Specyfikacja funkcjonalna

Ganics

Adrian Chmiel

4 czerwca 2024

1 Cel projektu

Celem projektu jest wykorzystanie sieci GAN do generacji obrazów ze zmienionym stylem. Program oferuje dwa możliwe style konwersji:

1. **Vincent Van Gogh Art** - obrazy będą przypominać dzieła sztuki Vincenta Van Gogha
2. **Cartoon** - obrazy będą przypominać zdjęcia pochodzące z kreskówki, bądź komiksu

2 Informacje ogólne

Program został napisany w języku Python w wersji 3.10. Program korzysta z następujących bibliotek:

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

3 Uruchamianie programu

Aby uruchomić aplikację bezpośrednio z **kodu źródłowego**, zalecanym podejściem jest utworzenie nowego wirtualnego środowiska oraz zainstalowanie w nim wymaganych bibliotek/pakietów. Zakładając, że korzystamy z *conda* możemy to wykonać następującymi komendami:

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

Instalację jupytera można pominąć, jeśli nie zamierzamy korzystać z notebooków.

Dla posiadaczy kart graficznych NVIDIA warto również wykonać następującą komendę, która umożliwi korzystanie z CUDA:

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

Program można uruchomić poprzez wywołanie pliku ***main.py***:

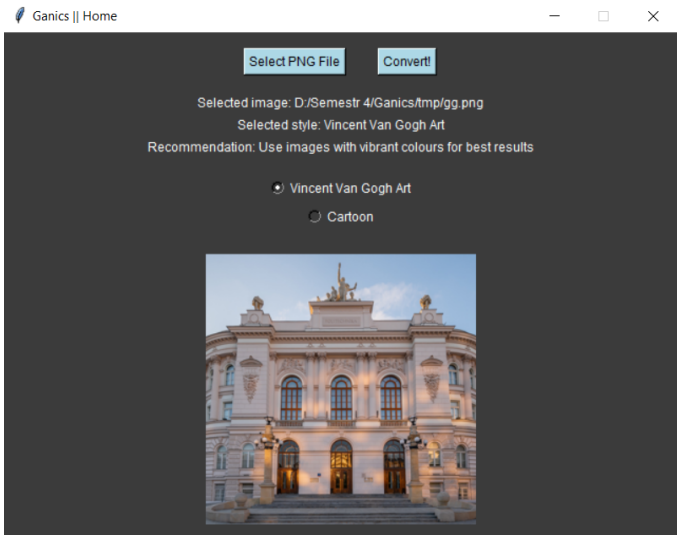
```
python src/main.py
```

4 Interfejs graficzny

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

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

Przykładowy zrzut ekranu interfejsu graficznego znajduje się poniżej:



5 Pliki wejściowe oraz wyjściowe

1. **Główna aplikacja**
   Program umożliwia wybranie ścieżki do pliku wejściowego, który będzie poddany konwersji. Warto zaznaczyć, że plik wejściowy musi być obrazem w formacie *.jpg*, *.jpeg* lub *.png*.

   Natomiast plikiem wyjściowym niezależnie od początkowego formatu jest zawsze obraz w formacie *.png* o rozdzielczości *512x512*.

2. **Notebooki**
   W przypadku notebooków jako wejście podajemy całe zbiory obrazów (więcej informacji poniżej), które zostaną wykorzystane do nauki modelu. Jako wyjście otrzymujemy wizualizacje obrazujące, jak model sprawuje się po zakończeniu nauki oraz wyeksportowany plik z wagami modelu, który później jest wykorzystywany jako komponent głównej aplikacji.

6 Sposób działania aplikacji

Zamysł działania aplikacji widoczny jest poniżej:

1. Wybór pliku oraz stylu konwersji
2. Translacja stylu obrazu przy wykorzystaniu **CycleGAN**
   Obraz powstały w tym procesie ma rozdzielczość 256x256.
3. Upscaling obrazu przy wykorzystaniu **SRGAN** do rozdzielczości 512x512
4. Zapisanie obrazu w formacie *.png*

Istotnym faktem jest, że aplikacja **nie zadziała**, jeżeli nie znajdzie odpowiednich wag modelu, o czym użytkownik zostanie poinformowany błędem. W takim wypadku należy uruchomić notebooki, aby nauczyć model, bądź pobrać gotowe pliki dostępne na *Google Drive* pod linkiem znajdującym się 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** oraz **dyskryminatora**. Generator ma za zadanie generować nowe obrazy, które są podobne do tych ze zbioru danych, natomiast dyskryminator ma za zadanie rozróżniać, czy obraz pochodzi z generatora, czy ze zbioru danych. Proces szkolenia polega na rywalizacji obu modeli, gdzie generator stara się oszukać dyskryminator, a dyskryminator stara się nauczyć rozróżniać obrazy. W efekcie końcowym generator jest w stanie generować obrazy bardzo podobne do tych ze zbioru danych.

Więcej informacji na temat działania tych modeli dostępne jest w dokumencie: *Notatka_o_GANach.pdf*

8 Zbiory danych

Aby uzyskać zbiory danych do nauki modeli można wypakować dostępne pliki *.zip*, które zawierają wyselekcjonowane przeze mnie obrazy z wymienionych poniżej zbiorów danych. Alternatywną opcją jest skorzystanie z linków do tych zbiorów znajdujących się w pliku *README.md*. To podejście umożliwi samodzielny wybór zdjęć, które posłużą do trenowania.

Zbiory danych, które zostały wykorzystane do nauki modeli to:

- **Familyguy** (ze zbioru Cartoon Classification)
  Wykorzystany do szkolenia CycleGAN odpowiedzialnego za konwersję stylu na kreskówkę, bądź komiks.
- **VincentVanGogh** (ze zbioru Vincent Van Gogh Art)
  Wykorzystany do szkolenia kolejnego CycleGAN odpowiedzialnego za konwersję stylu na dzieła sztuki Vincenta Van Gogha.
- **natural_images**
  Przykładowe zdjęcia, których styl przekształcany był na jeden z dwóch wyżej wymienionych podczas trenowania.
- **mscoco**
  Zbiór, który dodatkowo rozszerzał zbiór danych z naturalnymi zdjęciami wykorzystywany do nauki modeli.

9 Ograniczenia

Konwersja stylu jest stosunkowo złożonym zadaniem, dlatego czas treningu w notatnikach mocno zależy od mocy obliczeniowej komputera. Z tego powodu zalecane jest, aby wykorzystywać kartę graficzną NVIDIA, która znacząco przyspiesza proces uczenia. Wszystkie notatniki są dostosowane tak, aby korzystać z CUDA, jeżeli jest ono dostępne tzn. zostało zainstalowane wymagane do tego oprogramowanie.

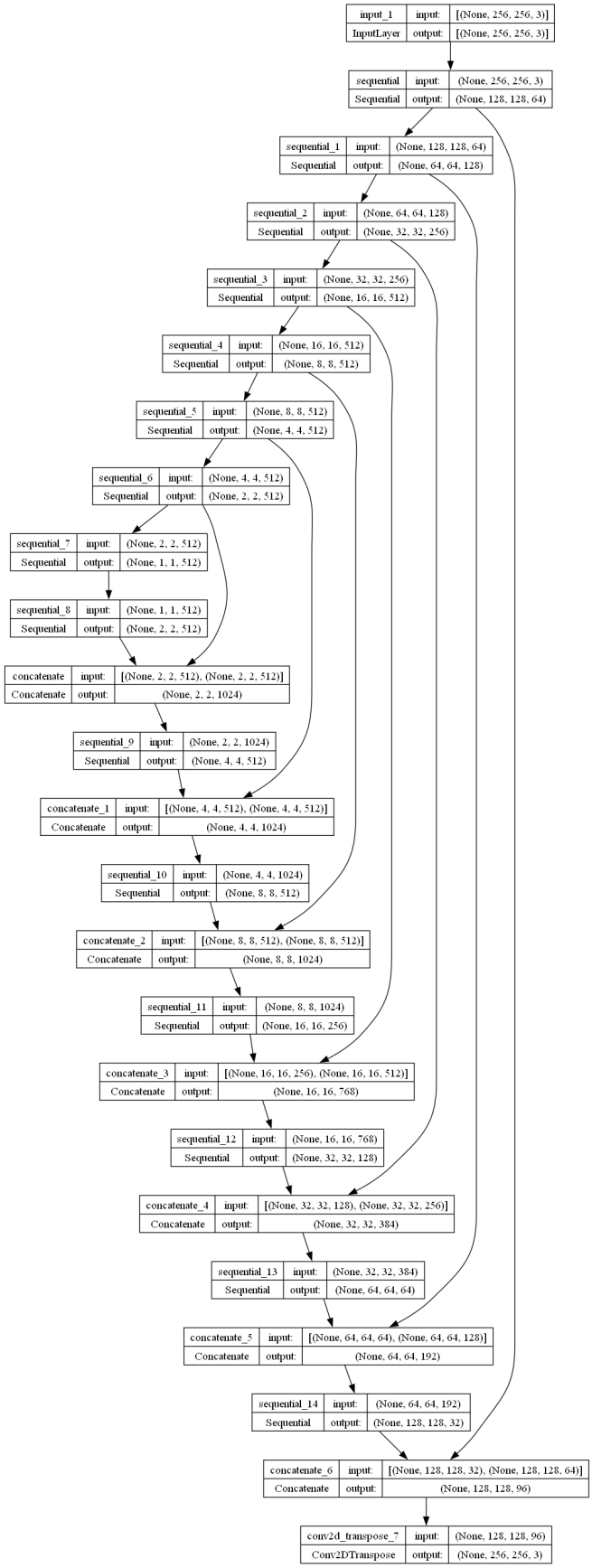
Warto również zaznaczyć, że styl kreskówkowy jest stosunkowo prosty, dlatego model będzie ucinał na szczegółowości zdjęć podczas konwersji. W przypadku zdjęć zbyt skomplikowanych wynik może być niezadowalający.

10 Wizualizacja wykorzystanych modeli

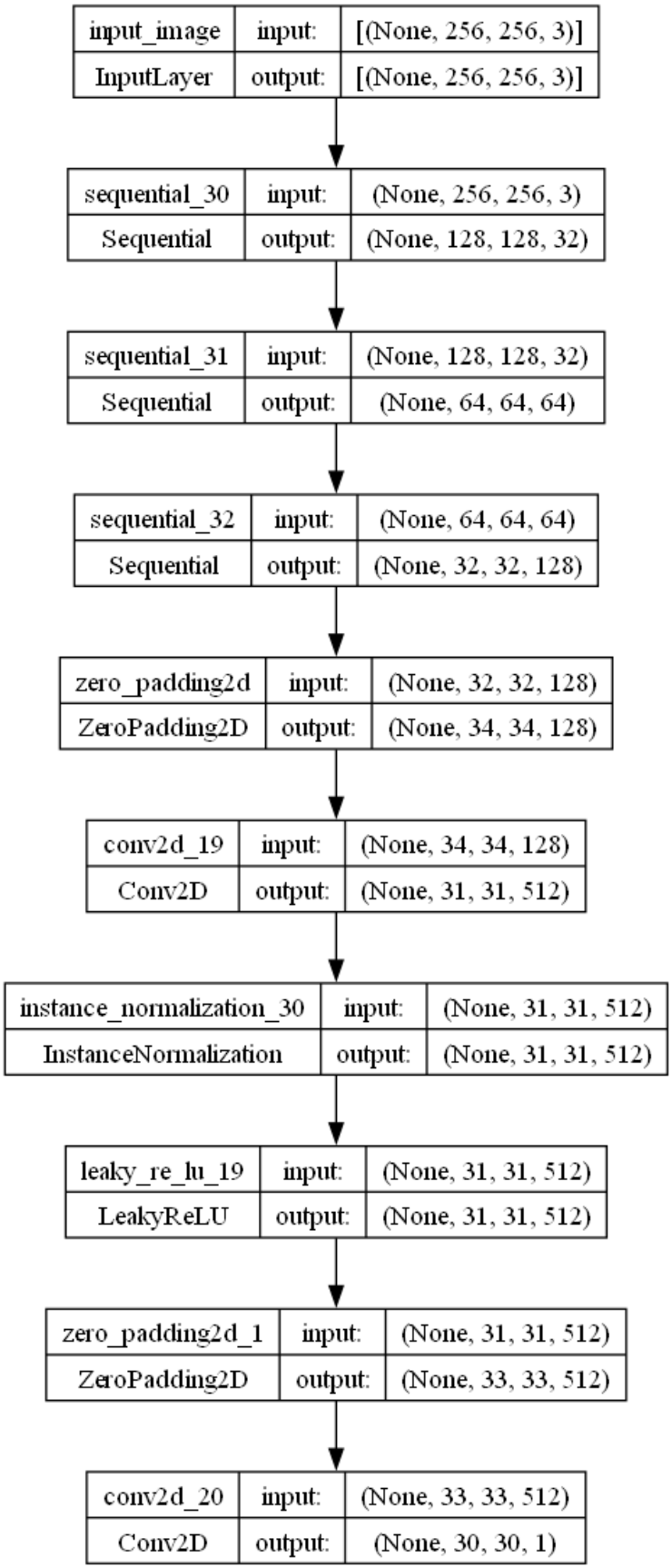
Na następnych stronach widoczne są wizualizacje modeli wykorzystywanych w aplikacji. Wizualizacje te zostały wygenerowane przy użyciu biblioteki *pydot*. Przedstawiają one zarówno generator, jak i dyskryminator dla obu modeli, wraz z poszczególnymi warstwami oraz ich połączeniami. Więcej informacji dostępne jest w dokumentach:

- *Notatka_o_GANach.pdf*
- *Notatka_o_warstwach.pdf*

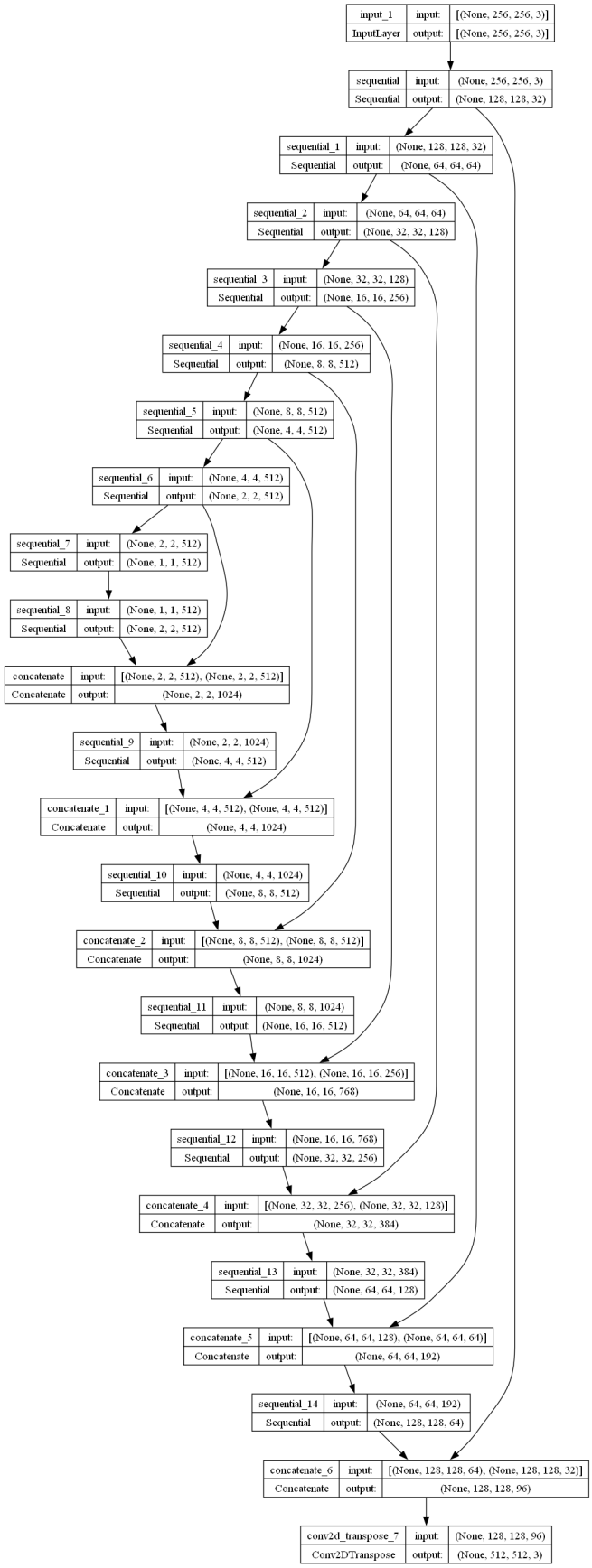
1. CycleGAN - Generator



2. **CycleGAN - Dyskryminator**



3. SRGAN - Generator



4. **SRGAN - Dyskryminator *(PatchGAN)***