Deep Learning- Final Course Project

Shaked Cohen (ID. 207881962), Ori Cohen (ID. 207375783)

Submitted as final project report for the DL course, BIU, 2021

1. Abstract

GAN is considered extremely data-hungry, with a good reason. Training a GAN with limited data often results in overfitting, mode collapse and will probably have trouble converging.
This difficulty only gets highlighted in more complex GAN-based methods, like CycleGAN.
In this paper, we will explain how we managed to train a CycleGAN for (photo to monet) style-transfer purpose, given a very limited data - 30 Monet paintings, achieving fairly good results - Kaggle score 63.74737 MiFID score.
Our methods include transfer learning, GAN-tailored data augmentation(DiffAugment, DAG, probability based transformation), data scraping for related style paintings, synthetic data creation using state-of-the-art GANs like StyleGAN2-ADA, lightweight GAN, both using transfer learning, as well as many other methods we will cover.

2. Related Work

CycleGAN was first discussed in https://arxiv.org/abs/1703.10593.
However, in order to achieve better results for cases when we have a small dataset, require high resolution output, with limited GPU resources, the key is to improve CycleGAN stepping stones - GAN itself. Some attempts were recently made in that field. https://arxiv.org/pdf/2006.05338.pdf introduces DiffAugment - data augmuntations aiming to improve training without defecting the generator learned distribution, using inversive augmentation on the discriminator. Although they prove that theoretically only inversive augmentation will not defect the generator or his convergence, they show that practically some other transformations might benefit the learning.
Another approach, shows that as long as the augmentations happen with probability p on each train step, the generator will eventually learn the right distribution, and the augmunation will not lead the generator to lose touch with the designated distribution. These methods were introduced and used in
https://arxiv.org/pdf/2006.06676v2.pdf
https://arxiv.org/abs/2103.00397
and integrated in the state-of-the-art Nvidia's StyleGAN2-ADA https://arxiv.org/abs/2006.06676
https://github.com/NVlabs/stylegan2-ada
Having the previous p probability automatically decay and rise during training, often matching StyleGAN2 results with an order of magnitude fewer images.

As our guideline for few-data image synthesis we used https://arxiv.org/abs/2101.04775 , tweaking with the dataset, data preprocessing and transfer learning.

3. CycleGAN

3.1. Baseline CycleGAN

As a baseline, we implemented the CycleGAN exactly as introduced in the main paper. We then tweaked with the Generator and Discriminator a bit, and evaluated the results after a few epoches (at this point we evaluated with what our eyes can see), along with some tips and implementations we found online.
We used *MSE* for Generator & Discriminator loss, and generators loss adversarial + cycle + identity.
The baseline achieves very impressive results for 300 Monet paintings. After only 70 epoches (5 hours of training on Colab Pro).

However, results were pretty poor on 30 Monet photos.

3.2. Data Augmentation

At this point, we realized data augmentation is essential in order to improve results. However, it's unclear how the transformations on the dataset would not lead the generator to learn transformed distribution as if it were the objective distribution, e.g. generate rotated images.

Deep dive into the literature in that area, led us to come across papers dealing with that issue. Two main solutions were introduced:

1. Transformations will be only introduced to the discriminator, s.t. the generator is fed with the original image x, but the discriminator receives T(x) when T is the transformer. Another approach creates multiple linear heads for the discriminator, each for every transformation. Third one, creates k entirely different discriminators for k different transformations.
2. Every transformation will happen with probability p, when p can be either constant or dynamically changed at each epoch, like in StyleGAN2-ADA.

We used method 1, utilizing DiffAugment, “Substituting every D(x) with D(T(x)), where x can be real images or fake images, D is the discriminator, and T is the DiffAugment operation” - taken from their documentation. Immediate improvement was not observed, however when applying DiffAugment, learning seems to continue for longer periods without having the score detoriert. Our conclusion is that it helped to avoid overfitting and possibly failure to converge.

Method 2 was heavily used during the ***Synthetic Image Generation*** phase.

4. Synthetic Image Generation - Data Amplification

If you can’t beat them, join them.

Tinkering with GANs brought us the idea that if data is missing - why not create him?
Our goal was to generate synthetic images that are “as close to Monet painting as possible”. This is a rathar pretentious goal. For starters, it is not a well defined goal. One possibility is to use a FID-alike metric to calculate similarity, however there is no guarantee(that we know of) that a GAN training on data with low FID score to a target domain distribution, would generate images with a lesser FID score than its training data. Logic says that it won’t.

Training on 30 Monet paintings to produce more samples is also not an option - 30 samples alone is not enough, even for the state-of-the-art GAN. Above that, this data is well used and augmented in our CycleGAN already.

Transfer Learning to the rescue.
We hypothesized that applying a pre-trained model StyleGAN2-ADA/lightweight-GAN, and fine tuning with our 30 Monet painting, would produce good results. With the images we output from this GAN would serve our CycleGAN as if it was a genuine Monet painting.

Synthetic Image Generation Experiments

First StyleGAN2-ADA was experimented. Implementation was taken from the paper’s official pytorch GitHub repository. You can find our usage here(note that this is just to get an idea of the experiment, parameters were tweaked):
https://colab.research.google.com/drive/1I0ubGNcRLOjRifBYr2hrNid-9TtfulY0?usp=sharing

As for pre-trained, we tried both ffhq-256-noaug and celebahq-res256, Nvidia’s official pre-trained models. Both well trained on face generation, 256-256 resolution. ffhq is trained without ADA-augmuntation, celebahq is trained with ADA having 0.5 p target. For any further information it is recommended to read the paper or access their GitHub repository.

Each experiment ran for a few thousand first steps (--kimg), and results were inspected. (results were similar for both pre-trained, ffhq is shown arbitrarily)

Truth be told, at first the outcome was horrifying. These are the most children-friendly images we found. However, giving the model some time to tune, FID-50k score deteriorated significantly from 345 to 210, with some better looking results:

(1000 kimg 345 fid50k, 3000 kimg 210 fid50k, left to right)

Unfortunately, fid50k started to rise up eventually, leading to producing unpleasant results. We believe the reason is that the pre-trained faces network had no landscape, horizon, or basically any abstract objects embedded in it. Faces were just too different from Monet painting.
Finding a pre-trained alternative in a 256x256 resolution and same model architecture turned out to be not as simple as you would imagine. So we did it ourselves.

However, this time we used lightweight-GAN as our out-of-the-box model. The reason is that StyleGAN2-ADA requires orders of magnitude more computation resources and time to train, and possibly more train samples.

4.2. Training Data - Scraping, Preprocessing, CLIP Filtering

Now this might be the most interesting part of our research.
We encountered an idea brought by Robert A. Gonsalves in his Medium post about GANScapes https://robgon.medium.com/, where he used scraping in order to collect training data, and filtered it using OpenAI's CLIP text-image NN. https://arxiv.org/abs/2103.00020

Applying his idea in our domain, with small revisions to his code and adaptation to our needs, Monet-like impressionists-style artists' paintings, living around Monet's time, were scraped from WikiArt. Specifically, landscapes, as we noticed that our CycleGAN performs best on landscapes. 2080 non-Monet paintings were scrapped.

Then, these scraped images were preprocces to match 256x256 resolution, using hybrid aspect ratio averaging technique, with this code https://colab.research.google.com/drive/1cXBPjP1Zw2GHoPlosBLGOW2Jl_ozBImg?usp=sharing, also heavily inspired by Robert's work.

At last, all images were fed to a pre-trained CLIP model, ranking the image fitness score to the description: "monet painting". The sorted output was amazing: The first approximately 200 were looking as if they were painted by monet, extreme resemblance. 200-400 has strong similarity to Monet's style. An untrained eye (like ours) could see that the rest were not Monet's.

Could you tell the difference?

Monet’s

images ranked 0-3

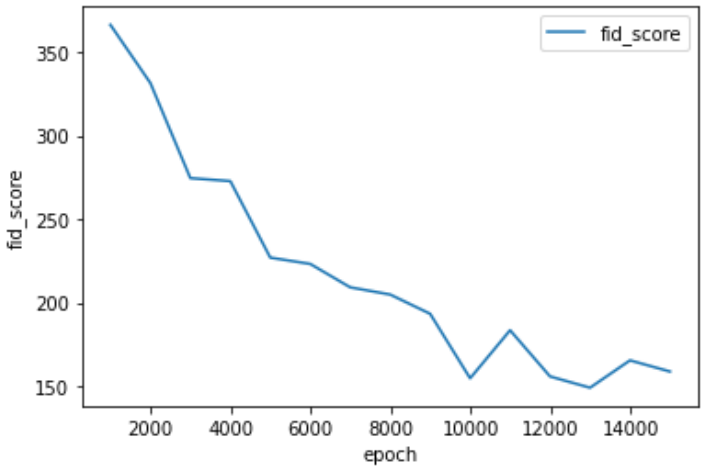
images ranked 380-382

images ranked 2077-2079

The first 400 were chosen to be the input for the lightweight-GAN, denoted from now on as monetlike-GAN.

After only 14 epoches (a few hours on Colab), the FID score the GAN achieved on his dataset was around 150 - not bad, but not the best. For comparison, when I calculated the FID for 30 Monet's on 30 different Mont's, the score was around 120.

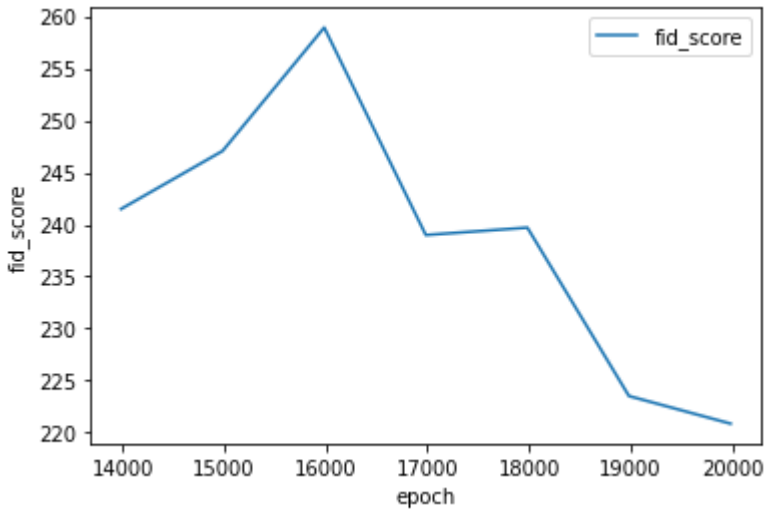
Unfortunately, we couldn't afford the training time and computational cost the rest of the training would have taken, however it was shown that lightweight-GAN can achieve a very small FID score. If applied in our experiment, it would have probably improved our final results significantly.



images and interpolation generated by monet-likeGAN (interpolation will not work on PDF)

Transfer Learning was utilized at this point, building on the monetlikeGAN as a base model, and starting to train on our 30 Monet paintings. No layers were frozen at this experiment, although it should be a good idea for any future work, since the base model dataset(monetlike) is similar to objective(monet) dataset. The outcome model is denoted from now on as monetGAN.

8



Looks like our monet-GAN is on the right track!
Sadly, we can't afford the time the model takes to learn. It's a real shame - Our own Monet paintings generator could do magic. We'll leave that for future work, and update if we reach any interesting conclusions here: https://colab.research.google.com/drive/1cCykVtIN2fv79bOCNVKLpEKIIaEFVzYW?usp=sharing

As an alternative, an attempt was made to train the CycleGAN directly on the gathered 400 monetlike + 30 monet images. This attempt has a very strong basis: the 400 monet-like images have FID score 89.89 with Monet's 300 images.
The results were so impressive that we felt good about stopping here. See the *Evaluation* section.

5. Evaluation

Inference notebook is provided in the Code section.
During inference, 7000-1000 images are generated from the CycleGAN, and then submitted to the Kaggle competition "I'm Something of a Painter Myself".
Team Name: TNZs
Kaggle public score: 63.74737

6. Discussion and Future Work

30 Monet's paintings were chosen based on our experiment-based assumption that learning landscapes is easier for the generator. The distribution is probably less diversive in similar images, resulting in less struggle for the GAN to generate, and discriminator to discriminate. Future work might include a scientific-proven method to choose the best 30 images, either supporting or contradicting our assumption. There is also the option to brute force all combinations or a subset of them, however the computational cost and time makes this method impractical.

Training a StyleGAN2-ADA instead of the lightweight-GAN should produce superior results given sufficient time to learn, or using a pre-trained network that learns on data more similar to a painting or a landscape.

Transfer learning should be applied in the CycleGAN itself, freezing layers depending on similarity between dataset domains. This might be a sufficient alternative to the entire synthetic data creation.

Filtering images using CLIP can be used in other phases. Generated outputs of the CycleGAN can be filtered by CLIP, leaving only the top-N images most fitting to description of “monet”, “monet painting” etc. this description can be played with as well.
Another method could use CLIP to describe generator images (either generated by monetGAN, CycleGAN or both), filtering any image that the description does not contain “monet” in it.

7. Code

All training and scripts were performed on Google Colab GPU, Free and Pro, using checkpoints.

Training Notebook
https://colab.research.google.com/drive/1V49FU2Pauee9NvjS4qhnNXFw4wUVEti_?usp=sharing

Inference Notebook
https://colab.research.google.com/drive/1H-CETjZltYqxgrzAA6SaUaV5bvKXj907?usp=sharing

---
The following Colab notebooks were forked from Robert A. Gonsalves as described in the paper, applying his idea in our problem domain, with revisions to his code and adaptation to our needs. This code is CC BY-SA licensed.

Colab notebook scraping Impressionist landscapes paintings of artists born between 1800 and 1950.
https://colab.research.google.com/drive/1oDFC4sfSlQfhq-M9-dQNudZe5hKABySn?usp=sharing

Colab notebook for image preprocessing and resizing to 256x256
https://colab.research.google.com/drive/1cXBPjP1Zw2GHoPlosBLGOW2Jl_ozBImg?usp=sharing

Colab notebook for image filtering with CLIP. Notebook was used to find the top 400 monet-like images, among the 2080 collected, matching the semantic search phrase "monet painting".
https://colab.research.google.com/drive/1LDgBT3WJg5sZ9knRTBGMbOVuUjXJ18aL?usp=sharing
---

8. References

CycleGAN

https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

10
https://github.com/keras-team/keras-io/tree/master/examples/generative

Data Efficient GANs

https://github.com/NVlabs/stylegan2-ada
https://arxiv.org/abs/2101.04775
https://arxiv.org/pdf/2006.06676v2.pdf
https://github.com/sutd-visual-computing-group/dag-gans
https://www.researchgate.net/publication/342094517_Towards_Good_Practices_for_Data_Augmentation_in_GAN_Training
https://github.com/odegeasslbc/FastGAN-pytorch
https://github.com/lucidrains/lightweight-gan/blob/main/lightweight_gan

DiffAugment:

https://arxiv.org/pdf/2006.05338.pdf
https://hanlab.mit.edu/projects/data-efficient-gans/
https://arxiv.org/abs/2103.00397
https://github.com/VITA-Group/Ultra-Data-Efficient-GAN-Training
https://arxiv.org/pdf/2006.10738v1.pdf