Exploring Architectural Styles with Variations of GANs

PREPARED FOR

CS464: Introduction to Machine Learning

Bilkent University

Fall 2019

PREPARED BY

Anar Huseynov

Bahadir Durmaz

Doren Calliku

Ismayil Mammadov

Ufuk Turker

1. Introduction

Problem: Our task is to create architectural pictures which are not skewed, or perceptually troubling, based on previously images. In this project we are experimenting on GAN architectures to create some meaningful results related to architectural styles. Having taken the algorithm from Goodfellow et al.'s paper of 2014, we have gone through different implementations to find which one fits best our purposes.

Method: We did not have to create the algorithm from scratch - we just worked on making the parts work. First we dealt with preprocessing of the data, as our data did not have the parameters the models makes use of. After that we tried to build up the ground truth by creating a Vanilla GAN which makes use of Multilayer perceptron structure. We had some difficulties with it. After a while we passed to CNN GAN. Here we implemented the Generator and Discriminator using Convolutional Neural Network as architectural bricks. After these steps, still the pictures did not look as well as we expected - so we tuned the parameters of the models.

Results: The results have a steady increase of improvement, still we understood that GANs can come with really skewed images and still pass the discriminator. So, out of the pictures created there are a small amount of pictures which are useful, but their resolution is still low.

2. Problem Description

Question we are trying to answer:
Can we use GANs to make useful generated images which can be later used in different applications like in World generation in Game industry and Architectural Restoration?

Use cases:
Game Industry: *Creating a new world:* We can translate the knowledge of the buildings and architecture in a new dataset of generated images where players can feel like they are playing in the Byzantine era.
Architects: *Discriminate:* the project will be able to classify each image into particular architectural styles based on their features. *Combine styles of different Eras:* furthermore it will be able to combine these features through generator. *Renaissance:* it can help in renaissance(returning the old in a new form), or restoration of buildings.

3. Methods

Model

We have been using a GAN Architecture (Generative Adversarial Network) with a

- Generator (deceptive) network that starts with a vector (of random noise)
- Discriminator (detective) network that is trained on specific architectural style buildings (for more check the next section- Data).

Both architectural models were as follows:

Discriminator:

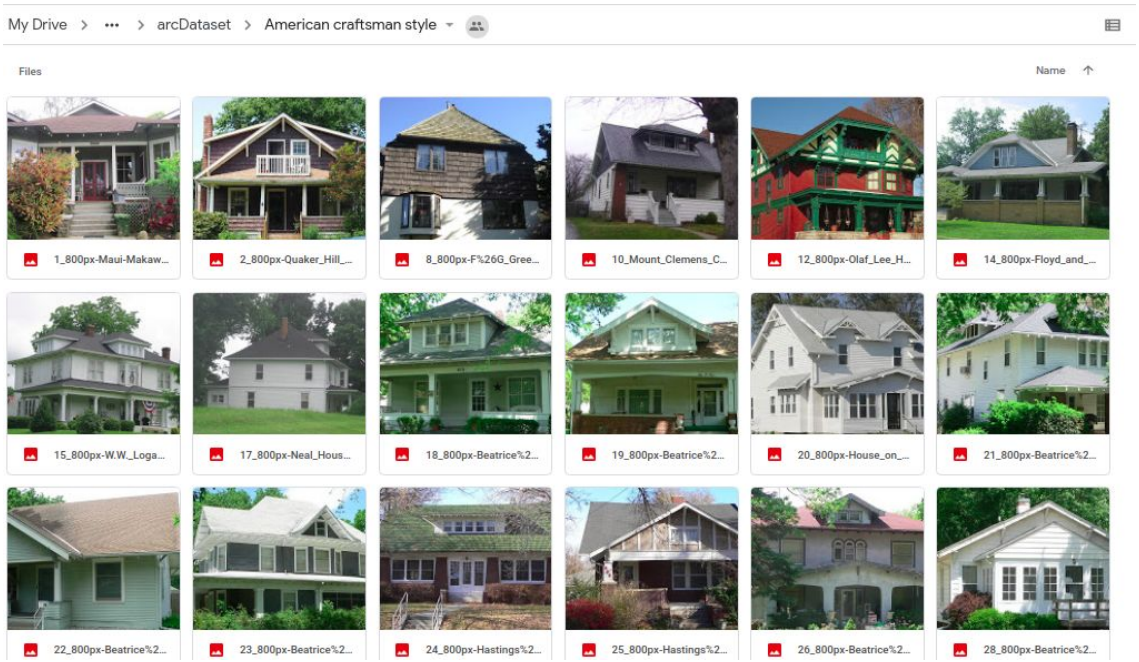
1. Conv2d, LeakyRELU
2. Conv2d, BatchNorm, LeakReLU
3. Conv2d, BatchNorm, LeakyReLU
4. Conv2d, BatchNorm, LeakyReLU
5. Conv2d, Sigmoid()

Generator:

1. ConvTranspose2d, BatchNorm2d, ReLU
2. ConvTranspose2d, BatchNorm2d, ReLU
3. ConvTranspose2d, BatchNorm2d, ReLU
4. ConvTranspose2d, BatchNorm2d, ReLU
5. ConvTranspose2d, Tan()

Data

We worked with a main dataset[1]. This dataset was used for classifying architectural styles in this paper [2]. The dataset consists of around 5000 images of from 25 architectural styles, and the images are taken from Wikimedia. The following is a sample from dataset.



Picture 1. Sample from Dataset

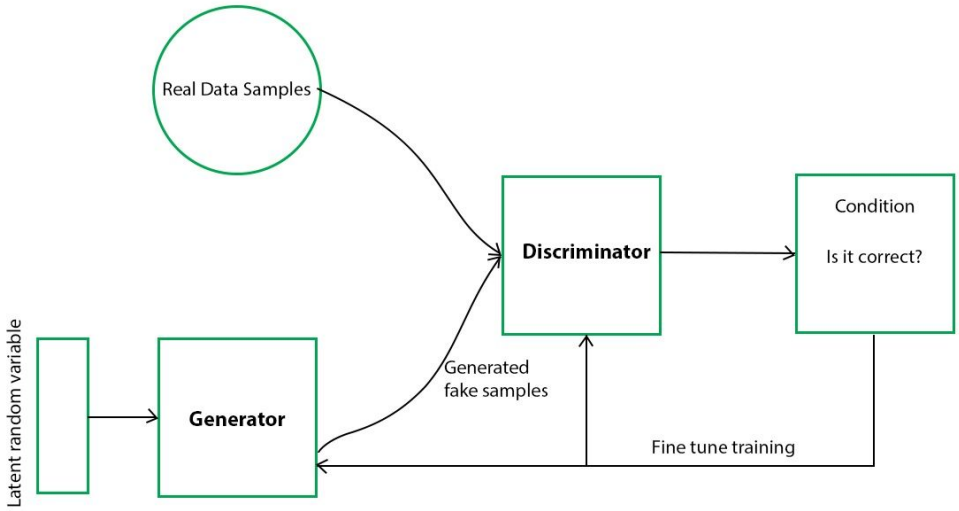
Descriptive Features of the dataset:
Total Number of Pictures: ~5000
Size of Pictures: 100-200 KB

Preprocessing

For the general GAN architecture invariant case, the preprocessing steps we were planning to work on a generalized edge fitting/noise reduction algorithm, still we went for something simpler in Pytorch.

1. Resize Image.
2. Center crop (Our images were not square images from the dataset).
3. Optional: To GrayScale()(made the learning easier, used in the beginning)
4. To Tensor (for usage from the GPU)
5. Normalize (helps reduce skewness and learn better)

Ground Truth



Picture 2. https://www.geeksforgeeks.org/generative-adversarial-network-gan/

As seen from the picture, the algorithm can be seen as a race between the Generator and Discriminator to be better than the other - and from this process across large amounts of iterations both models improve. Pseudo Code for Vanilla GAN:

- Loading the original and the pre-processed data
- Initializing workers, betas, learning rates, batch, image and noise sizes
- Building discriminator (NN.Sequential)

- Building generator network with 3 linearly connected layers as well as ReLUs in between
- Building standard Loss function with mean Binary Cross-Entropy loss to calculate the loss for each matched real and fake (output of generator) photo
- Building an Adam optimizer with pre initialized lr and betas.
- Combining previous components to run the GAN on specific number of epochs and back-propagate in accordance with the losses to optimize weights

Vanilla GAN paired with BCE loss function without any convolutional layers is the bare minimum GAN for our specific problem. Still, this did not work as the multilayer perceptron did not do well with the image data (because of lack of spatial dependency).

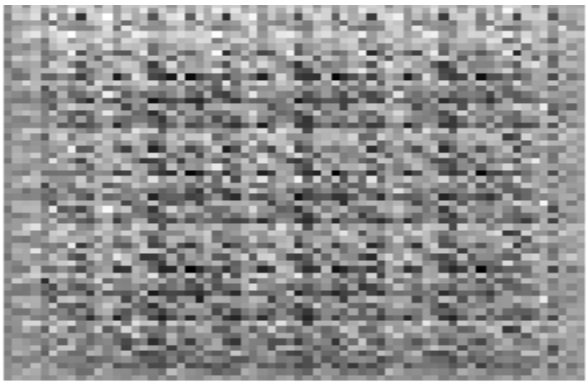
Improvements

Changes to architecture:

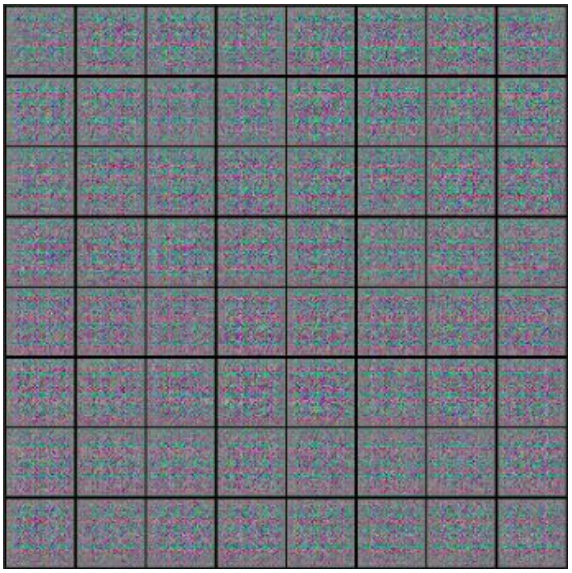
- **CNN usage- DCGAN:** The original Vanilla GAN does not allow real spatial reasoning but DCGAN do since it relies on convolutional neural network [6]. In that case spatial structure of the image like sharp edges will be preserved. In order to create DCGAN, both generator and discriminator were modified in a way that additional convolutional layers were added to both discriminator and generator [6] as seen in the model subsection.

4. Results

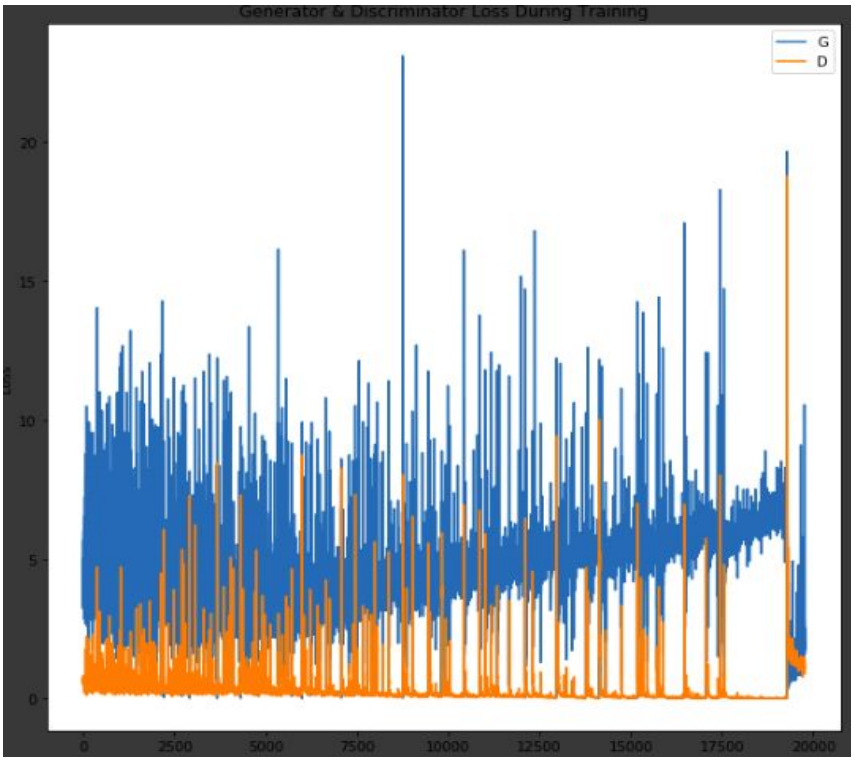
Can we apply the multilayer perceptron? These were the results we got, and we coud not understand why did it happen to be like this.



How do the results evolve with DCGAN? How do the pictures become better even though they started from random noise?



How do the loss functions look like for the Discriminator and Generator?



5. Discussion and Conclusions

How was our experience?

There were many new terms we needed to work on, and some of them we did not understand until the last week of classes. So, our model worked only after checking with sources online, and some of the parameters: like Logits, seem related specifically to these tasks - working with tensors and similar.

What happened to the Vanilla model? Why were the results not representative of what we expected?

Adopting the tuning parameters for VanillaGAN did not work, also we had to interrupt the learning after a point because these models learn very slowly (because of all the layers and updates).

Were you able to build what you set out to do?

To a certain extent. Building DCGAN was a tough task as it needed a lot of time to run. For a whole run we had to wait for at least 5-6 hours, which made the whole process slow and triggering. The results', even though dependent on our model's parameters, are only at a low resolution.

Future directions?

For the models that have been working so far, we have understood that it is not possible for these results to be useful for the aims we wanted. Neural Style Transfer would be a better application for the use cases.

6. Appendix

[1] "Architecture dataset". Online: https://www.kaggle.com/wwymak/architecture-dataset#1036_800px-Cadick_Apartments.jpg. [Accessed: 2.11.2019].
[2] Zhe Xo, Dacheng tao, A.C.Tso "Architectural Style Classification Using Multinomial Latent Logistic Regression". Online: https://pdfs.semanticscholar.org/0ed0/eb02de7579c714236c480f06faf239f3cd95.pdf. [Accessed: 2.11.2019].
[3] "Architecture by style". Online: https://commons.wikimedia.org/wiki/Category:Architecture_by_style. [Accessed: 2.11.2019].
[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza. "Generative Adversarial Nets". Online: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf. [Accessed: 2.11.2019].
[5] X.Mao, Q.Li, H.Xie, R.Y.K.Lao, Z.Wang "Multi-class Generative Adversarial Networks with the L2 Loss Function". Online: https://arxiv.org/pdf/1611.04076v1.pdf. [Accessed: 22.11.2019].

[6] A.Radford, L.Metz, S.Chintala “Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Network”. Online: https://arxiv.org/pdf/1511.06434.pdf. [Accessed: 22.11.2019].
[7] J.Brownlee “A Gentle Introduction to CycleGAN for Image Translation”. Online: https://machinelearningmastery.com/what-is-cyclegan/. [Accessed: 22.11.2019].
[8] L.Gatys, “Image Style Transfer Using Convolutional Neural Networks” Online: openaccess.thecvf.com/content_cvpr_2016/papers/Gatys_Image_Style_Transfer_CVPR_2016_paper.pdf [Accessed: 24.11. 2019]

Division Of Work

PHASE 1:

Report Set-up - DOREN

Preprocessing - UFUK

Setting up ground truth GAN

- Implementing Vanilla GAN - DOREN, BAHADIR
- Report for setting up ground truth - BAHADIR

Exploration for changes

- Existing Improvements’ Range (LSGAN, DCGAN, CycleGAN)- ANAR
- Preprocessing Improvements - UFUK
- Building vs NonBuilding Detection- ISMAYIL
- Neural Style Transfer - DOREN

PHASE 2:

Ground truth GAN

- Implementing Vanilla GAN - BAHADIR, ANAR

Exploration for changes

- DCGAN - ANAR, DOREN, ISMAYIL, BAHADIR