Raport
Reprezentacja Wiedzy i Wnioskowanie

Dominika Kicilińska

Projekt został przygotowany na bazie tematu pracy magisterskiej:
Detekcja syntetycznie generowanych twarzy z wykorzystaniem metod głębokiego uczenia i interpretowalnej sztucznej inteligencji.

Opis (praca magisterska): Celem niniejszej pracy magisterskiej jest zbadanie skuteczności nowoczesnych algorytmów głębokiego uczenia w rozpoznawaniu syntetycznie generowanych twarzy, tworzonych przez modele typu GAN oraz dyfuzyjne. W pracy analizowane są klasyczne metody forensyki cyfrowej, oraz metody wykorzystjące współczesne sieci neuronowe, takie jak CNN oraz Vision Transformers. Szczególny nacisk położono na interpretowalność uzyskanych wyników z wykorzystaniem technik Explainable AI, pozwalających wskazać charakterystyczne cechy odróżniające obrazy sztuczne od autentycznych.

Przeznaczenie projektu

Temat projektu: **System wnioskujący o autentyczności obrazu twarzy z wykorzystaniem wiedzy eksperckiej i wyników analizy modelu głębokiego uczenia.**

Przeznaczenie projektu: System ma za zadanie **wnioskować o tym, czy obraz przedstawiający twarz jest prawdziwy, czy syntetyczny**, wykorzystując **dane z modelu głębokiego uczenia (np. CNN / ViT)** oraz **reguły wiedzy eksperckiej / ontologię**, które opisują charakterystyczne cechy obrazów generowanych przez różne modele (GAN, diffusion model).

- warstwa “percepcyjna” → analiza obrazu (deep learning)
- warstwa “wnioskująca” → system wiedzy, który interpretuje wyniki i formułuje decyzję

Umożliwienie interpretowalnej klasyfikacji syntetycznych twarzy w kontekście forensyki cyfrowej

Cel badawczy

Cel badawczy tworzenia sieci GAN:
Celem badawczym projektu jest **zaprojektowanie i empiryczna weryfikacja architektury sieci GAN, zdolnej do syntezy realistycznych obrazów ludzkich twarzy na podstawie rzeczywistego zbioru danych** CelebA, oraz analiza wpływu wybranych elementów architektury i strategii treningowych na jakość oraz stabilność procesu generacji.

Specyfikacja funkcjonalności systemu – opis przypadków użycia

PRZYPADKI UŻYCIA SYSTEMU

1. **Moduł GAN** – generowanie syntetycznych obrazów twarzy
   Przypadek użycia: Generowanie syntetycznych obrazów twarzy

Aktorzy

- System (moduł GAN)
- Zbiór danych autentycznych twarzy (źródło danych)

Opis

Moduł GAN jest odpowiedzialny za generowanie syntetycznych obrazów ludzkich twarzy na podstawie zbioru treningowego zawierającego autentyczne fotografie. Celem działania modułu jest wytworzenie obrazów możliwie najbardziej zbliżonych do rzeczywistych, zawierających realistyczne cechy anatomiczne twarzy.

Warunki początkowe

- dostępny zbiór autentycznych zdjęć twarzy
- zainicjalizowany i skonfigurowany model GAN

Przebieg zdarzenia

- moduł GAN pobiera autentyczne obrazy twarzy ze zbioru treningowego.
- sieć generatora uczy się rozkładu danych rzeczywistych.
- generator tworzy syntetyczne obrazy twarzy.
- dyskryminator ocenia jakość wygenerowanych obrazów.

Proces uczenia trwa do momentu uzyskania satysfakcjonującej jakości generowanych obrazów. Wygenerowane obrazy są zapisywane w zbiorze danych jako obrazy syntetyczne.

Warunki końcowe

Zbiór **syntetycznych obrazów twarzy** gotowy do wykorzystania przez moduł klasyfikacyjny.

2. **Moduł klasyfikacyjny CNN** – rozpoznawanie autentyczności obrazu
   Przypadek użycia: Klasyfikacja obrazu twarzy

Aktorzy

- Użytkownik systemu
- Moduł klasyfikacyjny CNN

Opis
Moduł klasyfikacyjny CNN analizuje obrazy twarzy dostarczone na wejściu systemu i dokonuje klasyfikacji binarnej, określając, czy dany obraz przedstawia twarz autentyczną, czy syntetyczną.

Warunki początkowe
- wytrenowany model CNN
- obraz wejściowy w odpowiednim formacie

Przebieg zdarzenia
- użytkownik lub inny moduł systemu przekazuje obraz twarzy do modułu CNN.
- obraz zostaje poddany normalizacji i przeskalowaniu.
- model CNN przeprowadza ekstrakcję cech wizualnych.
- na podstawie cech model podejmuje decyzję klasyfikacyjną.
- wynik klasyfikacji (autentyczny / syntetyczny) zostaje zwrócony.

Warunki końcowe
Określona klasa obrazu wraz z prawdopodobieństwem przynależności do klasy.

3. **Moduł XAI** – interpretacja decyzji modelu (Explainable AI)
   Przypadek użycia: Wyjaśnienie decyzji klasyfikacyjnej

Aktorzy
- użytkownik systemu
- moduł XAI

Opis
Moduł XAI umożliwia interpretację decyzji podjętej przez model klasyfikacyjny CNN poprzez identyfikację obszarów obrazu, które miały największy wpływ na wynik klasyfikacji.

Warunki początkowe
- wytrenowany model CNN
- obraz wejściowy oraz wynik jego klasyfikacji

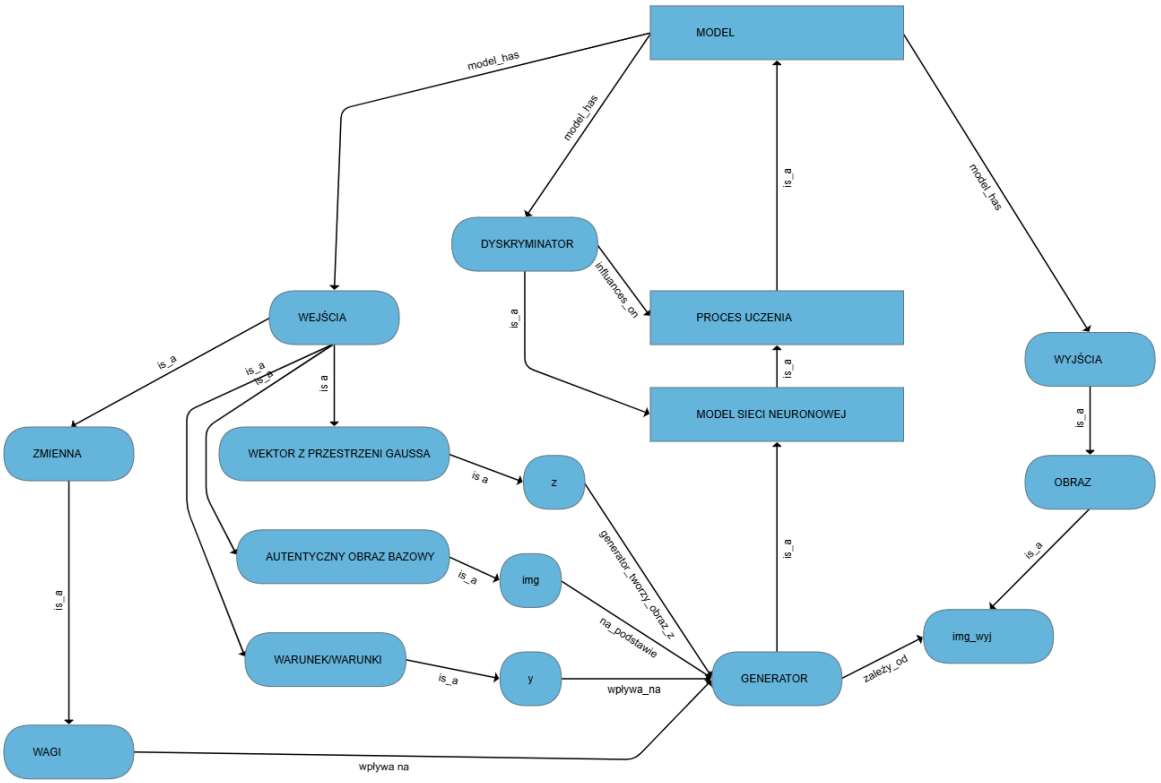
Scenariusz podstawowy
- moduł XAI otrzymuje obraz wejściowy oraz decyzję modelu CNN.
- wybrana metoda XAI (np. Grad-CAM) analizuje aktywacje sieci.
- generowana jest mapa istotności cech obrazu.
- mapa istotności zostaje nałożona na obraz wejściowy.
- użytkownik otrzymuje wizualne i opisowe wyjaśnienie decyzji modelu.

Warunki końcowe

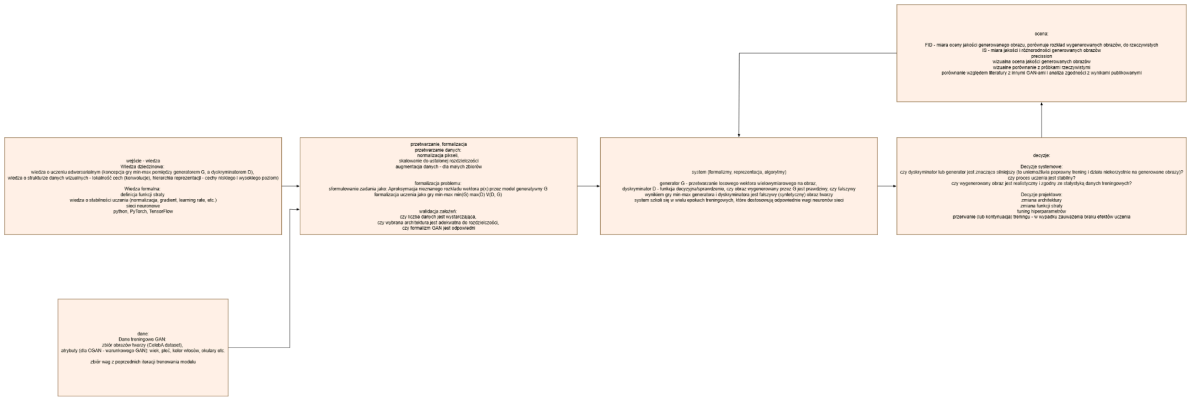
Wyjaśnienie decyzji klasyfikacyjnej w formie wizualnej i/lub opisowej.

Ontologia i schemat projektu

ONTOLOGIA (moduł GAN)



SCHEMAT (moduł GAN)



model generatywny GAN

1. Definicja: architektura deep learningu, która wykorzystuje trening adwersarialny między dwoma sieciami w celu generowania realistycznych próbek

2. Komponenty modelu:
    a. Generator (G) - Sieć, której zadaniem jest generowanie syntetycznych przykładów na podstawie danych treningowych
    b. Dyskryminator (D) - sieć, której zadaniem jest odróżnianie instancji syntetycznych od rzeczywistych danych.
    c. Przestrzeń utajona (Z) - wektor wejściowy szumu (z), zwykle próbkowany z rozkładu Gaussa, który jest podstawą generowania nowego obrazu.
3. Warianty modelu:
    a. Conditional GAN (CGAN): Wariant preferowany w generowaniu twarzy ze względu na możliwość kontrolowania generacji przez dołączenie warunku y (np. atrybutu) do wejść G i D
    b. StyleGAN: Znany z generowania bardzo realistycznych obrazów o wysokiej jakości. Używa sieci mapującej do transformacji z→w (wektor utajony do wektora stylu) i warstw AdaIN do wprowadzania informacji o stylu na każdej rozdzielczości
    c. DCGAN (Deep Convolutional GAN): Podstawa dla większości GANów obrazowych, wykorzystująca sploty zamiast warstw w pełni połączonych oraz normalizację wsadową (BN) dla stabilizacji treningu

dane i sterowanie

1. definicja: zbiór informacji, który jest używany do szkolenia modelu oraz elementów, które pozwalają na kontrolowanie wygenerowanego obrazu.
2. typ danych/kontroli:
    a. zbiór treningowy - zbiory danych obrazów twarzy, np CelebA
    b. dane warunkujące (y) - informacja dołączana do G i D w modelach CGAN. Może to być: klasa danych, maska segmentacji semantycznej, mapa krawędzi.
    c. obraz wzorcowy (Isrc) - obraz dostarczający cech stylu do przeniesienia na generowany obraz.
    d. wektor warunkowy - tablica cech używana do kontrolowanego/warunkowego generowania obrazów.

procesy i mechanizmy szkolenia

1. definicja: metody i funkcje używane do optymalizacji sieci GAN w celu generowania wysokiej wierności.
2. Proces/Mechanizm:
    a. trening adwersarialny - jego celem jest dążenie do równowagi, minimalizując funkcję straty G i maksymalizując funkcję straty D
    b. progresywny wzrost (ProGAN) - metoda zwiększająca stabilność treningu i jakość obrazu poprzez stopniowe zwiększanie rozdzielczości.

c. inwersja GAN - proces odnalezienia kodu utajonego (w) odpowiadającego danemu rzeczywistemu obrazowi, niezbędny do edycji prawdziwych fotografii.
d. domain alignment - konieczne do przeniesienia stylu między różnymi domenami
e. mieszany trening wsadowy - technika stosowana w celu zapewnienia stałego stosunku danych rzeczywistych do generowanych w każdej partii, co zapobiega ignorowaniu wektora warunkowego przez generator.

ocena i wyniki

1. definicja: metody ilościowe i jakościowe do pomiaru realizmu, różnorodności i spójności generowanych obrazów.
2. metryki:
    d. FID - mierzy odległość między rozkładami aktywacji obrazów rzeczywistych i syntetycznych, niższe wyniki oznaczają lepszą jakość percepcyjną
    e. inception score IS - mierzy jakość i różnorodność generowanych obrazów, wyższy wynik jest pożądany
    f. LPIPS Learned Perceptual Image Patch Similarity - mierzy prawdopodobieństwo percepcyjne między dwoma obrazami, używany do oceny różnorodności generacji.
    g. SWD Sliced Wasserstein Distance - mierzy statystyczną odległość rozkładów patch distribution w niskim poziomie

Dane wejściowe

Aby sieć dobrze uczyła się generowania syntetycznych twarzy, dane wejściowe muszą spełniać poniższe wymogi, ponieważ niska jakość danych prowadzi do pozostawiania artefaktów w generowanych obrazach, utrudnia uczenie dyskryminatora i obniża realizm gotowych wyników:

- wyraźnie widoczna twarz (frontalna lub półprofil),
- brak silnych zniekształceń i rozmyć,
- odpowiednie oświetlenie,
- wystarczająca rozdzielczość obrazu,
- ograniczona liczba przesłonięć twarzy (np. maski, ręce).

Dane wejściowe do modułu GAN pochodzą z ogólnodostępnych, akademickich zbiorów danych, powszechnie wykorzystywanych w badaniach nad rozpoznawaniem twarzy.

Przykładowe źródła, na których będzie opierał się trening to powszechnie dostępne datasety (Kaggle), które zawierają różnorodne obrazy ludzkich twarzy i

są wykorzystywane w treningu sieci generatywnym oraz są często cytowane w literaturze naukowej zgłębiającej temat generowania ludzkich twarzy:

- CelebA – duży zbiór zdjęć celebrytów o wysokiej jakości
- FFHQ (Flickr-Faces-HQ) – zbiór twarzy o bardzo wysokiej rozdzielczości
- LFW (Labeled Faces in the Wild) – zbiór referencyjny

Implementacja

Implementację zaczęłam od stworzenia pierwszego szkicu kodu, który zamieściłam na swoim githubie.

https://github.com/d0mi04/GAN-learning-repository/blob/GAN-first-approach/summary/GAN-first-approach-summary.md

Rozbudowywanie pliku generatora przeniosłam do osobnego pliku Google Colab (załącznik discriminator.ipynb)

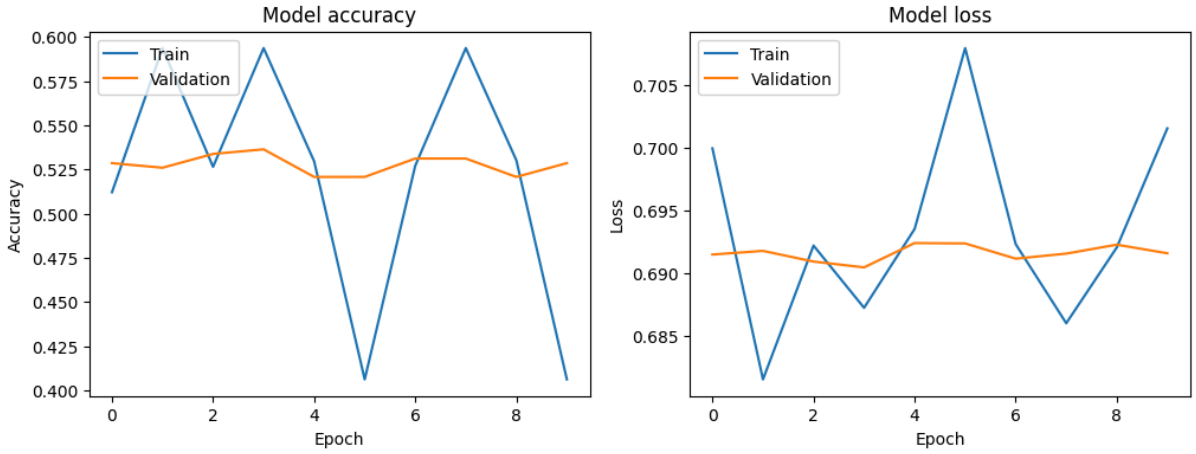
Pierwsza próba polepszenia wyników dyskryminatora, poprzez rozbudowanie prostego modelu konwolucyjnej sieci neuronowej:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_size,
image_size, 3)),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Conv2D(256, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(1024, activation='relu'), # Increased units
    Dropout(0.5),
    Dense(512, activation='relu'),  # Added another dense layer
    Dropout(0.5),
    Dense(1, activation='sigmoid') # Sigmoid for binary classification
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
```

```
              metrics=['accuracy'])

model.summary()
```

Nowa architektura modelu zawiera większą liczbę warstw konwolucyjnych i neuronów w warstwach gęstych, aby model miał większą zdolność do uczenia się złożonych wzorców.

- dodatkowa warstwa konwolucyjna (Conv2D) z 256 filtrami, aby zwiększyć zdolność modelu do wykrywania bardziej złożonych cech w obrazach.
- zwiększona liczba jednostek w warstwach Dense, pierwsza warstwa Dense 1024 jednostki (zamiast 512), dołożyłam też drugą warstwę Dense z 512 jednostkami, ponieważ większa liczba neuronów w tych warstwach pozwala modelowi na naukę bardziej złożonych zależności.
- dodatkowa warstwa Dropout, aby jeszcze skuteczniej zapobiegać przetrenowaniu, szczególnie biorąc pod uwagę zwiększoną złożoność modelu.



Wykresy pokazują jednak, że pomimo zmian w architekturze, model wciąż nie uczy się skutecznie. Krzywe dokładności (training accuracy i validation accuracy) pozostają płaskie, oscylując wokół 50-55%, co jest praktycznie równoznaczne z losowym zgadywaniem. Podobnie, krzywe straty (loss i val_loss) są płaskie i nie wykazują spadku, co świadczy o braku uczenia. To sugeruje, że problem jest głębszy niż tylko początkowa architektura. Może to oznaczać, że:

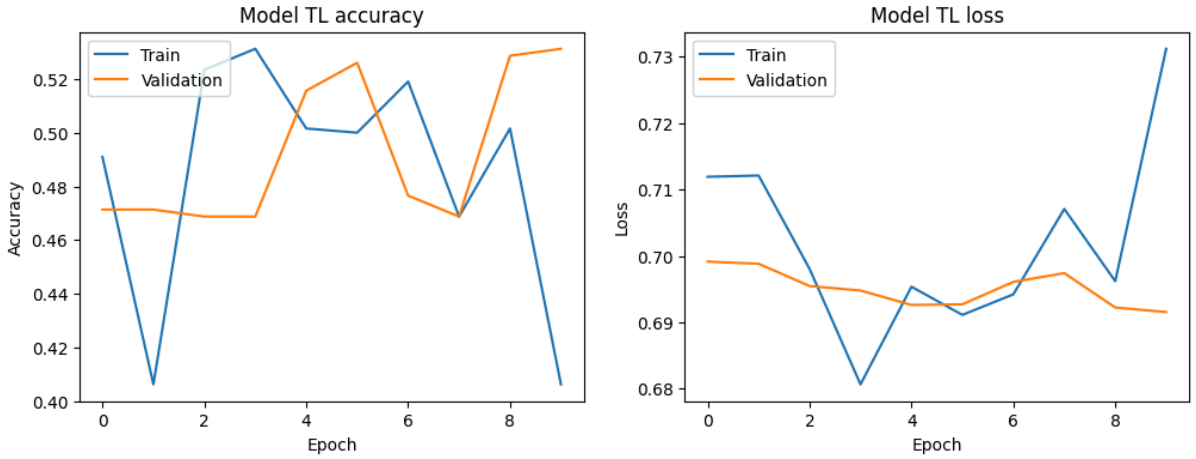
- dane są zbyt trudne do rozróżnienia dla modelu od zera, nawet z rozbudowaną architekturą.
- model jest zbyt płytki i wymaga jeszcze bardziej zaawansowanej struktury, która pozwoli mu uchwycić skomplikowane cechy obrazów.
- większa liczba epok może pomóc, ale przy tak płaskich krzywych, prawdopodobnie nie rozwiąże problemu całkowicie.

Druga próba polepszenia wyników dyskryminatora, poprzez implementację bardziej rozbudowanego modelu, wykorzystującego gotową, znaną sieć ResNet50:

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

# 1. Załadowanie wstępnie wytrenowanego modelu ResNet50
# include_top=False oznacza, że nie załadujemy ostatniej warstwy
klasyfikacyjnej
# weights='imagenet' oznacza użycie wag wytrenowanych na zbiorze ImageNet
base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(image_size, image_size, 3))

# 2. Zamrożenie warstw bazowego modelu
# To zapobiega aktualizowaniu wag już wytrenowanych na ImageNet
for layer in base_model.layers:
    layer.trainable = False

# 3. Dodanie nowych warstw klasyfikacyjnych na szczyt modelu bazowego
x = base_model.output
x = GlobalAveragePooling2D()(x) # Warstwa do spłaszczenia wyjścia z warstw
konwolucyjnych
x = Dense(1024, activation='relu')(x) # Pierwsza warstwa Dense
x = Dropout(0.5)(x) # Dropout dla regularyzacji
x = Dense(512, activation='relu')(x) # Druga warstwa Dense
x = Dropout(0.5)(x) # Drugi Dropout
predictions = Dense(1, activation='sigmoid')(x) # Warstwa wyjściowa dla
klasyfikacji binarnej

# Utworzenie nowego modelu
model_tl = Model(inputs=base_model.input, outputs=predictions)

# 4. Skompilowanie nowego modelu
model_tl.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 5. Wyświetlenie podsumowania nowego modelu
model_tl.summary()
```

Zaimplementowałam model ResNet50 jako bazę do transfer learningu, załadowałam wstępnie wytrenowany model ResNet50 z wagami ImageNet, ale bez warstw klasyfikacyjnych (bo te będę budować sama). Większość parametrów (Non-trainable params) została zamrożona (około 23,5 miliona), co oznacza, że są wykorzystywane nauczone cechy z ImageNet. Dodałam nowe warstwy

klasyfikacyjne (gęste warstwy z Dropoutem) na szczycie zamrożonej bazy, aby dopasować model do zadania binarnej klasyfikacji. To znacznie skróci czas treningu i zazwyczaj prowadzi do lepszych wyników.



Wykresy z treningu modelu ResNet50 z transfer learningiem niestety pokazują, że model nadal nie uczy się skutecznie. Krzywe dokładności i straty są płaskie, co oznacza, że model nie wykazuje poprawy w rozpoznawaniu obrazów. To, że nawet z tak potężnym modelem wstępnie wytrenowanym na ImageNet nie uzyskałam lepszych wyników, sugeruje, że problem może leżeć w dopasowaniu cech nauczonych na ImageNet do tego konkretnego zestawu danych (real/fake faces).

Kolejnym etapem pracy nad zwiększeniem dokładności modelu będzie wymiana datasetu. Może się okazać, że wybrany na ten moment dataset ma zbyt małą liczbę danych i dane są zbyt różnorodne, stąd słaby wynik uczenia.