Using GANs to Produce Art in a Particular Style from Semantic Maps

Mathew Tucker

Wichita State University

December 2, 2022

Wichita State University

December 4, 2022

Abstract

We investigate the role of noise in the training process of an image-to-image style transference GAN. We will do this by comparing two models trained on the same training data, a set of semantic maps and their target image; however, in one of the models, we will add noise to the semantic maps prior to the training process. We will then apply our models to a semantic map that was not used in the training process and measure the Euclidean distance, treating the RGB pixel values as spacial coordinates, of the generated image from its target image to compare the accuracy of our two models.

A Soft Introduction into GANs

Imagine the scenario:

You want to train your computer to “generate" images in the style of a particular artist.

How would you go about that?

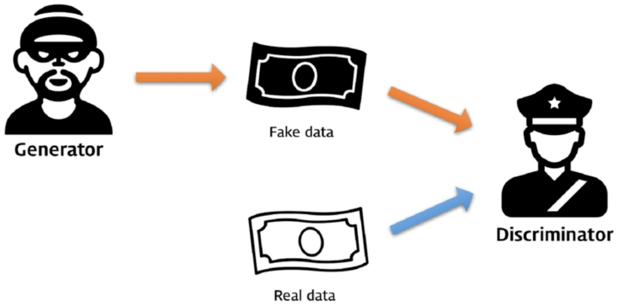
Generator:

“I’ve seen the things, now I’ll TRY to make it based on the distribution I’ve calculated from them."

Why won’t this work?

You only found a distribution for what “could be a thing," now you need to eliminate what “isn’t a thing."

GANs:



Let's call them "D" and "G".

According to Goodfellow et. al., in their landmark paper “Generative Adversarial Nets" in which they establish the design of GANs, *D* and *G* are caught in a *“two-player minimax game" utility functions:*

$$\begin{cases} U_G = \mathbb{E}_{z \sim p_z(z)} \log\left(1 - D[G(z)]\right) \\ U_D = \mathbb{E}_{x \sim p_{data}(x)} \left(\log D(x)\right) + \mathbb{E}_{z \sim p_z(z)} \log\left(1 - D[G(z)]\right) \end{cases}$$

What we'll need to *quickly* "review":

- Game Theory
- Statistics
- Probability Theory

Game Theory

What is a game?

Normal Form Game:

A game in normal form is the ordered triple, $G = (N,\ (S_i)_{i\in N},\ (u_i)_{i\in N})$ where,

- $N = \{1,\ 2,\ 3, \cdots,\ n\}$ is a finite set of players.
- S_i is the set of strategies of player i, for ever player $i \in N$.
- Note: $S = \prod_{i=1}^{n} S_i = S_1 \times S_2 \times \cdots \times S_n$
- $u_i : S \to \mathbb{R}$ is a (utility)function which maps each vector of strategies, $s = (s_i)_{i\in N}$, to their payoffs(utility), $u_i(s)$, for each player $i \in N$.
- *Note: $u(s) := (u_i(s))_{i\in N} \in \mathbb{R}^N$ is the set of all vectors of strategies.*

Expected Payoff/Utility Function:

The j^{th} player's expected payoff for playing a strategy s_j^l is $u_j(s^*) = \sum_{l=1}^{L} p^l u_j^l \left(s_j^l, s_{-j}^{*l}\right)$ where p^l is the probability that s_{-j}^{*l} is played.

Example of Expected Payoff:

$(p_C,\ p_R)$	$S_C(1)$	$S_C(2)$
$S_R(1)$	$(2,\ -2)$	$(0,\ 1)$
$S_R(2)$	$(-4,\ 5)$	$(5,\ 6)$

Assume player R plays strategy 2. If player C plays their strategy 1 $1/3$ of the time, then player R can expect a payoff of:

$$v_r = \frac{1}{3}(5) + \frac{2}{3}(6) = \frac{17}{3}$$

Minimax Game:

This is a bit of a misnomer. A minimax game is a game where one player seeks to minimize their opponents payoff while the opponent seeks to maximize their payoff.

The solution to such a strategy pair is a saddle point in the game.

Examples of games where minimax is commonly implemented:

- Tic-tac-toe
- Backgammon
- Chess

Statistics and Probability Theory

The things we'll need to review here:

- Bayes' Theorem
- Bernoulli probability distribution
- Expected Value
- Loss-Functions
- Cross Entropy

Bayes‘ Theorem:

Bayes’ Theorem follows immediately from the fact that a probability function is a measure, $0 \le p \le 1$, on your population space.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Bernoulli probability distribution:

The Bernoulli distribution is a special case of the Binomial distribution. Recall in the case of the Binomial distribution-

$$P(k;\ n,\ p) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k \in \{0,\ 1,\ 2,\ \cdots,\ n\}$$

The Bernoulli distribution is the case where $n = 1$ so we get,

$$P(k;\ p) = p^k (1-p)^{1-k}, \quad k \in \{0,\ 1\}$$

Expected Value:

For continuous random variable x with a distribution function $\mu(x)$ the expected value of $f(x)$ is:

$$\mathbb{E}\big(f(x)\big) = \int_{\mathbb{R}} f(x)\mu(x)dx.$$

And in the case of discrete data, we have:

$$\mathbb{E}\big(f(Y)\big) = \sum_{\forall y_i \in Y} f(y_i)\mu(y_i)$$

Loss-Functions:

In neural networks, the gradient of a loss-function is used in what's called backpropagation to update the weights between neuron layers.

In machine learning, there several “loss-functions" that are regularly used:

(y_i is the ground truth, and $\hat{y}_i$ is the predicted value)

- Mean Square Error/L^2 Loss-

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

- Mean Absolute Error/L^1 Loss-

$$\frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

- Mean Bias Error-

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)$$

- Categorical Cross Entropy Loss-

$$-\sum_{i=1}^{n} y_i \log[\hat{y}_i]$$

- Binary Cross Entropy Loss-

$$-\frac{1}{n}\sum_{i=1}^{n} \left(y_i \log[\hat{y}_i] + (1-y_i)\log[1-\hat{y}_i]\right)$$

Binary Cross Entropy is the loss-function (originally)used by GANs.

Cross Entropy:

Given two distributions, p and q

$$H(p,q) = H(p) + D_{KL}(p||q) = -\mathbb{E}_{x\sim p}(\log q) = -\sum_{x\in X} p(x)\log[q(x)]$$

Entropy-

$$H(p) = -\mathbb{E}(\log p) = -\sum_{\forall x \in X} p(x) \log \left[p(x) \right]$$

Kullback–Leibler/Relative divergence-

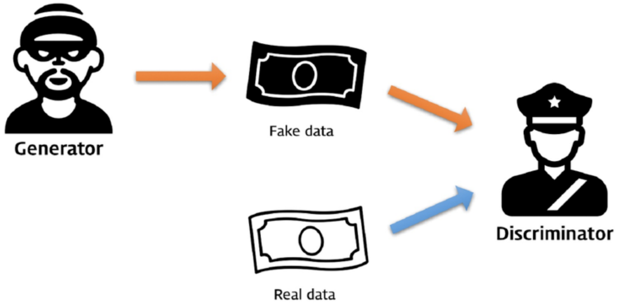
$$D_{KL}(p||q) = \mathbb{E}(\log \frac{p}{q}) = \sum_{\forall x \in X} p(x) \log \left[\frac{p(x)}{q(x)} \right]$$

Suppose x was to be classified as one of two objects then $x \in \{0,\ 1\}$, $p \in \{y,\ 1-y\}$, and $q \in \{\hat{y},\ 1-\hat{y}\}$

$$H(p,q) = -\mathbb{E}_p(\log q) = y\log[\hat{y}] + (1-y)\log[1-\hat{y}]$$

Generative Adversarial Networks

A GAN is the combination of two deep neural networks, a generator and a discriminator.



The point of this architecture is to pit the goal of the two networks against each other in a zero-sum game(payoff$_G$ = −payoff$_D$) who's minimax(strictly speaking, this assumption of zero-sum/minimax is not always true. It is made here for the sake of simplicity) solution can be treated as a loss function and used to improve the training of the two networks via back-propagation.

If $D(x) \in [0,\ 1]$ is the discriminator's confidence that an image is real and $G(z)$ is a generated image then, the payoff's to each plays is:

$$\begin{cases} U_G = \mathbb{E}_{z \sim p_z(z)} \log\left(1 - D[G(z)]\right) \\ U_D = \mathbb{E}_{x \sim p_{data}(x)} \left(\log D(x)\right) + \mathbb{E}_{z \sim p_z(z)} \log\left(1 - D[G(z)]\right) \end{cases}$$

Functionally, what the code does, is it tries minimize the difference between $p_{data}(x)$ and $p_z(z)$ by adjusting $G(z)$ in such a way as to optimize the utility function for both players.

For proofs related to a GAN's ability to converge on to a solution, see:

https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Insofar as images are concerned, some of the elements that a GAN will try to come to understand/recognize the relation between are:

- Shapes
- Colour
- Position
- Size
- Style

Experiment

It should be noted, the code we will be using in this experiment isn’t a GAN as defined by Goodfellow et. al. Instead, we will be using a variation known as pix2pix.

It is our assumption that adding noise to the X variables in our training data we will achieve higher accuracy in the convergence of our model.

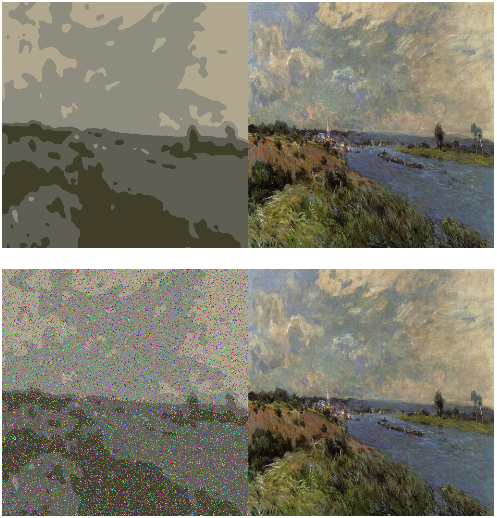
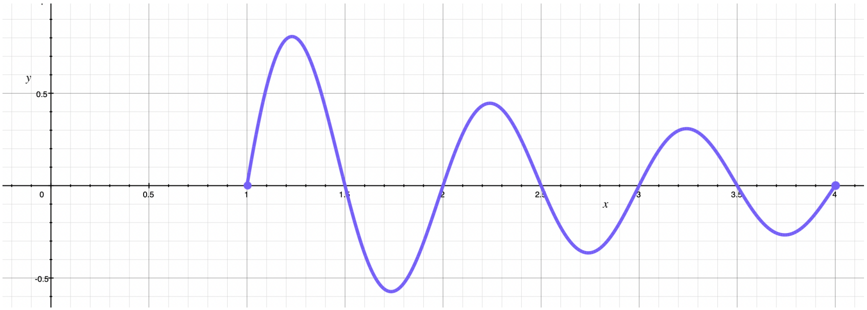


Figure: Two semantic maps, without noise(top) and with noise(bottom), and their target image

Our Reasoning:



Another reason for our assumption is that it will cause the AI to place higher weights on the relative geometries than on the colours.

For example, blue can be from water, sky, a flower, a dress, etc. But, if it stretches the top of the semantic map, we might be able to assume it’s more likely to be a sky or flowers than a dress.

How:

- Train two models
- We add noise to training data
- We apply model to image with known solution
- We treat RGB 3-tuple as Cartesian coordinates and find mean and variance in the deviation of our models' generated images from their target image

Results

At a glance, it appears that replacing $\sim 1/4$ of the pixels of the semantic maps in the training data with noise did not improve training.

Central Tendencies of the RGB values of each Pixel - No Noise

	Target	n.Noise	Targ - n.Noise
Mean	(103.99, 82.54, 69.62)	(103.98, 79.94, 66.42)	(120.98, 111.83, 111.82)
Variance	(2372.71, 1899.24, 2051.54)	(2457.71, 1751.89, 1993.43)	(11903.34, 12446.53, 12797.39)
Mean Norm	151.94	149.48	227.50

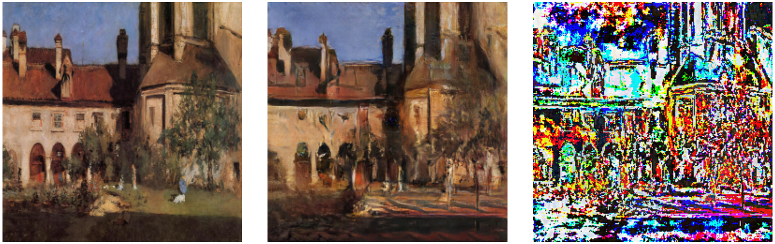


Figure: Target Image, Generated Image with no Noise, The Difference between the two Images

Central Tendencies of the RGB values of each Pixel - With Noise

	Target	w.Noise	Targ - w.Noise
Mean	(103.99, 82.54, 69.62)	(160.38, 129.81, 109.72)	(122.63, 103.69, 95.86)
Variance	(2372.71, 1899.24, 2051.54)	(10128.91, 11518.32, 10715.73)	(4565.15, 4263.17, 5614.78)
Mean Norm	151.94	248.83	201.46

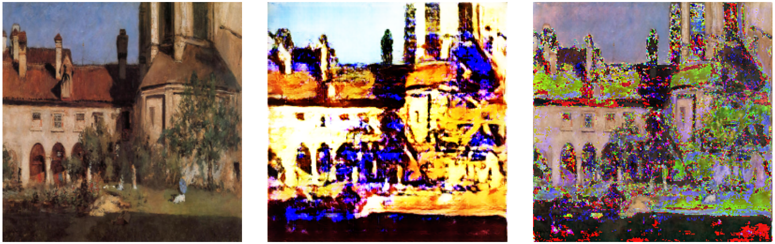


Figure: Target Image, Generated Image with Noise, The Difference between the two Images

Conclusions

Replacing $\sim 1/4$ of the pixels in for all of the semantic maps, in the training data set, with noise negatively impacts the training process.

Possible Changes to the Setup:

- Vary the amount of noise added and the number of semantic maps noise is added to.
- Use a semantic map with noise for validation when training on data with noise.

Possible Changes to the Code:

- Lines 29-35 in config.py and lines 28-29 of dataset.py - changing just one of the images seems ill advised.
- `BATCH_SIZE`, `LEARNING_RATE`, `L1_LAMBDA`, `NUM_EPOCHS` in config.py may need to be tweaked

References I

Foster, D. (2019). Generative deep learning: teaching machines to paint, write, compose, and play. O’reilly.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative Adversarial Nets. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

Hany, J., & Walters, G. (2019). Hands-On Generative Adversarial Networks with PyTorch 1.x. Packt Publishing Ltd.

Isola, P., Zhu, J.-Y., Zhou, T., Efros, A., & Research, B. (2018). Image-to-Image Translation with Conditional Adversarial Networks. https://arxiv.org/pdf/1611.07004.pdf

Kalin, J. (2018). Generative Adversarial Networks Cookbook. Packt Publishing Ltd.

References II

Maheshkar, S. (2021, September 10). What Is Cross Entropy Loss? A Tutorial With Code. W& B. https://wandb.ai/sauravmaheshkar/cross-entropy/reports/What-Is-Cross-Entropy-Loss-A-Tutorial-With-Code--Vml Accessed 28 November 2022

Maschler, M., Solan, E., & Zamir, S. (2013). Game Theory. Cambridge University Press.

Perarnau, G. (2017, March 17). Fantastic GANs and where to find them. guimperarnau.com. https://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them. Accessed 28 November 2022

Robert Duncan Luce, & Raiffa, H. (1989). Games and decisions introduction and critical survey. Dover Publications.

Zhu, J.-Y., Park, T., Isola, P., Efros, A., & Research, B. (2020). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. https://arxiv.org/pdf/1703.10593.pdf

Code

A copy of this presentation and the accompanying paper and code can be found through my personal website:

tuckersideas.net

Thank You