Deep Learning- Final Course Project

Lior Shimon (ID. 341348498) 18 august 2021

Picture to Monet

1 Introduction

In this project our goal was to add Monet-syle to approximately 7000 images, using Cycle GAN model. Moreover, we were allowed to use 30 painting of Monet only as training dataset.

1.1 Related Works

J. Zhu, T. Park, P. Isola, A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," arXiv:1703.10593 (2017).

This paper introduced the Cycle Gan theory for image to image translation, and compare their result to previous model for the same task such as Bayesian framework, or Neural Style Transfer.

Note: every source code used for implementing our model are referenced at the beginning of each relevant cells in the Training Phase Notebook.

1.2 Framework

using python , we worked on Google Colab Notebook and used their GPU services.
Our data was stored in Google Drive.
We used Pytorch Library for building and training our model.

2 Solution

2.1 General approach

after the 30 Monet Painting selection and augmentation, we built a CycleGan model, using a Unet as generators Model and a CNN as Discriminators.

2.2 30 Paintings Constraints

2.2.1 Painting Selection

Selecting a "good" dataset is a necessary condition for a successful training.
In order to Select 30 paintings over the 300 Monet paintings offered by Kaggle Dataset, we first looked for algorithmic solutions for painting selection.
We tried SimCLR framework for contrastive learning representation, and later we tried to classify painting by their shape using edge detection.
However we didn't reached a significant and balanced representation of the full dataset. So we Decided to Select those 30 paintings manually, first by classifying the type of landscape Monet painted among those 300 Picture.
We decided to classify them into 6 classes: Countryside, cities, night , Nature and River, infrastucture (Bridge, single house, castle,..), and port/boats.
lastly, to select our 30 painting dataset, we selected paintings among those class proportionally to their occurences in the original 300 Monet dataset.
Hopefully, our selection covers the 300 dataset.

2.2.2 Data Augmentation

in order to reach back 300 painting from our 30 selected paintings, after normalization, we choosed to operate Horizontal and Vertical Flip , playing with brightness, and color changing (HUE).

2.3 Cycle Gan Design

2.3.1 Inputs

every image input is a 256*256 picture with channel = 3 (RGB)
before feeding the network , we normalized each image with mean 0 and std 0.5 for each channel.

2.3.2 Generators

we used Unet as autoencoder for both picture and Monet Generator.
In Order to improve the performance of our model, we pre processed the Unet before using it as Generator.
Feeding the Unet with 300 random picture from the picture Dataset, we train it to output an image as close as possible as its input.
without this preprocess step, our cyclegan model was outputing random noise.
We wanted the Generators to be ready for style transfer during the training

phase instead of wasting time trying to reconstruct his original input.

2.3.3 Discriminators

we used a 50*50 patch based Convolutional Neural Network for both picture and Monet Discriminator.
Our CNN is composed of 4 block layer; every one of them is composed of a Convolutional layer, Instance Norm, and a LeakyRelu activation Function
as advised by Amyjang at
https://www.kaggle.com/amyjang/monet-cyclegan-tutorial/notebookBuild-the-generator. hyper-parameters are chosen according to the same author as well.

2.3.4 Hyper-Parameters

we initialize the weights of all our model following normal law of mean 0, std 0.02.
we used Mean Squared Error as loss for the discriminator , and L1 loss for Identity loss.
the generators loss were composed of identity loss (lambda = 5), cycle loss (lambda= 0.5), and "standard' loss (tricking the discriminator)
we used the same learning rate of 2e-4 for all our networks.
num epoch: 28

As Optimizer, we used Adam optimizer for every network in our model.
Colab limiting the GPU memory, we used small batches of size 2.

3 Measures

Putting aside loss values, and the average ouptut from Discriminators, We used the Frechet Inception Distance to measure results of our model , since this is the tool used by the Kaggle competetion to give a score on the output file committed by the participants. we sticked to the FID score given by the Kaggle competition, and reach the best FID score of approximately 100 with the hyperparameters describe previously.

4 Discussion

since we had a bad time management, we did not tested enough hyper parameters to see which one fits best our goal. we discovered how Cycle Gan worked from within the code, we discovered that this kind of Deep Learning algorithm need a real strategy from the beginning of the project in order to eventually reach the goal. Since the FID score and our human judgement were not precise

enough, we understood we were lacking of measurement unit to quantify how good our model was to add Monet style to a regular picture, or any Gan based deep learning algorithm. We would have prefer to run the training model more epochs than 28.

5 Code

file containing the 30 chosen Monet painting for training our model:(Drive File)

https://drive.google.com/drive/folders/1V3kKvIkTTfBCt49B5oSI8KC8srDDQkCw?usp=sharing

Training phase Notebook:

https://colab.research.google.com/drive/1tFAbcqbRuPvanEUO8zTv8Hc5zlhFgD4m?usp=sharing

Inference Notebook:

www.google.com https://colab.research.google.com/drive/1C4Ie9Zo9JG0LV4QcR-TFeqdfWqhoPcT?$usp = sharing$

$FakeMonetoutput$

https://drive.google.com/drive/folders/1QohgLb$_o cdRArY7Za-fIRH-tFcFxxLa3?usp = sharing$