

Sprawozdanie i specyfikacja projektu

Ganics

Adrian Chmiel

9 czerwca 2024

Link do repozytorium: <https://github.com/chmieladr/Ganics>

1 Cel projektu

Celem projektu jest wykorzystanie sieci GAN do generowania obrazów z przetłumaczonym stylem. Program oferuje dwie możliwe opcje konwersji stylu:

1. **Sztuka Vincenta Van Gogha** - obrazy będą przypominać dzieła Vincenta Van Gogha
2. **Kreskówka** - obrazy będą przypominać obrazy z kreskówek (lub komiksów)

Po przetłumaczeniu stylu obraz jest upscalowany do rozdzielczości $512x512$. Ostateczny obraz jest zapisywany w formacie `.png`.

2 Wykorzystane technologie

Program został napisany w języku **Python 3.10**. Wykorzystuje następujące biblioteki:

- **matplotlib** - do tworzenia wykresów i wizualizacji danych
- **numpy** - do operacji na macierzach
- **opencv-python** - do przetwarzania obrazów
- **pandas** - do analizy danych
- **pillow** - do przetwarzania obrazów
- **pydot** - do wizualizacji struktury modelu
- **tensorflow** - do tworzenia i trenowania modeli GAN
- **tkinter** - do stworzenia prostego interfejsu graficznego
- **tqdm** - do wyświetlania postępu w trenowaniu modelu

Wybrałem język programowania Python, ponieważ jest powszechnie używany w dziedzinie uczenia maszynowego i sztucznej inteligencji. Oferuje szeroki zakres bibliotek i narzędzi, które ułatwiają realizację takich projektów. Spośród wielu bibliotek do uczenia maszynowego wybrałem **Tensorflow**, ponieważ jest jedną z najpopularniejszych bibliotek do tworzenia sieci neuronowych i korzystałem z niej już wcześniej w ramach innych projektach.

Dla tego projektu zdecydowałem się na użycie wersji **Python 3.10** głównie dlatego, że jest to ostatnia wersja wspierana przez **Tensorflow 2.10**. Nowe wersje tej biblioteki nie obsługują natywnie akceleracji GPU na systemie operacyjnym **Windows**, co jest kluczowe dla procesu uczenia, ponieważ znacznie skraca czas potrzebny na trenowanie sieci. Mimo że mógłbym ograniczyć projekt do systemów **Linux**, chciałem uczynić go bardziej dostępnym.

Oprócz tego wykorzystałem **Jupyter Notebooks** do trenowania modeli. Ich główną zaletą jest możliwość uruchamiania kodu w małych fragmentach, co ułatwia znalezienie problemów i zrozumienie kodu. Są dostępne w katalogu `notebooks`.

3 Uruchomienie programu

Aby uruchomić aplikację bezpośrednio z **kodu źródłowego**, zaleca się utworzenie nowego środowiska wirtualnego i zainstalowanie wymaganych bibliotek/pakietów w nim. Zakładając, że korzystamy z `condy`, możemy to zrobić za pomocą następujących poleceń:

```
conda create -n Ganics python=3.10 -y  
conda activate Ganics  
conda install jupyter -y  
pip install -r requirements.txt
```

Instalację Jupyter można pominąć, jeśli nie planujemy korzystać z notebooków.

Dla posiadaczy kart graficznych *NVIDIA* warto również uruchomić poniższe polecenie, które umożliwi korzystanie z *CUDA* (pod warunkiem, że zainstalowano inne niezbędne oprogramowanie):

```
conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0 -y
```

Program można uruchomić, wywołując plik **main.py**:

```
python src/main.py
```

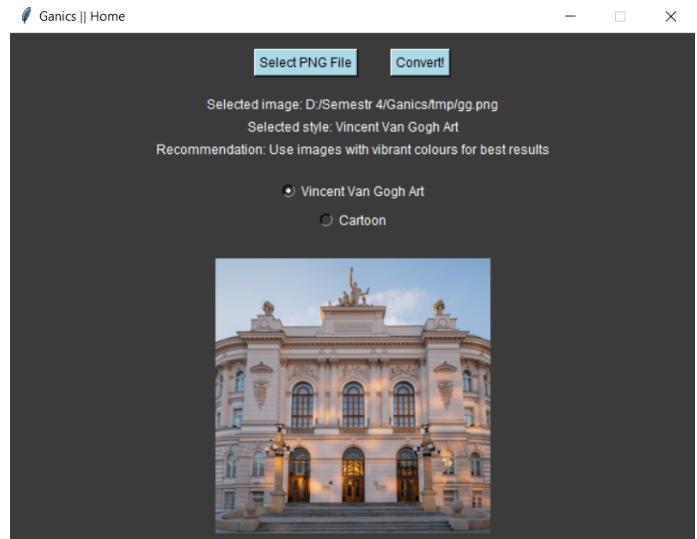
Program został przetestowany na systemach operacyjnych *Windows* i *Linux*.

4 Interfejs graficzny

Interfejs graficzny programu został stworzony przy użyciu biblioteki *tkinter*. Pozwala on na:

- Wybór pliku wejściowego
- Wybór stylu konwersji
- Konwersję i zapisanie obrazu
- Podgląd obrazu przed i po konwersji

Przykładowy zrzut ekranu interfejsu użytkownika jest widoczny po prawej stronie.



5 Pliki wejściowe i wyjściowe

1. Aplikacja główna

Program pozwala na wybranie ścieżki do pliku wejściowego, który zostanie przekonwertowany. Warto zauważyć, że plik wejściowy musi być obrazem w formacie *.jpg*, *.jpeg* lub *.png*.

Plik wyjściowy, niezależnie od początkowego formatu, zawsze jest obrazem w formacie *.png* o rozdzielcości *512x512*.

2. Notebooki

W przypadku notebooków dostarczamy całe zestawy obrazów jako dane wejściowe (więcej informacji poniżej), które zostaną wykorzystane do trenowania modelu. Jako wynik otrzymujemy wizualizacje ilustrujące działanie modelu po trenowaniu oraz wyeksportowany plik wag modelu, który jest później wykorzystywany jako składnik aplikacji głównej.

6 Działanie aplikacji

Poniżej przedstawiono koncepcję działania aplikacji:

1. Wybór pliku i stylu konwersji
2. Przetłumaczenie stylu obrazu za pomocą **CycleGAN**
Otrzymany obraz ma rozdzielcość 256x256.
3. Upscaling obrazu za pomocą **SRGAN** do rozdzielcości 512x512

4. Zapisanie obrazu w formacie `.png`

Ważnym faktem jest, że aplikacja **nie będzie działać**, jeśli nie znajdzie odpowiednich wag modelu, o czym użytkownik zostanie poinformowany błędem. W takim przypadku konieczne jest albo uruchomienie notebooków do trenowania modelu, albo pobranie gotowych plików dostępnych na *Google Drive* pod linkiem w pliku *how_to_get_models.md*.

7 Teoria

Program opiera się na dwóch modelach sieci GAN:

- **CycleGAN**
- **SRGAN** (z dyskryminatorem *PatchGAN*)

GAN (*Generative Adversarial Network*) to rodzaj sztucznej sieci neuronowej, która składa się z dwóch modeli: **generatora** i **dyskryminatora**. Zadaniem generatora jest generowanie nowych obrazów podobnych do tych z zestawu danych, podczas gdy zadaniem dyskryminatora jest rozróżnianie, czy obraz pochodzi od generatora, czy od zestawu danych. Proces trenowania polega na rywalizacji obu modeli, gdzie generator próbuje oszukać dyskryminator, a dyskryminator próbuje nauczyć się rozróżniania obrazów. W rezultacie generator jest w stanie generować obrazy bardzo podobne do tych z zestawu danych.

8 Zbiory danych

Aby uzyskać zbiory danych do trenowania modeli, można wypakować dostępne pliki `.zip`, które zawierają obrazy przeze mnie wybrane z poniższych zbiorów danych. Alternatywną opcją jest skorzystanie z linków do tych zbiorów dostępnych w pliku *README.md*. Dzięki temu podejściu można samodzielnie wybrać obrazy, które zostaną wykorzystane do trenowania.

Zbiory danych wykorzystane do trenowania modeli to:

- **Familyguy** (z zestawu danych Cartoon Classification)
Użyty do trenowania CycleGAN odpowiedzialnego za konwersję stylu na kreskówkę (lub komiks).
- **VincentVanGogh** (z zestawu danych Vincent Van Gogh Art)
Użyty do trenowania kolejnego CycleGAN odpowiedzialnego za konwersję stylu na dzieła Vincenta Van Gogha.
- **natural_images**
Przykładowe obrazy, których styl został przekształcony na jeden z dwóch wyżej wymienionych podczas trenowania modeli.
- **mscoco** Zbiór danych, który dodatkowo rozszerzył zestaw danych z obrazami naturalnymi wykorzystanymi do trenowania modeli.

9 Ograniczenia

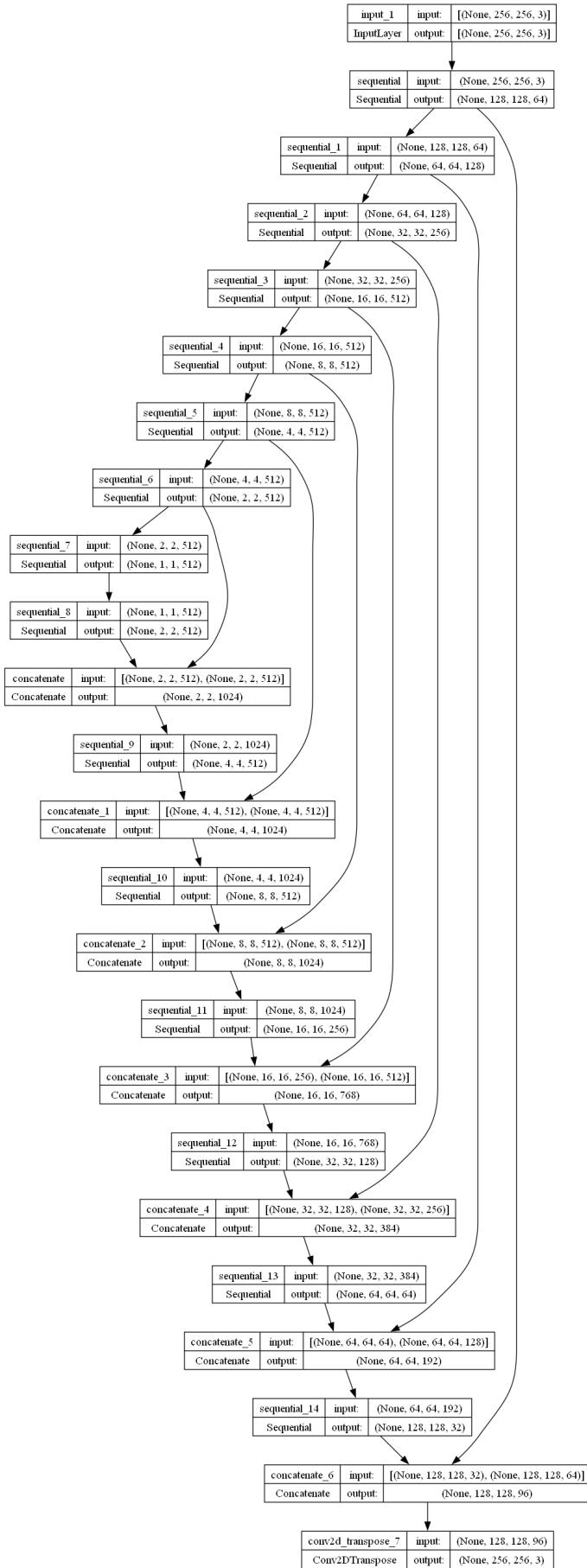
Konwersja stylu jest zadaniem stosunkowo skomplikowanym, dlatego czas trenowania w notebookach silnie zależy od mocy obliczeniowej komputera. Z tego powodu zaleca się korzystanie z karty graficznej *NVIDIA*, która znacznie przyspiesza proces nauki. Wszystkie notebooki są dostosowane do korzystania z *CUDA*, jeśli jest dostępne, czyli wymagane oprogramowanie zostało zainstalowane.

Warto również zauważać, że styl kreskówki jest stosunkowo prosty, więc model podczas konwersji obrazów obiektów może być niezadowalający.

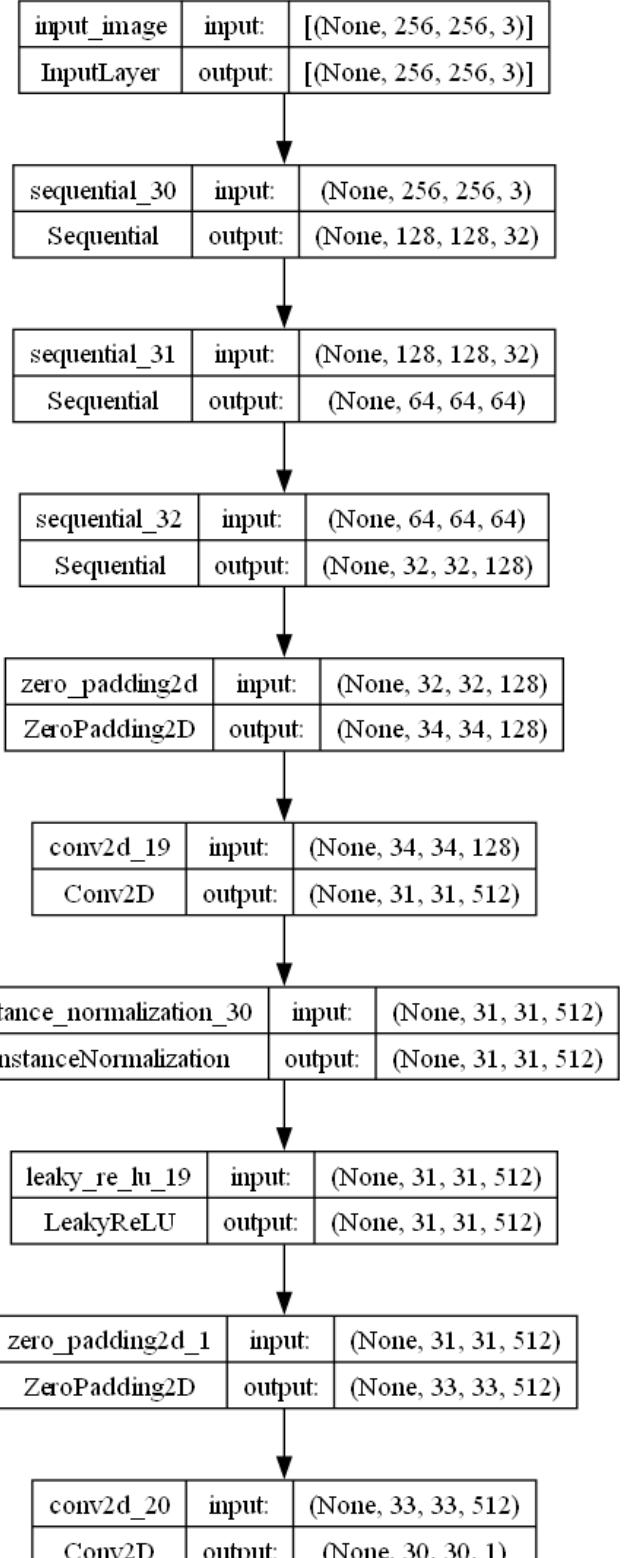
10 Wizualizacja modeli

Na kolejnych stronach widoczne są wizualizacje modeli wykorzystanych w aplikacji. Wizualizacje te zostały wygenerowane za pomocą biblioteki *pydot*. Pokazują zarówno generator, jak i dyskryminator obu modeli, wraz z poszczególnymi warstwami i ich połączeniami. Więcej informacji dostępnych jest w dokumencie *Notes.pdf*.

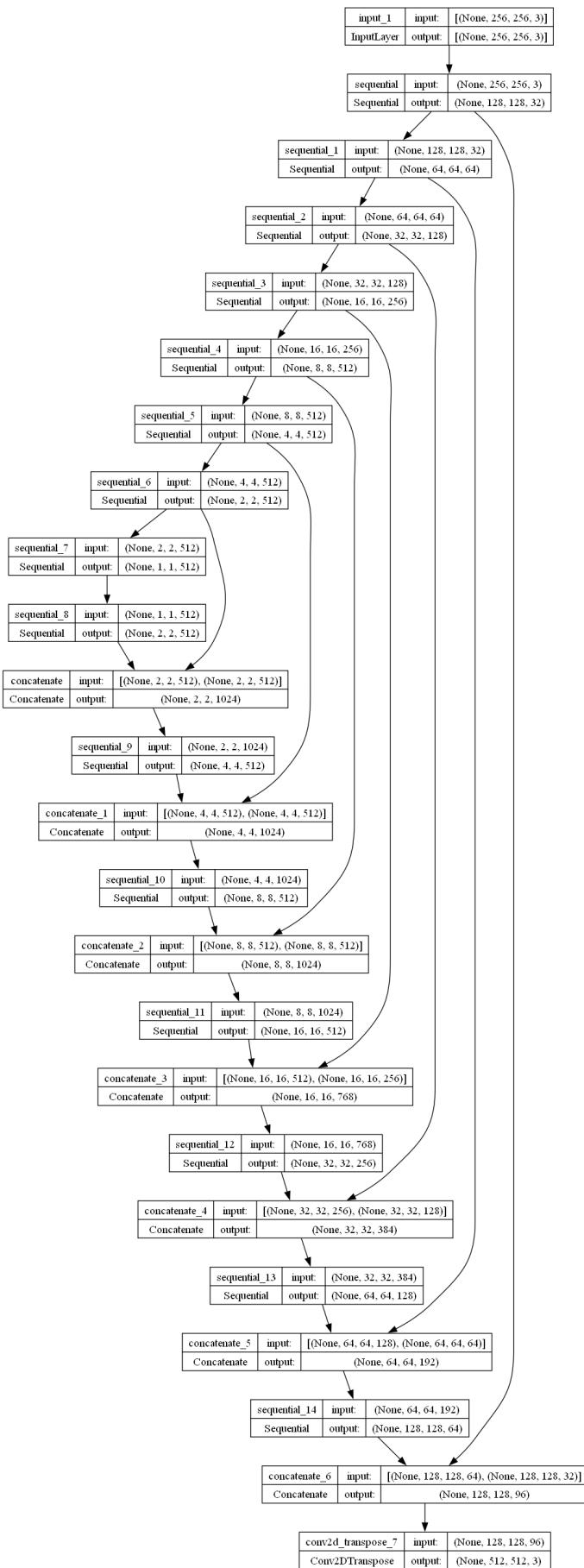
10.1 CycleGAN - Generator



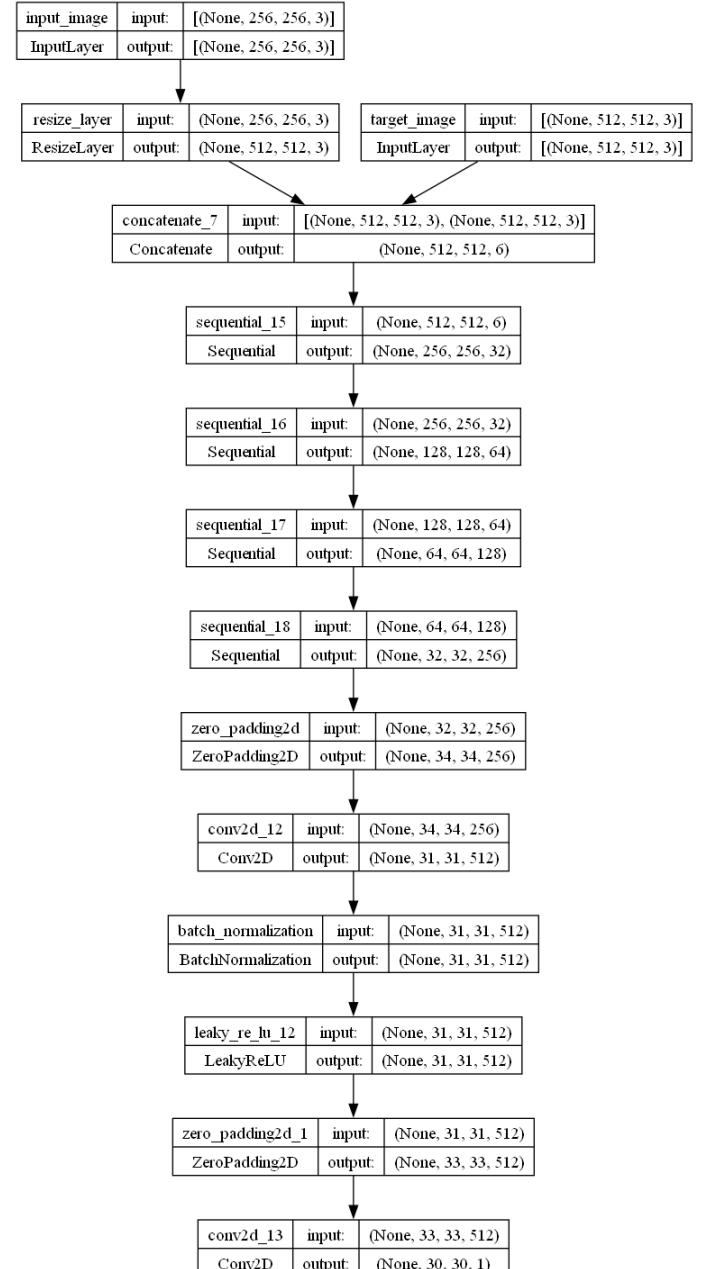
10.2 CycleGAN - Dyskryminator



10.3 SRGAN - Generator



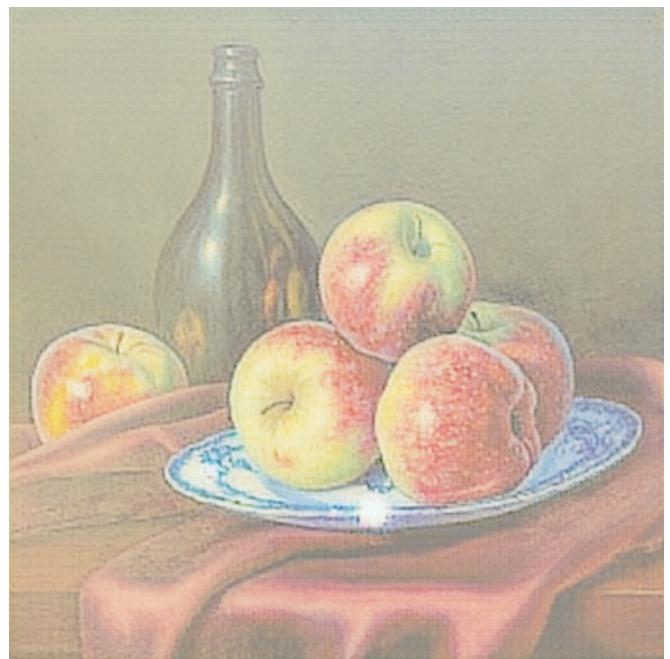
10.4 SRGAN - Dyskryminator (PatchGAN)

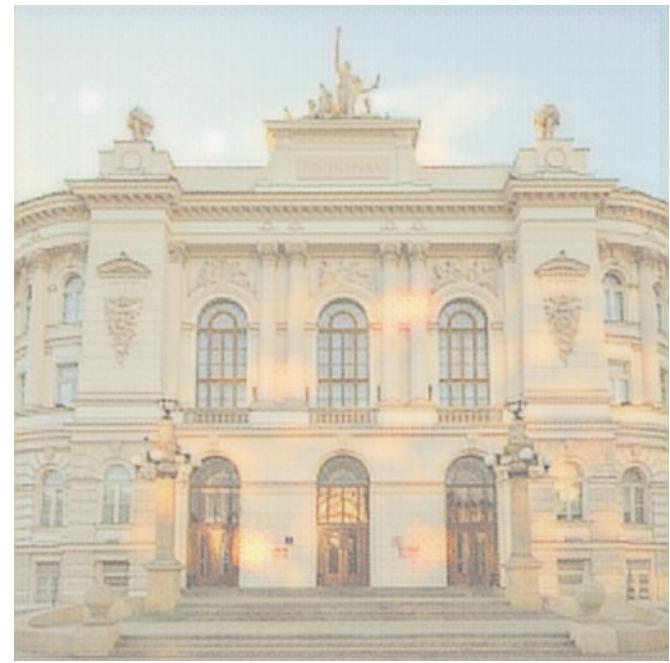


11 Przykładowe wyniki

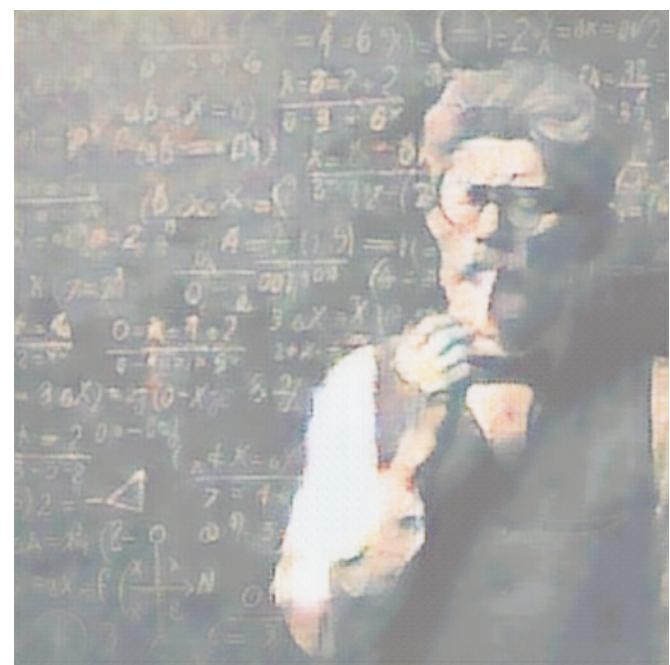
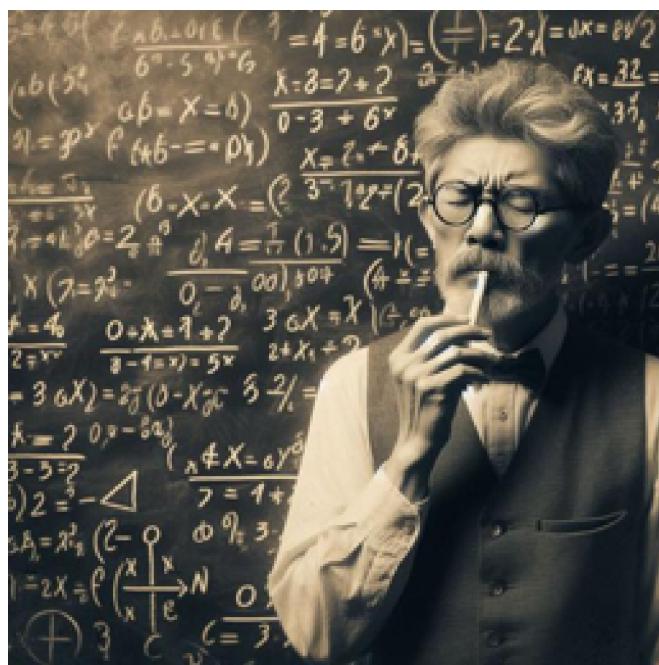
Poniżej można zobaczyć przykładowe obrazy wyeksportowane z aplikacji. Obrazy po lewej stronie to obrazy wejściowe, a obrazy po prawej stronie to wyniki konwersji.

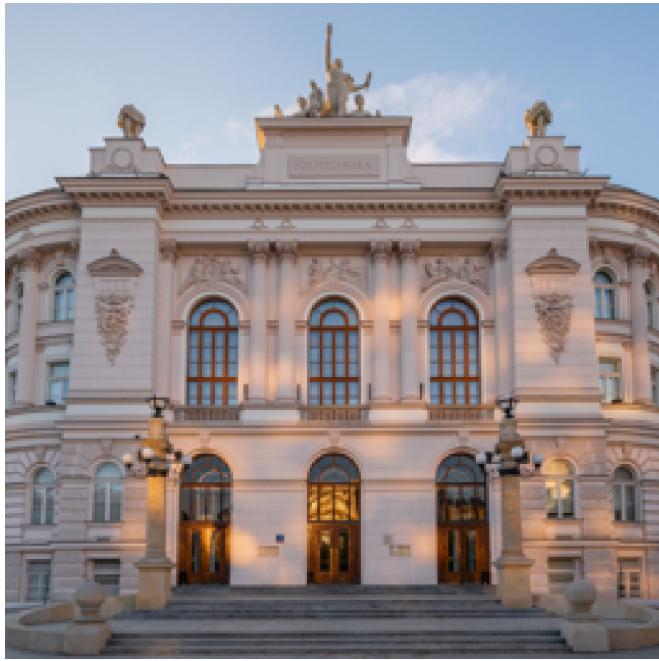
11.1 Sztuka Vincent Van Gogha





11.2 Kreskówka





12 Wnioski

- Program umożliwia konwersję obrazów na dwa różne style: sztuka Vincenta Van Gogha i kreskówka.
- Używa dwóch modeli: CycleGAN do konwersji stylu i SRGAN do skalowania obrazu.
- Program został wyposażony w graficzny interfejs, który umożliwia łatwy wybór pliku wejściowego i stylu konwersji.
- Aplikacja wymaga dostępności wag modelu, które można uzyskać, uruchamiając notebooki lub pobierając je z podanego linku.
- Program działa na systemach operacyjnych Windows i Linux.
- Właściwy wybór liczby epok jest kluczowy dla jakości finalnego obrazu. Ważne jest, aby patrzeć na wyniki i monitorować strategię podczas treningu, aby uniknąć przeuczenia.