# UNSUPERVISED REPRESENTATION LEARNING ON CLIP ART WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

**Jared Wilber**
jdwlbr@gmail.com
4th year Statistics undergraduate

**Nicolas Mon**
n_mon@berkeley.edu
1st year Master Info Mgmt & Systems

## ABSTRACT

General Adversarial Networks (GANS) have gained a lot of popularity in recent years, in particular for unsupervised tasks such as representation learning. However, these networks are notoriously hard to train, with noisy generated images and unstable loss in the early stages of training being two major problems. In this work, we tackle the novel task of training GANS on clip-art images - in particular, on Pokémon. In addition, we propose a technique for stabilizing the GAN's learning by training on images preprocessed with the assistance of a convolutional autoencoder. We found training on these reconstructed representations of the original data resulted more stable learning and faster convergence for our GAN. Finally, we evaluate the approach of first training a GAN on these preprocessed images and then swapping them out with the original images mid-training. Although we hoped that this would allow the GAN to converge to the finer detailed original images faster, we discovered that overall this method in fact resulted in less stable learning in the long run as a result of the data swap.

## 1 INTRODUCTION

Generating data and feature representations from unlabeled datasets has been an active area of research. These newly generated images can then be used for a variety of tasks: in computer vision, for example, such images can be used to augment supervised learning tasks for

image classification in computer vision, or to generate new landscapes or agents in video games. Multitudes of images have been generated from network architectures, but there is lacking research specific to clip-art. There are various different methods to generate new images, such as Generative Adversarial Networks (Goodfellow et al., 2014) or Variational Autoencoders (Kingma et al., 2013). In this paper, we opt to use Deep Convolutional Generative Adversarial Networks (Radford et al., 2015) to model clip-art images. In addition, we use a traditional Convolutional Autoencoder to preprocess our image data to test if training on these preprocessed images would achieve result in more stable training of our DCGAN.

We make the following contributions in this paper:

- We propose using convolutional autoencoder-fed images as a pre-processing step to create better GAN convergence.
- We evaluate a novel technique that attempts to achieve faster GAN convergence on fine-grained images

## 2 RELATED WORK

### 2.1 Unsupervised Representation Learning of Image Data

Learning to represent unlabeled image data is a common problem that has been addressed but numerous computer vision techniques. One such approach is to train convolutionally stacked autoencoders to learn to encode images into a lower dimensional space, and then reconstruct the images from the lower dimensional space as accurately as possible (Vincent et al., 2010).

### 2.2 Generating Images

The task of generating new images is also well researched, and numerous different approaches have been proposed. Among them, one such proposed approach is an alteration on the traditional autoencoder discussed in Vincent's paper. This approach learns also learns to encode images into lower dimensional space, with the added stipulation that the latent representation space follows an intractable distribution, such as a Gaussian (Kingma et al., 2013). In doing so, you can adjust the latent variables according to their distribution to produce nearly infinite images from your latent space. This approach also allows for vector arithmetic on the latent feature space to produce representations of new images.

Another proposed approach to the problem of image generation is to use Generative Adversarial Networks. In this architecture, a generator uses stacked deconvolution layers to produce an image from a vector of noise, while both generated and original images are passed to

a discriminator which uses stacked convolutional layers to classify images as ‘real’ or ‘fake’ (Radford et al., 2015). These networks use the Minimax loss approach to train the networks together. As the generator learns to produce images that look more and more like the original data, the discriminator gets better at telling the real images from the generated ones. The intuition is that the generator will eventually learn to produce images that are indistinguishable from the original data, thus also learning an effective internal representation of the image data.

## 3 APPROACH AND MODEL ARCHITECTURE

To train our GAN model, we will use an Amazon Web Services (AWS) E2 server. For our AWS server, we have selected to use the p2.XL instance to begin training on the MNIST dataset, which is the smallest AWS instance type within the ‘GPU compute’ category. We have set up the needed dependencies on the server and implemented GAN on MNIST data as well as a dataset of Dog images in order to get comfortable with GAN architecture. Once we were comfortable with our performance on the MNIST and Dogs, we started using our GAN on the scraped dataset of pokemon clip art images.

For our DCGAN model architecture, we have chosen to implement a model very similar to the one used in *Unsupervised representation learning with deep convolutional generative adversarial networks* (Radford et al., 2015). The generator featured 4 convolutional layers, each of which used relu activations with batch normalizations before each activation. Every activation in the generator network is relu, with the exception of the final layer which used tanh activation. For this reason, the pixel values of our data were normalized between -1 and 1. A depiction of the generator’s architecture is attached below.

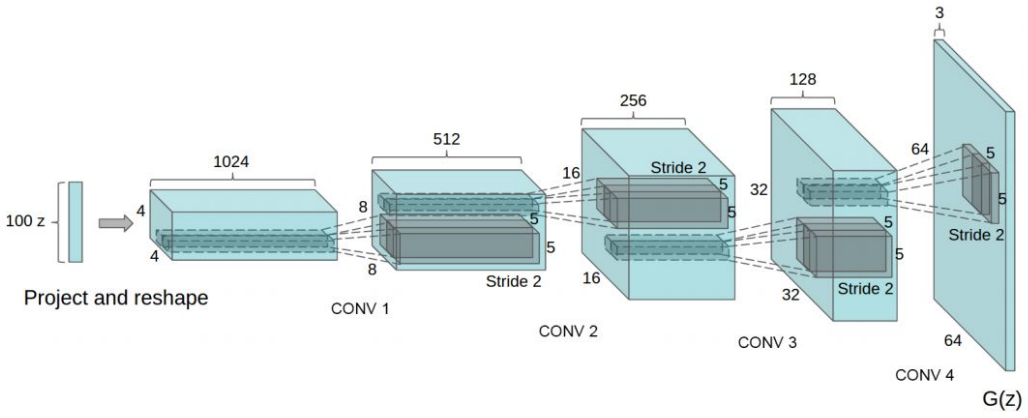


Figure 1: Generator architecture for our DCGAN

Via repeated training attempts, we learned that when the discriminator's architecture is constructed as a reverse mirror image of the generator, it resulted in the discriminator being too strong. When this happened, the discriminator's loss would shoot to 0 and our generator would not be able to learn effectively. To avoid this, we removed two convolutional layers from the discriminator, as well as batch normalization from all layers. We also decreased the learning rate for the discriminator to be half of that of the generator. In doing so, we achieve a nice balance of strength between the generator and the discriminator that results in better training.

# 4 DETAILS OF ADVERSARIAL TRAINING

## 4.1 MNIST

To familiarize ourselves with the GAN architecture and training, we initially begun with using MNIST data to train our GAN on. With this simple dataset, we were able to produce very realistic looking images with the GAN generator. See the generated images below:

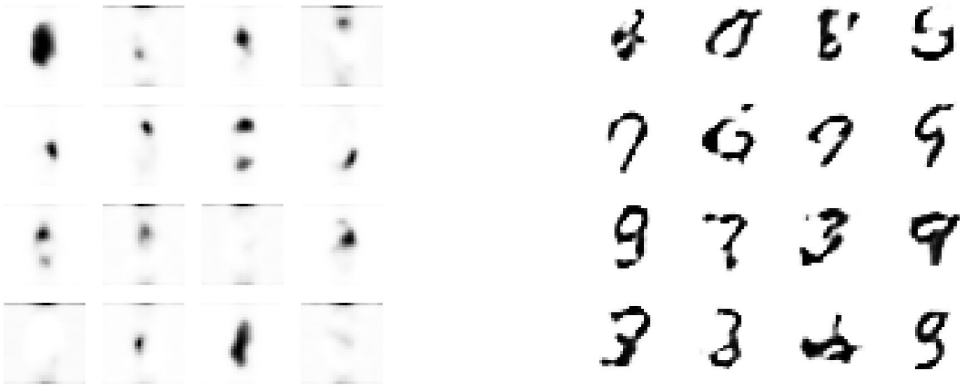

Figure 2: Early (left) and late (right) stages of MNIST GAN generated images

## 4.2 Dogs

After MNIST, we decided to move on to a more complex image dataset, so we used pictures of dogs from the Kaggle Cats vs Dogs dataset. For simplicity, we converted these images to grayscale and resized them to 64x64x1.

Compared to MNIST, the GAN had a much harder time training on dog images, as the Dog images varied a great deal in background and what orientation of the dogs in the photographs. Therefore, we decided to change to a dataset of Pokemon clipart, as these images are much more consistent and all have the same white background. Because of this, we believe the GAN will have an easier time learning to represent this more consistent visual data.

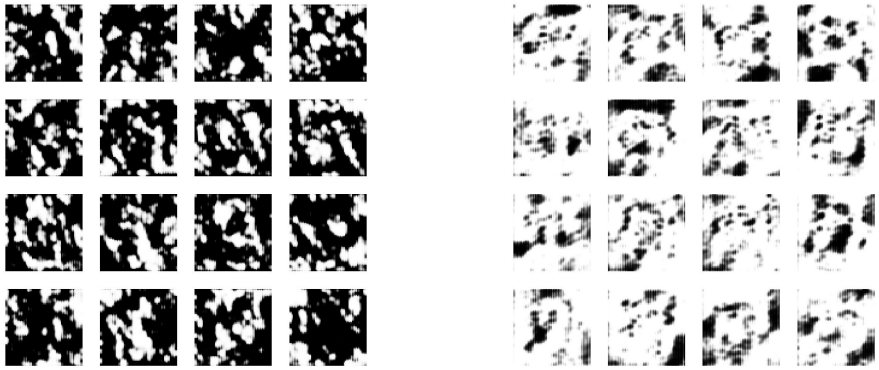

Figure 3: Early (left) and late (right) stages of GAN generated images on dogs dataset.

## 4.3 Pokemon Clip Art

We scraped clip art of all 701 Pokemon to create a consistent dataset of pokemon images that we could train our GAN on. Although for the dogs dataset we converted to grayscale, we kept the RGB colors in this dataset while still compressing the images before training, resulting in 701 samples of data with dimensionality 64x64x3. A sample of the compressed pokemon images is shown below.

Figure 4: A sample of the 64x64 pokemon images our GAN was trained on

In addition to compressing the Pokemon images to 64x64x3, we also experimented with further preprocessing the images with the assistance of a convolutional autoencoder. By first training a traditional autoencoder, and then passing our training data all the way through the autoencoder architecture (both encoding the images and decoding the latent vector representations), we obtained a new dataset of blurrier representations of our original images. Results of this process, as well as a visualization of the latent vector space is shown below.

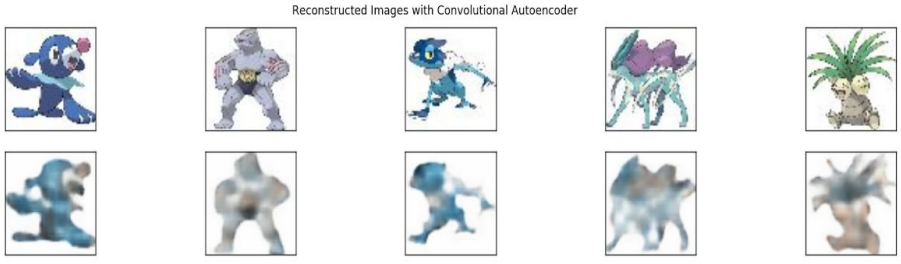


Figure 5: Original images (top) and results of autoencoder preprocessing (bottom)

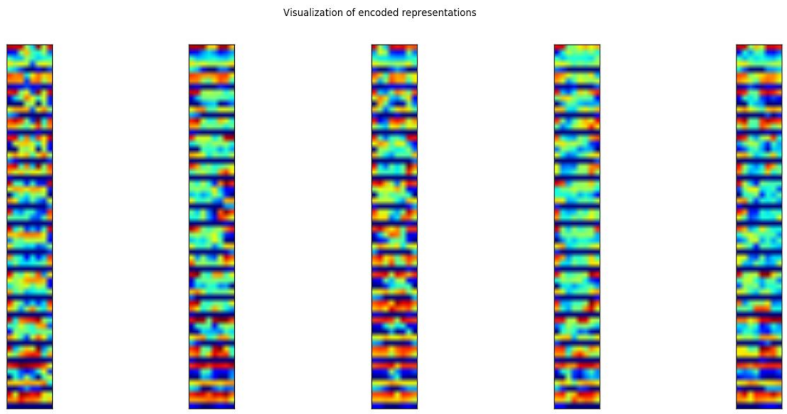


Figure 6: Visualization of Encoded Pokemon Images

Dealing with the original clip art images resulted in very unstable training for our DCGAN. Our hypothesis is that this is a result of the fine details present in clip art images, details the model has difficulty learning. When trained on the original images, the generated images were very noisy and did not seem to take a recognizable form. To mitigate this problem, we came up with a solution in which we first encode the images in a lower dimensional representation (via a convolutional autoencoder), and then train the DCGAN on images reconstructed from these latent vector representations. Not surprisingly, the GAN has an easier time learning to represent these reconstructed images, and training is more stable.

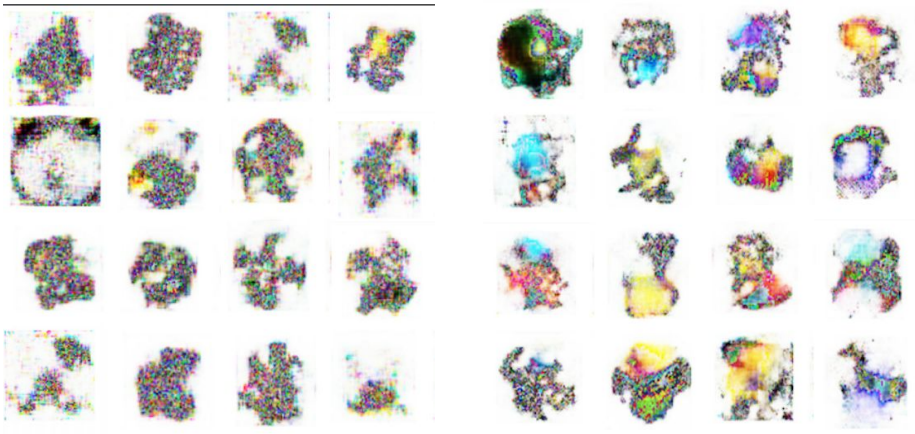

Figure 7: Early (left) and late (right) stages of GAN generated images on pokemon dataset

While these generated images definitely begin to take form, we cannot hope to generate fine grained images by training only on these reconstructed images, as the reconstruction process removes these fine details. Therefore, we propose the following approach to achieving stable learning on fine grain images. To stabilize the GAN's learning in the early stages of training, we begin by training the GAN on these reconstructed representations of our dataset in the early

phases of training. Then, by swapping these reconstructed images with our original ones, we believe we can achieve more stable learning and faster convergence in the long run.

To test this proposal, we trained our GAN two separate times from scratch. In the first run of GAN training, which we use as our control, we simply train the GAN on the original Pokemon data and record the losses from our generator and discriminator. Then, in the second run of from-scratch GAN training, we initially train on images preprocessed with a convolutional autoencoder, then switch to training the GAN on the original images mid training. By comparing the loss plots and generated images of the two approaches, we will be able to evaluate the effectiveness of this proposed technique to stabilize initial GAN learning. stabilizing the GAN learning in the early stages of training by initially preprocessing the images with a convolutional autoencoder. By training the GAN on these blurrier representations of our dataset in the early phases of training, we believe we can achieve more stable learning and faster convergence.

## 5 EVALUATION OF RESULTS

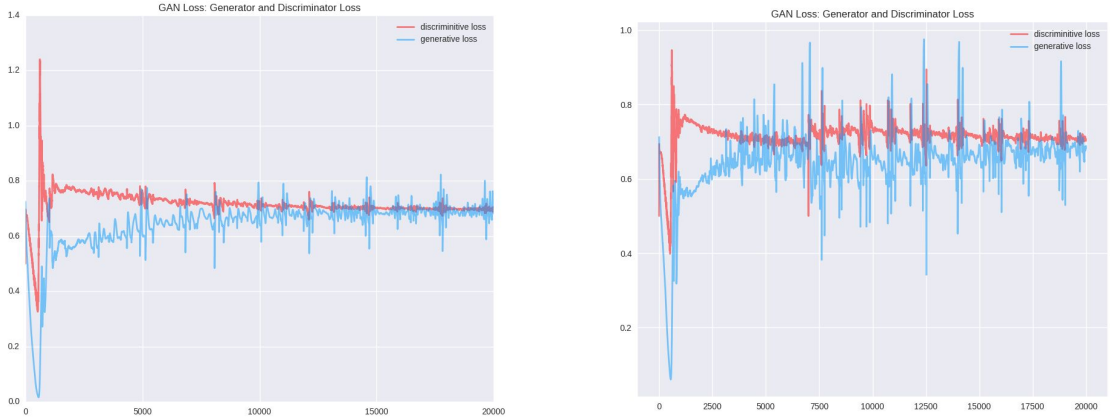


Figure 8: GAN loss when trained on original data only (left) vs. GAN loss when swapping data mid training

Although we were hopeful for this method to achieve stable learning of fine grain details in the long run, the opposite appeared to be true. In the beginning, the GAN trained on the reconstructed images did have more stable loss than the GAN trained on the original (as evidenced by the smaller initial waveform in the above plot on the left). However, when the swap for the original data came at 7000 epochs, this greatly destabilized the learning in our new-approach, resulting in inconsistent loss for many epochs to come. Meanwhile, when a GAN trained on the original images from the start reaches this point in its training, its learning rate has already stabilized and remains stable for additional epochs. There was no distinguishable

difference in generated images after 20,000 epochs of both approaches, so it is unclear how the new approach effects the images generated, if at all.

## 6 CONCLUSION

Generative Adversarial Networks have enjoyed much popularity in the last few years, especially for representation learning applications. However, these networks often have difficulty achieving stable training, as well as producing images that take concrete form. In this work, we have attempted to address both of these problems with the addition of an additional preprocessing technique. We found that when we used a Convolutional Autoencoder to encode and then reconstruct our images, our GAN's learning was more stable when trained on these reconstructed representations than it was when training on the original images. In addition, the generated images seemed to take better form when using this additional preprocessing step. However, this method will never be able to generate images with fine details, as this detail is lost in the mentioned preprocessing step. We then evaluated the effectiveness of first training a GAN on these preprocessed images and then swapping them out with the original images mid-training. Despite our hopes that this would allow the GAN to converge faster and be able to generate fine grained images, we discovered that overall this method in fact resulted in less stable learning in the long run as a result of the data swap. In the future, in order to generate detailed images like that of our original dataset, we may play around with increasing the depth of our discriminator. Although removing convolutional layers helped initially stabilize learning in the GAN, perhaps this simpler discriminator hurts the GAN's performance later on as the discriminator could not recognize the more complex patterns necessary in later phases of the training.

## REFERENCES


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

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. *NIPS*, 2014.

Kingma, Diederik P and Welling, Max. Auto-Encoding Variational Bayes. In *The 2nd International Conference on Learning Representations* (ICLR), 2013.

P. Baldi, ”Autoencoders, Unsupervised Learning, and Deep Architectures”. *Journal of Machine Learning Research-Proceedings Track*, vol. 27, pp. 37-50, 2012.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, 11(Dec):3371–3408*, 2010.