Project report and specification

Ganics

Adrian Chmiel

June 9, 2024

Link to the repository: *https://github.com/chmieladr/Ganics*

1 Project's goal

The aim of the project is to utilize GAN networks for generating images with translated style. The program offers two possible style conversion modes:

1. **Vincent Van Gogh Art** - images will resemble the artworks of Vincent Van Gogh
2. **Cartoon** - images will resemble pictures from cartoons (or comics)

After the style is transformed, the image is upscaled to a resolution of *512x512*. The final image is saved in the *.png* format.

2 Technologies used

The program was written in **Python 3.10**. It uses the following libraries:

- **matplotlib** - for creating plots and visualizing data
- **numpy** - for matrix operations
- **opencv-python** - for image processing
- **pandas** - for data analysis
- **pillow** - for image processing
- **pydot** - for visualizing model structure
- **tensorflow** - for creating and training GAN models
- **tkinter** - for creating a simple graphical interface
- **tqdm** - for displaying progress in model training

I have used Python programming language because it is widely used in the field of machine learning and artificial intelligence. It offers a wide range of libraries and tools that facilitate the development of such projects. Out of many machine learning libraries, I have chosen **Tensorflow** because it is one of the most popular libraries for creating neural networks and I have also used it in the past for other projects.

For this project I have decided to use **Python 3.10** version mostly because it is the last version supported by *Tensorflow 2.10*. Newer versions of this library don't natively support GPU acceleration on **Windows** operating system, which is crucial for the learning process as it heavily reduces the time needed for training the network. While I could have restricted the project to *Linux* systems, I wanted to make it more accessible.

Apart from that I have also utilized **Jupyter Notebooks** for training the models. They're main advantage is the ability to run code in small chunks, which makes it easier to find the issues and understand the code. They are available in the *notebooks* directory.

3 Launching the program

In order to run the application directly from the **source code**, it is recommended to create a new virtual environment and install the required libraries/packages in it. Assuming that we are using *conda*, we can do this with the following commands:

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

The installation of Jupyter can be skipped if we do not plan to use notebooks.

For NVIDIA graphics card owners, it is also worth running the following command, which will enable the use of CUDA (provided that you have other essential software installed):

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

The program can be launched by calling the ***main.py*** file:

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

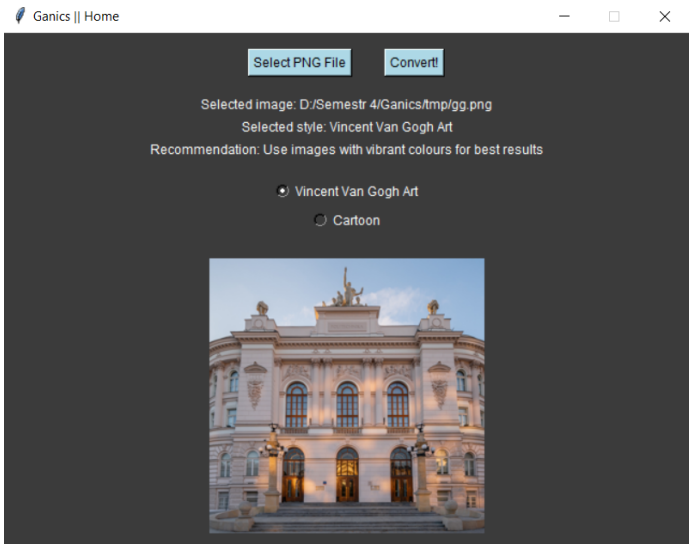
Program has been tested on both *Windows* and *Linux* operating systems.

4 Graphical interface

The graphical interface of the program was created using the *tkinter* library. It allows:

- Selection of the input file
- Selection of the conversion style
- Converting and saving the image
- Preview of the image before and after conversion

Sample screenshot of the user interface is visible on the right.



5 Input and output files

1. **Main application**
   The program allows selecting the path to the input file that will be converted. It is worth noting that the input file must be an image in the *.jpg*, *.jpeg*, or *.png* format.

   The output file, regardless of the initial format, is always an image in the *.png* format with a resolution of *512x512*.

2. **Notebooks**
   When it comes to notebooks, we provide entire sets of images as input (more information below), which will be used to train the model. As output, we receive visualizations illustrating how the model performs after training and an exported model weights file, which is later used as a component of the main application.

6 How the application works

The application's operation concept is visible below:

1. File selection and conversion style
2. Image style translation using **CycleGAN**
   The resulting image has a resolution of 256x256.
3. Image upscaling using **SRGAN** to a resolution of 512x512

4. Saving the image in the *.png* format

Important fact is that the application **will not work** if it does not find the appropriate model weights, which the user will be informed about by an error. In such a case, it is necessary to either run the notebooks to train the model, or download the ready-made files available on *Google Drive* under the link in the *how_to_get_models.md* file.

7 Theory

The program is based on two GAN network models:

- **CycleGAN**
- **SRGAN** (with *PatchGAN* discriminator)

GAN *(Generative Adversarial Network)* is a type of artificial neural network that consists of two models: **generator** and **discriminator**. The generator's task is to generate new images that are similar to those from the dataset, while the discriminator's task is to distinguish whether the image comes from the generator or the dataset. The training process involves a competition between both models, where the generator tries to deceive the discriminator, and the discriminator tries to learn to distinguish images. As a result, the generator is able to generate images very similar to those from the dataset.

8 Datasets

To obtain datasets for training models, you can extract the available *.zip* files, which contain images selected by me from the datasets listed below. An alternative option is to use the links to these datasets available in the *README.md* file. This approach will allow you to independently choose images that will be used for training.

The datasets used for training models are:

- **Familyguy** (from the Cartoon Classification dataset)
  Used to train CycleGAN responsible for converting the style to a cartoon (or comic).
- **VincentVanGogh** (from the Vincent Van Gogh Art dataset)
  Used to train another CycleGAN responsible for converting the style to Vincent Van Gogh's artworks.
- **natural_images**
  Sample images, the style of which was transformed into one of the two mentioned above during training.
- **mscoco**
  A dataset that additionally expanded the set of data with natural images used to train the models.

9 Limitations

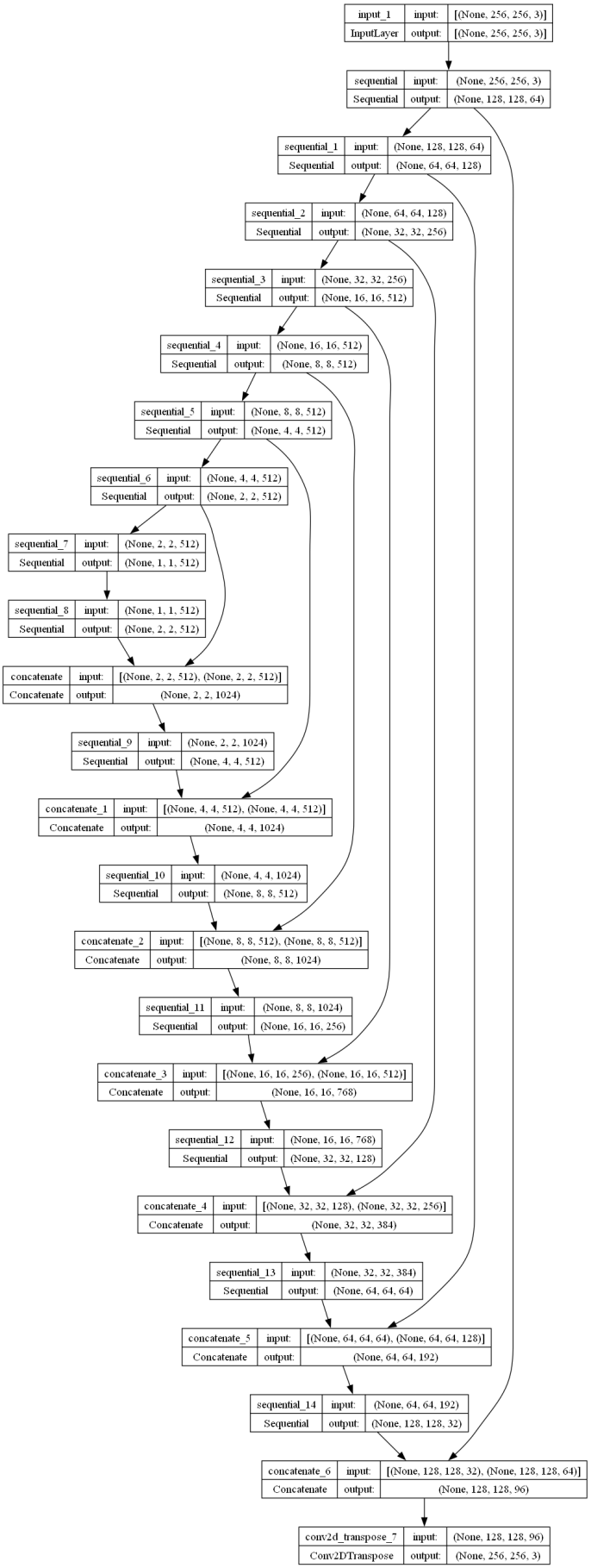
Style conversion is a relatively complex task, so the training time in notebooks strongly depends on the computational power of the computer. For this reason, it is recommended to use an NVIDIA graphics card, which significantly speeds up the learning process. All notebooks are adapted to use CUDA if available, that is the required software has been installed.

It is also worth noting that the cartoon style is relatively simple, so the model will cut off the details of the images during conversion. In the case of too complex images, the result may be unsatisfactory.

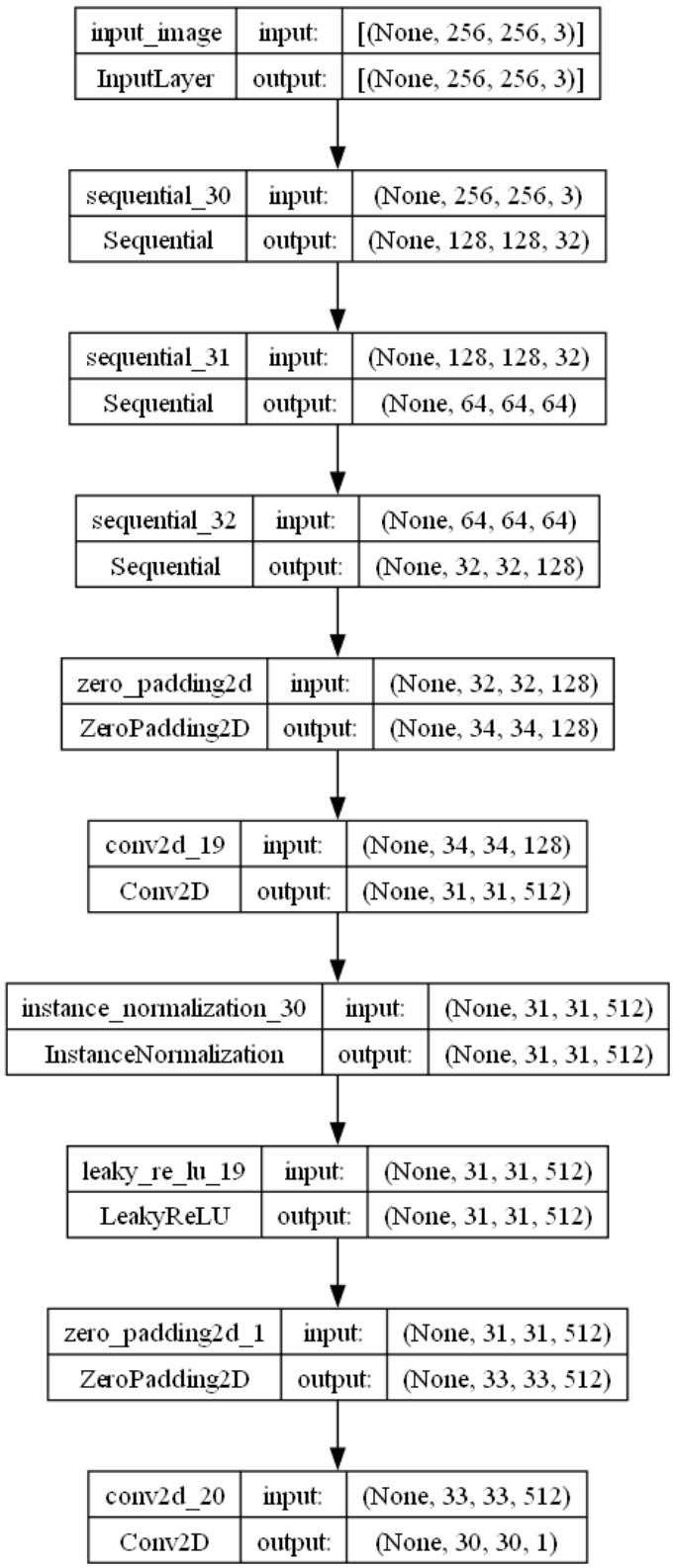
10 Visualization of featured models

On the following pages, visualizations of the models used in the application are visible. These visualizations were generated using the *pydot* library. They show both the generator and the discriminator for both models, along with individual layers and their connections. More information is available in the *Notes.pdf* document.

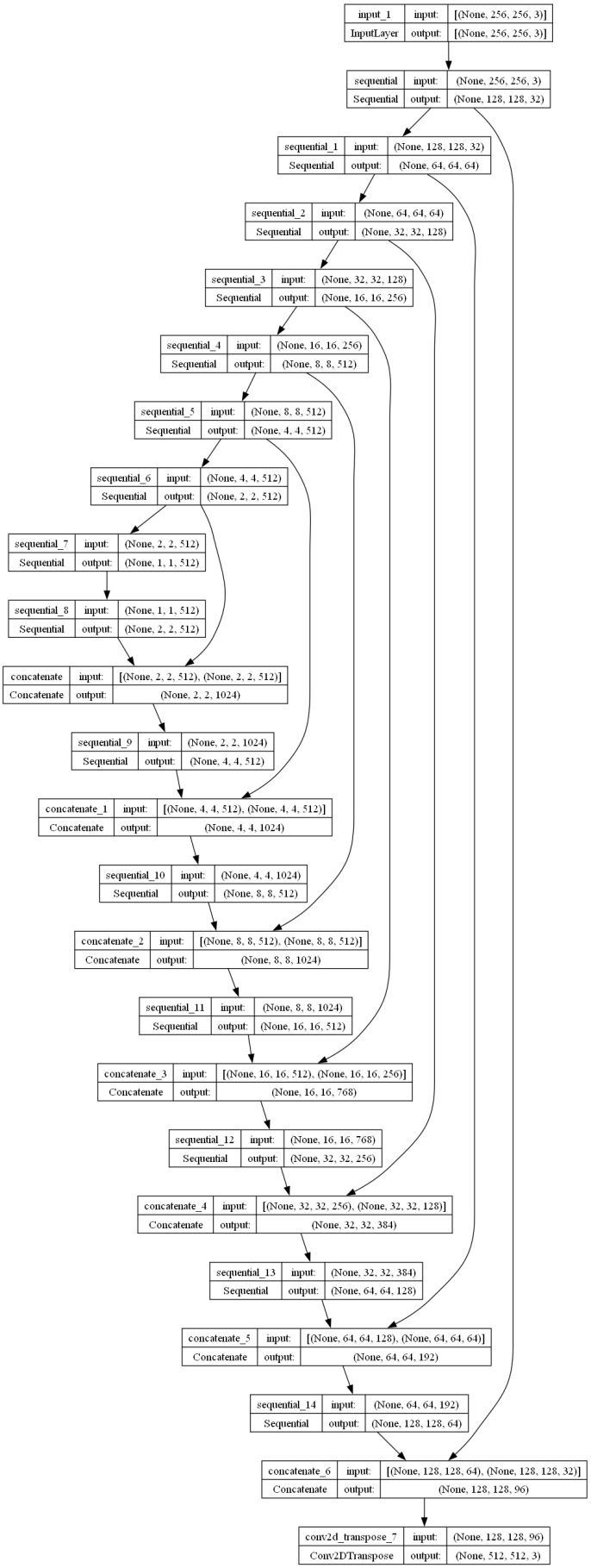
10.1 CycleGAN - Generator



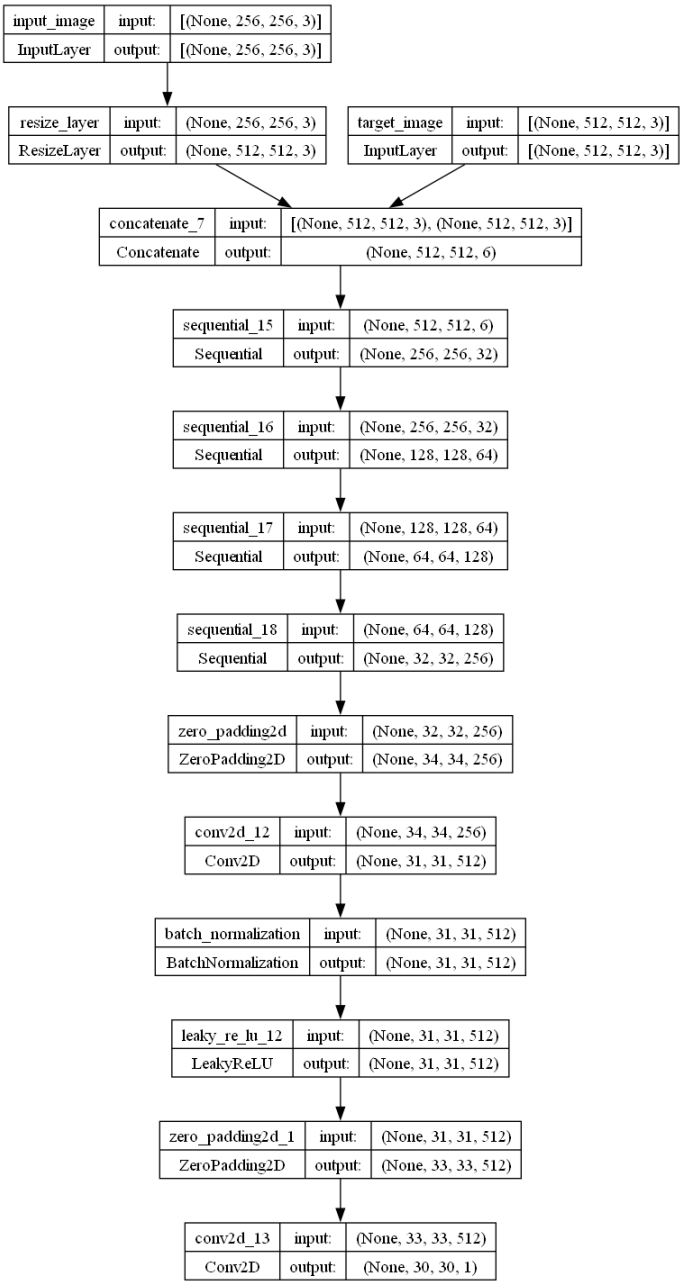
10.2 CycleGAN - Discriminator



10.3 SRGAN - Generator



10.4 SRGAN - Discriminator *(PatchGAN)*

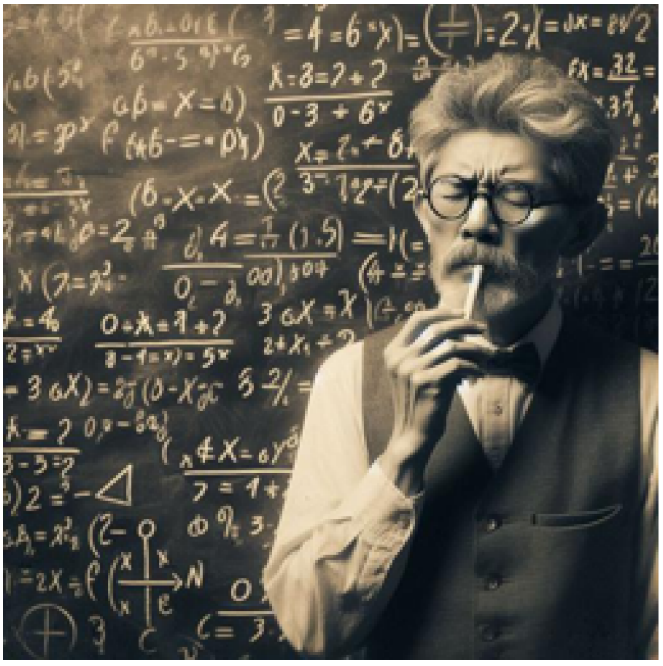


11 Example results

Below you can see example images exported from the application. The images on the left are the input images, and the images on the right are the results of the conversion.

11.1 Vincent Van Gogh Art

11.2 Cartoon



12 Conclusions

- The program allows for the conversion of images to two different styles: Vincent Van Gogh Art and Cartoon.
- It two models: CycleGAN for style conversion and SRGAN for image upscaling.
- The program has been provided with graphical interface that allows for easy selection of the input file and conversion style.
- The application requires the availability of model weights, which can be obtained by running the notebooks or downloading them from the provided link.
- The program works on both Windows and Linux operating systems.
- Proper choice of epochs amount is crucial for the quality of the final image. It is essential to look at the results and monitor the loss during training to avoid overfitting.