Robot vs. Robot:
Evening the Field in Deep Convolutional GANs for Small Datasets

Morgan J. Weaver
Department of Computer Science and Software Engineering, Seattle University
Seattle, Washington, 98122 USA
weaverm1@seattleu.edu

Abstract

Generative Adversarial Networks (GANs) have enjoyed increasing popularity in recent years due to their unique properties among unsupervised machine learning paradigms, namely the ability to continuously improve generated output quality while simultaneously improving discriminative ability via the use two neural networks in an adversarial manner. This is accomplished by setting up a kind of arms race between a generative network and a discriminative network, with the generator attempting to produce output indistinguishable from actual samples, and the discriminator tasked with discerning generated vs actual inputs.

Here, we attempt to improve a Deep Convolutional GAN (DCGAN) in order to improve the loss ratio and convergence between generator and discriminator, which, for small datasets, can leave the generator lagging behind the discriminator. Hyperparameter modification and tuning indicate promising avenues for finding an appropriate ratio of generator to discriminator iterations to approach parity between the two networks' respective loss, for overall improvement in generated image performance *and* discriminator performance in a small image training dataset.

Introduction

Generative Adversarial Networks, though fairly new in the field of unsupervised machine learning, have proven very promising as a new avenue of simultaneous generation and classification based on sample sets of 2D and 3D image, text, music, and other input without the need for extensive labeling of training sets.

Prominent machine learning researcher and director of AI at Facebook Yann LeCun was recently quoted,

> *There are many interesting recent developments in deep learning…The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.*

Much of the current interest surrounding GANs has been focused on their ability to train on very few pieces of labeled data and produce realistic outputs as well as complete other useful tasks, such as sharpening and adding detail to blurry and incomplete images including faces with an obstruction or a section removed, and second-pass processing of intermediate generated images (see Related Work).

A basic GAN (see Figure 0) is arranged as follows: a set of two neural networks are implemented, with one tasked with directly learning the conditional probability P(y|x), where x is some input from either a dataset or artificially produced by the generative model, and y is some class label. The other neural net, called the generator, is tasked with learning the joint probability of input data and labels simultaneously, and is further used to generate new samples based on this learning. Generative models have the useful property of being able to create novel outputs based on extracted information even when there are not labels, which is what makes this model possible, as the generator's outputs are fed to the discriminator, which attempts to discern whether the input was generated or from a non-generated dataset.

This can also be thought of as a two-player minmax scenario for the Generator (G) and Discriminator (D) represented by the function,
in which x is an actual image from the provided data distribution p_{data}, and z is a vector populated with noise, obtained from distribution p_z.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))],$$

Caveats

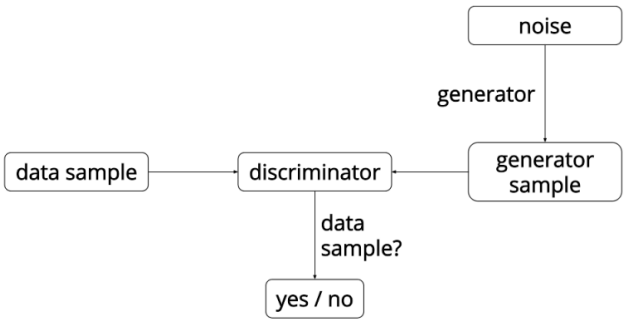

Figure 0: General structure of a GAN. Source: https://ishmaelbelghazi.github.io/ALI

There are several inherent performance issues with GANs, many of which are simply the inheritance of whatever underlying neural network model is chosen, be it feed-forward, recurrent, or convolutional. One specific issue as mentioned in Goodfellow et al. is the so-called "Helvetica Scenario", or mode collapse, in which the generator maps several input vector points (sampled from random noise) to the same output, resulting in repeating, similar images, as was seen in this experiment. Unlike non-adversarial, supervised networks, GANs do not produce image features that are not actually in the training distribution—for example, a mouse that is somewhere *between* grey and white would not result from a GAN, as opposed to a mouse whose coloration came from sampling from the probability distribution {white, grey} and is thus assigned grey.

GANs are also notorious, regardless of neural net implementation, for the requirement of careful tuning of hyperparameters concerning the ratio of generator to discriminator training, which this paper will explore in more detail.

Another known issue for the specific library utilized in this report among other GAN libraries is fast convergence resulting in poor output quality, for which the author of DCGAN-tensorflow attempted to correct for by updating the generator network twice (iterating over a training image batch) for every time that the discriminator function is updated. While better than a 1:1 ratio, this is not ideal.

The present research endeavor aims to find an optimal ratio of iterations for the generator vs the discriminator, as the discriminator tends to demonstrate very low loss, while the generator may have a lass ratio of over 4000:1 relative to the discriminator. By locating the ideal ratio, we aim to improve the quality of the generated images while increasing the quality of output early on in training iterations.

Related Work

In the seminal publication by Goodfellow et al. (2014), the researchers propose for the first time the concept of the Generative Adversarial network with two simultaneously, adversarially trained neural networks (multilayer perceptrons, here). The authors cite the contributions of this new architecture as: 1) finding a way to avoid the so-called intractable probabilistic calculations in deep generative models related to maximum likelihood estimation, 2) avoiding the use of Markov chains for training, and 3) taking advantage of piecewise linear units in a generative network.

Radford, Metz and Chintala (2016) go on to refine this model, in the now-widely adopted DCGAN, with the major advantages being that the model sidesteps MLE as in GANs, leverages recent advances in Convolutional Neural Network (CNN) architectures, doesn't require supervised learning, produces more stable output than GANs, and has the useful property of feature extraction. The last quality offers interpolation and manipulation of specific filters to tune the desired output. E.g., in a set of facial images, one might extract a vector depicting a face with sunglasses, and another for a woman's neutral face, and interpolate between the two vectors to intermediate stages, or add further features in rich multidimensional space. Vector arithmetic allows addition and subtraction of these vectors.

The concept of the DCGAN is even further refined in Springenberg (2016), in which the CatGAN, or Categorical Generative Adversarial Network, is introduced as a model that trains on unlabeled or partially-labeled data, extending a discriminative clustering technique known as regularized information maximization (RIM). RIM and its relatives, such as maximum margin clustering (MMC), discriminate between categories with an emphasis on boundaries rather than centroids or class exemplars as in generative clustering methods such as k-means and Gaussian mixture models. The CatGAN model attempts to distance itself from the GAN by eliminating some of the problems associated with overfitting, such as preserving too much of the information present in the input samples.

A recent and very successful approach was demonstrated by Zhang and colleagues (2016) with a layered GAN model, in which a sole text description is provided (e.g., "This bird is small and red with a short beak and a white breast"), and after encoding into 100+-dimension latent space representation, a phase-1 GAN generates an image with primitive shapes and colors, which is then passed with the original text to a stage 2 GAN which refines the image, adding vivid photorealistic details such as sharpened edges and defined shapes as opposed to the blurring and mutation seen in many GAN models. Relative to previous approaches, Zhang and colleagues' StackGan is capable of producing very high quality photorealistic images, a valua-

ble contribution to the current body of GAN knowledge and implementation.

An additional and recent contribution to GAN research consists of the Wasserstein GAN, by researchers Arjovsky, Chintala and Bottu (2017), which skips the typical Gaussian noise component needed in neural nets for training in order to avoid estimating P_r, the actual data distribution density, a technique that leads to blurring and fuzzing of output images. The researchers optimize a typical GAN structure by using their GAN architecture to minimize an approximation of the Earth-Mover (or Wasserstein) distance, which avoids estimating the density of the set of real data in the traditional manner of probability density estimation. The Wasserstein/Earth mover distance is given here,

where Π(Pr, Pg) is the set of all joint distributions γ(x, y). This γ(x, y) would indicate how much "mass" (hence the Earth Mover denomination) must be moved from *x* to *y* in order to transform the distributions P_r into the distribution

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right],$$

P_g. The Earth Mover, a.k.a. Wasserstein, distance then is the price of the cheapest transport plan, replacing the existing maximum likelihood paradigm that is the basis of most unsupervised learning in GANs:

$$\max_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

This approach eliminates much of the mode dropping issue mentioned previously, as well as allowing for less noise in output images as the Gaussian noise is no longer needed.

Methods

Libraries

This project utilized a widely available Python v2.3+-based DCGAN library found on GitHub that was based upon the paper by Goodfellow (2014), and adapted for Tensorflow (https://github.com/carpedm20/DCGAN-tensorflow), which was later forked and modified (https://github.com/morganjweaver/DCGAN-tensorflow) by the author to accommodate the extra hyperparameter g_advantage, which determines the number of iterations the generator is allowed to undertake in training relative to each iteration of the discriminator. This hyperparameter was input as a flag in the basic training command.

Dataset

The dataset consisted of 192 color mouse faces from a larger set of animal faces. Each image measured 150 x 150 pixels in jpeg format.

Hardware

All training and testing was completed on an Amazon EC2 P2.xlarge instance, with a single Nvidia Tesla K80 GPU (approximately 4 teraflops) running Ubuntu Linux.

Results and Analysis

Training was run for between 500 and 5000 64-item batched epochs for a variety of generator:discriminator iteration ratios. For the purpose of this project, we have included results for the default 2:1 ratio of generator:discriminator iterations, the experimental 4:1 ratio, and the experimental 50:1 ratio. As loss ratios were as great as 4669:1 between generator and discriminator, Figure 1 displays the results in *log form* over the course of 20 training epochs for a less dramatic yet still meaningful comparison between the three ratios over successive training epochs. An ideal result for the log of the ratios would be 0, the log of a perfect 1:1 ratio in which the generator and discriminator are closely matched, which also results in higher quality output images containing less blurring and warping of the subects. As shown in Figure 1, the computationally expensive 50:1 hyperparameter optimization has the best log loss ratio of those examined.

Further support may be found in the accompanying image samples, which compare image progress at 100 iterations vs 400 iterations for each of two experimental ratios

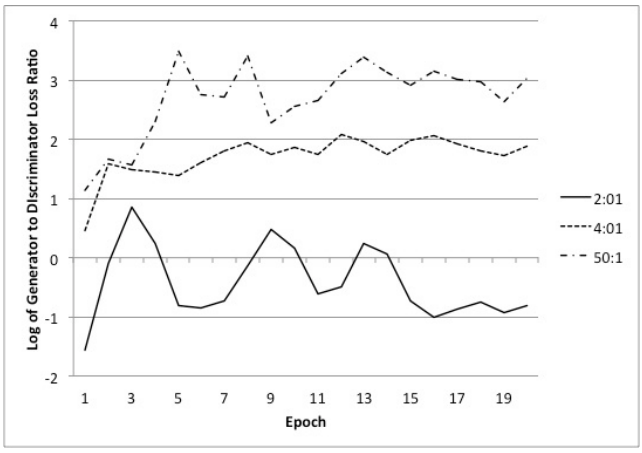

Figure 1: Log of Loss Ratio of Generator to Discriminator

and the default ratio. Figure 2 depicts a rather blurry and most un-mouselike set, with remarkable mode collapse, and little improvement over the next 300 epochs. Overall

image quality is slightly higher for the 50:1 set of images in Figures 6 and 7, though mode collapse is still an issue, with the 4:1 tuning in Figures 4 and 5 being somewhere in the middle in overall quality and showing slightly less improvement *between* sets than the 50:1 tuning.

These results would suggest that a nonlinear improvement can be obtained in adjusting the competitive advantage of the generative model.

Discussion

The overall performance of the GAN explored herein saw a marked improvement in image output quality with the addition of the g_advantage hyperparameter, which helped even the competition between generator and discriminator by allowing extra training iterations for the generator, also avoiding fast convergence for small datasets, which would normally result in poor quality output.

Though a fourfold advantage to the generator showed great improvements, the advantage conferred by this approach did not scale linearly, as the 50:1 tuning was marginally better but still showed some blurriness as can be seen in Figures 6 and 7, with loss ratios appreciably closer to the ideal 1:1.

Further experiments might include larger batch sizes and greater numbers of iterations for the different generator advantage ratios, which were limited by the EC3 instance's 80 GB storage, which was quickly depleted by logs and output images.

Larger datasets, further sampling at high generator:discriminator iteration ratios (upwards of 5000, preferably), and various hyperparameter modifications such as batch size are also further avenues of exploration.

Mode collapse was seen at all tunings, though a larger image training set would probably alleviate this to some extent, as would a newer GAN model such as Wasserstein.

Conclusion

Generative Adversarial Networks, specifically those employing underlying Deep Convolutional neural networks, show promise for novel unsupervised and semi-supervised machine image learning applications, especially those with limited labeled training data. These applications span many fields, from entertainment to cryptography, and recent research efforts have brought creative and interesting new possibilities to light, such as 2-phase Deep Convolutional GANs, StackGANs, Wasserstein GANs and CatGANs.

The research at hand contributes to the basic Deep Convolutional GAN model by taking the refinement of two training iterations for the generator network a step further and implementing this as a hyperparameter for ease of training refinement, conferring a competitive advantage to the generator network which may be especially useful for optimizing training in small datasets.

Acknowledgements

The author gratefully acknowledges the mentorship and insight of Leo Dirac, Principal Engineer, Amazon, for his intuitive understanding of GAN architectures and willingness to expound upon them.

Reference

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein GAN. *Arxiv preprint* arXiv:1701.07875v2.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S., Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems,* 2672-2680.

Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representational learning with deep convolutional generative adversarial networks. *arXiv preprint* arXiv:1511.06434.

Springenberg, J.T. 2016. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *International Conference on Learning Representations.*

Zhang,H.; Xu, T.; Li, H.; Huang, X.; Wang, X.; and Metaxas, D. 2016. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint* arXiv:1612.03242v1.

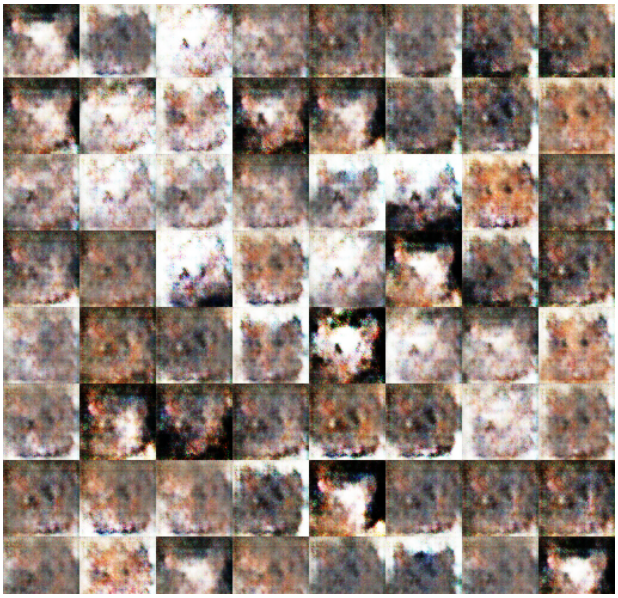

Figure 2: Image produced after 100 iterations, default ratio

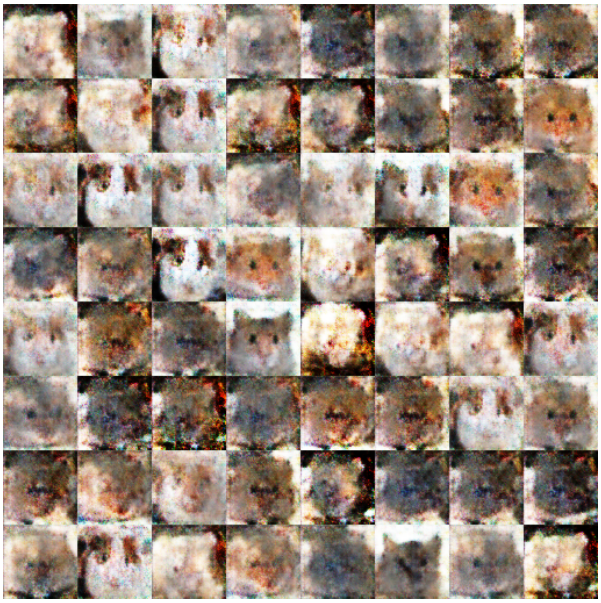

Figure 3: Image produced after 400 iterations, default ratio

Figure 4: Image produced after 100 iterations, 4:1 generator:discriminator advantage

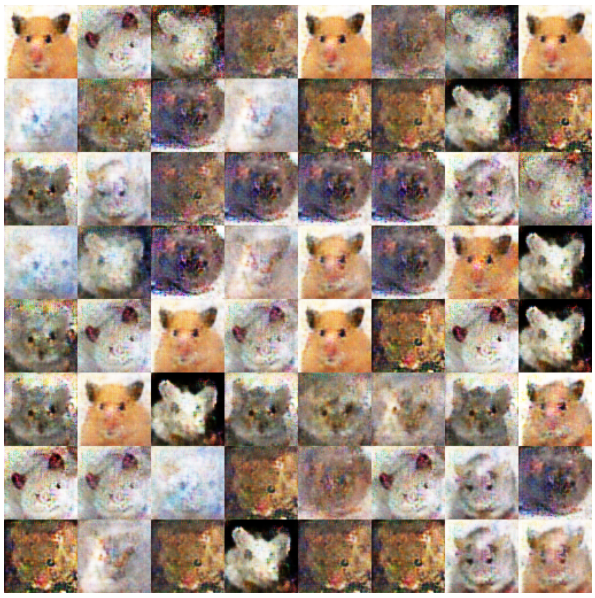

Figure 5: Image produced after 400 iterations, 4:1 generator:discriminator advantage

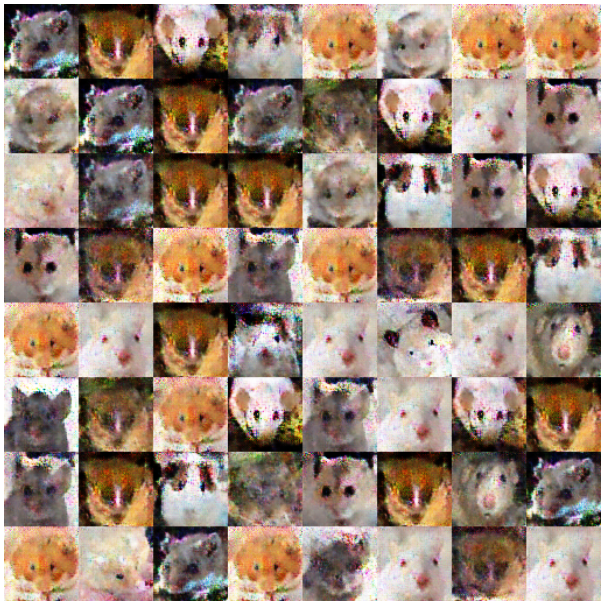

Figure 6: Image produced after 100 iterations, 50:1 generator:discriminator advantage

Figure 7: Image produced after 400 iterations, 50:1 generator:discriminator advantage