

Auto Doodling Using Generative Modeling

CS230 Final Project Report

1st Yiting Zhang
KLA Corp.
Milpitas, CA 95035, USA
yizhan@stanford.edu

2nd Tuo Shi
KLA Corp.
Milpitas, CA 95035, USA
tshi86@stanford.edu

3rd Xinhua Ren
Google
Mountain View, CA 94043, USA
xinhua19@stanford.edu

Abstract—Two generative adversarial networks were applied to generate human sketches based on the Quick, Draw! dataset. With the additional loss function, Auxiliary Classifier GAN (AC-GAN) was found to have better performance in generating high quality fake images, while conditional GAN (C-GAN)’s outputs retained a certain level of blurry and discontinuity. With learned knowledge through the course, sampling truncation trick was applied to generate samples with high fidelity. Fréchet inception distance (FID) and kernel inception distance (KID) were used as evaluation metrics to compare all trained models and their scores in general match with human intuition.

Keywords—GAN, image generative model, FID, KID, Truncation Trick

I. INTRODUCTION

Generative Adversarial Networks (GANs) are attracting more attention nowadays. Researchers have made substantial progress on both the theory and applications. A large number of GAN variants have been introduced for artificially generating high-quality images, videos and audio.

Learning sketch that shows interesting features of something observed is an essential training in helping young children perform fundamental visual analysis of everyday spaces and improve hand-eye coordination.[1] However, not every parent can teach sketch or provide various sketch examples on the same object. Thus, we want to utilize the neural network’s generative modeling ability to create an auto doodling bot which can generate numerous sketches based on specified categories and thus help in early education.

In order to generate a sketch based on its category, a conditional GAN model is required. In this paper, two classic GAN models which generate a fake sample with at a specific condition, are implemented according to their published literatures[2, 3] The first conditional version of GAN (c-GAN) was introduced by Mirza *et al.* in 2014. [2] The authors added an additional input layer with values of one-hot-encoded images labels to train a generator with image tagging ability. Auxiliary classifier GAN (AC-GAN) [3] shares samilarly in principle to the C-GAN. Unlike C-GAN, the input to AC-GAN’s discriminator is an image, while it outputs the probability of whether the image is real and its predicted classification as shown in Figure 1.

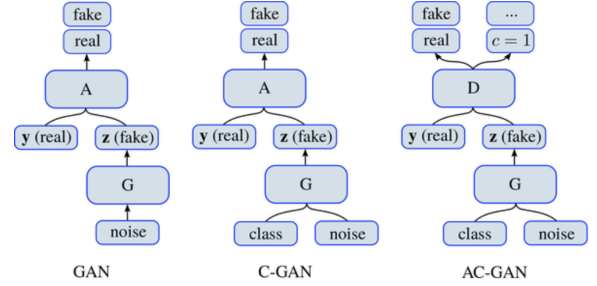


Figure 1. A comparison of different network architectures: GAN, C-GAN and AC-GAN (image borrowed from [4])

II. DATASET

This project uses ten manual selected sketch categories from the Quick, Draw! Dataset, which contains 50 million human drawings across 345 categories. [5] It is a unique and world largest doodling dataset, which has helped deep learning researchers observe patterns in how people around the world draw, as well as helped artists create innovative artworks. Each category contains 12,800 sketches and thus the training dataset contains a total of 128,000 images. Five samples from each category are shown in Figure 2.

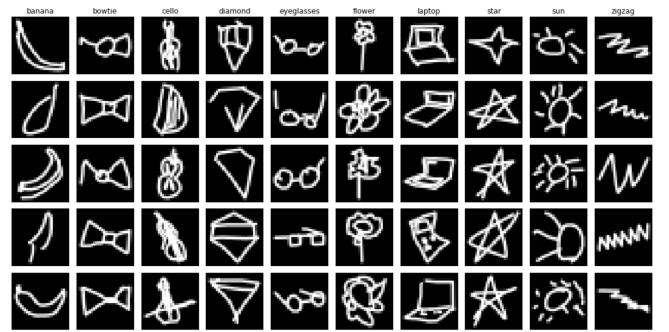


Figure 2. Training samples in selected ten categories:(left to right) banana, bowtie, cello, diamond, eyeglasses, flower, laptop, star, sun and zigzag

A. Preprocessing

The open-sourced dataset has been preprocessed and split into different formats to ensure fast and convent application and exploration. For this project, we are interested in the final drawing of all strokes and thus can omit the drawing sequence recorded inside the full raw data.

We plan to use the simplified drawings which have been rendered into a 28 x 28 grayscale bitmap in numpy format.

B. Dataset cleaning

After we examined our preliminary generator outputs, we are unsatisfied with the model outputs and suspect its due to a certain amount of low quality / mis labelled data in our dataset. By inspecting the dataset, each category contains 2-10% drawings that are unfinished or mislabelled. We consider those data as low quality noisy training data and decide to manually remove them from the training set. Even with a team of three, manually checking 128,000 images is too much workload and thus we decide to shrink our clean dataset to 12,800. Table I. Summarized our manual low quality image counts on the first 1700 images in each category.

TABLE I. LOW QUALITY IMAGE COUNTS

Category	Counts	Presentence Ratio
banana	146	8.6%
bowtie	104	6.1%
cello	173	10.2%
diamond	101	6.0%
eyeglasses	109	6.4%
flower	47	2.8%
laptop	74	4.4%
star	114	6.7%
sun	45	2.6%
zigzag	74	4.4%

As this part of data cleaning work has just finished right before the milestone report due time, all model results shown in this report were trained with raw dataset which contains bad images.

III. GAN MODELS AND DISCUSSIONS

A. C-GAN base model and preliminary result

Considering the image shape is relatively small (28, 28) with a single channel, a simple C-GAN architecture is chosen for this project. In order to make the architecture clear, Figure 3 shows a plot of the generator model (right) and the discriminator model (left). Inside the discriminator model, a second input is added to take an integer for the class label of the image. This has the effect of generating the input image conditional on the given class label. The class label is then passed through an embedding layer with size of 10. This means that each of the 10 classes (0 through 9) will map to a different 10-element vector representation that will be learned by the discriminator model. The output of the embedding is then passed to three fully connected hidden layers with leaky Relu activation. We used tanh and sigmoid as output layers for the generator and discriminator, respectively. The cost of C-GAN is similar as GAN:

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

,where $D(\mathbf{x}|\mathbf{y})$ and $G(\mathbf{z}|\mathbf{y})$ demonstrates we are discriminating and generating an image given a label \mathbf{y} .

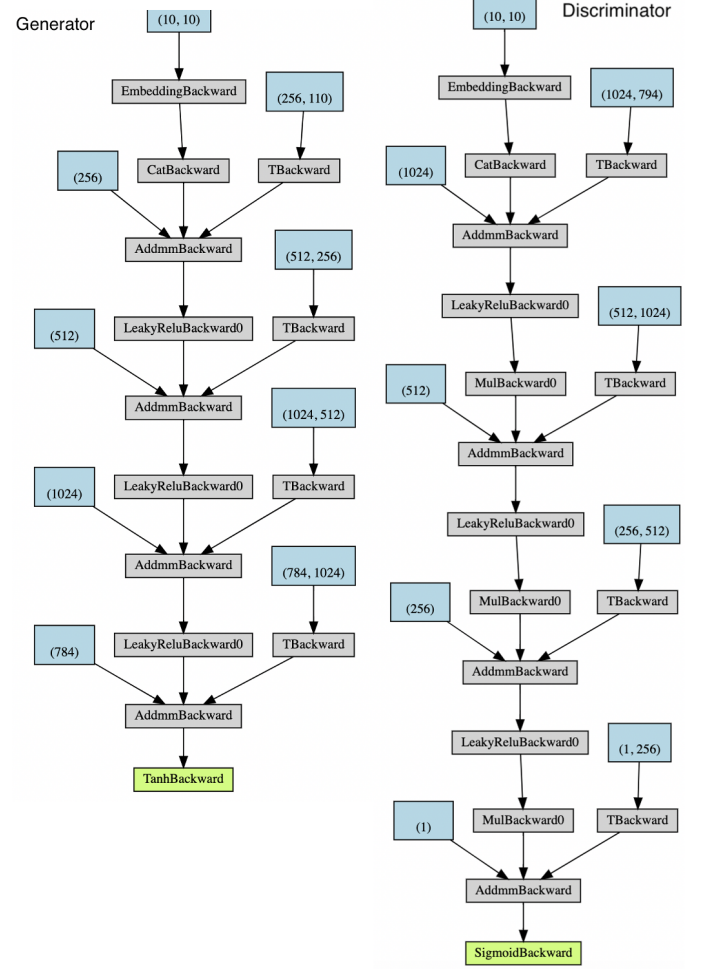


Figure 3. C-GAN network graph: generator (right) and discriminator (left)

The C-GAN model is trained with batch size 128 and total 300 epochs. In the initial stage of training, we observed larger variation of both generator and discriminator loss, and noticed the continuous quality improvement in the generated images. When the training exceeded 100 epoches, we noticed the model close to converge, with both D, G loss remaining much smaller variation and no significant improvement in terms of fake image quality. Figure 4. shows generated images at epoch 300. It is clear that the fake image is much noisier and blurry compared to the real images. We suspect two potential root causes: inconsistent data quality as discussed in Sec II. B and model architecture may be too simple that the model is now having a high bias issue.

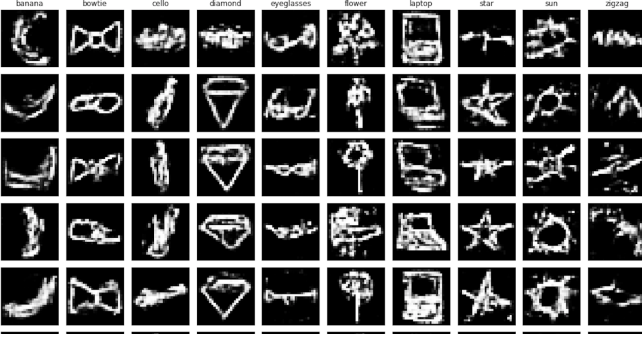


Figure 4. C-GAN generated images at epoch 300

B. AC-GAN base model and preliminary result

In addition to the C-GAN base model, an AC-GAN model is developed to improve generated image quality. Table II below shows the network architecture of its generator model.

TABLE II. AC-GAN GENERATOR MODEL ARCHITECTURE

Layer Type	Parameters	Output Dimension
Embedding Layer	class#: 10; input size:(1,100)	(1,100)
Linear		(1,128*8*8)
Reshape		(1,128,8,8)
BatchNorm		(1,128,8,8)
Upsample	factor = 2	(1,128,16,16)
Conv2d	channel = 128, filter = 3, stride = 1, padding = 1	(1,128,16,16)
BatchNorm		(1,128,16,16)
LeakyRelu	alpha = 0.2	(1,128,16,16)
Upsample	factor = 2	(1,128,32,32)
Conv2d	channel = 64, filter = 3, stride = 1, padding = 1	(1,64,32,32)
BatchNorm		(1,64,32,32)
LeakyRelu	alpha = 0.2	(1,64,32,32)
Conv2d	channel = 1, filter = 3, stride = 1, padding = 1	(1,1,32,32)
Tanh		(1,1,32,32)

The discriminator module is defined by totally 14 layers with repeat user defined block patterns. The general pattern of each block is defined as following flow:

Conv2d → LeakyRelu → Dropout → Batchnorm(optional)

The conv2d layer is a 2d convolution layer with filter size = 3, stride = 2 and padding = 1. The LeakyRelu alpha factor is set to 0.2 as a constant. The probability to keep an element is 0.75 in the dropout layer. The input for this user

defined block is input filter number, output filter number and boolean flag for batchnorm.

The discriminator neural network is defined as shown in following table:

TABLE III. DISCRIMINATOR NEURAL NETWORK LAYER STRUCTURE

Layer Type	Parameters	Output Dimension
block	input filter = 1 output filter = 16 batchnorm disabled	(1,16,16,16)
block	input filter = 16 output filter = 32 batchnorm enabled	(1,32,8,8)
block	input filter = 32 output filter = 64 batchnorm enabled	(1,64,4,4)
block	input filter = 64 output filter = 128 batchnorm enabled	(1,128,2,2)

The output of the above neural net is used to generate validation prediction and label prediction. The validation prediction layer consists of a linear layer followed by sigmoid activation. The label prediction layer consists of a linear layer followed by softmax activation.

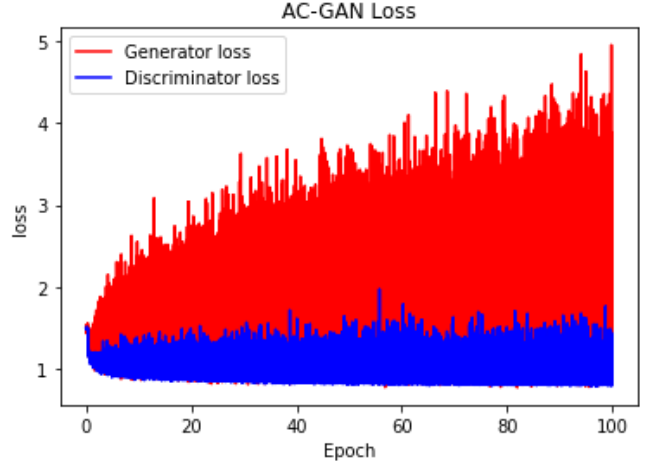


Figure 5. AC-GAN Generator and Discriminator training loss plot of 100 epoch.

Both Generator and discriminator loss is shown in Figure 5. It clearly shows that both generator loss and discriminator loss oscillates and does not converge.

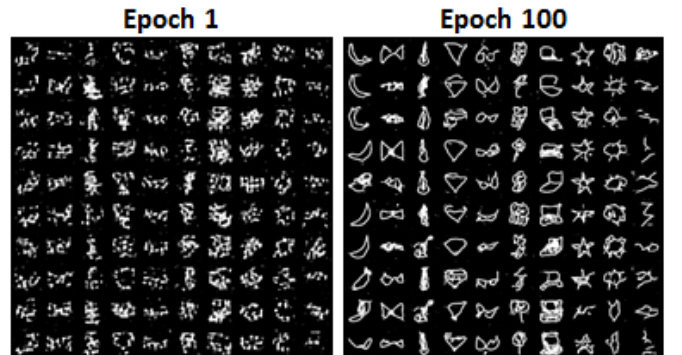


Figure 6. AC-GAN Generated images at the first epoch and epoch 100.

The randomly generated doodle drawings are shown in Figure 6. Ten different categories are separated in 10 columns. The comparison shows that the generator generates fake images with very poor quality at its initial stage. After 100 epochs of training, the generator generates doodle drawing with similar quality as input image data based and shows clear signature of different categories. More details regarding GAN models comparison and evaluation are discussed in Sec. IV.

C. C-GAN model improvements with data cleaning and hyperparameter tuning

Compared with the AC-GAN outputs, our C-GAN model shows much weaker performance. The team suspects the large presence of low quality data may contribute to this and thus create a cleaned dataset with ten times smaller amount as test force. Without adjusting the model architecture and hyperparameter but just using the cleaned dataset, the model does show improvement with human evaluation and this lately got confirmed with our evaluations with the Fréchet Inception Distance (FID) and Kernel Inception Distance (KID) scores. We only obtain 1.67 reduction in FID and 5.57 reduction in KID compared with the model trained with raw dataset. One potential reason data cleaning only plays a subtle role in model improvement is that while we cleaned up the dataset, we also cut a large amount (~90%) of the dataset due to lack of algorithm to automate the cleaning process. As shown in Figure 7, the team noticed that the banana category benefits the most with data cleaning as its raw dataset contains the second largest noise data and its drawing’s relative simplicity may suffer the most of noise data impact.

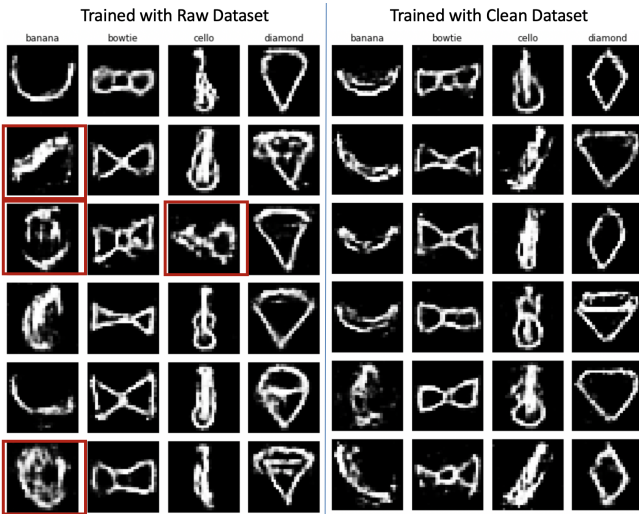


Figure 7. Comparison of generated images with model trained with raw dataset (left) and clean dataset (right). Red boxes marked human unrecognizable fake images.

In addition to the data cleaning, the team tried hyperparameters tuning with adjusting batch size, adding batch norm layer, adjusting dropout probabilities as well as adjusting learning rate. Unfortunately, none of the mentioned techniques works well in our model tuning. For

certain parameter adjustments from the suggested default value, we even observe a worsen result. For the concern of content limitation and most importantly our reader’s time, those failed attempts and outputs are skipped in this report.

D. Truncation Impact

With no luck in hyperparameter tuning, the team applied the truncation trick on our C-GAN clean data model. The truncation trick is a latent sampling procedure for generative adversarial networks, where generator noise is sampled from a truncated normal (where values which fall outside a range are resampled to fall inside that range). Its original implementation was reported in Megapixel Size Image Creation with GAN[6]. In BigGAN, the authors found that truncation provided a boost to the Inception Score and FID by allowing fine control over the trade-off between sample fidelity and variety.[7]

During the training, The generator noise is trained with sampling from normal distribution $\mathcal{N}(0, 1)$. After the training, if we sample more around zero with a largely truncated normal distribution, our generator produces high quality less diversified images (high fidelity sampling at truncation-0.7). With more noise sampled towards distribution tails, we start to see low quality images with truncation-0.25. The fidelity of these images becomes lower due to our generator unable to get its weights to make that noise vector into a beautiful realistic image. it fails to collect as much feedback on its realism on those noise vectors sampled from these regions during training.

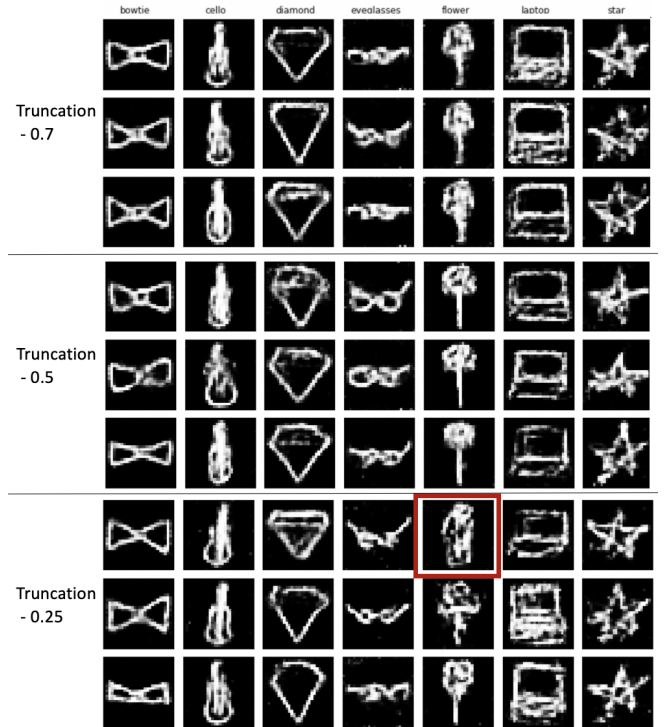



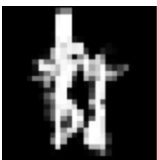

Figure 8. Impact of image quality with truncation level. Value more close to 0 indicates more less truncation in the distribution tail and its distribution more close to normal distribution. The red box marked a human unrecognizable fake image.



IV. MODEL EVALUATION

Evaluating the quality of GAN model outputs is an open and difficult problem. Two metrics, FID and KID, are selected to evaluate the quality of images generated by generative adversarial networks, and lower scores have been shown to correlate well with higher quality images.

The Fréchet Inception Distance (FID), invented by Heusel *et al.*[8] measures the similarity of the samples' representations in the classic Inception-V3 object detection architecture to those of samples from the target distribution. The FID fits a Gaussian distribution to the hidden activations (for this project, we use the pool3 layer, of dimension 2048) for each distribution and then reports the Fréchet distance as FID scores. One major problem the team faces while applying the FID evaluation is that FID estimates exhibit strong bias for sample n even up to 10,000. As the major computation pipeline is built on Google Colab and accessing large mount data on mounted google drive can be extremely difficult and slow. We tried our best to evaluate model samples with 10K, but still not certain if the sample amount is sufficient enough. For comparison and confirmation of the evaluation, we also applied Kernel Inception Distance (KID) as a secondary evaluation metric. KID estimates are unbiased, and standard deviations shrink quickly even for small n in thousands range. Binkowski *et al.* [9] used a polynomial kernel to avoid correlations with the objective of Maximum Mean Discrepancy GANs as well as to avoid tuning any kernel parameters. In addition to distribution mean and variance, KID also computes skewness. Table IV summarizes our four GAN models' FID (with 10K samples) and KID (with 1K samples) scores. AC GAN gets the highest FID score with its sharp outline while C-GAN-Clean data scores highest for KID. The team think this may be due to KID credits outputs' shape similarity more. Although the team likes the samples generated with truncation better, its scores are very close to non-truncated C-GAN clean data. This may reflect the tradeoff between fidelity and diversity while scoring.

TABLE IV. MODELS' FID KID SCORES

	Sample Image	FID-10K	KID-1K
Real Image		0.	0.
C-GAN -Raw Data		62.80	10.31
C-GAN -Clean Data		61.13	4.74

C-GAN -Truncation0.5		63.96	4.84
AC-GAN		41.21	5.86

V. CONCLUSION

Two GAN models were implemented and compared through this work. Although neither of them is great enough to successfully produce a human unrecognizable replica, the team got a chance to learn and understand the model architectures in detail. Data cleaning and sampling truncation are found two major knobs in improving fake images' quality. Evaluating the models is a hard task for the team as none has used the dataset for GAN. Despite lack of comparable work from literature, FID and KID scores are computed and found in general align with human judgement.

REFERENCES

- [1] <https://kidscountryinc.com/2016/07/21/6-benefits-drawing-time-children/>
- [2] arXiv:1411.1784
- [3] arXiv:1610.09585
- [4] arXiv:1710.10564
- [5] <https://github.com/googlecreativelab/quickdraw-dataset>
- [6] arXiv:1706.00082
- [7] arXiv:1809.11096
- [8] arXiv:1706.08500
- [9] arXiv:1801.01401

CONTRIBUTIONS

Yiting Zhang worked on data pre-processing, data cleaning, C-GAN model setup and training, model quality improvement and generated images evaluation.

Tuo Shi worked on building and training two ACGAN models. The initial model was unsuccessful due to the concatenate method for the input noise source and one hot representation of image class. To fix the problem, another ACGAN model was trained by implementing an embedding layer in the generator model. The latter ACGAN model generates images with high quality.

Xinhua Ren worked on coding for preliminary models implementation and development, and wrote technique points in the report.

APPENDIX. A

The project code repository is located at:
<https://github.com/yitingz/AutoDoodling>