

# Wydział Matematyki i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

---

## Projekt aplikacji generującej obraz w stylu słynnego malarza na podstawie podanego zdjęcia

---

*Autorzy:*

ANTONI ADASZKIEWICZ  
PATRYK BURZYCKI  
ADRIAN CIEŚLA  
MICHAŁ CZAPNIK  
ANITA CZECH

1 czerwca 2023

## Spis treści

1	Wstęp	2
1.1	Opis	2
1.2	Podział zadań	2
2	Wykorzystane narzędzia	3
3	Wybrane modele	3
3.1	SZUM	3
3.2	CycleGAN	3
4	Dane treningowe	5
4.1	SZUM	5
4.2	GAN	5
5	Proces uczenia i otrzymane efekty	6
5.1	SZUM	6
5.1.1	Góry	6
5.1.2	Lasy	7
5.1.3	Morza	7
5.1.4	Generowanie bezpośrednio obrazu w stylu Moneta	8
5.2	GAN	9
5.2.1	Monet	9
5.2.2	Van Gogh	11
5.2.3	Cezanne	12
5.2.4	Turner	13
5.2.5	Bob Ross	13
5.2.6	Ukiyo-e	14
5.3	Porównanie	15
6	Dodatkowe eksperymenty	15
6.1	Zdjęcie twarzy	15
6.2	Obraz	16
7	Aplikacja użytkownika	17
8	Podsumowanie	17
9	Bibliografia	18

# 1 Wstęp

Na Ziemi zdążyło pojawić się już wielu wybitnych malarzy. Ich dzieła do dziś podziwiane są przez rzesze ludzi i niejednokrotnie wskrzeszane przez kolejne pokolenia. Z pewnością niejeden człowiek zastanawiał się kiedyś, jak wyglądałyby dzieła danego malarza, gdyby urodził się w dzisiejszych czasach, bądź żył w zupełnie innym środowisku, niż oryginalnie. Jeszcze kilka lat temu znalezienie rozwiązania na tą zagwestkę wydawało się niemożliwe. Dziś możemy już tego dokonać z wykorzystaniem generowania obrazów za pomocą uczenia maszynowego i tym właśnie zajęliśmy się w tym projekcie.

## 1.1 Opis

Celem naszego projektu było stworzenie aplikacji umożliwiającej użytkownikowi wygenerowanie obrazu w stylu wybranego malarza na podstawie wgranego zdjęcia. By tego dokonać, wybrani zostali znani artyści o rozpoznawalnych stylach i na podstawie ich twórczości i dostępnych narzędzi wytrenowane zostały różne modele odpowiedzialne za tworzenie obrazów.

## 1.2 Podział zadań

### **Adrian Cieśla:**

1. Stworzenie aplikacji okienkowej
2. Wizualizacja wyników w aplikacji

### **Patryk Burzycki:**

1. Generator, dyskryminator GAN
2. Funkcje strat GAN
3. Wytrenowanie GAN

### **Anita Czech:**

1. Przygotowanie zbiorów treningowych
2. Wytrenowanie modeli

### **Michał Czapnik:**

1. Zbudowanie i wytrenowanie modelu do odszumiania

### **Antoni Adaszkiewicz:**

1. Generacja szumów danych
2. Odszumianie

## 2 Wykorzystane narzędzia

Język programowania, który użyliśmy, to Python (w wersji 3+). W celu wykonania projektu użyliśmy następujących narzędzi/bibliotek:

- tkinter - tworzenie okienka aplikacji
- customtkinter - stylizacja okienka aplikacji
- PIL - wykonywanie operacji na obrazach oraz ładowanie zdjęć
- tensorflow - przetwarzanie zdjęć oraz funkcjonalności uczenia maszynowego
- pyTorch - operacje na tensorach oraz sieciach neuronowych
- numpy - struktury/typy danych (zintegrowane z tensorflow)
- matplotlib - do wizualizacji wyników

## 3 Wybrane modele

### 3.1 SZUM

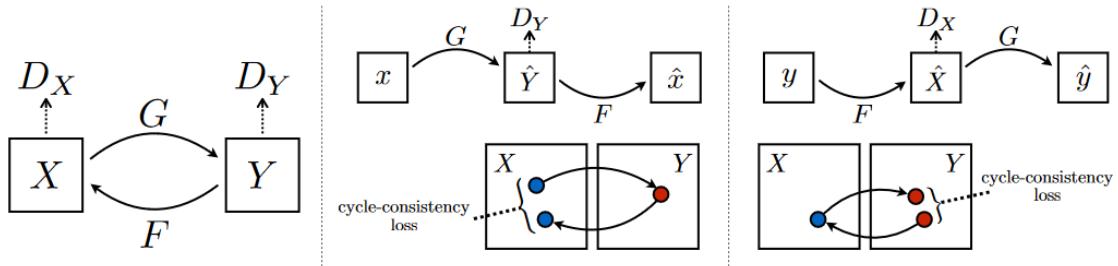
Modele szumowe jako pierwsze pojawiły się w pracy Sohl'a-Dickstein'a z 2015 roku "Deep Unsupervised Learning using Nonequilibrium Thermodynamics." Polegają one na odwróceniu procesu zaszumiania danych. Najpierw do danych iteracyjne dodawana jest mała ilość szumu, a następnie Model uczy się odwracać ten proces poprzez próbę oszacowania szumu dodanego w danym kroku. Dzięki iteracyjnej funkcji szumu model ten jest bardziej stabilny od poniżej opisanych modeli GAN, które przechodzą od szumu do ostatecznego wyniku w pojedynczym kroku.

### 3.2 CycleGAN

GAN (Generative Adversarial Network) to rodzaj architektury sztucznej sieci neuronowej, dzięki któremu komputery są w stanie tworzyć m.in. nowe obrazy. Idea ich działania, zaprezentowana w 2014 roku przez Ian'a Goodfellowa, opiera się na (co najmniej) dwóch sieciach neuronowych, konkurencyjnych ze sobą w formie gry o sumie zerowej. Jedną z tych sieci stanowi model generatora odpowiedzialny za tworzenie np. nowych zdjęć. Drugą jest model dyskryminatora, który stwierdza jak "realistyczne" są wygenerowane zdjęcia i czy są w stylu jak w podanym zbiorze treningowym. Oznacza to w skrócie, że generator uczy się jak oszukać dyskryminator, a dyskryminator jak coraz dokładniej rozróżnić prawdziwe przykłady danych od sztucznie wygenerowanych przez generator. Ostatecznym celem jest osiągnięcie równowagi, w której generator generuje bardzo wiarygodne przykłady danych, których dyskryminator nie jest w stanie odróżnić od prawdziwych danych.

CycleGAN to wariant (jeden z wielu istniejących) GANa, który umożliwia stworzenie naszej aplikacji. Jego architektura składa się z dwóch generatorów i dwóch dyskryminatorów, dzięki którym sieć uczy się przeprowadzać transformację pomiędzy dwoma domenami (np. dwoma zestawami zdjęć). Oznacza to, że model uczy się nie tylko generowania obrazów ze zdjęć, ale także tworzyć zdjęcia z obrazów. Dzięki temu oraz użyciu dodatkowej funkcji

strat po "domknięciu cyklu" można uzyskać cykliczną spójność między dwiema domenami. Oznacza to, że np. oryginalne zdjęcie, na podstawie którego wygenerowany został obraz, powinno być jak najbliżej identyczności, ze zdjęciem wygenerowanym z powrotem na podstawie tego obrazu. Dzięki temu wygenerowane wyniki zachowują cechy oryginalnych danych. Jest to szczególnie istotne w przypadku generowania obrazów na podstawie zdjęć, ponieważ chcemy, aby obraz przedstawiał wszystko to, co było na zdjęciu, jedynie przy niewielkich zmianach, dzięki którym wynikowy obraz będzie wyglądał rzeczywiście jak obraz namalowany przez danego artystę. Działanie cycleGAN zostało przedstawione na rysunku 1.



Rysunek 1: cycleGAN ( $D_x, D_y$  - dyskryminatory, G, F - generatory, X, Y - domeny danych)

## 4 Dane treningowe

### 4.1 SZUM

Dane treningowe służące do wytrenowania modelu generującego obraz z szumu zostały pozyskane z platformy Kaggle i zawierały różnego rodzaju krajobrazy podzielone na 5 klas:

- Wybrzeża
- Pustynie
- Lasy
- Lodowce
- Góry

Cały dataset został podzielony na 3 podkategorie: dane treningowe, walidacyjne i testowe.

### 4.2 GAN

Użyte w projekcie dane treningowe to zbiory obrazów, a dokładniej krajobrazów, autorstwa różnych malarzy. Każdy ze zbiorów użyty został do wytrenowania innego modelu odpowiadającego wybranemu artyście. Podczas ich wybierania zwracano uwagę na styl malowania artysty (czy był charakterystyczny, czy wyróżniał się w porównaniu z innymi), ilość namalowanych obrazów, a w szczególności pejzaży, a także ich dostępność. Ostatecznie przygotowano 6 zbiorów:

1. *Claude Monet* - francuski malarz, czołowy przedstawiciel impresjonizmu. W procesie uczenia wykorzystano dwa zbiory obrazów Moneta - jeden zawierał 300 obrazów malarza, drugi 1000. Przedstawiały one głównie łąki, wybrzeża, krajobrazy miejskie, często ukazywały ten sam widok w różnych porach roku.
2. *Vincent van Gogh* - holenderski malarz postimpresjonistyczny z XIX wieku. Do treningowania modelu wykorzystano 254 zdjęcia obrazów malarza. W tym zbiorze znajdują się obrazy łąk, rzek, a także miast i pól, namalowanych żywymi, wyrazistymi kolorami.
3. *Paul Cezanne* - postimpresjonistyczny malarz francuski. Zbiór składa się z 229 obrazów w stonowanych, głównie chłodnych kolorach, przedstawiających domy, drzewa.
4. *J.M.W. Turner* - angielski malarz, prekursor impresjonizmu. Najmniejszy wykorzystany zbiór, składa się bowiem z 178 obrazów przedstawiających m.in. krajobrazy morskie. Są to obrazy jasne, w stonowanych, dosyć bladych kolorach, z dużą przewagą żółci.
5. *Bob Ross* - amerykański malarz. W projekcie wykorzystano 250 obrazów namalowanych przez niego podczas programu "The Joy of Painting", którego był twórcą i prezenterem. Wykorzystane obrazy, namalowane wyrazistymi kolorami, przedstawiają głównie krajobrazy z górami.

6. *Ukiyo-e* - rodzaj malarstwa i drzeworytu japońskiego, funkcjonujący w okresie Edo. Na tle wybranych impresjonistycznych malarzy, wyróżnia się barwami i kreską. Wykorzystany zbiór składał się z 277 obrazów, przedstawiających m.in. fale, wybrzeża, góry. Na niektórych z nich znajdowały się ptaki i ludzie, co wpłynęło na otrzymane wyniki.

Zbiory Moneta, van Gogha, Cezanne'a i Ukiyo-e'a są częścią datasetu cycle\_gan oferowanego przez TensorFlow, z wyjątkiem jednego z modeli Moneta, dla którego dataset (zawierający 300 obrazów) poowany został z Kaggle datasets. Zbiór obrazów Boba Rossa to przekonwertowane do .jpg i przeskalowane do odpowiednich wymiarów zdjęcia zebrane przez Jared'a Wilber'a [referencja do literatury]. Zbiór obrazów Turnera został zebrany poprzez wybranie dostępnych w internecie obrazów i ich ręczne kadrowanie oraz przeskakowanie. Wszystkie zdjęcia były w formacie .jpg i wymiarach 256 na 256 pikseli.

Przy trenowaniu modeli dla każdego z wyżej wymienionych artystów/stylów, wykorzystywano zbiór zdjęć, na których modele uczyły się generować obrazy. Zbiór składał się z 6287 różnorodnych zdjęć przedstawiających m.in. budynki, samochody, plaże, lasy. Jednym modelem, który wykorzystywał inny zbiór zdjęć, był pierwszy model Moneta (300 obrazów malarza). W tym przypadku zbiór był większy i składał się z 7038 zdjęć.

## 5 Proces uczenia i otrzymane efekty

### 5.1 SZUM

Do trenowania modeli, ze względu na korzystną szybkość wykonywania kodu, wykorzystana została platforma Kaggle. Modele były trenowane przez 800 epok, ostateczny model jednak był wybrany spośród wersji modelu trenowanego przez 600 epok oraz wersji trenowanej do końca. Na zdjęciach możemy zaobserwować artefakty w postaci niespójnych elementów krajobrazu, np. drzewa wyrastające z ziemi, nad którymi znajduje się kolejna "lewitująca" warstwa ziemi, z której znowu wyrastają drzewa. W niektórych przypadkach modele te są w stanie wygenerować zdjęcia, które mogą uchodzić za prawdziwe. Poniżej zawarte są opisy i zdjęcia wygenerowane przez modele.

#### 5.1.1 Góry

Do generowania zdjęć gór wytrenowane zostały 2 modele, mountain-smooth i mountain-focused. Poniżej przedstawiono kilka z najlepszych wygenerowanych zdjęć.



(a) Zdjęcie 1



(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 2: wygenerowane zdjęcia górzystego terenu (smooth)



(a) Zdjęcie 1



(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 3: wygenerowane zdjęcia górz (smooth)

#### 5.1.2 Lasy

Do generowania zdjęć lasów wytrenowano 1 model. Poniżej przedstawiono kilka z najlepszych wygenerowanych zdjęć.



(a) Zdjęcie 1



(b) Zdjęcie 2

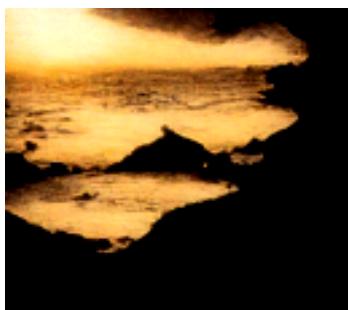


(c) Zdjęcie 3

Rysunek 4: wygenerowane zdjęcia lasów

#### 5.1.3 Morza

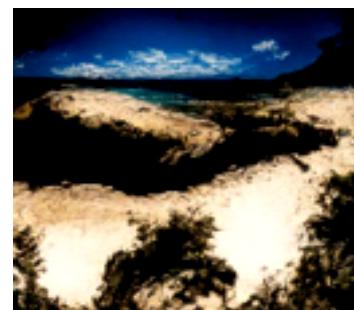
Do generowania zdjęć morza wytrenowane zostały 3 modele, seaside-dark, seaside-light i seaside-unstable. Poniżej przedstawiono kilka z najlepszych wygenerowanych zdjęć.



(a) Zdjęcie 1



(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 5: wygenerowane zdjęcia morza (dark)



(a) Zdjęcie 1



(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 6: wygenerowane zdjęcia morza (light)



(a) Zdjęcie 1



(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 7: wygenerowane zdjęcia morza (unstable)

#### 5.1.4 Generowanie bezpośrednio obrazu w stylu Moneta

Model szumowy do bezpośredniego generowania obrazów w stylu Moneta nie daje zadowalających efektów. Spowodowane jest to najprawdopodobniej zbyt małym rozmiarem treningowego datasetu (około 300 zdjęć). Dodatkowo obrazy treningowe zawierają zbyt różnorodną zawartość, co dodatkowo osłabia proces treningowy.



(a) Zdjęcie 1

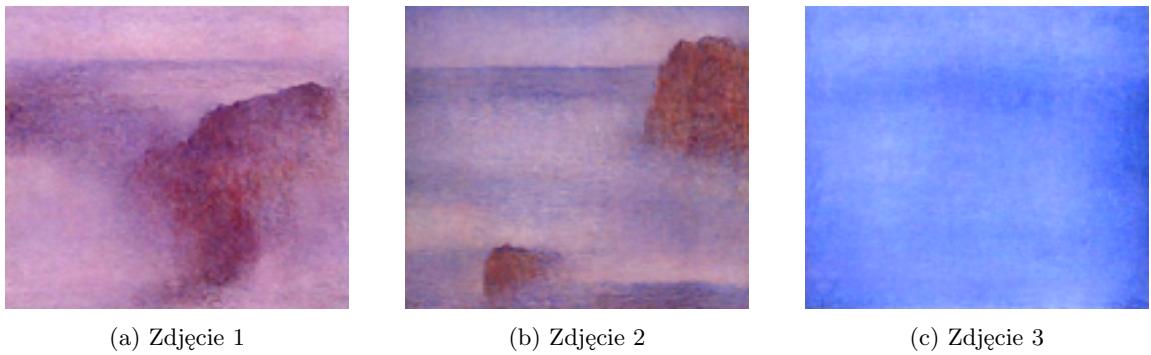


(b) Zdjęcie 2



(c) Zdjęcie 3

Rysunek 8: wygenerowane obrazy Moneta (1000 epok)



(a) Zdjęcie 1

(b) Zdjęcie 2

(c) Zdjęcie 3

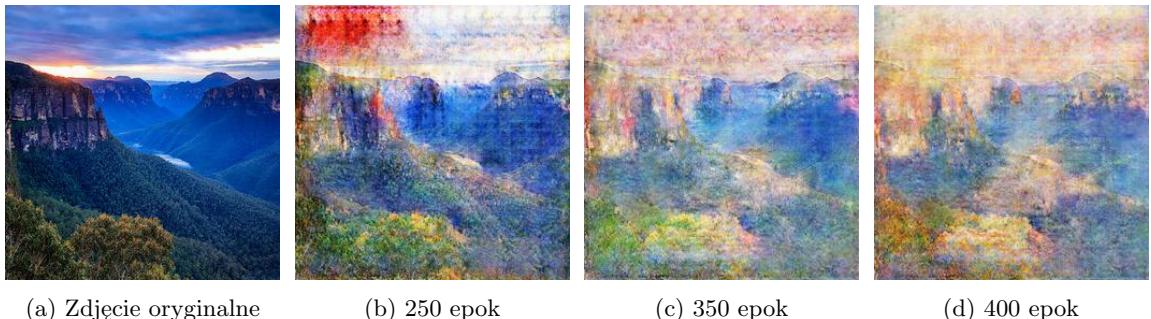
Rysunek 9: wygenerowane obrazy Moneta (1200 epok)

## 5.2 GAN

Do trenowania modeli, ze względu na korzystną szybkość wykonywania kodu, wykorzystana została platforma Kaggle. Z uwagi na różne wielkości zbiorów treningowych i ograniczony czas trwania jednej sesji, każdy z modeli był trenowany przez inną ilość epok. Dodatkowo sprawdzono wpływ tej ilości na jakość otrzymanych wyników. W tym celu zapisywane były stany generatorów i dyskryminatorów w trakcie uczenia (zazwyczaj co 100 epok). Poniżej zamieszczono opis rezultatów dla każdego stworzonego przez nas modelu.

### 5.2.1 Monet

Pierwszy model Moneta (dataset składający się z 300 zdjęć) trenowany był przez 400 epok. Na generowanych zdjęciach widoczne były artefakty, co skłoniło nas do zmienienia zbioru treningowego na większy, z którego usunięta została martwa natura, portrety i inne obrazy, pozostawiając w nim same pejzaże. Poniżej przedstawione zostały przykładowe obrazy wygenerowane na podstawie pierwszego modelu, z podziałem na ilość epok - 250, 350, 400.



(a) Zdjęcie oryginalne

(b) 250 epok

(c) 350 epok

(d) 400 epok

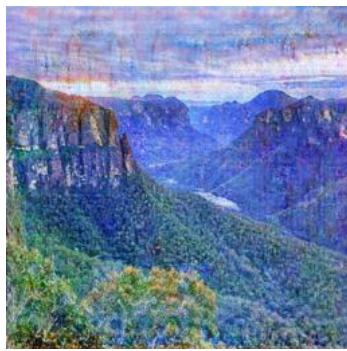
Rysunek 10: Różnice w generowanych obrazach w zależności od ilości epok

Jak widać na przedstawionych zdjęciach, z każdą kolejną epoką, zmieniają się kolory na wygenerowanym zdjęciu, a krawędzi są wygładzane.

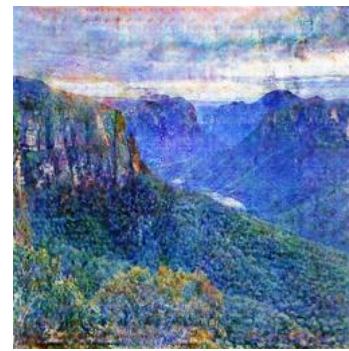
Drugi model, trenowany był przez 100 epok (na ich ilość wpływał rozmiar zbioru treningowego), a przykładowe wygenerowane zdjęcia zamieszczone poniżej.



(a) Zdjęcie oryginalne



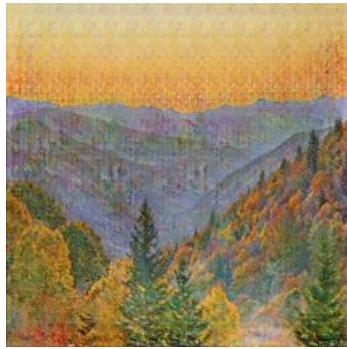
(b) 50 epok



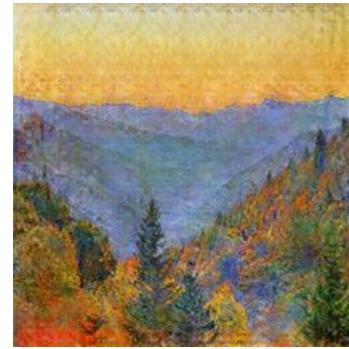
(c) 100 epok



(d) Zdjęcie oryginalne



(e) 50 epok



(f) 100 epok

Rysunek 11: Porównanie efektów w zależności od ilości epok dla drugiego modelu

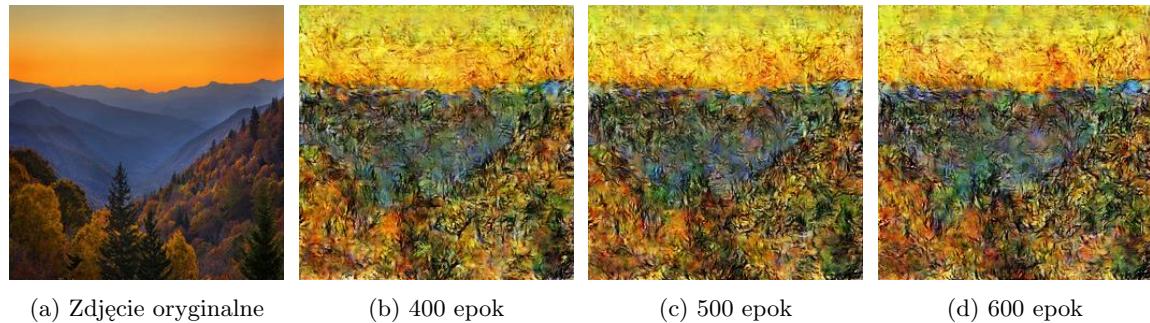
Z uwagi na niewielką różnicę wielkości porównywanych epok, wygenerowane zdjęcia są do siebie dosyć podobne.



Rysunek 12: Porównanie efektów dla obu modeli

### 5.2.2 Van Gogh

Model wykorzystany w aplikacji do generowania obrazów w stylu Van Gogh'a, trenowany był przez 600 epok. W trakcie trenowania zapisane zostały modele dla 400 i 500 epok, a efekty z nich otrzymane zostały przedstawione poniżej.



Rysunek 13: Różnice w generowanych obrazach w zależności od ilości epok

W przypadku modelu van Gogh, po wytrenowaniu modelu przez 400 epok, zwiększenie ich nie wpływa mocno na otrzymywane wyniki. Generowane obrazy są do siebie mocno zbliżone, czasami da się zauważać niewielkie różnice w śladzie "pędzla".



(a) Zdjęcie oryginalne (b) Wygenerowany obraz (c) Zdjęcie oryginalne (d) Wygenerowany obraz

Rysunek 14: Otrzymane efekty

Model daje zadowalające efekty. Choć nie zawsze tworzy obrazy zbliżone do zdjęcia wejściowego, to dobrze naśladuje styl malarza i kolorystykę jego obrazów, w której przeważają odcienie żółci.

### 5.2.3 Cezanne

Model generujący obrazy w stylu Cezanne był trenowany przez 600 epok.



(a) Zdjęcie oryginalne (b) 400 epok (c) 500 epok (d) 600 epok

Rysunek 15: Różnice w generowanych obrazach w zależności od ilości epok

Choć różnice są mało zauważalne, to można zwrócić uwagę na zwiększającą się ilość "plam" farb wraz ze wzrostem ilości epok. Dodatkowo - jak widać na drugim zestawie zdjęć (d,e,f) przy 600 epokach zaczyna zanikać linia przechodząca przez środek obrazu.

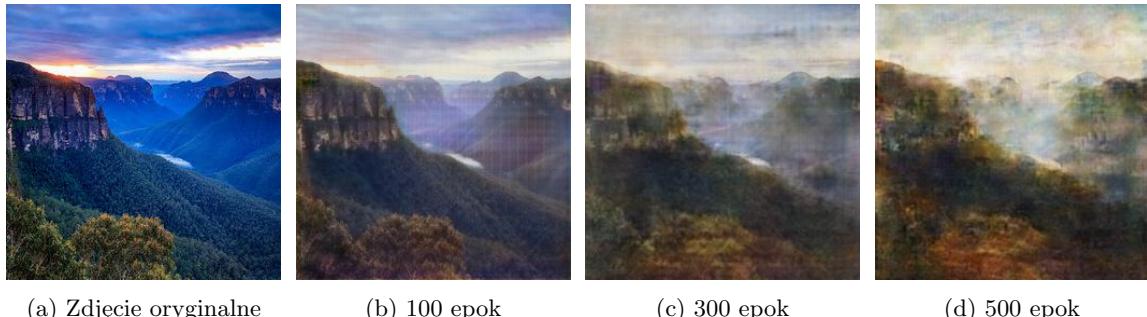


(a) Zdjęcie oryginalne (b) Wygenerowany obraz (c) Zdjęcie oryginalne (d) Wygenerowany obraz

Rysunek 16: Otrzymane efekty

#### 5.2.4 Turner

Model Turnera trenowany był przez 500 epok.



(a) Zdjęcie oryginalne      (b) 100 epok      (c) 300 epok      (d) 500 epok

Rysunek 17: Różnice w generowanych obrazach w zależności od ilości epok

Wraz ze zwiększającą się ilością epok zdjęcie jest rozmazywane, zmieniają się także ich kolory. Tracą swoją jaskrawość, na wygenerowanym obrazie są znacznie bardziej stonowane.

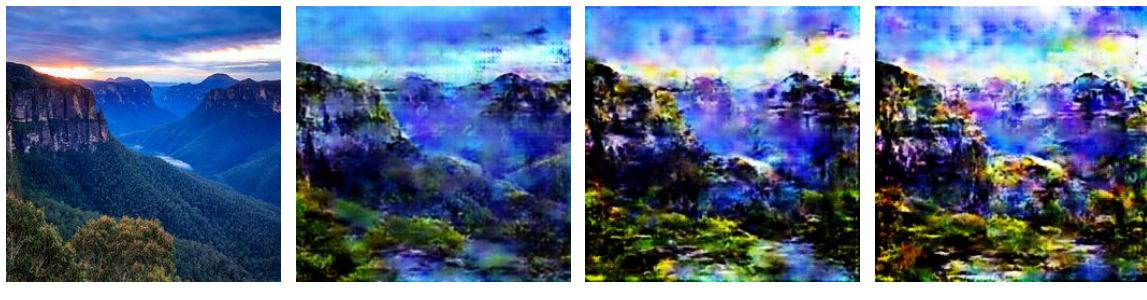


(a) Zdjęcie oryginalne    (b) Wygenerowany obraz    (c) Zdjęcie oryginalne    (d) Wygenerowany obraz

Rysunek 18: Otrzymane efekty

#### 5.2.5 Bob Ross

Model trenowany był, podobnie jak model Cezanne, przez 600 epok.



(a) Zdjęcie oryginalne    (b) 200 epok    (c) 400 epok    (d) 600 epok

Rysunek 19: Różnice w generowanych obrazach w zależności od ilości epok

W porównaniu z poprzednimi modelami różnice pomiędzy epokami są bardziej widoczne. Przy wyższych epokach, kolory są żywksze, zwiększyony jest też między nimi kontrast.

### 5.2.6 Ukiyo-e

Początkowo model generujący obrazy w stylu Ukiyo-e był trenowany przez 50 epok na zbiorze danym zawierającym 825 zdjęć. Z uwagi na występowanie w nim obrazów zwierząt, portretów, otrzymane rezultaty nie były zadowalające. Przykładowe generowane zdjęciem zamieszczone poniżej.

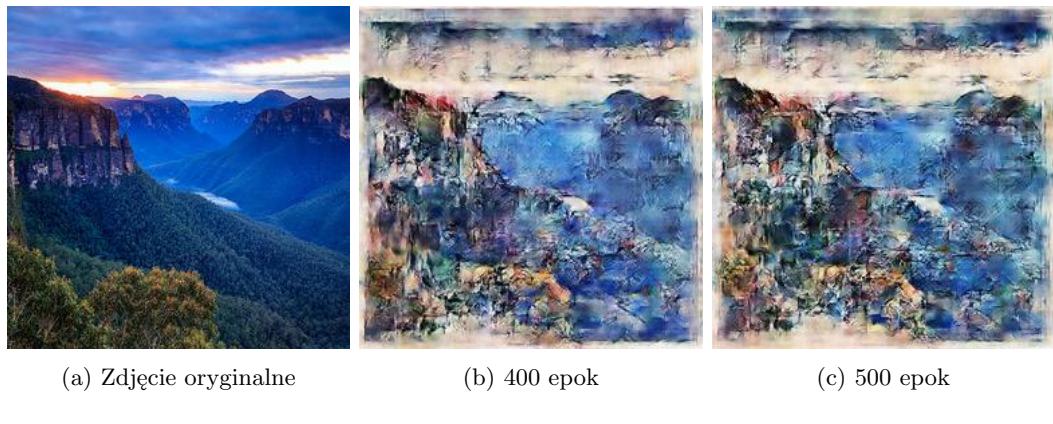


(a) Zdjęcie oryginalne

(b) Wygenerowany obraz

Rysunek 20: Otrzymane efekty

W wygenerowanym zdjęciu pojawiły się nieoczekiwane przez nas ptaki. Z uwagi na takie i inne, podobne artefakty, dataset został znacznie zmniejszony. Do modelu wykorzystanego w aplikacji wykorzystano zbiór opisany w sekcji 4.2. Sam model trenowany był przez 500 epok.



(a) Zdjęcie oryginalne

(b) 400 epok

(c) 500 epok

Rysunek 21: Różnice w generowanych obrazach w zależności od ilości epok

Dla porównywanych epok, 500 i 600 nie ma widocznych różnic w generowanym obrazie.



(a) Zdjęcie oryginalne (b) Wygenerowany obraz (c) Zdjęcie oryginalne (d) Wygenerowany obraz

Rysunek 22: Otrzymane efekty

Kolory obiektów na wygenerowanych obrazach są wyblaknięte i bardziej jednolite. Zmienia się także kolor tła, co widać na obrazie (d) - niebieskie niebo zostało zamienione beżowym tłem.

### 5.3 Porównanie

Należy zauważyć, że choć przedstawione w poprzednich sekcjach wygenerowane zdjęcia są zadowalającej jakości, to otrzymywane efekty zależą od zdjęcia podanego na wejściu. Zdarza się, że nie jest ono w żaden sposób przetwarzana, niektóre generatory tworzą czasem artefakty i nie zawsze jest ono podobne do oryginalnego zdjęcia.

## 6 Dodatkowe eksperymenty

Rozdział ten jest poświęcony na próby wykorzystania modeli w celach innych niż zamierzonych, oraz zaobserwowania efektów. Modele zostały wytrenowane na obrazach przedstawiających krajobrazy, sprawdzone zostały sytuacje, w których model potencjalnie może sobie nie radzić.

### 6.1 Zdjęcie twarzy



Rysunek 23: Oryginał



Rysunek 24: Monet



Rysunek 25: Monet



Rysunek 26: Cezanne



Rysunek 27: Cezanne



Rysunek 28: Ross



Rysunek 29: Turner

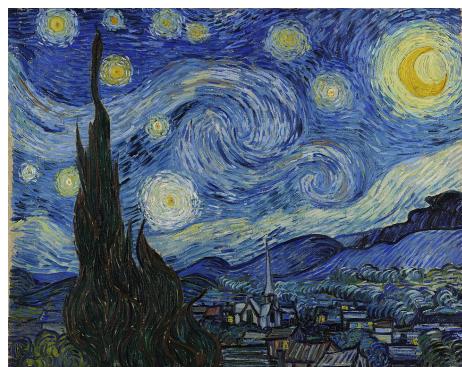


Rysunek 30: Ukiyoe



Rysunek 31: Gogh

## 6.2 Obraz



Rysunek 32: Oryginał



Rysunek 33: Monet



Rysunek 34: Monet



Rysunek 35: Cezanne



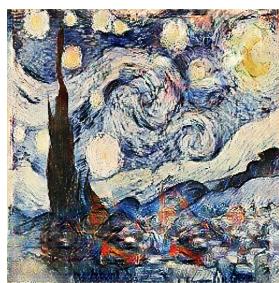
Rysunek 36: Cezanne



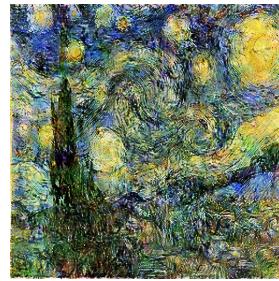
Rysunek 37: Ross



Rysunek 38: Turner



Rysunek 39: Ukiyo-e



Rysunek 40: Van Gogh

## 7 Aplikacja użytkownika

Aplikacja użytkownika służy do graficznej wizualizacji efektów uczenia maszynowego, nad którym pracowaliśmy w tym projekcie. Składa się ona z dwóch okienek:

- pokazującego oryginalny stan wczytanego zdjęcia
- pokazującego stan wczytanego zdjęcia uzyskany przez przetworzenie za pomocą modelu GAN

We wczytaniu zdjęcia pomagają znajdujące się pod okienkami przyciski, które umożliwiają dostosowanie najodpowiedniejszej dla użytkownika opcji, między innymi: wybór malarza, na podstawie którego generowany będzie obraz wynikowy, zapisanie wynikowego zdjęcia na dysku komputera, wczytanie własnego zdjęcia, bądź wygenerowanie go z szumu. W przypadku tego ostatniego, użytkownik ma możliwość wyboru modelu, który będzie odpowiedzialny za wygenerowanie obrazu końcowego.

## 8 Podsumowanie

W ramach projektu stworzona została aplikacja okienkowa umożliwiająca wygenerowanie zdjęcia krajobrazu lub wgranie własnego zdjęcia, oraz przetworzenie go, aby imitował obraz namalowany przez sławnego artystę. Opracowana aplikacja nie tylko dostarcza narzędzi do generowania unikalnych i artystycznych obrazów, ale również zapewnia łatwą i intuicyjną obsługę dla użytkowników. Interfejs graficzny umożliwia wybór zdjęcia, wybranie preferowanego stylu i przetworzenie go w czasie rzeczywistym, co daje natychmiastowy podgląd na efekty. Dzięki temu nawet osoby nieposiadające specjalistycznej wiedzy z dziedziny grafiki komputerowej mogą korzystać z aplikacji i tworzyć wyjątkowe obrazy.

Do generowania obrazów został wykorzystany model szumowy, nauczony na zbiorach danych przedstawiających odpowiednie rodzaje krajobrazów. Modele te generują jednak obrazy o rozmiarze zbyt małym, dlatego, aby zwiększyć rozdzielcość, używany jest model MDSR do upscalingu.

Do przekształcania na obrazach został wykorzystany model GAN'owy, nauczony przy pomocy danych zdjęć obrazów konkretnych malarzy.

Ostateczne wyniki projektu potwierdzają, że opracowane modele i aplikacja spełniają oczekiwania projektowe, dostarczając nowoczesne i innowacyjne narzędzia do przetwarzania obrazów.

## 9 Bibliografia

1. [https://en.wikipedia.org/wiki/Generative\\_adversarial\\_network](https://en.wikipedia.org/wiki/Generative_adversarial_network)
2. <https://oioit.pl/gan-czyli-jak-nauczyc-komputer-kreatywnego-myslenia/>
3. [https://en.wikipedia.org/wiki/J.\\_M.\\_W.\\_Turner](https://en.wikipedia.org/wiki/J._M._W._Turner)
4. <https://www.wikiart.org/>
5. [https://github.com/jwilber/Bob\\_Ross\\_Paintings](https://github.com/jwilber/Bob_Ross_Paintings)
6. <https://arxiv.org/pdf/1703.10593.pdf>