

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ INFORMATYKI

Projekt dyplomowy

System do rozpoznawania obrazów generowanych automatycznie

System for recognition of automatically generated images

Autor: Marta Bukowińska, Olaf Fertig, Wojciech Fortuna
Kierunek: Informatyka
Opiekun pracy: dr hab. inż. Bartłomiej Śnieżyński, prof. AGH

Kraków, 2026

W przygotowaniu niniejszej pracy inżynierskiej wykorzystano narzędzia językowe oparte na sztucznej inteligencji w celu poprawy przejrzystości, spójności oraz ogólnej jakości tekstu.

Narzędzia te wspierają takie zadania jak korekta gramatyczna, udoskonalanie stylu oraz proponowanie sformułowań. Jednakże treść, idee, wyniki badań oraz wnioski przedstawione w pracy należą wyłącznie do autorów. Wszelkie błędy lub braki pozostają odpowiedzialnością autorów.

Streszczenie

Celem pracy było zaprojektowanie i implementacja systemu do rozpoznawania obrazów generowanych automatycznie, umożliwiającego określenie ich pochodzenia, analizę obrazów z wykorzystaniem wybranych metod detekcji oraz wizualną interpretację wyników działania modeli.

W ramach projektu opracowano dwa warianty systemu: pełnofunkcjonalny, umożliwiający integrację wielu metod detekcji, analizę metadanych, generację raportów oraz interpretację wyników, oraz wersję przeznaczoną dla środowisk o ograniczonych zasobach, opartą na elastycznej architekturze i dynamicznym zarządzaniu zadaniami.

System wykorzystuje moduły analityczne obejmujące analizę EXIF, weryfikację C2PA i PRNU, analizę wizualną ELA i FFT oraz modele klasyfikacyjne oparte na CNN, ConvNeXt i CLIP. Kluczowym elementem rozwiązania jest meta-model integrujący wyniki tych metod, który osiągnął najwyższą skuteczność detekcji, przewyższając podejścia bazowe.

Przeprowadzone eksperymenty potwierdziły, że integracja informacji pochodzących z różnych poziomów analizy obrazu zwiększa odporność systemu na zróżnicowanie generatorów. Praca pokazuje, że podejście modułowe umożliwia stworzenie praktycznego systemu do analizy obrazów syntetycznych, przydatnego w kontekście bezpieczeństwa informacji i wykrywania treści generowanych sztucznie.

Spis treści

Streszczenie	5
1 Cel prac i wizja produktu	9
1.1 Informacje ogólne	9
1.2 Opis dziedziny problemu	9
1.2.1 Historia i rozwój modeli generacji obrazów	10
1.2.2 Ewolucja metod detekcji obrazów syntetycznych	12
1.3 Motywacja	13
1.4 Rola produktu	13
1.5 Wizja systemu	14
1.5.1 Ogólny opis wymagań	14
1.5.2 Ogólna koncepcja metod analizy	15
1.5.3 Konkurencyjne rozwiązania	16
1.5.4 Studium wykonalności	17
1.5.5 Analiza ryzyka	19
1.6 Wkład pracy	20
1.7 Podsumowanie rozdziału	21
2 Projekt systemu	22
2.1 Charakterystyka użytkownika systemu	22
2.2 Zakres i specyfikacja wymagań systemu	22
2.3 Opis procesów biznesowych	24
2.4 Scenariusze użytkowania i testowania	26
2.5 Makietki interfejsu użytkownika	29
2.6 Podsumowanie rozdziału	31
3 Wybrane aspekty realizacji	32
3.1 Ogólny opis projektu	32
3.1.1 Modułowa struktura systemu	32
3.1.2 Zasada działania systemu	33
3.1.3 Schemat koncepcyjny systemu	34
3.2 Stos technologiczny	34
3.2.1 Język programowania i środowisko wykonawcze	34
3.2.2 Frameworki uczenia maszynowego	34
3.2.3 Biblioteki do przetwarzania obrazu i sygnału	35
3.2.4 Modele semantyczne i analiza kontekstowa	36
3.2.5 Analiza metadanych i cech pomocniczych	36
3.2.6 Źródła danych i modele generatywne wykorzystywane w projekcie	36
3.2.7 Narzędzia pomocnicze, wdrożeniowe i testowe	37

3.2.8	Uzasadnienie wyboru stosu technologicznego	37
3.3	Moduły analityczne systemu	38
3.3.1	Moduł ekstrakcji metadanych EXIF	39
3.3.2	Moduł weryfikacji C2PA	40
3.3.3	Moduł weryfikacji PRNU	41
3.3.4	Metoda ELA z klasyfikatorem SVC (ELA, Error Level Analysis)	47
3.3.5	Metoda FFT (analiza widma częstotliwościowego)	52
3.3.6	Metoda CLIP zero-shot	55
3.3.7	Model CLIP z klasyfikatorem SVM	56
3.3.8	Model CNN StyleGAN	59
3.3.9	Model ConvNeXt Tiny do detekcji obrazów StyleGAN	60
3.3.10	Model GAN vs Diffusion	62
3.3.11	Model ConvNeXt Diffusion	63
3.3.12	Model CLIP do klasyfikacji modelu generatora	65
3.3.13	Zintegrowany pipeline decyzyjny z meta-modelem agregującym	66
3.3.14	Moduł raportowania	69
3.4	Architektura systemu	69
3.4.1	Cel i założenia architektoniczne	69
3.4.2	Ogólny podział systemu	70
3.4.3	Frontend	71
3.4.4	Backend API	72
3.4.5	Interfejs API	72
3.4.6	Rdzeń analityczny	72
3.4.7	Moduły analizy obrazów	73
3.4.8	Analiza wsadowa	73
3.4.9	Ograniczenia i bezpieczeństwo danych wejściowych	73
3.4.10	Obsługa błędów i odporność systemu	74
3.4.11	Środowisko uruchomieniowe	74
3.4.12	Podsumowanie	74
3.5	Zapewnienie jakości oprogramowania	74
3.5.1	Weryfikacja ręczna metod analizy obrazu	75
3.5.2	Prosta walidacja działania klasyfikacji	75
3.5.3	Ręczne testy działania interfejsu użytkownika	75
3.5.4	Testy jednostkowe i testy komponentów backendu	76
3.5.5	Podsumowanie	76
3.6	Podsumowanie rozdziału	77
4	Skalowalna wersja systemu	78
4.1	Architektura systemu	78
4.2	System zadań i dynamiczne zarządzanie stanem modeli	79
4.2.1	Modelowanie stanów i cykl życia zadania	79
4.2.2	Orkiestracja i StateManager	80
4.2.3	Klasyfikacja modeli: Heavy vs Utility	80
4.3	Manifest modelu jako fundament uniwersalnego interfejsu	80
4.3.1	Struktura definicji parametrów	80
4.3.2	Dynamiczne renderowanie widoku (Dynamic UI)	81
4.3.3	Zalety separacji logicznej	81
4.4	Struktura wyników analizy i zarządzanie artefaktami	81

4.4.1	Standard wyjściowy AnalyzerOutput	81
4.4.2	Proces utrwalania wyników i artefaktów	82
4.5	System zarządzania zasobami (Assets Service)	82
4.5.1	Separacja obsługi danych binarnych	82
4.5.2	Asynchroniczne przetwarzanie paczek ZIP	83
4.5.3	Biblioteka zasobów	83
4.5.4	Integracja z warstwą prezentacji	83
4.6	Podsumowanie architektury i logiki systemu	83
5	Organizacja pracy	85
5.1	Charakterystyka projektu i sposób jego realizacji	85
5.2	Osoby w projekcie	86
5.3	Zespół i podział obowiązków	86
5.4	Organizacja prac i wykorzystane narzędzia	87
5.5	Przebieg prac	87
5.6	Podsumowanie rozdziału	88
6	Wyniki projektu	89
6.1	Podsumowanie zrealizowanych funkcji finalnego produktu	89
6.2	Główne scenariusze działania	94
6.3	Wyniki testów metod detekcji	98
6.3.1	Wyniki testów metod detekcji REAL vs AI	98
6.3.2	Wyniki testów metod identyfikacji generatora obrazów	104
6.4	Podsumowanie rozdziału	107
7	Zakończenie	108
Kod źródłowy		110
Spis rysunków		114
Spis tabel		115

Rozdział 1

Cel prac i wizja produktu

1.1. Informacje ogólne

Celem projektu jest przygotowanie oprogramowania, które pozwala na rozpoznawanie obrazów wygenerowanych przez narzędzia sztucznej inteligencji.

System ma wykorzystywać nowoczesne metody uczenia maszynowego i analizy obrazów, w tym modele Contrastive Language-Image Pre-training (CLIP), sieci konwolucyjne (CNN), analizy artefaktów kompresji i szumu sensora oraz cech w dziedzinie częstotliwości (ELA, PRNU, FFT), metody oparte na weryfikacji pochodzenia treści i metadanych (C2PA) oraz podejścia oparte na transfer learning.

W projekcie przewidziano również zastosowanie rozwiązań hybrydowych, łączących analizę semantyczną obrazów z detekcją charakterystycznych artefaktów powstających w procesie generowania grafiki przez sztuczną inteligencję. Ważnym elementem jest również zapewnienie objaśnialności działania modeli, m.in. poprzez wykorzystanie metod wizualizacyjnych, takich jak Grad-CAM, pozwalających na identyfikację obszarów obrazu, które mają największy wpływ na decyzje modelu.

Efektem pracy będzie narzędzie integrujące różne algorytmy, umożliwiające skuteczne i przejrzyste rozpoznawanie obrazów generowanych sztucznie za pomocą wybranych metod oraz pozwalające na analizę wielu obrazów w trybie wsadowym.

1.2. Opis dziedziny problemu

W ostatnich latach techniki generowania obrazów przy użyciu sztucznej inteligencji osiągnęły poziom realizmu, który utrudnia rozróżnienie obrazów syntetycznych od rzeczywistych. Jak wskazują badania [18], skuteczność w rozpoznawaniu sztucznych obrazów przez ludzi wynosi jedynie około 64%. Ta trudność ma istotne znaczenie w kontekście bezpieczeństwa informacji oraz weryfikacji autentyczności treści wizualnych. W odpowiedzi na te wyzwania powstaje potrzeba opracowania metod umożliwiających automatyczne wykrywanie i klasyfikowanie obrazów generowanych przez AI.

Rozpoznawanie obrazów generowanych przez AI ma zastosowanie w wielu obszarach, w tym:

- **Bezpieczeństwo informacji i walka z dezinformacją** - wykrywanie fałszywych obrazów wykorzystywanych w mediach społecznościowych oraz materiałach promocyjnych.

- **Kontrola jakości i autentyczności treści** - w dziedzinie sztuki cyfrowej, reklamy i mediów.
- **Badania naukowe** - analiza i weryfikacja modeli generatywnych oraz ich artefaktów.
- **Weryfikacja dowodów i bezpieczeństwo publiczne** - wykrywanie manipulacji w materiałach dowodowych, co może mieć kluczowe znaczenie w sprawach sądowych i kryminalnych.

Problematyka ta wiąże się z szeregiem wyzwań technicznych:

- **Ciągły rozwój modeli generatywnych** - systemy wykrywania muszą być w stanie generalizować do nowych modeli i stylów generowanych obrazów, które mogą różnić się strukturą, artefaktami i realizmem.
- **Różnorodność artefaktów generowanych przez różne modele AI** - skuteczne wykrywanie wymaga stosowania zróżnicowanych metod analizy obrazu i cech semantycznych.
- **Skalowalność i wydajność systemu** - w praktycznych zastosowaniach konieczne jest szybkie przetwarzanie dużych zbiorów danych oraz optymalizacja modeli AI.
- **Interpretowalność decyzji modelu** - użytkownik systemu powinien móc zrozumieć, na podstawie jakich cech obrazu podjęto decyzję o klasyfikacji.

Rozwiązywanie tego problemu wymaga połączenia wiedzy z zakresu uczenia maszynowego, przetwarzania obrazów, analizy semantycznej oraz inżynierii oprogramowania. Stosowanie nowoczesnych algorytmów (CNN, CLIP, FFT), technik hybrydowych oraz mechanizmów zapewniających wydajność systemu jest kluczowe dla osiągnięcia skutecznej detekcji obrazów generowanych przez sztuczną inteligencję.

1.2.1. Historia i rozwój modeli generacji obrazów

Technologie generowania obrazów znacznie rozwinięły się na przestrzeni ostatnich lat. Postęp ten był napędzany przez rozwój metod uczenia głębokiego, a wraz z nim coraz bardziej realistyczne i złożone modele generatywne. Poniżej przedstawiono przegląd najważniejszych etapów rozwoju tych technik.

Rewolucja GAN - od 2014 roku

Znaczący przełom w generowaniu obrazów nastąpił wraz z wprowadzeniem Generative Adversarial Networks (GAN) w 2014 roku przez Iana Goodfellowa i współpracowników [6]. GAN składa się z dwóch sieci neuronowych – generatora i dyskryminatora – które uczą się w rywalizacji, co pozwala na tworzenie obrazów o wysokiej jakości i złożoności. Przykłady:

- **DCGAN** - stabilna architektura oparta na konwolucjach,
- **BigGAN** - obrazy wysokiej rozdzielczości z dużą różnorodnością klas,
- **StyleGAN / StyleGAN2** - fotorealistyczne obrazy z kontrolą nad stylem i strukturą,
- **CycleGAN, GauGAN** - translacje obraz–obraz i generacja bez nadzoru.

Modele GAN spopularyzowały zjawisko deepfake'ów i zapoczątkowały szeroką dyskusję na temat zagrożeń związanych z generowaniem syntetycznych treści. Przykładowe obrazy wygenerowane przez model GAN przedstawiono na rys. 1.1.



Rysunek 1.1: Przykładowe obrazy wygenerowane przez StyleGAN, źródło: [4]

Modele dyfuzyjne – nowa era generacji (od 2020 roku)

Modele dyfuzyjne wprowadziły znaczący skok jakościowy w generowaniu obrazów [40]. Uczę się odwracania procesu stopniowego dodawania szumu, co pozwala na tworzenie obrazów o wysokiej szczegółowości, spójności semantycznej i wizualnej, a także umożliwia kontrolowaną edycję treści. Wśród najważniejszych modeli wyróżnia się:

- **DDPM** – podstawowy model dyfuzyjny wprowadzający zasadę odwracania szumu,
- **Latent Diffusion Models (LDM)** – generacja w przestrzeni latentnej, przyspieszająca działanie,
- **Stable Diffusion** – otwartoźródłowy model umożliwiający szeroką generację obrazów i eksperymenty twórcze,
- **DALL·E 2 i 3 (OpenAI)** – modele tekst→obraz wykorzystujące techniki dyfuzyjne w połączeniu z modelami językowymi, pozwalające na dokładne odwzorowanie złożonych opisów,
- **MidJourney** – komercyjny system tekst→obraz o charakterze artystycznym.

Dzięki udostępnieniu otwartoźródłowych modeli dyfuzyjnych, takich jak Stable Diffusion i LDM, generowanie obrazów stało się szeroko dostępne, co zwiększyło zarówno możliwości twórcze, jak i zagrożenia związane z dezinformacją. Przykładowe obrazy wygenerowane przez model Stable Diffusion przedstawiono na rys. 1.2.



Rysunek 1.2: Przykładowe obrazy wygenerowane przez Stable Diffusion, źródła: [4], [37]

1.2.2. Ewolucja metod detekcji obrazów syntetycznych

Wraz z rosnącą jakością obrazów generowanych przez sztuczną inteligencję rozwijały się również metody służące do ich identyfikacji. Poniższa sekcja przedstawia główne etapy ewolucji podejść detekcyjnych, od klasycznych technik analizujących piksele po współczesne metody wykorzystujące modele wielkoskalowe.

Wczesne techniki – analiza artefaktów i sygnałów niskopoziomowych

Pierwsze metody detekcji koncentrowały się na identyfikacji artefaktów typowych dla manipulacji takich jak fotomontaż czy retusz. Obejmowały m.in.:

- analizę kompresji JPEG,
- ocenę anomalii w strukturach szumu,
- metody oparte na PRNU oraz ELA [7].

Choć techniki były w pewnym stopniu skuteczne, szybko okazało się, że mogą nie radzić sobie z coraz bardziej zaawansowanymi treściami syntetycznymi.

Detekcja obrazów generowanych przez GAN

Wraz z popularyzacją sieci GAN głównym obszarem badań stało się rozpoznawanie artefaktów charakterystycznych dla konkretnych architektur generatywnych. Wprowadzono:

- sieci CNN trenowane na zbiorach obrazów prawdziwych i syntetycznych [29],
- metody analizujące widmo częstotliwościowe obrazów [15].

Podejścia te były skuteczne, ale często słabo uogólniały się na nowe modele generatywne.

Wyzwania związane z modelami dyfuzyjnymi

Modele dyfuzyjne generują obrazy pozbawione wielu artefaktów typowych dla GAN-ów, co wymusiło opracowanie nowych metod detekcji. Zaczęto analizować głębsze cechy reprezentacji obrazu, a nie tylko jego niskopoziomowe własności.

W tej generacji metod zaczęto stosować:

- modele oparte na Vision Transformerach (ViT),
- analizę cech semantycznych z modeli takich jak CLIP [27],
- hybrydowe podejścia łączące cechy pikselowe, widmowe i semantyczne.

W wielu pracach wykazano, że najlepsze wyniki osiąga się poprzez analizę wielowarstwowych reprezentacji — różne warstwy modelu „widzą” inne aspekty obrazu, co pozwala konstrukcjom detekcyjnym działać bardziej ogólnie.

Współczesne trendy

Najbardziej zaawansowane współczesne metody detekcji wykorzystują wielkoskalowe modele wizji, często pretrenowane na danych multimodalnych. Do głównych kierunków rozwoju należą:

- łączenie reprezentacji z wielu warstw modeli ViT [31],
- wykorzystanie modeli wielomodalnych do analizy spójności treści obraz–tekst,
- adaptacyjne transformatory i hybrydy cech w nowoczesnych detektorach [23].

Metody te stanowią podstawę współczesnych systemów wykrywania treści syntetycznych.

1.3. Motywacja

Dynamiczny rozwój narzędzi generujących obrazy przez sztuczną inteligencję sprawia, że coraz trudniej odróżnić treści syntetyczne od rzeczywistych. Sytuacja ta rodzi istotne zagrożenia – od rozpowszechniania dezinformacji, przez manipulację opinią publiczną, po ryzyko wprowadzania fałszywych materiałów do postępowań sądowych.

W związku z tym pojawia się potrzeba stworzenia narzędzi wspierających rzetelną weryfikację autentyczności obrazów. Motywacją projektu jest zapewnienie społeczeństwu, instytucjom i organom ścigania środków pozwalających na wiarygodną ocenę materiałów wizualnych, co ma znaczenie zarówno w ochronie bezpieczeństwa publicznego, jak i w utrzymaniu zaufania do informacji w mediach.

Dodatkową motywacją jest szybki rozwój metod wykrywania obrazów AI. Tworzenie systemu, który może być łatwo rozbudowywany o nowe techniki detekcji, pozwala na utrzymanie jego aktualności i skuteczności w miarę pojawiania się coraz bardziej zaawansowanych narzędzi generatywnych.

Projekt daje także możliwość porównania różnych metod detekcji. Taka funkcjonalność pozwala nie tylko na wybór najlepszego podejścia dla konkretnych zastosowań, ale także na ciągłe doskonalenie algorytmów i rozwój badań nad wykrywaniem obrazów generowanych przez AI.

1.4. Rola produktu

Produkt pełni rolę narzędzia wspierającego weryfikację autentyczności obrazów, dostarczając rzetelnych informacji o ich pochodzeniu i charakterze. Umożliwia nie tylko identyfikację materiałów potencjalnie zmanipulowanych lub wygenerowanych przez sztuczną inteligencję, ale także przejrzystą analizę procesu klasyfikacji – użytkownik może zobaczyć, które obszary obrazu miały największy wpływ na decyzję modelu (np. dzięki Grad-CAM), zapoznać się z opisem stosowanych metod detekcji oraz uzyskać szczegółowe raporty z wynikami analizy, które dokumentują stopień pewności modelu i wykryte artefakty. Taka funkcjonalność wspiera zarówno działania operacyjne, jak i badania naukowe, umożliwiając świadome i uzasadnione korzystanie z wyników systemu.

System został zaprojektowany z myślą o kilku grupach użytkowników, które łączy potrzeba rzetelnej oceny pochodzenia treści graficznych, choć ich oczekiwania wobec systemu są nieco zróżnicowane.

Badacze i analitycy danych wizualnych

Pierwszą grupę stanowią **badacze i analitycy danych wizualnych**, którzy mogą wykorzystywać system do prowadzenia eksperymentów i testów porównawczych. Dla nich istotne są przejrzyste wyniki oraz dostęp do szczegółowych raportów, które wspierają proces analizy i badań nad wiarygodnością treści wizualnych.

Organy ścigania i specjaliści ds. cyberbezpieczeństwa

Drugą grupę tworzą **organy ścigania oraz specjaliści z zakresu cyberbezpieczeństwa**, dla których system stanowi wsparcie w analizie potencjalnie zmanipulowanych lub fałszywych materiałów. Największą wartość ma tu możliwość uzyskania wiarygodnej klasyfikacji oraz raportu zawierającego informacje o stopniu pewności i wykrytych nieprawidłowościach, co może być pomocne w procesach dochodzeniowych lub eksperckich.

Moderatorzy i administratorzy platform internetowych

Kolejną grupę odbiorców stanowią **moderatorzy oraz administratorzy platform internetowych**, którzy zajmują się oceną wiarygodności treści wizualnych publikowanych przez użytkowników. Dla nich kluczowy jest tryb wsadowy, który pozwala na szybką i łatwą ocenę dużej liczby obrazów. Dzięki temu system może wspierać moderatorów w podejmowaniu świadomych decyzji dotyczących oznaczania lub dalszej weryfikacji treści.

Zwykli użytkownicy Internetu

Ostatnią, ale najliczniejszą grupę stanowią **zwykli użytkownicy Internetu**, którzy coraz częściej stykają się z nieautentycznymi treściami w mediach społecznościowych. W ich przypadku system pełni rolę informacyjną i edukacyjną - umożliwia szybkie przesłanie obrazu i uzyskanie jasnej odpowiedzi, czy został on wygenerowany przez AI, co zwiększa świadomość i bezpieczeństwo w korzystaniu z treści wizualnych online.

Podsumowanie

System stanowi wszechstronne narzędzie do weryfikacji obrazów generowanych przez AI, łącząc potrzeby różnych grup użytkowników z przejrzystą analizą procesu klasyfikacji, wizualizacją obszarów wpływających na decyzję modelu, opisem stosowanych metod oraz raportowaniem wyników. Dzięki temu wspiera świadomie podejmowanie decyzji, edukację oraz dokumentowanie autentyczności treści wizualnych w środowisku cyfrowym.

1.5. Wizja systemu

1.5.1. Ogólny opis wymagań

Projektowany system powinien spełniać szereg ogólnych wymagań funkcjonalnych i nie-funkcjonalnych, które wyznaczają kierunek jego działania oraz jakość.

Wymagania funkcjonalne

- możliwość analizy obrazów pod kątem ich autentyczności,
- obsługa zarówno pojedynczych plików, jak i większych zbiorów danych,
- prezentacja wyników wraz z dodatkowymi informacjami wspierającymi interpretację,
- generowanie raportów z przeprowadzonych analiz,
- możliwość konfiguracji wybranych parametrów i metod,
- wsparcie dla porównywania różnych podejść detekcji.

Wymagania niefunkcjonalne

- zapewnienie odpowiedniej wydajności i niezawodności,
- intuicyjna i dostępna obsługa systemu,
- przenośność i łatwość uruchomienia na różnych środowiskach,
- możliwość łatwego rozszerzania o nowe metody analizy,
- bezpieczeństwo przetwarzanych danych,
- transparentność działania i wyników.

1.5.2. Ogólna koncepcja metod analizy

System umożliwia użytkownikowi wybór pojedynczych metod analizy obrazów lub skorzystanie z modułu integrującego wyniki wielu metod w jedną ocenę końcową. Obejmuje szeroki przekrój podejść, obejmujący zarówno klasyczne techniki forensyczne, analizę metadanych, jak i nowoczesne metody uczenia maszynowego.

- **Analiza niskopoziomowa (forensyczna)** – wykrywanie artefaktów i cech obrazu powstających podczas akwizycji, przetwarzania lub generacji, m.in. szum sensora (PRNU), Error Level Analysis (ELA) i analiza w dziedzinie częstotliwości (FFT).
- **Analiza pochodzenia i metadanych** – weryfikacja źródła obrazu i informacji towarzyszących, m.in. EXIF oraz C2PA.
- **Analiza semantyczna i modele uczenia maszynowego** – głębokie modele konwolucyjne, hybrydowe i multimodalne (np. CLIP), umożliwiające rozróżnianie obrazów generowanych przez różne klasy modeli AI, w tym GAN i modele dyfuzyjne.
- **Moduły integrujące wyniki** – agregacja ocen z różnych metod i prezentacja ich w sposób czytelny dla użytkownika, umożliwiająca zarówno wybór wgląd w wyniki poszczególnych metod, jak i ocenę zbiorczą.

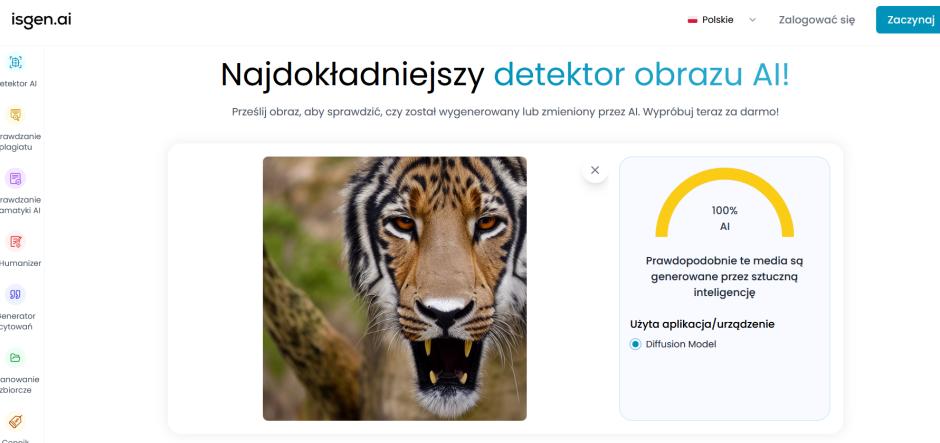
1.5.3. Konkurencyjne rozwiązania

Dynamiczny rozwój technologii generatywnej spowodował powstanie wielu narzędzi i metod służących do wykrywania obrazów syntetycznych. Istniejące rozwiązania można podzielić na dwie główne grupy: narzędzia komercyjne oraz metody badawcze publikowane w literaturze naukowej.

Narzędzia komercyjne i platformy detekcji

W ostatnich latach pojawiło się wiele serwisów internetowych oraz aplikacji komercyjnych, które oferują użytkownikom detekcję obrazów generowanych przez sztuczną inteligencję. Przykłady takich systemów obejmują:

- **isgen.ai** – platforma analizująca prawdopodobieństwo generacji obrazu przez AI, a także identyfikująca model AI, który wygenerował obraz. Sprawdza, czy obraz jest w całości generowany przez AI, modyfikowany przez AI lub zmieniany za pomocą technologii de-epfake, jednak nie zapewnia szczegółów o procesie analizy. [21] (Rys. 1.3)
- **AI or Not** – serwis oferujący klasyfikację obrazów oraz wykrywanie ich syntetycznego pochodzenia. Narzędzie działa wyłącznie online i nie udostępnia modeli ani kodu. [2]
- **Hive Moderation** – komercyjna platforma oferująca detekcję treści generowanych przez sztuczną inteligencję, w tym obrazów. System udostępnia klasyfikację obrazów pod kątem ich pochodzenia (AI vs. rzeczywiste) oraz identyfikację wybranych modeli generatywnych. Rozwiązanie działa w formie API i nie zapewnia wglądu w szczegóły architektury ani procesu decyzyjnego. [3]



Rysunek 1.3: Aplikacja isgen.ai do detekcji obrazów AI

Wszystkie wymienione narzędzia mają wspólną cechę: działają jako usługi zewnętrzne, co oznacza brak pełnej kontroli nad danymi użytkownika oraz brak możliwości integracji z lokalnym środowiskiem obliczeniowym. Ponadto często nie umożliwiają porównywania metod, analizy wielu obrazów jednocześnie, ani konfiguracji parametrów analizy.

Metody badawcze i otwartozródłowe

Równolegle w środowisku akademickim powstało wiele podejść do wykrywania obrazów syntetycznych. Obejmują one zarówno techniki oparte na analizie niskopoziomowych cech obrazu (np. szum, artefakty pikselowe), jak i metody wykorzystujące głębokie sieci neuronowe

do rozpoznawania wzorców charakterystycznych dla obrazów generowanych przez AI. W literaturze pojawiają się także rozwiązania semantyczne, które badają spójność treści wizualnej i kontekstowej, oraz podejścia hybrydowe łączące analizę cech niskiego poziomu z informacją semantyczną.

Chociaż te metody stanowią solidną podstawę teoretyczną, większość z nich funkcjonuje głównie jako fragmentaryczne implementacje lub prototypy badawcze. Często brakuje im pełnej integracji w jednym narzędziu, wsparcia dla analizy wsadowej, raportowania wyników czy wizualizacji procesu decyzyjnego, co ogranicza ich praktyczne zastosowanie.

Ograniczenia istniejących rozwiązań

Zarówno narzędzia komercyjne, jak i metody badawcze mają kilka wspólnych ograniczeń:

- brak możliwości elastycznego porównywania różnych metod detekcji,
- ograniczona interpretowalność wyników (np. brak wizualizacji Grad-CAM),
- brak integracji wielu podejść w jednym narzędziu,
- brak dostępu do pełnego procesu analizy,
- brak wersji lokalnej, umożliwiającej pracę w środowisku o ograniczonym dostępie do Internetu,
- ograniczona możliwość rozszerzania rozwiązania o nowe metody.

Różnice na tle proponowanego systemu

Projektowany system wyróżnia się podejściem modułowym, które umożliwia:

- integrację wielu odmiennych metod detekcji (CNN, CLIP, ELA, PRNU, FFT),
- łatwe dodawanie kolejnych metod analizy w formie wtyczek (pluginów),
- działanie w pełni lokalne, bez konieczności wysyłania danych do zewnętrznych serwisów,
- porównywanie skuteczności różnych metod na wspólnych zbiorach testowych,
- zapewnienie interpretowalności za pomocą wizualizacji (np. Grad-CAM),
- generowanie raportów nadających się do wykorzystania w badaniach, analizach i procesach dochodzeniowych.
- prezentowanie w aplikacji opisu działania każdej metody, tak aby użytkownik rozumiał, na jakiej podstawie metoda dokonuje klasyfikacji.

W rezultacie proponowany system wypełnia lukę pomiędzy narzędziami badawczymi a komercyjnymi, dostarczając elastyczne, transparentne i w pełni kontrolowane przez użytkownika rozwiązanie.

1.5.4. Studium wykonalności

Przy wyborze technologii celem było stworzenie elastycznej, wydajnej i łatwej w rozwoju aplikacji do rozpoznawania obrazów generowanych przez sztuczną inteligencję.

Technologie backendowe

Do implementacji logiki backendowej wybrano **FastAPI** — nowoczesny framework Python umożliwiający tworzenie wydajnych usług RESTful z asynchroniczną obsługą żądań. FastAPI zapewnia łatwą integrację z modułami uczenia maszynowego oraz automatyczne generowanie dokumentacji API, co ułatwia dalszy rozwój i testowanie systemu.

Technologie frontendowe

Interfejs użytkownika został opracowany w dwóch wariantach. Pierwszy to nowoczesna aplikacja webowa w **React** z TypeScript, wykorzystująca biblioteki shadcn/ui i lucide-react, obsługująca wizualizacje Grad-CAM, konfigurację parametrów analizy oraz przetwarzanie wsadowe plików ZIP po stronie przeglądarki. Drugi wariant stanowi frontend stworzony w **Vue.js**, umożliwiający dynamiczną prezentację wyników analiz i wizualizacji.

Modele i przetwarzanie danych

System został zaprojektowany w sposób modułowy, umożliwiający stosowanie różnych metod detekcji obrazów, zarówno klasycznych technik forensycznych, jak i modeli głębokiego uczenia. Do implementacji modeli i operacji obliczeniowych wykorzystano **PyTorch**, wybrany ze względu na wydajność, wsparcie dla GPU oraz bogatą funkcjonalność w zakresie budowy i treningu sieci neuronowych.

Do przetwarzania i analizy obrazów zastosowano biblioteki **OpenCV**, **NumPy** oraz **SciPy**, które zapewniają szybkie operacje na macierzach, przetwarzanie obrazów w różnych dziedzinach (pikselowej i częstotliwościowej) oraz łatwą integrację z modelami uczenia maszynowego.

Takie połączenie technologii pozwala na efektywne i elastyczne przetwarzanie obrazów oraz integrację wyników różnych metod detekcji.

Zarządzanie modelami i pamięcią

Aby umożliwić pracę z dużymi modelami klasyfikacyjnymi przy ograniczonej pamięci GPU (np. 12GB), wersja skalowalna systemu wspiera dynamiczne ładowanie wag modeli. Modele są ładowane do pamięci tylko w momencie ich użycia, a po zakończeniu analizy pamięć GPU jest zwalniana, co pozwala na równoczesne korzystanie z różnych metod detekcji bez przekraczania dostępnych zasobów.

Podsumowanie

Wybrane technologie zostały dobrane w sposób zapewniający:

- wydajną obsługę zarówno analizy pojedynczych obrazów, jak i trybu wsadowego,
- łatwą rozbudowę o nowe metody detekcji,
- przejrzystą prezentację wyników oraz mechanizmów działania metod dla użytkownika,
- możliwość pracy lokalnej, bez przesyłania danych do zewnętrznych serwisów.

Analiza technologiczna i przygotowane prototypy wykazały, że wybrane rozwiązania są spójne, nowoczesne i odpowiednie do realizacji celu projektu.

1.5.5. Analiza ryzyka

Analizę ryzyka przedstawiono w tabeli 1.1.

Tabela 1.1: Analiza ryzyka dla projektu systemu rozpoznawania obrazów AI

Ryzyko	Przeciwdziałanie	Prawdopodobieństwo
Ograniczenia pamięci GPU przy dużych modelach (max 12 GB)	Dynamiczne ładowanie modeli, kolejkowanie analiz, optymalizacja wykorzystania pamięci	Wysokie
Nieintuicyjny interfejs dla użytkowników nietechnicznych (np. policjantów)	Opracowanie intuicyjnej dokumentacji użytkownika, zapewnienie łatwego uruchomienia aplikacji (np. przy użyciu Dockera)	Średnie
Opóźnienia w analizie wsadowej przy dużych zbiorach obrazów	Optymalizacja przetwarzania wsadowego, wykorzystanie GPU, równolegle uruchamianie analiz	Średnie
Trudności w dodawaniu nowych metod detekcji przez użytkownika	Stworzenie intuicyjnego i elastycznego interfejsu wtyczek	Średnie
Zbyt niska skuteczność metod detekcji	Wykorzystanie wielu modeli o różnych architekturach, trenowanych na różnych zbiorach danych	Wysokie
Trudności w pozyskaniu wystarczająco zróżnicowanych danych treningowych i testowych	Wykorzystanie obrazów z wielu publicznych zbiorów danych, obejmujących obrazy generowane przez różne modele AI	Wysokie
Trudności z detekcją nieznanego dla modelu generatora obrazów	Wdrożenie klasyfikacji na poziomie rodziny generatorów zamiast poszczególnych wersji, co umożliwia detekcję obrazów z nieznanymi modeli	Wysokie

Podsumowanie działań w zakresie ryzyka

W trakcie przygotowywania prototypu systemu zauważono pewne ograniczenia, m.in. związane z wykorzystaniem pamięci GPU przy dużych modelach. Zostały one zaadresowane poprzez zaimplementowanie skalowalnej wersji systemu, która wprowadza mechanizm dynamicznego ładowania modeli, co poprawiło wydajność analizy. Pozostałe zidentyfikowane ryzyka uwzględniono na etapie projektowania i trenowania modeli lub wyeliminowano podczas wstępnego testowania, stosując praktyki minimalizujące ich wystąpienie. W efekcie większości ryzyk udało się zapobiec przed integracją modeli z aplikacją, a system jest przygotowany do dalszego bezpiecznego rozwoju i rozszerzania o nowe metody detekcji.

Podsumowanie wizji

Projektowany system jest elastyczną i modułową platformą do wykrywania obrazów generowanych przez sztuczną inteligencję, integrującą zarówno klasyczne metody forensyczne, analizę metadanych, jak i nowoczesne modele uczenia maszynowego. Umożliwia pracę w pełni lokalną, wybór pojedynczych metod lub integrację wyników wielu analiz w celu uzyskania zbiorczej oceny autentyczności obrazu, przy zachowaniu interpretowalności wyników poprzez wizualizacje, np. Grad-CAM. Dzięki zastosowaniu nowoczesnych technologii backendowych i frontendowych system jest wydajny, skalowalny oraz gotowy do pracy z dużymi zbiorami danych i ograniczoną pamięcią GPU. Rozwiązań projektowych oraz mechanizmy zarządzania ryzykiem zapewniają niezawodność, bezpieczeństwo danych i łatwość rozwoju, co pozwala systemowi wypełniać lukę między prototypami badawczymi a narzędziami komercyjnymi.

1.6. Wkład pracy

W ramach realizowanej pracy przygotowano kompletny system służący do rozpoznawania obrazów generowanych przez narzędzia sztucznej inteligencji oraz jego skalową wersję. Wkład obejmuje zarówno opracowanie warstwy analitycznej, jak i stworzenie w pełni działającej aplikacji wyposażonej w interfejs użytkownika oraz tryb wsadowy. Najważniejsze elementy wkładu własnego obejmują:

- **Wybór, implementacja i integracja metod detekcji oraz klasyfikacji obrazów**, obejmujących techniki oparte na sieciach konwolucyjnych, modelach CLIP, podejściach transfer learning, metody forensyczne (FFT, PRNU, ELA) oraz analizę metadanych. System pozwala na określenie, czy obraz jest rzeczywisty czy syntetyczny oraz, w przypadku obrazów syntetycznych, wskazanie prawdopodobnej technologii użytej do ich tworzenia. Każda metoda została zaadaptowana do wspólnej architektury aplikacji.
- **Wdrożenie funkcji objaśnialności**, umożliwiających prezentację map aktywacji Grad-CAM w celu wizualizacji obszarów istotnych dla klasyfikacji oraz wyświetlanie artefaktów wykrywanych przez wybrane metody analizy. Funkcjonalność ta jest dostępna w wynikach zwracanych przez modele i prezentowana w interfejsie użytkownika.
- **Zaprojektowanie i implementacje systemu raportowania**, generującego szczegółowe dane o wynikach analizy, poziomie pewności modeli i wykrytych artefaktach. Raporty obejmują pliki JSON dla pojedynczych obrazów oraz zbiorczy plik CSV.
- **Stworzenie dwóch wersji lokalnej aplikacji działającej w trybie interaktywnym i wsadowym**, umożliwiającej prostą i intuicyjną obsługę. Wersja skalowalna została zoptymalizowana pod kątem pracy na komputerze z kartą graficzną do 12 GB VRAM.
- **Opracowanie modularnej architektury rozszerzalnej** o kolejne metody detekcji.
- **Przygotowanie zbioru danych testowych** z wykorzystaniem obrazów generowanych przez różne modele sztucznej inteligencji (GAN, Stable Diffusion, DALL·E).
- **Przeprowadzenie testów wydajnościowych i jakościowych**, obejmujących porównanie różnych metod, ich konfiguracji, skuteczności, szybkości działania oraz odporności na błędy danych wejściowych.

Realizacja wszystkich tych elementów doprowadziła do powstania spójnych i funkcjonalnych narzędzi, które umożliwiają analizę obrazów zarówno w zastosowaniach operacyjnych, jak i badawczych, zapewniając jednocześnie transparentność działania i możliwość dalszego rozwoju.

1.7. Podsumowanie rozdziału

W rozdziale pierwszym przedstawiono kontekst i motywację dla realizacji projektu systemu rozpoznawania obrazów generowanych przez sztuczną inteligencję. Omówiono znaczenie problemu zarówno w kontekście weryfikacji autentyczności treści wizualnych, jak i zastosowań naukowych oraz operacyjnych. Przedstawiono także historię i rozwój technologii generowania oraz detekcji sztucznych obrazów.

Scharakteryzowano główne grupy użytkowników systemu, wskazując ich potrzeby i oczekiwania, co pozwoliło określić rolę produktu w różnych scenariuszach zastosowań.

Przedstawiono również wizję systemu, w tym ogólne wymagania funkcjonalne i niefunkcjonalne oraz wybrane podejścia detekcji obrazów, podkreślając znaczenie modułowej architektury, elastyczności i interpretowalności wyników.

Dodatkowo przedstawiono przegląd istniejących rozwiązań konkurencyjnych, wskazując ich główne ograniczenia oraz różnice w stosunku do projektowanego systemu, w tym szerszy zakres stosowanych metod detekcji, możliwość integracji wyników wielu podejść oraz większą elastyczność i interpretowalność prezentowanych wyników.

W ramach Studium Wykonalności przeprowadzono analizę technologiczną obejmującą wybór narzędzi i frameworków niezbędnych do implementacji systemu, wykazując ich adekwatność do charakteru projektu i realizowanego systemu.

Dokonano także wstępnej analizy ryzyka związanego z projektem, identyfikując potencjalne problemy, takie jak ograniczenia pamięci GPU, trudności w dodawaniu nowych metod czy wykrywaniu obrazów generowanych przez nieznane modele. Wskazano strategie minimalizacji tych ryzyk na etapie trenowania modeli i przygotowania prototypu systemu.

Całość stanowi solidną podstawę do szczegółowego opisu projektu, analiz technicznych i realizacji systemu w dalszej części pracy.

Rozdział 2

Projekt systemu

2.1. Charakterystyka użytkownika systemu

System jest przeznaczony głównie do użytku lokalnego i posiada jednego podstawowego aktora - **użytkownika aplikacji**. Projekt nie przewiduje podziału na role ani zróżnicowanych poziomów dostępu. Każdy użytkownik korzystający z aplikacji ma dostęp do pełnego zestawu funkcjonalności, obejmującego przesyłanie obrazów, otrzymywanie wyników klasyfikacji oraz generowanie raportu.

Takie uproszczenie architektury pozwala skupić się na funkcjonalności detekcji obrazów generowanych przez AI, bez konieczności wprowadzania mechanizmów zarządzania uprawnieniami.

2.2. Zakres i specyfikacja wymagań systemu

W ramach bieżącego projektu zrealizowano pełny zakres zaplanowanych funkcjonalności, obejmujący zarówno podstawowe, jak i rozszerzone elementy systemu rozpoznawania automatycznie generowanych obrazów. Celem było stworzenie kompletnego narzędzia, które umożliwia użytkownikom skuteczne i wygodne analizowanie autentyczności materiałów wizualnych oraz ocenę skuteczności zastosowanych metod.

Szczegółowe wymagania funkcjonalne i niefunkcjonalne wraz z priorytetami zostały przedstawione w tabelach. Priorytety zostały określone zgodnie z metodą **MoSCoW**. Metoda ta pozwala na kategoryzację wymagań według ich znaczenia dla projektu:

- **MUST** (musi być zrealizowane) - kluczowe funkcje, których implementacja jest niezbędna do spełnienia podstawowych celów systemu,
- **SHOULD** (powinno być zrealizowane) - ważne funkcje, których brak nie uniemożliwia działania systemu, ale ogranicza jego użyteczność,
- **COULD** (może być zrealizowane) - funkcje opcjonalne, które zwiększą wygodę lub funkcjonalność, lecz nie są kluczowe dla głównych celów projektu,

Wymagania funkcjonalne

Wymagania funkcjonalne definiują konkretne działania, które system ma umożliwić użytkownikowi oraz sposoby interakcji z aplikacją. Funkcje, które powinien spełniać system, przedstawiono w tabeli [2.1](#).

Tabela 2.1: Wymagania funkcjonalne systemu wraz z priorytetami

Wymaganie funkcjonalne	Priorytet
Możliwość przesyłania pojedynczych obrazów do analizy w popularnych formatach (JPEG, PNG, BMP).	MUST
Możliwość analizy wsadowej wielu obrazów poprzez przesłanie pliku archiwum ZIP z plikami graficznymi.	MUST
Klasyfikacja obrazów jako rzeczywiste (<i>REAL</i>) lub wygenerowane przez sztuczną inteligencję (<i>AI</i>).	MUST
Prezentacja poziomu pewności klasyfikacji w postaci wartości liczbowej lub procentowej.	SHOULD
Możliwość wyboru zestawu metod analizy wykorzystywanych podczas detekcji obrazu.	MUST
Prezentacja opisu działania każdej metody analitycznej.	SHOULD
Generowanie wizualizacji obszarów decyzyjnych modeli (np. map Grad-CAM).	MUST
Prezentacja metadanych EXIF, jeśli są dostępne w obrazie.	SHOULD
Agregacja wyników wielu metod analizy i prezentacja ich w spójnej formie.	MUST
Generowanie raportu z podsumowaniem analizy obrazów w formacie JSON.	MUST
Generowanie zbiorczego raportu z analizy wsadowej w formacie CSV.	MUST
Uwzględnienie w raporcie dodatkowych miar statystycznych.	COULD
Eksport wyników analizy z poziomu interfejsu użytkownika.	MUST
Obsługa błędów przy przesyłaniu nieodpowiednich formatów plików.	MUST
Możliwość konfiguracji parametrów analizy, takich jak próg decyzyjny (<i>threshold</i>).	SHOULD
Identyfikacja potencjalnej metody generacji obrazu (np. GAN, model dyfuzyjny).	MUST

Wymagania niefunkcjonalne

Wymagania niefunkcjonalne opisują ogólne właściwości jakościowe systemu, mające wpływ na jego użyteczność, wydajność i niezawodność. Założenia, które powinien spełniać system, przedstawiono w tabeli 2.2 :

Tabela 2.2: Wymagania niefunkcjonalne systemu wraz z priorytetami

Wymaganie niefunkcjonalne	Priorytet
Zapewnienie analizy obrazów w rozsądny czasie, zarówno w trybie pojedynczym, jak i wsadowym.	SHOULD
Odporność na błędy pojedynczych metod analizy i zapewnienie ciągłości działania.	MUST
Intuicyjny i dostępny przez przeglądarkę internetową interfejs użytkownika.	SHOULD
Przenośność i możliwość uruchomienia aplikacji na systemach Windows i Linux.	COULD
Architektura systemu umożliwiająca łatwe dodawanie nowych metod analizy.	SHOULD
Zapewnienie transparentności działania poprzez prezentację szczegółowych wyników metod analizy.	MUST
Reagowanie na błędne lub nieobsługiwane pliki komunikatem informacyjnym.	MUST
Zapewnienie stabilnej i powtarzalnej skuteczności klasyfikacji w ramach możliwości zastosowanych metod detekcji.	MUST
Możliwość lokalnego uruchomienia z wykorzystaniem GPU o ograniczonej pamięci (np. do 12 GB).	SHOULD

Podsumowanie

Zrealizowany system spełnia wszystkie założone wymagania funkcjonalne i niefunkcjonalne, tworząc spójne i kompletne narzędzie do rozpoznawania obrazów generowanych przez sztuczną inteligencję. Aplikacja jest elastyczna, odporna na błędy i umożliwia przejrzystą prezentację wyników analizy zarówno w formie interaktywnej, jak i wsadowej. Ponadto, dzięki możliwości konfiguracji metod i parametrów, system może być z łatwością dostosowywany do różnych scenariuszy działania, stanowiąc solidną podstawę dla dalszego rozwoju w zakresie detekcji treści syntetycznych.

2.3. Opis procesów biznesowych

Aplikacja została zaprojektowana w celu weryfikacji autentyczności obrazów cyfrowych poprzez określenie, czy dany plik graficzny został wygenerowany przez system sztucznej inteligencji, czy też stanowi materiał rzeczywisty. System udostępnia funkcjonalność analizy obrazów w dwóch trybach pracy: interaktywnym oraz wsadowym.

Niezależnie od wybranego trybu, realizowany jest ten sam logiczny proces biznesowy, obejmujący przyjęcie danych wejściowych, ich weryfikację, przeprowadzenie analizy oraz wygenerowanie wyników w ustrukturyzowanej postaci. Różnice pomiędzy trybami dotyczą jedynie sposobu przekazania danych oraz formy prezentacji rezultatów.

Dane wejściowe

Danymi wejściowymi systemu są pliki graficzne w popularnych formatach, takich jak JPEG, PNG oraz BMP.

- W trybie interaktywnym użytkownik przesyła pojedynczy obraz do analizy.
- W trybie wsadowym użytkownik przesyła archiwum ZIP zawierające wiele obrazów, które są automatycznie rozpakowywane i analizowane seryjnie.

Użytkownik może dodatkowo określić konfigurację analizy, w tym zestaw metod, które mają zostać wykorzystane. Szczegóły implementacyjne poszczególnych metod nie są istotne z perspektywy procesu biznesowego.

Przebieg procesu analizy

Proces analizy obrazu składa się z następujących etapów:

1. Inicjalizacja systemu Po uruchomieniu aplikacji przygotowywane jest środowisko pracy, a dostępne moduły analizy obrazu są ładowane i rejestrowane w systemie. Zapewnia to gotowość aplikacji do przetwarzania danych.

2. Przyjęcie i weryfikacja danych System odbiera dane wejściowe od użytkownika, sprawdza ich poprawność oraz egzekwuje ograniczenia dotyczące formatu i rozmiaru plików. Na tym etapie dane są przygotowywane do dalszego przetwarzania.

3. Analiza obrazu Dla każdego obrazu uruchamiany jest proces analizy z wykorzystaniem wybranych metod detekcji. Wyniki poszczególnych metod są następnie agregowane w spójny rezultat, obejmujący wynik klasyfikacji oraz wartości liczbowe opisujące decyzję systemu.

4. Generowanie wyników Po zakończeniu analizy system generuje raporty zawierające wyniki przetwarzania. W zależności od trybu pracy, wyniki są prezentowane użytkownikowi bezpośrednio w interfejsie lub udostępniane w postaci plików do pobrania.

5. Obsługa błędów System jest odporny na błędy pojedynczych metod analitycznych. Niepowodzenie jednej z metod nie przerywa całego procesu analizy, a informacje o błędach są zapisywane w raportach wynikowych.

Rezultaty procesu

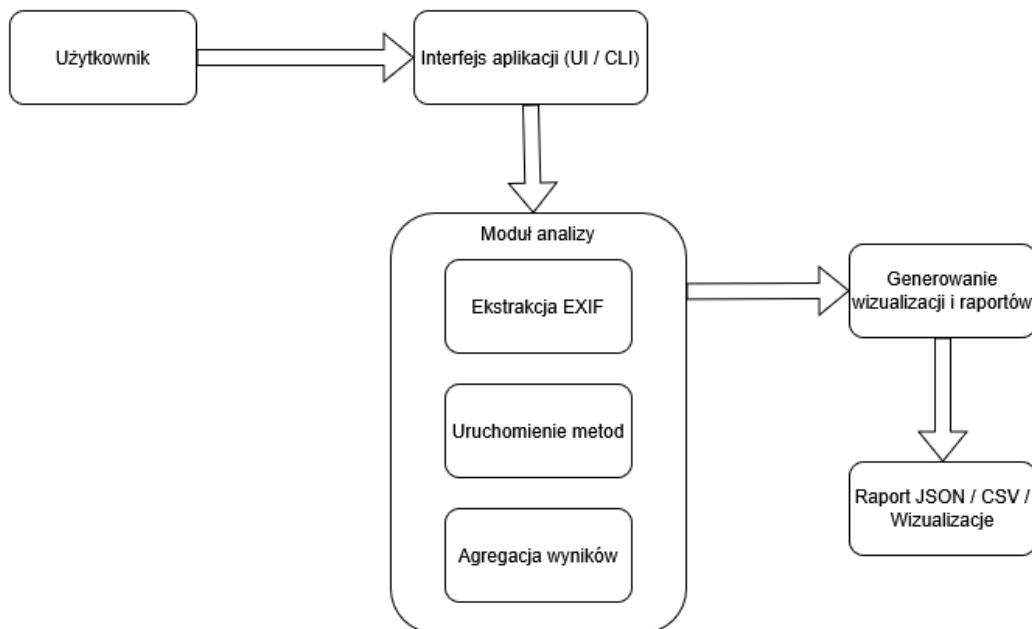
Efektem działania systemu są kompletne dane wynikowe, które mogą być wykorzystywane do dalszej analizy. Rezultaty obejmują:

- wynik klasyfikacji obrazu,
- wartości liczbowe opisujące decyzję systemu,
- wybrane metadane i informacje diagnostyczne,
- raporty w formatach **JSON** oraz **CSV**.

Opisany proces biznesowy przedstawia pełny przepływ informacji od momentu przekazania danych wejściowych aż do uzyskania końcowych wyników analizy.

Diagram 2.1 przedstawia wysokopoziomowy przepływ danych w systemie. Ilustruje on główne etapy przetwarzania informacji, od momentu przekazania danych wejściowych przez użytkownika, poprzez warstwę API i logikę aplikacyjną, aż do wygenerowania wyników analizy oraz raportów końcowych.

Diagram ma charakter poglądowy i nie odwzorowuje szczegółowej logiki algorytmów analitycznych, koncentrując się wyłącznie na relacjach pomiędzy głównymi komponentami systemu.

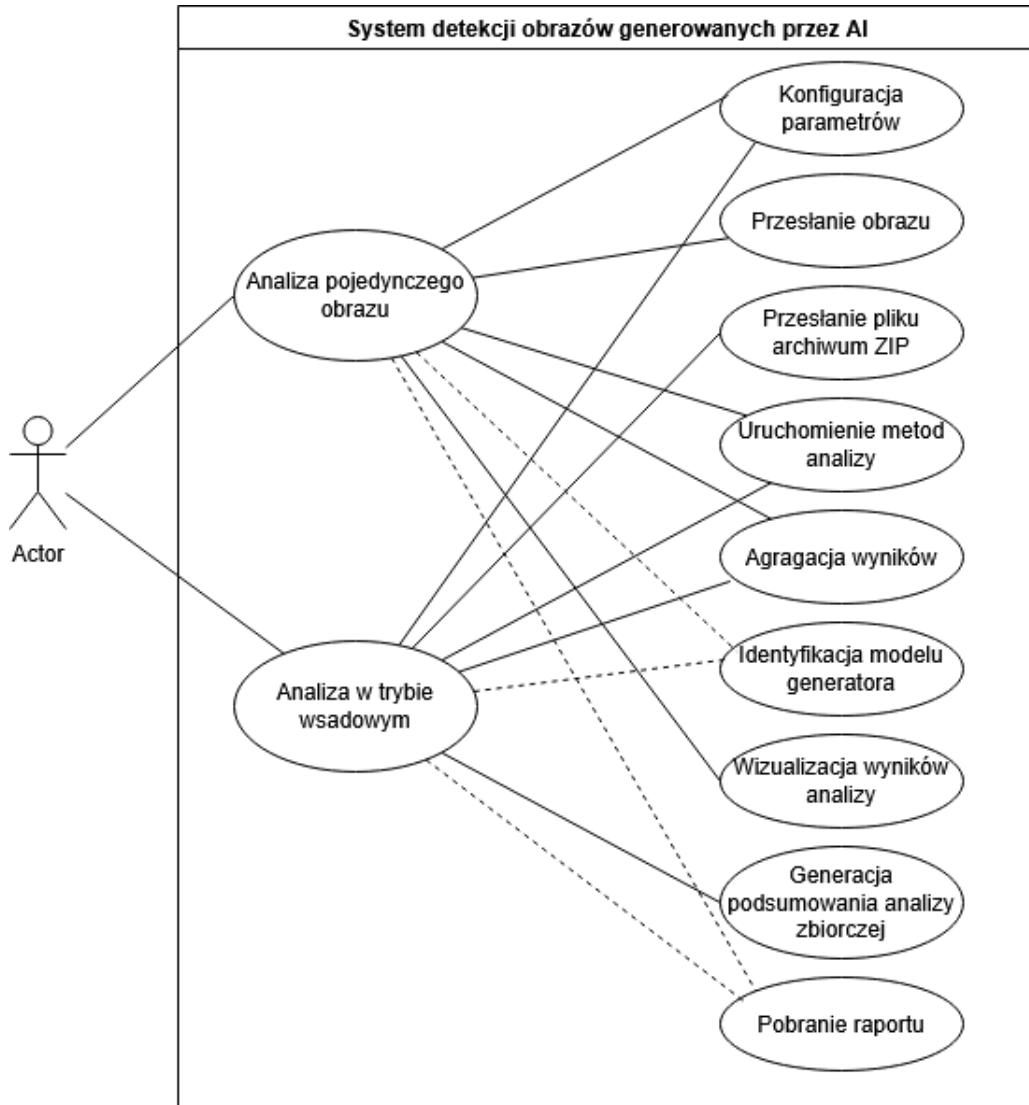


Rysunek 2.1: Diagram przepływu danych (DFD) w systemie

2.4. Scenariusze użytkowania i testowania

W niniejszej sekcji przedstawiono przykładowe scenariusze użytkowania oraz testowania aplikacji. Scenariusze te odzwierciedlają typowe sytuacje, w których system jest wykorzystywany w praktyce, a także przypadki testowe służące weryfikacji poprawności i stabilności działania aplikacji.

W celu ogólnego zobrazowania interakcji użytkownika z systemem oraz zakresu funkcjonalności aplikacji, na rysunku 2.2 przedstawiono diagram przypadków użycia (UML Use Case), stanowiący punkt odniesienia dla opisanych dalej scenariuszy.



Rysunek 2.2: Diagram przypadków użycia systemu detekcji obrazów generowanych przez AI

Scenariusz 1 – Analiza pojedynczego obrazu

Cel: Weryfikacja autentyczności pojedynczego obrazu z wykorzystaniem interfejsu użytkownika.

Przebieg:

1. Użytkownik uruchamia aplikację w trybie interaktywnym.
2. Przesyła pojedynczy obraz w obsługiwany formacie.
3. System rozpoczyna proces analizy obrazu.
4. Po zakończeniu analizy prezentowany jest wynik klasyfikacji oraz dane uzupełniające.
5. Użytkownik może pobrać raport z analizy w formacie JSON oraz zestawienie wyników w formacie CSV.

Rezultat: System poprawnie analizuje obraz i prezentuje kompletne wyniki bez wystąpienia błędów.

Scenariusz 2 – Analiza wsadowa wielu obrazów

Cel: Jednoczesna analiza wielu obrazów oraz uzyskanie zbiorczych wyników przetwarzania.
Przebieg:

1. Użytkownik wybiera tryb wsadowy w aplikacji.
2. Przesyła archiwum ZIP zawierające wiele obrazów.
3. System automatycznie rozpakowuje archiwum i analizuje każdy obraz niezależnie.
4. Po zakończeniu przetwarzania generowane są raporty dla poszczególnych obrazów oraz zbiorcze zestawienie wyników.

Rezultat: System poprawnie przetwarza wszystkie obrazy i generuje kompletne dane zbiorcze.

Scenariusz 3 – Obsługa niepoprawnych danych wejściowych

Cel: Sprawdzenie odporności systemu na błędne lub nieobsługiwane dane wejściowe.
Przebieg:

1. Użytkownik przesyła plik w nieobsługiwany formacie.
2. System wykrywa nieprawidłowe dane wejściowe.
3. Aplikacja odrzuca dane i zwraca komunikat o błędzie.

Rezultat: System nie przerywa działania i informuje użytkownika o problemie w czytelny sposób.

Scenariusz 4 – Testowanie działania systemu

Cel: Weryfikacja poprawności działania systemu na zestawie danych testowych.
Przebieg:

1. Operator testowy przygotowuje zbiór obrazów testowych.
2. System analizuje obrazy w trybie wsadowym.
3. Generowane raporty i zestawienia wyników stanowią podstawę do dalszej, zewnętrznej analizy.

Rezultat: System generuje kompletne dane wynikowe umożliwiające ocenę poprawności działania aplikacji.

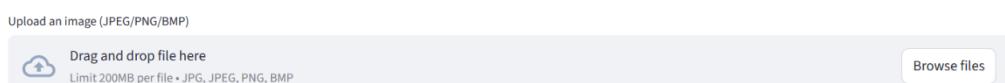
Podsumowanie

Przedstawione scenariusze użytkowania i testowania potwierdzają poprawność działania systemu w typowych oraz granicznych przypadkach użycia. Opisane procesy pokazują, że aplikacja spełnia założone wymagania funkcjonalne oraz jest stabilna i odporna na błędy danych wejściowych.

2.5. Makiety interfejsu użytkownika

W celu zobrazowania sposobu interakcji użytkownika z aplikacją przygotowano makiety interfejsu użytkownika (rys. 2.3–2.6). Przedstawiają one kluczowe ekranы systemu oraz rozmieszczenie elementów funkcjonalnych, takich jak pola wczytywania obrazów, ustawienia oraz sekcje prezentujące wyniki. Makiety te pełnią rolę koncepcyjnego projektu interfejsu, wspierając proces projektowania i weryfikacji użyteczności systemu przed jego implementacją.

AI-Image Detector



Rysunek 2.3: Makieta głównego ekranu aplikacji – przesyłanie obrazu do analizy

Result

Verdict

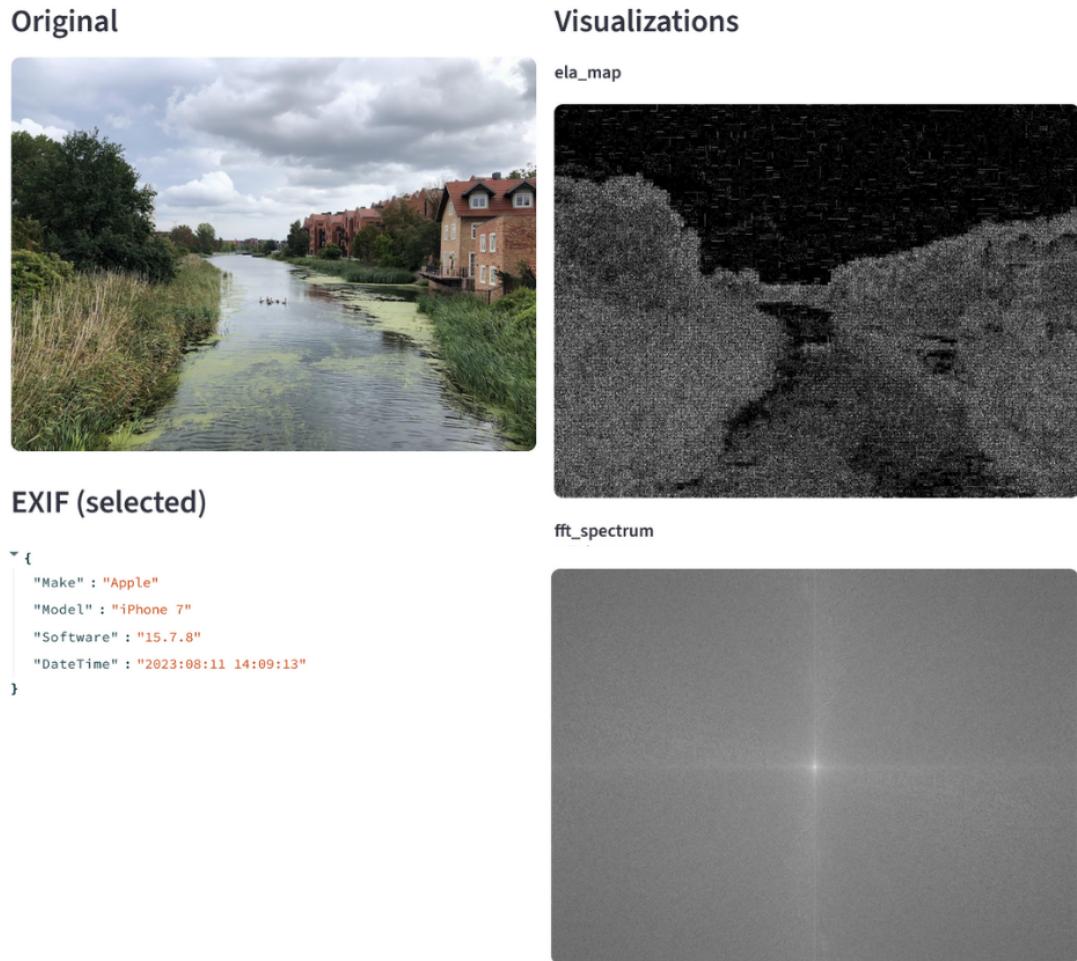
REAL

↑ score=0.408

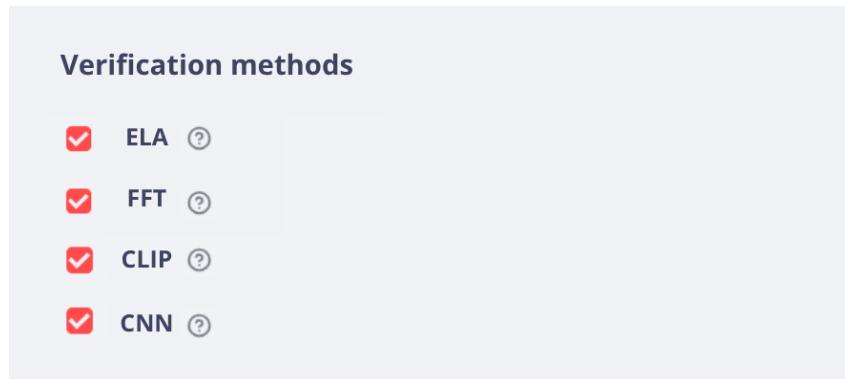
> Metric details

Download JSON report

Rysunek 2.4: Makieta wyników analizy



Rysunek 2.5: Makieta wizualizacji analizy i metadanych EXIF



Rysunek 2.6: Makieta wyboru metod weryfikacji obrazów

2.6. Podsumowanie rozdziału

W tym rozdziale przedstawiono zakres funkcjonalności projektowanego systemu rozpoznawania obrazów generowanych przez sztuczną inteligencję, obejmujący definicję głównego aktora systemu, wymagania systemowe oraz opis kluczowych procesów biznesowych.

Szczegółowo określono wymagania funkcjonalne i niefunkcjonalne, uporządkowane zgodnie z metodą MoSCoW, co pozwoliło na jasne wskazanie priorytetów projektowych. Opisano również proces przetwarzania danych, od momentu wczytania obrazów, poprzez analizę z wykorzystaniem wielu metod detekcji i klasyfikacji, aż po agregację oraz prezentację wyników w postaci raportów i wizualizacji.

Rozdział uzupełniono o scenariusze użytkowania i testowania, które ilustrują praktyczne zastosowanie systemu w różnych trybach pracy oraz potwierdzają poprawność i niezawodność jego działania. Przedstawione makiety interfejsu użytkownika umożliwiły zobrazowanie sposobu interakcji z aplikacją oraz struktury prezentowanych wyników.

Rozdział 3

Wybrane aspekty realizacji

3.1. Ogólny opis projektu

Celem opracowanego systemu jest automatyczna analiza obrazów cyfrowych w celu określania ich pochodzenia oraz rozróżnienia obrazów wygenerowanych przez narzędzia sztucznej inteligencji od obrazów pochodzących z rzeczywistych urządzeń rejestrujących. System stanowi narzędzie wspierające weryfikację autentyczności treści wizualnych, łącząc klasyczne metody z zakresu *digital image forensics* z nowoczesnymi technikami uczenia maszynowego.

Projektowany system ma charakter aplikacji uruchamianej lokalnie, składającej się z warstwy interfejsu użytkownika oraz warstwy backendowej realizującej właściwe przetwarzanie obrazu. Interfejs użytkownika uruchamiany jest w przeglądarce internetowej i komunikuje się z backendem, który odpowiada za uruchamianie poszczególnych metod detekcji, agregację wyników oraz generowanie raportów.

3.1.1. Modułowa struktura systemu

Architektura systemu oparta jest na koncepcji niezależnych modułów analizy, z których każdy realizuje odrębną metodę detekcji obrazów syntetycznych. Moduły te obejmują zarówno klasyczne techniki forensyczne, jak i metody oparte na uczeniu maszynowym. Każdy moduł posiada jednolity interfejs wejścia i wyjścia, dzięki czemu może być uruchamiany samodzielnie lub w połączeniu z innymi metodami.

Do głównych komponentów systemu należą:

- **Interfejs użytkownika (frontend)** – umożliwia wczytywanie obrazów, wybór metod analizy, konfigurację parametrów (np. progu decyzyjnego) oraz prezentację wyników w formie liczbowej i wizualnej.
- **Backend analityczny** – odpowiada za obsługę logiki aplikacji, przekazywanie obrazu do wybranych modułów analitycznych, zbieranie wyników oraz ich dalsze przetwarzanie.
- **Moduły analizy obrazu** – zbiór niezależnych metod detekcji, obejmujących m.in.:
 - analizę sygnału PRNU,
 - metodę Error Level Analysis (ELA),
 - analizę widmową w dziedzinie częstotliwości (FFT),
 - analizę metadanych obrazu (EXIF, C2PA),

- model CLIP zeroshot,
 - model CLIP z klasyfikatorem SVM,
 - modele CNN i ConvNeXt wytrenowane na obrazach generowanych przez StyleGAN,
 - model ConvNeXt do klasyfikacji obrazów GAN vs. diffusion,
 - model ConvNeXt do rozpoznawania rodzin modeli dyfuzyjnych,
 - model CLIP do rozpoznawania nowoczesnych modeli dyfuzyjnych.
- **Moduł integrujący** – moduł odpowiedzialny za łączenie wyników pochodzących z wielu metod detekcji w jedną ocenę końcową, wykorzystywany w trybie analizy łącznej.
 - **Moduł raportowania** – generuje raporty z analizy, w szczególności w formacie JSON oraz CSV, zawierające wyniki cząstkowe, wynik końcowy oraz dane pomocnicze.

Zastosowanie takiej struktury umożliwia elastyczne konfigurowanie procesu analizy przez użytkownika, a także ułatwia dalszą rozbudowę systemu o nowe metody detekcji.

3.1.2. Zasada działania systemu

Działanie systemu można opisać jako sekwencję następujących kroków:

1. **Wczytanie i przygotowanie obrazu** — system przyjmuje pojedynczy plik graficzny lub folder, weryfikuje format oraz wykonuje operacje wstępne, takie jak ujednolicenie przestrzeni barw i normalizacja danych wejściowych. Równolegle odczytywane są metadane EXIF, które są prezentowane użytkownikowi jako informacja kontekstowa.
2. **Konfiguracja analizy przez użytkownika** — użytkownik wybiera jedną lub więcej metod detekcji albo uruchamia tryb klasyfikacji łącznej. Dodatkowo użytkownik może ustawić próg decyzyjny, określający od jakiej wartości wyniku obraz ma zostać uznany za wygenerowany przez AI (domyślnie próg wynosi 0,5).
3. **Analiza obrazu** - Obraz przekazywany jest do backendu, który inicjuje działanie wybranych modułów analitycznych lub modułu integrującego. Każda metoda przetwarza obraz niezależnie i zwraca wynik liczbowy oraz dane pomocnicze.
4. **Prezentacja wyniku i interpretacja** — w przypadku analizy pojedynczego obrazu aplikacja wyświetla użytkownikowi wyniki metod, wygenerowane wizualizacje charakterystyczne dla danej metody, dodatkowe metryki oraz końcowy *score*. Na podstawie ustalonego progu system formułuje decyzję klasyfikacyjną (obraz AI / obraz rzeczywisty), przy czym użytkownik ma możliwość zmiany progu w zależności od oczekiwanej kompromisu między liczbą wykryć a liczbą fałszywych alarmów.
5. **Generowanie raportu i eksport** — wyniki analizy mogą zostać zapisane w postaci raportu, w szczególności w formacie JSON, zawierającego wynik końcowy, wyniki cząstkowe, metryki oraz informacje dodatkowe. W trybie wsadowym raport generowany jest dla całego zbioru obrazów w formacie CSV.

Taki sposób działania pozwala na porównywanie skuteczności poszczególnych metod, ich łączenie w ramach jednego procesu decyzyjnego oraz zachowanie przejrzystości całego procesu analizy.

3.1.3. Schemat koncepcyjny systemu

Na rysunku 3.4 przedstawiono wysokopoziomowy schemat architektury systemu, ilustrujący główne komponenty oraz przepływ danych pomiędzy nimi. Diagram ukazuje interakcję pomiędzy interfejsem użytkownika, backendem, oraz modułami detekcji.

3.2. Stos technologiczny

W celu realizacji systemu wykrywania obrazów generowanych przez sztuczną inteligencję zastosowano zestaw technologii obejmujący język programowania, frameworki uczenia maszynowego, biblioteki do przetwarzania obrazu i sygnału, a także narzędzia pomocnicze wykorzystywane na etapie trenowania modeli, kalibracji metod, wdrożenia systemu oraz wizualizacji wyników. Dobór poszczególnych technologii został podkutowany ich wydajnością, stabilnością, dostępnością implementacji wymaganych algorytmów oraz możliwością dalszej rozbudowy systemu.

3.2.1. Język programowania i środowisko wykonawcze

Cały system został zaimplementowany w języku **Python 3.11**, który zapewnia bogaty ekosystem bibliotek do analizy obrazu, uczenia maszynowego oraz przetwarzania sygnałów. Python został wybrany ze względu na szeroką dostępność gotowych narzędzi forensycznych, możliwość szybkiego prototypowania metod oraz pełne wsparcie dla obliczeń akcelerowanych sprzętowo.

Środowisko wykonawcze systemu obejmuje:

- **CUDA 11.x** oraz **cuDNN**, wykorzystywane do przyspieszenia obliczeń tensorowych na procesorach graficznych,
- środowiska wirtualne **Python venv** oraz **Conda**, umożliwiające izolację zależności i reprodukwalność środowiska,
- komputery wyposażone w układy **NVIDIA GPU**, niezbędne do obsługi modeli CLIP, ConvNeXt, Stable Diffusion oraz do trenowania własnych modeli generatywnych.

Do celów treningu i testowania własnych modeli wykorzystywano również środowisko **Google Colab** z dostępem do kart **GPU T4**, co umożliwiało przeprowadzanie eksperymentów w chmurze bez potrzeby posiadania lokalnego sprzętu GPU.

Równolegle przygotowano wariant środowiska uruchomieniowego systemu działający w trybie CPU, przeznaczony do wdrożeń serwerowych oraz testów integracyjnych.

3.2.2. Frameworki uczenia maszynowego

PyTorch

Podstawowym frameworkiem uczenia maszynowego wykorzystywanym w projekcie jest **PyTorch**. Biblioteka ta została użyta zarówno do implementacji metod analitycznych, jak i do pracy z modelami generatywnymi oraz semantycznymi. PyTorch wykorzystano m.in. do:

- trenowania własnego modelu DCGAN,
- trenowania sieci CNN do klasyfikacji obrazów generowanych przez AI,

- transfer learningu modelu ConvNeXt,
- wczytywania i ewaluacji modeli CLIP,
- generowania obrazów przy użyciu modeli BigGAN i Stable Diffusion,
- realizacji obliczeń widmowych oraz operacji tensorowych.

Istotną zaletą PyTorch jest dynamiczny graf obliczeń, który ułatwia implementację niestandardowych procedur analitycznych oraz eksperymentalnych metod detekcji.

CatBoost i scikit-learn

Do realizacji mechanizmu klasyfikacji łączonej wykorzystano bibliotekę **CatBoost**, implementującą algorytmy gradient boosting na drzewach decyzyjnych. CatBoost został wybrany ze względu na dobrą stabilność przy pracy na heterogenicznych zbiorach cech oraz odporność na korelacje pomiędzy danymi wejściowymi.

Model ten pełni rolę meta–klasyfikatora, integrującego wyniki metod forensycznych oraz semantycznych w jedną końcową ocenę pochodzenia obrazu.

Dodatkowo wykorzystano bibliotekę **scikit-learn** do realizacji pomocniczych operacji uczenia maszynowego, walidacji modeli oraz przetwarzania wyników. Modele meta–klasyfikacyjne były serializowane i ładowane z użyciem biblioteki **joblib**.

3.2.3. Biblioteki do przetwarzania obrazu i sygnału

OpenCV

Biblioteka **OpenCV** stanowi podstawowe narzędzie wykorzystywane w niskopoziomowym przetwarzaniu obrazu. Została ona użyta m.in. do:

- generowania map błędu w metodzie ELA,
- realizacji filtrów wygładzających i operacji morfologicznych w analizie PRNU,
- konwersji przestrzeni barw oraz normalizacji obrazów wejściowych.

Zaletą OpenCV jest wysoka wydajność wynikająca z implementacji kluczowych operacji w języku C++.

NumPy, SciPy i PyWavelets

Biblioteki **NumPy** oraz **SciPy** zostały wykorzystane do realizacji obliczeń numerycznych i statystycznych, w szczególności:

- obliczeń macierzowych oraz transformacji FFT,
- analizy widma mocy obrazu,
- dopasowania rozkładów statystycznych wykorzystywanych przy kalibracji progów decyzyjnych.

Dodatkowo wykorzystano bibliotekę **PyWavelets** do analizy cech falowych obrazu, stosowanych w wybranych metodach detekcji artefaktów generatywnych.

Pillow

Biblioteka **Pillow** została użyta jako podstawowe narzędzie do wczytywania, zapisu oraz wstępnej normalizacji obrazów wejściowych w różnych formatach graficznych.

3.2.4. Modele semantyczne i analiza kontekstowa

W projekcie wykorzystano modele semantyczne oparte na architekturze CLIP. Do tego celu zastosowano biblioteki **open_clip_torch** oraz oficjalną implementację **OpenAI CLIP**. Modele te umożliwiały analizę spójności semantycznej obrazu oraz ekstrakcję wysokopoziomowych reprezentacji cech wizualnych.

Do pracy z nowoczesnymi architekturami sieci wizji komputerowej wykorzystano bibliotekę **timm**, zapewniającą dostęp do szerokiej gamy pretrenowanych modeli, w tym ConvNeXt.

3.2.5. Analiza metadanych i cech pomocniczych

W systemie zastosowano również narzędzia do analizy cech niezwiązańych bezpośrednio z pikselami obrazu. W szczególności wykorzystano:

- bibliotekę **face-recognition** do detekcji obecności twarzy, wykorzystywaną do adaptacyjnego uruchamiania wybranych metod detekcji,
- bibliotekę **c2pa-python** do analizy manifestów C2PA oraz identyfikacji deklarowanej proveniencji treści wizualnych.

3.2.6. Źródła danych i modele generatywne wykorzystywane w projekcie

Modele generatywne zostały wykorzystane w projekcie na potrzeby kalibracji metod detekcji oraz walidacji skuteczności systemu. W szczególności użyto:

- **DALL-E 2** (OpenAI),
- **BigGAN**,
- **Stable Diffusion 1.5**,
- **DCGAN**.

Do trenowania i testowania systemu wykorzystano m.in. następujące źródła danych:

- **140k Real and Fake Faces** (Kaggle),
- **Artifact Dataset** (Kaggle),
- **Dragon Dataset** (Hugging Face),
- **FFHQ (Flickr-Faces-HQ)** (Kaggle),
- **Camera Photos vs Ai generated Photos Classifier** (Kaggle)
- **Natural Images Dataset** (Kaggle),
- **Susy Dataset** (Hugging Face),
- **Defactify Image Dataset** (Hugging Face),
- **Pexels**.

3.2.7. Narzędzia pomocnicze, wdrożeniowe i testowe

Modele uczenia maszynowego oraz pliki wag były dystrybuowane z wykorzystaniem platformy **Hugging Face Hub** oraz biblioteki **huggingface_hub**, umożliwiającej automatyczne pobieranie i wersjonowanie modeli.

System został przygotowany do uruchamiania w środowisku kontenerowym z wykorzystaniem narzędzi **Docker** oraz **Docker Compose**, co zapewnia reprodukowalność środowiska uruchomieniowego oraz izolację zależności.

Do testowania poprawności działania interfejsów API oraz przetwarzania wsadowego wykorzystano bibliotekę **pytest**, wraz z mechanizmami testów integracyjnych frameworka FastAPI.

Do generowania wykresów, map oraz materiałów wizualnych wykorzystano bibliotekę **Matplotlib**. Narzędzia te posłużyły do wizualizacji widma FFT, map ELA i PRNU, a także do tworzenia wykresów statystycznych wykorzystywanych podczas analiz kalibracyjnych.

Dodatkowo, dla sieci CNN oraz ConvNeXt zastosowano technikę **Grad-CAM**, umożliwiającą wizualizację obszarów obrazu mających największy wpływ na decyzję klasyfikacyjną modelu.

Interfejs użytkownika systemu został zaimplementowany jako aplikacja webowa w technologii **React** z wykorzystaniem **TypeScript**, co ułatwia utrzymanie kodu oraz zapewnia bezpieczeństwo typów w warstwie prezentacji i komunikacji z API. Do budowy komponentów interfejsu wykorzystano bibliotekę komponentów UI (**shadcn/ui**), natomiast elementy ikonograficzne dostarczane są przez **lucide-react**.

W aplikacji frontendowej zastosowano również bibliotekę **fflate** do obsługi archiwów ZIP po stronie przeglądarki, co umożliwia przetwarzanie wyników analizy wsadowej (rozpakowanie plików CSV/JSON/PNG) oraz tworzenie paczek z wizualizacjami generowanymi przez metody detekcji. Komunikacja z backendem realizowana jest poprzez żądania HTTP z wykorzystaniem formatu **multipart/form-data** (upload obrazów i archiwów ZIP) oraz odpowiedzi w formacie **JSON** i **application/zip**.

3.2.8. Uzasadnienie wyboru stosu technologicznego

Zastosowany stos technologiczny został dobrany w taki sposób, aby zapewnić wysoką wydajność obliczeniową, elastyczność projektową oraz możliwość dalszego rozwoju systemu. Klu czowe kryteria wyboru obejmowały:

- możliwość wykorzystania akceleracji GPU,
- kompatybilność narzędzi w ramach jednego środowiska programistycznego,
- dostępność modeli pretrenowanych i sprawdzonych implementacji,
- otwartość i możliwość replikacji wyników,
- łatwość rozszerzania systemu o nowe metody analizy.

W efekcie powstał spójny i nowoczesny stos technologiczny, umożliwiający realizację wieloaspektowej analizy obrazów cyfrowych oraz skutecną detekcję treści generowanych przez sztuczną inteligencję.

3.3. Moduły analityczne systemu

System składa się z zestawu niezależnych modułów analitycznych, z których każdy realizuje określony rodzaj analizy obrazów. Moduły te obejmują zarówno klasyczne techniki forensyczne, jak i analizę metadanych oraz modele uczenia maszynowego i sieci neuronowe. Każdy moduł został zaimplementowany w formie komponentu z ujednoliconym interfejsem, co umożliwia ich łatwą integrację z aplikacją oraz pozwala użytkownikowi konfigurować zestaw metod używanych podczas analizy.

Dla modułów opartych na uczeniu maszynowym przedstawiono proces przygotowania danych treningowych, architekturę modelu oraz metody ewaluacji, zapewniając pełną przejrzystość decyzji analitycznych. Moduły klasyczne oraz służące do ekstrakcji danych (np. metadane EXIF) opisano w kontekście ich działania na obrazach i sposobu interpretacji wyników w systemie.

W skład systemu wchodzą następujące moduły analityczne:

1. **Moduł analizy metadanych EXIF** - odpowiedzialny za odczyt oraz prezentację metadanych zapisanych w pliku obrazu, takich jak informacje o aparacie, parametrach ekspozycji czy oprogramowaniu użytym do obróbki. Dane EXIF pełnią w systemie funkcję informacyjną i wspierającą interpretację wyników, lecz nie są bezpośrednio wykorzystywane w procesie klasyfikacji.
2. **Moduł weryfikacji C2PA** - realizujący sprawdzanie obecności oraz poprawności podpisów autentyczności zgodnych ze standardem C2PA. W przypadku wykrycia prawidłowego manifestu C2PA moduł dostarcza silnej przesłanki potwierdzającej pochodzenie obrazu, która może zostać wykorzystana jako filtr decyzyjny lub dodatkowy sygnał w procesie analizy.
3. **Moduł PRNU** - realizujący ekstrakcję sygnału szumu charakterystycznego dla fizycznych matryc światłoczułych oraz obliczanie miar dopasowania, wykorzystywanych do oceny prawdopodobieństwa pochodzenia obrazu z rzeczywistego urządzenia rejestrującego.
4. **Moduł ELA (Error Level Analysis)** - analizujący rozkład błędów ponownej kompresji JPEG w celu wykrycia anomalii strukturalnych typowych dla obrazów syntetycznych lub edytowanych.
5. **Moduł analizy widmowej (FFT)** - odpowiedzialny za badanie charakterystyki częstotliwościowej obrazu, w szczególności nachylenia widma mocy, które wykazuje istotne różnice pomiędzy obrazami naturalnymi a generowanymi przez modele sztucznej inteligencji.
6. **Model CLIP zeroshot** - służący do ekstrakcji semantycznych reprezentacji obrazu i analizy ich zgodności z cechami typowymi dla fotografii rzeczywistych, umożliwiając detekcję obrazów AI bez konieczności dodatkowego treningu.
7. **Model CLIP z klasyfikatorem SVM** - łączenie ekstrakcji cech przez CLIP z klasyfikacją SVM; SVM jest trenowany na reprezentacjach CLIP w celu rozróżnienia obrazów realnych i syntetycznych.
8. **Model CNN StyleGAN** - prosta sieć konwolucyjna wytrenowana na obrazach twarzy rzeczywistych oraz generowanych przez StyleGAN; model dokonuje binarnej klasyfikacji obrazów jako realne lub wygenerowane przez StyleGAN.

9. **Model ConvNeXt StyleGAN** - nowocześniejsza architektura ConvNeXt wytrenowana analogicznie do CNN StyleGAN, wykorzystująca lepsze warstwy konwolucyjne do bardziej precyzyjnego rozpoznawania obrazów realnych i generowanych przez StyleGAN.
10. **Model GAN vs Diffusion** - ConvNeXt wytrenowany do klasyfikacji obrazów AI na poziomie rodzaju generatora (GAN vs. modele dyfuzyjne); model nie rozróżnia real/AI, a wskazuje typ użytego generatora.
11. **Model ConvNeXt Diffusion** - ConvNeXt wytrenowany do klasyfikacji obrazów wygenerowanych przez różne rodziny modeli dyfuzyjnych. Umożliwia rozpoznanie konkretnej grupy modelu AI odpowiedzialnego za wygenerowanie obrazu.
12. **Model CLIP Diffusion** - model oparty na architekturze CLIP, wykorzystujący semantyczne reprezentacje obrazu do przypisania potencjalnego generatora obrazu spośród nowoczesnych modeli dyfuzyjnych, w tym DALL-E i Midjourney.
13. **Moduł integracji wyników** - odpowiedzialny za agregację wyników uzyskanych z wybranych metod analizy i wyznaczenie końcowej oceny pochodzenia obrazu.

3.3.1. Moduł ekstrakcji metadanych EXIF

Jednym z komponentów systemu jest moduł ekstrakcji metadanych EXIF, którego zadaniem jest odczyt wybranych informacji opisowych zapisanych w nagłówku pliku graficznego. Moduł ten pełni funkcję pomocniczą i nie bierze bezpośredniego udziału w procesie klasyfikacji obrazu jako wygenerowanego przez AI lub pochodzącego z rzeczywistego źródła.

Moduł działa na obrazie wczytanym do pamięci operacyjnej i realizuje odczyt metadanych EXIF z wykorzystaniem standardowych mechanizmów udostępnianych przez bibliotekę Pillow [32]. W trakcie przetwarzania analizowany jest zbiór znaczników EXIF, z którego wybierany jest ograniczony podzbiór pól o charakterze opisowym, zgodnych ze specyfikacją formatu [22].

W aktualnej implementacji ekstraktowane są wyłącznie następujące pola:

- Make — producent urządzenia rejestrującego,
- Model — model urządzenia,
- Software — oprogramowanie użyte do zapisu lub przetworzenia obrazu,
- DateTime — data i czas utworzenia lub zapisu pliku,
- Artist — autor obrazu,
- Copyright — informacje o prawach autorskich.

Wynikiem działania modułu jest słownik zawierający wyłącznie wartości tekstowe odpowiadające wykrytym znacznikom EXIF. W przypadku braku danych EXIF, braku wskazanych pól lub wystąpienia błędu odczytu, moduł zwraca pustą strukturę danych.

Informacje pozyskane z metadanych EXIF nie są wykorzystywane do automatycznej klasyfikacji obrazu. Dane te są dołączane do raportu końcowego systemu w celach informacyjnych oraz diagnostycznych i mogą stanowić wsparcie dla analizy eksperckiej, w szczególności przy interpretacji wyników metod opartych na analizie treści obrazu.

3.3.2. Moduł weryfikacji C2PA

Jednym z komponentów systemu jest moduł weryfikacji C2PA (Content Credentials), którego zadaniem jest analiza metadanych pochodzenia obrazu zapisanych w postaci kryptograficznie podpisanego manifestu. Standard C2PA umożliwia dołączanie do plików graficznych informacji opisujących proces ich powstania oraz historię edycji, w tym użycie narzędzi opartych na sztucznej inteligencji [9].

Moduł C2PA nie analizuje zawartości pikselowej obrazu, lecz opiera się wyłącznie na danych opisowych zapisanych w manifeście dołączonym do pliku. W implementacji systemu manifest odczytywany jest bezpośrednio z pliku obrazu, dlatego metoda wymaga dostępu do oryginalnej ścieżki pliku. W przypadku jej braku analiza treści manifestu nie jest możliwa.

Proces działania modułu obejmuje następujące etapy:

1. próbę odczytu manifestu C2PA z pliku obrazu oraz pobranie informacji o stanie walidacji kryptograficznej podpisu,
2. parsowanie manifestu do postaci struktury danych,
3. analizę wszystkich asercji zapisanych w manifeście w celu wykrycia znanych wskaźników generatywnego pochodzenia obrazu lub jego edycji z użyciem narzędzi AI.

W ramach analizy sprawdzane są w szczególności wartości pola *digitalSourceType* występujące zarówno w treści asercji, jak i w zarejestrowanych działaniach (*c2pa.actions*). Za istotne uznawane są przypadki wskazujące na treści wytworzone algorytmicznie (*trainedAlgorithmicMedia*) oraz kompozyty zawierające elementy wygenerowane przez AI (*compositeWithTrainedAlgorithmicMedia*). Dodatkowo uwzględniane są wybrane wzorce spotykane w manifestach generowanych przez popularne narzędzia, w tym charakterystyczne kombinacje pól opisowych występujące w manifestach obrazów generowanych przez systemy z rodziną DALL-E.

Na podstawie zawartości manifestu moduł przypisuje obraz do jednej z klas jakościowych:

- **AI-origin** — obraz wygenerowany przez model sztucznej inteligencji,
- **AI-edited** — obraz edytowany z wykorzystaniem narzędzi AI,
- **unknown** — brak jednoznacznych informacji o użyciu AI,
- **unreadable-manifest** — brak możliwości poprawnego odczytu manifestu.

Wynikiem działania modułu jest obiekt typu *MethodResult*, zawierający wartość decyzyjną *score* oraz zestaw metryk opisowych. Metoda nie generuje wizualizacji, ponieważ analiza nie dotyczy treści obrazu.

Pole *score* pełni w systemie rolę sygnału logicznego. Jeżeli analiza manifestu wskazuje na generatywne pochodzenie obrazu lub jego edycję z użyciem narzędzi AI, a jednocześnie podpis manifestu został poprawnie zwalidowany kryptograficznie, metoda zwraca wartość *score = 99.9*. Taki wynik jest interpretowany jako silny i jednoznaczny sygnał pochodzenia obrazu z AI i powoduje natychmiastowe zakończenie dalszej analizy.

W każdej innej sytuacji, w szczególności w przypadku braku manifestu, błędu jego odczytu, niepoprawnej walidacji podpisu (również wtedy, gdy manifest zawiera wskaźniki użycia AI) lub braku znanych wskaźników użycia AI, metoda zwraca *score = null*. Jeżeli moduł C2PA jest jedynym aktywnym komponentem analizy, wynikowi przypisywana jest etykieta *UNKNOWN*. W przeciwnym przypadku rezultat metody nie wpływa na dalszy przebieg analizy, a decyzja końcowa pozostawiana jest pozostałym komponentom systemu.

W polu `metrics` zwracane są dodatkowe informacje diagnostyczne, w tym etykieta klasyfikacji jakościowej, opis uzasadniający wynik analizy oraz informacje dotyczące stanu walidacji manifestu. Dane te są wykorzystywane wyłącznie do celów raportowych i diagnostycznych.

3.3.3. Moduł weryfikacji PRNU

Jednym z komponentów systemu jest moduł weryfikacji PRNU (*Photo-Response Non-Uniformity*), którego zadaniem jest analiza charakterystycznego, sprzętowo uwarunkowanego szumu sensora obrazu w celu oceny, czy badany obraz pochodzi z rzeczywistego urządzenia rejestrującego. Zjawisko PRNU wynika z mikroskopowych, nieusuwalnych niejednorodności czułości poszczególnych pikseli matrycy światłoczułej, które prowadzą do powstania powtarzalnego wzorca szumowego obecnego w każdym obrazie zarejestrowanym przez dane urządzenie.

W modelu obserwacyjnym PRNU traktowane jest jako słaby, multiplikatywny składnik sygnału „wmieszany” w obraz użyteczny. Odpowiednia ekstrakcja reszty szumowej oraz jej porównanie z fingerprintem referencyjnym pozwalają na wnioskowanie o pochodzeniu obrazu na poziomie konkretnego urządzenia lub modelu kamery. W przeciwnieństwie do metod ukierunkowanych na wykrywanie artefaktów generacji syntetycznej, analiza PRNU służy do *potwierdzania* fizycznego pochodzenia obrazu. Z tego względu moduł ten pełni w systemie rolę pomocniczą: pozytywne dopasowanie PRNU stanowi silny sygnał wspierający autentyczność obrazu, natomiast brak dopasowania nie jest traktowany jako dowód syntetycznego pochodzenia.

Moduł działa w oparciu o porównanie obrazu wejściowego z zestawem referencyjnych fingerprints sensorów zapisanych lokalnie w postaci plików .npz. Każdy fingerprint zawiera wzorzec PRNU \mathbf{K} , odpowiadającą mu maskę jakości \mathbf{M}_{ref} oraz opcjonalne metadane `native_hw`, opisujące natywną rozdzielcość obrazu, dla której fingerprint został wyznaczony.

W implementacji przyjęto klasyczny schemat ekstrakcji reszty szumowej i dopasowania, a dobór rozwiązań praktycznych wspierających stabilność sygnału PRNU odniesiono do podejścia przedstawionego w [24].

Model szumu i estymacja fingerprintów referencyjnych

Przyjmowany jest klasyczny model obrazu [25]:

$$\mathbf{I} = \mathbf{I}_0 \odot (1 + \mathbf{K}) + \Theta, \quad (3.1)$$

gdzie \mathbf{I}_0 oznacza idealny obraz bezszumowy, \mathbf{K} — komponent PRNU, a Θ pozostałe źródła zakłóceń (szum odczytu, kwantyzacja, kompresja). Po zastosowaniu procedury odszumiania $\mathcal{D}(\cdot)$ uzyskuje się resztę szumową:

$$\mathbf{W} = \mathcal{D}(\mathbf{I}) = \mathbf{I} - \hat{\mathbf{I}}, \quad (3.2)$$

która zawiera komponent PRNU oraz zakłócenia losowe.

Dla fingerprintu referencyjnego estymacja wykonywana jest na zbiorze N obrazów zarejestrowanych przez to samo urządzenie. Maksymalno-wiarygodnościowy estymator PRNU [25] przyjmuje postać:

$$\hat{\mathbf{K}} = \frac{\sum_{i=1}^N \mathbf{W}_i \odot \mathbf{I}_i}{\sum_{i=1}^N \mathbf{I}_i \odot \mathbf{I}_i}, \quad (3.3)$$

gdzie \mathbf{W}_i i \mathbf{I}_i oznaczają odpowiednio resztę szumową i obraz wejściowy dla i -tej próbki. W praktycznej implementacji stosowana jest wersja ważona estymatora, w której piksele nasycone oraz o niskiej wiarygodności są eliminowane przy użyciu maski jakości.

Budowa fingerprintów w projekcie. Fingerprinty referencyjne przygotowano offline na bazie zbiorów *Dresden Dataset* [12], *Imagine* [20] oraz *FloraView* [14]. Proces ich budowy obejmował:

1. ekstrakcję reszty szumowej metodą falkową (fala db8, poziom dekompozycji $L = 4$, programowanie *soft*),
2. estymację fingerprintu PRNU z wielu obrazów zgodnie z równaniem (3),
3. bazowe wzmacnienie fingerprintu metodą *SEA* (*Spectral Equalization Algorithm*),
4. centralne przycięcie fingerprintu do rozmiaru 1024×1024 .

Wszystkie fingerprinty są zapisane w postaci kanonicznej 1024×1024 . **Na etapie weryfikacji nie wykonuje się żadnego dodatkowego przeskalowania fingerprintów.**

Przyjęcie kanonicznego rozmiaru 1024×1024 stanowi decyzję projektową wynikającą z kompromisu pomiędzy stabilnością statystyczną sygnału PRNU, kosztem obliczeniowym oraz ujednoliceniem geometrii fingerprintów w całej bazie referencyjnej.

Wzmocnienie fingerprintu metodą SEA. Metoda SEA polega na wyrównaniu widma amplitudowego fingerprintu w dziedzinie częstotliwości [16]. Dla fingerprintu \mathbf{K} wyznaczana jest transformata Fouriera:

$$\mathbf{F} = \mathcal{F}(\mathbf{K}),$$

a następnie dla każdej częstotliwości radialnej r obliczana jest średnia amplituda $A(r)$. Widmo fingerprintu normalizowane jest przez tę charakterystykę:

$$\mathbf{F}_{\text{eq}}(u, v) = \frac{\mathbf{F}(u, v)}{A(r(u, v)) + \epsilon},$$

po czym wykonywana jest odwrotna transformacja Fouriera. Operacja ta redukuje dominujące składowe niskoczęstotliwościowe i wzmacnia względny udział drobnoskalowych struktur charakterystycznych dla PRNU. Po wzmacnieniu fingerprint jest centrowany i normalizowany do jednostkowego odchylenia standardowego.

Ekstrakcja reszty szumowej dla obrazu testowego

Dla badanego obrazu wyznaczana jest reszta szumowa pełniąca rolę zapytania (*query fingerprint*). Procedura ta jest spójna z procesem budowy fingerprintów referencyjnych i obejmuje następujące kroki:

1. **Konwersja do luminancji.** Obraz wejściowy konwertowany jest do luminancji:

$$\mathbf{I} = 0.299R + 0.587G + 0.114B,$$

a następnie normalizowany do zakresu $[0, 1]$.

2. **Dopasowanie geometrii.** Dla fingerprintu o rozmiarze $S \times S$ (w praktyce $S = 1024$):

- jeżeli dostępne są metadane *native_hw*(H, W), obraz testowy przeskalowywany jest do tej rozdzielczości (interpolacja liniowa),
- następnie wykonywany jest centralny crop $S \times S$.

3. **Maska nasycenia.** Definiowana jest maska:

$$\mathbf{M}_{\text{test}}(x, y) = \begin{cases} 1 & 0.05 < \mathbf{I}(x, y) < 0.95, \\ 0 & \text{w przeciwnym razie.} \end{cases}$$

Przyjęte progi odcinają piksele silnie niedoświetlone oraz nasycone, dla których sygnał PRNU charakteryzuje się niskim stosunkiem sygnału do szumu lub ulega zniekształceniom wynikającym z klipowania. Wartości progowe dobrano empirycznie jako kompromis pomiędzy eliminacją niewiarygodnych pikseli a zachowaniem wystarczającego pokrycia maską.

4. **Lokalne wyzerowanie średniej.**

$$\mathbf{I}_{\text{LZM}} = \mathbf{I} - \text{Blur}_{3 \times 3}(\mathbf{I}).$$

Zastosowanie niewielkiego filtra rozmywającego o rozmiarze 3×3 ma na celu usunięcie wolnozmiennych składowych obrazu (np. oświetlenia i struktury sceny) przy minimalnym wpływie na drobnoskalowe struktury charakterystyczne dla PRNU. Rozmiar filtru dobrano jako minimalny zapewniający stabilność operacji lokalnego wyzerowania średniej.

5. **Ekstrakcja reszty szumowej metodą falkową.** Dla \mathbf{I}_{LZM} wykonywana jest dwuwymiarowa dekompozycja falkowa (fala db8, $L = 4$). Parametr szumu estymowany jest estymatorem MAD z diagonalnych współczynników najwyższego poziomu:

$$\sigma = \frac{\text{median}(|\mathbf{cD}_L - \text{median}(\mathbf{cD}_L)|)}{0.6745}.$$

Następnie stosowane jest progowanie *soft* z progiem:

$$T = \sigma \sqrt{2 \log N},$$

gdzie N oznacza liczbę pikseli obrazu. Po rekonstrukcji obrazu odszumionego $\hat{\mathbf{I}}$ reszta szumowa wyznaczana jest jako:

$$\mathbf{W} = \mathbf{I}_{\text{LZM}} - \hat{\mathbf{I}}.$$

Parametry dekompozycji falkowej (fala db8, poziom $L = 4$, progowanie *soft*) zostały przyjęte zgodnie z praktyką stosowaną w analizie PRNU oraz w celu zachowania spójności z procesem budowy fingerprintów referencyjnych. Ustawienia te zapewniają dobrą separację składników szumowych przy zachowaniu stabilności estymacji.

Zastosowany estymator MAD stanowi odporną miarę wariancji szumu, a współczynnik 0.6745 odpowiada normalizacji dla rozkładu normalnego, zgodnie z klasycznym ujęciem estymacji szumu w dziedzinie falkowej.

Porównanie z fingerprintami referencyjnymi

Dla fingerprintu referencyjnego \mathbf{K} wyznaczana jest wspólna maska:

$$\mathbf{M} = \mathbf{M}_{\text{ref}} \odot \mathbf{M}_{\text{test}},$$

a następnie sygnały:

$$\mathbf{X} = \mathbf{W} \odot \mathbf{M}, \quad \mathbf{Y} = (\mathbf{I} \odot \mathbf{K}) \odot \mathbf{M}.$$

Po centrowaniu obu sygnałów obliczana jest znormalizowana korelacja krzyżowa (NCC) w dziedzinie częstotliwości:

$$\mathbf{C} = \frac{\mathcal{F}^{-1}(\mathcal{F}(\mathbf{X}) \cdot \overline{\mathcal{F}(\mathbf{Y})})}{\|\mathbf{X}\|_2 \|\mathbf{Y}\|_2 + \epsilon}.$$

Miara dopasowania oparta jest na statystyce PCE (*Peak-to-Correlation Energy*) [17]. Dla maksimum C_{\max} mapy \mathbf{C} oraz sąsiedztwa 11×11 wokół piku:

$$\text{PCE} = \text{sign}(C_{\max}) \cdot \frac{C_{\max}^2}{\frac{1}{|\Omega \setminus \mathcal{N}|} \sum_{(r,c) \in \Omega \setminus \mathcal{N}} \mathbf{C}(r,c)^2}.$$

Rozmiar sąsiedztwa piku 11×11 przyjęto zgodnie z typową praktyką w analizie PCE, umożliwiającą skuteczne wykluczenie lokalnego maksimum korelacji przy estymacji energii tła, bez nadmiernego ograniczania obszaru odniesienia.

Dodatkowo stosowane są warunki jakościowe:

- $\text{mean}(\mathbf{M}) \geq 0.01$,
- $\text{std}(\mathbf{X}) \geq 0.001$.

Warunki jakościowe wprowadzono w celu eliminacji przypadków, w których obliczenia korelacyjne byłyby niestabilne lub pozbawione wartości diagnostycznej, np. przy bardzo małym pokryciu maską lub znikomej energii reszty szumowej. Wartości progowe dobrano empirycznie jako proste heurystyki zapewniające stabilność numeryczną obliczeń.

Reguła decyzji i kalibracja progu

Dla obrazu testowego wyznaczana jest wartość:

$$\text{PCE}_{\max} = \max_j \text{PCE}(\mathbf{K}_j, \mathbf{W}),$$

czyli maksymalna wartość statystyki PCE uzyskana spośród wszystkich fingerprintów referencyjnych \mathbf{K}_j . Na podstawie PCE_{\max} stosowana jest reguła progowa:

$$\text{PCE}_{\max} \geq \tau \Rightarrow \text{dopasowanie pozytywne}.$$

Motywacja i metoda doboru progu. W praktyce największym ryzykiem dla modułu PRNU w kontekście weryfikacji autentyczności jest *fałszywe dopasowanie* dla obrazów syntetycznych (AI), tj. sytuacja, w której obraz niepochodzący z kamery uzyskuje wysoką wartość statystyki PCE względem któregoś z fingerprintów referencyjnych. Z tego względu próg decyzyjny τ został dobrany w sposób konserwatywny, tak aby odciąć cały zaobserwowany ogon rozkładu PCE_{\max} dla obrazów syntetycznych.

Kalibracja progu polegała na analizie maksymalnych wartości statystyki PCE_{\max} uzyskiwanych dla obrazów syntetycznych oraz rzeczywistych. Dla każdego obrazu rozpatrywano najlepsze dopasowanie PRNU (TOP-1), tj. największą wartość PCE spośród wszystkich fingerprintów referencyjnych. Następnie wyznaczono największą wartość PCE_{\max} zaobserwowaną w zbiorze obrazów AI:

$$\max_{\text{AI}} = \max_{i \in \mathcal{D}_{\text{AI}}} \text{PCE}_{\max}^{(i)}.$$

Próg decyzyjny ustalono jako

$$\tau = \max_{\text{AI}} +m, \quad (3.4)$$

gdzie m jest niewielkim, dodatnim marginesem bezpieczeństwa wprowadzonym jako decyzja projektowa, nie wynikającą bezpośrednio z formalnego modelu statystycznego ani z jednoznacznej optymalizacji na danych empirycznych. Jego rola polega na konserwatywnym odsumieniu progu decyzyjnego ponad maksymalną wartość PCE_{\max} zaobserwowaną dla obrazów syntetycznych w procesie kalibracji, z uwzględnieniem możliwej niepewności estymacji ekstremum na skończonym zbiorze próbek. W niniejszej implementacji przyjęto wartość $m = 1$, stanowiącą niewielki bufor bezpieczeństwa, zgodny z pomocniczą i potwierdzającą rolą modułu PRNU w systemie.

Należy podkreślić, że niewielkie zmiany wartości parametru m nie mają istotnego wpływu na działanie całego systemu, ponieważ brak przekroczenia progu nie skutkuje decyzją negatywną, lecz jedynie brakiem sygnału potwierdzającego z modułu PRNU.

Ustawienia eksperymentu kalibracyjnego. Eksperyment kalibracyjny przeprowadzono na 1000 losowo wybranych obrazach z podzbioru val zbioru *SuSy* [19]. Dla każdego obrazu obliczono wartość PCE_{\max} względem dostępnych fingerprintów referencyjnych. Zastosowany margines bezpieczeństwa wynosił $m = 1$.

W rezultacie próg decyzyjny przyjął postać:

$$\tau = \max_{\text{AI}} +1,$$

a jego wartość została wpisana do implementacji modułu PRNU jako stała `PCE_THRESHOLD`.

Interpretacja wyniku. Jeżeli $PCE_{\max} \geq \tau$, dopasowanie do kamery uznawane jest za pozytywne, co interpretowane jest jako **silny i jednoznaczny sygnał fizycznego pochodzenia obrazu** (zarejestrowanego przez rzeczywiste urządzenie). W takim przypadku moduł PRNU zwraca wartość `score = 0.1`. Należy podkreślić, że wartość ta *nie* stanowi prawdopodobieństwa, lecz jest technicznym kodem sygnału pozytywnego, używanym w celu ujednolicenia interfejsu wyników pomiędzy różnymi metodami analizy.

Jeżeli $PCE_{\max} < \tau$, moduł PRNU *nie wydaje rozstrzygnięcia*. W takiej sytuacji zwracana jest wartość `score = null`, oznaczająca brak wystarczających podstawa do potwierdzenia pochodzenia obrazu na podstawie sygnału PRNU. Brak dopasowania PRNU nie jest interpretowany jako dowód syntetycznego pochodzenia obrazu, lecz wyłącznie jako brak informacji diagnostycznej z tego konkretnego źródła.

W przypadku, gdy moduł PRNU jest *jedynym aktywnym komponentem* analizy, `score = null` skutkuje przypisaniem obrazu do klasy `UNKNOWN`. Należy podkreślić, że etykieta ta nie jest wynikiem działania samej metody PRNU, lecz decyzją systemu nadzawanego w sytuacji braku sygnałów decyzyjnych. Jeżeli natomiast w analizie uczestniczą inne moduły (np. detektory artefaktów generacji syntetycznej), wynik modułu PRNU nie wpływa na dalszy przebieg analizy ani na decyzję końcową, która podejmowana jest na podstawie sygnałów dostarczonych przez pozostałe komponenty systemu.

Metryki i wizualizacja

Oprócz wartości `score`, moduł zwraca zestaw metryk diagnostycznych obejmujących m.in. wartość PCE , położenie piku korelacji, energię tła, pokrycie masek oraz statystyki reszty szu-

mowej. Dodatkowo generowana jest wizualizacja mapy korelacji (po przesunięciu fftshift i logarytmicznej kompresji dynamiki), ułatwiająca ekspercką ocenę jakości dopasowania.

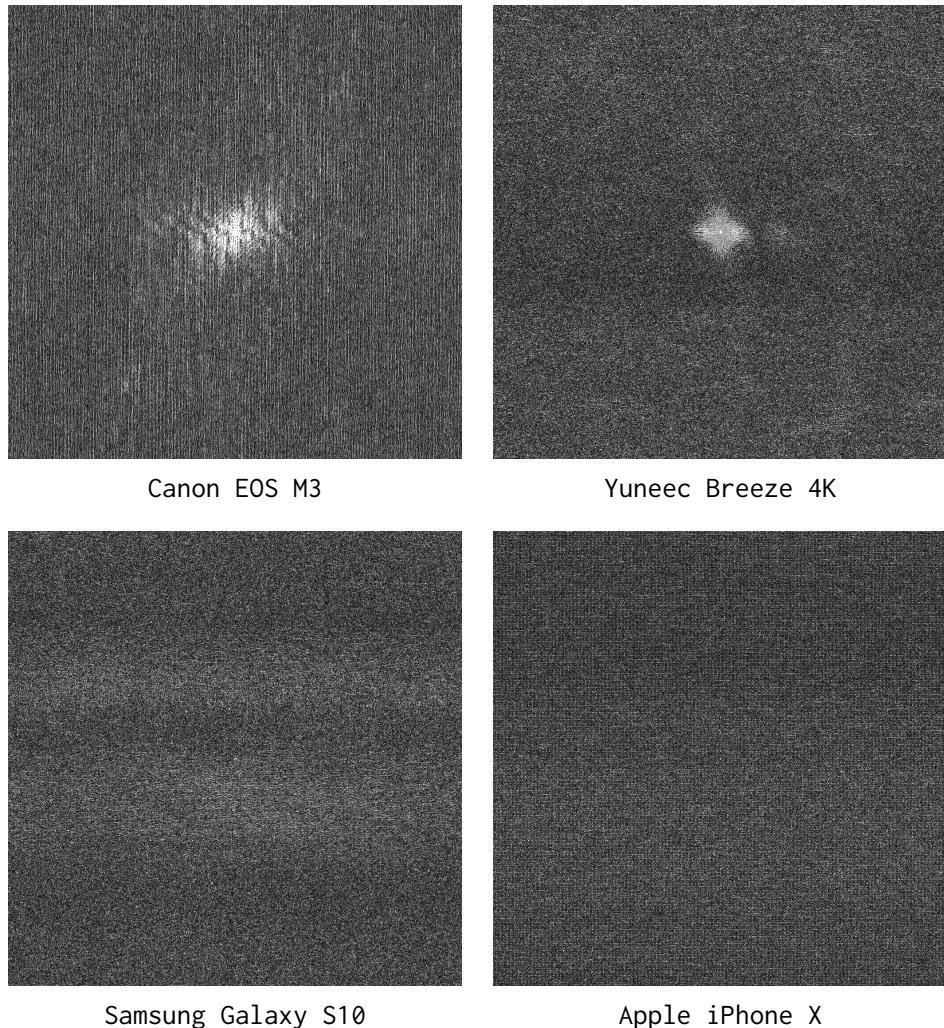
Zakres i różnorodność bazy fingerprints. W systemie wykorzystano lokalną bazę fingerprints PRNU zapisaną w postaci plików .npz (katalog `fingerprints_1024`). Baza ta obejmuje łącznie **139 fingerprintów** wygenerowanych dla różnych urządzeń rejestrujących, w tym:

- aparatów fotograficznych klasy DSLR i mirrorless (np. Canon EOS, Nikon Z/D, Sony A),
- aparatów kompaktowych oraz starszych modeli konsumenckich (np. Canon Ixus/PowerShot, Nikon CoolPix, Olympus mju, Pentax Optio, Ricoh GX, Casio EX, Fujifilm FinePix),
- urządzeń mobilnych (np. Apple iPhone/iPad, Samsung Galaxy, Google Pixel, Huawei, Xiaomi, Motorola, LG, OnePlus, Nokia),
- kamer pokładowych oraz urządzeń rejestrujących obraz wideo (np. DJI Spark, Yuneec Breeze).

Dzięki takiej różnorodności baza obejmuje szerokie spektrum sensorów oraz typowych łańcuchów przetwarzania obrazu, w tym również silnie „mobilne” pipeline’y ISP, co zwiększa praktyczną użyteczność modułu PRNU w warunkach rzeczywistych.

Interpretacja map korelacji. Na potrzeby diagnostyki metoda generuje mapę korelacji **C** pomiędzy resztą szumową obrazu testowego a fingerprintem referencyjnym (po uwzględnieniu masek jakości). Wizualizacja mapy korelacji pozwala jakościowo ocenić strukturę odpowiedzi dopasowania: w szczególności umożliwia sprawdzenie, czy odpowiedź jest zdominowana przez pojedyncze lokalne maksimum, czy też ma bardziej rozproszony charakter. Ponieważ rozkład wartości **C** zależy od wielu czynników (m.in. zawartości sceny, kompresji, odszumiania, transformacji geometrycznych oraz pokrycia maską), mapa korelacji traktowana jest w systemie jako artefakt diagnostyczny wspierający interpretację metryk liczbowych (*w szczególności* PCE), a nie jako samodzielne kryterium decyzyjne.

Przykładowe mapy korelacji dla wybranych modeli Rysunek 3.1 przedstawia przykładowe mapy korelacji dla czterech wybranych modeli z bazy fingerprintów.



Rysunek 3.1: Przykładowe mapy korelacji PRNU dla czterech modeli z bazy fingerprintów.

Uwagi praktyczne. Z uwagi na różnice w natywnych rozdzielczościach oraz pipeline'ach ISP, wartości PCE mogą różnić się pomiędzy urządzeniami nawet przy poprawnym dopasowaniu. W praktyce największą stabilność dopasowania obserwuje się dla przypadków, w których geometria obrazu testowego jest zgodna z native_hw fingerprintu oraz gdy obraz nie został poddany agresywnemu odszumianiu lub silnej kompresji stratnej. Z tego powodu mapa korelacji oraz metryki takie jak pokrycie maski i odchylenie standardowe reszty szumowej są wykorzystywane jako artefakty diagnostyczne wspierające interpretację wyniku.

3.3.4. Metoda ELA z klasyfikatorem SVC (ELA, Error Level Analysis)

Kolejnym komponentem systemu jest metoda oparta na analizie poziomów błędu rekodowania JPEG (*Error Level Analysis – ELA*), należąca do klasycznych technik informatyki śledczej obrazu. Celem metody jest identyfikacja nienaturalnych wzorców różnic pomiędzy obrazem wejściowym a jego wersją ponownie zapisaną w formacie JPEG, które mogą wskazywać na modyfikacje obrazu lub jego pochodzenie syntetyczne.

Intuicja stojąca za ELA polega na obserwacji, że fragmenty obrazu pochodzące z różnych źródeł lub poddane odmiennym procesom przetwarzania mogą wykazywać różną „odporność” na ponowną kompresję JPEG. Różnice te ujawniają się jako lokalne wzrosty amplitudy błędu po

rekompresji, co było szeroko opisywane w literaturze dotyczącej analizy autentyczności obrazów [30].

W implementacji metoda ELA realizowana jest jako klasyfikator hybrydowy. W pierwszym etapie konstruowana jest mapa ELA, a następnie wynikowa reprezentacja przekształcana jest do wektora cech i klasyfikowana przez model SVC (*Support Vector Classifier*). Model klasyfikatora oraz parametry przetwarzania wstępnego wczytywane są z plików zasobów dołączonych do projektu: `ela_svc_model.joblib` (model) oraz `preprocess.json` (parametry).

Definicja mapy ELA

Niech $\mathbf{I} \in \{0, \dots, 255\}^{H \times W \times 3}$ oznacza obraz RGB. W pierwszym kroku wykonywana jest standaryzacja formatu obrazu (konwersja do RGB z uwzględnieniem orientacji EXIF), a następnie ponowna kompresja JPEG o ustalonym parametrze jakości Q :

$$\mathbf{I}_Q = \mathcal{J}_Q(\mathbf{I}), \quad (3.5)$$

gdzie $\mathcal{J}_Q(\cdot)$ oznacza operator zapisu i ponownego odczytu JPEG.

Mapa ELA definiowana jest jako bezwzględna różnica pomiędzy obrazem wejściowym a jego rekompresją:

$$\mathbf{D} = |\mathbf{I} - \mathbf{I}_Q|. \quad (3.6)$$

W implementacji różnica jest dodatkowo wzmacniana stałą $A > 0$ oraz obcinana do zakresu $[0, 255]$:

$$\mathbf{D}_A = \text{clip}(A \cdot \mathbf{D}, 0, 255). \quad (3.7)$$

Parametry Q oraz A przechowywane są w pliku konfiguracyjnym `preprocess.json` i odpowiadają za czułość metody na subtelne artefakty kompresji.

Ekstrakcja cech

Aby uzyskać wektor cech o stałym wymiarze, mapa ELA jest skalowana do rozmiaru $S \times S$ (z wykorzystaniem interpolacji biliniowej), a następnie normalizowana do zakresu $[0, 1]$:

$$\tilde{\mathbf{D}} = \frac{1}{255} \text{resize}(\mathbf{D}_A, S, S). \quad (3.8)$$

Ostateczny wektor cech powstaje przez spłaszczenie tensora:

$$\mathbf{x} = \text{vec}(\tilde{\mathbf{D}}) \in \mathbb{R}^{3S^2}. \quad (3.9)$$

Dodatkowo, na potrzeby metryk diagnostycznych mapa ELA przekształcana jest do skali szarości (luminancji) zgodnie z klasyczną kombinacją kanałów RGB. Pozwala to na obliczenie globalnych statystyk opisujących rozkład energii błędu rekodowania niezależnie od informacji barwnej.

Klasyfikacja SVC i mapowanie na score

Dla wyekstrahowanego wektora cech \mathbf{x} stosowany jest klasyfikator SVC z jądrem RBF. Funkcja decyzyjna modelu przyjmuje postać:

$$g(\mathbf{x}) = \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \quad K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \|\mathbf{u} - \mathbf{v}\|_2^2). \quad (3.10)$$

W procesie uczenia zastosowano potok StandardScaler \rightarrow SVC, co odpowiada standaryzacji cech:

$$\mathbf{z} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\sigma}. \quad (3.11)$$

Ponieważ klasyfikator SVC zwraca wartość funkcji decyzyjnej $g(\mathbf{x})$ (margines), w implementacji jest ona mapowana do przedziału $[0, 1]$ przy użyciu funkcji sigmoidalnej:

$$\text{score} = \sigma(g(\mathbf{x})) = \frac{1}{1 + \exp(-g(\mathbf{x}))}. \quad (3.12)$$

Tak zdefiniowany score stanowi gładką miarę „AI-owości” w spójnym interfejsie systemu, jednak nie jest formalnie skalibrowanym prawdopodobieństwem.

Uczenie modelu ELA+SVC w projekcie

Model ELA+SVC wytrenowano na zbiorze 2000 obrazów typu *landscape*, obejmującym:

- 1000 obrazów wygenerowanych przez samodzielnie wytrenowany model DCGAN (klasa AI),
- 1000 fotografii krajobrazowych pobranych z serwisu Pexels (klasa REAL).

Dla obu klas wyznaczano mapy ELA, a następnie uczono binarny klasyfikator SVC na wektorach cech $\mathbf{x} = \text{vec}(\tilde{\mathbf{D}})$. W pliku `preprocess.json` zapisane są parametry przetwarzania (`img_size`, `ela_quality`, `ela_amplify`), natomiast w pliku `ela_svc_model.joblib` utrwalony jest uczony potok przetwarzania.

Zwarcane wartości i wizualizacja diagnostyczna

Metoda ELA zwraca obiekt typu `MethodResult`, zawierający:

- **name**: identyfikator metody (`ela`),
- **score**: wartość $\sigma(g(\mathbf{x})) \in [0, 1]$,
- **metrics**: m.in. wartość funkcji decyzyjnej `decision_score = g(x)` oraz dodatkowe statystyki mapy ELA (np. średnia, odchylenie standardowe, percentile i miary koncentracji energii),
- **visuals_b64**: mapę ELA jako obraz PNG zakodowany w Base64, dostępną pod kluczem `compression_artifacts_map`.

Wizualizacja mapy ELA pełni funkcję diagnostyczną i interpretacyjną. Obszary o podwyższonym błędzie rekompresji mogą wskazywać na lokalne nieregularności struktury obrazu, zgodnie z intuicją podejścia ELA opisaną w literaturze [30].

Trening modelu DCGAN do generowania obrazów treningowych dla metody ELA

Do wygenerowania kontrolowanych próbek syntetycznych wykorzystywanych w treningu metody ELA wytrenowano generatywną sieć przeciwną (GAN), opartą na architekturze Deep Convolutional GAN (DCGAN), zgodnej z oryginalną koncepcją przedstawioną w pracy Radford i in. [33]. Model ten został wybrany jako relatywnie prosty, a jednocześnie stabilny generator obrazów w rozdzielczości 256×256 , umożliwiający wytwarzanie realistycznych obrazów scen naturalnych.

Do treningu modelu wykorzystano zbiór obrazów LHQ (*Large-scale High-Quality Image Dataset*), udostępniony w ramach projektu ALIS [38]. Zbiór ten zawiera wysokiej jakości obrazy naturalnych krajobrazów, charakteryzujące się dużą różnorodnością struktur, teksturow oraz warunków oświetleniowych, co czyni go odpowiednim źródłem danych rzeczywistych do treningu modeli generatywnych.

Architektura DCGAN

Architektura generatora oraz dyskryminatora oparta jest na klasycznej strukturze DCGAN [33], wykorzystującej sekwencje warstw konwolucyjnych (transponowanych w generatorze) oraz normalizację partii. Generator G_θ odwzorowuje wektor losowy $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ do obrazu RGB o rozdzielczości 256×256 , natomiast dyskryminator D_ϕ mapuje obraz wejściowy do jednowymiarowej odpowiedzi skalarnej, interpretowanej jako miara realistyczności próbki.

W stosunku do oryginalnej architektury wprowadzono drobne modyfikacje wynikające z nowoczesnych praktyk treningowych, w szczególności zastosowanie normalizacji spektralnej w warstwach dyskryminatora w celu poprawy stabilności uczenia.

Zmodyfikowana funkcja straty: hinge loss

Chociaż w oryginalnej pracy DCGAN [33] zaproponowano użycie binarnej krzyżowej entropii jako funkcji kosztu, w niniejszej implementacji zastosowano zmodyfikowaną funkcję straty typu *hinge loss*, powszechnie wykorzystywaną we współczesnych wariantach GAN-ów ze względu na lepszą stabilność procesu uczenia oraz wyższą jakość generowanych obrazów [26].

Funkcja straty dla dyskryminatora ma postać:

$$\mathcal{L}_D = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\max(0, 1 - D_\phi(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\max(0, 1 + D_\phi(G_\theta(\mathbf{z})))], \quad (3.13)$$

natomiast funkcja straty generatora:

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [D_\phi(G_\theta(\mathbf{z}))]. \quad (3.14)$$

Procedura uczenia i optymalizacja

Uczenie sieci realizowane jest w schemacie naprzemiennej optymalizacji, w którym najpierw aktualizowane są parametry dyskryminatora przy stałych parametrach generatora, a następnie aktualizowane są parametry generatora. Do optymalizacji zastosowano algorytm Adam z parametrami:

$$\beta_1 = 0.5, \quad \beta_2 = 0.999, \quad \eta_G = 2 \cdot 10^{-4}, \quad \eta_D = 1 \cdot 10^{-4}.$$

Przyjęte wartości parametrów optymalizatora Adam odpowiadają ustawieniom powszechnie stosowanym w treningu modeli typu DCGAN i ich współczesnych wariantów, rekomendowanym w literaturze oraz praktyce eksperymentalnej, i zostały dobrane w celu zapewnienia stabilności procesu uczenia, a nie jako wynik odrębnej procedury strojenia hiperparametrów.

Dodatkowo stosowano uśrednianie wykładnicze wag generatora (EMA), które wykorzystywano na etapie generowania próbek, co pozwalało uzyskać bardziej stabilne i wizualnie spójne obrazy.

Monitorowanie jakości generowanych próbek

Jakość generowanych obrazów monitorowano za pomocą metryki Fréchet Inception Distance (FID), porównującej statystyki rozkładów cech obrazów rzeczywistych i syntetycznych. Trening zakończono na epoce 182, ponieważ wartość FID nie ulegała istotnej poprawie przez kolejne 10 epok (183–192), co uznano za sygnał osiągnięcia konwergencji jakości generowanych próbek.

Wykorzystanie wygenerowanych obrazów

Po zakończeniu treningu generatora DCGAN wygenerowano zbiór obrazów syntetycznych, który następnie poddano etapowi selekcji jakościowej. Celem tego etapu było odrzucenie próbek o bardzo niskiej zawartości informacyjnej, które mogłyby negatywnie wpływać na proces uczenia klasyfikatora ELA, w szczególności poprzez wprowadzenie trywialnych lub nieinformatywnych przykładów klasy *AI*.

Każda wygenerowana próbka $\mathbf{x} \in [0, 1]^{3 \times H \times W}$ była analizowana przy użyciu prostej heurytyki detekcji obrazów „zbyt płaskich”, tj. niemal jednorodnych wizualnie. Próbka była odrzucona, jeżeli spełniony był co najmniej jeden z następujących warunków:

- globalne odchylenie standardowe intensywności pikseli

$$\sigma(\mathbf{x}) < \tau_{\text{std}},$$

- zakres wartości jasności

$$\max(\mathbf{x}) - \min(\mathbf{x}) < \tau_{\text{range}},$$

- bardzo niska średnia wartość modułu gradientu obrazu, wyznaczona z użyciem filtru Sobela:

$$\mathbb{E}[\|\nabla_{\text{Sobel}}(\mathbf{x})\|] < \tau_{\text{edge}}.$$

W praktyce oznacza to, że próbki niemal jednolite kolorystycznie, pozbawione wyraźnych struktur krawędziowych lub detali przestrzennych (np. obrazy „szare”, rozmyte lub o minimalnym kontraście) były automatycznie eliminowane ze zbioru. W implementacji zastosowano progi empiryczne:

$$\tau_{\text{std}} = 0.06, \quad \tau_{\text{range}} = 0.20, \quad \tau_{\text{edge}} = 0.04,$$

które dobrano empirycznie na podstawie wizualnej inspekcji wygenerowanych próbek oraz analizy rozkładów prostych statystyk obrazu, w taki sposób, aby usuwać jedynie skrajnie nieinformacyjne próbki, nie ograniczając jednocześnie różnorodności wizualnej danych syntetycznych.

Proces generowania był iteracyjny i obejmował mechanizm nadpróbkowania, polegający na generowaniu większej liczby obrazów niż docelowa i zapisywaniu jedynie tych, które spełniały kryteria jakościowe. Takie podejście pozwoliło uzyskać zbiór syntetycznych obrazów charakteryzujących się wystarczającą złożonością strukturalną.

Ostatecznie wyselekcjonowane obrazy syntetyczne zostały wykorzystane jako kontrolowana klasa *AI* w procesie treningu klasyfikatora ELA z modelem SVC. Ich rola polegała na dostarczeniu przykładów generatywnych, dla których artefakty kompresji JPEG oraz rozkład błędów rekodowania różnią się od obrazów rzeczywistych. Dzięki temu metoda ELA mogła uczyć się rozróżniania subtelnych wzorców charakterystycznych dla obrazów generowanych oraz pochodzących z rzeczywistych sensorów fotograficznych.

3.3.5. Metoda FFT (analiza widma częstotliwościowego)

Jednym z komponentów systemu detekcji jest metoda oparta na analizie statystycznych właściwości widma częstotliwościowego obrazu w dziedzinie Fouriera. Metoda ta pełni rolę pomocniczego klasyfikatora i dostarcza dodatkowego sygnału wykorzystywanego w procesie agregacji wyników wielu niezależnych metod.

Podejście to bazuje na obserwacji, że obrazy generowane przez modele sztucznej inteligencji mogą wykazywać systematyczne różnice w rozkładzie energii widma częstotliwościowego w porównaniu do fotografii rzeczywistych. Zależności tego typu były wcześniej analizowane w literaturze, m.in. w kontekście rozbieżności widma Fouriera obrazów syntetycznych oraz naturalnych [13, 15].

Przygotowanie obrazu i transformata Fouriera

Obraz wejściowy jest w pierwszym kroku konwertowany do skali szarości, z uwzględnieniem orientacji zapisanej w metadanych EXIF. Następnie obraz jest przeskalowywany do stałego rozmiaru 512×512 pikseli, co zapewnia porównywalność analizowanych widm oraz jednorodną rozdzielcość przestrzeni częstotliwościowej.

Dla tak przygotowanego obrazu obliczana jest dwuwymiarowa dyskretna transformata Fouriera (FFT), po czym wykonywane jest przesunięcie widma (`fftshift`), tak aby składowe niskoczęstotliwościowe znalazły się w centrum obrazu widma. Analizie podlega moduł transformaty:

$$A(u, v) = |\mathcal{F}\{I(x, y)\}|.$$

W celu ograniczenia wpływu dużej dynamiki wartości widma stosowana jest transformacja logarytmiczna postaci $\log(1 + A)$ oraz normalizacja względem maksymalnej wartości widma.

Ekstrakcja cech widmowych

Analiza widma prowadzona jest w układzie radialnym. Dla każdego punktu widma wyznaczana jest jego odległość od środka układu częstotliwości:

$$r(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2},$$

gdzie (u_0, v_0) oznacza współrzędne środka widma.

Zakres promieni $[0, r_{\max}]$ dzielony jest na $N = 8$ rozłącznych pasm radialnych o jednakowej szerokości. Dla każdego pasma obliczany jest udział energii widma względem całkowitej energii:

$$f_i = \frac{\sum_{(u,v) \in \Omega_i} A(u, v)}{\sum_{u,v} A(u, v)}, \quad i = 1, \dots, 8,$$

gdzie Ω_i oznacza zbiór punktów należących do i -tego pierścienia radialnego.

Liczba pasm radialnych $N = 8$ dobrano jako kompromis pomiędzy rozdzielcością opisu widma a stabilnością estymacji statystyk w poszczególnych pasmach; wartość ta pozwala na uchwycenie ogólnego rozkładu energii częstotliwościowej bez nadmiernego zwiększania wymiaru wektora cech.

Dodatkowo wyznaczany jest współczynnik udziału wysokich częstotliwości:

$$f_{\text{HF}} = \sum_{i=5}^8 f_i,$$

który stanowi miarę względnej ilości energii skupionej w górnej połowie pasm częstotliwościowych.

Uzupełniająco obliczane są dwie globalne statystyki znormalizowanego logarytmicznego widma: wartość średnia μ_{FFT} oraz odchylenie standardowe σ_{FFT} . Ostateczny wektor cech przyjmuje postać:

$$\mathbf{f} = [f_{\text{HF}}, \mu_{\text{FFT}}, \sigma_{\text{FFT}}, f_1, \dots, f_8]^\top.$$

Klasyfikacja regresją logistyczną

Klasyfikacja realizowana jest z wykorzystaniem regresji logistycznej wytrenowanej wcześniej na opisanych cechach widmowych. Model uczyony jest w trybie nadzorowanym na zbalansowanym zbiorze obrazów rzeczywistych i syntetycznych, a po zakończeniu uczenia jego parametry wykorzystywane są wyłącznie w trybie inferencyjnym.

Dla danego obrazu obliczana jest wartość funkcji decyzyjnej:

$$z = \alpha + w_{\text{HF}} f_{\text{HF}} + w_{\mu} \mu_{\text{FFT}} + w_{\sigma} \sigma_{\text{FFT}} + \sum_{i=1}^8 w_i f_i,$$

a następnie wyznaczane jest prawdopodobieństwo przynależności do klasy obrazów generowanych przez sztuczną inteligencję:

$$p_{\text{AI}} = \frac{1}{1 + e^{-z}}.$$

Wartość $p_{\text{AI}} \in [0, 1]$ interpretowana jest jako stopień przekonania metody, że analizowany obraz ma pochodzenie syntetyczne.

Dane treningowe i przygotowanie zbioru

Model regresji logistycznej wykorzystywany w metodzie FFT trenowano na zbalansowanym zbiorze o rozmiarze 4000 obrazów, obejmującym 2000 przykładów klasy REAL oraz 2000 przykładów klasy AI. Zbiór został zbudowany przez agregację kilku niezależnych źródeł danych, tak aby zapewnić zróżnicowanie wykorzystywanych modeli generatywnych, typów scen oraz charakterystyk obrazów.

Dobór danych treningowych miał na celu zapewnienie możliwie szerokiego pokrycia potencjalnych źródeł i właściwości obrazów, z jakimi metoda może się spotkać w praktyce. Zróżnicowanie zbioru umożliwia uczenie modelu ogólnych zależności statystycznych w domenie częstotliwościowej, zamiast dopasowania do specyficznych cech pojedynczego generatora, stylu wizualnego lub konkretnego zbioru danych.

Cześć AI (2000). Zbiór obrazów syntetycznych utworzono z następujących komponentów:

1. **Własne generacje (360 obrazów).** Wygenerowano po 120 obrazów dla każdej z trzech technologii: *BigGAN*, *Stable Diffusion* oraz *DALLÉ 2*. Dla każdej technologii wygenerowano obrazy w sześciu kategoriach semantycznych: *animals*, *architecture*, *city*, *food*, *nature*, *people*, po 20 obrazów na kategorie (łącznie $3 \cdot 6 \cdot 20 = 360$).
2. **Zbiór Kaggle [5] (170 obrazów AI).** Wylosowano obrazy oznaczone w tym zbiorze jako AI, traktując je jako dodatkowe źródło zróżnicowanych przykładów syntetycznych.
3. **Zbiór SuSy [19] (1470 obrazów AI).** Wylosowano obrazy z podzbioru val w celu zwiększenia różnorodności generatorów, rozdzielczości oraz typów scen.

Część REAL (2000). Zbiór fotografii rzeczywistych skonstruowano analogicznie jako agregację kilku źródeł:

1. **Pexels (360 obrazów REAL).** Pobrano fotografie w sześciu kategoriach semantycznych: *animals*, *architecture*, *city*, *food*, *nature*, *people*, po 60 zdjęć na kategorię (łącznie $6 \cdot 60 = 360$), aby zapewnić porównywalne pokrycie typów scen jak w części syntetycznej.
2. **Zbiór Kaggle [5] (170 obrazów REAL).** Wylosowano fotografie oznaczone jako REAL.
3. **Zbiór zdjęć aparatu (670 obrazów REAL).** Wykorzystano fotografie pochodzące z wydzielonego zbioru testowego przygotowanego na potrzeby eksperymentów PRNU, obejmujące obrazy z realnych urządzeń o wysokich rozdzielczościach.
4. **Natural Images [28] (800 obrazów REAL).** Wylosowano po 100 obrazów z każdej z ośmiu klas: *Airplane*, *Car*, *Cat*, *Dog*, *Flower*, *Fruit*, *Motorbike*, *Person* (łącznie $8 \cdot 100 = 800$).

Zwracane wartości i artefakty diagnostyczne

Metoda FFT zwraca obiekt typu `MethodResult`, który zawiera:

- **name:** identyfikator metody (`fft`),
- **score:** wartość $p_{AI} \in [0, 1]$ wyznaczoną przez model regresji logistycznej,
- **metrics:** zestaw metryk diagnostycznych obejmujących wartości cech widmowych, parametry modelu oraz wartość funkcji decyzyjnej z ,
- **visuals_b64:** wizualizację logarytmicznej amplitudy widma Fouriera, umożliwiającą jakaś ocenę rozkładu energii częstotliwościowej obrazu.

Wizualizacja widma konstruowana jest jako:

$$\mathbf{S} = \frac{\log(1 + |\mathbf{F}_{\text{shift}}|)}{\max \log(1 + |\mathbf{F}_{\text{shift}}|) + \epsilon},$$

i służy wyłącznie celom diagnostycznym oraz interpretacyjnym.

3.3.6. Metoda CLIP zero-shot

Jednym z komponentów systemu jest metoda klasyfikacji zero-shot oparta na modelu CLIP (*Contrastive Language–Image Pre-training*), której celem jest oszacowanie, czy analizowany obraz jest bardziej podobny do fotografii generowanej przez model sztucznej inteligencji, czy do fotografii rzeczywistej. Metoda nie wymaga trenowania klasyfikatora na danych zadaniowych i wykorzystuje wyłącznie uprzednio wytrenowany model CLIP [34].

W implementacji zastosowano wariant modelu ViT-B/32, który mapuje zarówno obrazy, jak i opisy tekstowe do wspólnej przestrzeni embeddingów. Klasyfikacja realizowana jest poprzez porównanie reprezentacji obrazu z reprezentacjami ręcznie zdefiniowanych opisów tekstowych (promptów), odpowiadających różnym klasom semantycznym.

Dla potrzeb metody przygotowano trzy grupy promptów: opisujące obrazy generowane przez modele AI (*AI-photo*), fotografie rzeczywiste (*REAL-photo*) oraz obrazy nieinformatywne (*UNKNOWN*). Wprowadzenie trzeciej klasy pełni rolę stabilizującą i jest koncepcyjnie zgodne z podejściami OOD (out-of-distribution), w których model otrzymuje możliwość przypisania części masy prawdopodobieństwa do kategorii nieprzynależności lub wysokiej niepewności [**fu2025clipscope**]. Pozwala to ograniczyć wpływ przypadków skrajnych (np. obrazów jednolitych lub silnie zdegradowanych) oraz umożliwia konstrukcję jednowymiarowego wyniku poprzez renormalizację klas *AI-photo* i *REAL-photo*.

Embeddingi obrazu i tekstu oraz miara podobieństwa

Niech $f_I(\cdot)$ oznacza enkoder obrazu CLIP, a $f_T(\cdot)$ enkoder tekstu. Dla obrazu \mathbf{I} oraz promptu tekstowego t wyznaczane są znormalizowane embeddingi:

$$\hat{\mathbf{v}} = \frac{f_I(\mathbf{I})}{\|f_I(\mathbf{I})\|_2}, \quad \hat{\mathbf{u}}(t) = \frac{f_T(t)}{\|f_T(t)\|_2}. \quad (3.15)$$

Podobieństwo obrazu do promptu liczone jest jako iloczyn skalarny, odpowiadający cosinusowi kąta pomiędzy wektorami znormalizowanymi:

$$s(\mathbf{I}, t) = \hat{\mathbf{v}}^\top \hat{\mathbf{u}}(t). \quad (3.16)$$

Zbiory promptów dzielone są na trzy grupy: \mathcal{T}_{AI} , \mathcal{T}_{REAL} oraz \mathcal{T}_{UNK} . Dla każdej grupy wyznaczane jest maksymalne podobieństwo:

$$z_{AI}(\mathbf{I}) = \max_{t \in \mathcal{T}_{AI}} s(\mathbf{I}, t), \quad z_{REAL}(\mathbf{I}) = \max_{t \in \mathcal{T}_{REAL}} s(\mathbf{I}, t), \quad z_{UNK}(\mathbf{I}) = \max_{t \in \mathcal{T}_{UNK}} s(\mathbf{I}, t). \quad (3.17)$$

Trójklasowy rozkład softmax i wynik binarny

Wektor logitów $\mathbf{z}(\mathbf{I}) = [z_{AI}, z_{REAL}, z_{UNK}]$ skalowany jest temperaturą τ (w implementacji $\tau = 50$), a następnie przekształcany funkcją softmax do rozkładu trójklasowego:

$$p_{AI}^{(3)}(\mathbf{I}), p_{REAL}^{(3)}(\mathbf{I}), p_{UNK}^{(3)}(\mathbf{I}) = \text{softmax}(\tau \cdot \mathbf{z}(\mathbf{I})). \quad (3.18)$$

Ponieważ interfejs systemu wymaga jednowymiarowego wyniku zgodnego z pozostałymi metodami, finalny score wyznaczany jest jako prawdopodobieństwo warunkowe klasy *AI-photo*

po wyłączeniu masy prawdopodobieństwa przypisanej klasie *UNKNOWN*:

$$\text{score}(\mathbf{I}) = p_{AI}^{(2)}(\mathbf{I}) = \frac{p_{AI}^{(3)}(\mathbf{I})}{p_{AI}^{(3)}(\mathbf{I}) + p_{REAL}^{(3)}(\mathbf{I})}. \quad (3.19)$$

Wartość z (3.19) stanowi **pseudo-prawdopodobieństwo** i nie jest skalibrowanym prawdopodobieństwem w sensie statystycznym. Należy ją interpretować jako monotoniczną miarę przechyłu w stronę klasy *AI-photo* względem klasy *REAL-photo*.

Wizualizacja: mapa różnic podobieństwa na poziomie patchy

Dla modelu ViT-B/32 obraz dzielony jest na tokeny (patch-e), dla których wyznaczane są znormalizowane embeddingi $\hat{\mathbf{p}}_i$. Następnie konstruowane są centroidy tekstowe dla klas *AI-photo* oraz *REAL-photo* jako średnie embeddingów promptów, po czym wykonywana jest normalizacja:

$$\hat{\mathbf{c}}_{AI}, \quad \hat{\mathbf{c}}_{REAL}. \quad (3.20)$$

Dla każdego patcha obliczana jest różnica podobieństw:

$$\Delta_i = \hat{\mathbf{p}}_i^\top \hat{\mathbf{c}}_{AI} - \hat{\mathbf{p}}_i^\top \hat{\mathbf{c}}_{REAL}. \quad (3.21)$$

Znormalizowana mapa Δ renderowana jest jako półprzezroczysta nakładka na obraz, stanowiąca artefakt diagnostyczny ułatwiający jakościową analizę decyzji metody na pojedynczych przykładach.

Zwracane wartości i artefakty diagnostyczne

Metoda CLIP zero-shot zwraca obiekt typu `MethodResult`, który zawiera:

- **name**: identyfikator metody (`clip_zeroshot`),
- **score**: wartość $p_{AI}^{(2)}(\mathbf{I}) \in [0, 1]$ zgodnie z równaniem (3.19),
- **metrics**: zestaw metryk diagnostycznych obejmujących prawdopodobieństwa klas w wariancie dwu- i trójklasowym (`p_ai_bin`, `p_nonai_bin`, `p_ai_3way`, `p_nonai_3way`, `p_unknown_3way`) oraz miary niepewności w postaci entropii rozkładów (`entropy_2way`, `entropy_3way`),
- **visuals_b64**: wizualizację typu heatmap, stanowiącą przestrzenną mapę różnicy podobieństwa lokalnych fragmentów obrazu do opisów *AI-photo* oraz *REAL-photo*.

Zwracane artefakty mają charakter wyłącznie diagnostyczny i interpretacyjny i służą do analizy zachowania metody w przypadkach granicznych oraz do wizualnej inspekcji decyzji.

3.3.7. Model CLIP z klasyfikatorem SVM

Kolejna zaimplementowana metoda opiera się na wykorzystaniu bogatych reprezentacji cech oferowanych przez modele wizyjno-językowe (VLM). Główną inspiracją dla tego podejścia jest praca [10], w której autorzy stawiają tezę, że do skutecznej detekcji obrazów syntetycznych nie jest wymagany intensywny trening na ogromnych, domenowych zbiorach danych.

Zamysł metody opiera się na wykorzystaniu enkodera obrazu modelu **CLIP** jako ekstraktora cech. W przeciwieństwie do klasycznych detektorów skupionych na niskopoziomowych artefaktach, podejście to wykorzystuje cechy ekstrahowane z **przedostatniej warstwy sieci** (*next – to – lastlayer*), co pozwala uchwycić sygnatury generatywne na wyższym poziomie semantycznym.

Kluczowe aspekty tej metody, uwzględnione w implementacji, to:

- **Strategia few-shot:** Wykorzystanie jedynie niewielkiej liczby par obrazów rzeczywistych i syntetycznych (referencyjnych) do zamodelowania przestrzeni decyzyjnej.
- **Klasyfikacja liniowa:** Zastosowanie liniowego klasyfikatora **SVM** trenowanego na wektorach cech CLIP, co minimalizuje ryzyko przeuczenia i poprawia generalizację na nieznane wcześniej generatory.

Przygotowanie zbioru danych i parowanie obrazów

W celu zapewnienia wysokiej zdolności do generalizacji oraz minimalizacji błędów semantycznych, proces tworzenia zbioru danych oparto na strategii parowania obrazów (*paired image strategy*) zaproponowanej w pracy [10]. Podejście to polega na generowaniu syntetycznych odpowiedników dla konkretnych obrazów rzeczywistych, co wymusza na klasyfikatorze skupienie się na artefaktach procesu syntezy zamiast na różnicach w zawartości sceny.

Jako bazę dla obrazów autentycznych wykorzystano zbiór **FFHQ** (*Flickr-Faces-HQ*), zawierający wysoką jakość portrety ludzkie. Warstwę syntetyczną wygenerowano przy użyciu modelu dyfuzyjnego **Stable Diffusion v1.5** (runwayml/stable-diffusion-v1-5).

Zamiast generacji wyłącznie na podstawie tekstu, zastosowano technikę **Img2Img** (*image-to-image*), która pozwala na transformację istniejącego obrazu rzeczywistego R_i w jego syntetyczny odpowiednik F_i . Proces ten można opisać za pomocą funkcji:

$$F_i = \mathcal{G}_{\text{Img2Img}}(R_i, \tau, S) \quad (3.22)$$

gdzie τ oznacza opis tekstowy (*prompt*), a S jest parametrem siły modyfikacji (*strength*).

W implementacji wykorzystano następujące parametry techniczne:

- **Prompt (τ):** „*detailed photograph of a face, 8k, photorealistic, identical features*” – wymuszający zachowanie wysokiej jakości przy jednoczesnym generowaniu nowych struktur charakterystycznych dla AI.
- **Strength ($S = 0,55$):** Kluczowy parametr kontrolujący stopień modyfikacji obrazu wewnętrznego. Wartość ta pozwala na zachowanie tożsamości osoby z obrazu referencyjnego, przy jednoczesnym całkowitym przerysowaniu tekstury i wprowadzeniu śladów dyfuzyjnych.
- **Parametry inferencji:** Liczba kroków odszumiania (*inference steps*) została ustalona na 45, a współczynnik prowadzenia (*guidance scale*) na 7,5.

Wynikiem procesu jest zbiór **400 par** (R_i, F_i), gdzie każdy obraz syntetyczny posiada etykietę powiązaną z jego rzeczywistym pierwowzorem. Wszystkie powiązania są składowane w pliku metadanych (`meta.csv`). Takie podejście umożliwia późniejszą ekstrakcję cech za pomocą modelu CLIP oraz trening klasyfikatora SVM.

Klasyfikacja cech CLIP za pomocą Linear SVM

Kolejny etap prac obejmował stworzenie modelu klasyfikacyjnego, zdolnego do odróżnienia obrazów rzeczywistych od syntetycznych na podstawie reprezentacji wygenerowanych przez model wizyjno-językowy. Proces ten podzielono na fazę ekstrakcji cech oraz fazę optymalizacji klasyfikatora liniowego.

Do ekstrakcji cech wizualnych wykorzystano model **CLIP ViT-L/14** w wersji trenowanej na zbiorze *DataComp-XL*. Zgodnie z metodą zaproponowaną w literaturze przedmiotu [10], zrezygnowano z pobierania cech z ostatniej warstwy multimodalnej na rzecz **przedostatniej warstwy sieci (next-to-last layer)**. W implementacji technicznej uzyskano to poprzez usunięcie warstwy projekcyjnej (`visual.proj = None`), co pozwoliło na uzyskanie surowych wektorów cech wizualnych o wymiarze 1024.

Proces uczenia przeprowadzono na wyekstrahowanych cechach z przygotowanych wcześniej **400 par obrazów** (łącznie 800 próbek). Jako model bazowy wybrano liniową maszynę wektorów nośnych (**Linear SVM**).

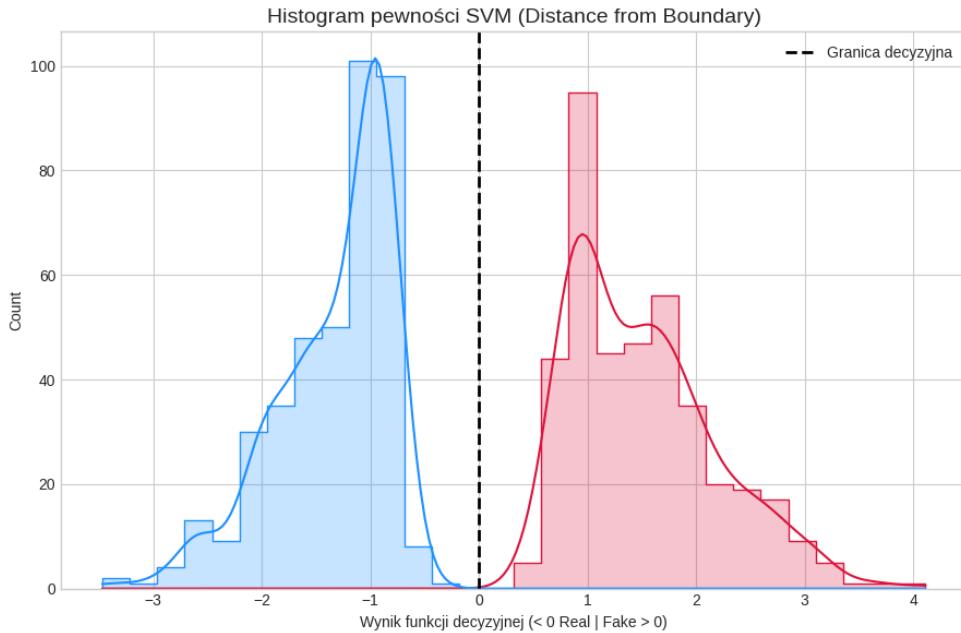
W celu uzyskania optymalnych parametrów modelu zastosowano procedurę przeszukiwania siatki (*Grid Search*) z 5-krotną walidacją krzyżową. Optymalizacji poddano parametr regularizacji C , testując wartości z zakresu $[0,001; 100]$, oraz wagę klas (*class weight*). Klasyfikator uczy się wyznaczać hiperpłaszczyznę rozdzielającą klasy poprzez minimalizację funkcji błędu:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(w^T x_i + b)) \quad (3.23)$$

gdzie x_i reprezentuje 1024-wymiarowy wektor cech wizualnych CLIP, a $y_i \in \{-1, 1\}$ oznacza etykietę przynależności do klasy (-1 dla obrazów rzeczywistych, 1 dla syntetycznych).

Analiza skuteczności i separowalności klas

Skuteczność doboru cech oraz proces optymalizacji modelu zweryfikowano poprzez szczegółową analizę wyników uczenia. Wykorzystanie procedury *Grid Search* pozwoliło na wyznaczenie optymalnych hiperparametrów klasyfikatora Linear SVM: parametr regularizacji $C = 0,01$ oraz brak wag klas (*class_weight: None*). Tak skonfigurowany model osiągnął bardzo wysoką dokładność w procesie 5-krotnej walidacji krzyżowej, wynoszącą **96,12%**.



Rysunek 3.2: Rozkład wyników funkcji decyzyjnej.

Pewność predykcji modelu oraz stopień separowalności klas w 1024-wymiarowej przestrzeni cech CLIP poddano analizie za pomocą funkcji decyzyjnej (*decision function*), mierzącej odległość próbek od hiperplaszczyzny rozdzielającej. Histogram rozkładu tych wyników 3.2 wykazał wyraźną separację między badanymi klasami. Obrazy rzeczywiste (FFHQ) skupiają się w obszarze wartości ujemnych, podczas gdy obrazy syntetyczne (Stable Diffusion v1.5) zajmują obszar wartości dodatnich.

3.3.8. Model CNN StyleGAN

Model CNN został zaprojektowany jako lekka, konwolucyjna sieć neuronowa przeznaczona do binarnej klasyfikacji obrazów na rzeczywiste oraz syntetyczne (*real vs. AI*). Metoda opiera się na uczeniu nadzorowanym i wykorzystuje zdolność sieci konwolucyjnych do automatycznej ekstrakcji cech wizualnych, w tym subtelnych artefaktów charakterystycznych dla obrazów generowanych przez modele sztucznej inteligencji, co jest zgodne z obserwacjami przedstawionymi w pracy [39].

Model przyjmuje na wejściu obrazy RGB o rozmiarze 224×224 pikseli i zwraca pojedynczą wartość z zakresu $(0, 1)$, interpretowaną jako prawdopodobieństwo przynależności obrazu do klasy syntetycznej.

Architektura modelu

Zastosowana architektura składa się z czterech bloków konwolucyjnych, z których każdy zawiera:

- warstwę konwolucyjną o jądrze 3×3 ,
- normalizację typu Batch Normalization,
- funkcję aktywacji ReLU,
- warstwę maksymalnego próbkowania (*MaxPooling*).

Po części konwolucyjnej następuje spłaszczenie map cech oraz część w pełni połączona z mechanizmem *dropout*, ograniczającym przeuczenie modelu. Warstwa wyjściowa wykorzystuje funkcję sigmoidalną, umożliwiającą interpretację wyniku jako prawdopodobieństwa klasy.

Szczegółową strukturę sieci przedstawiono w tabeli 3.1.

Tabela 3.1: Architektura modelu CNN do klasyfikacji obrazów rzeczywistych i syntetycznych

Warstwa	Rozmiar wyjścia	Parametry
Wejście	$3 \times 224 \times 224$	–
Conv2D + BN + ReLU + MaxPool	$32 \times 112 \times 112$	3×3
Conv2D + BN + ReLU + MaxPool	$64 \times 56 \times 56$	3×3
Conv2D + BN + ReLU + MaxPool	$128 \times 28 \times 28$	3×3
Conv2D + BN + ReLU + MaxPool	$256 \times 14 \times 14$	3×3
Flatten	$256 \cdot 14 \cdot 14$	–
Fully Connected	256	ReLU
Dropout	256	$p = 0.5$
Wyjście (FC)	1	Sigmoid

Proces treningu

Model został wytrenowany na pełnym zbiorze 140k-real-and-fake-faces [1]. Dataset zawiera 140 000 obrazów twarzy, obejmujących zarówno fotografie rzeczywiste, jak i obrazy syntetyczne wygenerowane przez model StyleGAN. Dane podzielono na zbiory treningowy, walidacyjny oraz testowy.

W trakcie treningu zastosowano augmentację danych, obejmującą m.in. losowe odbicia poziome, rotacje oraz modyfikacje jasności i kontrastu. Celem augmentacji było zwiększenie odporności modelu na zmienność danych wejściowych.

Funkcją straty była binarna entropia krzyżowa:

$$\mathcal{L} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})],$$

gdzie y oznacza etykietę rzeczywistą, a \hat{y} — predykcję modelu.

Do optymalizacji wykorzystano algorytm AdamW, a współczynnik uczenia regulowany był dynamicznie z użyciem strategii *ReduceLROnPlateau*. Trening prowadzono przez 14 epok, monitorując dokładność na zbiorze walidacyjnym. Na zbiorze testowym model osiągnął 97,6% dokładności (accuracy).

3.3.9. Model ConvNeXt Tiny do detekcji obrazów StyleGAN

W celu klasyfikacji obrazów na rzeczywiste oraz syntetyczne (wygenerowane przez model StyleGAN) zaimplementowano model oparty na architekturze **ConvNeXt Tiny**, będący rozszerzeniem metody zaproponowanej w poprzedniej sekcji. Wybór tego modelu podyktowany był jego wysoką skutecznością w zadaniach rozpoznawania obrazów oraz zdolnością do ekstrakcji subtelnych cech teksturalnych, które często odróżniają obrazy AI od fotografii.

Model wykorzystuje technikę uczenia transferowego (*transfer learning*), bazując na wagach wstępnie wytrenowanych na zbiorze ImageNet-1K. Wejście sieci stanowią obrazy RGB o rozdzielcości 224×224 pikseli.

Dataset

Proces przygotowania danych obejmował podział zbioru [1] na trzy rozłączne podzbiory:

- **zbiór treningowy:** $5 \cdot 10^4$ obrazów, wykorzystany do optymalizacji wag modelu;
- **zbiór walidacyjny:** 10^4 obrazów, służący do monitorowania procesu uczenia i doboru hiperparametrów;
- **zbiór testowy:** 10^4 obrazów, użyty wyłącznie do końcowej oceny zdolności generalizacyjnych modelu.

Architektura systemu

Architektura składa się z dwóch głównych modułów: modelu bazowego (*backbone*) oraz dedykowanej głowicy klasyfikacyjnej (*classification head*):

- **Backbone (ConvNeXt Tiny):** Wykorzystany jako ekstraktor cech, kończący się warstwą *Global Average Pooling*, która generuje wektor o rozmiarze 768.
- **Głowica klasyfikacyjna:** Składa się z warstwy normalizacji (*LayerNorm*), dwóch warstw liniowych rozdzielonych funkcją aktywacji ReLU oraz mechanizmów regularyzacji *dropout* w celu zapobiegania przeuczeniu.

Szczegółową strukturę warstw przedstawiono w tabeli 3.2.

Tabela 3.2: Architektura modelu opartego na ConvNeXt Tiny wykorzystana w badaniach.

Warstwa / Moduł	Rozmiar wyjścia	Parametry / Aktywacja
Obraz wejściowy	$3 \times 224 \times 224$	–
ConvNeXt Tiny (Backbone)	768	Wagi ImageNet
Layer Normalization	768	–
Dropout	768	$p = 0.4$
Linear (Fully Connected)	128	ReLU
Dropout	128	$p = 0.3$
Linear (Wyjście)	1	Logit

Proces treningu

Trening modelu został podzielony na dwa etapy (tzw. *staged training*), co pozwala na stabilne dostosowanie wag do specyfiki zbioru *140k-real-and-fake-faces*:

1. **Etap 1 (Transfer Learning):** Zamrożono wszystkie parametry modelu bazowego sieci, trenując wyłącznie nowo dodaną głowicę klasyfikacyjną. Zastosowano współczynnik uczenia (*learning rate*) wynoszący 10^{-3} przez 5 epok.
2. **Etap 2 (Fine-tuning):** Odmrożono ostatni blok konwolucyjny (*Stage 3*), pozwalając na precyzyjne dostrojenie filtrów do artefaktów generatywnych StyleGAN. W tym etapie zmniejszono współczynnik uczenia do 10^{-4} .

Wyniki inferencji na zbiorze testowym

W procesie ewaluacji modelu po drugim etapie dostrajania, na wydzielonym zbiorze testowym uzyskano wysoką skuteczność klasyfikacji, osiągając wynik metryki accuracy na poziomie **99.55%**.

3.3.10. Model GAN vs Diffusion

Celem opisywanego modelu jest klasyfikacja obrazów wygenerowanych przez sztuczną inteligencję do jednej z dwóch głównych rodzin generatorów: modeli typu **GAN** oraz modeli **dyfuzyjnych**. Model ten nie rozstrzyga, czy obraz jest rzeczywisty, lecz zakłada, że analizowany materiał pochodzi z generatora AI i koncentruje się na identyfikacji jego klasy źródłowej.

Architektura modelu

Jako model bazowy wykorzystano architekturę **ConvNeXt-Tiny**, wstępnie wytrenowaną na zbiorze ImageNet-1K. Zastosowanie modelu preturenowanego umożliwia wykorzystanie transfer learningu i znacząco przyspiesza proces konwergencji podczas treningu.

Oryginalna warstwa klasyfikacyjna została zastąpiona nową warstwą liniową, dostosowaną do zadania klasyfikacji binarnej (GAN vs Diffusion). W procesie uczenia:

- zamrożono większość warstw ekstrakcji cech,
- odblokowano ostatni blok konwolucyjny sieci,
- trenowano od nowa warstwę klasyfikacyjną.

Takie podejście pozwala zachować ogólne reprezentacje wizualne wyuczone na dużym zbiorze danych, jednocześnie adaptując model do specyficznych artefaktów charakterystycznych dla generatorów obrazów AI.

Przetwarzanie danych wejściowych

Obrazy wejściowe były skalowane do rozmiaru 224×224 piksele oraz normalizowane zgodnie ze statystykami zbioru ImageNet. Dla zbioru treningowego zastosowano augmentację danych w postaci losowego odbicia horyzontalnego, co zwiększa odporność modelu na zmiany orientacji obrazu.

Przygotowanie danych treningowych

Do treningu modelu wykorzystano zbiór obrazów pochodzących z [4]. Zbiór ten zawiera obrazy w katalogach podzielonych na różne modele generatywne. Dane wejściowe zostały podzielone na dwie klasy odpowiadające rodzinom generatorów:

- **GAN** – obejmujące obrazy wygenerowane przez modele takie jak StyleGAN, BigGAN, CycleGAN czy ProGAN,
- **Diffusion** – obejmujące obrazy pochodzące z modeli dyfuzyjnych, m.in. Stable Diffusion, DDPM, GLIDE oraz Latent Diffusion.

Dla każdego generatora rekurencyjnie wczytano obrazy w formatach PNG oraz JPEG, następnie zrównoważono liczbę obrazów z obu klas, w wyniku czego zbiór zawierał po 14910 obrazów dla każdej z klas. Dane podzielono na zbiory treningowy oraz testowy w proporcji 80/20.

W celu uniknięcia stronniczości modelu przeprowadzono balansowanie klas, polegające na losowym ograniczeniu liczby obrazów klasy GAN do liczności klasy Diffusion, osobno dla zbiorów treningowego i testowego.

Proces treningu

Model trenowano przez 8 epok z wykorzystaniem funkcji straty *CrossEntropyLoss*, odpowiednie dla klasyfikacji wieloklasowej, przy czym w docelowej konfiguracji model rozwiązuje problem binarny. Optymalizację przeprowadzono algorytmem AdamW z rozdzielonymi współczynnikami uczenia:

- wyższym dla warstwy klasyfikacyjnej,
- niższym dla odblokowanego fragmentu sieci bazowej.

Dodatkowo zastosowano harmonogram zmiany współczynnika uczenia oparty na funkcji Cosine Annealing oraz trening w trybie mieszanej precyzji (AMP), co pozwoliło ograniczyć zużycie pamięci GPU i skrócić czas uczenia.

Po zakończeniu treningu model osiągnął wysoką skuteczność klasyfikacji na zbiorze testowym, uzyskując 93.9% accuracy.

Integracja z systemem

Dla każdego obrazu zwraca:

- przewidywaną klasę generatora (GAN lub Diffusion),
- poziom pewności predykcji,
- wizualizację Grad-CAM, wskazującą obszary obrazu istotne dla decyzji modelu.

3.3.11. Model ConvNeXt Diffusion

Model ConvNeXt Diffusion służy do wieloklasowej klasyfikacji obrazów w celu identyfikacji potencjalnej rodziny generatora dyfuzyjnego, z którego pochodzi analizowany obraz. W odróżnieniu od modeli binarnych, jego zadaniem jest bardziej szczegółowa analiza pochodzenia obrazu AI, obejmująca rozróżnienie pomiędzy różnymi wersjami i architekturami modeli dyfuzyjnych oraz klasą obrazów rzeczywistych.

Model klasyfikuje obraz do jednej z ośmiu klas:

- **sd1** – modele Stable Diffusion 1.x,

- **sd2** – Stable Diffusion 2.x,
- **sdxl** – rodzina Stable Diffusion XL,
- **sd3** – Stable Diffusion 3 oraz modele pokrewne,
- **ssd** – modele typu SSD,
- **pixart** – modele PixArt,
- **other_diffusion** – pozostałe modele dyfuzyjne,
- **real** – obrazy rzeczywiste.

Architektura modelu

Podobnie jak w pozostałych modułach opartych na sieciach konwolucyjnych, jako model bazowy zastosowano architekturę **ConvNeXt-Tiny** wstępnie wytrenowaną na zbiorze ImageNet-1K.

Oryginalna warstwa klasyfikacyjna została zastąpiona nowym blokiem obejmującym warstwę Dropout oraz warstwę liniową, dostosowaną do klasyfikacji ośmioklasowej. W procesie uczenia zamrożono większość warstw ekstrakcji cech, a odblokowano jedynie dwa ostatnie bloki konwolucyjne oraz warstwę klasyfikacyjną, co umożliwia adaptację modelu do subtelnego artefaktów charakterystycznych dla poszczególnych rodzin modeli dyfuzyjnych.

Przetwarzanie danych wejściowych

Obrazy wejściowe były skalowane do rozmiaru 224×224 piksele oraz normalizowane zgodnie z parametrami zbioru ImageNet. Dla zbioru treningowego zastosowano augmentację danych obejmującą losowe przycięcia oraz odbicia horyzontalne, co zwiększa odporność modelu na zmiany skali i kompozycji obrazu.

Zbiór danych

Do treningu oraz ewaluacji modelu wykorzystano publiczny zbiór danych DRAGON [11], zawierający obrazy rzeczywiste oraz obrazy wygenerowane przez szeroką gamę nowoczesnych modeli dyfuzyjnych. Każdej próbce przypisana jest informacja o modelu generującym obraz, na podstawie której dokonano mapowania do jednej z ośmiu klas.

Zbiór danych przetwarzany był w trybie strumieniowym, co umożliwiło efektywną pracę z bardzo dużą liczbą próbek bez konieczności lokalnego przechowywania całego zestawu danych.

Proces treningu

Model trenowano przez 8 epok z wykorzystaniem funkcji straty *CrossEntropyLoss* z wygładzaniem etykiet (label smoothing), co ogranicza nadmierną pewność predykcji i poprawia generalizację. Do optymalizacji wykorzystano algorytm AdamW z różnymi współczynnikami uczenia dla warstwy klasyfikacyjnej oraz odblokowanych fragmentów sieci bazowej.

Dodatkowo zastosowano harmonogram Cosine Annealing oraz trening w trybie mieszanej precyzji (AMP), co pozwoliło przyspieszyć proces uczenia i ograniczyć zużycie zasobów obliczeniowych.

Model osiągnął dokładność klasyfikacji na poziomie około 71% na pełnym zbiorze testowym, co potwierdza jego zdolność do rozróżniania pomiędzy głównymi rodzinami nowoczesnych modeli dyfuzyjnych oraz obrazami rzeczywistymi.

3.3.12. Model CLIP do klasyfikacji modelu generatora

Model CLIP został włączony do systemu jako dodatkowy moduł analityczny, w szczególności z myślą o rozpoznawaniu obrazów generowanych przez nowoczesne modele dyfuzyjne, w tym modele zamknięte i komercyjne, takie jak Midjourney czy DALL·E 3. W przeciwnieństwie do podejść opartych wyłącznie na analizie niskopoziomowych artefaktów wizualnych, model ten wykorzystuje reprezentacje o charakterze semantycznym, co pozwala uchwycić różnice wynikające z wysokopoziomowych cech generowanych obrazów. Inspiracją do implementacji tej metody były badania pokazujące skuteczność wykorzystania reprezentacji CLIP do detekcji obrazów generowanych przez różne modele AI [8, 27].

Zadaniem modelu jest przypisanie obrazu do jednej z pięciu klas odpowiadających generatorom:

- **sd21** – Stable Diffusion 2.1,
- **sdxl** – Stable Diffusion XL,
- **sd3** – Stable Diffusion 3,
- **dalle3** – DALL·E 3,
- **midjourney** – Midjourney.

Architektura modelu

Jako model bazowy zastosowano architekturę **CLIP ViT-B/32**. Model generuje dla obrazu wektor cech o stałym wymiarze $d = 512$:

$$\mathbf{z} = f_{\text{CLIP}}(\mathbf{x}), \quad \mathbf{z} \in \mathbb{R}^{512},$$

gdzie \mathbf{x} oznacza obraz wejściowy, a f_{CLIP} – funkcję ekstrakcji cech wizualnych.

Na wyjściu CLIP zastosowano prosty klasyfikator liniowy (tzw. *linear probe*), realizujący mapowanie:

$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{z} + \mathbf{b},$$

gdzie $\mathbf{W} \in \mathbb{R}^{C \times 512}$, $C = 5$ oznacza liczbę klas, a $\hat{\mathbf{y}}$ to wektor logitów klas.

Strategia uczenia

Zastosowano podejście **częściowego fine-tuningu**. Większość parametrów modelu CLIP została zamrożona, a uczeniu podlegały jedynie:

- ostatni blok transformera wizualnego,
- warstwa normalizacji końcowej (`ln_final`),
- projekcja wyjściowa,
- warstwa klasyfikacyjna (linear probe).

Zbiór danych

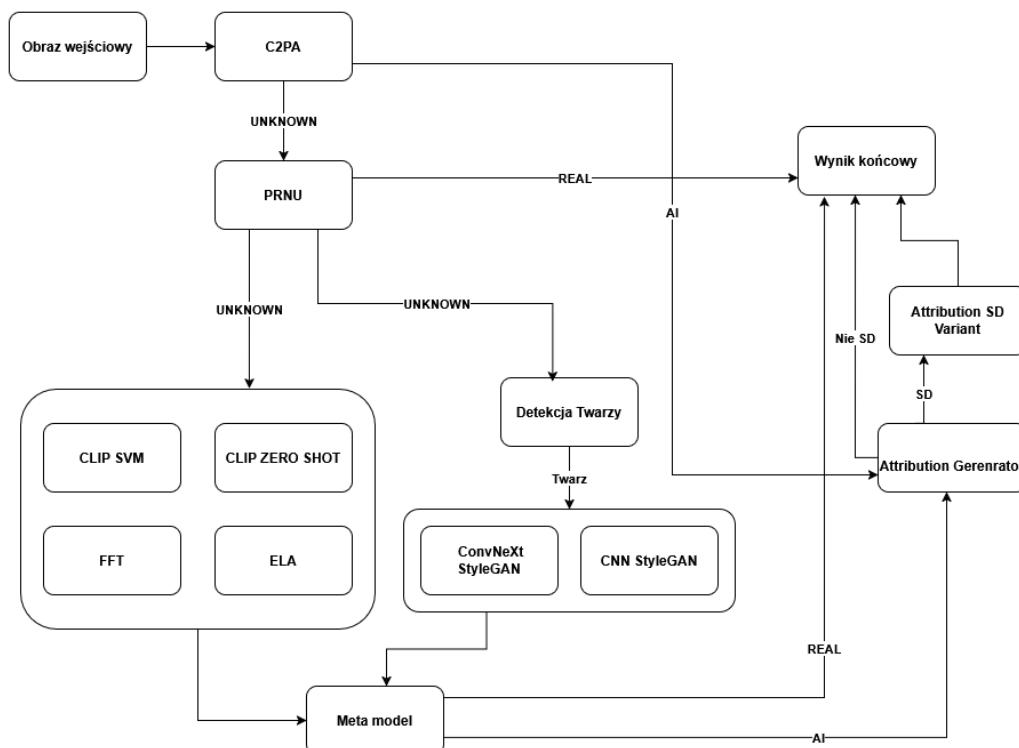
Model trenowano i ewaluowano na zbiorze **Defactify Image Dataset** [36], zawierającym obrazy wygenerowane przez różne nowoczesne modele generatywne. Etykiety źródłowe zmieniano do pięciu docelowych klas generatorów.

3.3.13. Zintegrowany pipeline decyzyjny z meta-modelem agregującym

System realizuje analizę obrazów w ramach zintegrowanego pipeline'u decyzyjnego, który łączy wiele metod detekcyjnych oraz mechanizm agregacji oparty na meta-modelu. Pipeline ten stanowi podstawowy tryb działania systemu i jest konfigurowany poprzez wybór jednego z trzech wariantów: `combined_methods`, `combined_methods_fast` lub `meta_model_only`.

Celem pipeline'u jest wygenerowanie końcowego werdyktu AI/REAL oraz ciągłej wartości decyzyjnej `score_ai`, przy jednoczesnym zachowaniu pełnej ścieżki diagnostycznej w postaci metryk i wizualizacji generowanych przez poszczególne metody składowe.

Ogólny przebieg zintegrowanego pipeline'u decyzyjnego w wariancie `combined_methods`, wraz z regułami bramkującymi oraz etapem agregacji przez meta-model, przedstawiono na rysunku 3.3.



Rysunek 3.3: Schemat zintegrowanego pipeline'u decyzyjnego w wariancie `combined_methods`, obejmujący reguły bramkujące, metody bazowe oraz agregację wyników przez meta-model

Warianty pipeline'u

Zintegrowany pipeline może działać w jednym z trzech wariantów:

- **combined_methods** — wariant pełny, obejmujący reguły bramkujące, uruchomienie metod bazowych oraz agregację wyników przez meta-model.

- **combined_methods_fast** — wariant przyspieszony, w którym pomijane są wybrane kosztowne obliczeniowo komponenty, przy zachowaniu agregacji przez meta-model.
- **meta_model_only** — wariant ograniczony do metod niezbędnych do zbudowania wektora cech meta-modelu, bez stosowania reguł bramkujących.

Wybrany wariant determinuje zestaw uruchamianych metod oraz kolejność ich wykonania, jednak interpretacja wyniku końcowego pozostaje jednolita.

Reguły bramkujące (early exit)

W wariantach combined_methods oraz combined_methods_fast pipeline może zakończyć analizę na wczesnym etapie, jeżeli spełnione zostaną deterministyczne warunki bramkujące.

C2PA. Jeżeli metoda C2PA wykryje poprawnie zwalidowane informacje o generatywnym pochodzeniu obrazu, zwraca specjalną wartość `score = 99.9`. Sygnał ten traktowany jest jako jednoznaczny dowód pochodzenia syntetycznego i powoduje natychmiastowe zakończenie analizy z werdyktem AI. W takim przypadku meta-model nie jest uruchamiany.

PRNU. W wariantie combined_methods może zostać uruchomiona metoda PRNU. Jeżeli metoda zwróci sygnał bramkujący `score = 0.1`, pipeline kończy dalsze przetwarzanie obrazu i zwraca werdykt oparty wyłącznie na tym wyniku.

Metody bramkujące nie są traktowane jako źródła cech wejściowych dla meta-modelu i pełnią wyłącznie rolę sterującą przebiegiem analizy.

Metody bazowe wykorzystywane w agregacji

Jeżeli analiza nie zostanie zakończona na etapie bramkowania, uruchamiany jest zestaw metod bazowych, których wyniki wykorzystywane są do budowy wektora cech meta-modelu. Zestaw ten obejmuje różnorodne algorytmy detekcyjne analizujące obraz z odmiennych perspektyw, takich jak cechy statystyczne, strukturalne, częstotliwościowe czy semantyczne.

Każda metoda bazowa zwraca wartość `score` z zakresu $[0, 1]$, interpretowaną jako miękka miara przynależności obrazu do klasy AI. Wartości te nie są interpretowane samodzielnie, lecz traktowane jako cechy wejściowe dla mechanizmu agregacji realizowanego przez meta-model.

Konkretna lista metod bazowych wykorzystywanych w agregacji jest określona na etapie trenowania meta-modelu i zapisywana wraz z nim w postaci uporządkowanej listy nazw (`model_names`). W fazie wnioskowania system uruchamia dokładnie ten sam zestaw metod, zachowując spójność pomiędzy etapem uczenia a działaniem pipeline'u.

Detekcja twarzy i metody zależne od treści

Część metod bazowych jest wyspecjalizowana w analizie obrazów zawierających twarze (*face-only*). W celu zapewnienia spójności przetwarzania, informacja o obecności twarzy poznawana jest w jednolity sposób zarówno w procesie uczenia, jak i w fazie wnioskowania.

Detekcja twarzy realizowana jest z wykorzystaniem biblioteki `face_recognition` i algorytmu HOG. Dla każdego obrazu wyznaczana jest binarna flaga `has_face`, informująca o obecności co najmniej jednej twarzy.

Jeżeli obraz nie zawiera twarzy, metody *face-only* nie są uruchamiane, a ich wynik zastępowany jest wartością neutralną $p_{AI} = 0.5$.

Reprezentacja wejściowa meta-modelu

Niech M oznacza liczbę metod bazowych użytych w agregacji. Każda metoda m zwraca wartość $p_{AI}^{(m)} \in [0, 1]$. Wektor cech meta-modelu budowany jest bezpośrednio z tych wartości lub — w zależności od konfiguracji — z ich transformacji logitowej:

$$x^{(m)} = \text{logit}\left(p_{AI}^{(m)}\right) = \log\left(\frac{p_{AI}^{(m)}}{1 - p_{AI}^{(m)}}\right), \quad (3.24)$$

z numerycznym obcięciem argumentu do przedziału $[\varepsilon, 1 - \varepsilon]$, $\varepsilon = 10^{-6}$.

W sytuacjach awaryjnych (np. `NaN` lub `Inf` zwrócone przez metodę bazową) wartości zastępowane są neutralnym 0.5, a wszystkie niefinitywne elementy wektora cech są zerowane. Kolejność cech w wektorze jest ściśle określona przez listę `model_names` zapisaną wraz z meta-modelem i musi być zachowana w fazie wnioskowania.

Uczenie meta-modelu

Agregacja wyników realizowana jest przez binarny klasyfikator `CatBoostClassifier` z funkcją straty `Logloss` oraz metryką `AUC`. Meta-model trenowany jest na zbiorze *Tiny Gen Image* z wykorzystaniem wyników metod bazowych. W zależności od dostępności sprzętu obliczeniowego wykorzystywany jest tryb GPU lub CPU.

Wytrenowany model zapisywany jest do pliku `meta_model.joblib` wraz z kompletom metadanych konfiguracyjnych, obejmujących m.in. listę metod bazowych, sposób przekształcenia cech wejściowych oraz parametry detekcji twarzy.

Atrybucja źródła generacji

Jeżeli końcowy werdykt pipeline'u wskazuje na klasę AI, system może uruchomić dodatkowy etap atrybucji, którego celem jest identyfikacja rodzaju lub potencjalnego źródła generacji obrazu.

Proces atrybucji ma charakter hierarchiczny. W pierwszym etapie uruchamiana jest metoda identyfikująca ogólną klasę generatora obrazu. W drugim etapie, wykonywanym warunkowo, uruchamiana jest metoda identyfikująca wariant generatora, jeżeli wynik pierwszego etapu wskazuje na generator z rodziny modeli dyfuzyjnych.

Moduły atrybucyjne nie wpływają na decyzję detekcyjną ani na działanie meta-modelu i służą wyłącznie do wzbogacenia raportu końcowego.

Wynik końcowy

Wynikiem działania zintegrowanego pipeline'u jest raport w formacie JSON, zawierający:

- końcowy werdykt (AI/REAL) oraz wartość `score_ai`,
- wartości `score` wszystkich metod bazowych oraz (jeżeli aktywny) wynik meta-modelu,
- metryki diagnostyczne per-metoda,
- wizualizacje diagnostyczne generowane przez poszczególne komponenty,
- opcjonalne wyniki atrybucji źródła generacji.

Meta-model pełni rolę aggregatora sygnałów pochodzących z wielu detektorów, nie zastępując ich działania i zachowując pełną ścieżkę interpretacyjną systemu.

3.3.14. Moduł raportowania

Moduł raportowania odpowiada za agregację, ujednolicenie oraz udostępnianie wyników uzyskiwanych z poszczególnych modułów analizy obrazu. Jego zadaniem jest przekształcenie wyników cząstkowych metod detekcji w spójną reprezentację raportową, która może być wykorzystywana zarówno przez interfejs użytkownika, jak i w dalszej analizie eksperymentalnej.

W przypadku analizy pojedynczego obrazu moduł raportowania generuje raport w formacie JSON o ustalonej strukturze, niezależnej od liczby i rodzaju uruchomionych metod detekcji. Raport ten zawiera końcowy werdykt systemu (AI, REAL lub UNKNOWN) wraz z ciągłą miarą decyzyjną, wyniki cząstkowe poszczególnych metod, a także metryki diagnostyczne i dane pomocnicze, takie jak wybrane informacje z metadanych obrazu. W przypadku obrazów zaklasyfikowanych jako syntetyczne raport może dodatkowo zawierać wyniki atrybucji źródła generacji, np. rozpoznanie klasy generatora lub wariantu modelu dyfuzyjnego.

Raport JSON stanowi podstawowy kontrakt komunikacyjny pomiędzy backendem analitycznym a warstwą prezentacji. Jednolita struktura raportu pozwala na elastyczne konfigurowanie procesu analizy bez konieczności modyfikacji sposobu prezentacji wyników w interfejsie użytkownika.

W trybie analizy wsadowej moduł raportowania obsługuje przetwarzanie wielu obrazów jednocześnie. W przypadku korzystania z interfejsu API obrazy są przekazywane w postaci archiwum ZIP, a wyniki analizy zwracane są również jako archiwum ZIP zawierające zestaw artefaktów wynikowych. Podstawowym elementem jest raport tabelaryczny w formacie CSV, w którym każdy wiersz odpowiada jednemu obrazowi i zawiera końcowy wynik analizy, wyniki metod cząstkowych, metryki diagnostyczne oraz informacje o ewentualnych błędach przetwarzania.

Uzupełnieniem raportu CSV są pliki podsumowujące analizę całego zbioru. Obejmują one zliczenia obrazów przypisanych do poszczególnych klas decyzyjnych oraz raport statystyczny w formacie JSON, zawierający zagregowane statystyki opisowe kolumn wynikowych. W raporcie statystycznym uwzględniane są m.in. średnie, odchylenia standardowe, wartości minimalne i maksymalne, wybrane percentile oraz informacje o rozkładzie danych kategorycznych, takie jak liczba wartości unikalnych i najczęściej występujące kategorie. Dodatkowo możliwa jest identyfikacja przypadków skrajnych, np. obrazów o najwyższych i najniższych wartościach miary decyzyjnej.

Analogiczny zestaw raportów generowany jest w przypadku uruchomienia systemu lokalnie w trybie wsadowym. Wyniki analizy zapisywane są w katalogu wyjściowym w postaci plików CSV i JSON o tej samej strukturze, co w przypadku analizy realizowanej przez API. Zapewnia to spójność reprezentacji wyników niezależnie od sposobu uruchomienia systemu oraz umożliwia bezpośrednie porównywanie rezultatów uzyskanych w różnych trybach pracy.

3.4. Architektura systemu

3.4.1. Cel i założenia architektoniczne

Celem projektowanego systemu jest umożliwienie analizy autentyczności obrazów cyfrowych w sposób interaktywny oraz wsadowy, z wykorzystaniem wielu niezależnych metod analizy. System został zaprojektowany jako aplikacja klient–serwer, z wyraźnym podziałem odpo-

wiedzialności pomiędzy poszczególne komponenty oraz z naciskiem na modularność i możliwość dalszej rozbudowy.

Na etapie projektowania przyjęto następujące założenia architektoniczne:

- wyraźne oddzielenie warstwy prezentacji od warstwy obliczeniowej,
- stabilny i jednoznaczny interfejs komunikacji pomiędzy frontendem a backendem,
- modularna struktura backendu umożliwiająca łatwe dodawanie nowych metod analizy,
- obsługa zarówno pojedynczych obrazów, jak i analizy wsadowej,
- możliwość uruchamiania systemu w środowisku kontenerowym.

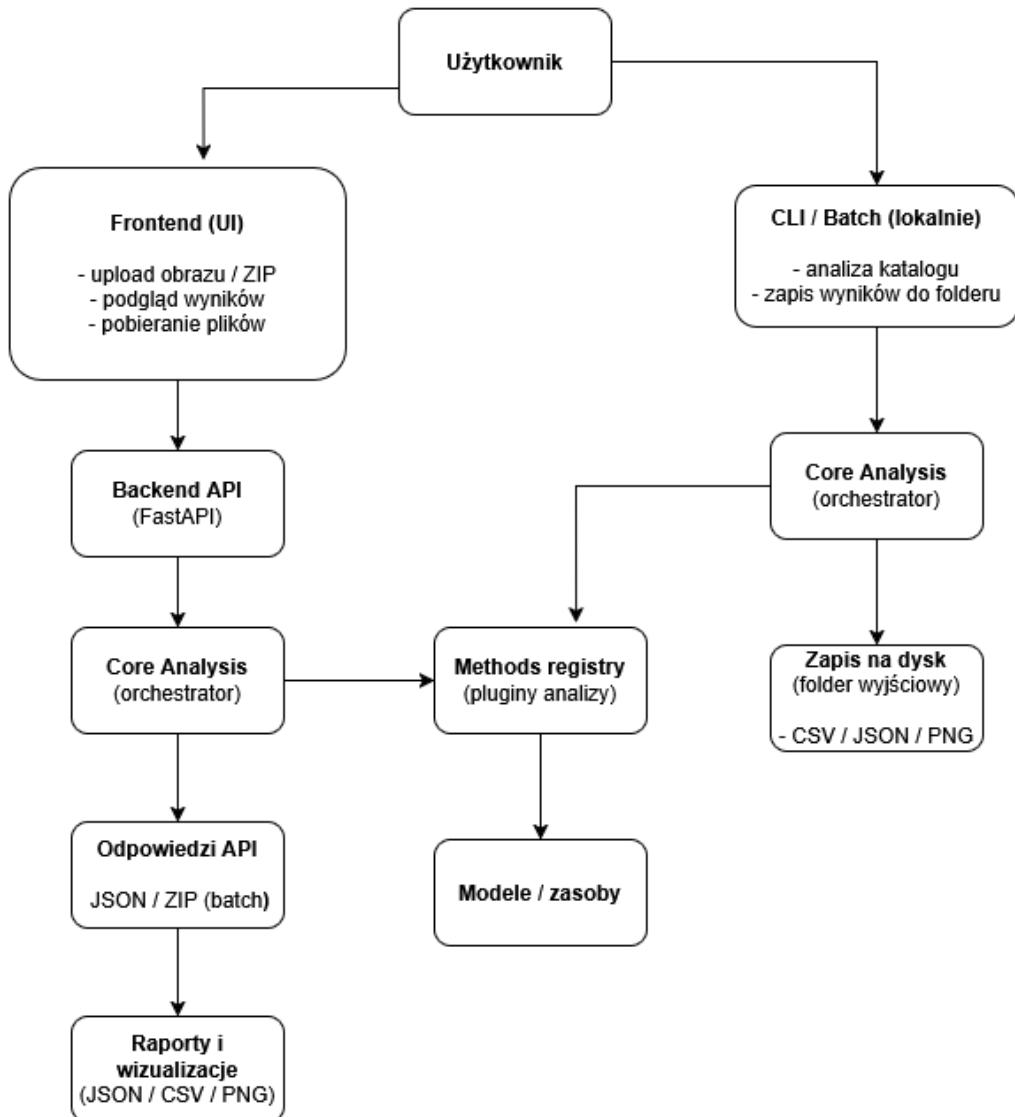
System nie jest architekturą mikroserwisową – backend stanowi jeden logiczny serwis, wewnętrznie podzielony na warstwy funkcjonalne.

3.4.2. Ogólny podział systemu

Z architektonicznego punktu widzenia system składa się z następujących głównych komponentów:

- aplikacji frontendowej (warstwa prezentacji),
- backendowego API (warstwa komunikacyjna),
- rdzenia analitycznego (warstwa aplikacyjna),
- zestawu modułów analizy obrazów,
- środowiska uruchomieniowego.

Każdy z komponentów posiada jasno określony zakres odpowiedzialności, a komunikacja pomiędzy nimi odbywa się poprzez zdefiniowane interfejsy. Schemat architektury przedstawiono na rys. 3.4.



Rysunek 3.4: Ogólna architektura systemu oraz przepływ danych pomiędzy interfejsem użytkownika, backendem API i trybem wsadowym (CLI)

3.4.3. Frontend

Frontend systemu jest aplikacją webową pełniącą rolę warstwy prezentacji. Jego głównym zadaniem jest umożliwienie użytkownikowi interakcji z systemem oraz komunikacja z backendem za pomocą protokołu HTTP.

Do głównych zadań frontendu należą:

- umożliwienie konfiguracji analizy (wybór metod, ustawienie parametrów),
- obsługa przesyłania danych wejściowych w postaci obrazu lub archiwum ZIP,
- prezentacja wyników zwróconych przez backend,
- udostępnienie użytkownikowi artefaktów wynikowych do pobrania.

Frontend nie zawiera logiki analitycznej ani algorytmów detekcji. Traktuje backend jako źródło prawdy i prezentuje wyłącznie dane zwrócone przez interfejs API.

3.4.4. Backend API

Backend API stanowi centralny punkt komunikacji systemu. Został on zrealizowany jako aplikacja serwerowa udostępniająca zestaw endpointów HTTP.

Do odpowiedzialności backendu API należą:

- obsługa żądań HTTP pochodzących z frontendu,
- walidacja danych wejściowych,
- egzekwowanie ograniczeń dotyczących rozmiaru i liczby przesyłanych plików,
- bezpieczna obsługa plików tymczasowych,
- wywoływanie rdzenia analitycznego,
- serializacja wyników analizy do postaci odpowiedzi HTTP.

Backend API jest systemem bezstanowym – każde żądanie przetwarzane jest niezależnie, bez utrzymywania sesji użytkownika.

3.4.5. Interfejs API

Backend udostępnia następujące podstawowe endpointy:

- endpoint kontrolny umożliwiający sprawdzenie dostępności systemu,
- endpoint zwracający listę dostępnych metod analizy,
- endpoint analizy pojedynczego obrazu,
- endpoint analizy wsadowej obrazów przesłanych w archiwum ZIP.

Interfejs API stanowi stabilny kontrakt pomiędzy frontendem a backendem. Zmiany w implementacji algorytmów analizy nie wymagają modyfikacji interfejsu, o ile zachowany jest format odpowiedzi.

3.4.6. Rdzeń analityczny

Rdzeń analityczny stanowi warstwę aplikacyjną systemu. Odpowiada on za sterowanie przebiegiem analizy oraz agregację wyników pochodzących z poszczególnych metod.

Zakres odpowiedzialności rdzenia obejmuje:

- interpretację konfiguracji analizy,
- uruchamianie wybranych metod analizy,
- zbieranie i normalizację wyników,
- budowę raportu wynikowego w ujednoliconej strukturze,
- obsługę analizy wsadowej.

Rdzeń analityczny jest niezależny od warstwy API oraz frontendu i może być wykorzystywany również w innych kontekstach, takich jak interfejs wiersza poleceń.

3.4.7. Moduły analizy obrazów

Algorytmy detekcji zostały wydzielone do osobnych modułów, zlokalizowanych w dedykowanym katalogu. Każda metoda analizy stanowi niezależny komponent, implementujący wspólny interfejs.

Takie podejście umożliwia:

- łatwe dodawanie nowych metod analizy,
- wymienność istniejących metod,
- rozwój i testowanie algorytmów bez wpływu na architekturę systemu.

Rdzeń analityczny komunikuje się z modułami analizy wyłącznie poprzez zdefiniowany kontrakt, bez znajomości szczegółów ich implementacji.

3.4.8. Analiza wsadowa

System umożliwia analizę wielu obrazów w trybie wsadowym. Tryb ten nie stanowi osobnego systemu, lecz jest rozszerzeniem podstawowej funkcjonalności analizy pojedynczego obrazu.

W trybie wsadowym:

- obrazy analizowane są kolejno z wykorzystaniem tego samego rdzenia analitycznego,
- wyniki zapisywane są w formacie tabelarycznym,
- generowane są zbiorcze artefakty umożliwiające dalszą analizę danych.

3.4.9. Ograniczenia i bezpieczeństwo danych wejściowych

Istotnym elementem architektury systemu jest kontrola danych wejściowych przekazywanych przez użytkownika. Ograniczenia te egzekwowane są na poziomie backendu API, co pozwala na ochronę rdzenia analitycznego przed nieprawidłowymi lub potencjalnie szkodliwymi danymi.

W systemie wprowadzono m.in.:

- ograniczenia dotyczące maksymalnego rozmiaru przesyłanych plików,
- ograniczenia liczby obrazów przetwarzanych w trybie wsadowym,
- weryfikację typu i struktury danych wejściowych,
- bezpieczną obsługę plików tymczasowych.

Takie podejście pozwala na zwiększenie stabilności systemu oraz minimalizację ryzyka błędów wynikających z niepoprawnych danych wejściowych.

3.4.10. Obsługa błędów i odporność systemu

Architektura systemu uwzględnia obsługę sytuacji wyjątkowych, takich jak błędy w trakcie przetwarzania danych, niepoprawne dane wejściowe czy błędy pochodzące z poszczególnych modułów analizy.

Błędy wykrywane są możliwie jak najbliżej źródła ich powstania. Backend API odpowiada za raportowanie błędów związanych z komunikacją i walidacją danych, natomiast rdzeń analityczny oraz moduły analizy odpowiadają za sygnalizowanie problemów występujących w trakcie przetwarzania.

Informacje o błędach przekazywane są do użytkownika w sposób ustrukturyzowany, co umożliwia jednoznaczna interpretację problemu bez ujawniania szczegółów implementacyjnych systemu.

3.4.11. Środowisko uruchomieniowe

Backend systemu został przygotowany do uruchamiania w środowisku kontenerowym. Konteneryzacja umożliwia zapewnienie spójnego środowiska wykonawczego oraz ułatwia wdrażanie systemu.

Proces uruchamiania backendu obejmuje:

- przygotowanie środowiska i zależności,
- pobranie wymaganych modeli,
- wstępne załadowanie bibliotek i zasobów,
- uruchomienie serwera API.

Zastosowanie kontenerów pozwala na oddzielenie środowiska wykonawczego aplikacji od systemu operacyjnego hosta, co umożliwia uruchamianie backendu w identycznej konfiguracji niezależnie od platformy oraz zwiększa powtarzalność i przewidywalność działania systemu.

3.4.12. Podsumowanie

Zaprojektowana architektura systemu opiera się na wyraźnym podziale odpowiedzialności pomiędzy poszczególne komponenty. Takie podejście sprzyja czytelności rozwiązania, jego rozszerzalności oraz łatwości utrzymania. System został zaprojektowany w sposób umożliwiający dalszy rozwój, w szczególności poprzez dodawanie nowych metod analizy bez konieczności modyfikacji warstwy prezentacji ani interfejsu API.

3.5. Zapewnienie jakości oprogramowania

Proces zapewniania jakości systemu miał charakter przede wszystkim praktyczny i ręczny. Projekt nie wykorzystywał zaawansowanych testów automatycznych; zamiast tego stosowano prostą, inżynierską weryfikację działania poszczególnych modułów poprzez:

- ręczne uruchamianie algorytmów na pojedynczych obrazach,
- sprawdzanie, czy uzyskane metryki są logiczne i mieścią się w oczekiwanych zakresach,
- ocenę poprawności działania klasyfikatorów na małych próbach,

- testowanie interfejsu użytkownika na kilku przykładowych scenariuszach,
- przeprowadzenie testów jednostkowych elementów backendu.

3.5.1. Weryfikacja ręczna metod analizy obrazu

Każda użyta metoda (PRNU, FFT–Slope, ELA, Grad–CAM, CLIP, CNN) była testowana osobno poprzez uruchamianie jej na kilku reprezentatywnych przykładach obrazów. Celem nie było przeprowadzenie pełnej walidacji statystycznej, lecz upewnienie się, że:

- wartości metryk mieszczą się w sensownych przedziałach,
- metoda odróżnia przynajmniej podstawowe przypadki (AI vs. real),
- implementacja nie zawiera błędów obliczeniowych (np. dzielenie przez zero, błędne wczytywanie obrazów),
- wyniki dla podobnych obrazów są powtarzalne.

Przykładowo dla PRNU ręcznie porównywano wartości PCE dla wybranych obrazów prawdziwych i syntetycznych, aby potwierdzić, że korelacja dla zdjęć realnych faktycznie jest wyższa niż dla obrazów sztucznych.

W przypadku FFT–Slope sprawdzano, czy obliczony współczynnik nachylenia widma jest ujemny (zgodne z teorią naturalnych obrazów) oraz czy obrazy syntetyczne generowane przez DCGAN dają charakterystycznie spłaszczony wykres.

3.5.2. Prosta walidacja działania klasyfikacji

Dla modułów klasyfikujących wykonywano krótkie testy polegające na przepuszczeniu kilku przykładowych obrazów i sprawdzeniu, czy wynikowa etykieta jest zgodna z intuicją.

Weryfikowano m.in.:

- czy klasyfikator prawidłowo rozpoznaje obrazy DCGAN jako AI,
- czy prawdziwe fotografie z Pexels są klasyfikowane jako real,
- czy wartości confidence nie są podejrzanie niskie (co wskazywałoby na błąd w normalizacji wejść),
- czy integracja wyników z wielu metod działa prawidłowo.

Testy tego typu pozwoliły wykryć kilka nieścisłości, np. błędne przekazywanie rozdzielczości do algorytmu ELA, które następnie poprawiono.

3.5.3. Ręczne testy działania interfejsu użytkownika

Interfejs użytkownika testowano manualnie, symulując różne typy wejść:

- ładowanie obrazów w różnych formatach (JPEG, PNG),
- przypadki brzegowe — małe obrazy, brak EXIF,

- wybór błędnej ścieżki,
- przetwarzanie wielu obrazów w kolejności.

Po każdym scenariuszu sprawdzano, czy aplikacja zachowuje się poprawnie oraz czy komunikaty błędów są zrozumiałe.

3.5.4. Testy jednostkowe i testy komponentów backendu

Oprócz ręcznej weryfikacji działania systemu, w projekcie zastosowano również testy jednostkowe obejmujące kluczowe elementy backendu. Testy te miały na celu sprawdzenie poprawności działania poszczególnych komponentów w izolacji, bez konieczności uruchamiania całego systemu.

Testy jednostkowe zostały zaimplementowane z wykorzystaniem framework'a *pytest* i obejmowały m.in.:

- testy warstwy API (poprawność odpowiedzi endpointów, obsługa błędnych danych wejściowych),
- testy rdzenia analitycznego (poprawność przepływu danych oraz integracji wyników),
- testy przetwarzania wsadowego i generowania artefaktów,
- testy modułów pomocniczych odpowiedzialnych za agregację wyników i generowanie statystyk.

W testach stosowano techniki izolacji zależności, takie jak zastępowanie rzeczywistych algorytmów i modeli obiektami testowymi (mock), co pozwalało skupić się na logice systemowej, a nie na kosztownych obliczeniach.

Testy jednostkowe umożliwiły wykrycie potencjalnych błędów na wczesnym etapie rozwoju projektu, w szczególności w obszarach związanych z obsługą danych wejściowych, przepływem informacji pomiędzy komponentami oraz serializacją wyników.

3.5.5. Podsumowanie

Proces zapewnienia jakości oprogramowania w projekcie opierał się na połączeniu testów ręcznych oraz testów jednostkowych backendu. Takie podejście było w pełni wystarczające dla projektu inżynierskiego o charakterze badawczo-aplikacyjnym.

Ręczna weryfikacja umożliwiła ocenę sensowności wyników poszczególnych metod analizy obrazu oraz poprawności działania interfejsu użytkownika, natomiast testy jednostkowe pozwoliły na systematyczne sprawdzenie poprawności działania kluczowych komponentów backendu.

Zastosowane metody zapewnienia jakości potwierdziły stabilność i poprawność działania systemu, a wykryte nieprawidłowości były eliminowane na bieżąco w trakcie procesu implementacji.

3.6. Podsumowanie rozdziału

W niniejszym rozdziale przedstawiono wybrane aspekty realizacji zaprojektowanego systemu detekcji obrazów generowanych przez modele sztucznej inteligencji. Omówiono ogólną strukturę projektu, zastosowany stos technologiczny oraz architekturę systemu, ze szczególnym uwzględnieniem podziału na frontend, backend API oraz rdzeń analityczny.

Szczegółowo zaprezentowano zaimplementowane moduły analityczne, obejmujące zarówno klasyczne metody forensyczne (EXIF, C2PA, PRNU, ELA, FFT), jak i podejścia oparte na uczeniu maszynowym i głębokim (CLIP, CNN, ConvNeXt). Przedstawiono również modele służące do identyfikacji rodzaju generatora obrazu oraz zintegrowany pipeline decyzyjny z meta-modelem agregującym wyniki poszczególnych metod.

Rozdział uzupełniono opisem mechanizmów obsługi analizy wsadowej, bezpieczeństwa danych wejściowych oraz odporności na błędy. Całość zamyka omówienie działań związanych z zapewnieniem jakości oprogramowania, w tym testów manualnych i walidacji poprawności działania systemu, potwierdzających realizację założeń projektowych.

Rozdział 4

Skalowalna wersja systemu

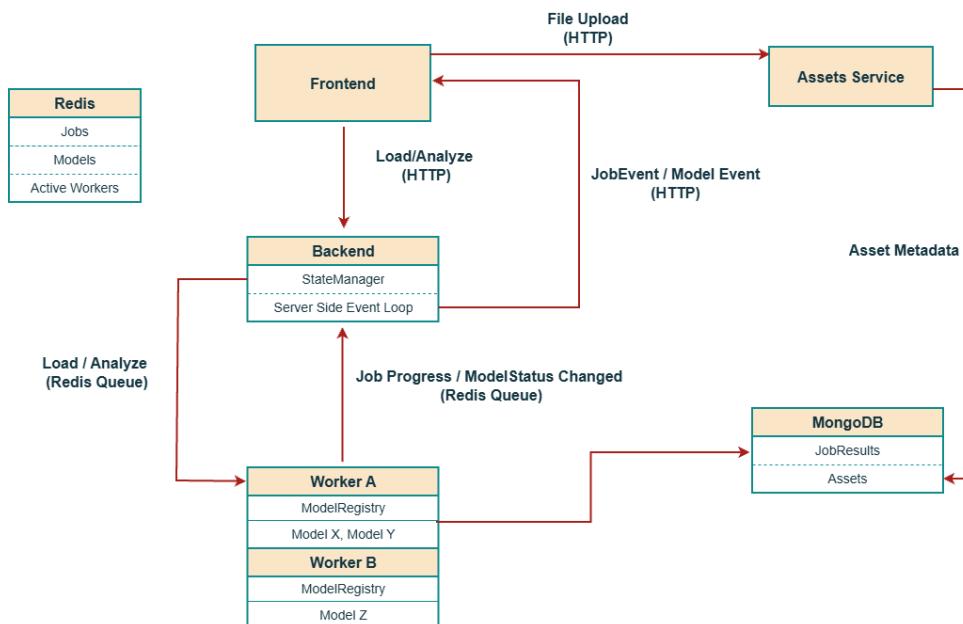
4.1. Architektura systemu

Skalowalna wersja systemu została zaprojektowana w modelu **rozproszonym (ang. distributed system)**, co umożliwia wyraźną separację odpowiedzialności pomiędzy warstwą obsługi żądań użytkownika, zarządzaniem zasobami plików oraz zasobożernymi obliczeniami związanymi z przetwarzaniem modeli sztucznej inteligencji. Architektura opiera się na asynchronicznej komunikacji i mikro-serwisowym podejściu do komponentów wykonawczych [4.1](#).

Kluczowe komponenty architektury to:

- **Backend API (FastAPI):** Centralny punkt styku dla interfejsu użytkownika, odpowiedzialny za orkiestrację zadań, zarządzanie stanem modeli oraz komunikację z bazami danych.
- **System Workerów (Asynchroniczne przetwarzanie):** Procesy wykonawcze odpowiedzialne za dynamiczne ładowanie modeli ML i przeprowadzanie właściwej analizy obrazów.
 - Każdy Worker funkcjonuje jako **osobny kontener**, co zapewnia pełną izolację środowisk uruchomieniowych.
 - **Docker pełni rolę supervisora**, zarządzając cyklem życia kontenerów. Pozwala to na łatwe skalowanie horyzontalne systemu poprzez uruchamianie kolejnych instancji workerów w zależności od zapotrzebowania na moc obliczeniową (np. dostępność jednostek GPU).
- **Assets Service:** Dedykowany serwis do zarządzania plikami wejściowymi (zdjęciami i paczkami ZIP), który odciąża główny backend od transferu ciężkich danych binarnych.
- **Redis (Message Broker & Real-time State):** Służy jako centralna magistrala komunikacyjna. Pełni funkcję kolejki zadań (ang. Task Queue) oraz dynamicznego rejestru stanu całego klastra, przechowując informacje o aktywnych workerach, przypisaniach modeli oraz trwających analizach.
- **MongoDB (Persistence Layer):** Trwały magazyn danych, w którym składowane są finalne wyniki analiz oraz metadane biblioteki danych binarnych.
- **Frontend (Vue.js):** Reaktywna aplikacja typu SPA (ang. Single Page Application), która w czasie rzeczywistym wizualizuje stan systemu, postęp prac oraz wyniki analiz.

Dzięki takiemu podejściu, system charakteryzuje się wysoką responsywnością – nawet długotrwała analiza obrazu przeprowadzana przez workera nie blokuje interfejsu użytkownika ani nie uniemożliwia przyjmowania nowych zleceń przez serwer API. Konteneryzacja przy użyciu Dockera zapewnia natomiast powtarzalność środowiska i ułatwia wdrożenie systemu na zróżnicowanych maszynach obliczeniowych.



Rysunek 4.1: Ogólna architektura systemu.

4.2. System zadań i dynamiczne zarządzanie stanem modeli

Kluczową cechą tej wersji systemu jest zdolność do inteligentnego zarządzania zasobami obliczeniowymi poprzez mechanizm równoważenia obciążenia oraz asynchroniczną orkiestrację zadań. Pozwala to na optymalizację zużycia pamięci RAM i VRAM, co jest krytyczne w środowiskach z ograniczonymi zasobami sprzętowymi.

4.2.1. Modelowanie stanów i cykl życia zadania

System rozróżnia dwa główne byty podlegające monitorowaniu: **Zadanie (Job)** oraz **Status Modelu (Model Status)**. Cykl życia zadania opisany jest przez zbiór stanów: CREATED, QUEUED, RUNNING, FINISHED oraz FAILED.

Równolegle, system śledzi stan konkretnych instancji modeli na workerach. Model może znajdować się w stanie:

- QUEUED: Model został przypisany do workera i oczekuje na inicjalizację.
- LOADING: Trwa proces ładowania wag modelu do pamięci.
- LOADED: Model jest gotowy do natychmiastowej inferencji.
- ERROR: Wystąpił błąd podczas próby uruchomienia modelu.

4.2.2. Orkiestracja i StateManager

Za logikę przypisywania zadań odpowiada komponent StateManager znajdujący się w warstwie backendu. Realizuje on strategię **równoważenia obciążenia** (ang. *load balancing*), wybiegając dla nowego modelu workera o najniższym aktualnym obciążeniu.

Proces orkiestracji przebiega w następujący sposób:

1. **Rezerwacja:** StateManager atomowo rezerwuje miejsce dla modelu w rejestrze Redis, przypisując go do konkretnego identyfikatora `worker_id`.
2. **Dysponowanie:** Zadanie ładowania (`load`) lub analizy (`analyze`) jest przesyłane do dedykowanej kolejki danego workera.
3. **Egzekucja:**
 - `load`: ModelRegistry dynamicznie instancjonuje interfejs modelu i ładuje go do pamięci.
 - `analyze`: Worker zarządza odczytem zdjęć z systemu plików, przekazując je sekwencyjnie do metody `analyze()` modelu.

4.2.3. Klasyfikacja modeli: Heavy vs Utility

System wprowadza podział modeli ze względu na ich zapotrzebowanie na zasoby, co definiowane jest w manifeście za pomocą pola `requires_load`.

- **Modele lekkie (Utility):** Są ładowane automatycznie podczas startu workera. Pełnią role pomocnicze i są zawsze dostępne.
- **Modele ciężkie (Heavy):** Są ładowane dopiero w momencie wysłania pierwszej komendy `load`. System pozwala również na ich ręczne zwolnienie z pamięci, co uwalnia zasoby dla innych detektorów.

Dzięki takiemu podejściu możliwe jest utrzymywanie szerokiej biblioteki detektorów przy jednoczesnym ograniczeniu liczby modeli aktywnie rezydujących w pamięci GPU do tych, które są w danej chwili faktycznie wykorzystywane.

4.3. Manifest modelu jako fundament uniwersalnego interfejsu

Jednym z najważniejszych założeń projektowych systemu jest jego **agnostyczność względem konkretnych implementacji modeli ML**. Cel ten został osiągnięty poprzez wprowadzenie ustandaryzowanego pliku definicji — `ModelManifest`. Pełni on rolę tzw. „źródła prawdy” (ang. *Source of Truth*), opisując nie tylko metadane modelu, ale również jego interfejs wejściowy i wyjściowy.

4.3.1. Struktura definicji parametrów

Klasa `ModelManifest`, zaimplementowana z wykorzystaniem biblioteki `Pydantic`, wymusza ścisłą strukturę opisującą możliwości modelu. Kluczowym elementem są parametry (`ModelProperty`), które zawierają:

- **Metadane techniczne:** nazwa, typ danych (`int`, `float`, `bool`) oraz wartości domyślne.

- **Definicję UI:** obiekt ParameterUI określa, jak dany parametr ma być renderowany na froncie (np. jako suwak slider lub przełącznik checkbox).
- **Reguły validacji:** zakresy wartości (min, max) oraz krok suwaka, co pozwala na wstępna weryfikację poprawności danych już na poziomie interfejsu.

4.3.2. Dynamiczne renderowanie widoku (Dynamic UI)

Dzięki takiemu podejściu, warstwa frontendowa (ModelView.vue) nie posiada zakodowanych na sztywno formularzy dla konkretnych modeli. Zamiast tego, aplikacja wykonuje proces **dynamicznej introspekcji**:

1. Frontend pobiera manifest wybranego modelu z API.
2. Na podstawie listy parametrów, przy użyciu dyrektyw sterujących (np. v-for), generowane są odpowiednie komponenty interfejsu użytkownika.
3. Struktura tabeli wyników jest budowana dynamicznie w oparciu o pole table_columns zdefiniowane w manifeście.

4.3.3. Zalety separacji logicznej

Zastosowanie manifestu przynosi wymierne korzyści architektoniczne:

- **Łatwość rozszerzania:** Dodanie nowego modelu nie wymaga modyfikacji kodu frontendu ani backendu — wystarczy umieścić nowy folder z plikiem manifest.json w katalogu modeli.
- **Spójność danych:** Gwarancja, że parametry przesłane przez użytkownika są zawsze zgodne z oczekiwaniami algorytmu detekcyjnego.
- **Personalizacja raportów:** Każdy model może definiować własny tytuł sekcji szczegółowych (key_value_title) oraz artefaktów wizualnych (artifacts_title), co pozwala na lepsze dostosowanie prezentacji wyników do specyfiki danego problemu.

4.4. Struktura wyników analizy i zarządzanie artefaktami

System wprowadza ustandaryzowany protokół przekazywania wyników z modeli uczenia maszynowego do warstwy prezentacji. Pozwala to na zachowanie spójności interfejsu przy jednoczesnym wsparciu dla bardzo zróżnicowanych danych wyjściowych, takich jak werdykty klasifikasiacyjne, mapy ciepła (np. Grad-CAM) czy szczegółowe parametry techniczne obrazu.

4.4.1. Standard wyjściowy AnalyzerOutput

Każdy model zaimplementowany w systemie musi zwracać dane w formacie klasy AnalyzerOutput. Struktura ta pełni rolę interfejsu pośredniczącego i składa się z trzech klużowych komponentów:

- **table_row:** Słownik zawierający dane przeznaczone do głównej tabeli wyników (np. wynik punktowy, decyzja binarna).

- **kv (Key-Value):** Zbiór dodatkowych metadanych wyświetlanych w widoku szczegółowym, które nie wymagają osobnej kolumny w tabeli.
- **artifacts:** Kolekcja obiektów obrazów (instancji klasy PIL Image), generowanych podczas inferencji, takich jak wizualizacje warstw neuronowych czy maski segmentacji.

4.4.2. Proces utrwalania wyników i artefaktów

Worker po zakończeniu analizy każdego pliku przeprowadza proces transformacji danych surowych w trwałe zasoby:

1. **Zapis plików:** Obrazy z kolekcji artifacts są zapisywane na dysku serwera w dedykowanym folderze zadania (RESULTS_DIR/job_id). Każdy plik otrzymuje unikalną nazwę łączącą etykietę artefaktu z nazwą pliku źródłowego.
2. **Mapowanie ścieżek:** Fizyczne ścieżki do plików są konwertowane na publiczne adresy URL (static_path), co umożliwia ich bezpośrednie serwowanie do frontendu.
3. **Agregacja JobResult:** Wszystkie wyniki częściowe są składane w obiekt JobResult, który zawiera zbiorczą tabelę oraz słownik danych dodatkowych dla każdego pliku (FileResultDetail).

Finalny obiekt JobResult jest zapisywany w bazie **MongoDB**. Warstwa prezentacyjna (ModelView.vue) wykorzystuje te dane do budowy wielopoziomowego raportu:

- **Tabela zbiorcza:** Wykorzystuje dane z table_row do wyświetlenia porównawczego zestawienia wszystkich przeanalizowanych plików.
- **Modal szczegółów:** Po wybraniu konkretnego wiersza, użytkownik uzyskuje dostęp do oryginalnego zdjęcia, sekcji metadanych Key-Value oraz galerii wygenerowanych artefaktów wizualnych.

Dzięki tak zaprojektowanej strukturze, system nie tylko informuje o wyniku analizy, ale również dostarcza wizualnych dowodów (ang. explainability) na to, dlaczego model podjął określoną decyzję, co jest kluczowe w systemach detekcji manipulacji obrazem.

4.5. System zarządzania zasobami (Assets Service)

System został wyposażony w dedykowany moduł Assets Service, pracujący jako niezależna usługa. Jego głównym zadaniem jest zarządzanie cyklem życia plików wejściowych, ich składowanie oraz przygotowanie do procesów analizy.

4.5.1. Separacja obsługi danych binarnych

Wprowadzenie osobnej usługi dla zasobów (ang. assets) wynika z chęci uniknięcia blokowania głównego serwisu API (Backend API) przez długotrwałe operacje wejścia/wyjścia (I/O). Dzięki temu:

- **Backend API** zajmuje się wyłącznie lekką orkiestracją i logiką stanów.
- **Assets Service** odpowiada za obsługę strumieniowego przesyłania plików graficznych oraz archiwów.

4.5.2. Asynchroniczne przetwarzanie paczek ZIP

System oferuje mechanizm przesyłania hurtowego danych za pomocą formatu ZIP. Proces ten jest realizowany w sposób asynchroniczny, aby nie zamrażać interfejsu użytkownika:

1. **Upload:** Użytkownik przesyła plik ZIP do /assets/upload.
2. **Background Task:** Serwis natychmiast zwraca identyfikator asset_id ze statusem accepted, podczas gdy w tle uruchamiane jest zadanie process_zip_task.
3. **Ekstrakcja i walidacja:** Usługa rozpakowuje archiwum, stosując zabezpieczenia przed atakami typu Zip Slip (weryfikacja ścieżek docelowych), zlicza pliki i aktualizuje metadane w bazie MongoDB.
4. **Ready State:** Po pomyślnym przetworzeniu, status zasobu zmienia się na ready, co sygnalizuje systemowi gotowość do przeprowadzenia analizy.

4.5.3. Biblioteka zasobów

W odróżnieniu od klasycznego podejścia, gdzie zdjęcia są przesyłane każdorazowo do analizy, projekt wprowadza koncepcję **trwałej biblioteki zasobów**. Rozwiązanie to niesie ze sobą szereg zalet:

- **Efektywność transferu:** Użytkownik wgrywa dany zestaw zdjęć tylko raz, a następnie może go wielokrotnie poddawać analizie przy użyciu różnych modeli detekcyjnych.
- **Abstrakcja źródła:** Modele w procesie analizy nie operują bezpośrednio na plikach przesyłanych przez sieć, lecz na obiektach FileLink wskazujących na fizyczne ścieżki w systemie plików współdzielonym z workerami.
- **Organizacja danych:** Każdy zasób posiada własną nazwę wyświetlaną (display_name) oraz znacznik czasu, co ułatwia zarządzanie dużymi zbiorami danych testowych.

4.5.4. Integracja z warstwą prezentacji

Warstwa frontendowa (AssetsView.vue) komunikuje się bezpośrednio z usługą zasobów, umożliwiając monitorowanie postępu przetwarzania w czasie rzeczywistym. Dzięki wykorzystaniu magazynu stanów AssetStore (opartego na bibliotece **Pinia**), informacje o dostępnych zasobach są synchronizowane między różnymi widokami aplikacji, co pozwala na wygodne wybieranie gotowych zestawów danych z poziomu formularza analizy modelu.

4.6. Podsumowanie architektury i logiki systemu

Przedstawiona architektura stanowi spójne środowisko do przeprowadzania analizy obrazów, oparte na wyraźnym rozdzieleniu ról pomiędzy poszczególnymi modułami. Centralnym założeniem projektu jest maksymalna izolacja procesów oraz standaryzacja komunikacji, co przekłada się na stabilność i przewidywalność działania całego systemu.

Kluczowe aspekty przedstawionego rozwiązania to:

- **Separacja odpowiedzialności:** Podział na Backend API, Assets Service oraz niezależne procesy wykonawcze (Workery) pozwala na optymalizację operacji wejścia/wyjścia i zapobiega blokowaniu interfejsu użytkownika podczas transferu dużych zbiorów danych.
- **Agnostyczność i uniwersalność:** Dzięki zastosowaniu mechanizmu *ModelManifest*, system staje się platformą niezależną od konkretnych implementacji algorytmów. Pozwala to na dołączanie nowych modeli bez ingerencji w kod źródłowy frontendu czy backendu.
- **Dynamiczne zarządzanie stanem:** Wykorzystanie brokera Redis do orkiestracji zadań zapewnia pełną kontrolę nad cyklem życia analizy oraz pozwala na bieżąco monitorować dostępność i stan poszczególnych modeli w pamięci.
- **Integralność danych i wyników:** Standaryzacja formatów *AnalyzerOutput* oraz *JobResult* gwarantuje spójną prezentację danych dla użytkownika końcowego, niezależnie od specyfiki wybranego detektora.

Tak zaprojektowana struktura nie tylko ułatwia bieżące zarządzanie procesami analizy, ale również tworzy solidne fundamenty pod dalszy rozwój bazy modeli przy zachowaniu jednolitych standardów UX i bezpieczeństwa danych.

Rozdział 5

Organizacja pracy

5.1. Charakterystyka projektu i sposób jego realizacji

Projekt miał charakter badawczo–rozwojowy, łącząc tworzenie aplikacji z badaniem metod detekcji obrazów generowanych przez sztuczną inteligencję. W części badawczej analizowano skuteczność klasycznych technik forensycznych, analizy metadanych oraz nowoczesnych modeli uczenia maszynowego, takich jak CLIP i ConvNeXt. Część rozwojowa obejmowała implementację zintegrowanej aplikacji umożliwiającej korzystanie z wybranych metod, generowanie raportów i wizualizację wyników, co pozwoliło na praktyczne testowanie i zastosowanie opracowanych rozwiązań.

Na początku prac określono ogólne wymagania dotyczące działania aplikacji, w szczególności konieczność stworzenia narzędzia umożliwiającego analizę autentyczności obrazów, integrację różnych metod detekcji oraz generowanie raportów z wynikami analizy. Te założenia stanowiły stabilną podstawę projektu.

Znaczna część szczegółowych decyzji, w szczególności dotyczących doboru konkretnych modeli detekcji oraz sposobu ich integracji, powstawała jednak stopniowo w trakcie realizacji. Wynikało to z faktu, że w zakresie detekcji obrazów generowanych istnieje wiele konkurencyjnych podejść, obejmujących zarówno klasyczne metody analizujące artefakty obrazowe, jak i nowoczesne modele uczenia maszynowego. W trakcie prac konieczne było przetestowanie kilku z nich oraz określenie, które najlepiej spełnią założenia projektu, co prowadziło do stopniowego doprecyzowywania szczegółów i iteracyjnych zmian w planie.

Proces realizacji projektu miał charakter przyrostowo–iteracyjny, poprzedzony krótką fazą eksploracyjną. Na początku powstał prototyp aplikacji wykorzystujący metody detekcji ELA i FFT, co pozwoliło zweryfikować podstawową koncepcję. Następnie prace skupiły się na treningu i testowaniu wybranych modeli oraz poszukiwaniu najbardziej obiecujących podejść detekcyjnych. Ostatni etap obejmował integrację sprawdzonych rozwiązań z backendem oraz frontendem, a także dopracowanie architektury aplikacji jako kompletnego systemu.

Taki sposób pracy umożliwił elastyczne zarządzanie wymaganiami oraz efektywne dostosowywanie zakresu projektu do wyników bieżących testów. Pozwolił także na ograniczenie ryzyka poprzez szybkie sprawdzanie hipotez technicznych oraz ocenę opłacalności poszczególnych rozwiązań.

5.2. Osoby w projekcie

W projekcie uczestniczył trzyosobowy zespół deweloperski oraz promotor pełniący rolę klienta i konsultanta merytorycznego.

Zespół realizował wszystkie etapy projektu, począwszy od fazy prototypowania, przez eksperymenty z różnymi metodami detekcji, aż po integrację funkcjonalności backendu i frontendu. Każdy członek zespołu wnosił swoje kompetencje i wspólnie podejmował decyzje dotyczące implementacji. Zadania były przydzielane w zależności od bieżących potrzeb projektu.

Promotor, pełniąc rolę klienta, zdefiniował wymagania, a następnie nakierowywał zespół na priorytetowe funkcjonalności oraz dostarczał bieżące wskazówki, co pozwalało skutecznie kształtować kierunek rozwoju aplikacji i dostosowywać rozwiązania do wymagań.

5.3. Zespół i podział obowiązków

Poniżej przedstawiono szczegółowy podział obowiązków w zespole, z wyszczególnieniem głównych zadań realizowanych przez poszczególnych członków projektu.

- **Marta Bukowińska**

- Implementacja modelu CNN do detekcji obrazów StyleGAN.
- Implementacja modelu ConvNeXt do detekcji typu generatora obrazu (GAN vs Diffusion).
- Implementacja modelu ConvNeXt do detekcji generatora obrazu (modele Diffusion).
- Implementacja modelu opartego o CLIP do detekcji generatora obrazu (nowoczesne modele Diffusion).
- Opracowanie testów metod detekcji generatora.
- Odpowiedzialność za część dokumentacji projektu.
- Organizacja spotkań i pracy zespołu.

- **Olaf Fertig**

- Implementacja skalowalnej wersji backendu aplikacji.
- Implementacja frontendu aplikacji.
- Dokumentacja wersji skalowalnej systemu (rozdział 4).
- Implementacja modelu sieci CNN ConvNeXt do detekcji obrazów StyleGAN .
- Implementacja modelu opartego o CLIP i Liniowy SVM do detekcji obrazów syntetycznych.

- **Wojciech Fortuna**

- Implementacja pełnofunkcjonalnej wersji backendu systemu.
- Implementacja pełnofunkcjonalnej wersji frontendu aplikacji.
- Implementacja metody analizy błędu kompresji JPEG (ELA).
- Implementacja metody analizy widmowej FFT–Bands.
- Implementacja metody detekcji PRNU (ekstrakcja i dopasowanie sygnału).

- Implementacja metody detekcji opartej na modelu CLIP w trybie zero-shot.
- Implementacja modułu analizy danych EXIF.
- Implementacja modułu weryfikacji podpisów C2PA.
- Implementacja oraz wytrenowanie modelu DCGAN.
- Stworzenie pipeline'ów decyzyjnych integrujących wyniki wielu metod w jedną końcową klasyfikację AI/REAL, wraz z implementacją meta-modelu decyzyjnego.
- Opracowanie skryptów do generowania oraz pobierania danych testowych (obrazy AI tworzone w różnych technologiach oraz zdjęcia rzeczywiste).
- Implementacja testów jednostkowych dla backendu.
- Odpowiedzialność za część dokumentacji projektowej.

5.4. Organizacja prac i wykorzystane narzędzia

W realizacji projektu wykorzystano szereg narzędzi wspierających organizację pracy, zarządzanie zadaniami oraz komunikację w zespole. Poniżej przedstawiono zestawienie najważniejszych aplikacji i platform, które umożliwiły efektywną współpracę.

- **Messenger** - wykorzystywany do szybkiej komunikacji w zespole, organizacji terminów spotkań oraz wymiany krótkich informacji dotyczących bieżących zadań.
- **Discord** - platforma używana do organizowania spotkań zespołu, prezentacji realizowanych przez każdego członka etapów prac i dyskusji na ich temat, a także do uporządkowania ważnych informacji i przydatnych materiałów.
- **Microsoft Teams** - narzędzie wykorzystane do komunikacji z promotorem pracy.
- **Github** - system kontroli wersji kodu źródłowego, służący do zarządzania repozytorium projektu, śledzenia zmian oraz integracji poszczególnych modułów aplikacji.

5.5. Przebieg prac

W trakcie realizacji projektu można wyodrębnić kilka kluczowych etapów, które obejmowały fazę przygotowawczą, eksperymenty badawcze, implementację oraz testowanie i integrację poszczególnych komponentów systemu. Opisano je dokładniej poniżej:

- **Faza eksploracyjna i prototypowanie** – przygotowanie wstępnego prototypu z prostymi metodami forensycznymi (ELA i FFT), generacją raportów i wyników w formie graficznej; testowanie bibliotek i frameworków, określenie ograniczeń i priorytetów zdefiniowanych wymagań.
- **Dobór modeli i trening** – poszukiwanie zbiorów danych do treningu modeli; eksperymenty z różnymi modelami CNN i CLIP; wybór metod do implementacji i dostosowanie parametrów.
- **Tworzenie aplikacji** – implementacja logiki aplikacji (backend) i interfejsu użytkownika (frontend), z uwzględnieniem zarządzania pamięcią GPU oraz mechanizmów dynamicznego ładowania modeli

- **Integracja systemu** – integracja wybranych metod detekcji z aplikacją i implementacja meta-modelu.
- **Testy i optymalizacja** – sprawdzenie działania aplikacji w trybie interaktywnym i wsadowym, optymalizacja wydajności oraz stabilności działania.

5.6. Podsumowanie rozdziału

W rozdziale przedstawiono strukturę zespołu projektowego oraz rolę promotora pełniącego funkcję klienta i konsultanta merytorycznego. Omówiono przyjęty sposób realizacji projektu, który łączył fazę eksploracyjną z iteracyjnym i przyrostowym podejściem do implementacji aplikacji. Przedstawiono również główne etapy przebiegu prac, obejmujące kolejno planowanie, analizę rozwiązań, projektowanie oraz implementację poszczególnych funkcjonalności. Wyróżniono także narzędzia wspierające współpracę i organizację pracy zespołu.

Rozdział 6

Wyniki projektu

6.1. Podsumowanie zrealizowanych funkcji finalnego produktu

Niniejszy projekt zakończył się opracowaniem kompletnego systemu służącego do detekcji obrazów generowanych przez sztuczną inteligencję. Finalna wersja oprogramowania obejmuje zarówno metody analizy sygnałowej, jak i semantycznej oraz interfejs użytkownika, pozwalający na wygodne wykorzystanie całości funkcjonalności. Poniżej przedstawiono zestaw funkcji, które zostały w pełni zrealizowane w finalnym produkcie.

Funkcje związane z analizą obrazu

- Obsługa plików graficznych w formatach JPEG, PNG oraz BMP, wraz z automatycznym sprawdzaniem integralności obrazu i wczytywaniem metadanych, jeśli są dostępne.
- Wyznaczanie sygnału PRNU i obliczanie miary dopasowania PCE, umożliwiającej detekcję śladu sensora charakterystycznego dla zdjęć wykonanych prawdziwą kamerą.
- Analiza widmowa FFT i wyznaczanie cechy *slope*, pozwalającej rozróżnić naturalny rozkład energii częstotliwości w zdjęciach *REAL* od widma obrazów generowanych przez modele AI.
- Analiza błędu kompresji JPEG (ELA), wykorzystująca lokalne różnice w błędzie rekonstrukcji do identyfikacji nienaturalnych struktur typowych dla obrazów syntetycznych.
- Analiza semantyczna oparta o model CLIP, umożliwiająca określenie stopnia podobieństwa obrazu do klas *REAL* oraz *AI-generated* w przestrzeni wektorowej osadzeń kontrastywnych.
- Analiza przy użyciu lekkich modeli CNN oraz ConvNeXt, które pozwalają odróżnić obrazy rzeczywiste od syntetycznych, wykorzystując automatycznie wyekstrahowane cechy wizualne i subtelne artefakty generacyjne.
- Identyfikacja potencjalnego modelu lub rodzaju generatora obrazu, realizowana przy użyciu dedykowanych modeli ConvNeXt oraz CLIP, umożliwiająca przypisanie obrazu do określonej rodziny generatorów (GAN lub modele dyfuzyjne) lub konkretnego modelu generatywnego.

Dostępne funkcje analizy, możliwość konfiguracji progu i przesyłania pliku prezentuje strona główna systemu [6.1](#).

Analiza metadanych i zabezpieczeń

- Ekstrakcja i ocena struktury EXIF, pozwalająca zweryfikować spójność i autentyczność metadanych aparatu, ekspozycji, obiektywu oraz parametrów technicznych.
- Obsługa oraz weryfikacja podpisów C2PA, jeśli są obecne w obrazie, co umożliwia potwierdzenie integralności pliku i identyfikację urządzenia bądź oprogramowania, które go wygenerowało.

Analizę metadanych prezentuje widok [6.2](#).

Funkcje klasyfikacyjne

- Meta–klasyfikator integrujący wyniki kilku metod detekcji (PRNU, FFT, ELA, CLIP, EXIF, C2PA) i generujący końcową decyzję: *REAL* lub *AI-generated*, wraz z możliwością analizy wkładu poszczególnych komponentów w decyzję.
- Wyznaczanie prawdopodobieństwa syntetyczności obrazu, co pozwala użytkownikowi zrozumieć “siłę” decyzji algorytmu (jak bardzo obraz przypomina dane AI w przestrzeni cech). Prezentację wyniku przedstawia widok [6.5](#).
- Pipeline łączony w pełnej i szybkiej wersji, który w pierwszej kolejności sprawdza metadane i sygnały kamery (np. C2PA, PRNU), a następnie, jeśli brak jednoznacznego wyniku, uruchamia pozostałe detektory, a ich wyniki są przekazywane do meta-modelu. Pipeline działa w dwóch wariantach: pełnym i szybkiej wersji.

Funkcje interfejsu użytkownika

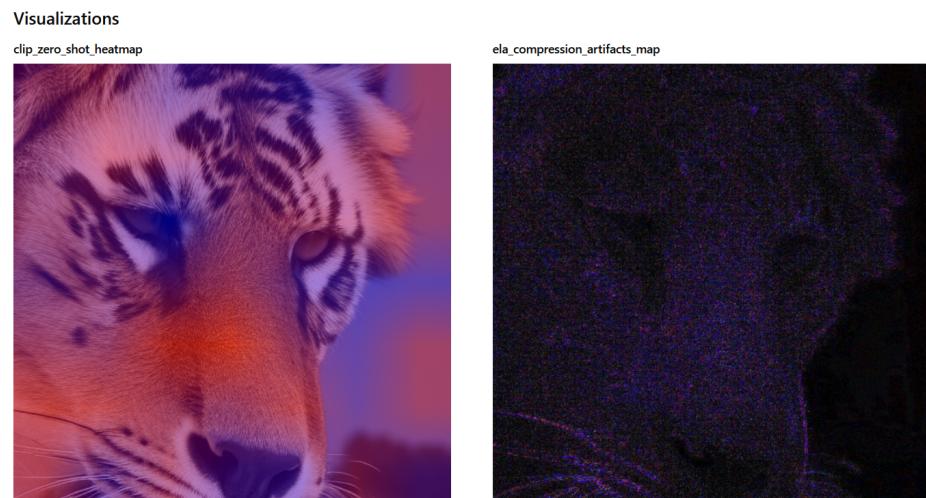
- Możliwość analizy pojedynczego obrazu z prezentacją wszystkich wyników cząstkowych oraz decyzji końcowej w jednej, spójnej karcie wyników.
- Tryb wsadowy (batch mode) - automatyczna analiza wielu obrazów jednocześnie oraz generowanie raportu zbiorczego z wynikami klasyfikacji, co pokazuje widok [6.9](#).
- Wizualizacja w formie map cieplnych, wskazujących obszary obrazu mające największy wpływ na ocenę różnych modeli detekcji (CLIP, CNN, ConvNeXt) oraz mapy charakterystyk forensycznych, takich jak ELA czy widmo FFT, co ułatwia interpretację wyników poszczególnych metod. Prezentują to widoki [6.3](#), [6.4](#), [6.6](#).
- Wyświetlanie szczegółowych metryk z analizy, zawierających m.in.: wartość PCE, nachylenie widma FFT, statystyki ELA, ocenę wiarygodności metadanych oraz prawdopodobieństwa klasyfikacji generowanych przez modele ML (np. CLIP, CNN, ConvNeXt). Część z metryk prezentuje widok [6.7](#).

The screenshot shows the 'Settings' section with an 'AI verdict threshold' slider set at 0.50. Below it is a list of 'Verification methods' with 'combined_methods' checked. A note below says 'How does it work?' followed by a bullet point about 'attrib_generator (AI generator family recognition)'. On the right, there's a 'Drag and drop file here' input field and a 'Browse files' button.

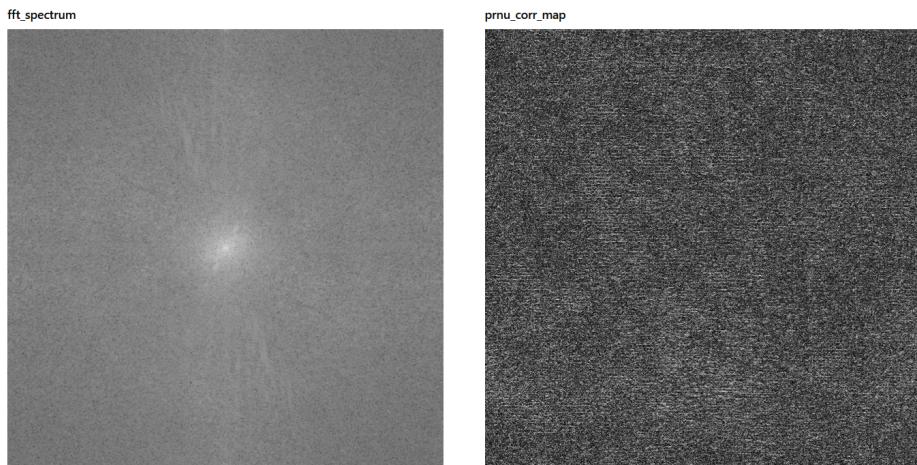
Rysunek 6.1: Główny strona systemu.

The screenshot shows the 'Original' image of a tiger's face. To its right are sections for 'EXIF (selected)' (empty), 'C2PA manifest' (empty), and 'Visualizations' which include a 'clip_zero_shot_heatmap' showing a heatmap of the tiger's face and an 'ela_compression_artifacts_map' showing a dark map with purple artifacts.

Rysunek 6.2: Widok obrazu i analizy metadanych w systemie.



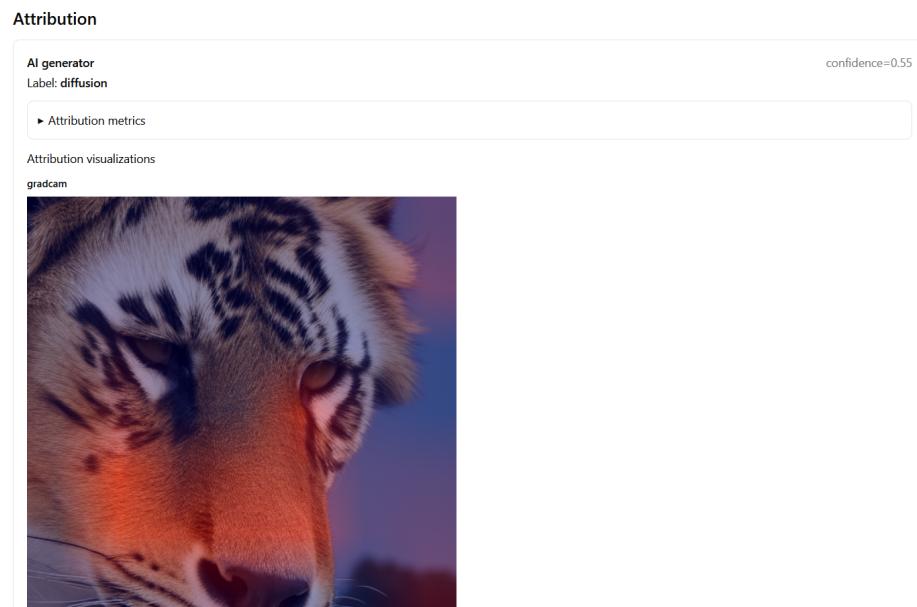
Rysunek 6.3: Widok wizualizacji mapy cieplnej CLIP i mapy ELA.



Rysunek 6.4: Widok wizualizacji metod FFT i PRNU.



Rysunek 6.5: Widok prezentacji wyniku klasyfikacji REAL vs AI.



Rysunek 6.6: Prezentacja wyniku i wizualizacji klasyfikacji rodziny modelu Diffusion vs GAN.

▼ Metric details

```
{
  "prnu_pce": 24.061892660542043,
  "prnu_peak": 0.005234093408586339,
  "prnu_noise_energy": 0.0000011385527382223043,
  "prnu_cov": 0.9679698944091797,
  "prnu_xstd": 0.0014495980720439553,
  "prnu_gate_cov_failed": 0,
  "prnu_gate_xstd_failed": 0,
  "prnu_peak_offset_r": -98,
  "prnu_peak_offset_c": -70,
  "prnu_peak_offset_norm": 120.43255373859678,
  "prnu_mask_ref_coverage": 1,
  "prnu_mask_test_coverage": 0.9679698944091797,
  "prnu_res_mean": 1.8505278376323986e-7,
  "prnu_res_std": 0.0014619490830227733,
  "prnu_best_fingerprint_name": "Lenovo_TabE7_1280x1600_1024_sea_advHF_wavelet",
  "prnu_native_hw_used": [
    1280,
    1600
  ],
  "prnu_crop_size": 1024,
  "prnu_pce_threshold": 66.5824,
  "prnu_wavelet": "db8",
}
```

Rysunek 6.7: Widok szczegółowych metryk.

Downloads

[Download report \(JSON\)](#)

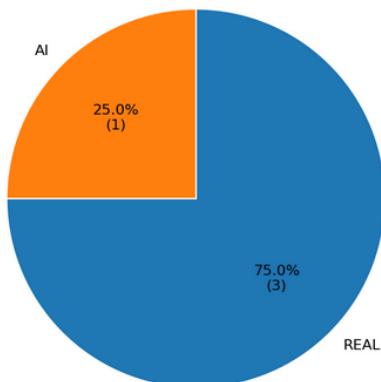
[Download visualizations \(ZIP\)](#)

Rysunek 6.8: Widok opcji pobrania raportu i wizualizacji.

Batch summary

REAL: 3 • AI: 1 • UNKNOWN: 0

Batch verdicts (n=4)



Downloads

[Download results \(CSV\)](#)

[Download summary \(JSON\)](#)

[Download chart \(PNG\)](#)

Rysunek 6.9: Widok podsumowania analizy w trybie wsadowym.

Dodatkowe komponenty zrealizowane w ramach projektu

- Skrypty do generowania i pobierania danych treningowych AI (m.in. DCGAN, BigGAN, Stable Diffusion) oraz pobierania zdjęć prawdziwych z serwisów takich jak Pexels, a następnie ich przygotowania do kalibracji progów detekcji.
- Procedury testowe i walidacyjne, obejmujące ręczne i automatyczne sprawdzanie jakości działania poszczególnych modułów oraz poprawności przepływu danych w całym pipeline.
- Moduły eksportu wyników, umożliwiające zapis raportu z analizy w formatach tekstowych (np. JSON) oraz dalsze wykorzystanie rezultatów w narzędziach zewnętrznych. Opcje dostępnych do pobrania raportów w trybach interaktywnym i wsadowym prezentują widoki [6.8](#), [6.9](#).

Całość stanowi kompletny system detekcji obrazów AI, integrujący wiele metod z różnych dziedzin - od przetwarzania sygnałów, przez analizę widmową i kompresyjną, aż po nowoczesne modele semantyczne i kryptograficzne zabezpieczenia treści.

6.2. Główne scenariusze działania

W tej sekcji przedstawiono typowe scenariusze korzystania z aplikacji. Opisano zarówno pracę w interfejsie graficznym (analiza pojedynczych obrazów i plików ZIP), jak i analizę wsadową uruchamianą z linii poleceń.

Scenariusz 1: Jednorazowa analiza pojedynczego obrazu

Najbardziej naturalnym sposobem korzystania z systemu jest analiza pojedynczego obrazu w interfejsie webowym.

1. Użytkownik uruchamia backend i frontend poleceniami:

```
docker compose up
npm run dev
```

Po uruchomieniu interfejs użytkownika dostępny jest pod adresem: <http://localhost:5173>.

2. W panelu bocznym wybiera:

- próg decyzji *AI verdict threshold* (domyślnie 0,5),
- metody, które mają zostać użyte (np. ela, fft, c2pa, lub combined_methods).

3. W sekcji *Upload an image* użytkownik wgrywa plik (JPG/JPEG/PNG/BMP).

4. Funkcja `analyze_image()`:

- zapisuje obraz tymczasowo na dysku,
- wczytuje go oraz ekstrahuje pola EXIF,

- uruchamia wybrane metody detekcji zarejestrowane w REGISTRY.
5. W górnej części UI prezentowane są:
- po lewej: oryginalny obraz (skalowany z zachowaniem proporcji),
 - po prawej: kluczowe pola EXIF w formacie JSON oraz dedykowana sekcja z opisem stanu manifestu C2PA (jeśli metoda c2pa zwróciła wynik).
6. Sekcja *Visualizations* wyświetla wszystkie wizualizacje zwrócone przez metody, na przykład:
- mapę błędu kompresji JPEG (ELA),
 - reprezentację widma lub energii wysokich częstotliwości (FFT),
 - wizualizacje residualu / korelacji (PRNU),
 - mapy cieplne modeli CNN.
7. W sekcji *Result* system prezentuje:
- końcową etykietę: AI, REAL lub UNKNOWN,
 - wartość score (prawdopodobieństwo AI według meta–modelu lub średniej z metod bazowych),
 - pasek postępu obrazujący poziom pewności, o ile wynik nie został oznaczony jako UNKNOWN.
8. W sekcji *Metric details* użytkownik może podejrzeć:
- szczegółowe metryki dla każdej metody (np. ela_elap90, fft_hf_ratio, wskaźniki PRNU),
 - składowe components zawierające score każdej metody.
9. W sekcji *Attribution* system prezentuje wyniki klasyfikacji rodzaju potencjalnego generatora obrazu:
- etykietę (np. diffusion lub GAN),
 - wartość confidence dla predykcji modelu,
 - szczegółowe metryki i wizualizację mapy cieplnej.
10. Użytkownik ma możliwość pobrania kompletnego raportu w formacie JSON za pomocą przycisku *Download report (JSON)* oraz wizualizacji za pomocą przycisku *Download visualizations (ZIP)*.

Scenariusz 2: Wykrycie obrazów oznaczonych jako AI w C2PA

Drugi ważny scenariusz dotyczy obrazów, które zawierają manifest C2PA z wyraźną informacją, że treść jest generowana przez sztuczną inteligencję.

1. W panelu bocznym użytkownik wybiera konfigurację obejmującą metodę c2pa (najczęściej poprzez aktywację pipeline-u combined_methods lub combined_methods_fast).

2. Po wgraniu obrazu funkcja `analyze_image()` w pierwszej kolejności uruchamia metodę `c2pa`. Jeśli zwrócony score jest równy 99.9, traktowane jest to jako bardzo silny sygnał, że obraz pochodzi z AI:

- pozostałe metody nie są już uruchamiane,
- etykieta końcowa ustawiana jest na AI,
- score końcowy przyjmuje wartość z metody C2PA.

3. W interfejsie:

- w sekcji *C2PA manifest* użytkownik widzi opis przyczyny decyzji (np. label i komentarz),
- jest jasno zaznaczone, że decyzja pochodzi z podpisu C2PA, a nie z analizy sygnałowej obrazu.

Taki scenariusz pozwala bardzo szybko klasyfikować obrazy, które już w metadanych zawierają wiarygodną informację o pochodzeniu AI.

Scenariusz 3: Wykorzystanie PRNU jako bramki lub sygnału niepewności

Kolejny scenariusz dotyczy szczególnej roli metody PRNU w logice systemu.

1. Przy aktywnym pipeline `combined_methods` funkcja `analyze_image()` uruchamia metodę `prnu` na początku przetwarzania.

2. Jeśli `prnu` zwróci score równy 0.1, traktowane jest to jako bardzo silny sygnał, że obraz jest REAL:

- inne metody nie są uruchamiane,
- etykieta końcowa przyjmuje wartość REAL,
- score końcowy jest równy wartości z PRNU.

3. Wprowadzono też logiczne reguły dla sytuacji, w których PRNU nie jest w stanie niczego wiarygodnie powiedzieć. Jeżeli:

- jedyną (lub jedną z dwóch) uruchomionych metod jest PRNU (ewentualnie PRNU + C2PA),
- score PRNU wynosi dokładnie 1.0 (interpretowane jako *brak wiarygodnej oceny*),

wówczas:

- etykieta końcowa ustawiana jest na UNKNOWN,
- pasek score nie jest wyświetlany,
- w sekcji metryk komponent `prnu` prezentowany jest jako "unknown".

4. Dzięki temu system nie wymusza sztucznej pewności w sytuacjach, w których dane są zbyt słabe do wiarygodnej analizy PRNU.

Scenariusz 4: Tryb wsadowy

System przewiduje także scenariusz do analizy wsadowej wielu obrazów.

1. W panelu bocznym użytkownik wybiera odpowiednie metody.
2. Zamiast pojedynczego obrazu użytkownik przesyła plik archiwum ZIP.
3. Funkcja `run_batch()`:
 - rekurencyjnie przegląda katalog wejściowy,
 - filzuje pliki graficzne (JPEG/PNG/BMP),
 - dla każdego obrazu wywołuje funkcję `analyze_image()`.
4. W sekcji *Batch summary* system prezentuje dokładną liczbę obrazów z katalogu sklasyfikowanych jako REAL, AI i UNKNOWN oraz wykres kołowy przedstawiający rozkład wyników.
5. Użytkownik ma możliwość pobrania podsumowania w formie raportu za pomocą przycisku *Download summary (JSON)*, zbiorczych wyników za pomocą przycisku *Download results (CSV)* oraz wykresu kołowego za pomocą przycisku *Download chart (PNG)*.

Scenariusz 5: Analiza wsadowa z linii poleceń

Ostatni scenariusz dotyczy przetwarzania większej liczby obrazów bez uruchamiania interfejsu graficznego, w trybie wsadowym.

1. Użytkownik uruchamia aplikację z parametrem `-batch`, np.:

```
python app.py <input_dir> --out <output_dir> --methods ela,fft --threshold 0.5
```

2. Dla każdego pliku zapisywany jest:

- szczegółowy raport w formacie JSON (zawierający wynik, metryki, ewentualne wizualizacje zakodowane w Base64),
- wpis w pliku `summary.csv` z najważniejszymi polami: ścieżką do obrazu, etykietą AI/REAL/UNKNOWN, score, wybranymi metrykami oraz ścieżką do pliku raportu.

3. Tryb wsadowy jest wykorzystywany głównie do:

- testowania systemu na dużych datasetach,
- zbierania statystyk jakościowych,
- integracji z zewnętrznymi pipeline'ami analitycznymi.

6.3. Wyniki testów metod detekcji

6.3.1. Wyniki testów metod detekcji REAL vs AI

W tej sekcji przedstawiono rezultaty działania poszczególnych metod detekcji oraz meta–modelu łączącego je w jeden klasyfikator. Wyniki te pochodzą z niezależnych eksperymentów:

- testów na zbiorach przygotowanych ręcznie (Pexels vs. DCGAN / SD / DALL·E 2 / BigGAN),
- testów metod bazowych i meta–modelu na zbiorze treningowym i walidacyjnym Tiny–GenImage [41].

Wyniki metod bazowych: ELA i FFT na zbiorach autorskich

Pierwsza seria testów dotyczyła modeli trenowanych na niewielkich, jednorodnych zestawach danych:

- **ELA:** zdjęcia krajobrazowe z serwisu Pexels vs. obrazy syntetyczne generowane przez DCGAN.
- **FFT:** zdjęcia rzeczywiste z Pexels oraz obrazy generowane skryptami wykorzystującymi modele Stable Diffusion, DALL·E 2 i BigGAN.

Wyniki przedstawiono w tabeli 6.1.

Tabela 6.1: Wyniki metod ELA i FFT na małych zestawach testowych.

Metoda	Accuracy (test)	ROC AUC (test)
ELA (200 zdjęć testowych)	0.8200	0.9405
FFT (245 zdjęć testowych)	0.6148	0.6612

Wnioski: Metoda ELA osiąga bardzo wysoką skuteczność na prostym zadaniu odróżniania krajobrazów Pexels od obrazów DCGAN, co wskazuje na silną odmienność artefaktów kompresji JPEG w tych dwóch domenach. Metoda FFT wypada słabiej — generatory SD, DALL·E 2 i BigGAN produkują widma częstotliwości bardziej zblizone do zdjęć rzeczywistych.

Wyniki te pokazują jednak przede wszystkim ograniczenia testów na niewielkich, jednorodnych zbiorach: łatwo o *dataset bias*, a metody mogą uczyć się różnic nieistotnych z punktu widzenia rzeczywistego zadania wykrywania obrazów AI.

Wyniki na zbiorze Tiny–GenImage

Kolejna seria eksperymentów wykorzystuje duży i różnorodny zbiór Tiny–GenImage, zawierający wiele typów danych i modeli generatywnych. To właśnie na nim trenowano meta–model, który następnie stał się częścią finalnego systemu.

Wyniki na zbiorze treningowym

Tabela 6.2 przedstawia wyniki metod bazowych na zbiorze treningowym. Modele CNN StyleGAN i ConvNeXt StyleGAN były trenowane i testowane wyłącznie na obrazach zawierających twarze (FACE).

Tabela 6.2: Wyniki metod bazowych na zbiorze Tiny-GenImage (TRAIN).

Metoda	ACC	AUC	n
ELA	0.561	0.668	28000
FFT	0.599	0.712	28000
CLIP (<i>zero-shot</i>)	0.658	0.748	28000
CLIP + SVM	0.776	0.885	28000
CNN StyleGAN (FACE)	0.629	0.504	1468
ConvNeXt StyleGAN (FACE)	0.629	0.504	1468

Wyniki na zbiorze walidacyjnym

Tabela 6.3 prezentuje wyniki na zbiorze walidacyjnym. Modele CNN StyleGAN i ConvNeXt StyleGAN są wyłącznie na podzbiorze FACE (387 obrazów).

Tabela 6.3: Wyniki metod bazowych na zbiorze Tiny-GenImage (VAL).

Metoda	ACC	AUC	n
ELA	0.510	0.617	7000
FFT	0.655	0.784	7000
CLIP (<i>zero-shot</i>)	0.665	0.751	7000
CLIP + SVM	0.784	0.901	7000
CNN StyleGAN (FACE)	0.612	0.531	387
ConvNeXt StyleGAN (FACE)	0.612	0.507	387

Wnioski: W przeciwieństwie do małych zestawów, na dużym i zróżnicowanym zbiorze:

- Najlepsze wyniki spośród metod bazowych osiąga CLIP + SVM.
- CNN StyleGAN i ConvNeXt StyleGAN były trenowane wyłącznie na obrazach twarzy generowanych przez model StyleGAN, co ogranicza ich zdolność generalizacji do innych typów obrazów i generatorów.
- metoda ELA traci praktycznie całą moc dyskryminacyjną,
- Metody FFT i CLIP zero-shot utrzymują stabilną, umiarkowaną skuteczność.

Potwierdza to tezę, że żadna pojedyncza metoda nie wystarcza do skutecznej detekcji obrazów AI w szerokiej domenie danych.

Wyniki meta-modelu

Meta-model decyzyjny integruje wyniki wszystkich metod bazowych. Wyniki prezentowane są osobno dla obrazów twarzy (FACE) oraz pozostałych (NO FACE), a także dla całego zbioru (ALL).

Zbiór treningowy

Wyniki meta-modelu na zbiorze treningowym przedstawiono w tabeli 6.4.

Tabela 6.4: Wyniki meta-modelu na zbiorze treningowym.

Podzbiór	ACC	AUC	n
FACE	0.956	0.991	1468
NO FACE	0.940	0.984	26532
ALL	0.940	0.985	28000

Zbiór walidacyjny

Wyniki meta-modelu na zbiorze walidacyjnym przedstawiono w tabeli 6.5.

Tabela 6.5: Wyniki meta-modelu na zbiorze walidacyjnym.

Podzbiór	ACC	AUC	n
FACE	0.943	0.988	387
NO FACE	0.944	0.986	6613
ALL	0.944	0.986	7000

Wnioski końcowe:

- Meta-model znaczco przewyższa wszystkie pojedyncze metody, osiągając AUC bliskie 0.99 na podzbiorze FACE i ponad 0.98 na NO FACE.
- Modele CNN i ConvNeXt StyleGAN służą głównie do detekcji obrazów twarzy, dlatego same w sobie nie są wystarczające.
- Łączenie wielu metod detekcji - analiz sygnałowych, widmowych i semantycznych - jest istotnie skuteczniejsze niż stosowanie któregośkolwiek z algorytmów osobno.
- Wyniki potwierdzają zasadność zastosowania wielomodułowego pipeline'u z warstwą agregacji decyzji.

Ocena meta–modelu na danych artystycznych i grafice wektorowej

W celu sprawdzenia zachowania meta–modelu w domenach odmiennych od klasycznych zdjęć fotograficznych przeprowadzono dodatkową ocenę na danych artystycznych oraz grafice wektorowej. Zbiory te charakteryzują się inną strukturą wizualną, uproszczoną geometrią oraz ograniczoną obecnością tekstur i artefaktów typowych dla fotografii, co czyni zadanie detekcji obrazów generowanych przez AI szczególnie wymagającym.

Przygotowanie danych Dla każdej z dwóch domen (*art* oraz *vector graphic*) przygotowano zbalansowane zbiory zawierające zarówno obrazy rzeczywiste, jak i syntetyczne.

- **Dane syntetyczne (AI):** obrazy zostały wygenerowane przy użyciu modeli DALL·E 2 oraz Stable Diffusion. Dla każdej domeny wygenerowano po 30 obrazów artystycznych oraz 30 grafik wektorowych z każdego modelu, co dało łącznie 60 obrazów syntetycznych na domenę.
- **Dane rzeczywiste (human):** obrazy artystyczne wylosowano ze zbioru Met Art, natomiast grafiki wektorowe pochodziły ze zbioru OpenClipart. Dla każdej domeny wylosowano po 60 obrazów, zapewniając równowagę pomiędzy klasami.

Łącznie każdy zbiór testowy składał się ze 120 obrazów (60 real, 60 AI). Dane te nie były wykorzystywane na żadnym etapie treningu ani walidacji modeli.

Wyniki — obrazy artystyczne Macierz pomyłek dla obrazów artystycznych została przedstawiona w tabeli 6.6.

Tabela 6.6: Macierz pomyłek meta–modelu dla obrazów artystycznych

	REAL	AI
REAL	49	11
AI	9	51

Meta–model osiągnął wysoką skuteczność detekcji w tej domenie, poprawnie klasyfikując większość obrazów syntetycznych oraz rzeczywistych. Osiągnięta dokładność wyniosła 0.83, a wartości czułości dla obu klas pozostały zbliżone.

Wyniki — grafika wektorowa Macierz pomyłek dla grafiki wektorowej została przedstawiona w tabeli 6.7.

Tabela 6.7: Macierz pomyłek meta–modelu dla grafiki wektorowej

	REAL	AI
REAL	48	12
AI	24	36

W tej domenie obserwuje się wyraźny spadek skuteczności detekcji obrazów generowanych przez AI. Meta–model częściej błędnie klasyfikował grafiki syntetyczne jako rzeczywiste, co skutkowało obniżeniem dokładności do poziomu 0.70.

Wnioski

- Meta–model zachowuje dobrą skuteczność w domenie obrazów artystycznych, mimo braku podobnych danych w zbiorze treningowym.
- Grafika wektorowa stanowi znacznie bardziej wyzwane detekcyjne, co wynika z uproszczonej struktury wizualnej oraz ograniczonej liczby cech niskopoziomowych.
- Wyniki potwierdzają, że skuteczność detekcji obrazów generowanych przez AI jest silnie zależna od charakteru danych, a domeny niefotograficzne wymagają odrębnego traktowania.
- Eksperyment ten uzupełnia wyniki uzyskane na zbiorze Tiny-GenImage, dostarczając dodatkowego spojrzenia na zachowanie systemu w praktycznych, niestandardowych scenariuszach.

Ocena meta–modelu na danych fotograficznych z różnych generatorów

Kolejny eksperyment miał na celu ocenę skuteczności meta–modelu w domenie fotorealistycznych obrazów przedstawiających typowe sceny i obiekty, zarówno rzeczywistych, jak i generowanych przez różne klasy modeli generatywnych. W przeciwieństwie do wcześniejszych testów benchmarkowych, skupiono się na realistycznym scenariuszu obejmującym obrazy o charakterze fotografii internetowej.

Przygotowanie danych Zbiór testowy obejmował sześć kategorii semantycznych: *people, city, nature, animals, food, architecture*. Dla każdej kategorii przygotowano po 30 obrazów z czterech źródeł:

- **Pexels** — rzeczywiste zdjęcia fotograficzne pochodzące z publicznego serwisu stockowego Pexels, udostępniającego wysokiej jakości fotografie wykonywane przez ludzi i wykorzystywane m.in. w zastosowaniach komercyjnych,
- **DALL·E 2** — obrazy syntetyczne generowane przez komercyjny model dyfuzyjny,
- **Stable Diffusion** — obrazy syntetyczne generowane lokalnie przez model dyfuzyjny,
- **BigGAN** — obrazy syntetyczne generowane przez klasyczny model typu GAN.

Łącznie dla każdej kategorii oceniono 120 obrazów, co daje 720 obrazów w całym eksperymentie (180 obrazów rzeczywistych oraz 540 obrazów syntetycznych). Dane te nie były wykorzystywane na żadnym etapie treningu ani walidacji modeli.

Wyniki klasyfikacji Tabela 6.8 przedstawia liczbę obrazów zaklasyfikowanych jako *REAL* oraz *AI* przez meta–model, w zależności od źródła danych i kategorii.

Tabela 6.8: Wyniki klasyfikacji REAL vs AI dla danych fotograficznych.

Kategoria	Źródło	REAL	AI
animals	Pexels	29	1
	DALL·E 2	13	17
	Stable Diffusion	4	26
	BigGAN	0	30
architecture	Pexels	29	1
	DALL·E 2	14	16
	Stable Diffusion	14	16
	BigGAN	0	30
city	Pexels	27	3
	DALL·E 2	9	21
	Stable Diffusion	15	15
	BigGAN	0	30
food	Pexels	29	1
	DALL·E 2	14	16
	Stable Diffusion	14	16
	BigGAN	0	30
nature	Pexels	28	2
	DALL·E 2	16	14
	Stable Diffusion	14	16
	BigGAN	0	30
people	Pexels	30	0
	DALL·E 2	18	12
	Stable Diffusion	15	15
	BigGAN	0	30

Metryki zbiorcze Na podstawie wyników zagregowanych ze wszystkich kategorii obliczono podstawowe metryki klasyfikacyjne dla zadania *REAL vs AI*:

- **Accuracy:** 0.77
- **Recall (AI):** 0.70

- **Recall (REAL):** 0.96

Wysoka wartość czułości dla klasy *REAL* wskazuje, że meta–model bardzo rzadko błędnie klasyfikuje zdjęcia rzeczywiste jako obrazy generowane przez AI. Jednocześnie umiarkowany poziom recall dla klasy *AI* świadczy o rosnącym stopniu podobieństwa nowoczesnych obrazów syntetycznych do fotografii wykonywanych przez ludzi.

Wnioski

- Meta–model niemal bezbłędnie identyfikuje obrazy generowane przez **BigGAN**, co potwierdza dużą odmiennosć tej klasy generatorów od współczesnych modeli dyfuzyjnych oraz fotografii rzeczywistych.
- Zdjęcia rzeczywiste z serwisu **Pexels** są w zdecydowanej większości poprawnie rozpoznawane jako obrazy realne.
- Obrazy generowane przez **Stable Diffusion** są częściej klasyfikowane jako syntetyczne niż obrazy z **DALL·E 2**, co sugeruje większą obecność subtelnych artefaktów charakterystycznych dla lokalnych modeli dyfuzyjnych.
- Największą trudność detekcyjną stanowią obrazy generowane przez **DALL·E 2**, które w wielu przypadkach są klasyfikowane podobnie jak fotografie rzeczywiste.
- Eksperyment potwierdza, że skuteczność detekcji obrazów generowanych przez AI jest silnie zależna od rodzaju generatora, nawet w obrębie tej samej domeny fotograficznej.

6.3.2. Wyniki testów metod identyfikacji generatora obrazów

Test modelu GAN vs Diffusion

Test przeprowadzono na zbiorze z datasetu *Tiny-GenImage* [41], w którym klasa **GAN** składała się z obrazów wygenerowanych przez model *BigGAN*, natomiast klasa **Diffusion** obejmowała obrazy z pozostałych modeli dyfuzyjnych w zbiorze. Liczba próbek była zbalansowana: po 500 obrazów na klasę.

Tabela 6.9: Metryki klasyfikacji obrazów Diffusion vs GAN na zbiorze walidacyjnym.

Klasa	Precision	Recall	F1-score	n
Diffusion	0.83	0.94	0.88	500
GAN	0.93	0.81	0.86	500
Average	0.88	0.87	0.87	1000
Accuracy				0.87

Wnioski:

- Model osiągnął wysoką skuteczność (87%) w rozróżnianiu obrazów generowanych przez modele dyfuzyjne i GAN.

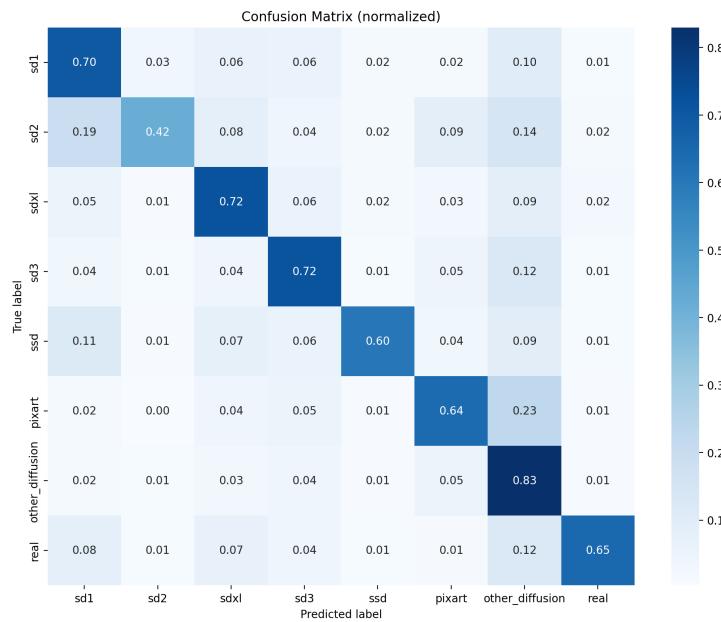
- Obrazy **diffusion** były rzadko mylone z GAN (recall 0.94), natomiast część obrazów **GAN** została błędnie sklasyfikowana jako diffusion (recall 0.81).
- Wyniki wskazują, że model jest skuteczny w rozróżnianiu typów generatorów.

Test modelu ConvNeXt Diffusion

Model **ConvNeXt Diffusion** został przetestowany na zbiorze testowym DRAGON, obejmującym obrazy generowane przez różne modele dyfuzyjne.

Tabela 6.10: Metryki klasyfikacji modelu ConvNeXt Diffusion na zbiorze DRAGON.

Klasa	Precision	Recall	F1-score	n
sd1	0.6570	0.6992	0.6774	1200
sd2	0.5915	0.4200	0.4912	400
sdxl	0.8251	0.7212	0.7697	2400
sd3	0.6513	0.7158	0.6820	1200
ssd	0.7954	0.6025	0.6856	800
pixart	0.6937	0.6417	0.6667	1200
other_diffusion	0.6802	0.8296	0.7475	2400
real	0.6860	0.6500	0.6675	400
Accuracy		0.7100		10000
Macro avg	0.6975	0.6600	0.6735	10000
Weighted avg	0.7162	0.7100	0.7085	10000



Rysunek 6.10: Macierz pomyłek modelu ConvNeXt Diffusion na zbiorze testowym DRAGON.

Wnioski z wyników przedstawionych w tabeli 6.10 i macierzy pomyłek 6.10:

- Model osiągnął ogólną dokładność (**accuracy**) na poziomie 0.71, co wskazuje na umiarowaną skuteczność w wieloklasowej klasyfikacji modeli dyfuzyjnych.
- Najwyższy recall uzyskano dla klasy **other_diffusion** (0.83), co sugeruje, że model najpewniej rozpoznaje obrazy z tej kategorii.
- Najniższy recall obserwuje się w klasie **sd2** (0.42), co oznacza, że obrazy tej rodziny są często mylnie klasyfikowane.
- Analiza macierzy pomyłek wskazuje, że największa część błędnych predykcji dla danego modelu najczęściej trafia do klasy **other_diffusion**, co jest intuicyjne, ponieważ obejmuje ona obrazy z bardziej zróżnicowanych architektur niż pozostałe klasy.

Test modelu CLIP Diffusion

Model CLIP Diffusion został przetestowany na zbiorze testowym Defactify Image Dataset [36], obejmującym obrazy wygenerowane przez nowoczesne modele dyfuzyjne oraz modele komercyjne. Celem eksperymentu była wieloklasowa atrybucja pochodzenia obrazu do konkretnego generatora.

Tabela 6.11: Metryki klasyfikacji modelu CLIP Diffusion na zbiorze Defactify.

Klasa	Precision	Recall	F1-score	Support
sd21	0.45	0.57	0.50	7500
sdxl	0.49	0.49	0.49	7500
sd3	0.48	0.42	0.45	7500
dalle3	0.54	0.49	0.51	7500
midjourney	0.52	0.49	0.51	7500
Accuracy		0.49		37500
Average	0.50	0.49	0.49	37500

Wnioski na podstawie tabeli 6.11:

- Model osiągnął ogólną dokładność na poziomie 49%, co wskazuje na wysoką trudność zadania precyzyjnej atrybucji pochodzenia obrazów generowanych przez nowoczesne modele syntezy.
- Najlepsze wyniki względem metryki F1-score uzyskano dla klas *dalle3* oraz *midjourney*, jednak różnice pomiędzy klasami pozostają niewielkie, co sugeruje znaczne podobieństwo rozkładów cech wizualnych.
- Relatywnie niska skuteczność klasyfikacji może wynikać z faktu, że współczesne modele dyfuzyjne oraz modele komercyjne generują obrazy o wysokiej jakości, pozbawione wyraźnych, niskopoziomowych artefaktów charakterystycznych dla starszych generatorów.
- Otrzymane wyniki są spójne z obserwacjami przedstawionymi w pracy [35], gdzie bazowy model autorów zbioru danych osiągnął dokładność na poziomie 44,9%, co potwierdza trudność problemu atrybucji pochodzenia w scenariuszu nowoczesnych generatorów.

6.4. Podsumowanie rozdziału

Rozdział prezentuje wyniki prac realizowanych w ramach projektu, obejmujące zarówno opracowanie i implementację systemu detekcji obrazów AI, jak i jego testy w różnych scenariuszach użytkowania. System integruje analizę sygnałową (PRNU, FFT, ELA), semantyczną (CLIP), lekkie sieci CNN/ConvNeXt oraz moduły atrybucji pochodzenia (GAN vs Diffusion, modele dyfuzyjne). Obsługuje różne formaty graficzne, analizuje metadane (EXIF, C2PA), generuje wizualizacje wyników i raporty, działając w trybie interaktywnym oraz wsadowym.

Testy wykazały, że metody bazowe mają ograniczenia, natomiast meta-model znaczaco poprawia skuteczność, osiągając AUC bliskie 0,99. System umożliwia także umiarkowaną atrybucję generatorów, skuteczną przy rozróżnianiu GAN vs Diffusion, choć trudną w przypadku nowoczesnych komercyjnych modeli dyfuzyjnych.

Rozdział 7

Zakończenie

Celem niniejszego projektu było zaprojektowanie i implementacja systemu detekcji obrazów generowanych przez sztuczną inteligencję, wykorzystującego różnorodne metody analizy: sygnałowe, widmowe, semantyczne oraz metadane. Założeniem było odejście od pojedynczych, wyspecjalizowanych detektorów na rzecz modułowego pipeline'u, w którym wiele niezależnych źródeł informacji wspiera końcową decyzję klasyfikacyjną.

Zrealizowany system spełnia postawione cele. Powstało kompletne oprogramowanie, umożliwiające analizę pojedynczych obrazów oraz przetwarzanie wsadowe, z czytelnym interfejsem użytkownika oraz możliwością wizualizacji wyników poszczególnych metod. Każdy z zaimplementowanych komponentów (PRNU, FFT, ELA, CLIP, EXIF, C2PA, CNN, ConvNeXt) wnosi odmienny typ informacji, a ich integracja w postaci meta-modelu znaczaco poprawia jakość detekcji w porównaniu do metod pojedynczych.

Przeprowadzone eksperymenty wykazały, że metody bazowe charakteryzują się różną skutecznoscia w zależności od rodzaju danych i stopnia ich zróżnicowania. Szczególnie widoczne było to w przypadku ELA, która dobrze radziła sobie w wąskiej domenie, lecz traciła skutecznosc na bardziej ogólnych zbiorach. Z kolei metody FFT oraz CLIP okazały się stabilniejsze, jednak dopiero ich połączenie w meta-klassyfikatorze pozwoliło osiągnąć wysoką skutecznosc (ACC = 0.94, AUC = 0.99 na zbiorze Tiny-GenImage). Dodatkowe testy dotyczące identyfikacji rodzaju modelu generatywnego wykazały, że rozróżnianie rodzin generatorów (np. GAN vs. modele dyfuzyjne) jest możliwe, jednak skutecznosc tego zadania maleje wraz ze wzrostem podobieństwa architektur oraz jakości generowanych obrazów.

Z punktu widzenia autorów projekt można uznać za udany. Udało się nie tylko zaimplementować złożony system detekcji w podstawowej i skalowej wersji, lecz także zweryfikować jego działanie na rzeczywistych i syntetycznych danych pochodzących z różnych źródeł. Projekt pozwolił na praktyczne poznanie ograniczeń współczesnych metod detekcji obrazów AI oraz potwierdził, że podejście wielomodułowe jest bardziej odporne na zmienność generatorów niż pojedyncze algorytmy.

Ze względu na ograniczony czas, który mógł być przeznaczony na realizację pracy, nie udało się zrealizować wszystkich pomysłów. System może być dalej rozwijany poprzez:

- rozszerzenie zbioru fingerprinów PRNU o kolejne modele urządzeń,
- zastosowanie nowszych modeli semantycznych lub multimodalnych,
- automatyczną adaptację progów decyzyjnych do nowych generatorów AI,
- zastosowanie metod detekcji nieznanych generatorów (open-set),
- integrację wersji podstawowej systemu z wersją skalową.

- integrację z systemami weryfikacji treści w środowiskach produkcyjnych.

Podsumowując, projekt stanowi funkcjonalną i dobrze ugruntowaną podstawę do dalszych prac badawczych lub inżynierskich w obszarze detekcji treści syntetycznych i analizy wiarygodności obrazów cyfrowych.

Kod źródłowy

Pełne kody źródłowe obu systemów są dostępne w repozytoriach GitHub:

- <https://github.com/Wojciech-Fortuna/ai-image-detector>
- <https://github.com/aslafdev/MultiModelInference>

Bibliografia

- [1] *140k Real and Fake Faces*. <https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces>. 2023.
- [2] *AI or Not – AI Image Detector*. <https://www.aiornot.com>.
- [3] *AI-Generated Content Detection*. <https://hivemoderation.com/ai-generated-content-detection>. 2025.
- [4] *Artifact Dataset – Artificial (Fake) and Factual (Real) Image Dataset for Synthetic Image Detection*. <https://www.kaggle.com/datasets/awsaf49/artifact-dataset>. Kaggle, 2023.
- [5] *Camera Photos vs AI Generated Photos Classifier (dataset)*. <https://www.kaggle.com/datasets/rafsunahmad/camera-photos-vs-ai-generated-photos-classifier>. Kaggle.
- [6] T. Chakraborty, U. Reddy K S, S. M. Naik, M. Panja i B. Manvitha. „Ten Years of Generative Adversarial Nets (GANs): A Survey of the State-of-the-Art”. W: *arXiv preprint arXiv:2308.16316* (2023). URL: <https://arxiv.org/abs/2308.16316>.
- [7] A. Challa i in. „Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks”. W: *Preprints.org* (2023). URL: <https://www.preprints.org/manuscript/202310.0547>.
- [8] D. Cioni, C. Tzelepis, L. Seidenari i I. Patras. „Are CLIP features all you need for Universal Synthetic Image Origin Attribution?” W: *arXiv preprint arXiv:2408.09153* (2024). URL: <https://arxiv.org/abs/2408.09153>.
- [9] Coalition for Content Provenance and Authenticity. „C2PA Technical Specification”. W: *C2PA* (2023). URL: <https://c2pa.org/specifications/specifications/1.3/>.
- [10] D. Cozzolino, G. Poggi, R. Corvi, M. Nießner i L. Verdoliva. „Leveraging CLIP for Synthetic Image Detection”. W: *IEEE Open Journal of Signal Processing* 5 (2024). Dostęp: 2025-12-15, s. 141–152. doi: [10.1109/OJSP.2023.3340552](https://doi.org/10.1109/OJSP.2023.3340552). URL: <https://grip-unina.github.io/ClipBased-SyntheticImageDetection/>.
- [11] *Dragon Dataset*. <https://huggingface.co/datasets/lesc-unifi/dragon>. 2023.
- [12] *Dresden Image Database (Kaggle mirror)*. <https://www.kaggle.com/datasets/micscodes/dresden-image-database>.
- [13] T. Dzanic, K. Shah i F. Witherden. „Fourier Spectrum Discrepancies in Deep Network Generated Images”. W: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020. URL: <https://papers.nips.cc/paper/2020/hash/1f8d87e1161af68b81bace188a1ec624-Abstract.html>.
- [14] *FloreView Dataset*. <https://lesc.dinfo.unifi.it/FloreView/Dataset/>.

-
- [15] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa i T. Holz. „Leveraging Frequency Analysis for Deep Fake Image Recognition”. W: *International Conference on Machine Learning (ICML)*. 2020. URL: <https://proceedings.mlr.press/v119/frank20a/frank20a.pdf>.
 - [16] M. Goljan i J. Fridrich. „Camera Identification from Cropped and Scaled Images”. W: *IEEE Transactions on Information Forensics and Security* 7.1 (2012), s. 296–308. doi: [10.1109/TIFS.2011.2168216](https://doi.org/10.1109/TIFS.2011.2168216). URL: https://ws2.binghamton.edu/fridrich/Research/Crop_scale.pdf.
 - [17] M. Goljan, J. Fridrich i T. Filler. „Sensor Noise Camera Identification: Large-Scale Evaluation”. W: *IEEE Transactions on Information Forensics and Security* 4.2 (2009), s. 205–224. doi: [10.1109/TIFS.2009.2016007](https://doi.org/10.1109/TIFS.2009.2016007). URL: <https://ws2.binghamton.edu/fridrich/Research/EI7254-18.pdf>.
 - [18] M. Högemann, J. Betke i O. Thomas. „What you see is not what you get anymore: a mixed-methods approach on human perception of AI-generated images”. W: *Frontiers in Artificial Intelligence* (2025). URL: <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1707336/full>.
 - [19] *HPAI-BSC SuSy Dataset*. <https://huggingface.co/datasets/HpAI-BSC/SuSy-Dataset/tree/main/data>.
 - [20] *IMAGINE Image Database*. <https://kisi.pcz.pl/imagine/>.
 - [21] *isgen.ai – AI Image Detector*. <https://isgen.ai/pl/detektora-obrazu-AI>.
 - [22] Japan Electronics and Information Technology Industries Association. „Exchangeable Image File Format for Digital Still Cameras (Exif) Version 2.3”. W: *JEITA Standard* (2012). URL: https://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf.
 - [23] H. Liu, Z. Tan, C. Tan, Y. Wei, Y. Zhao i J. Wang. „Forgery-aware Adaptive Transformer for Generalizable Synthetic Image Detection”. W: *arXiv preprint arXiv:2312.16649* (2023). URL: <https://arxiv.org/abs/2312.16649>.
 - [24] Y. Liu, Y. Xiao i H. Tian. „Plug-and-Play PRNU Enhancement Algorithm with Guided Filtering”. W: *Sensors* 24.23 (2024), s. 7701. doi: [10.3390/s24237701](https://doi.org/10.3390/s24237701). URL: <https://www.mdpi.com/1424-8220/24/23/7701>.
 - [25] J. Lukas, J. Fridrich i M. Goljan. „Digital Camera Identification from Sensor Pattern Noise”. W: *IEEE Transactions on Information Forensics and Security* 1.2 (2006), s. 205–214. doi: [10.1109/TIFS.2006.873602](https://doi.org/10.1109/TIFS.2006.873602). URL: <https://ieeexplore.ieee.org/document/1634362>.
 - [26] T. Miyato, T. Kataoka, M. Koyama i Y. Yoshida. „Spectral Normalization for Generative Adversarial Networks”. W: *International Conference on Learning Representations (ICLR)*. 2018. URL: <https://openreview.net/forum?id=B1QRgziT->.
 - [27] A. G. Moskowitz, T. Gaona i J. Peterson. „Detecting AI-Generated Images via CLIP”. W: *arXiv preprint arXiv:2404.08788* (2024). URL: <https://arxiv.org/abs/2404.08788>.
 - [28] *Natural Images (dataset)*. <https://www.kaggle.com/datasets/prasunroy/natural-images>. Kaggle.
 - [29] Y. Nirkin, Y. Keller i T. Hassner. „Exposing GAN-Generated Faces Using Convolutional Neural Networks”. W: *CVPR Workshops*. 2019. URL: <https://arxiv.org/abs/1812.08247>.

-
- [30] A. Pandey i A. Mitra. *Detecting and Localizing Copy-Move and Image-Splicing Forgery*. Accessed: 2025-12-15. 2022. arXiv: 2202.04069. URL: <https://arxiv.org/abs/2202.04069>.
 - [31] N. Park, K. Kim, J. Choe i H. Shim. „Rethinking the Use of Vision Transformers for AI-Generated Image Detection”. W: *arXiv preprint arXiv:2303.00000* (2025). URL: <https://arxiv.org/abs/2303.00000>.
 - [32] Pillow Contributors. *Pillow: Python Imaging Library (Fork)*. <https://python-pillow.org/>. Accessed: 2026-01-12.
 - [33] A. Radford, L. Metz i S. Chintala. „Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. W: *arXiv* (2016). arXiv:1511.06434. URL: <https://arxiv.org/abs/1511.06434>.
 - [34] A. Radford i in. „Learning Transferable Visual Models From Natural Language Supervision”. W: *arXiv* (2021). URL: <https://arxiv.org/abs/2103.00020>.
 - [35] R. Roy i in. „Defactify: A Benchmark for Fine-Grained Attribution of AI-Generated Images”. W: *arXiv preprint arXiv:2601.00553* (2026). URL: <https://arxiv.org/abs/2601.00553>.
 - [36] R. Roy. *Defactify_Image_Dataset*. https://huggingface.co/datasets/Rajarshi-Roy-research/Defactify_Image_Dataset. 2026.
 - [37] *Stable Diffusion Data*. <https://www.kaggle.com/datasets/mohannadymansalah/stable-diffusion-dataaaaaaaaa>. Kaggle.
 - [38] Universome / ALIS Project. *LHQ: Large-scale High-Quality Image Dataset*. <https://github.com/universome/alis/tree/master>. 2021.
 - [39] S.-Y. Wang, O. Wang, R. Zhang, A. Owens i A. A. Efros. „CNN-generated images are surprisingly easy to spot... for now”. W: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. URL: <https://arxiv.org/abs/1912.11035>.
 - [40] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui i M.-H. Yang. „Diffusion Models: A Comprehensive Survey of Methods and Applications”. W: *ACM Computing Surveys* (2024). doi: [10.1145/3626235](https://doi.org/10.1145/3626235). URL: <https://arxiv.org/abs/2209.00796>.
 - [41] S. Yang. *Tiny GenImage: A Lightweight Version of the GenImage Dataset for AI-Generated Image Detection*. <https://www.kaggle.com/datasets/yangsangtai/tiny-genimage>. 2025.

Spis rysunków

1.1 Przykładowe obrazy wygenerowane przez StyleGAN, źródło: [4]	11
1.2 Przykładowe obrazy wygenerowane przez Stable Diffusion, źródła: [4], [37] . .	11
1.3 Aplikacja isgen.ai do detekcji obrazów AI	16
2.1 Diagram przepływu danych (DFD) w systemie	26
2.2 Diagram przypadków użycia systemu detekcji obrazów generowanych przez AI	27
2.3 Makieta głównego ekranu aplikacji – przesyłanie obrazu do analizy	29
2.4 Makieta wyników analizy	29
2.5 Makieta wizualizacji analizy i metadanych EXIF	30
2.6 Makieta wyboru metod weryfikacji obrazów	30
3.1 Przykładowe mapy korelacji PRNU dla czterech modeli z bazy fingerprints.	47
3.2 Rozkład wyników funkcji decyzyjnej.	59
3.3 Schemat zintegrowanego pipeline'u decyzyjnego w wariancie combined_methods, obejmujący reguły bramkujące, metody bazowe oraz agregację wyników przez meta-model	66
3.4 Ogólna architektura systemu oraz przepływ danych pomiędzy interfejsem użytkownika, backendem API i trybem wsadowym (CLI)	71
4.1 Ogólna architektura systemu.	79
6.1 Główny strona systemu.	91
6.2 Widok obrazu i analizy metadanych w systemie.	91
6.3 Widok wizualizacji mapy cieplnej CLIP i mapy ELA.	91
6.4 Widok wizualizacji metod FFT i PRNU.	92
6.5 Widok prezentacji wyniku klasyfikacji REAL vs AI.	92
6.6 Prezentacja wyniku i wizualizacji klasyfikacji rodziny modelu Diffusion vs GAN.	92
6.7 Widok szczegółowych metryk.	93
6.8 Widok opcji pobrania raportu i wizualizacji.	93
6.9 Widok podsumowania analizy w trybie wsadowym.	93
6.10 Macierz pomyłek modelu ConvNeXt Diffusion na zbiorze testowym DRAGON. .	105

Spis tabel

1.1	Analiza ryzyka dla projektu systemu rozpoznawania obrazów AI	19
2.1	Wymagania funkcjonalne systemu wraz z priorytetami	23
2.2	Wymagania niefunkcjonalne systemu wraz z priorytetami	24
3.1	Architektura modelu CNN do klasyfikacji obrazów rzeczywistych i syntetycznych	60
3.2	Architektura modelu opartego na ConvNeXt Tiny wykorzystana w badaniach. .	61
6.1	Wyniki metod ELA i FFT na małych zestawach testowych.	98
6.2	Wyniki metod bazowych na zbiorze Tiny-GenImage (TRAIN).	99
6.3	Wyniki metod bazowych na zbiorze Tiny-GenImage (VAL).	99
6.4	Wyniki meta–modelu na zbiorze treningowym.	100
6.5	Wyniki meta–modelu na zbiorze walidacyjnym.	100
6.6	Macierz pomyłek meta–modelu dla obrazów artystycznych	101
6.7	Macierz pomyłek meta–modelu dla grafiki wektorowej	101
6.8	Wyniki klasyfikacji REAL vs AI dla danych fotograficznych.	103
6.9	Metryki klasyfikacji obrazów Diffusion vs GAN na zbiorze walidacyjnym. . .	104
6.10	Metryki klasyfikacji modelu ConvNeXt Diffusion na zbiorze DRAGON. . . .	105
6.11	Metryki klasyfikacji modelu CLIP Diffusion na zbiorze Defactify.	106