Unsupervised Feature Learning for Digital Art Using Info-GAN

Trevor Gordon
M.S Electrical Engineering
Columbia University
tjg2148@columbia.edu

Rakshith Kamath
M.S Electrical Engineering
Columbia University
rk3165@columbia.edu

***Abstract*—Much of the data we interact with on a day to day basis like images, video and text is represented in a high dimensional structure. There is a need to be able to generate some of this realistic high dimensional data. In many cases, much of this data can be represented in a lower dimensional structure. Generative Adversarial Networks (GANs) provide a mechanism for generating high dimensional target data from a lower dimensional latent space. Further, it can be advantageous to have control in the generative process like deciding between different categories of generated content. But, to do this, one typically needs to start with a significant amount of training labels. In this paper we discuss and implement ideas from [2] who aim to give more control to the generative process of GANs in an unsupervised manner by encouraging structure in the lower dimensional latent space.**

I. Introduction

Unsupervised learning can be defined as the general problem of extracting value from large amounts of unlabeled data. The purpose of representation learning, a prominent method for unsupervised learning, is to develop a representation that exposes significant semantic traits as easily decodable elements using unlabeled data. A method for learning such representations is likely to exist, and it will be beneficial for a variety of downstream tasks in reinforcement learning, such as classification, regression, visualization, and policy learning.

While unsupervised learning is problematic since the important downstream tasks are unknown at the time of training, a disentangled representation, which explicitly reflects the salient properties of a data instance, should be useful for the relevant but unknown tasks. For example, a usable disentangled representation for a collection of faces might allocate a different set of dimensions for each of the following attributes: facial expression, eye color, hairdo, presence or absence of spectacles, and the corresponding person's identification. Natural tasks that require knowledge of the data's prominent qualities, such as face identification and object recognition, can benefit from a disentangled representation.

Generative modeling drives a substantial portion of unsupervised learning research. It is motivated by the belief that the ability to synthesize, or "create," observed data entails some level of understanding, and it is hoped that a good generative model will learn a disentangled representation automatically, despite the fact that perfect generative models with arbitrarily bad representations are easy to construct. This can also be used as a method to discover and search high-dimensional spaces for data discovery and disentangling the information at such high dimensional space into a latent space of much smaller dimension.

The contributions of this paper are as follows. We

- Provide a literature review of early GAN papers
- Describe the key points of InfoGAN
- Implement InfoGAN and provide an analysis of it's ability to discover categories with MNIST
- Extend InfoGAN to produce images from a popular digital art dataset: CryptoPunks
- Give an analysis of emergent features we discover

II. Literature Review

A. *Original Generative Adversarial Network (GAN) paper*

Generative Adversarial Networks (GANs) were first proposed by [3] as a means to generate realistic output. While there were generative models that came before it, this method did not require markov chains or unrolled approximate inference networks during the training or generation process. This paper has inspired many important papers since. GANs are made up of a Generator (G) which takes in a low dimensional latent state noise vector and produces a higher dimensional target output like an image. The Discriminator (D) takes in real or generated images from the generator and is tasked with determining which samples are real or fake. The two are trained at the same time. After training the generator can be used to produce realistic content or the discriminator can be used as a classifier.

The loss function for GAN's are similar to that of binary cross entropy loss. This is because The Discriminator D acts as binary classifier which helps in distinguishing the fake and real samples that fed into it. Hence it tries to maximize the value of Discriminator. Here the real valued samples are set to 1 in the output of the classifier and 0 for the generated ones. In regards to the generator the we try to minimize the loss of that we get from images when we put it in the discriminator. Hence, we get a min max problem to solve with and this is solved here by pitting two neural networks G and D against each other so as to drive the generative modelling process.

The original network can be extended with different categories as proposed in [4]. Here, the Discriminator and Generator take in a label for the data. When considering the MNIST

dataset, this refers to the label of the digit. This is useful as the networks don't need to completely re-learn separate features for all categories in the dataset. A limitation of this approach is that the label for the dataset needs to be known in advance.

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

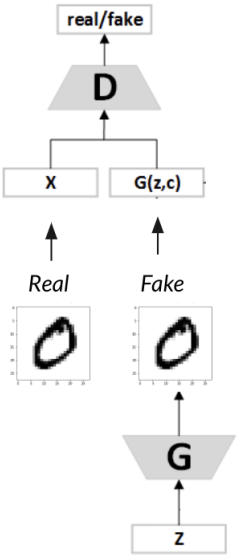

Figure 1. GAN architecture image adopted from [1]

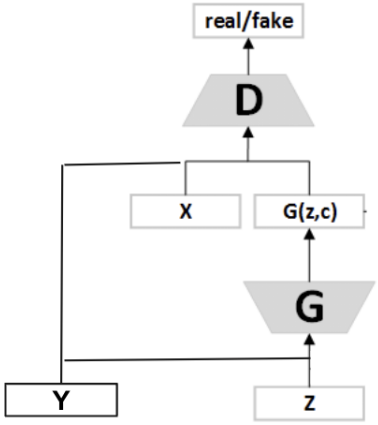

Figure 2. C-GAN architecture image adopted from [1]

B. *Info Generative Adversarial Network (InfoGAN)*

By default GANs are tasked with translating a random noise vector into high dimensional output that can fool the Discriminator. But, by default, there isn't any guaranteed regularity in how this is done. It's possible the GAN, when left to it's own devices, learns a chaotic translation between input noise and the output data. It's possible that some noise variables amount for huge variation in output and other noise variables barely change the output.

The InfoGAN builds upon the original GAN's by ensuring that some latent state variables have a noticible affects on the GAN output. Further, it is able to do this in an unsupervised manner. The authors propose specifying the latent vector is split into a pure noise part and a part that will be used for control. By maximizing the mutual information between a subset of the latent vector and the output image, the authors are able to ensure these subset noise variables learn meaningful features that can be controlled in the generative process.

This is achieved with the help of adding another network in the architecture called the Q network which helps us in calculating the loss.

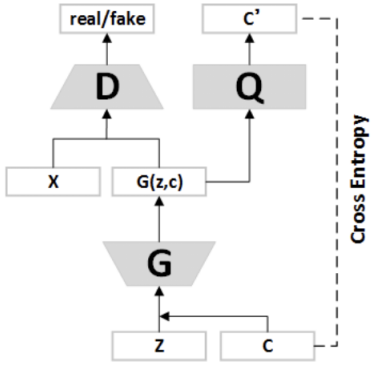

Figure 3. InfoGAN architecture image adopted from [1]

The way InfoGANs learn this variables in latent space is due to the construction of the loss function in an clever way and changing the original loss function as follows-

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c)) \quad (1)$$

where V(D,G) is the loss function of the original GAN and I(.;.) calculates the mutual information between 2 variables. Here, we try to maximize the mutual information due to the fact that its negative of the minimizer with respect to G. The mutual information loss ensures that changing this variable will have a noticeable change on the output image. This ensures that the change in c reflects on the G(z,c) as well and this ensures that the model learns features that are relevant and helps reducing the Generator loss. We want there to be a noticeable change in the output of the generator when we change the c.

But the issue that we face with the calculation of mutual information is that we have a posterior term in its calculation which cannot be calculated directly.

$$I(c;G(z,c)) = H(c) - H(c|G(z,c)) \quad (2)$$

This is mitigated by approximating the mutual information and calculating it using an auxiliary function that gives us

approximation for mutual information and gives us a good lower bound for the same.

$$I(c;G(z,c)) = \mathop{\mathbb{E}}_{x\sim G(z,c)}[\mathop{\mathbb{E}}_{c'\sim P(c|x)}[logP(c'|x)]] + H(c) \quad (3)$$

$$I(c;G(z,c)) = \mathop{\mathbb{E}}_{x\sim G(z,c)}[D_{KL}(P(.|x)||Q(.|x)) + \mathop{\mathbb{E}}_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c)$$

$$I(c;G(z,c)) \geq \mathop{\mathbb{E}}_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c) \quad (4)$$

Here, we see that the original posterior is replaced with a KL divergence term and a auxiliary function log Q(c'—x). This is obtained from the loss function that the Q network calculates hence was introduced in this architecture. This calculates the loss by reverse engineering the process where it tries to learn the latent variables c given G(z,c).

Further mathematically the lower bound for the mutual information is calculated as shown below-

$$\begin{aligned} L_I(G,Q) &= \mathop{\mathbb{E}}_{c\sim P(c),x\sim G(z,c)}[logQ(c|x)] + H(c) \\ &= \mathop{\mathbb{E}}_{x\sim G(z,c)}[\mathop{\mathbb{E}}_{c'\sim P(c|x)}[logQ(c'|x)]] + H(c) \\ &\leq I(c;G(z,c)) \end{aligned}$$

In practice, we realize this lower bound by approximating Q as a neural network that has mostly shared weights with the Discriminator. For the categorical variables, we add a categorical cross entropy loss on the Q's recovery of the categorical variable. For continuous variables we output a mean and a standard deviation for each continuous variable assuming diagonal covariance matrix. The loss is calculated as the negative log probability of the Gaussian distribution for the continuous variable input. As a result, the Q network has large loss when it only predicts wide distribution for the continuous variables and has a small loss when it predicts a narrow distribution with mean close to the true continuous variable. To summarize, the discriminator is updated based on it's ability to correctly detect the fake images. The Q network is updated according to it's ability to recover the special latent codes. The Generator is updated based on it's ability to fool the generator plus the ability of the Q network to recover the latent codes. The magic happens here as the Generator is incentivized to output images that encode features in them such that the Q network can recover the codes. The Generator is fed random permutations of all latent codes and all outputted images across this space must also look like realistic images. As a result, the Generator discovers real patterns in the data.

Given this knowledge and mathematical structure that enforces the model to learn in an unsupervised manner certain features, in the next section we try and discuss various experiments with different data sets and discuss how it tries to learn various features for various problems and also learn to harness the latent space.

III. Experiment Setup

The training process is as follows: The models are initialized with a specific input latent vector size and output image size. The input vector is specified by the length of pure noise, the number of categories in it's categorical part, and the number of continuous noise variables. We load the target data set. Before sampling from the Generator we need to generate random samples of the latent vector. For the pure noise part we sample from a zero mean Gaussian with unit variance. For the special categorical variable we uniformly sample over of all categorical possibilities and one hot encode these numbers. For the continuous variables we sample from a uniform distribution of [-1, 1]. To train the discriminator we first train it on a half batch of samples from the true data set with labels of 1 and then with a half batch of samples from the generator with label 0. The true images are taken as a random batch from the training set. We randomly generate a half batch of latent vectors and pass this through the generator network to get the generated images. Next, we sample another batch of latent space vectors and propagate this through the generator as well as the discriminator and Q network. The weights for the discriminator are updated based on it's ability to correctly identify the real images. The Q network including shared weights with D is updated based on it's ability to recreate the special categorical and continuous latent codes based solely on the output image. The generator is updated based on it's ability to trick the Discriminator as well as based on it's ability to provide to generate images such that the Q network can recover these special latent state variables.

A. MNIST digits data set

Our first set of experiments are done using the popular MNIST data set consisting of 28x28 gray scale images of handwritten digits. This data set was used in the original paper and is well suited for this problem for a number of reasons. First, there are inherent categories for each of the 10 digits. Next, differences in writing style result in many possible continuous variables to discover. For example, the thickness or rotation of digits.

We run our MNIST experiments with a single categorical variable containing 10 categories. We use 4 continuous variables and a length 62 noise vector.

The model architecture for the MNIST data set are mentioned in Table III-A.

Table I
Model Architecture for MNIST data set

Discriminator D and Recognition Network Q	Generator Network G
Input 28x28 Gray Image	Input 74 dimension variable
4x4 conv. 64 ,lReLU, stride 2	FC 1024. ,ReLU, Batchnorm
4x4 conv. 128 ,lReLU, stride 2, Batchnorm	FC 7x7x128, ReLU, Batchnorm
FC 1024 lReLU, Batchnorm	4x4 upconv 64, ReLU, Batchnorm, Stride 2
FC output layer for D FC 128-Batchnorm-lReLU-FC output for Q	4x4 upconv. 1 channel

B. CryptoPunk Dataset

In a second set of experiments we look to generate digital art from the popular CryptoPunk data set. These images are 32x32x3 images. These pixelized images are also well suited for this problem as there are a variable of face shapes, expressions, skin tons as well as interesting accessories like hats, beards, and cigarettes.

The model architecture for the CryptoPunk data set are mentioned in Table II.

Contrary to the well explored and rather simple data set for MNIST, the CrytoPunk dataset hasn't been extensively analyzed in this way. We look to explore the space of discovered continuous and categorical components.

We run our CryptoPunk experiments with a single categorical variable containing 10 categories. We use 4 continuous variables and a length 62 noise vector.

Table II
MODEL ARCHITECTURE FOR CRYPTOPUNK DATASET

Discriminator D and Recognition Network Q	Generator Network G
Input 32x32 Gray Image	Input 74 dimension variable
4x4 conv. 64 ,lReLU, stride 2	FC 3x6x6x128. ,ReLU
4x4 conv. 128 ,lReLU, stride 2, Batchnorm	FC 6x6x128. ,ReLU
FC 1024 lReLU, Batchnorm	FC 6x6x128. ,ReLU,Batchnorm
FC output layer for D FC 128-Batchnorm-lReLU-FC output for Q	4x4 conv. 64 ,ReLU, stride 2
	4x4 conv. 3 ,tanh, stride 2

C. Vector Addition

If a disentangled representation is learned in the the latent space, it is possible to perform useful mixing and matching that isn't possible in image space. One might wish to take the hair color from one individual with with short hair and give this hair color to another individual. One might think that you can simply subtract the images to isolate the hair color and then add this to the other image. Unfortunately this isn't possible in image space as the hair pixels would need to line up exactly to do this with distorting the image. But, this is possible in the latent space! If we are able to learn a representation specifically for hair color, we will be able to isolate features in vector space and add them. When we recover the image in image space we will find the features have been added without noticeable pixelation.

IV. RESULTS AND DISCUSSION

A. Experiment 1: MNIST Digit Dataset

We were able to successfully reproduce the results from the original paper. An example of generated images are shown in Figure IV-A. All image are from the same pure noise vector. The rows show all permutations of the categorical variable. The columns show a span across the continuous variable. It can be seen that in the categorical rows we have recovered most of the true numerical categories. Along the columns we have discovered the ability to draw digits with varying levels of thickness. The original authors uncovered the same ability. These results will collected with a ratio of 0.4 to 1 for the continuous variable loss compared to the categorical loss. We found experimentally that this ratio determined what control the categorical and continuous variable have on the output. If the values are close together, the continuous variable will start changing aspects of the output that we would rather the categorical variable change. For example, the continuous variable will start changing from one number to the next.

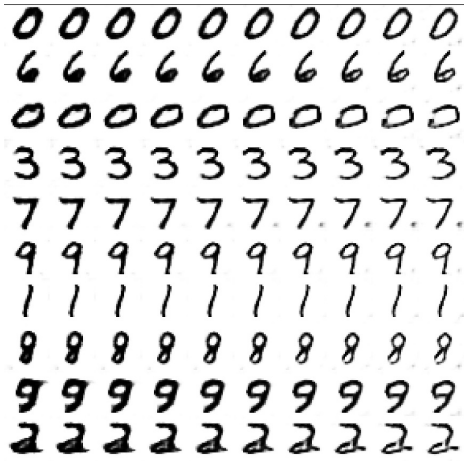

Figure 4. Example output from our implementation of InfoGAN. All image are from the same pure noise vector. The rows show all permutations of the categorical variable. The columns show a span across the same range as the continuous control latent variable

1) Emergent Categories

Furthermore, we performed some analysis on the the discovered categories to determine how well they matched with the original. To do this, we started by generating 500 fake images and together with 500 real images. We reduced the high dimensional images down to 2 dimensions using T-SNE as described in [5]. We plotted the data with the true categories for the real data and the unordered discovered categories. The result is shown in Figure IV-A1. It can be seen that the clusters of our discovered categories overlap well the original categories.

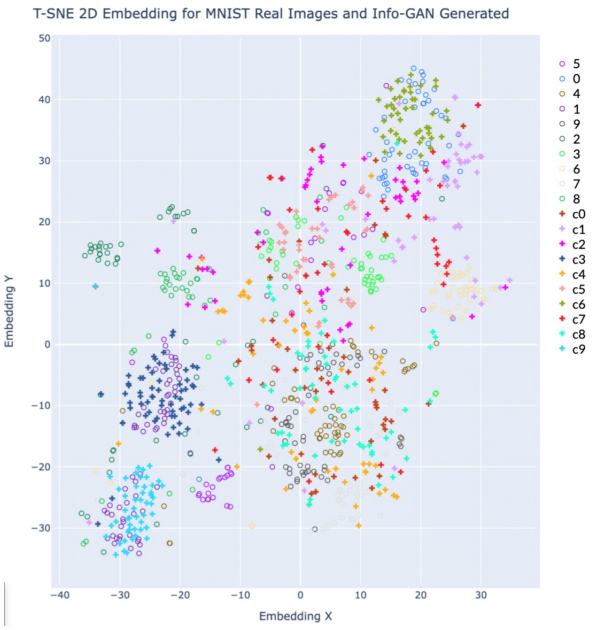

Figure 5. Real and fake images compressed to 2D using T-SNE. MNIST plotted with real labels 0-9 and circles. GAN images plotted with discovered categories c0-c9 as +'s

B. Experiment 2: CryptoPunk Dataset

For the CryptoPunk data set we were able to generate convincing output images that look like images from the original data set! An example of these output images is shown in Figure 6. While we found we were able to generate realistic images, we recognize some potential drawbacks. We found that images in our generated data set mix and match many features from the training data set like hair, eye wear, face shape and other accessories.

Furthermore, we found that our model is able to discover natural categories in the CryptoPunk data. An example of categories is sh9. It can be seen that the same emergent categories are present in each of the four blocks. Starting from the top row as row 1, there is dark bushy hair in row 1, dark-framed blue eye-wear in row 6, pink and orange head tops in row 8 and blue head wear in row 9. In order of top left, top right, bottom left, bottom right, we see emergent continuous variable control of cigarettes, Mohawk hair, smoking pipe, and head shape. We note that these discovered categories are not stable. With continued training the model continuous to find other viable options for categories. Further, although this is an unsupervised machine learning problem, it is challenging to determine how many categories should be chosen without knowing the data set. We tried running the model with 20 categories as shown in Figure 10 but were not able to learn meaningful categories. Furthermore, as more categories are added one must modifying the relative weights of the error to adjust for the change in difficulty with more or less categories.

1) Vector Addition

We performed vector addition by taking similar images with and without a cigarette. We subtracted to isolation the cigarette in latent space. Then, we took separate images that did not contain a cigarette and adding our latent isolation of the cigarette. We found that when the latent representation is converted back to the high dimensional image is that the accessory gets added in the image. This is shown in Figure 7.

Similar results can be seen in Figure 8 where we are able to add a pipe to the mouth of the person. Interesting thing to note is that in the resultant image we also get to see the puff of smoke along with the pipe. This gives us an example where the points which are further apart in the high dimensional space are closely connected and the pipe and puff are seen together in the latent space which is learn by the model pretty well.

Figure 7. Vector Addition shown for NFTs by adding cigarette

Figure 8. Vector Addition shown for NFTs by adding Pipe

C. Analysis Of Results

The CryptoPunk data set contains 32x32x3 pixels or 3072 dimensions. We found that we can generate a wide variety of images with 62 noise variables, 1 categorical variable and 4 continuous variables or 67 dimensions. This is an important realization for generative modeling as we don't need to work in such a high dimensional space. Even more importantly, humans don't think of differences in images on a pixel by pixel basis. We consider differences in images by higher level features. We found that InfoGAN provides a powerful way of detecting and giving control of this inherent low dimensional structure.

One drawback of this approach is that it requires significant manual tuning of some hyper parameters. Specifically, we found that changing the number of variables as well as the relative loss between the categorical, continuous, and normal GAN loss drastically changed our results. It is possible that

Figure 6. Images from the true CryptoPunk data set on the left and randomly generated images from our InfoGAN on the right.

an iterative grid search could be used to find these hyper parameters but this would take a very long time due to the long training time required for GANs.

V. Conclusion and Future Work

In this study, we were able to learn disentangled representations in a completely unsupervised manner for high-dimensional data.This has the ability to be used as a high-dimensional data discovery tool. Reconstruction of image based on latent space information depends on the size of the latent space we try to train on and as well as relevant features that can be learnt for a given problem. Difficulty in learning and getting the exact manifold for the Latent space we learn due to its unsupervised nature and also the complexity of the manifold.

References

[1] InfoGAN. https://www.researchgate.net/figure/The-architectures-of-InfoGAN-and-SCGAN-InfoGAN-attempts-to-separate-the-condition-which_fig1_327949545, 2013. Online; accessed 8 May 2022.

[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[4] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[5] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

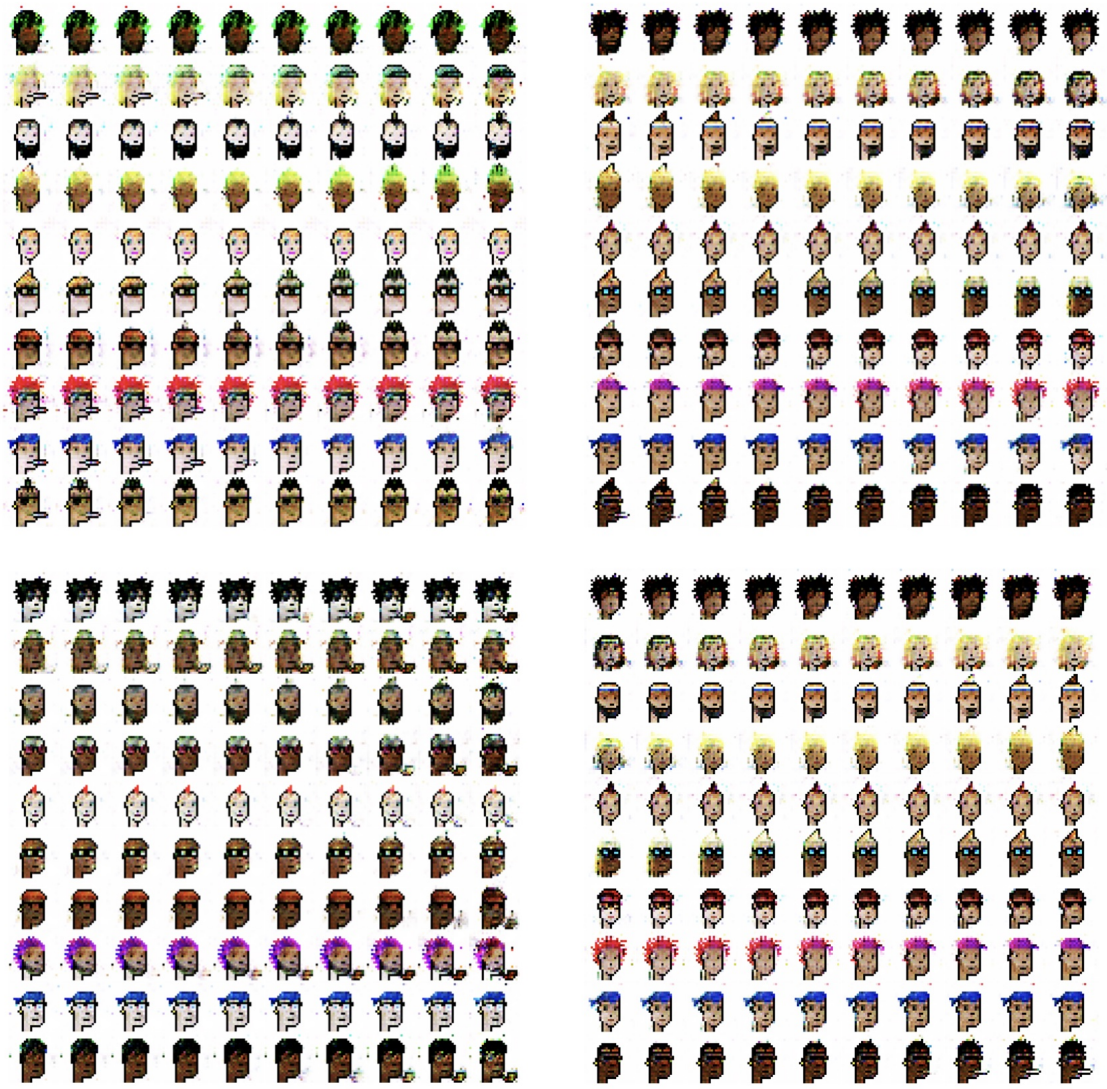

Figure 9. Emergent categorical and continuous variable control in InfoGAN generated images of CryptoPunk data set. Each block of images corresponds with rows showing image for each of the 10 categories and columns spanning from -1.5 to 1.5 for one continuous variable. Each of the four blocks corresponds with one continuous variable. It can be seen that the same emergent categories are present in each of the four blocks. Starting from the top row as row 1, there is dark bushy hair in row 1, dark-framed blue eye-wear in row 6, pink and orange head tops in row 8 and blue head wear in row 9. In order of top left, top right, bottom left, bottom right, we see emergent continuous variable control of cigarettes, Mohawk hair, smoking pipe, and head shape

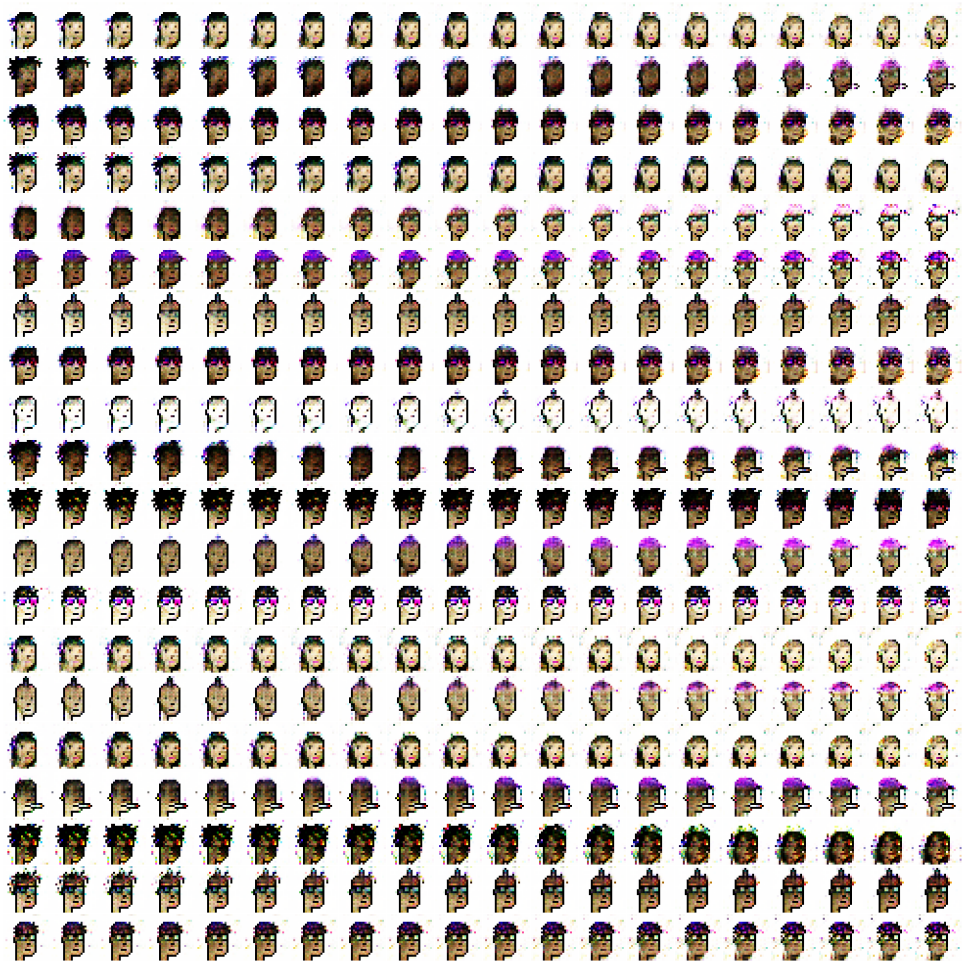

Figure 10. Trying to learn 20 categories with the CryptoPunk data set.