AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa

Zastosowanie generatywnych sieci współzawodniczących do generowania etykietowanych danych

Application of generative adversarial networks for generating labeled data

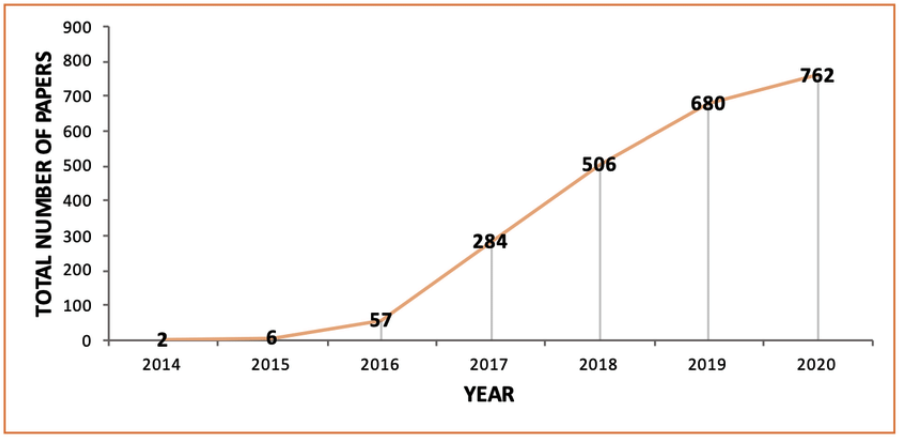
Autor: *Norbert Sak*
Kierunek studiów: *Automatyka i robotyka*
Opiekun pracy: *dr hab. inż. Joanna Kwiecień*

Kraków, 2023

Spis treści

1. Wprowadzenie

W dzisiejszych czasach dane stały się jednym z kluczowych zasobów w wielu dziedzinach technologii i nauki. Uczenie maszynowe jest obszarem, w którym ilość oraz jakość posiadanych danych znacząco wpływa na wyniki osiągane przez modele. Często jednak dostęp do danych odpowiedniej jakości jest mocno ograniczony, a szczególnie do danych etykietowanych, których zdobycie jest często związane z dużymi nakładami czasowymi i finansowymi. Techniki generatywne, takie jak generatywne sieci współzawodniczące (GAN, ang. *generative adversarial networks*), pozwoliły na znaczne poszerzenie perspektyw w kontekście generowania syntetycznych danych o wysokiej jakości i odwzorowaniu szczegółów z danych bazowych. Na podstawie wykresu 1.1 można zauważyć znaczący wzrost, z roku na rok, popularności sieci GAN od momentu ich wprowadzenia w roku 2014. Od tamtego momentu wprowadzonych zostało wiele modyfikacji i wariantów sieci GAN, które pozwalają na generowanie danych o jeszcze lepszej jakości i odwzorowaniu szczegółów. Niniejsza praca ma na celu szczegółowe zbadanie potencjału oraz wyzwań związanych z generowaniem danych etykietowanych przy użyciu generatywnych sieci współzawodniczących.



Rys. 1.1. Liczba publikacji naukowych związanych z sieciami GAN od czasu ich wprowadzenia w 2014 roku do roku 2020. Źródło: [1].

1.1. Cele pracy

Celem pracy jest wykorzystanie generatywnych sieci współzawodniczących do generowania etykietowanych danych poprzez zaprojektowanie odpowiedniej architektury sieci GAN i nauczeniu modelu. Następnie przeprowadzenie ewaluacji jakości danych generowanych przez sieć GAN, między innymi poprzez stworzenie prostego modelu klasyfikacyjnego do sprawdzenia czy dodanie danych syntetycznych do zbioru treningowego składającego się z danych rzeczywistych będzie w stanie poprawić wyniki działania modelu. Jako dane w pracy użyte zostaną dane medyczne, a konkretnie zdjęcia rentgenowskie płuc podzielone na dwie kategorie: zdrowe oraz z zapaleniem płuc. Pozwoli to zobrazować jak ważną rolę sieci GAN mogą odgrywać w niezwykle ważnych i kluczowych dziedzinach, takich jak wspomniana medycyna.

Finalnym efektem pracy jest dokonanie ewaluacji wygenerowanych danych etykietowanych, przy użyciu odpowiedniego wariantu sieci GAN, pod względem ich jakości i różnorodności oraz sprawdzenie czy ich wykorzystanie do uczenia modelu klasyfikacyjnego przyniesie pozytywne rezultaty.

1.2. Zawartość pracy

Struktura pracy jest następująca. W rozdziale 2 przedstawiony został problem danych etykietowanych. Poruszone zostały tematy: pozyskiwania danych etykietowanych, metod ich generowania, roli sieci GAN w tym procesie oraz wyzwania i ograniczenia. W rozdziale 3 szczegółowo omówione zostały sieci GAN. Przedstawiono ich cele, sposób działania, architekturę, wyzwania z nimi związane oraz różne rodzaje generatywnych sieci współzawodniczących. Rozdział 4 zawiera opis użytych technologii, przygotowania danych oraz implementacji sieci GAN. W rozdziale tym został także krótko opisany model klasyfikacyjny wykorzystany w procesie ewaluacji wyników. Metody ewaluacji obrazów syntetycznych wygenerowanych przez sieć GAN wraz z wynikami i ich analizą zostały przedstawione w rozdziale 5. W ostatnim rozdziale 6 zostały przedstawione wnioski wynikające z pracy wraz z potencjalnymi kierunkami rozwoju.

2. Problem danych etykietowanych

W rozdziale tym zostanie omówione czym są dane etykietowane oraz jakie trudności można napotkać podczas ich pozyskiwania. Zostaną także opisane metody generowania danych etykietowanych, rola sieci GAN w tym procesie oraz potencjalne wyzwania i ograniczenia.

2.1. Wprowadzenie do danych etykietowanych

W dziedzinie uczenia maszynowego jakość oraz ilość dostępnych danych odgrywa kluczową rolę. Szczególnie w specyficznych dziedzinach, takich jak na przykład medycyna, problem niedoboru danych etykietowanych staje się jeszcze bardziej istotny. Dzieje się tak, ponieważ dane te są wrażliwe i często trudno dostępne, co może skutkować zbyt małym zbiorem danych do odpowiedniego nauczenia konkretnego modelu.

Danymi etykietowanymi nazywane są zestawy danych, w których każdy pojedynczy element posiada odpowiednią etykietę, która opisuje jego klasyfikację lub kategorię w kontekście aktualnie analizowanego problemu. Przykładem takich danych mogą być zdjęcia znaków drogowych, gdzie każdy obiekt będzie miał przypisaną informację jaki znak znajduje się na zdjęciu, na przykład "zakaz wjazdu"czy "stop".

Etykietowane dane posiadają zastosowania w problemach:

- trenowania modeli - dane etykietowane stanowią fundament trenowania podczas uczenia nadzorowanego. Stosując dane wejściowe wraz z odpowiednimi etykietami model jest w stanie nauczyć się wzorców oraz relacji w danych. Dzięki temu nauczony model jest w stanie dokonywać poprawnych i dokładnych predykcji dla nowych, nie widzianych wcześniej przez niego danych.

- ewaluacji jakości modelu - posiadanie etykiet do danych znacząco pomaga w ocenie wydajności modelu. Po nauczeniu modelu na danych treningowych, można użyć danych testowych z etykietami, aby sprawdzić, jak dobrze model dokonuje predykcji w porównaniu z prawdziwymi etykietami.

- zapewnienie odpowiedniej jakości danych - dane z etykietami gwarantują wykorzystanie informacji wysokiej jakości podczas trenowania modeli uczenia maszynowego, co jest możliwe dzięki precyzyjnym etykietom. Jest to szczególnie ważne w dziedzinach takich jak medycyna, gdzie najmniejszy błąd może mieć bardzo poważne konsekwencje.

– poprawienie czasu uczenia - dane etykietowane często przyspieszają proces uczenia, ze względu na to, iż dostarczają modelowi jasne informacje co do tego, jakie wyniki powinien generować, w przeciwieństwie do uczenia nienadzorowanego.

W odniesieniu do dziedziny takiej jak medycyna, w której dokładność oraz pewność odrywają kluczową rolę, dane etykietowane są niesamowicie ważnym zasobem. Posiadanie ich pozwala na rozwój algorytmów, które mogą być w stanie pomóc lekarzom w diagnozowaniu chorób, czy nawet częściowej automatyzacji procesu opieki medycznej. Jednakże uzyskanie wysokiej jakości danych etykietowanych w medycynie jest trudnym i kosztownym zadaniem, co jeszcze bardziej zwiększa potrzebę na znajdywanie coraz to lepszych metod generowania takich danych.

2.2. Wyzwania w pozyskiwaniu danych etykietowanych

Uzyskanie wystarczająco dużej ilości wysokiej jakości danych etykietowanych często bywa dużym wyzwaniem. Dzieje się tak ze względu między innymi na: koszty związane z ręcznym etykietowaniem danych, problemy z prywatnością i etyką, czy ograniczoną dostępnością specyficznych danych.

Ręczne etykietowanie danych jest bardzo kosztownym procesem zarówno pod kątem czasu, jak i potrzebnych środków finansowych. W dziedzinie takiej jak medycyna, gdzie dokładność danych jest kluczowa, konieczne jest wykorzystanie ekspertów do dokładnej klasyfikacji i etykietowania danych. Na przykład dla obrazów medycznych, aby ręcznie przypisać etykiety do danych, specjaliści muszą spędzić wiele godzin na dokładne przeanalizowanie danych i ich odpowiednim oznaczeniu. Przeanalizowanie tysięcy takich obrazów wiąże się ze znacznymi kosztami finansowymi, które mogą nawet zahamować dalsze postępy technologiczne w danej dziedzinie.

Prywatność, prawo i kwestie etyczne pojawiają się w dyskusjach na temat zbierania i wykorzystywania danych. Szczególnie w dziedzinach tak wrażliwych jak medycyna. W wielu krajach dane pacjentów podlegają ścisłej ochronie, co powoduje, że nie mogą zostać wykorzystane bez odpowiedniej zgody. Pojawia się ryzyko naruszenia prywatności osób, gdy dane nie są odpowiednio anonimizowane. Często zbieranie danych, nawet do celów badawczych, budzi spore kontrowersje i wymaga starannego i dokładnego podejścia pod względem etycznym, czy prawnym.

Dane etykietowane również potrafią być często mało dostępne. W dziedzinach, takich jak medycyna, występują przypadki, które mają bardzo ograniczone występowanie. W przypadku rzadkich chorób znalezienie wystarczającej liczby danych do stworzenia odpowiednio dokładnego modelu może być bardzo trudnym wyzwaniem. Dodatkowo niektóre jednostki lub organizacje mogą posiadać dane wyłącznie dla siebie lub dane mogą być mocno poufne, co znacząco ogranicza możliwość trenowania modeli przez większą grupę badaczy. Takie sytuacje często występują w bardzo specyficznych i niszowych obszarach danej dziedziny.

2.3. Metody generowania danych etykietowanych

W obliczu rosnących wymagań w kontekście ilości i jakości danych do uczenia maszynowego, znalezienie efektywnych metod do generowania danych etykietowanych oraz radzenie sobie z ograniczoną liczbą danych z etykietą stało się kluczowe. Opracowanych zostało wiele metod, takich jak:

– tradycyjne metody augmentacji danych,

– *active learning*,

– uczenie częściowo nadzorowane,

– techniki oparte na przekształceniach,

– zastosowanie generatywnych sieci współzawodniczących.

Tradycyjne metody augmentacji polegają na wprowadzeniu niewielkich modyfikacji do istniejących danych i dodaniu ich do zbioru danych z taką samą etykietą jak w próbce bazowej. Do tradycyjnych metod augmentacji w kontekście przetwarzania obrazów można zaliczyć:

– rotacje, czyli obrócenie obrazu o zadaną liczbę stopni,

– skalowanie, czyli powiększenie lub pomniejszenie wymiarów obrazu,

– deformacje, czyli zdeformowanie, na przykład poprzez rozciąganie, kilku fragmentów obrazu,

– przesunięcie polegające na przesuwaniu obrazu w poziomie lub pionie o pewną wartość,

– dodanie losowego szumu do obrazu,

– przycinanie, czyli wycięcie części obrazu i przeskalowanie jej do pierwotnego wymiaru,

– odbicie w poziomie lub pionie,

– *Random Erasing* oznaczający proces usuwania z obrazu losowych fragmentów i zastępowaniu ich losowym szumem,

– zmiana jasności lub koloru (na przykład na czarno-biały),

– powiększanie lub pomniejszanie fragmentów obrazów.

Rys. 2.1. Przykład zastosowania kilku metod augmentacji danych (rotacja, odbicie, skalowanie i zmiana jasności). Źródło: [2].

Metody te mogą poprawić dokładność modelu, szczególnie przy bardzo niewielkiej liczbie danych. Jednak często nie są one wystarczające i potrzebne są bardziej skomplikowane, lepsze jakościowo metody generowania nowych danych etykietowanych. Przykładowo, w medycynie, proste transformacje nie są w stanie uwzględnić czynników, takich jak, rozmiar, lokalizacja, kształt, czy wygląd konkretnej patologii [3].

Jedną z metod wspomagającą ręczne etykietowanie danych jest *active learning*. Metoda ta polega na priorytetyzowaniu danych do etykietowania w taki sposób, aby miały one jak największy wpływ na efektywność uczenia modelu. Stosuje się ją przy dużych zbiorach danych bez etykiet, aby w pierwszej kolejności etykietować dane, które dostarczą jak najwięcej informacji do modelu uczenia maszynowego. Zaletą tej metody jest zmniejszenie potrzebnej liczby ręcznie etykietowanych danych do osiągnięcia satysfakcjonującej dokładności modelu uczenia maszynowego [4].

Uczenie częściowo nadzorowane (ang. *semi-supervised learning*) stosowane jest, gdy tylko niewielka część używanych danych ma etykietę. Metoda ta polega na wykorzystaniu zarówno danych etykietowanych jak i nieetykietowanych do uczenia modelu. W najbardziej podstawowej wersji tego podejścia model początkowo uczony jest na danych treningowych posiadających etykietę. Następnie, przy pomocy nauczonego modelu, etykietuje się pewną porcję danych bez etykiety i dodaje się je do zbioru danych treningowych przed następnym uczeniem modelu. Cały proces powtarzany jest iteracyjne, aż do osiągnięcia satysfakcjonujących wyników. Uczenie częściowo nadzorowane może być skutecznie łączone z metodą *active learning*, aby osiągnąć jeszcze lepsze wyniki [5].

Jedną z technik generowania nowych danych opartej na przekształceniach jest transfer stylu polegający na przenoszeniu stylistycznych cech jednego obrazu na inny obraz, jednocześnie zachowując jego oryginalną zawartość. Metoda ta pozwala na generowanie nowych wariacji istniejących obrazów i opiera się na wykorzystaniu głębokich sieci neuronowych. Technika ta często oparta jest na generatywnych sieciach współzawodniczących, których udział w generowaniu etykietowanych danych został szerzej opisany w rozdziale 2.4.

2.4. Rola sieci GAN w generowaniu danych etykietowanych

Generatywne sieci współzawodniczące wprowadzone zostały w roku 2014 w publikacji [6]. Od tamtego momentu sieci GAN stały się ważnym narzędziem w dziedzinie głębokiego uczenia. Generatywne sieci współzawodniczące są kluczowym narzędziem do generowania danych etykietowanych. W niniejszym rozdziale przedstawione zostanie kilka istotnych publikacji związanych z sieciami GAN w kontekście problemu generowania danych etykietowanych.

Pierwszą z publikacji jest [7]. Artykuł ten opisuje wykorzystanie sieci StackGAN w celu generowanie realistycznych obrazów z tekstu. Autorzy wykorzystali dwuetapową strukturę, gdzie w pierwszym etapie model przyjmując opis tekstowy generował obrazy o niskiej rozdzielczości. W drugim etapie model miał za zadanie, przyjmując obraz wygenerowany w etapie pierwszym oraz opis tekstowy, generowanie obrazów o wysokiej rozdzielczości i fotorealistycznych detalach. Publikacja prezentuje imponujące

wyniki, ponieważ obrazy generowane przez sieć StackGAN są niemal nie do odróżnienia od prawdziwych zdjęć. Mimo, iż sieć wykorzystana w tym artykule skupia się na generowaniu obrazów za pomocą opisów tekstowych, to obrazuje również możliwości w kontekście generowania danych etykietowanych przy pomocy architektury przedstawionej w publikacji. Sieć StackGAN mogłaby zostać odpowiednio zaadaptowana, w innym kontekście, aby dzięki podaniu odpowiedniej etykiety generować obrazy na jej podstawie.

W publikacji [8] przedstawiona została nowatorska metoda RenderGAN do generowania danych. Artykuł podejmuje problem generowania oznaczeń pszczół (znaki przypominające kody kreskowe), których zbiór etykietowanych danych jest bardzo mało liczny. Główną ideą RenderGAN jest wykorzystanie modelu 3D do generowania podstawowych obrazów, do którego parametry oznaczeń pszczół (pozycja, orientacja i konfiguracja bitów) są przewidywane przez generator z wektora szumu. Następnie obrazy są udoskonalane i dopasowywane do rzeczywistych danych za pomocą sekwencyjnie zastosowanych funkcji augmentacji (rozmycie, zmiana jasności, dodanie tła i detali). Autorom udało się wytrenować model, który był zdolny do generowania obrazów trudnych do odróżnienia od prawdziwych. Sieć była w stanie nauczyć się skomplikowanych struktur z danych wejściowych nieposiadających etykiet, co bardzo dobrze obrazuje możliwości sieci GAN do zastosowania ich w najróżniejszych dziedzinach i problemach.

Kolejną publikacją jest [9]. Artykuł ten stanowi bardzo ważny wkład w dziedzinę analizy obrazów medycznych, a w szczególności ich generowaniu. W publikacji autorzy podjęli problem klasyfikacji zmian w wątrobie przy użyciu głębokich sieci konwolucyjnych. Ze względu na bardzo ograniczony zbiór danych autorzy wykorzystali sieć GAN do wygenerowania dodatkowych danych syntetycznych, aby zwiększyć dokładność modelu. Udało im się wygenerować obrazy o odpowiedniej jakości i jednoczesnym zachowaniu cech kluczowych dla danych rodzajów zmian wątroby. Po dołączeniu danych syntetycznych do zestawu obrazów treningowych klasyfikatora, udało im się poprawić wyniki działania modelu w porównaniu do wyników dla samych danych prawdziwych. Publikacja ta pokazała jak bardzo pomocne mogą być generatywne sieci współzawodniczące w wygenerowaniu dodatkowych danych syntetycznych. Szczególnie, gdy zbiór prawdziwych danych jest ubogi, a tradycyjne metody augmentacji danych nie dostarczają wystarczająco informacji do modelu, aby osiągnąć satysfakcjonujące wyniki.

Podczas analizy literatury naukowej na temat generatywnych sieci współzawodniczących, a szczególnie w kontekście generowania danych etykietowanych można zauważyć, iż jest to ogromna innowacja w kierunku tworzenia coraz to bardziej realistycznych i użytecznych danych. Jest to szczególnie przydatne, gdy pierwotny zbiór danych realnych jest zbyt mały do nauczenia modelu, aby osiągał on satysfakcjonujące wyniki. Warto również zauważyć, iż sieci GAN znajdują swoje zastosowanie również w specjalistycznych i niezwykle ważnych dziedzinach takich jak medycyna. Techniki te są z roku na rok rozwijane i ulepszane, aby móc osiągać coraz to lepsze rezultaty.

2.5. Potencjalne wyzwania i ograniczenia

Generowanie syntetycznych danych etykietowanych przy użyciu generatywnych sieci współzawodniczących stwarza wiele nowych możliwości dla postępu w dziedzinie uczenia maszynowego. Jednak jak każda innowacyjna technologia, również sieci GAN niosą ze sobą pewne wyzwania i ograniczenia, takie jak:

– jakość generowanych danych,

– ryzyko nadmiernego dopasowania,

– problemy etyczne i prawne.

Głównym wyzwaniem w generowaniu syntetycznych danych jest zapewnienie im odpowiednio wysokiej jakości. Mimo tego, iż sieci GAN są zdolne do generowania realistycznych danych, to jednak nie zawsze są one wystarczająco dokładne z perspektywy najbardziej istotnych informacji zawartych w obrazie. Przykładowo wygenerowany obiekt może zawierać nieistotne lub mylące cechy, które mogą skutkować tym, iż model uczenia maszynowego będzie szkolony na błędnych danych. Wykorzystanie modelu nauczonego na danych złej jakości może mieć bardzo złe skutki. Na przykład, w dziedzinie medycyny, skutkiem może być błędna diagnoza, co może wydłużyć czas hospitalizacji pacjenta.

Korzystanie z syntetycznych danych podczas uczenia modelu może prowadzić do problemu nadmiernego dopasowania, zwłaszcza jeśli wygenerowane dane są bardzo podobne do siebie. Problem ten cechuje się tym, iż model osiąga bardzo dobre wyniki na danych treningowych, ale nie radzi sobie efektywnie z rzeczywistymi, nie widzianymi przez niego wcześniej, danymi.

Wykorzystanie danych syntetycznych może również nieść ze sobą wiele dyskusji na temat kwestii etycznych, czy prawnych. Istnieje wiele regulacji prawnych, które mogą w różny sposób ograniczać komercyjne wykorzystanie sztucznie wygenerowanych danych.

3. Sieci GAN

W niniejszym rozdziale zostaną przedstawione generatywne sieci współzawodniczące, sposób ich działanie, architektura i funkcja straty. Omówione będą również typowe problemy i wyzwania związane z sieciami GAN oraz różne rodzaje generatywnych sieci współzawodniczących.

3.1. Wprowadzenie do generatywnych sieci współzawodniczących

Uczenie głębokie jest podzbiorem uczenia maszynowego, który używa sztucznych sieci neuronowych w celu symulowania zachowania ludzkiego mózgu. Sieci te są uczone z danych wejściowych, wykorzystując wiele warstw i różnych technik, by wyciągnąć z nich wzorce i zależności w celu nauczenia się ich, aby móc je wykorzystać na przykład do predykcji, podejmowania decyzji czy nawet generowania nowych danych.

Deep learning znajduje zastosowanie w problemach:

- przetwarzania obrazów,
- analizy danych tekstowych,
- systemów rekomendacji,
- analizy danych dźwiękowych,
- generowania nowych danych.

Możemy wyróżnić wiele różnych rodzajów sztucznych sieci neuronowych:

- sieci jednokierunkowe,
- sieci rekurencyjne,
- sieci konwolucyjne,
- sieci syjamskie,
- autoenkodery,
- generatywne sieci współzawodniczące.

Sieci GAN odgrywają bardzo znaczącą rolę w dzisiejszym świecie, a ich popularność znacząco wzrosła w ostatnim czasie. Cała koncepcja tych sieci polega na zbudowaniu dwóch sieci neuronowych: generatora i dyskryminatora, które rywalizują ze sobą. Można je porównać do oszusta (generator), który tworzy fałszywe pieniądze i policjanta (dyskryminator). Zadaniem oszusta jest nauczenie się tworzyć tak dobre pieniądze, aby zmylić policjanta, którego rolą jest wykrywanie fałszywych pieniędzy. W taki właśnie sposób sieci te wzajemnie się uczą i stają się coraz to lepsze, aż dojdą do momentu kiedy generator jest w stanie generować obiekty, które stają się nie do odróżnienia od prawdziwych dla dyskryminatora.

Generatywne sieci współzawodniczące posiadają wiele różnych zastosowań w dziedzinie uczenia maszynowego i przetwarzania danych. Przykładami zastosowania sieci GAN jest:

- generowanie nowych obrazów (na przykład ludzkiej twarzy),
- generowanie obrazów z tekstu,
- zwiększanie rozdzielczości obrazów, czy filmów,
- ulepszanie starych nagrań poprzez dodanie kolorów, zwiększenie rozdzielczości i klatek na sekundę,
- generowanie odpowiedzi na podstawie zadanego pytania i kontekstu,
- transformacje obrazów (na przykład zamiana konia w zebrę)
- generowanie muzyki,
- przenoszenie stylu z jednego obrazu na drugi (na przykład stylu malarstwa znanych artystów, czego przykład można zaobserwować na rysunku 3.1),
- generowanie obrazów medycznych.

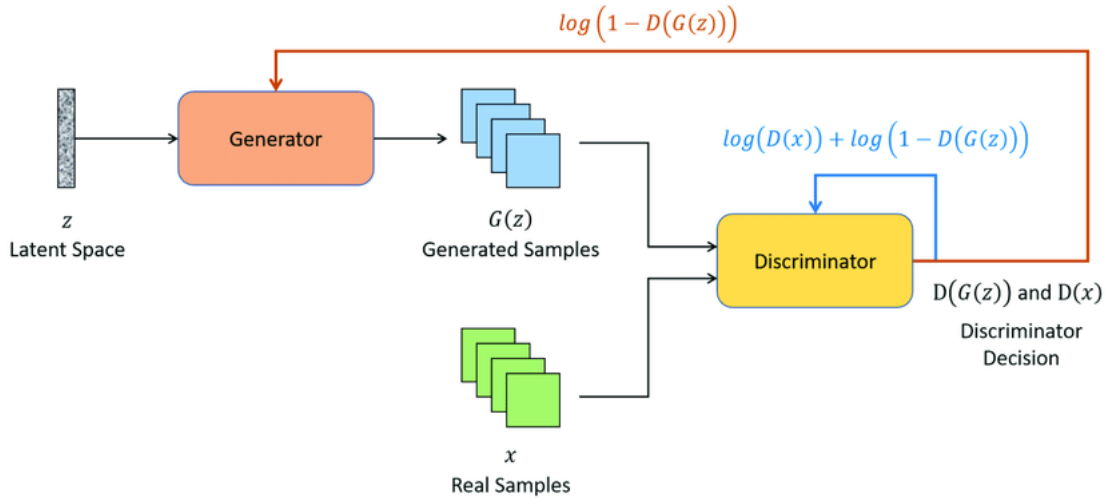
Rys. 3.1. Przykład wyniku sieci GAN do transferu stylu znanych artystów na zwykłym zdjęciu. Źródło: [10].

3.2. Proces trenowania sieci GAN

Dyskryminator oraz generator są uczone od zera, czyli obie sieci są losowo inicjalizowane na starcie. Typowe kroki podczas uczenia generatywnej sieci współzawodniczącej [6]:

1. Generator na wejściu przyjmuje losowy wektor, zazwyczaj o rozkładzie normalnym.
2. Następnie wejściowy szum przechodzi przez różne warstwy sieci i jest przez nie transformowany. Warstwy te mają na celu przetransformować go w taki sposób, aby nadać mu cechy charakterystyczne dla danych realnych. Na wyjściu generatora otrzymywane są wygenerowane dane syntetyczne.
3. Kolejnym krokiem jest przekazanie wygenerowanych danych do dyskryminatora, który dokonuje oceny autentyczności.
4. Następnie obliczana jest funkcja straty, która w dalszej kolejności przekazywana jest przy pomocy propagacji wstecznej do generatora i dyskryminatora. W procesie tym obliczane są gradienty funkcji straty dla odpowiednich wag i parametrów sieci. Etap ten pozwala na aktualizację wag i parametrów modeli. Dzięki niej generator jest w stanie generować coraz lepsze dane, natomiast dyskryminator jest w stanie dobrze rozróżniać dane prawdziwe od syntetycznych.
5. Wszystkie powyższe kroki powtarzane są iteracyjne, aż generator będzie w stanie generować dane nie do odróżnienia przez dyskryminator i zmusi go do zgadywania podczas rozróżniania danych prawdziwych i wygenerowanych przez sieć.

Kluczową rolę podczas uczenia sieci GAN odgrywa odpowiednia optymalizacja funkcji straty przez obie sieci, co dokładnie zostało opisane w rozdziale 3.3.



Rys. 3.2. Architektura podstawowej sieci GAN. Źródło: [11].

3.3. Funkcja straty w GAN

Funkcja straty w generatywnych sieciach współzawodniczących odgrywa kluczową rolę. Jest bardzo ważnym czynnikiem podczas trenowania dyskryminatora i generatora, ponieważ odpowiada za ocenę jakości danych generowanych przez sieć. Zgodnie z [6], funkcja straty opisana jest następującą zależnością:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{3.1}$$

gdzie:

- G – generator,
- D – dyskryminator,
- z – losowy szum, który podawany jest na wejście generatora,
- x – dane rzeczywiste,
- $G(z)$ – dane wygenerowane przez generator,
- $D(x)$ – wynik dyskryminatora dla danych prawdziwych,
- $D(G(z))$ – wynik dyskryminatora dla danych syntetycznych,
- $p_{\mathbf{data}}(\mathbf{x})$ – rozkład danych treningowych
- $p_{\mathbf{z}}(\mathbf{z})$ – rozkład losowego wektora z.

W podstawowej sieci GAN wykorzystywana jest funkcja straty binarnej entropii krzyżowej (BCE, ang. *binary cross entropy*), która w postaci ogólnej może być zapisana następująco [12, 13]:

$$L(y,\hat{y}) = -\left[y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right] \tag{3.2}$$

gdzie:

- y – etykieta rzeczywista (binarna),
- $\hat{y}$ – wyjściowe prawdopodobieństwo przynależności do klasy pozytywnej.

Dyskryminator jako wartości wejściowe może przyjmować dane prawdziwe (1) oraz fałszywe (0), natomiast wartość wyjściową sieci $\hat{y}$ można zapisać jako $D(x)$, czyli wynik dyskryminatora dla danych prawdziwych. Dla wartości prawdziwych y=1, co sprawia, że funkcja straty BCE będzie wyglądać w następujący sposób [13, 6]:

$$L(1,\hat{y}) = -\log(D(x)) \tag{3.3}$$

Natomiast dla danych syntetycznych y=0, a wartość wyjściową $\hat{y}$ można zapisać jako $D(G(z))$, czyli wynik dyskryminatora dla danych syntetycznych. Teraz więc funkcja straty wygląda w następujący sposób [13, 6]:

$$L(0, \hat{y}) = -\log(1 - D(G(z))) \tag{3.4}$$

Celem dyskryminatora jest jak najlepsze rozróżnienie danych prawdziwych od syntetycznych. Funkcję straty dyskryminatora można więc zapisać jako [13, 6]:

$$L(D) = -\left[\log(D(x)) + \log(1 - D(G(z)))\right] \tag{3.5}$$

Dyskryminator więc będzie dążył do zminimalizowania powyższej funkcji straty. Dla uproszczenia można pozbyć się minusów i w takim wypadku sieć będzie miała na celu maksymalizację wartości funkcji straty [13, 6].

$$L(D) = max\left[log(D(x)) + \log(1 - D(G(z)))\right] \tag{3.6}$$

Natomiast generator korzysta z funkcji straty dyskryminatora. Jego celem jest, aby dyskryminator zaczął klasyfikować dane syntetyczne jako prawdziwe. Będzie on więc dążył do minimalizacji następującego wyrażenia [13, 6]:

$$L(G) = min\left[\log(1 - D(G(z)))\right] \tag{3.7}$$

Stąd funkcje straty sieci GAN dla pojedynczej danej wejściowej można zapisać jako [13, 6]:

$$L(GAN) = GminDmax\left[log(D(x)) + \log(1 - D(G(z)))\right] \tag{3.8}$$

Generator będzie dążył do minimalizacji powyższej funkcji straty, natomiast dyskryminator do jej maksymalizacji. W taki właśnie sposób sieci będą ze sobą rywalizować, aby osiągnąć korzystne dla siebie wartości.

W praktyce, aby uniknąć problemu nienasycającego się gradientu (ang. *non-saturating gradient*), polegającego na tym, że gradient funkcji nie zmierza do zera w trakcie trenowania sieci, funkcję straty generatora można zapisać jako maksymalizacje poniższego wyrażenia [6]:

$$L(G) = max\left[\log(D(G(z)))\right] \tag{3.9}$$

3.4. Wyzwania i problemy w sieciach GAN

Pomimo tego, iż sieci GAN są bardzo dobrym narzędziem w generowaniu danych, to wiążą się z nimi również pewne problemy i wyzwania takie jak:

– nienasycający się gradient wspomniany w rozdziale 3.3,

– zanikający gradient (ang. *vanishing gradient*),

– załamanie trybu (ang. *mode collapse*),

– oscylacje strat,

– odpowiedni dobór hiperparametrów.

Zanikający gradient odnosi się do sytuacji, w której podczas propagacji wstecznej gradient staje się bardzo mały lub maleje do zera. Prowadzi to do uniemożliwienia efektywnego aktualizowania wag i parametrów modelu podczas uczenia.

Załamanie trybu zdarza się, gdy generator znajdzie małą liczbę próbek, które są w stanie oszukać dyskryminator. Wtedy generator nie jest w stanie wygenerować danych innych niż z tego zbioru, a więc nie będzie on w stanie uczyć się i generować realistycznych danych [14].

Oscylacje wartości funkcji straty generatora i dyskryminatora mogą zacząć gwałtownie drgać zamiast się stabilizować. Powinny istnieć niewielkie oscylacje funkcji straty, a w dłuższej perspektywie powinny one dążyć do względniej stabilizacji i stopniowego zmniejszania lub zwiększania się, a nie do chaotycznej zmiany wartości [14].

Dobór hiperparametrów sieci GAN nie jest prostym zadaniem. Sieci te są bardzo czułe na ich zmiany, a więc odpowiedni ich wybór jest kluczowy, aby sieć mogła osiągać pożądane wyniki. Z uwagi na dużą liczbę parametrów modelu w generatywnych sieciach współzawodniczących ich dobór jest sporym wyzwaniem i często przypomina on metodę "prób i błędów", a pomocne w tym procesie jest bardzo dobre zrozumienie jak działają sieci GAN.

W celu poradzenia sobie z powyższymi problemami i wyzwaniami opracowane zostały inne rodzaje sieci GAN, które modyfikują podstawową sieć GAN. Pozwalają one na zwalczanie typowych problemów dla sieci GAN, a przykładami najpopularniejszych takich sieci są na przykład WGAN oraz WGAN-GP, które zostaną omówione w następnym rozdziale.

3.5. Rodzaje sieci GAN

W ostatnich latach powstało wiele rozszerzeń tradycyjnych sieci GAN. Można wyróżnić między innymi następujące rodzaje sieci GAN:

– podstawowa sieć GAN (ang. *vanilla GAN*) omówiona już w powyższych rozdziałach,

– DCGAN

– Warunkowa sieć GAN

– WGAN

– WGAN-GP

– pix2pix

– CycleGAN

– StyleGAN

3.5.1. DCGAN

Sieć DCGAN (ang. *Deep Convolutional Generative Adversarial Network*), czyli głęboka konwolucyjna generatywna sieć współzawodnicząca jest rozszerzeniem tradycyjnej sieci GAN o wykorzystanie sieci konwolucyjnych, w generatorze i dyskryminatorze. Są one niezwykle przydatne w przetwarzaniu i generowaniu obrazów o bardzo wysokiej jakości.

Sieci konwolucyjne (CNN, ang. *convolutional neural network*) w porównaniu do tradycyjnych sztucznych sieci neuronowych są używane głównie w dziedzinie przetwarzania obrazów. Pozwalają one zakodować specyficzne cechy dla danego obrazu w architekturze sieci, co pozwala na wykrywanie wzorców w danym obrazie [15].

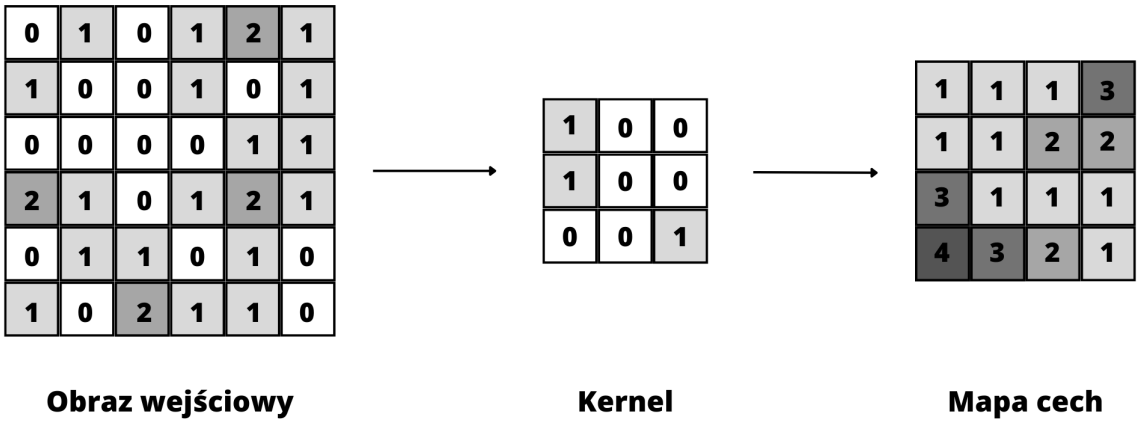
Sieci CNN zazwyczaj składają się z trzech rodzajów warstw:

– warstwy konwolucyjnej wraz z funkcją aktywacji ReLu (ang. *convolutional layer*),

– warstwy łączenia (ang. *pooling layer*),

– warstwy w pełni połączonej (ang. *fully-connected layer*).

Warstwa konwolucyjna odgrywa kluczową rolę w sposobie działania CNN. Kluczowym elementem tej warstwy są możliwe do nauczenia jądra (ang. *kernels*), czyli zwykle małe macierze, których wartości mogą na przykład zostać zainicjalizowane losowo. Kernel przechodzi po całym obrazie wejściowym i w każdym kroku oblicza iloczyn skalarny dwóch macierzy i zapisuje wynik tej operacji do nowej macierzy cech. Przykład tej operacji został przedstawiony na rysunku 3.3 . Każda warstwa konwolucyjna może posiadać kilka jąder z różnymi wartościami.



Rys. 3.3. Przykład operacji konwolucji. Źródło: opracowanie własne.

W warstwie konwolucyjnej można wyróżnić trzy kluczowe parametry dla operacji konwolucji:

– krok (ang. *stride*),

– dopełnienie (ang. *padding*).

– rozmiar jądra (ang. *kernel size*)

Pierwszy z nich określa krok z jakim kernel przemieszcza się po obrazie wejściowym. Na rysunku 3.3 krok wynosił 1. Wartość parametru stride równa 2 oznacza, że filtr przemieszcza się co dwa piksele. Skutkuje to zmniejszeniem wymiarów obrazu wejściowego o połowę.

Drugi z parametrów, czyli padding odpowiada za dodanie dodatkowych pikseli na zewnątrz obrazu wejściowego przed operacją konwolucji. Na rysunku 3.3 nie zastosowano dopełnienia. Padding często używany jest, aby zachować pierwotne wymiary przestrzenne obrazu lub bardziej ogólnie mówiąc precyzyjnie kontrolować wymiary przestrzenne obrazu wyjściowego. Przykładowo dla obrazu wejściowego o wymiarze 64 na 64 piksele, gdy zastosowana zostanie operacja konwolucji z jądrem o rozmiarze 4 na 4 i krokiem 2, to bez dopełnienia wymiar obrazu wyjściowego wyniósłby 31 na 31 pikseli. Jednak zastosowanie dopełnienia o wartości 1 pozwala na otrzymanie obrazu wyjściowego o wymiarach 32 na 32 piksele. Zabieg ten jest bardzo pomocny i stosowany na przykład w sieciach generatywnych, i pozwala na przeskalowanie obrazu na przykład równo o połowę.

Ostatni z nich, czyli rozmiar jądra, określa jakich wymiarów jest opisany powyżej kernel, który przechodzi po obrazie wejściowym.

Ogólny wzór na wymiar wyjściowy po operacji konwolucji można zapisać w następujący sposób:

$$W_{out} = \frac{W_{in} + 2 \times \text{padding} - W_{kernel}}{\text{stride}} + 1 \tag{3.10}$$

gdzie:

W_{out} – wymiar wyjściowy obrazu,

W_{in} – wymiar wejściowy obrazu,

W_{kernel} – wymiar jądra.

Następnie na kernel aplikowana jest funkcja aktywacji, zazwyczaj ReLU, aby wprowadzić nieliniowość do architektury sieci. Funkcja ta dla wartości mniejszych od zera zwraca zero, natomiast dla wartości większych bądź równych od zera zwraca przyjmowaną wartość bez żadnych zmian.

Kolejną warstwą w sieci CNN jest warstwa łączenia używana do redukcji wymiaru mapy cech, aby zachować tylko najważniejsze jej elementy. Najczęściej stosowaną metodą jest *max pooling* i polega na przejściu przez macierz cech po obszarach o niewielkich rozmiarach (na przykład 2×2) i zachowanie tylko wartości maksymalnej w tym regionie. Operacja ta pozwala zachować najważniejsze informacje w macierzy cech i jednocześnie redukuje jej wymiarowość.

Następnie warstwy te są powtarzane, aby zbudować więcej abstrakcji w sieci, czyli wyodrębnić coraz to bardziej złożone i wysokopoziomowe cechy z danych wejściowych. Na samym końcu stosowane są warstwy w pełni połączone. Warstwa ta przekształca macierz cech na pojedynczy wektor cech. Jest zbudowana analogicznie do zwykłej warstwy w sieci jednokierunkowej.

W sieciach DCGAN zwykle nie stosuje się warstwy łączenia oraz warstwy w pełni połączonej. Głębokie konwolucyjne generatywne sieci współzawodniczące w całości składają się bloków konwolucyjnych, w których skład wchodzi warstwa konwolucyjna wraz z normalizacją oraz funkcją aktywacji. Zamiast używania warstwy pooling'u stosuje się konwolucje z większym krokiem w celu zmiany wymiaru obrazu. Strategia ta pozwala zachować więcej cech na obrazie oraz zapewnia bardziej stabilny proces uczenia sieci DCGAN. To samo dotyczy warstwy w pełni połączonej, zamiast której również stosuje się operację konwolucji z odpowiednimi parametrami. Strategia ta pozwala między innymi na zmniejszenie liczby parametrów i lepszą generalizację w porównaniu do zastosowania warstwy w pełni połączonej.

W dyskryminatorze używane są klasyczne warstwy konwolucji mające na celu redukcję wymiaru obrazu wejściowego pomiędzy kolejnymi blokami. Natomiast w generatorze stosowana jest konwolucja transponowana, której działanie można określić jako "odwrotną" konwolucję. Ma ona na celu zwiększanie wymiarowości generowanego obrazu pomiędzy kolejnymi blokami poprzez dodanie zer wokół każdego piksela, których liczba określona jest przez wartość kroku. Operacja ta odbywa się przed przejściem jądra po obrazie wejściowym. W taki właśnie sposób z wejściowego wektora szumu finalnie otrzymywany jest obraz wyjściowy o pożądanych wymiarach.

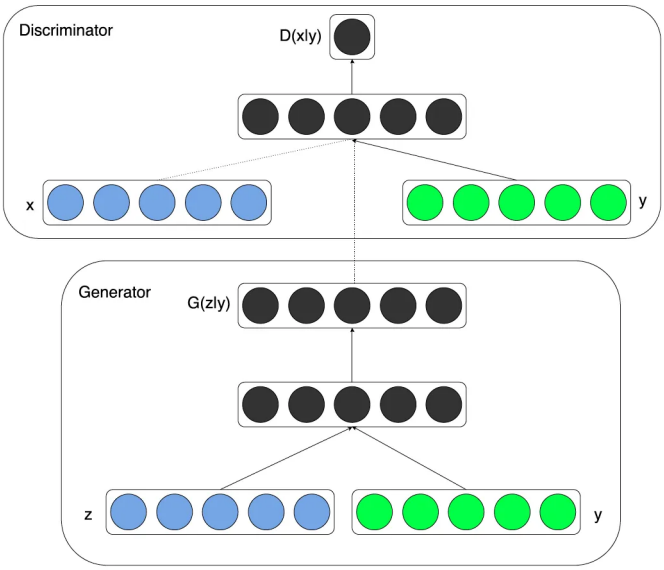
3.5.2. Warunkowa sieć GAN

Warunkowa sieć GAN jest rozszerzeniem standardowej sieci GAN poprzez dodanie do generatora i dyskryminatora dodatkowej informacji na wejściu. Może to być wszelkiego rodzaju informacja pomocnicza na przykład etykieta klasy. Informacja ta jest podawana do obu sieci jako dodatkowy element wejściowy, który łączony jest z danymi wejściowymi, a następnie całość traktowana jest jako dane wejściowe do obu sieci. Zostało to zobrazowane na rysunku 3.4. Funkcja straty dla warunkowej sieci GAN, zgodnie z oryginalnym artykułem o tej sieci, wygląda w następujący sposób [16]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x} \mid \mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z} \mid \mathbf{y})))] \tag{3.11}$$

gdzie:

y – dodatkowa informacja pomocnicza (na przykład etykieta klasy).



Rys. 3.4. Uproszczona wizualizacja warunkowej sieci GAN. Źródło: [16].

Warunkowa sieć GAN jest szczególnie przydatna w generowaniu etykietowanych danych. Główną zaletą wynikającą z jej stosowania jest zdolność do poprawy jakości generowanych obiektów w kontekście podanych etykiet lub informacji. Wprowadzenie dodatkowej informacji warunkowej na wejściu sprawia, iż modele są w stanie bardziej precyzyjnie kontrolować generowanie danych podczas uczenia oraz oferuje użytkownikowi możliwość generowania danych zgodnie z konkretnymi specyfikacjami, a nie losowo jak w przypadku zwykłej sieci GAN.

3.5.3. WGAN

Sieć WGAN, czyli Wasserstein GAN wprowadza kilka zmian do tradycyjnej sieci GAN, które poprawiają stabilność i jakość procesu uczenia sieci. Główną zmianą jest wprowadzenie nowej funkcji straty dla dyskryminatora i generatora. Zamiast binarnej entropii krzyżowej stosowana jest funkcja straty Wassersteina. W funkcji tej używane są etykiety -1 i 1 zamiast 0 i 1 jak w podstawowej sieci GAN. Dodatkowo wyjście generatora nie jest ograniczane do zakresu od 0 do 1 (nie stosuje się funkcji aktywacji sigmoid w ostatniej warstwie), co przekłada się na otrzymanie dowolnych wartości na wyjściu. Dyskryminator w sieci WGAN jest często nazywany krytykiem. Krytyk ma na celu zmaksymalizowanie różnicy pomiędzy jego przewidywaniami dla prawdziwych danych a przewidywaniami dla danych syntetycznych, natomiast generator dąży do minimalizacji tej różnicy. Dlatego właśnie funkcja straty sieci WGAN może zostać zapisana w następujący sposób:

$$L_{\text{Wasserstein}}(D,G) = DmaxGmin\left[\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}))]\right] \tag{3.12}$$

Jednak, aby uniknąć problemu nienasycającego się gradientu, funkcje straty dla generatora może zostać zapisana jako minimalizacja następującego wyrażenia:

$$L_{\text{Wasserstein}}(D, G) = Gmin\left[-\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}))]\right] \tag{3.13}$$

Funkcja straty Wassersteina może osiągać bardzo wysokie wartości. Dlatego też twórcy sieci WGAN zdecydowali się wprowadzić dodatkowe ograniczenie. Dyskryminator, czyli krytyk musi być ciągłą funkcją k-Lipschitz dla k=1, a jest tak kiedy dla dowolnych dwóch danych wejściowych x_1 i x_2 krytyk spełnia poniższą nierówność:

$$\frac{|D(x_1) - D(x_2)|}{|x_1 - x_2|} \leq 1 \tag{3.14}$$

gdzie:

$|D(x_1) - D(x_2)|$ – to bezwzględna różnica pomiędzy przewidywaniami dyskryminatora,

$|x_1 - x_2|$ – to bezwzględna różnica pomiędzy dwoma danymi wejściowymi (na przykład pomiędzy pikselami obrazów).

Zastosowanie funkcji straty Wassersteina pozbywa się zrównoważenia szkolenia generatora i dyskryminatora. Krytyk może być szkolony kilkukrotnie pomiędzy uczeniem generatora (na przykład 5 aktualizacji wag dyskryminatora do 1 generatora) [17] [14].

Do głównych zalet sieci WGAN należy poprawa stabilności procesu uczenia, co pomaga w redukcji problemów takich jak zanikający gradient. Dodatkowo dostarcza znacznie bardziej interpretowalną funkcję straty, która jest znacznie ściślej skorelowana z jakością generowanych obrazów. Jednak obcinanie wag nie jest najlepszym sposobem na wymuszenie ograniczenia Lipschitz'a i może prowadzić do pogorszenia zdolności modelu do uczenia się lub niestabilności treningu. Dlatego wprowadzono modyfikacje do sieci WGAN, które starają się rozwiązać ten problem inaczej niż poprzez proste obcinanie wag [17].

3.5.4. WGAN-GP

Sieć WGAN-GP wprowadza karę gradientu (GP, ang. *gradient penalty*), która zastępuje obcinanie wag i ma na celu kontrolowanie gradientów krytyka.

Kara gradientu wykorzystuje interpolację danych prawdziwych oraz wygenerowanych, korzystając z losowej liczby ϵ od 0 do 1, obliczaną w następujący sposób:

$$\hat{x} = \epsilon \cdot x_{real} + (1 - \epsilon)x_{fake} \tag{3.15}$$

gdzie:

$\hat{x}$ – zinterpolowane dane,

ϵ – losowa wartość od 0 do 1,

x_{real} – dane prawdziwe,

x_{fake} – dane wygenerowane.

Następnie zinterpolowane dane podawane są do krytyka, aby otrzymać wartości wyjściowe z krytyka dla danych mieszanych, z których obliczany jest gradient. Następnie obliczana jest norma L2 (norma euklidesowa) z tego gradientu. Idea kary gradientu polega na dążeniu do tego, aby obliczona norma była jak najbliższa wartości 1. Ze względu na to finalny wzór na karę gradientu wygląda następująco:

$$L_{GP} = \lambda \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \tag{3.16}$$

gdzie:

λ – współczynnik kary (dodatkowy hiperparametr modelu),

$\nabla_{\hat{x}} D(\hat{x})$ – gradient w odniesieniu do $\hat{x}$ z wartości wyjścia krytyka dla danych interpolowanych,

$\|...\|_2$ – norma L2.

Współczynnik kary służy do kontrolowania istotności kary gradientu w ogólnej funkcji straty. Całość podnoszona jest do kwadratu, aby nakładać jeszcze większą karę na krytyka im bardziej wartość normy odbiega od 1 [18]. Finalna wartość gradientu kary dodawana jest do funkcji straty używanej w sieci WGAN, przez co funkcja straty sieci WGAN-GP dla krytyka prezentuje się następująco:

$$L_{\text{Wasserstein-GP}}(D, G) = Dmax \left[\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[D(G(\mathbf{z}))] + L_{GP} \right] \tag{3.17}$$

Do głównych zalet sieci WGAN-GP zaliczana jest lepsza stabilność treningu w porównaniu do sieci WGAN oraz znacznie większa elastyczność ze względu na fakt, iż kara gradientu jest mniej restrykcyjna od obcinania wag. Jednak modyfikacja ta niesie także ze sobą kilka wyzwań takich jak: zwiększoną złożoność obliczeniową ze względu na operację obliczania gradientu oraz dodatkowy hiperparametr (współczynnik kary) do dostrojenia. Sieć WGAN-GP powinna być zdolna do generowania danych lepszej jakości w porównaniu do wersji bez kary gradientu [18].

3.5.5. Inne rodzaje sieci GAN

Sieć pix2pix to modyfikacja sieci GAN, która uczy się mapować obrazy z jednej domeny do obrazów z innej domeny korzystając ze sparowanych próbek danych. Sieć ta znajduje zastosowanie w zadaniach, które mają na celu przekształcenie jednego obrazu na inny, na przykład zdjęcie satelitarne na mapę konturową czy obrazu czarno-białego na kolorowy [19].

CycleGAN jest to sieć generatywna, która umożliwia wykonywanie przekształceń z obrazu na obraz w obu kierunkach, a nie w jednym jak w sieci pix2pix. Sieć ta korzysta z dwóch par składających się z generatora i dyskryminatora. Jedna z nich odpowiada za przekształcenia z obrazu A na B, natomiast druga para z B na A. W taki właśnie sposób sieci te uczą się wzajemnego przetwarzania obrazów [20].

StyleGAN jest to sieć GAN, która pozwala na generowanie obrazów bardzo dobrej jakości i jednocześnie posiadanie kontroli nad ich wyglądem. Idea tej sieci polega na manipulowaniu różnymi aspektami generowanych obrazów (na przykład kolorem czy stylem) [21].

4. Realizacja

W rozdziale tym przedstawione zostaną narzędzia i technologie wykorzystane w pracy. Zostanie także omówiona implementacja sieci GAN wraz z procesem uczenia oraz implementacja modelu wykorzystanego do ewaluacji danych syntetycznych.

4.1. Technologie i narzędzia

Podczas realizacji pracy wykorzystane zostały jedne z najpopularniejszych technologii i narzędzi w dziedzinie uczenia maszynowego, które zapewniły efektywny przebieg badań. Poniżej przedstawiony został krótki przegląd użytych technologii i narzędzi wraz z uzasadnieniem ich wyboru.

1. Jako główny język programowania do implementacji pracy wybrany został Python ze względu na jego ogromną wszechstronność oraz szeroki zasób bibliotek do analizy danych i uczenia maszynowego. Głównymi bibliotekami stosowanymi w pracy były: `NumPy`, `Matplotlib`, `Scikit-learn`, `SciPy` oraz `PyTorch`.

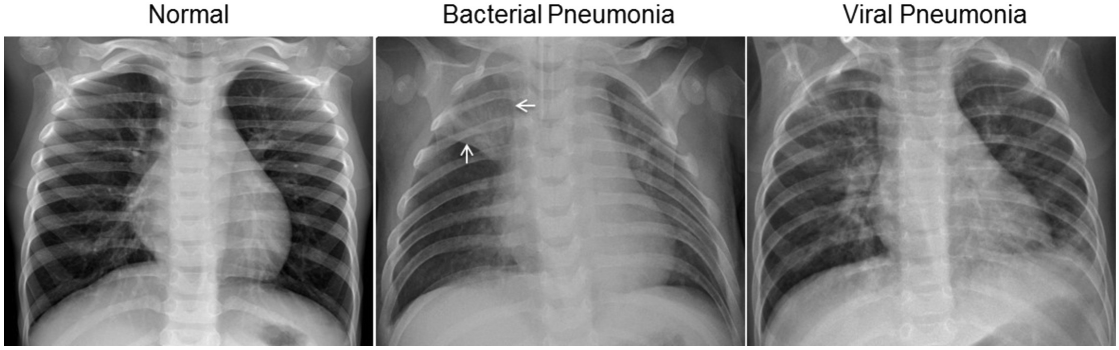
2. PyTorch jest to otwartoźródłowa biblioteka do uczenia maszynowego napisana w języku Python i stworzona przez oddział sztucznej inteligencji Facebook'a. Została ona użyta jako główna biblioteka do implementacji i uczenia sieci GAN. Charakteryzuje się ona wysoką elastycznością i modułową strukturą, co czyni ją szczególnie przydatną w badaniach naukowych. PyTorch pozwala na definiowanie dynamicznych obliczeń tensorowych, co jest szczególnie przydatne w projektowaniu i uczeniu głębokich sieci neuronowych.

3. Google Colab jest to środowisko Jupyter Nootebook, które działa w chmurze. Umożliwia pisanie i uruchamianie skryptów Pythona bezpośrednio w przeglądarce. Jest on przydatny w dziedzinie uczenia maszynowego, szczególnie głębokich sieci neuronowych, ponieważ oferuje korzystanie z mocnych GPU, co znacząco przyspiesza proces uczenia. Dlatego właśnie w pracy wykorzystano Google Colab w połączeniu z dyskiem Google, gdzie były przechowywane dane i wyniki. Pozwoliło to na nauczenie sieci GAN w dość przystępnym czasie, głównie dzięki korzystaniu z najmocniejszego GPU oferowanego przez Google Colab czyli `A100 GPU`.

,

4.2. Opis i przygotowanie danych

Dane wykorzystane w pracy pochodzą z platformy Kaggle [22], która jest popularnym źródłem zbiorów danych do wykorzystania w dziedzinie uczenia maszynowego i analizy danych. Tak, jak zostało to wspomniane we wstępie, zestaw danych zawiera zdjęcia rentgenowskie płuc, podzielone na dwie kategorie: zdrowe oraz z zapaleniem płuc. Obrazy zostały już podzielone na zbiór treningowy i testowy, co ułatwiło ich wczytanie i przetworzenie. Zbiór składa się łącznie z 5126 danych treningowych (1341 zdrowych i 3875 chorych) oraz 624 danych testowych (234 zdrowych i 390 chorych). Zestaw danych testowych został wykorzystany do ewaluacji modelu klasyfikacyjnego, a sieć GAN korzystała tylko z danych treningowych. Pobrany zbiór był już uprzednio sprawdzony pod względem jakości, wszystkie zdjęcia nieczytelne i o niskiej jakości zostały z niego usunięte, a diagnozy przypisane do poszczególnych obrazów sprawdzone przez specjalistów.



Rys. 4.1. Zdjęcia rentgenowskie klatki piersiowej: zdrowe (lewa strona), bakteryjne zapalenie płuc (środkowe) i wirusowe zapalenie płuc (prawa strona). Źródło: [23].

Na rysunku 4.1 można zaobserwować różnice pomiędzy zdjęciem rentgenowskim zdrowych płuc, a zdjęciem z zapalaniem płuc. Zdrowe zdjęcie rentgenowskie przedstawia czyste płuca bez obszarów nieprawidłowego zmętnienia. Bakteryjne zapalenie płuc (ang. *bacterial pneumonia*) charakteryzuje się jednym wyraźnym zmętnieniem, natomiast w przypadku wirusowego zapalenia płuc (ang. *viral pneumonia*) zmętnienia są bardziej rozproszone i rozsiane w obu płucach.

Dane zostały wczytane z dysku Google przy użyciu modułu `datasets` z biblioteki PyTorch. Przy wczytywaniu na dane zastosowane zostały następujące transformacje:

- konwersja zdjęć do skali szarości, aby uzyskany obraz był jednokanałowy,
- przeskalowanie obrazu do ustalonego wymiaru (w tym przypadku 128×128),
- konwersja obrazu do tensora (struktura danych używana w PyTorch),
- normalizacja obrazu do zakresu od -1 do 1.

Rozmiar obrazu został ustalony na 128×128 pikseli ze względu na to, iż przy większym rozmiarze czas uczenia sieci byłby zbyt duży.

Istotnym elementem w procesie przygotowania danych było zastosowanie zbalansowanego próbkowania, aby wyrównać liczbę danych z danej klasy w każdej partii danych. W tym celu wykorzystana została klasa `BalancedBatchSampler`, która grupuje zbiór danych według etykiet, a następnie losowo tworzy zbalansowane partie danych (rozmiar wsadu określony jest przy pomocy parametru `BATCH_SIZE`, który wynosi 128). Takie podejście gwarantuje równą liczbę obrazów z każdej klasy w każdej partii danych podczas uczenia modelu. Takie podejście pozwoli, aby model generował dane równie dobrej jakości dla obu etykiet.

Do zastosowania klasy `BalancedBatchSampler` i stworzenia Python'owego iteratora po zbiorze danych użyta została klasa `DataLoader` z PyTorch. Tak przygotowane i zapisane dane mogą już zostać wykorzystane do szkolenia modelu.

4.3. Implementacja modelu sieci GAN

4.3.1. Uzasadnienie wyboru wariantu sieci GAN

W pracy wybrana została architektura warunkowej głębokiej konwolucyjnej sieci współzawodniczącej Wassersteina z karą gradientu (ang. *Conditional DCWGAN-GP*). Taki wybór był w stanie połączyć zalety trzech rodzajów sieci GAN: warunkowej sieci GAN, DCGAN oraz WGAN-GP.

Zastosowanie warunkowości do sieci jest bardzo istotnym elementem w kontekście generowania danych etykietowanych. Dodanie etykiety do sieci pozwala na kontrolowanie wyjścia z generatora, aby tworzyć obrazy bazując na określonych etykietach i mieć pełną kontrolę nad generowanymi danymi.

Użycie warstw konwolucyjnych pozwoliło na generowanie obrazów realistycznych o wysokiej jakości. Warstwy te bardzo dobrze radzą sobie w identyfikowaniu różnych cech na obrazie, co w przypadku zdjęć rentgenowskich płuc było kluczowym elementem, aby móc wychwycić subtelne różnice pomiędzy obrazami. Konwolucja odpowiadała również za manewrowanie zmianą wymiarowości pomiędzy kolejnymi warstwami w generatorze i krytyku poprzez odpowiednim doborem parametrów takich jak: rozmiar jądra, krok oraz dopełnienie.

Wprowadzenie funkcji straty Wassersteina wraz z karą gradientu było również bardzo istotnym wyborem, ponieważ pozwoliło znacznie poprawić stabilność modelu oraz uniknąć problemu załamania trybu. Umożliwiło to także otrzymanie lepiej interpretowalnych wartości funkcji straty pod kątem jakości generowanych obrazów. Zastosowanie kary gradientu zamiast standardowego przycinania wag pozwoliło na uzyskanie jeszcze lepszej stabilności modelu.

4.3.2. Architektura krytyka

Schemat działania i architektura krytyka zostały zobrazowane na rysunku 4.2. Na wejściu sieć otrzymuje dwa tensory: obrazów oraz etykiet. Etykiety przekształcane są do wymiaru obrazu (128×128) przy

pomocy klasy `Embedding` z PyTorch, aby mogły zostać połączone z obrazami wejściowymi. Przekształcenie to umożliwia także modelowi zastosowanie informacji o etykiecie w każdym miejscu na obrazie. Po połączeniu na wejściu pierwszej warstwy konwolucji obraz ma wymiary $N \times 2 \times 128 \times 128$. Wartości te odpowiadają odpowiednio:

– liczbie danych w pojedynczej partii,

– liczbie kanałów (jeden kanał obrazu wejściowego w skali szarości wraz z kanałem odpowiadającym etykiecie)

– wysokość obrazu wejściowego,

– szerokość obrazu wejściowego.

W krytyku jako funkcja aktywacji użyto LeakyReLU. Funkcja ta w odróżnieniu od ReLU nie zeruje ujemnych wartości i działa w następujący sposób:

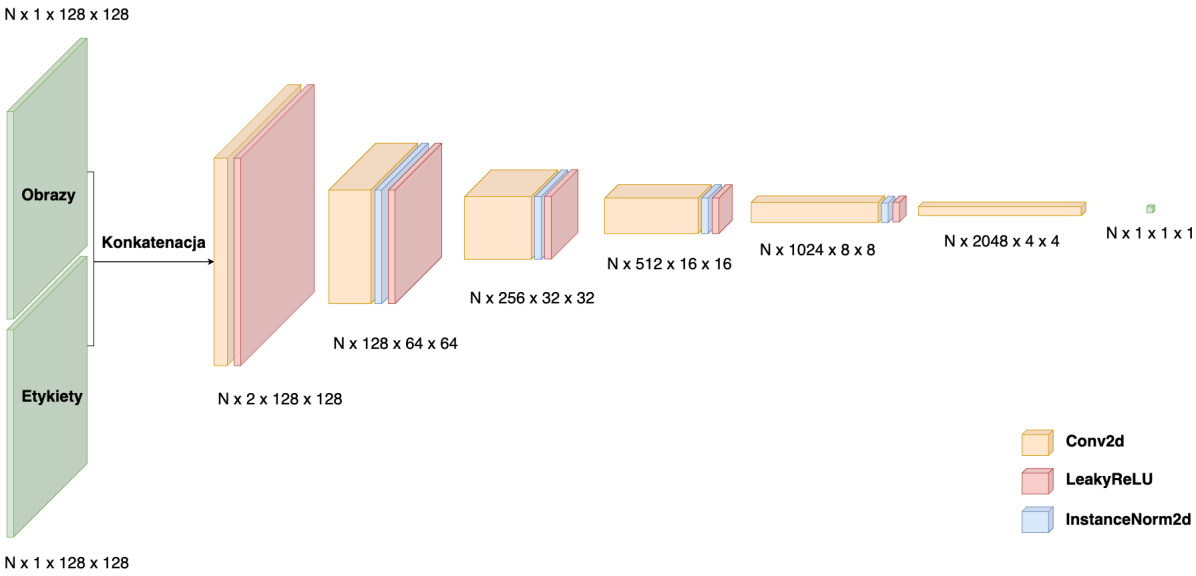
$$f(x) = \begin{cases} x & \text{, dla } x \geq 0 \\ \alpha x & \text{, dla } x < 0 \end{cases} \tag{4.1}$$

gdzie:

α – niewielka stała wartość, kontrolująca o ile zmniejszane są wartości negatywne.

Wybór funkcji LeakyReLU pozwala na zachowanie niewielkiego stałego gradientu dla wartości ujemnych, co zapobiega problemowi wymierających neuronów, który można napotkać podczas stosowania zwykłej funkcji ReLU. W pracy wartość parametru α została dobrana na podstawie artykułu wprowadzającego sieci DCGAN [24] i wynosiła 0.2.

Zastosowana została także normalizacja przy pomocy klasy `InstanceNorm2d` z PyTorch. Funkcja ta normalizuje niezależnie każdą próbkę danych. Dla porównania funkcja `BatchNorm2d`, dokonuje normalizacji danych na podstawie średniej i wariancji w całej ich partii.



Rys. 4.2. Architektura i schemat działania krytyka (N - liczba próbek w każdej partii danych). Źródło: opracowanie własne.

Model, poza pierwszą i ostatnią warstwą, składa się z bloków konwolucyjnych zawierających odpowiednio:

– dwuwymiarową warstwę konwolucyjną z rozmiarem jądra o wartości 4, krokiem 2 i dopełnieniem równym 1 (zastosowanej przy pomocy klasy `Conv2d` z PyTorch),

– dwuwymiarową normalizacje instancji (zastosowanej przy pomocy klasy `InstanceNorm2d` z PyTorch),

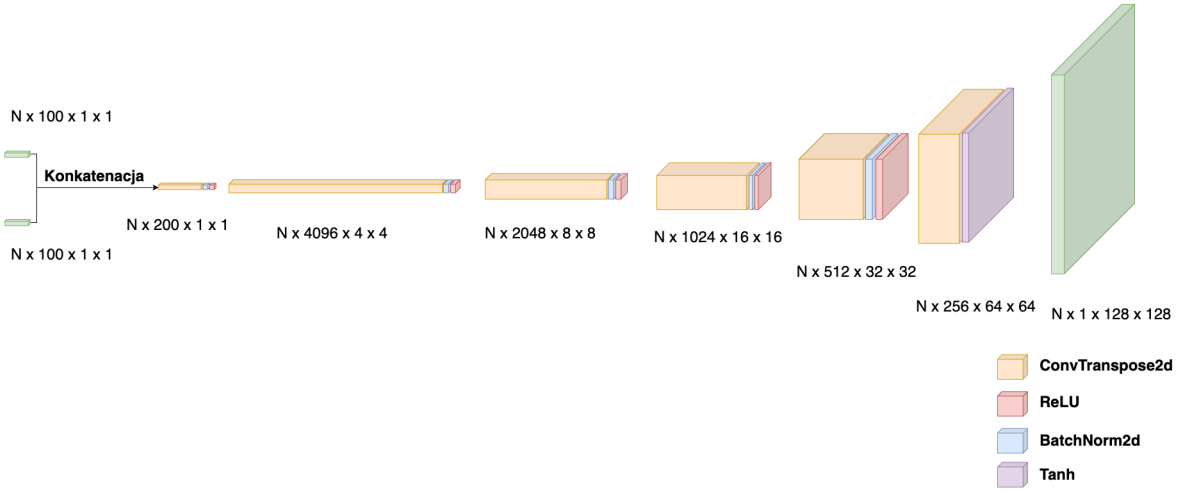
– funkcji aktywacji LeakyReLU z parametrem $\alpha = 0.2$ (zastosowanej przy pomocy klasy `LeakyReLU` z PyTorch).

W pierwszej warstwie nie została zastosowana operacja normalizacji. Jest to praktyka często stosowana w literaturze, aby w pierwszej warstwie dyskryminatora nie stosować normalizacji. Zabieg ten ma na celu uniknięcie straty lub zniekształceniu pewnych informacji z obrazu wejściowego. Natomiast w ostatniej warstwie zastosowana została sama operacja konwolucji (z dopełnieniem o wartości 0). Zadaniem ostatniej warstwy jest uzyskanie pojedynczej wartości wyjściowej. Zmiany wymiarów pomiędzy poszczególnymi blokami sieci zostały zobrazowane na rysunku 4.2.

4.3.3. Architektura generatora

Schemat działania i architektura generatora zostały zobrazowane na rysunku 4.3. Na wejściu sieć otrzymuje wektor szumu o rozmiarze $N \times 100 \times 1 \times 1$ oraz wartości etykiet obrazów do wygenerowania, przekształcone do tensora o takim samym wymiarze, również przy pomocy klasy `Embedding`. Następnie oba tensory są ze sobą łączone i podane na wejście do pierwszej warstwy sieci.

W generatorze zastosowana została normalizacja wsadowa, która wykonuje normalizację na całej partii danych. Natomiast jako funkcja aktywacji użyta została funkcja ReLU. Wybór ten został dokonany na podstawie analizy literatury dla różnych rodzajów sieci GAN.



Rys. 4.3. Architektura i schemat działania generatora (N - liczba próbek w każdej partii danych). Źródło: opracowanie własne.

W generatorze używana jest operacja konwolucji transponowanej. Architektura sieci, poza pierwszą i ostatnią warstwą, składa się z następujących bloków:

– dwuwymiarowej warstwy konwolucji transponowanej z rozmiarem jądra o wartości 4, krokiem 2 i dopełnieniem równym 1 (zastosowanej przy pomocy klasy `ConvTranspose2d` z PyTorch),

– dwuwymiarowej normalizacji wsadowej (zastosowanej przy pomocy klasy `BatchNorm2d` z PyTorch),

– funkcji aktywacji ReLU (zastosowanej przy pomocy klasy `ReLU` z PyTorch).

W pierwszej warstwie zastosowany jest taki sam blok jak wyżej, ale ze zmienionymi wartościami kroku na 1 oraz dopełnienia na 0. Ma to na celu otrzymanie, z wektora szumu i etykiet o rozdzielczości 1×1, obiektu o rozdzielczości 4×4. Natomiast w ostatniej warstwie po operacji konwolucji transponowanej nie zastosowana została normalizacja, a jedynie funkcja aktywacji. Jednak w tym przypadku używana jest funkcja tangensa hiperbolicznego (przy pomocy klasy `Tanh` z PyTorch), aby otrzymać obraz wynikowy z wartościami z zakresu od -1 do 1. Operacja ta ma na celu zachowanie zgodności z zakresem wartości obrazów wejściowych do modelu krytyka.

Wyjściem z generatora są jednokanałowe obrazy o pożądanym wymiarze wynoszącym 128×128 pikseli. Zmiany wymiarów pomiędzy poszczególnymi blokami sieci zostały zobrazowane na rysunku 4.3.

4.3.4. Proces trenowania modelu

Proces uczenia modelu poprzedzony jest następującymi krokami:

1. Stworzenie instancji klas krytyka i generatora.
2. Inicjalizacja wag krytyka i generatora. Została ona wykonana na podstawie artykułu o sieci DCGAN [24]. Polegała na losowym przypisaniu do wag wartości z rozkładu normalnego o średniej 0.0 i odchyleniu standardowym 0.02.
3. Stworzenie instancji optymalizatorów dla krytyka i generatora. Wykorzystany został optymalizator RMSprop z wartością współczynnika uczenia (ang. *learning rate*) wynoszącą 0.0002. Wartość współczynnika uczenia określa jak szybko model aktualizuje wagi. Zbyt wysoka wartość może spowodować oscylacje funkcji straty, natomiast zbyt mała wartość może sprawić, iż model będzie zbyt wolno zbiegał do optymalnego rozwiązania. Wartość ta została dobrana eksperymentalnie. Natomiast optymalizator został wybrany zgodnie z artykułem o sieciach WGAN [17]. Dla optymalizatora Adam, który również jest dość często używany w różnych publikacjach o sieciach GAN, model osiągał gorsze wyniki.

Następnie rozpoczynana jest główna pętla procesu uczenia która iteruje po liczbie epok (w pracy wynosząca 100). Wewnątrz następuje iteracja po obiekcie klasy `DataLoader` zawierającym iterator po danych treningowych, który zwraca partie danych treningowych oraz etykiet (o wcześniej ustalonym rozmiarze wsadu równym 128). Następnie wewnątrz rozpoczyna się pętla odpowiedzialna za uczenie krytyka. Liczba iteracji szkolenia krytyka przypadająca na jedną iterację treningową generatora została określona za pomocą parametru `CRITIC_ITERATIONS` wynoszącego 5. Każda iteracja składa się z następujących kroków:

1. Wygenerowanie losowego szumu.
2. Wygenerowanie danych syntetycznych z generatora.
3. Wyliczenie wartości wyjścia z krytyka dla danych prawdziwych i syntetycznych.
4. Obliczenie kary gradientu na podstawie wzoru 3.16 opisanego w rozdziale 3.5.4. Wartość współczynnika kary została ustalona na 10 zgodnie z artykułem na temat sieci WGAN-GP [18].
5. Obliczenie funkcji krytyka, uwzględniającej karę gradientu, na podstawie wzoru 3.17 z rozdziału 3.5.4.
6. Wyzerowanie gradientów krytyka.
7. Propagacja wsteczna funkcji straty dla krytyka.
8. Aktualizacja wag krytyka.

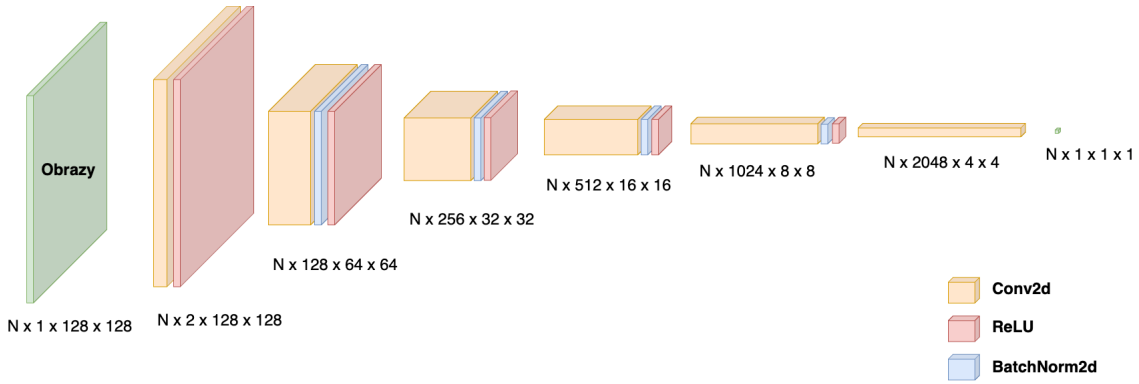
Po pięciu iteracjach szkolenia krytyka następuje czas na uczenie generatora, które składa się z następujących kroków:

1. Obliczenie wyjścia z krytyka dla syntetycznych danych.
2. Obliczenie wartości funkcji straty dla generatora.
3. Wyzerowanie gradientów generatora.
4. Propagacja wsteczna funkcji straty dla generatora.
5. Aktualizacja wag generatora.

Po zakończeniu procesu iteracji po wszystkich partiach danych następuje przejście do kolejnej epoki i kolejna iteracja po danych treningowym wraz z uczeniem obu modeli tak jak zostało to opisane powyżej. Na sam koniec całego procesu model generatora zapisywany jest do pliku, aby móc z niego skorzystać do wygenerowania nowych etykietowanych danych syntetycznych.

4.4. Implementacja modelu klasyfikacyjnego

Model klasyfikacyjny został zastosowany do ewaluacji jakości obrazów generowanych przez generator. Został on stworzony na wzór architektury krytyka, co można zaobserwować na rysunku 4.4. Zmianie uległy jedynie: funkcja aktywacji na ReLU oraz normalizacja na normalizację wsadową. Podczas procesu uczenia modelu klasyfikacyjnego zastosowano optymalizator Adam ze współczynnikiem uczenie wynoszącym 0.0001. Jako funkcje straty wykorzystano `CrossEntropyLoss` z PyTorch. Z tego powodu nie była wymagana funkcja aktywacji softmax w ostatniej warstwie klasyfikatora, ponieważ `CrossEntropyLoss` zawiera w sobie funkcję softmax. Liczba epok podczas uczenia klasyfikatora wynosiła 15, a liczba próbek w partii 64.



Rys. 4.4. Architektura i schemat działania klasyfikatora (N - liczba próbek w każdej partii danych). Źródło: opracowanie własne.

5. Eksperymenty

W rozdziale tym opisane zostaną metryki i metody wykorzystane do ewaluacji wyników warunkowej sieci DCWGAN-GP wraz z przedstawieniem i omówieniem ich wyników.

5.1. Proces ewaluacji

Proces ewaluacji wyników uwzględniał następujące etapy:

- analiza wartości funkcji straty krytyka i generatora,
- analiza jakości wygenerowanych obrazów,
- analiza metryk FID oraz IS,
- nauczenie modelu klasyfikacyjnego na danych rzeczywistych i sprawdzenie wyników (dane rzeczywiste jako dane testowe),
- sprawdzenie wyników klasyfikacji nauczonego modelu dla danych syntetycznych,
- nauczenie modelu klasyfikacyjnego na danych rzeczywistych + syntetycznych i sprawdzenie wyników (dane rzeczywiste jako dane testowe),

FID (ang. *Fréchet Inception Distance*) mierzy różnicę pomiędzy rozkładami obrazów rzeczywistych i syntetycznych. Metryka ta stosowana jest do oceny jakości obrazów generowanych przez generatywne sieci współzawodniczące. FID korzysta z modelu klasyfikacyjnego `Inception` do ekstrakcji wektorów cech dla obrazów rzeczywistych i wygenerowanych. Metryka obliczana jest w następujący sposób [25]:

$$FID(r, g) = \|\mu_r - \mu_g\|^2 + \mathrm{tr}\left(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}\right) \tag{5.1}$$

gdzie:

μ_r – wektor średnich dla obrazów rzeczywistych,

μ_g – wektor średnich dla obrazów wygenerowanych,

$\|\mu_r - \mu_g\|^2$ – kwadrat normy euklidesowej różnicy pomiędzy wektorami μ_r i μ_g,

tr – ślad macierzy,

Σ_r – macierz kowariancji cech wyekstrahowanych z obrazów rzeczywistych,

Σ_g – macierz kowariancji cech wyekstrahowanych z obrazów syntetycznych.

Niższa wartość FID oznacza, że obrazy mają podobne cechy. Zatem malejąca wartość FID podczas uczenia modelu GAN wskazuje na to, iż generowane obrazy są coraz bardziej zbliżone do rzeczywistych.

Metryka IS (ang. *Inception Score*) polega na ocenie różnorodności obrazów generowanych przez generatywną sieć współzawodniczącą oraz ich jakości. IS również korzysta z modelu `Inception` i opisana jest nastepującym wzorem:

$$IS = \exp\left(\mathbb{E}_x\left[KL(p(y|x)\,||\,p(y))\right]\right) \tag{5.2}$$

gdzie:

$\mathbb{E}_x$ – wartość oczekiwana po wszystkich obrazach x wygenerowanych,

KL – dywergencja Kullbacka-Leiblera, czyli miara różnicy pomiędzy dwiema dystrybucjami prawdopodobieństwa,

$p(y|x)$ – warunkowe prawdopodobieństwo klasy y dla danego obrazu x, uzyskane z modelu `Inception`,

$p(y)$ – prawdopodobieństwo marginalne klasy y po wszystkich obrazach syntetycznych.

Jeżeli obrazy generowane przez GAN są dobrej jakości, to powinny być one łatwo klasyfikowane przez model `Inception` jako należące do konkretnej klasy (czyli $p(y \mid x)$ powinno być wysokie dla tej klasy y i niskie dla innych klas). Oczekiwane jest również, aby GAN generował różnorodne obrazy, co oznacza, że prawdopodobieństwo marginalne $p(y)$ powinno być równomiernie rozłożone po wszystkich klasach. Dlatego stosuje się dywergencje KL do porównania tych dwóch dystrybucji [26]. Im wyższa wartość metryki IS, tym większa różnorodność obrazów generowanych przez sieć GAN.

Metryki FID i IS często są używane razem, ponieważ razem oferują informacje zarówno o jakości jak i różnorodności obrazów generowanych przez sieć GAN. Jednak metryki te nie zawsze są w stanie dostarczyć w pełni miarodajne wyniki, dlatego warto również dokonać samodzielnej oceny jakości i różnorodności generowanych obrazów.

Podczas stosowania klasyfikatora do oceny wpływu danych syntetycznych na wydajność modelu użyto następujących metryk do ewaluacji wyników:

- macierz pomyłek,
- dokładność,
- specyficzność,
- czułość,

– precyzja,

– F1-score.

Macierzy pomyłek (ang. *confusion matrix*) jest najpopularniejszym sposobem oceny jakości klasyfikacji i jest zdefiniowana w następujący sposób:

Tabela 5.1. Macierz błędu.

		Wynik faktyczny	
		pozytywny	negatywny
Przewidziany wynik	pozytywny	**TP**	**FP**
	negatywny	**FN**	**TN**

gdzie:

TP – (ang. *True Positive*) liczba przewidywań prawdziwie pozytywnych

TN – (ang. *True Negative*) liczba przewidywań prawdziwie negatywnych

FP – (ang. *False Positive*) liczba przewidywań fałszywie pozytywnych

FN – (ang. *False Negative*) liczba przewidywań fałszywie negatywnych

Dokładność (ACC, ang. *accuracy*) określa jaka część ze wszystkich klasyfikowanych danych została zaklasyfikowana poprawnie. Można ją obliczyć ze wzoru:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5.3}$$

Specyficzność (SPEC, ang. *specificity*) jest to metryka określająca, w jakim stopniu klasa faktycznie negatywna została przewidziana jako negatywna. W przypadku danych użytych w pracy będzie ona określać jaki procent zdjęć zdrowych płuc zostało uznanych za zdrowe. Jest ona opisana wzorem:

$$SPEC = \frac{TN}{TN + FP}. \tag{5.4}$$

Czułość (SENS, ang. *sensitivity*) jest to metryka określająca, w jakim stopniu klasa faktycznie pozytywna została przewidziana jako pozytywna. W przypadku danych użytych w pracy będzie ona określać jaki procent zdjęć chorych płuc zostało uznanych za chore. Można ją obliczyć ze wzoru:

$$SENS = \frac{TP}{TP + FN}. \tag{5.5}$$

Precyzja (PREC, ang. *precision*) jest to metryka określająca, w jakim stopniu dane przewidziane jako pozytywne były faktycznie pozytywne. W przypadku danych użytych w pracy będzie ona określać jaki procent zdjęć uznanych za chore były faktycznie zdjęciami chorych płuc. Jest ona opisana wzorem:

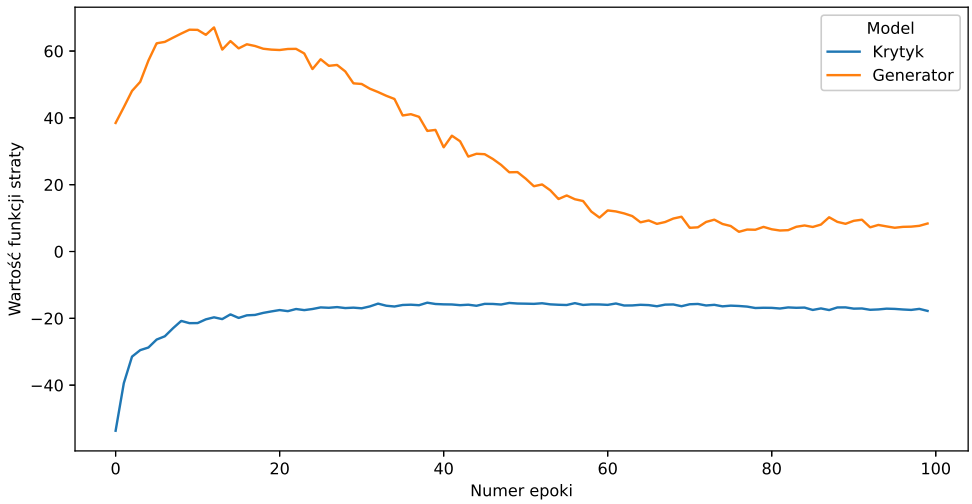
$$PREC = \frac{TP}{TP + FP}. \tag{5.6}$$

F1-score jest harmoniczną średnią czułości i precyzji. Jest ona szczególnie przydatna, gdy klasy są niezbalansowane.

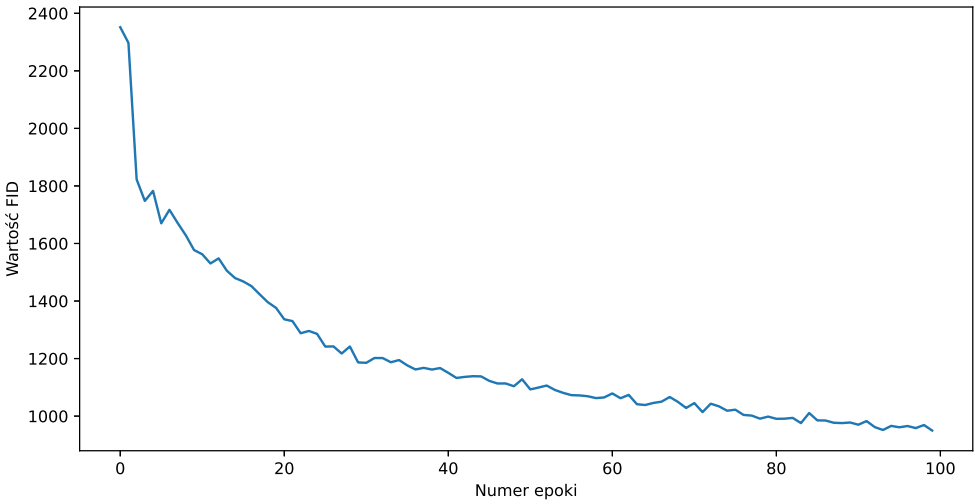
Na samym początku dokonana została ewaluacja wyników dla modelu klasyfikacyjnego uczonego wyłącznie na danych rzeczywistych. Następnie dokonano ewaluacji wyników dla modelu uczonego na zbiorze danych treningowych zwierającym zarówno dane rzeczywiste jak i syntetyczne. Do zbioru danych treningowych dołączono po 1000 obrazów dla każdej z klas. Warto również dodać, iż jako dane testowe w obu przypadkach zostały użyte dane rzeczywiste, aby porównanie było miarodajne i miało sens. Dodatkowo, dla modelu uczonego na danych rzeczywistych, dokonano sprawdzenia jak radzi on sobie z predykcją danych syntetycznych (również po 1000 wygenerowanych danych dla każdej z klas).

5.2. Przedstawienie i analiza wyników

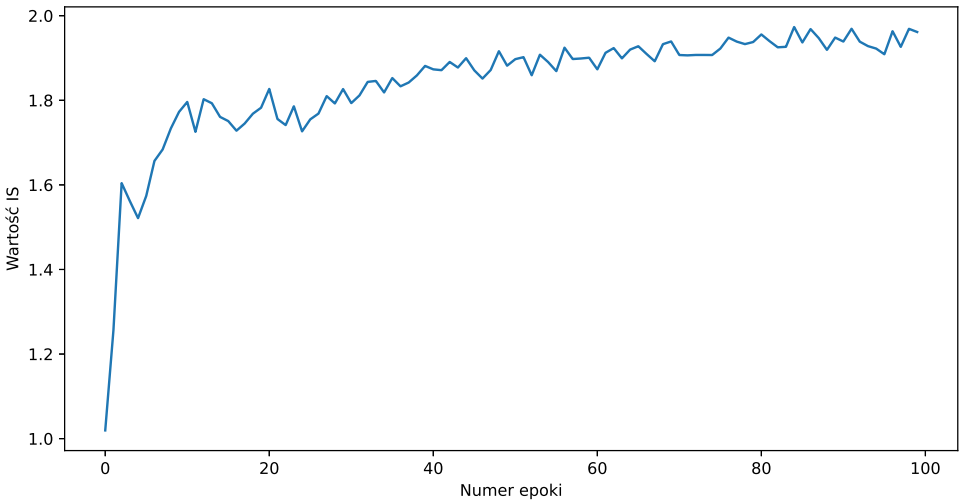
Poniżej zostały przedstawione odpowiednio wartości funkcji straty krytyka i generatora oraz metryk FID i IS. Wszystkie z tych wartości były mierzone w każdej iteracji po danych, a następnie uśredniane, aby otrzymać wartość dla danej epoki.



Rys. 5.1. Wykres uśrednionej wartości funkcji straty krytyka i generatora w każdej epoce. Źródło: opracowanie własne.



Rys. 5.2. Wykres uśrednionej wartości metryki FID w każdej epoce. Źródło: opracowanie własne.

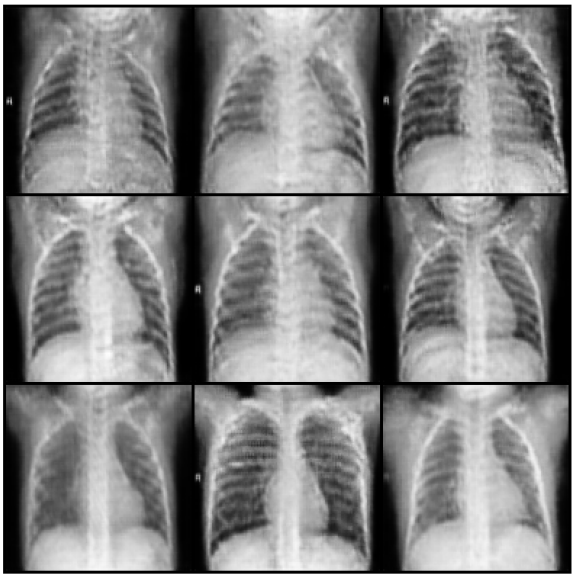


Rys. 5.3. Wykres uśrednionej wartości metryki IS w każdej epoce. Źródło: opracowanie własne.

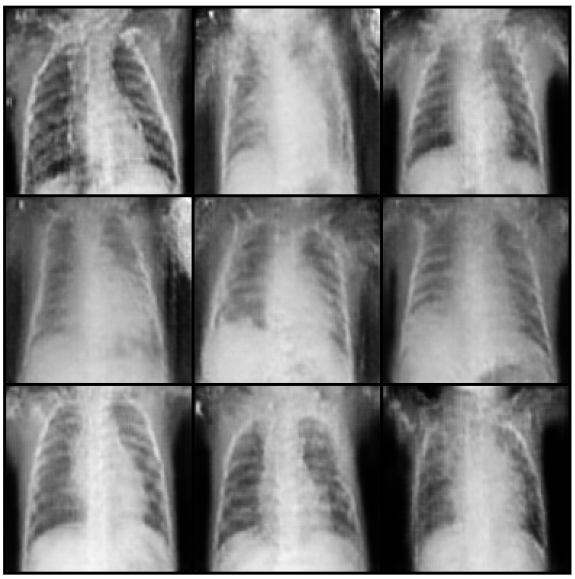
Na wykresie 5.1 można zauważyć, iż krytyk szybko osiąga pewną stabilizację funkcji straty (około 20 epoki). Oznacza to, że krytyk dość szybko uczy się rozpoznawać dane rzeczywiste od syntetycznych. W przypadku generatora początkowo można zauważyć wzrost funkcji straty wynikający z początkowych trudności w oszukaniu krytyka. Jednak po chwili funkcja straty dla generatora zaczęła maleć, co oznacza, iż model zaczynał generować coraz lepsze dane. Finalnie strata krytyka stabilizuje się w okolicach 80 epoki i jest bliżej wartości zerowej w porównaniu do straty krytyka. Oznacza, to iż generatorowi udało się oszukać krytyka, generując próbki trudne do odróżnienia od rzeczywistych.

Wartość metryki FID maleje z epoki na epokę, co można zauważyć na wykresie 5.2. Wartości wykazują tendencję malejącą, co oznacza, iż model był w stanie generować obrazy o coraz to bardziej podobnych cechach w porównaniu do danych rzeczywistych. Natomiast wartości IS, zobrazowane na wykresie 5.3, wykazują tendencję wzrostową, co również jest dobrym znakiem. Wartości tej metryki wskazują iż dane syntetyczne były lepsze jakościowo i bardziej różnorodne wraz z postępem procesu uczenia.

Na rysunkach 5.4 oraz 5.5 zostały zobrazowane odpowiednio wygenerowane zdjęcia zdrowych płuc oraz z zapaleniem płuc. Sieć była w stanie generować obrazy o dość dobrym odzwierciedleniu cech danych realnych dla obu klas. Szczególnie biorąc pod uwagę ograniczenie rozdzielczości do wymiary 128×128 pikseli.



Rys. 5.4. Wygenerowane zdjęcia rentgenowskie zdrowych płuc. Źródło: opracowanie własne.

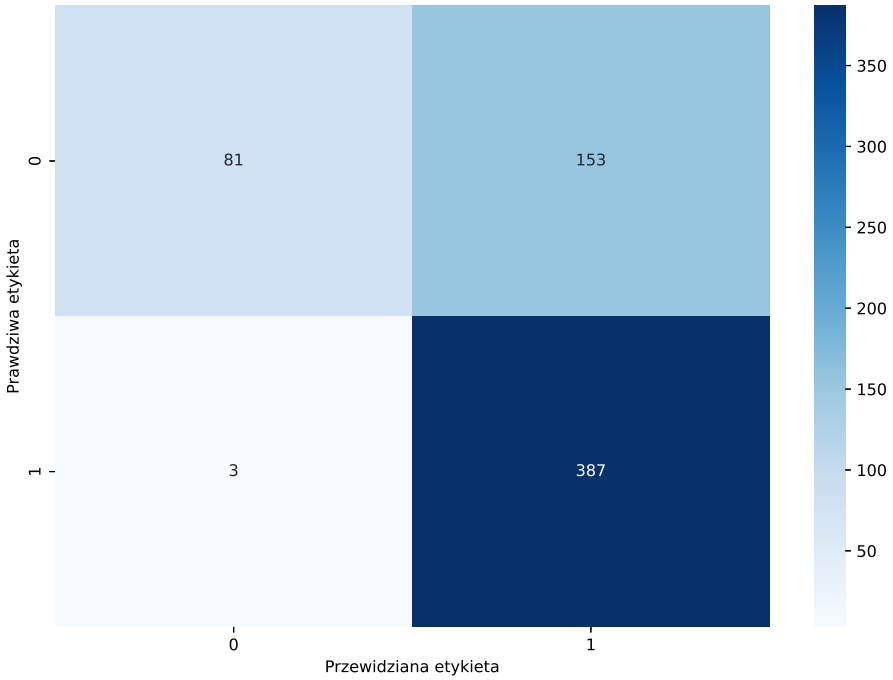


Rys. 5.5. Wygenerowane zdjęcia rentgenowskie z zapaleniem płuc. Źródło: opracowanie własne.

Natomiast na rysunku 5.6 przedstawiony został przykład błędnej generacji zdjęcia rentgenowskiego zdrowych płuc. Jest to jeden z nielicznych przypadków, gdzie jakość generowanego obrazu była bardzo niska. Powodem błędnej generacji mogła być ograniczona rozdzielczość zdjęć wejściowych. Najprawdopodobniej jej zwiększenie pozwoliłoby na wyeliminowanie takich błędów.

Rys. 5.6. Błędnie wygenerowane zdjęcia rentgenowskie zdrowych płuc. Źródło: opracowanie własne.

Klasyfikator uczony tylko na danych realnych, użyty do ewaluacji przydatności danych syntetycznych do poprawienia jego dokładności, osiągnął następujące wyniki.



Rys. 5.7. Macierz pomyłek klasyfikatora uczonego na danych rzeczywistych dla rzeczywistych danych testowych. Źródło: opracowanie własne.

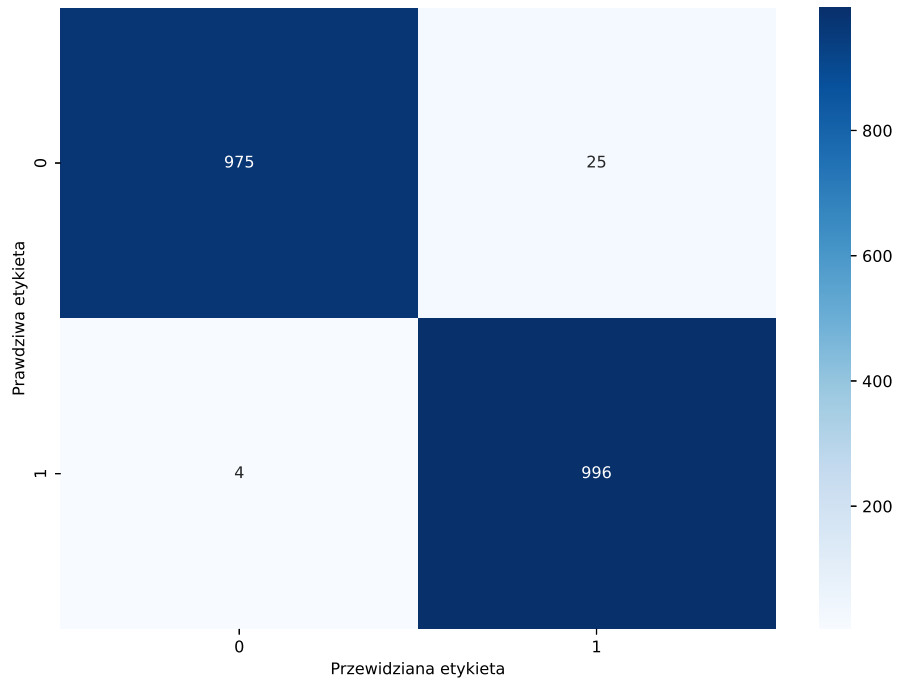
Tabela 5.2. Wyniki metryk klasyfikatora uczonego na danych rzeczywistych dla rzeczywistych danych testowych.

Metryka	**Wynik**
Dokładność	75%
Specyficzność	35%
Czułość	99%
Precyzja	72%
F1-Score	83%

Na podstawie powyższych wyników można zauważyć, iż klasyfikator uczony na danych rzeczywistych osiągnął dokładność na poziomie 75% i dużo lepiej radził sobie z przewidywaniem zdjęć z zapaleniem płuc. Klasyfikator osiągnął czułość na poziomie 99%, czyli praktycznie wszystkie zdjęcia faktycznie chorych płuc zostały sklasyfikowane jako chore. Jednak wynik specyficzności na poziomie 35% pokazuje, iż klasyfikator klasyfikuje dużo danych rzeczywistych zdrowych płuc jako chore. Niemniej jednak w przypadku danych medyczny dużo lepiej jest otrzymać wynik fałszywie pozytywny niż fałszywie negatywny. Inaczej mówiąc lepiej popełnić błąd klasyfikując zdrowe zdjęcie jako chore, ponieważ później ewentualne badania mogą finalnie wykluczyć występowanie choroby. Sklasyfikowanie

zdjęć chorych jako zdrowe może spowodować opóźnienie poprawnej diagnozy i ewentualnej hospitalizacji pacjenta lub nawet całkowity jej brak, co może być fatalne w skutkach. Wynik precyzji na poziomie 73% wskazuje na to, iż większość zdjęć przewidzianych jako chore były faktycznie chore.

Powyższy klasyfikator, po podaniu mu danych syntetycznych i dokonaniu klasyfikacji, osiągnął następujące wyniki.



Rys. 5.8. Macierz pomyłek klasyfikatora uczonego na danych rzeczywistych dla syntetycznych danych testowych. Źródło: opracowanie własne.

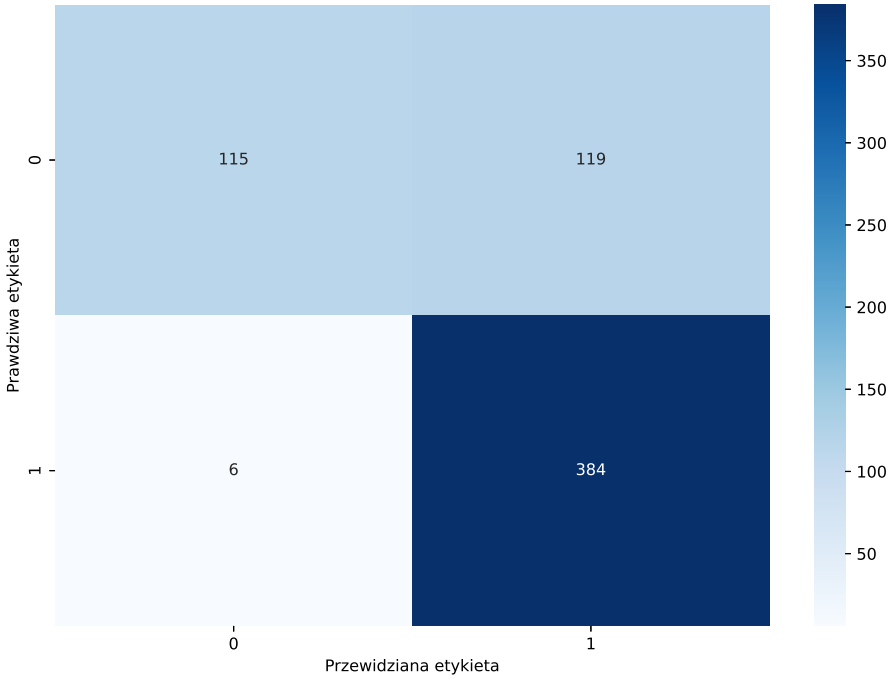
Tabela 5.3. Wyniki metryk klasyfikatora uczonego na danych rzeczywistych dla syntetycznych danych testowych.

Metryka	**Wynik**
Dokładność	99%
Specyficzność	98%
Czułość	99%
Precyzja	98%
F1-Score	99%

Model klasyfikacyjny nauczony na danych rzeczywistych był w stanie przewidywać etykiety wygenerowanych zdjęć rentgenowskich płuc z dokładnością na poziomie 99%. Wynik ten wskazuje na to,

iż generator był w stanie bardzo dobrze nauczyć się cech ze zdjęć rzeczywistych, które są istotne dla klasyfikatora. Zarówno zdjęcia zdrowych jak i chorych płuc były przewidywane niemal bezbłędnie z minimalną przewagą dla zdjęć rentgenowskich chorych płuc.

Klasyfikator uczony na danych rzeczywistych wraz z obrazami syntetycznymi osiągnął następujące wyniki.



Rys. 5.9. Macierz pomyłek klasyfikatora uczonego na danych rzeczywistych i syntetycznych dla rzeczywistych danych testowych. Źródło: opracowanie własne.

Tabela 5.4. Wyniki metryk klasyfikatora uczonego na danych rzeczywistych i syntetycznych dla rzeczywistych danych testowych.

Metryka	**Wynik**
Dokładność	80%
Specyficzność	49%
Czułość	98%
Precyzja	76%
F1-Score	86%

Po zastosowaniu danych syntetycznych podczas uczenia modelu udało się zwiększyć ogólną jego dokładność o około 5%. Przewidywanie klasy pozytywnej nie uległo znaczącej zmianie, jednak dla klasy

negatywnej można zauważyć wzrost metryki specyficzności z 35% do 49%. Można z tego wywnioskować, iż po dodaniu danych syntetycznych model był w stanie lepiej przewidywać zdjęcia zdrowych płuc, jednocześnie zachowując bardzo dobre wyniki w przewidywaniu zapalenia płuc.

6. Podsumowanie

W rozdziale tym przedstawione zostaną wnioski wynikające z otrzymanych wyników, ograniczenia i wyzwania napotkane podczas realizacji pracy oraz potencjalne perspektywy rozwoju.

W pracy udało się zaimplementować oraz wytrenować sieć warunkową DCWGAN-GP do generowania danych etykietowanych, a konkretnie zdjęć rentgenowskich płuc zdrowych oraz z zapaleniem. Wyuczony model był w stanie generować obrazy dobrej jakości wraz z odwzorowaniem istotnych cech z danych rzeczywistych. Wyniki metryk FID oraz IS wskazują na to, iż model z epoki na epokę był w stanie generować zdjęcia o lepszej jakości oraz większej różnorodności. Eksperymenty przy pomocy dodatkowego modelu klasyfikacyjnego pokazały potencjał zastosowania danych syntetycznych do polepszenia wyników predykcji modelu. W pracy udało się poprawić dokładność modelu poprzez dodanie danych wygenerowanych przez sieć warunkową DCWGAN-GP o około 5%. Dodatkowo sieć uczona na danych rzeczywistych była w stanie przewidywać etykietę danych syntetycznych z dokładnością wynoszącą 99%. Rezultat ten wskazuje na to, iż generator był w stanie dobrze nauczyć się kluczowych wzorców dla danej etykiety z danych wejściowych.

Jednym z głównych wyzwań podczas realizacji pracy były ograniczenia sprzętowe. Mimo zastosowania najlepszego procesora graficznego dostępnego na platformie Google Colab wymiar generowanych obrazów musiał zostać ograniczony do 128×128 pikseli. Większe rozdzielczości byłyby zbyt czasochłonne podczas procesu uczenia.

Jednak, pomimo ograniczenia wymiarów obrazów, czas potrzebny na nauczenie sieci był nadal znaczący, co spowodowało ograniczenie liczby eksperymentów z różnymi zestawami parametrów. Długotrwały proces uczenia nie tylko utrudniał odpowiednie dostrojenie modelu, ale również opóźniał ewaluację poszczególnych konfiguracji, co mogło wpłynąć na ostateczną jakość generowanych danych.

Potencjalnymi perspektywami rozwoju, mającymi na celu ulepszenie działania aktualnego modelu, mogłyby być:

- zastosowanie bardziej zaawansowanego sprzętu, aby model był w stanie generować zdjęcia o wyższej rozdzielczości, co potencjalnie może poprawić jakość oraz odwzorowanie szczegółów na generowanych obrazach,
- dalsze eksperymenty nad ulepszeniem architektury modelu oraz doborem parametrów,

– zastosowanie dodatkowych źródeł danych treningowych.

Podsumowując, poprzez zastosowanie odpowiedniego wariantu sieci GAN (*Conditional DCWGAN-GP*), udało się wygenerować dane etykietowane o bardzo wysokim podobieństwie do danych rzeczywistych, a ich jakość i różnorodność pozwoliła na efektywne zastosowanie ich do polepszenia predykcji modelu klasyfikacyjnego.

Bibliografia

[1] Zakarya Farou, Noureddine Mouhoub i Tomáš Horváth. „Data generation using gene expression generator". W: *Intelligent Data Engineering and Automated Learning–IDEAL 2020: 21st International Conference, Guimaraes, Portugal, November 4–6, 2020, Proceedings, Part II 21*. Springer. 2020, s. 54–65.

[2] Saulo Barreto. *Data Augmentation*. URL: *https://www.baeldung.com/cs/ml-data-augmentation* (term. wiz. 2023-08-28).

[3] Xin Yi, Ekta Walia i Paul Babyn. „Generative adversarial network in medical imaging: A review". W: *Medical image analysis* 58 (2019), s. 101552.

[4] Pengzhen Ren i in. „A survey of deep active learning". W: *ACM computing surveys (CSUR)* 54.9 (2021), s. 1–40.

[5] Yassine Ouali, Céline Hudelot i Myriam Tami. „An overview of deep semi-supervised learning". W: *arXiv preprint arXiv:2006.05278* (2020).

[6] Ian Goodfellow i in. „Generative adversarial nets". W: *Advances in neural information processing systems* 27 (2014).

[7] Han Zhang i in. „Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". W: *Proceedings of the IEEE international conference on computer vision*. 2017, s. 5907–5915.

[8] Leon Sixt, Benjamin Wild i Tim Landgraf. „Rendergan: Generating realistic labeled data". W: *Frontiers in Robotics and AI* 5 (2018), s. 66.

[9] Maayan Frid-Adar i in. „GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification". W: *Neurocomputing* 321 (2018), s. 321–331.

[10] softologyblog. *Style Transfer GANs (Generative Adversarial Networks)*. 2019. URL: *https://softologyblog.wordpress.com/2019/03/31/style-transfer-gans-generative-adversarial-networks/* (term. wiz. 2023-07-02).

[11] David Vint i in. „Automatic target recognition for low resolution foliage penetrating SAR images using CNNs and GANs". W: *Remote Sensing* 13.4 (2021), s. 596.

[12] PyTorch. *BCELOSS*. URL: *https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html* (term. wiz. 2023-07-06).

[13] Lakshmi Ajay. *Decoding the Basic Math in GAN — Simplified Version*. 2022. URL: *https://towardsdatascience.com/decoding-the-basic-math-in-gan-simplified-version-6fb6b079793* (term. wiz. 2023-07-05).

[14] David Foster. *Deep learning i modelowanie generatywne*. O'Reilly Media, Incorporated, 2021, s. 108–113.

[15] Keiron O'Shea i Ryan Nash. „An introduction to convolutional neural networks". W: *arXiv preprint arXiv:1511.08458* (2015).

[16] Mehdi Mirza i Simon Osindero. „Conditional generative adversarial nets". W: *arXiv preprint arXiv:1411.1784* (2014).

[17] Martin Arjovsky, Soumith Chintala i Léon Bottou. „Wasserstein generative adversarial networks". W: *International conference on machine learning*. PMLR. 2017, s. 214–223.

[18] Ishaan Gulrajani i in. „Improved training of wasserstein gans". W: *Advances in neural information processing systems* 30 (2017).

[19] Phillip Isola i in. „Image-to-image translation with conditional adversarial networks". W: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, s. 1125–1134.

[20] Jun-Yan Zhu i in. „Unpaired image-to-image translation using cycle-consistent adversarial networks". W: *Proceedings of the IEEE international conference on computer vision*. 2017, s. 2223–2232.

[21] Tero Karras, Samuli Laine i Timo Aila. „A style-based generator architecture for generative adversarial networks". W: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, s. 4401–4410.

[22] Paul Mooney. *Chest X-Ray Images (Pneumonia)*. URL: *https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia* (term. wiz. 2023-07-10).

[23] Daniel S Kermany i in. „Identifying medical diagnoses and treatable diseases by image-based deep learning". W: *cell* 172.5 (2018), s. 1122–1131.

[24] Alec Radford, Luke Metz i Soumith Chintala. „Unsupervised representation learning with deep convolutional generative adversarial networks". W: *arXiv preprint arXiv:1511.06434* (2015).

[25] Wikipedia. *Fréchet distance*. URL: *https://en.wikipedia.org/wiki/Fr%C3%A9chet_distance* (term. wiz. 2023-09-04).

[26] Tim Salimans i in. „Improved techniques for training gans". W: *Advances in neural information processing systems* 29 (2016).