Generative Adversarial Networks for Multiclass Image Generation

Dixee Kimball, Chenduo Huang, and Karen Yang

Stanford University, CS221 Fall 2016

Abstract

Recent research endeavors on realistic images synthesis have achieved success by adopting Generative Adversarial Networks (GAN) [1], a framework corresponding to a minimax two-player game between a generative network (G) and a discriminative network (D). This framework has shown to work well in producing images from a single category, in which D acts as a binary classifier discriminating between real and fake images. In this project, we explored GAN's potential in learning a set of high dimensional features to represent multiple image categories. We applied Auxiliary Classifier GAN (ACGAN) that conditioned on image classes to generate images from three datasets: MNIST(10 classes), CIFAR-10(10 classes) and Chars74K English Handwritten(62 classes). The experiments demonstrate that the our model can generate elegant images with clear distinctions across multiple classes.

1 Introduction

1.1 Related Work

Understanding and modeling the images manifold has been a longstanding computer vision problem. While searching for similar image is fairly simple (as in our baseline model), generating an image from scratch with some actual "creativity" requires the algorithms learning the complicated features embodied in image data distribution. It is only in the last two years, that there has been great advancement in image generation without much approximation and preconditioning, fueled by the development of generative adversarial networks, first proposed by Googlefellow et al.[1] and many of its variants.

The learning process of the Generative Adversarial Networks(GAN) is to train two neural networks, a Generator G and Discriminator D, in opposition to one another . The target of G is to learn the distribution pg over data x. By adopting the reparametrization trick, G starts from sampling input variables z from a uniform distribution $pz(z)$, then maps the input latent variables z to data space $G(z;\theta_G)$ through a differentiable network. On the other hand, D is a classifier $D(x;\theta_D)$ that aims to recognize whether an image is real (from training image data) or is fake (generated from generator G).

The minimax objective for original GAN model can be defined as follows:

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

However, several researches has shown [2] that the generated images become weirder and contains multiple features of different classes when one GAN model tries to learn features of multiple classes, which is not reasonable. The discriminator as a classifier will be confused by multiple features in the combined classes. We are thus motivated to propose a multi-class GAN that can guide the model to

learn the features of the multiple classes conditioned on their class labels.

Most previous deep generative models[3][2][4] that are based on the DC-GAN framework (Zhu et al. [3])) are category-specific (models trained across multiple categories show large reconstruction error)

There are certainly many challenges in training of GAN and its architecture needs to be carefully designed.[1][3][5]. In our ACGAN implementation, the multi-class implementation is even more difficult to train because it requires both G and D handle images across different classes, often containing mixed information and features. With limited computing resources and time, our ability to search for the optimal hyperparameters are limited and our implementation of the ACGAN architecture are mainly based on the existing literature on how to improve the general GAN training }[5]. We aim to exemplify the performances of our ACGAN model only on relatively small image resolution with moderate depth of convolution networks. Another interesting yet challenging task we have is to interpret the latent variables' impact on images (ideally, can be used to manipulate images by inverting their transform function), which might not be consistently intuitive to human observers.

1.2 Task Definition

In this work, we present two main objectives:

- Training a GAN model with a pair of generator network and multi-class discriminator. They share a competitive mini-max objective over classification of source (real/fake) and a collaborative objective to learn the distribution of multiple categories of images.
- Learning latent variables that correspond to semantically meaningful features across different categories of images.

2 Approach

2.1 Baseline Model

Since the task is to produce an image (found from database or generated), we employ a simple baseline model where we store a database consisting of images of our intended classes. Suppose the user wants to generate an image from a given sketch, for example, a photo of handwritten digit one on a piece of white paper. We then run a K Nearest Neighbour search, and simply output the average of top K image within the class of digit one that are "closest" to our sketch. We define the distance of two images as the Euclidean distance between two image vectors, where the values in each vector corresponds directly to the pixel values in each image.

This baseline model is not supposed to have any generative power since all it does is to naively select the closest images using pixel-wise comparison and output their average. Thus, it requires a lot of image information directly from the user input and is expected to see many artifacts when the user sketch is very different from the training dataset (e.g. variance in lighting, position, rotation, etc).

2.2 Oracle

In this project, since the evaluation of the quality of output images is subjective, we will let ourselves be the human discriminator (oracle) and make statements on whether the generated images look correct and elegant to us.

2.3 ACGAN Model

The Auxiliary Classifier GAN (ACGAN) is a variant of the GAN model where the discriminator produces both a probability distribution over sources and a probability distribution over class labels,

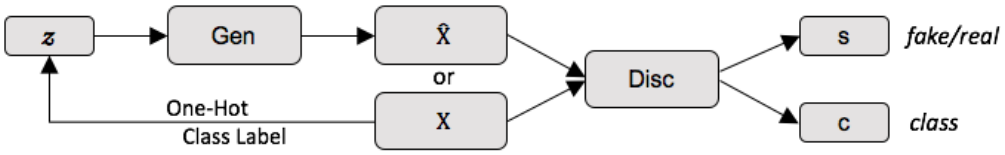


Figure 1: ACGAN Architecture

as shown in figure 1.

We borrow idea from a newly implemented Auxiliary Classifier GAN (ACGAN) framework [6]. The AC-GAN generator takes in a one hot label of the intended class concatenated with randomly generated noise. The discriminator not only attempts to discriminate generated images from real images, but also predicts the class of an input images regardless if it is generated or real using an categorical auxiliary classifier. In addition, a continuous auxiliary classifier can be added to give more insights to what some latent variables represent and how we can potentially manipulate the generated images.

The non-zero sum minimax objectives for our ACGAN model expressed in the form of softmax functions (final layer for classification):

$$L_S = E[\log P(S = real|X_{real})] + E[\log P(S = fake|X_{fake})] \tag{2}$$

$$L_C = E[\log P(C = c|X_{real})] + E[\log P(C = c|X_{fake})] \tag{3}$$

Where discriminator D is trained to maximize $L_S + L_C$; generator G is trained to maximize $L_C - L_S$.

Notice that the original GAN objective is equivalent to let discriminator D trained to maximize L_S and generator G to minimize L_S, which is a zero-sum game. In our model, both D and G share a competitive mini-max objective over L_S(classification of source) and a collaborative objective over L_C (distribution of multiple categories of images), as desired in our Objective 1.

In the ACGAN framework, the included continuous auxiliary classifier helps our model learn the meanings of input latent variables. Using this model, the user does not need to supply any sketches to indicate what category of images do they want. The user can simply specify the class of image they want to generate, and by manipulating the latent variables they can generate the image with certain characteristics (such as color, shape features). This is done by forcing the discriminator to perform additional task of predicting a subset of the latent variables, called z_{cont}, from which the samples are generated.

2.4 Datasets

We evaluated our proposed ACGAN models with 3 experiments on MNIST[7],CIFAR-10[8] and Chars74K English Handwritten[9] datasets.The details of these datasets are shown in Table1.

For CIFAR-10 dataset, the images are distorted into various different angles and positions to increase the number of samples. Both the CIFAR-10 and the EnglishHnd images are reshaped into a smaller image size as specified in Table1 for the purpose of faster training. Images across different classes are randomly shuffled before fed into each training batch.

2.5 Implementation

Broadly, in our ACGAN implementation, the generator G is a series of deconvolution layers with Relu nonlinearity that transform a noise vector z and a one-hot class label c into an image. The

Dataset	#Categories	#Samples	Image Size (post-processing)
MNIST	10	70,000	$28 \times 28 \times 1$
CIFAR-10	10	50,000	$24 \times 24 \times 3$
Chars74K EnglishHnd	62	3,410	$32 \times 32 \times 1$

Table 1: Statistics of datasets.

final output layer of G is Sigmoid for grayscale images (MNIST and EnglishHnd) and Tanh for RGB images (CIFAR-10). The discriminator D is a convolutional neural network with a Leaky ReLU nonlinearity. The final output layer produces the predicted class c is Sigmoid for grayscale images, and Softmax function for RGB images since it receives a 3-dim vector, and the predicted continuous latent variables z_{cont} is produced by a different output layer using Sigmoid function.

The Loss function is binary cross entropy loss function for real/fake classification,

The detailed trainning process and hyperparameters for each of our dataset are listed in Appendix A. The implementation of our proposed models are modified based on the a similar TensorFlow implementation of ACGAN [10] by using their library Sugartensor, a simplified wrapper for Tensorflow.

Our training and generation codes are available on this repository:`https://github.com/karenyang/MultiGAN`

The experiments using GAN models are run on Amazon EC2 g2.2xlarge instances with an GRID K520 GPU.

3 Results

Our analysis of the proposed ACGAN model consists of three sections with results from three different datasets. In the first section we also demonstrates the comparison between the baseline model and our ACGAN model on MNIST dataset. In the second section we also presents and explain the detailed learning curve for the CIFAR-10 dataset training, the most difficult one to train. In the third section, we try out ACGAN model with Chars74K English Handwritten dataset with up to 62 classes, and the distinguishable images output for each class indicate that ACGAN has the potential representation power over a large number of classes.

3.1 Evaluation and Baseline Comparison on MNIST

The detailed deep learning structure and hyperparameters for training MNIST Dataset are listed in Table 2 in Appendix A.

In Figure 2, we present the results after only 15 epoches of trainning, which is about 2.5K iterations. We as human oracle think the generator begins to generate similar images at 2.5K iterations that can fool the biased discriminator and human observers.

We then compare the performance of our ACGAN momdel and the baseline model by conditioning on the class number 1 to generate images of digit 1, as shown in Figure 3.

In the baseline model, we gave a user input sketch (in this case, an image of 1 from test set) and a class number 1. The program run the 10 nearest neighbour search within the class 1 images and output the average of them. The resulting image is blurry, not very realistic and by its nature, not 'creative'.

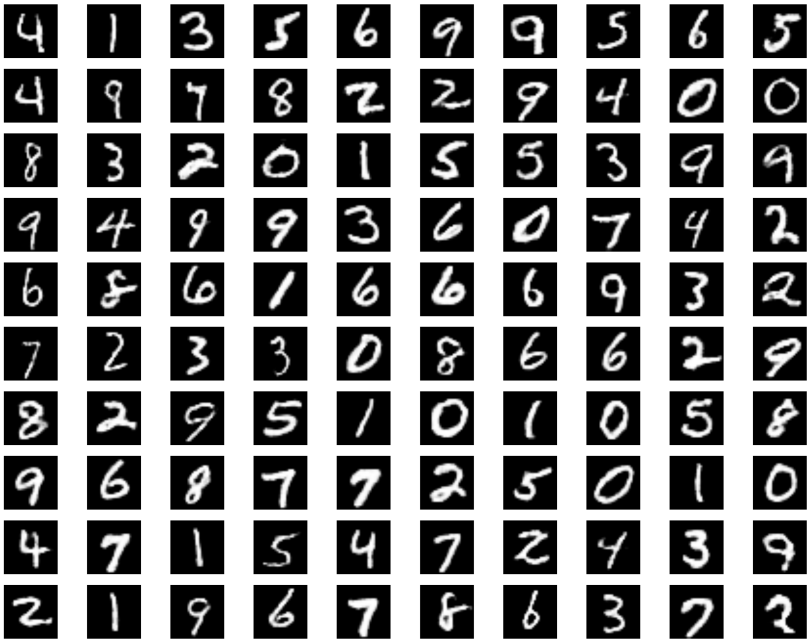


Figure 2: ACGAN generated random numbers from MNIST at Iteration 2.5K.

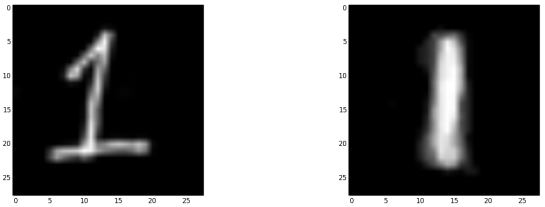
In the ACGAN model, by forcing the generator to output images with the first two z_{cont} set to linear interpolatedalues, the model generated images that clearly belong to class 1 and has certain diversity. We observe that these two latent variables correspond to the angle of the digit (from left to right) and the thickness (from top to bottom). The diversity of output images indicates that our model did learn the features of the images rather than simply memorizing what kind of output can best 'fool' the discriminator.

3.2 Evaluation and Training Process on CIFAR-10

The detailed deep learning structure and hyperparameters for training CIFAR-10 Dataset are listed in Table 3 in Appendix. A

We present the learning curve of the learning curves to illustrate the training process of our ACGAN model on CIFAR-10 dataset in Fugre 4. The learning curve shows that we have reached the desired training objective. In particular:

- The discriminator D is good at classifying the input source (fake/real) at first and the loss goes down, but later it went up as the generator G learned how to successfully fool the discriminator. As expected.
- The generator has a high initial L_S loss since it started from blank and can be easily identified by the D to be producing fake images, at about 80K iterations it learned how to fool the D, and after 120K it converges to a value of 0.5 . At the same time the D's loss rises. As expected.
- The class classification loss, which is shared by both G and D, has lowered in the beginning with no rebounce. The value remains at a constant values of about 1.6.



(a) User Sketch Input for Baseline Model

(b) Output from Baseline Model

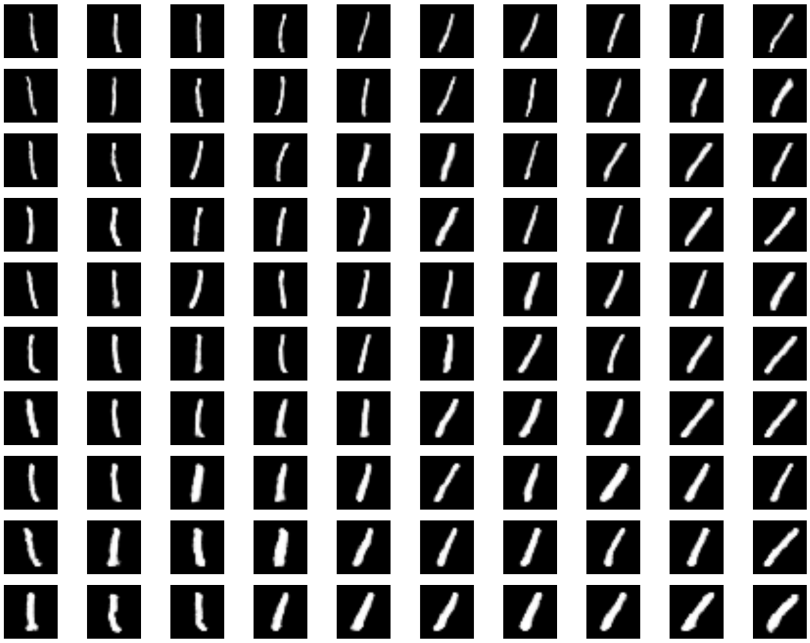


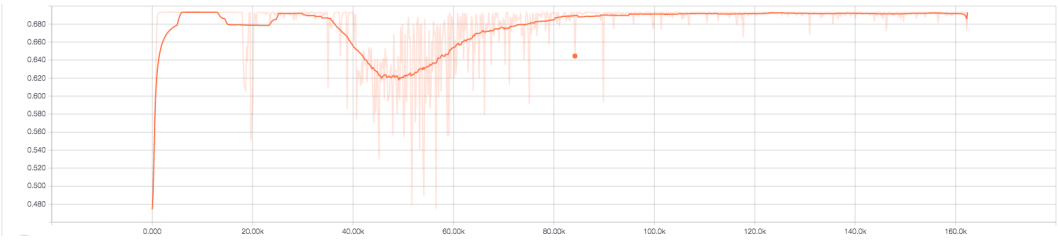
Figure 3: Top: Baseline model output on class 1. Given a user input sketch (in this case, an image of 1 from test set) and a class number the program run the 10 nearest neighbour search for the class and output their average. Bottom: ACGAN model output on class 1. User only needs to input desired class label for diverse generated images form ACGAN model, in which the thickness and angle feature can be manipulated by setting the z_{cont}.

- The continuous latent variable, which is also shared by both G and D has lowered in the beginning and approached 0.

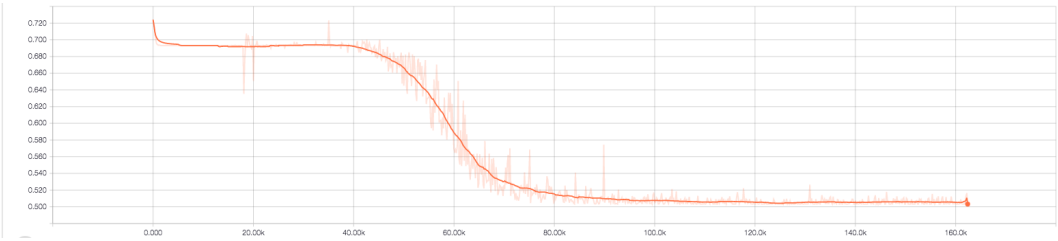
In Figure 5 we show the evolution of our ACGAN model's generated images. The model is able to generate images that is discernable starting at 200 epoches. At 250 epoch, some of the classes of images look good, such as the bird, dog and horse classes, while the other do not look realistic yet.

The final output is not as realistic as we human oracle wish them to be. There are several potential reasons:

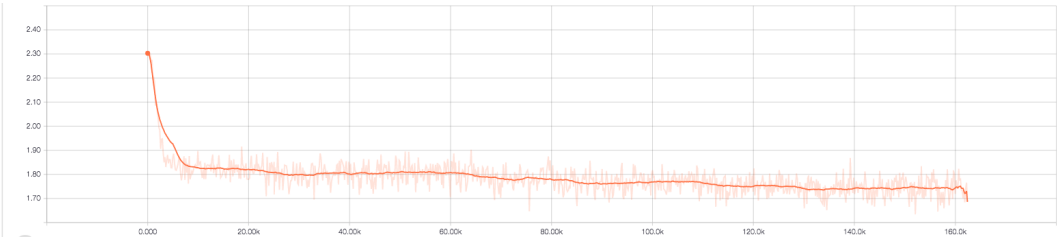
1. This shows the limitations of the generative power of our chosen ACGAN model for this particular small image dataset. The Cifar-10 might be particular hard to train becuase it is very tiny and information from the raw image data has high variance. we could have include



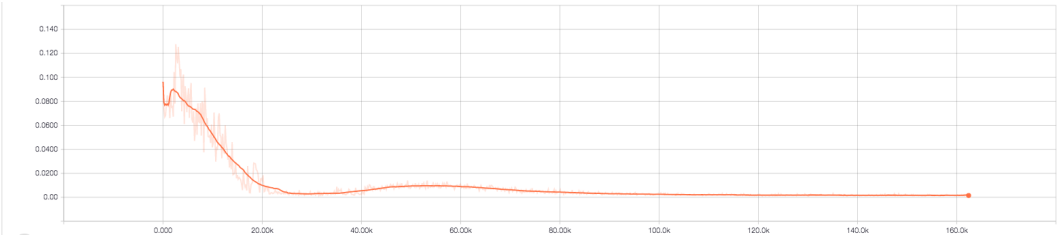
(a) D's Binary Cross Entropy Loss on fake/real classification



(b) G's Binary Cross Entropy Loss on whether fooled the D



(c) Class Prediction: Cross Entropy Loss on class classification.



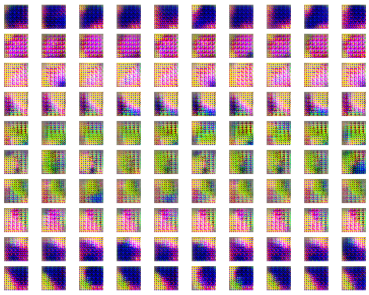
(d) Continuous Latent Variable Prediction: Mean Square Error Loss on matching the continuous variables

Figure 4: The learning curve of four loss functions. (a): The discriminator D's L_S is goes down and back up. As expected. (b): The generator's L_S is high initially and goes down as it learns the image data distribution. At the same time the D's loss rises. As expected. (c): The class classification loss, which is shared by both G and D, has lowered in the beginning with no rebounce. The value remains at a constant values of about 1.6. (d): The continuous latent variable, which is also shared by both G and D has lowered in the beginning and approached 0.

more deconvolutional layers or transposed convolution layers in the Generator. Furthermore, we could experiment on different optimzers, learning rate and batch sizes.

2. Not enough training iterations.Due to the limited GPU memory on the AWS server and we could not train further even we believe with more iterations it might get even better result.

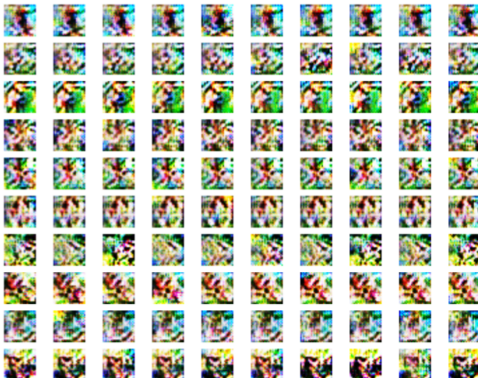
3. Not enought latent variable dimemsion (currently set to 100, in which 10 are used by one-hot class label encoding). With the 90 latent variables, it is difficult to capture the features of the images, especially when they are RGB images and each class is very different from the others.



(a) Epoch 4

(b) Epoch 9

(c) Epoch 100



(d) Epoch 200

(e) Epoch 250

Figure 5: The Training Process of ACGAN on CIFAR-10 Dataset

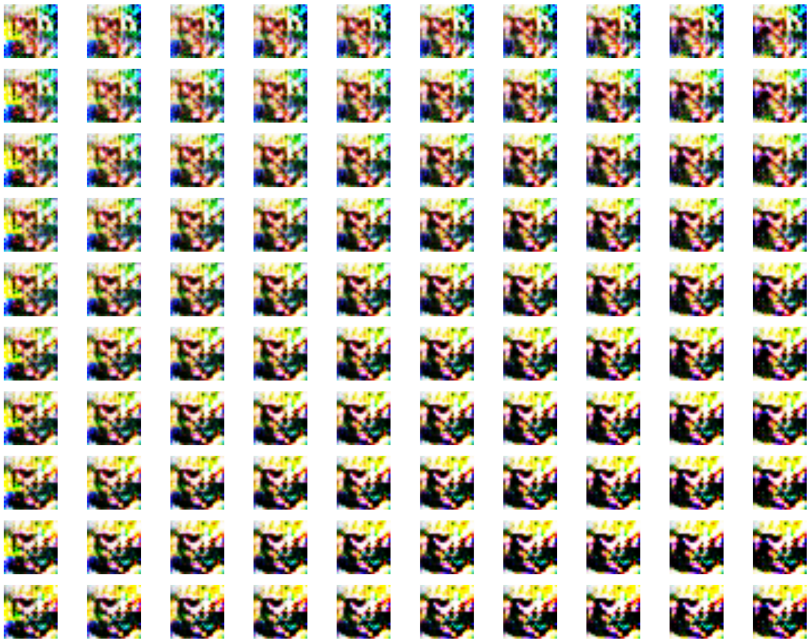


Figure 6: Epoch 250: Linear Interpolation on first 2 Z_{cont} with Class: Dog

3.3 Evaluation on Chars74K English Handwritten

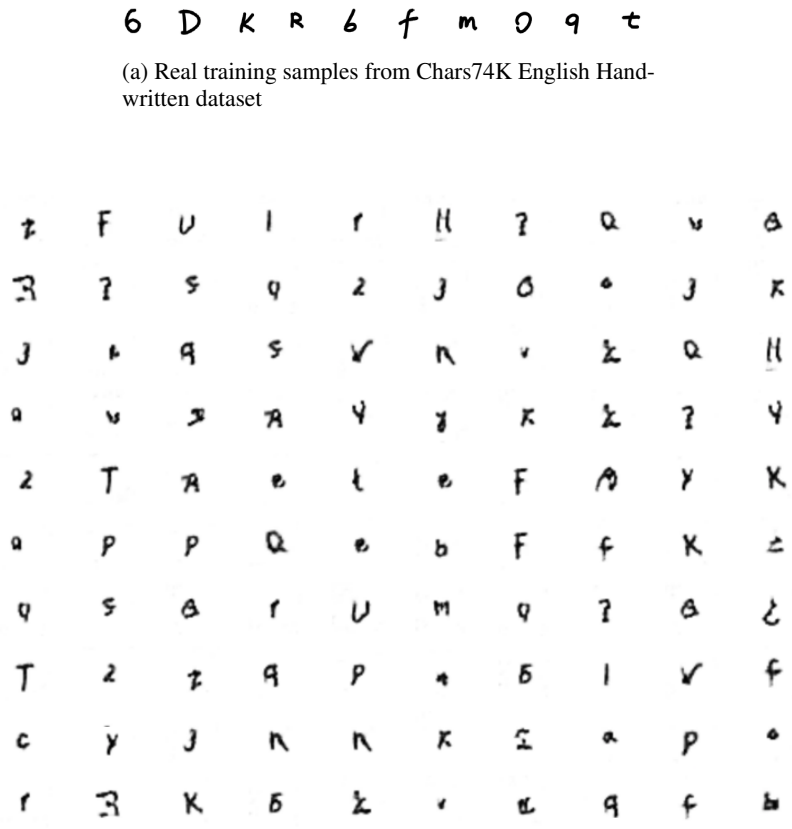
The detailed deep learning structure and hyperparameters for training Chars74k English Handwritten Dataset are listed in Table 4 in Appendix A.

In Figure 7, we experiment with a larger number of classes (# Class = 62) to test the potential of our model in capture the representation of images across many different categories. Within 80K iterations, the results have already shown a discernable pattern across digits, upperclass letters and lowerclass letters. It looks, from a human oracle perspective, quict similar compared to original images, even when all trainning images are scaled down to 32×32, 110 of original size. We believe with higher resolution, which certainly requires more computation power, the results will be more realistic. We think this demonstrates the potential of ACGAN in the use of trainning datasets with even larger class number.

4 Conclusions and Future Work

In this work we demonstrated that ACGANs can learn the representation and generate elegant and coherent image samples conditioned on one user-specified class out of multiple (10 and 62) distinctive image classes. We have explored interpolating the image output feature by manipulating a small subset of continuous latent variables. Our result reveals the generative potential of ACGAN model in a multi-class image generator AI that helps a user express and create.

While the results seems promising, this project is still in the exploring stage in the grand scheme.We can take the current progress further and try different hyperparameter search within the ACGAN framework. Given enough training time and computing resources, we can add more convolution-allayers and add batch normalization differently and try to improve our results, specifically on



(a) Real training samples from Chars74K English Handwritten dataset

(b) ACGAN generated random class output at 80K iteration

Figure 7: Chars74K English Handwritten Results

the CIFAR10 dataset. Once the generator produces more promising results, we can then vary the number of latent variables being targeted in our continuous auxiliary classifier, and have a better understanding about their effect by visualizing them.

Another objective that we initially proposed but did not get to finish within the scope of this project is successfully integrating the ACGAN model into the iGAN[2] framework, proposed by Zhu et al. , to produce a real-time demo. In this case, learning of latent variable is not required. But instead we can take the user's sketch from the interface, solve the HogNet optimization problem specified in the iGAN paper, and generate the image in the same fashion by reversing the transform function from $x = z \rightarrow G(z)$ to $z = x \rightarrow G^{-1}(z)$ and run stochastic gradient descent to get as close as the user sketch defined by the HogNet functions.

Acknowledgements

We would like to thank Prof. Percy Liang, Nishith Khandwala and Tim Shi for their helpful advices and kind support throughout the project and the course.

References

[1] 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.

[2] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

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

[4] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[5] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[6] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

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

[8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset, 2014.

[9] T de Campos. The chars74k dataset, 2009.

[10] Namju Kim. A tensorflow implementation of google's AC-GAN (Auxiliary Classifier GAN). `https://github.com/buriburisuri/ac-gan`, 2016. [Online; accessed 19-Nov-2016].

A Hyperparameters

Training on MNIST Dataset					
Operation	Kernel	Strides	Feature Maps	BN	Nonlinearity
$G(z)$ - $50 \times 1 \times 1$ input					
Linear	N/A	N/A	128	Y	ReLU
Transposed Convolution	4×4	2×2	64	Y	ReLU
Transposed Convolution	4×4	2×2	1	N	Sigmoid
$D(x)$ - $128 \times 1 \times 1$ input					
Convolution	4×4	1×1	64	N	Leaky ReLU
Convolution	4×4	1×1	128	N	Leaky ReLU
Linear	N/A	N/A	11	N	Sigmoid
BatchSize=32;numClass=10; numZcont=2; LReLUSlope=0.2; LR=$10^{-2}(G)/10^{-3}(D)$					

Table 2: Hyperparameters for MNIST Dataset

Training on CIFAR-10 Dataset					
Operation	Kernel	Strides	Feature Maps	BN	Nonlinearity
$G(z)$ - $110 \times 1 \times 1$ input					
Linear	N/A	N/A	128	Y	ReLU
Transposed Convolution	4×4	2×2	64	Y	ReLU
Transposed Convolution	4×4	2×2	3	N	Tanh
$D(x)$ - $128 \times 3 \times 3$ input					
Convolution	4×4	1×1	16	N	Leaky ReLU
Convolution	4×4	1×1	32	N	Leaky ReLU
Convolution	4×4	1×1	64	N	Leaky ReLU
Convolution	4×4	1×1	128	N	Leaky ReLU
Convolution	4×4	1×1	256	N	Leaky ReLU
Linear	N/A	N/A	11	N	Sigmoid, Softmax
BatchSize=100; numClass=10; numZcont=4; LReLUSlope=0.2; LR=$10^{-2}(G)/10^{-3}(D)$					

Table 3: Hyperparameters for CIFAR-10 Dataset

Training on Chars74K EnglishHnd Dataset					
Operation	Kernel	Strides	Feature Maps	BN	Nonlinearity
$G(z)$ - $150 \times 1 \times 1$ input					
Linear	N/A	N/A	128	Y	ReLU
Transposed Convolution	5×5	2×2	64	Y	ReLU
Transposed Convolution	5×5	2×2	1	N	Sigmoid
$D(x)$ - $128 \times 1 \times 1$ input					
Convolution	4×4	1×1	32	Y	Leaky ReLU
Convolution	4×4	1×1	64	Y	Leaky ReLU
Convolution	4×4	1×1	128	Y	Leaky ReLU
Linear	N/A	N/A	11	N	Sigmoid
BatchSize=32; numClass=62; numZcont=2; LReLUSlope=0.2; LR=$5^{-3}(G)/5^{-4}(D)$					

Table 4: Hyperparameters for Chars74K EnglishHnd Dataset

B More Baseline Model Results