Student id:

318213162

Student full name:

Yehonatan Ravoach

Describe your project in words:

I used the open source cycleGAN to implement face swap between my face to Ryan Reynhold and vise versa.

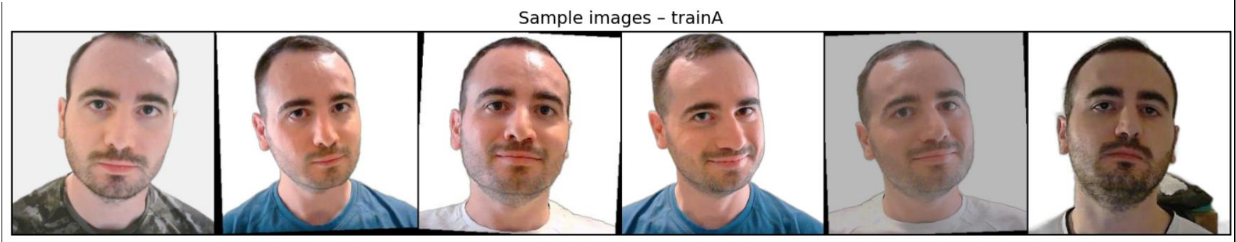
Method used to achieve project goals and why did you choose this method:

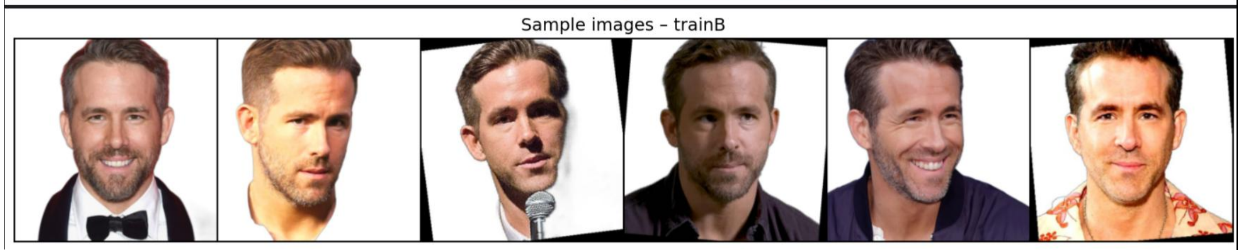
- Brief CycleGAN architecture (2 Generators, 2 Discriminators, key losses).
- Works with unpaired data.
- Chosen because it is lighter and faster than diffusion models yet effective for style/identity transfer.

Describe your dataset (size, resolution, show 10 screenshots of train and 10 of test images):

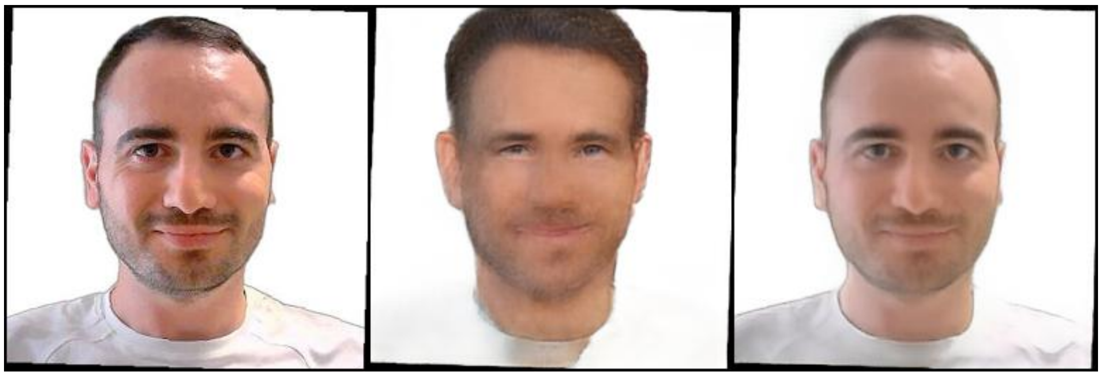
TrainA = 3000 images (1000 original live collected, 2 augmentations per image), Portrait, background cleaned
TrainB = 3000 images (1000 original manually collected from the Internet, 2 augmentations per image), Portrait, fine side looking, background cleaned.

When loading the data Loader, the images were resized to **256x256** to meet the open-source requirements and for reasonable training time (I considered using 512x512 but convinced it wouldn't change the results).

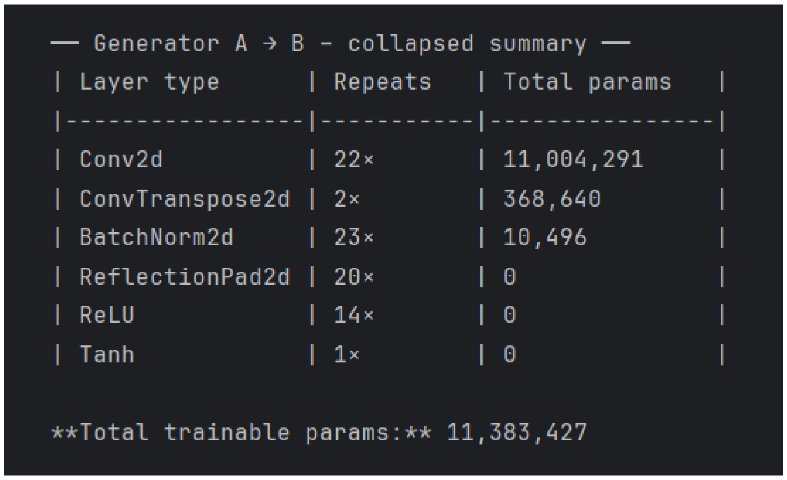




Screenshot depicting one input and its output of your project:

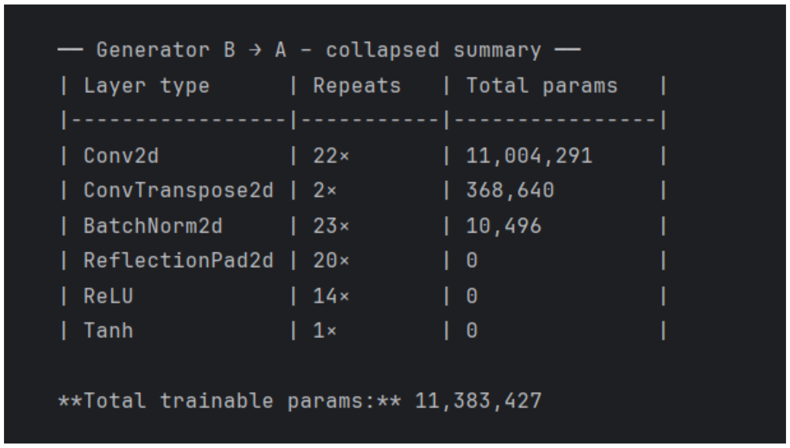


Screenshots of the summary of all DNNs involved:



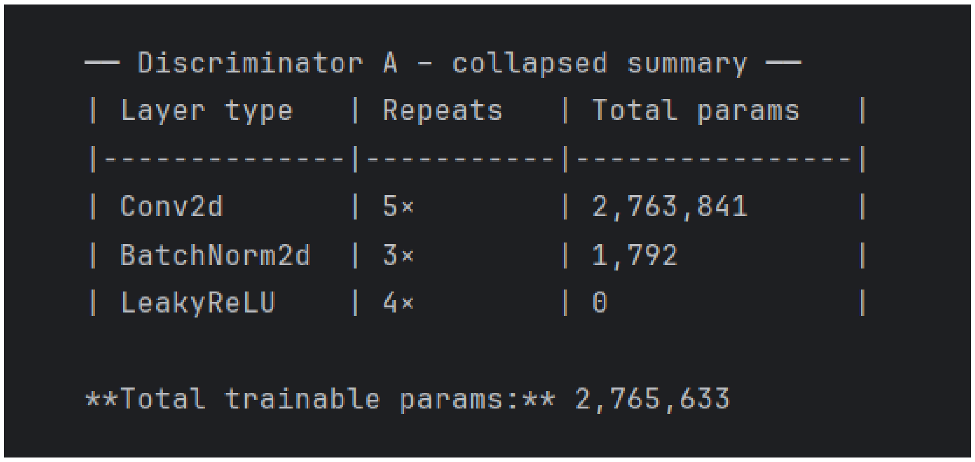
```
── Generator A → B – collapsed summary ──
| Layer type       | Repeats   | Total params   |
|------------------|-----------|----------------|
| Conv2d           | 22×       | 11,004,291     |
| ConvTranspose2d  | 2×        | 368,640        |
| BatchNorm2d      | 23×       | 10,496         |
| ReflectionPad2d  | 20×       | 0              |
| ReLU             | 14×       | 0              |
| Tanh             | 1×        | 0              |

**Total trainable params:** 11,383,427
```



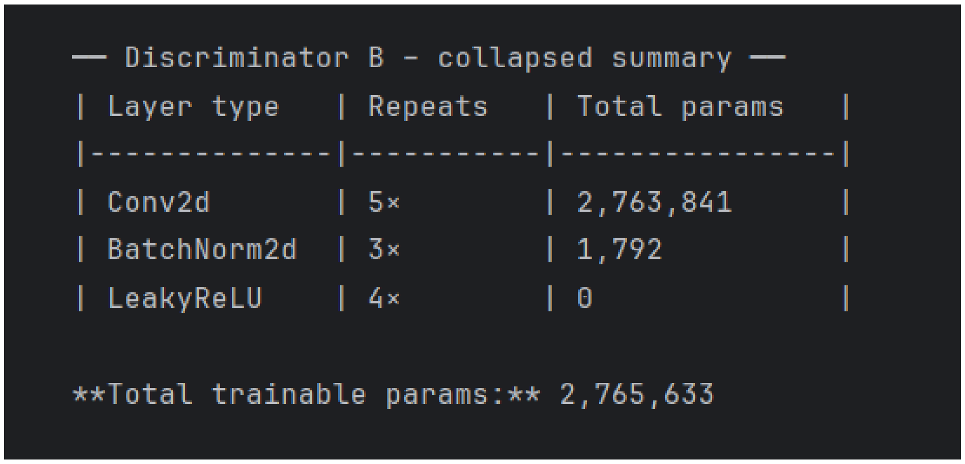
```
── Generator B → A – collapsed summary ──
| Layer type       | Repeats   | Total params   |
|------------------|-----------|----------------|
| Conv2d           | 22×       | 11,004,291     |
| ConvTranspose2d  | 2×        | 368,640        |
| BatchNorm2d      | 23×       | 10,496         |
| ReflectionPad2d  | 20×       | 0              |
| ReLU             | 14×       | 0              |
| Tanh             | 1×        | 0              |

**Total trainable params:** 11,383,427
```



```
── Discriminator A - collapsed summary ──
| Layer type   | Repeats    | Total params    |
|--------------|-----------|----------------|
| Conv2d       | 5×         | 2,763,841       |
| BatchNorm2d  | 3×         | 1,792           |
| LeakyReLU    | 4×         | 0               |

**Total trainable params:** 2,765,633
```



```
── Discriminator B - collapsed summary ──
| Layer type   | Repeats    | Total params    |
|--------------|-----------|----------------|
| Conv2d       | 5×         | 2,763,841       |
| BatchNorm2d  | 3×         | 1,792           |
| LeakyReLU    | 4×         | 0               |

**Total trainable params:** 2,765,633
```

Screenshots of the training process per epoch (all loss values):

During training I unfortunately forgot to save the full loss log or plot.
The values were monitored on-screen and followed a stable pattern:

- **Generator loss (G)** – started at ≈ 5.0 on the first epoch and quickly stabilised in the *1.5 – 2.5* range for the remainder of training.
- **Discriminator losses (DA and DB)** – consistently oscillated between *0.05 – 0.20*, indicating that both discriminators maintained good balance against their respective generators.

Describe how many epochs did you train, and how and why did you decide to stop:

I originally planned to run 200 epochs, but I stopped training at epoch 175 when the discriminator losses began collapsing toward zero—an early warning sign of mode collapse.

For project defense, I will represent the checkpoint number 175.

Screenshots of how training progress on a specific image (10 images overall):