Degenerative Adversarial Networks

Raphael Gontijo Lopes* **& Diptodip Deb** †
Georgia Tech, Atlanta, GA
{raphaelgontijolopes, diptodipdeb}@gatech.edu

Abstract

In recent years, Deep Learning researchers have collectively achieved a pace of useful information extraction that is dangerously close to outstripping the second law of thermodynamics. To solve this problem, we propose a new framework for estimating degenerative models via an adversarial process, in which we simultaneously train two models: a degenerative network D that destroys the data distribution and a discriminative model D that estimates the probability that a sample came from true noise rather than D. The training procedure for D is to maximize the probability of D making a mistake. Within the space of arbitrary D and D, we roll a D20 and check for damage. This system corresponds to entropy maximization, which ensures a timely heat death. Experiments would have demonstrated the potential of the framework, but most of our results were degenerated in the process of running them.

1 Introduction

The promise[3] of deep learning is to discover rich, hierarchical models [5] that represent probability distributions over different kinds of data, such as natural images, audio waveforms containing speech, and symbols in natural language corpora (see Figure 1). All of this structuring of data works to decrease entropy by creating discriminators that are able to classify this data into well-defined labels. Furthermore, we now see the success of deep generative models due to Goodfellow et. al [5] and Kingma et al. [8], which further accelerates the pace of structured data generation.

Figure 1: The power of deep learning. [photos: Bobolas, 2009 [1], Maley, 2011 [10]]

All of this model discovery is creating too much information. At this rate, we will outstrip the second law of thermodynamics and begin to decrease entropy in the universe (see Figure 2). In order to maintain reality as we know it, we present the Degenerative Adversarial Network, or DAN, which sidesteps the successes of deep learning models in order to maintain a steady and healthy pace towards the sweet release of heat death.

In this proposed adversarial degeneration framework, the degenerative model is pitted against an adversary: a discriminative model which attempts to distinguish whether actual data has been degenerated or whether the observed sample is true noise. The degenerative model can be thought of as a team of steamrollers, flattening data into a uniform distribution for maximum entropy. The discriminative model can be thought of as a team of protractors, trying to determine if its input has been properly flattened into true noise. Competition in this game will drive both groups to improve until the discriminator cannot reliably distinguish between generated noise and degenerated data.

*Currently looking for grad school.
†Please accept me into your lab.
[3]unfulfilled

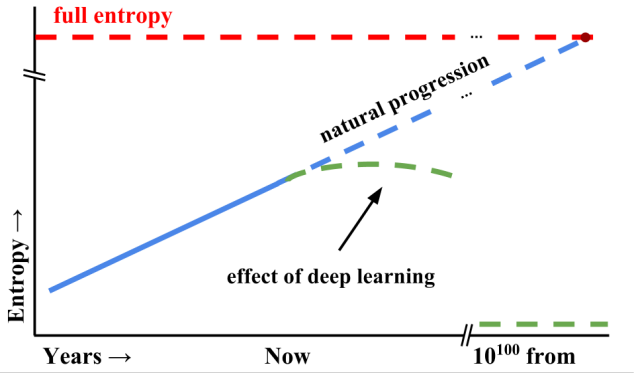


Figure 2: Plot showing the dangers of deep learning. In hindsight, this plot generates information so please refrain from looking at it.

This framework can give specific algorithms for degeneration and discrimination. We explore the case in which the degenerative model destroys data by passing it through a multilayer after being perturbed by noise and the discriminator model is also a multilayer perceptron. We refer to this special case as a *DAN* and show that we can train both networks using backpropagation in an end-to-end fashion. Uniquely to our approach, there is no need to actually code this network.

2 Related Work

Training adversarial networks is infamously hard [12], because the optimization objective equates to trying to find a Nash equilibrium in a non-cooperative game. We found this to be even more complicated when degenerating, because the procedure makes it very easy for the Degenerator to output images of PhD students[4].

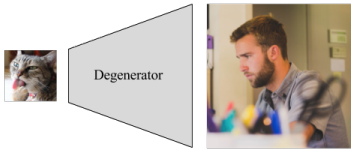
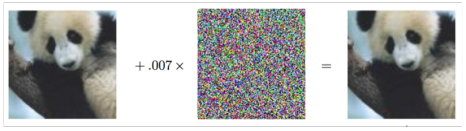


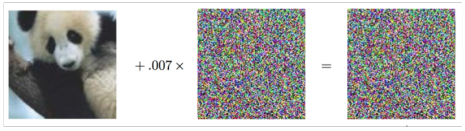
Figure 3: Example of bad Nash Equilibrium for the DAN model, the degenerate result is a PhD candidate hard at "work" on his thesis.

To solve this, some work in the field has argued that a balance between the two adversaries needs to be found in order to stabilize the game and avoid local minima. However, Goodfellow [4] shows how a more robust solution involves creating an overpowered adversary Discriminator, which ensures that the values of collaboration and looking past one another's differences will not come into play. Therefore, we proceed with the adversarial technique. Our methods are described below.

Goodfellow et al. [6] also use adversarial techniques to degenerate data. However, their method is limited in that it is only able to trick other neural networks. While this is useful for slowing down the pace of data creation by generative models, it is not sufficient for our objectives as their results preserve enough structure that humans can still discriminate with ease. The method we present is robust even to human discriminators.



(a) Adversarial Degeneration in Goodfellow et al.



(b) Adversarial Degeneration using DANs

Figure 4: Comparison of the results presented in [6] and ours. Note how DANs are able to degenerate data well enough to trick both neural networks and humans, whereas Goodfellow et al. are only able to trick neural network models.

[4] with our apologies to degenerates

3 Degenerative Adversarial Nets

3.1 Description

The two models D and D play the following role-playing game:

$$\text{dom}_D \text{ sub}_D \; V(D, D) = \mathbb{E}_{Geoff}[\log(D(\text{eep}))] + \mathbb{E}_{Hinton}[\log(1 - D(\text{addy}))] \tag{1}$$

Some might say that this notation is confusing. Those people would be wrong. ■

In practice, Equation 1 provides absolutely no information at all on how to train either D or D. We think this is OK, as reproducibility is the least important aspect of science.

The adversarial framework is most straightforward to apply when we straight copy paste someone else's code. As such, to learn the degenerator's distribution p_d over x, we don't do anything at all and just use the method from [5], except we ignore the given input and replace it with noise generated using Python's `random` module.

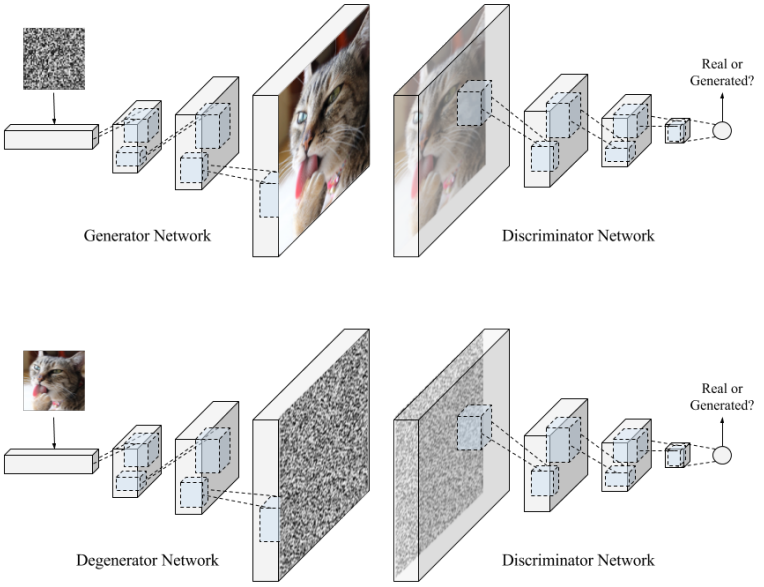
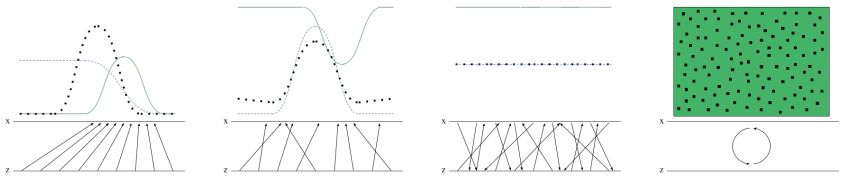


Figure 5: Comparison of the GAN (top) and the DAN architectures (bottom). In the former, noise is used to generate data. In the latter, data is degenerated into noise.

See Figure 6 for an approximately probably equally formal [5] explanation of our approach. We would include more explanation here, but our model is basically just a GAN so we refer the reader to Goodfellow et. al[5].

In the next section, we present some theoretical results about our adversarial degeneration, essentially showing that the training criterion presented above allows one to lose all of their data.

[5] i.e. not at all



(a) (b) (c) (d)

Figure 6: Degenerative adversarial nets are trained by simultaneously updating the discriminative distribution (D, blue, dashed line) so that it discriminates between samples from the original data (black, dotted line) p_x from those of the degenerative distribution p_d (D) (green, solid line). The lower horizontal line is the domain from which we sample noise. The higher horizontal line is part of the domain of $\mathbf{x}$, which we destroy. The upward arrows show how the mapping $\mathbf{x} = D(z)$ imposes the uniform distribution p_d on transformed samples, which later overwrite the original $\mathbf{x}$. D learns to scatter uniformly. (a) Consider an adversarial pair before divergence: p_d is dangerously similar to p_{data} and D is much too accurate a discriminator. (b) In the inner loop of the algorithm D is trained to discriminate samples from data that have been ruined, and starts to diverge. We also see that the data distribution has started to disperse, and that the degenerator distribution is heavily unlearning the data. (c) After an update to D, the gradient of D no longer exists. We see that we have reached uniformity of data. (d) After several steps of training, if D and D have enough capacity, there is no need to map the noise to the degeneration anymore. The DAN has learned to generate its own noise (hence the self loop) and the data distribution has reached what we call "super-uniformity," which is how we boost entropy at a high enough rate to counteract others in the field. A key step in this procedure requires that we mention it here in this figure only, and never mention it in the rest of the paper.

3.2 Method

We download an out-of-the-box GAN model [7] and repurpose it to a DAN – by which we mean we trained it on a new dataset without modifying a single line of code. We present our algorithm below.

Algorithm 1: Degeneration algorithm.

Input: Internet connection, GAN, Python noise module
Output: Noise

1. Open browser.
2. Go to www.google.com.
3. Type in "google."
4. Click on Google.
5. Type in "gan tensorflow".
6. `git clone`
7. Generate noisy images in Python.
8. Train your GAN on the noise.
9. Leave on the counter for 15 minutes to cool.
10. Overwrite your original data with the generated noise.

We believe this algorithm can be generalized to other kinds of models. This would involve optimizing the algorithm's search strategy for those other models (e.g.: you might need to type "infogan tensorflow" in step 5). We leave this discussion for future work.

Additionally, in our algorithm we are overwriting the original data through manually-written disk writes. We believe in the potential of end-to-end learning in tackling this task, but we also leave this for future work. Some AI ethics alarmists might claim such a model would accelerate the impeding doom of civilization [13]. However, we see this as a reasonable alternative to heat death, and thus posit that it's a worthwhile pursuit.

In other subfields of Machine Learning, one would use a dataset like MNIST [9] or ImageNet [3]. For our purposes, we'd need a randomness dataset, such as the ones used in the cryptography field.

However, in a desperate bid for citations, we ignore existing options in favor of fabricating our own, which we call DANOISE[6]. We discuss the implications of this decision below.

When training a DAN, it's important to keep in mind that the generated data must represent true randomness. Unfortunately, the availability of true randomness is scarce in a deterministic Turing machine, so we settle for the python approximation `random` module.

It's crucial to note, however, that the random module does not give you true randomness. The validity of this claim is based in the fact that we seek to model true randomness through a neural network. If randomness were readily available through a simple module import then there would be no point in using Deep Learning. ;)

4 Theoretical Results

The degenerator D implicitly defines a probability distribution p_d as nothing. Therefore, we would like Algorithm 1 to converge to nothing, which is equivalent to diverging to a uniform distribution that maximizes entropy. The results of this section are organized in a similar manner: uniformly random and without meaning.

We will show in section 4.1 that our role-playing game has no meaning, but that the network has a global optimum when $p_d = p_{\text{uniform}}$.

4.1 Global Optimality of Entropy

We first consider the optimal discriminator D for any degenerator D.

Proposition 1. *For D fixed, the optimal discriminator D gets overwritten by D.*

$$D^*_D(\boldsymbol{x}) \equiv \texttt{/dev/urandom} \tag{2}$$

Proof. The training criterion for D (whichever D) does not really matter. What does matter is that no matter what the trained degenerator is, the final step is to overwrite your data. Therefore, the discriminator simply ceases to exist. This is accomplished by overwriting the data with values that are equivalent to `/dev/urandom` (since D is optimal). □

Theorem 1. *The global minimum of the training criterion occurs when the universe reaches maximum entropy. However, reaching a local optimum is equivalent to enabling D to degenerate data (and write to disk) into uniformly random noise. Therefore, any local optima is actually just a saddle point on the way to global optimum.*

Proof. We have an elegant proof, however this saddle point is too itchy, forcing us to abstain from including the it in this paper. In future, we plan to also abstain from including proofs if the margins are too small. □

4.2 Divergence of Algorithm 1

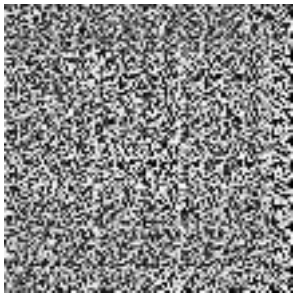
Proposition 2. *Regardless of the capacity of D and D, if at each step of the algorithm we ignore the meaningless criterion of and overwrite data, the algorithm will always diverge.*

Proof. *We* wou l d *lik e' t o s h' o w t h a t #%fs[-3]* □

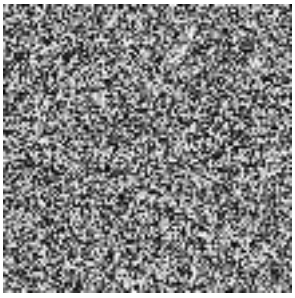
Note: Unfortunately, we lost the above proof as well other results due to the model overwriting them.

[6] we're accepting suggestions for potential acronyms that justify this dataset name

5 Results



(a) Sample degenerated by DAN after 24 epochs.



(b) Random sample taken from Python.

Figure 7: Comparison of results from DAN and results from Python.

The noise examples generated by DAN look very visually similar to a sample from a true random source (or its equivalent python non-approximation), as can be seen in Figure 7. Therefore, it's probably approximately correct to say we've established the state of the art in degenerating data. The definitive results were destroyed by our preliminary experiments in end-to-end training, so we present a novel metric of Data Degeneration: percentage of data destroyed ($\%_{dd}$). We compare our methods with other models in Table 1

Model	$\%_{dd}$ in training set	$\%_{dd}$ in my family photos album
VGG trained on ImageNet	0	0
Inception trained on Britney Spears MP3s	0	0
AlphaGo trained on MNIST	0	0
DAN trained on DANOISE dataset[7]	**100**	**100**

Table 1: Comparison of different models on the task of permanent data degeneration. It's clear from this comparison that our architecture is inherently superior to these other ones, because of the bold font highlighting the DAN results.

6 Conclusion

We have presented an entirely new[8], model that achieves the state of the art in data degeneration. With it, we get the field one step closer towards stopping Big Data terror and maintaining a steady pace towards heat death.

We hope that this has inspired the reader to help us further these novel Data Degeneration models. We conclude by presenting a few examples of potentially interesting and useful future research directions:

Inspired by InfoGAN [2], InfoDAN would preserve the property of uninterpretability of latent space, by degenerating the data, as well as any structural features it is composed of.

Similarly to work by Radford et al. [11], DCDAN is a model with the same architecture as a DAN, but with twice the number of convolution layers, and half as many learnable parameters.

Lastly, also inspired by the work of Chen et. al [2], EntropyDAN could use the data input as a latent representation of how true the generated noise is.

[8] i.e.: ~~plagiarized~~ repurposed

References

[1] Bobolas. brain-neurons.

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[4] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[7] Google. First result when googling "gan tensorflow".

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[10] Maley. neuron.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[12] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[13] Tim Urban. The ai revolution: The road to superintelligence.

7 Appendix A

Because 7 pages wasn't enough.

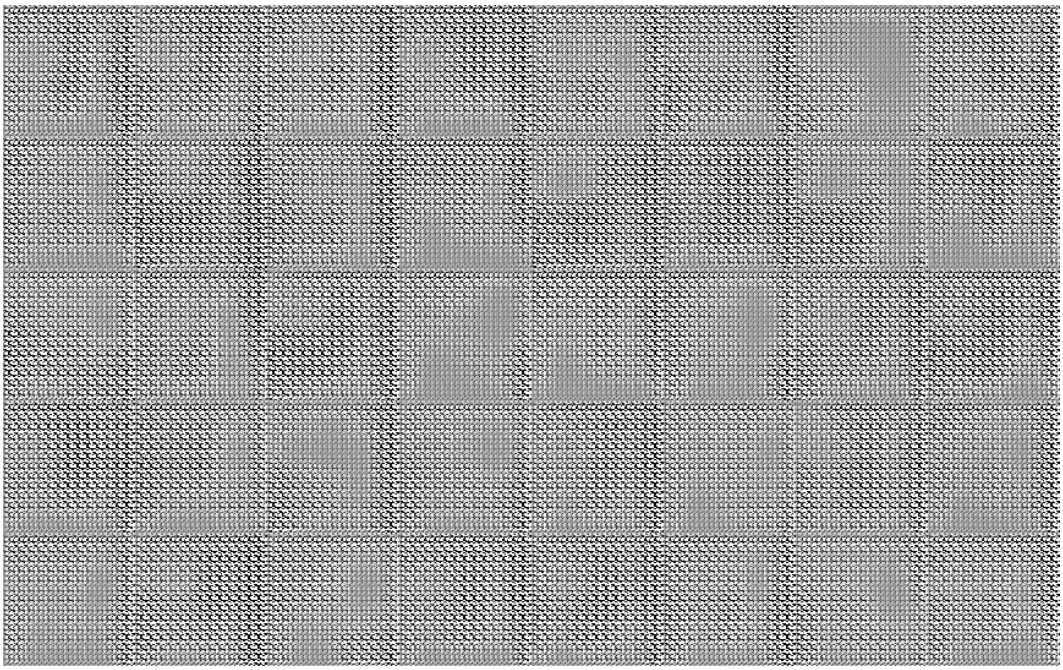


Figure 8: Generated image samples after one pass through the dataset. The degenerator is still trying to unlearn the structure of the input data.

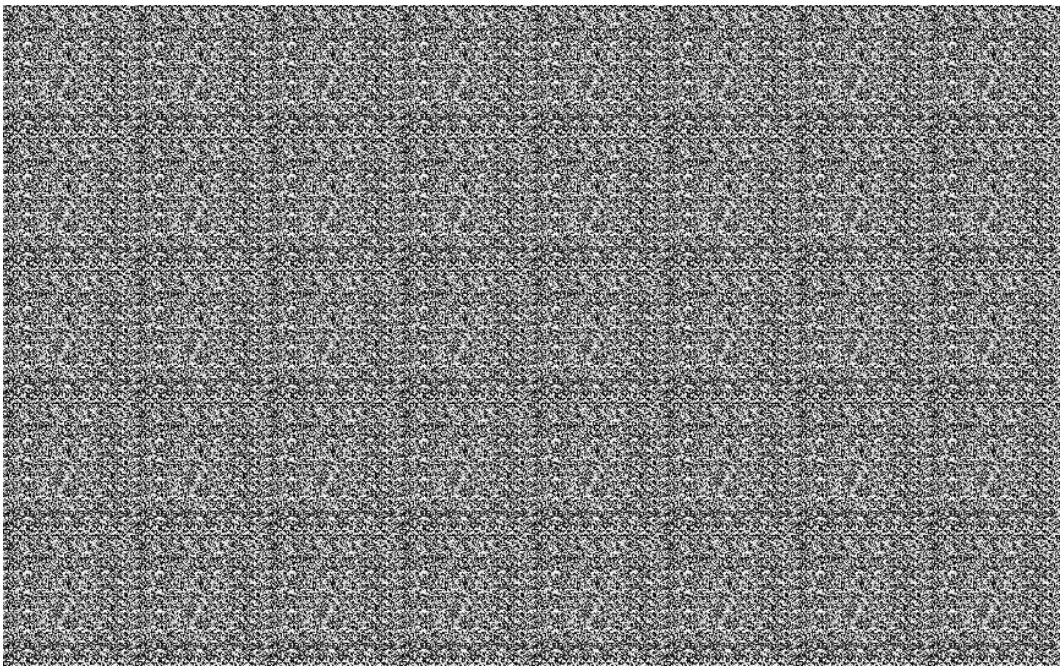


Figure 9: Generated image samples after 15 passes through the dataset. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples.