyjne z kolei bardzo dobrze zachowują informację obrazu z uwagi na warstwy konwolucyjne, ale nie są w stanie dokonać tak znaczącej redukcji wymiarowości z zachowaniem znaczących dla obrazu cech.

Przykładowa architektura autoenkodera konwolucyjnego zaimplementowana w pythonie:

```
class AutoencoderConvolutional(nn.Module):
    def __init__(self) -> None:
        super(AutoencoderConvolutional, self).__init__()

        self.encoder = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1),
            nn.LeakyReLU(0.1),
            nn.Conv2d(16, 32, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.Conv2d(128, 128, kernel_size=1, stride=1, padding=1),
        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(128, 128, kernel_size=3, stride=1, padding=1),
            nn.LeakyReLU(0.1),
            nn.ConvTranspose2d(128, 64, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.ConvTranspose2d(64, 32, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.ConvTranspose2d(32, 16, kernel_size=3, stride=2, padding=1),
            nn.LeakyReLU(0.1),
            nn.ConvTranspose2d(16, 1, kernel_size=3, stride=1, padding=1),
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)

        return decoded
```

Warstwa "self.encoder"jest enkoderem, która dokonuje zmiany wymiarów z obrazu z 1x64x64 na 128x8x8 poprzez 5 warst konwolucji. Warstwa "self.decoder"dokonuje powortu do pierwotnych wymiarów (przez transponowaną konwolucję) i jest odwrotnością enkodera.

2.4 Detekcja anomalii na podstawie cech

2.4.1 Podejście statystyczne - obliczanie "Z-score"

Każdą otrzymaną cechę można traktować jako niezależną od innych. Dany parametr najczęściej poddaje się rozkładowi normalnemu. Możemy wtedy obliczyć jak dana próbka danych jest odległa od statystycznego rozkładu normalnego (w odniesieniu do wariancji). W ten sposób oblicza się Z-score, czyli:

$$Zscore = (x_i - mean(x_n))/var(x_n) \tag{1}$$

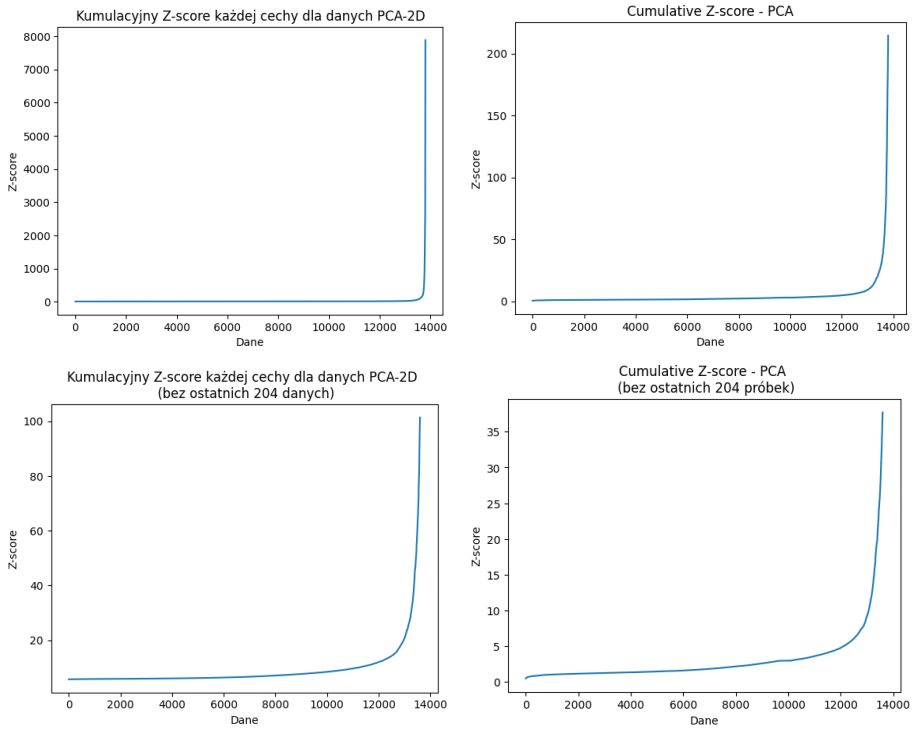
W ten sposób możemy oszacować jak bardzo dana próbka jest nietypowa [9]. W przypadku wielu rozłącznych parametrów możemy kumulować ten wynik, czyli obliczyć osobno dla każdego parametru i zsumować, w celu przewidzenia odległości próbki w odróżnieniu do innych przypadków.

2.4.2 Rekonstrukcja obrazu

Obraz rozkładany jest na cechy, czyli poddawany jest redukcji wymiarów. Po redukcji dokonuje się jego rekonstrukcji [12]. Im bardziej pierwotny obraz różni się od rekonstrukcji tym większa szansa, że obraz był anomalią lub, że znacząco różnił się od pozostałych w zbiorze danych.

2.4.3 Klasyfikacja jednej klasy

Mały podzbiór przypadków typowych poddaje się uczeniu klasyfikatorem SVM, ponieważ dobrze określa on największą odległość od innych przypadków. Następnie na wytrenowanym klasyfikatorze



Rysunek 1: Wykresy Z-Score (oś X przedstawia kolejne próbki danych, zaś oś Y przedstawia skumulowany Z-score dla danego obrazu)

sprawdza się cały zbiór i przypadki, który nie zostają przyporządkowane do klasy są anomaliami.

2.4.4 Klasteryzacja

Klasteryzacja polega na łączeniu ze sobą punktów w grupy na podstawie odległości (najczęściej euklidesowej lub manhattan). Wykorzystany algorytm DB-scan posiada dwa parametry: epsilon oraz min samples. Pierwszy określa maksymalną odległość jaką mogą mieć od siebie dwa punkty, żeby należeć do tej samej grupy, zaś min samples określa ile punktów musi należeć do grupy, żeby punkty nie zostały uznane za anomalie (outliery). W ten sposób na podstawie zredukowanych parametrów możemy dokonać klasteryzacji cech i określić, które punkty (czyli w naszym przypadku obrazy) są anomaliami.

3 Wyniki badań

3.1 Z-score

Z-score został obliczony dla wszystkich parametrów podczas rozkładu cech przy użyciu PCA oraz PCA-2D. PCA dokonuje redukcji do 60 wymiarów, zaś PCA-2D do 300.

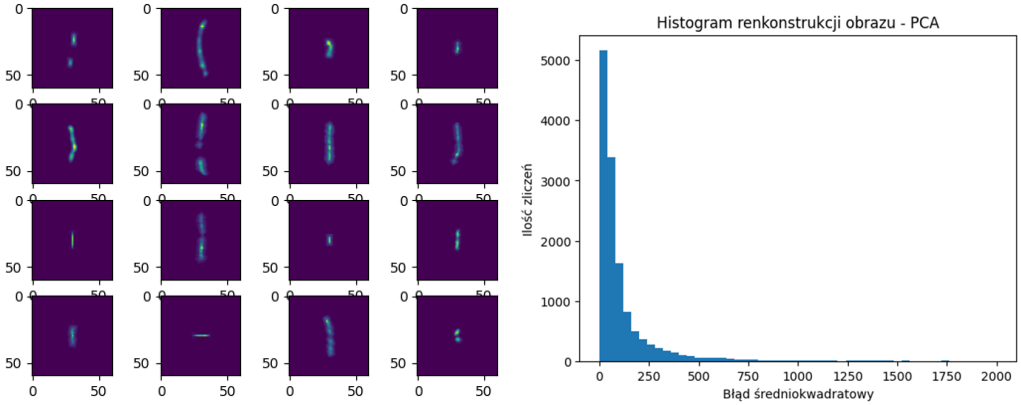
Na rysunku 1 bardzo wyraźnie widoczne jest rozróżnienie między przypadkami normalnymi i anomaliami. Jednak tylko mała część zbioru jest uznawana za anomalie. W określeniu przykładowego punktu odcięcia (progu od którego przypadek staje się anomalią) może pomóc wykluczenie pewnej części najbardziej niedopasowanych przypadków.

3.2 Rekonstrukcja obrazu

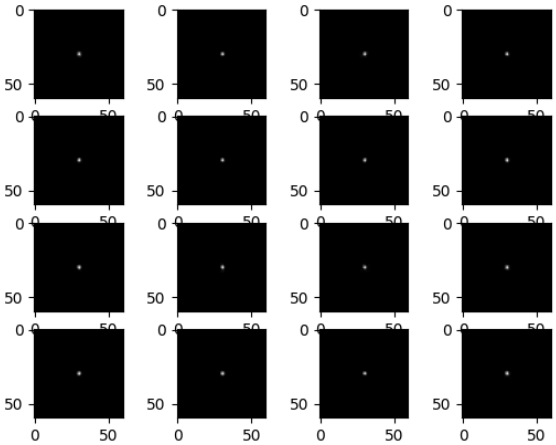
Tą metodą przetestowane zostały wszystkie metody ekstrakcji cech tj. PCA, PCA-2D, autoencodery liniowe oraz konwolucyjne.

Rysunek 2 przedstawia wyniki uzyskane dla metod PCA i PCA-2D. Trudno na jego podstawie jednoznacznie określić punkt odcięcia w którym obraz zaczyna być anomalią.

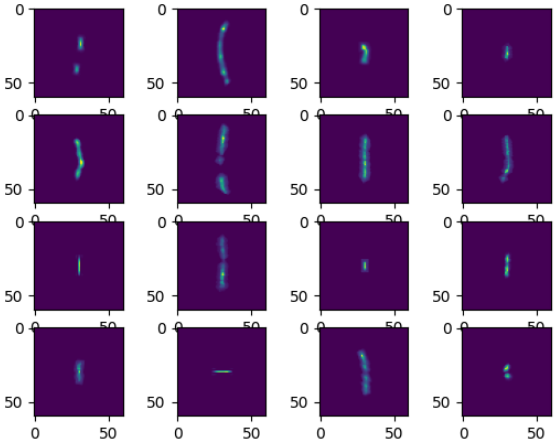
Wyniki dla autoendcodera konwolucyjnego są widoczne na rysnku 9. Dużo wyraźniej widać na nich spadek i odcięcie. Może to świadczyć o dużej różnicy w między obiektami normalnymi oraz tymi klasyfikowanymi jako anomalie.



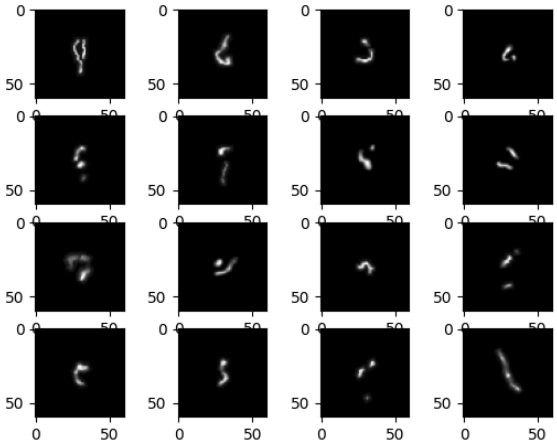
Rysunek 2: Histogram rekonstrukcji obrazu dla metod PCA i PCA-2D



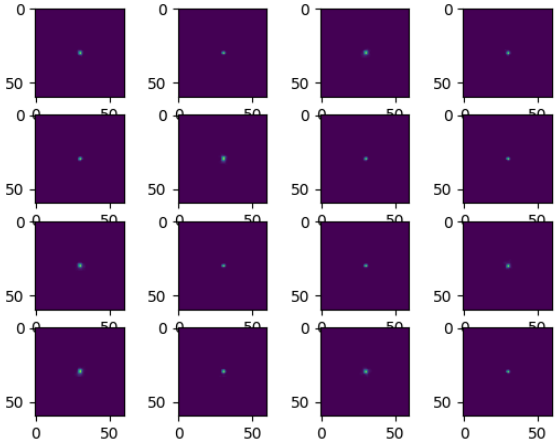
Rysunek 3: Najbardziej typowe obrazy PCA



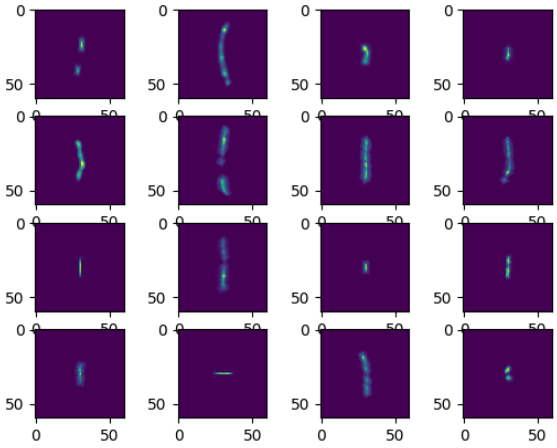
Rysunek 4: Wyniki około 5% największego błędu rekonstrukcji - PCA



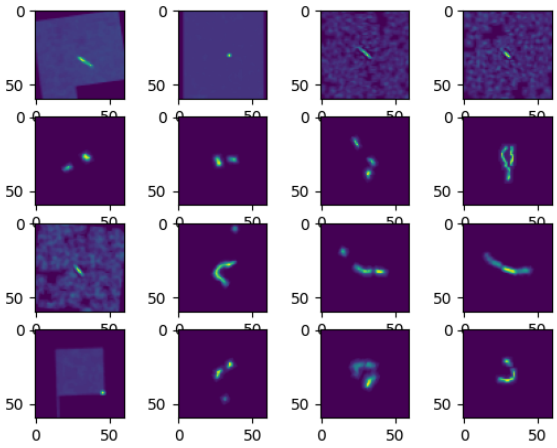
Rysunek 5: Największe anomalie PCA



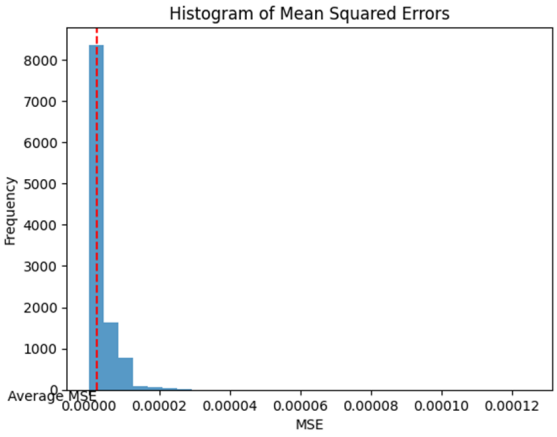
Rysunek 6: Najbardziej typowe obrazy PCA-2D



Rysunek 7: Wyniki około 5% największego błędu rekonstrukcji - PCA-2D



Rysunek 8: Największe anomalie PCA-2D



Rysunek 9: Histogram błedu dla autoencodera konwolucyjnego

		Outliery (%)	
Ilość próbek klasy 0	Ilość próbek klasy 1	2D-PCA	PCA
8000	600	0.1293	0.1665
8000	500	0.1230	0.1340
8000	450	0.1177	0.1247
8000	400	0.1161	0.0854
8000	200	0.1018	0.0441
8000	100	0.0597	0.0368
8000	50	0.0588	0.0157
11000	600	0.0841	0.1089
11000	500	0.0704	0.0944
11000	400	0.0617	0.0713
11000	300	0.0566	0.0579
11000	200	0.0539	0.0393
11000	100	0.0399	0.0315
11000	50	0.0384	0.0138

Rysunek 10: Ilość outlierów w zależności od liczebności klas podczas uczenia SVM

3.3 One-class SVM

Zamiast trenować klasyfikator na jednej klasie, zdecydowano się na trenowanie na dwóch klasach, obiektów normalnych i anomaliach. Obiekty i anomalie zostały zaklasyfikowane na podstawie wartości Z-score, którą otrzymały. W ten sposób wybierano poszczególne liczebności obu klas i porównano wyniki dla PCA i PCA-2D. Rysunek 10 przedstawia uzyskane rezultaty.

Uzyskane wyniki świadczą o przewadze metody PCA-2D z uwagi na większą stabilność wyników względem metody PCA, której rozrzut był bardzo duży i mocno zależał od liczebności klas. Ponadto uzyskane wyniki mogą się różnić od specyfiki problemu (czy chcemy rozpoznać wszystkie przypadki anomalii kosztem normalnych obiektów, czy chcemy rozpoznawać obiekty normalne kosztem anomalii).

3.4 Autoenkodery

Po kilkukrotych modyfikacjach autoenkodera, najlepsze rezultaty otrzymano w przypadku użycia architektury 2.3.2. W wyniku przetwarzania wstępnego obrazów - opisanego w rozdziale **2.2 Preprocessing** - na niektórych z nich 11 uwydatnione zostało w sposób znaczący tło.
To właśnie te obrazy autoenkoder klasyfikował najczęściej jako anomalie. Występuje na nich zdecydowanie więcej jaśniejszych pikseli (na obrazach "normalnych"tło jest czarne) i pomimo centralnego punktu ulokowanego w środku obrazu o najwyższej wartości jasności pikseli tworzących, rekonstrukcje tych obrazów były najmniej dokładne, dając przy tym największy błąd.
Warto zwrócić uwagę na wyodrębnione cechy autoenkodera (rysunek 12). Pozwalają one na zrozumienie tego *"co widzi"* autoencoder. Łatwo jest dostrzec, że z punktu widzenia modelu, liczy się też obrys (ramka) elementu centralnego, która to w przypadku obrazów o zwiększonej jasności tła znacząco odbiega w rozmiarze od standardowego przypadku (punktu).

Pomimo wszystko, przedstawiony model autoencodera był w stanie wyodrębić również obrazki 13, które według naszych założeń wstępnych, powinny być traktowane jako anomalie.

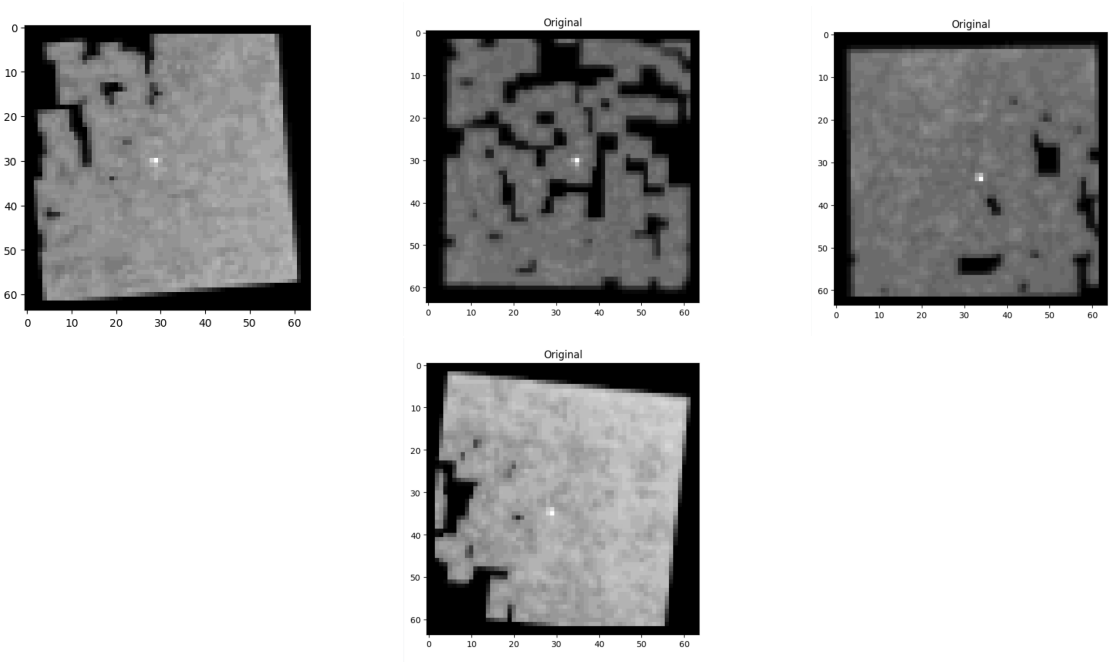
3.5 Klasteryzacja

Do klasteryzacji wykorzystano algorytm DB-scan z ustaloną wartością min_samples (3 lub 20) oraz progresywanie rosnącym ϵ w celu sprawdzenie ilości outlierów w zależności od przyjętej wartości. Wyniki przedstawione są na rysunku 14.

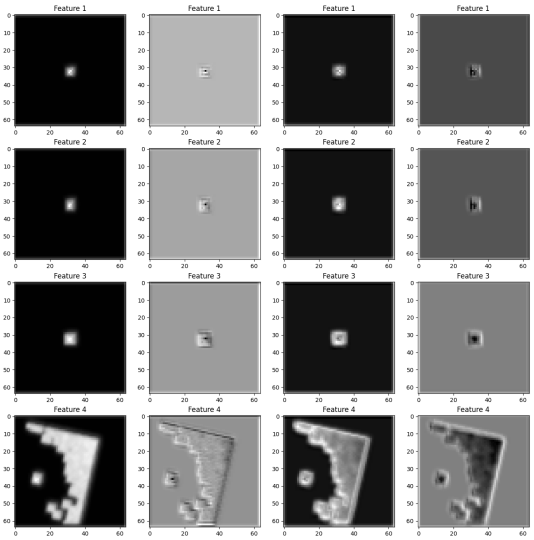
Ilość outlierów progresywanie spada podczas zwiększania odległości ϵ. Metoda prawdopodobnie wymaga korelacji, ponieważ klastrowanie cech z których każda może mieć inny rozmiar i rozkład sprawia, że bezpośrednie porównywanie cech nie oddaje dobrze ich wzajemnych korelacji.

4 Podsumowanie

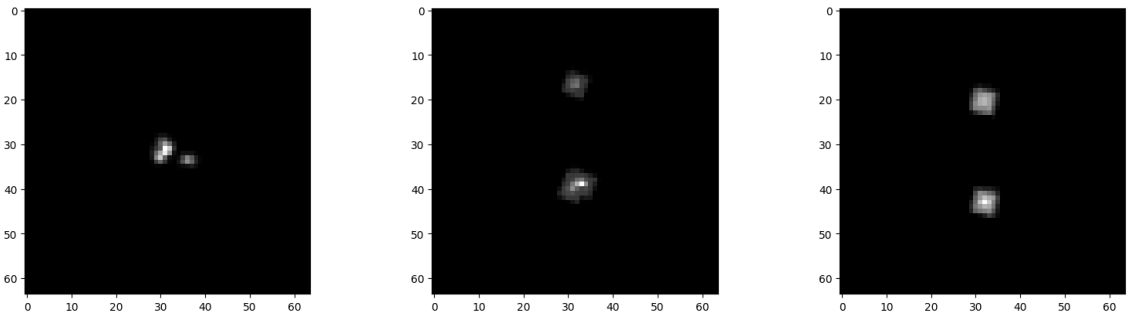
Detekcja anomalii bardzo mocno zależy od używanej metody oraz definicji anomalii. Wyniki uzyskane przez różne metody detekcji dają podobne rezultaty, jednak mogą znacząco różnić się w sposobie i podstawie podejmowanych decyzji. Najbardziej uniwersalną metodą uznana została rekonstrukcja obrazu, ponieważ dobrze oddaje charakter podobnych do siebie przypadków normalnych i jest bardziej czuła na mniejsze rozbieżności cech. Z-score jest natomiast świetną metodą do detekcji outlierów z bardzo wysoką precyzją, ponieważ dobrze znajduje on przypadki mocno odstające od zbioru danych. Największą zaletą klasyfikacji SVM jest łatwa możliwość binarnego



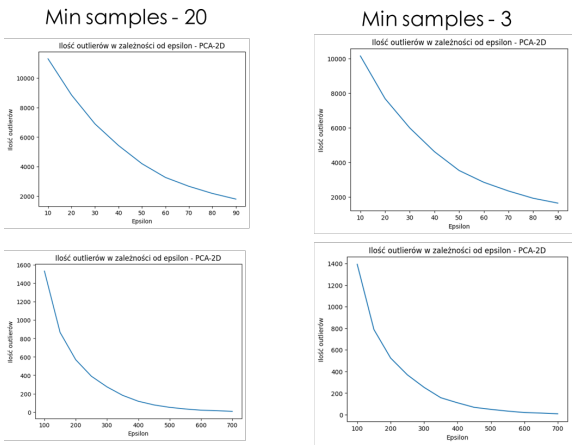
Rysunek 11: Obrazy z uwydatnionym tłem



Rysunek 12: Cechy po ekstrakcji przez autoenkoder



Rysunek 13: Pozostałe anomalie wykryte przez autoenkoder



Rysunek 14: Ilość outlierów w zależności od parametru ϵ

rozgraniczne między anomaliami i normalnymi obiektami i możliwość łatwego dostosowania parametrów do problemu. Klasteryzacja nie daje zbyt dobrych wyników z uwagi na brak normalizacji cech oraz brak równoważności cech (w PCA niektóre cechy mogą być bardziej znaczące niż inne). Odpowiedni preprocessing danych mógłby rozwiązać te problemy ale wymagane są dalsze testy w tej sprawie.

Problem z jednoznacznym stwierdzeniem co jest anomalią a co nie może być nierozstrzygalny, ponieważ nie da się binarnie rozgraniczyć przypadki na typowe i nietypowe (tak jak np. w przypadku klasyfikacji), anomalie dużo częściej zachowują się jak spektrum, dla którego można wyznaczyć stopień dopasowania obiektu do zbioru.

5 Kod źródłowy

Kod źródłowy dostępny jest na repozytorium na Githubie: https://github.com/MarcinZ20/Anomaly-Detection-in-CREDO-Dataset

Literatura

[1] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018.

[2] CREDO. https://credo.science//accesstodata, 2023.

[3] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, 2016.

[4] Tomasz Hachaj, Marcin Piekarczyk, and Łukasz Bibrzycki. Deep neural network architecture for low-dimensional embedding and classification of cosmic ray images obtained from cmos cameras. In *International Conference on Neural Information Processing*, pages 307–316. Springer, 2021.

[5] Piotr Homola, Dmitriy Beznosko, Gopal Bhatta, Łukasz Bibrzycki, Michalina Borczyńska, Łukasz Bratek, Nikolay Budnev, Dariusz Burakowski, David E Alvarez-Castillo, Kevin Almeida Cheminant, et al. Cosmic-ray extremely distributed observatory. *Symmetry*, 12(11):1835, 2020.

[6] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[7] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38, 2021.

[8] Liton Chandra Paul, Abdulla Al Suman, and Nahid Sultan. Methodological analysis of principal component analysis (pca) method. *International Journal of Computational Engineering & Management*, 16(2):32–38, 2013.

[9] Peter J Rousseeuw and Mia Hubert. Anomaly detection by robust statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1236, 2018.

[10] Jie Yang, Ruijie Xu, Zhiquan Qi, and Yong Shi. Visual anomaly detection for images: a systematic survey. *Procedia Computer Science*, 199:471–478, 2022.

[11] Daoqiang Zhang and Zhi-Hua Zhou. (2d) 2pca: Two-directional two-dimensional pca for efficient face representation and recognition. *Neurocomputing*, 69(1-3):224–231, 2005.

[12] Kang Zhou, Jing Li, Weixin Luo, Zhengxin Li, Jianlong Yang, Huazhu Fu, Jun Cheng, Jiang Liu, and Shenghua Gao. Proxy-bridged image reconstruction network for anomaly detection in medical images. *IEEE Transactions on Medical Imaging*, 41(3):582–594, 2021.