Conditional Generative Adversarial Net

E4040.2021Spring.AAAA.report
Weiminghui Ji wj2309, Shiyuan Zheng sz2936, Hongyi Zhou hz2700
Columbia University

***Abstract*—The problem of data generation conditioned on the class label can be divided into two kinds of problems: generating data conditioned on unstructured label (categorical label) or on structured label (e.g. texts). We study how to use conditional generative adversarial net (CGAN) and several other variants of GAN to deal with each problem. For the first problem, GAN, CGAN, ACGAN and InfoGAN are constructed and trained on two datasets: MNIST and Fashion-MNIST. The performance of the 4 GANs are compared and analyzed. We find that CGAN's performance is much better than the conventional GAN. InfoGAN has the best performance, and it doesn't differ too much from CGAN and ACGAN. All the three variants of GAN successfully learn to generate images given the label. For the second problem, we construct a new CGAN and train it on dataset "Oxford-102 flower dataset". We can find that the model can generate the images corresponding to the given text.**

1. Introduction

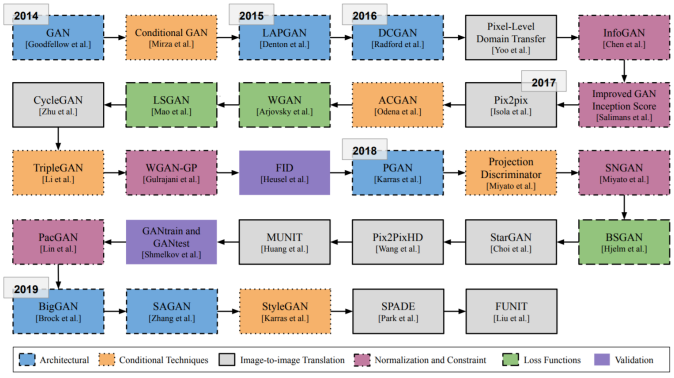

Figure 1: GAN Evolution

Background information. Probabilistic Generative Model can be used to generate random observable data. **Generative Adversarial Networks** (**GAN**) generates data by implicitly estimating the probability density function by using a network to transform a given random variable to a new random variable which follows the distribution of the data. [2] However, if the data have label, we can not control the what label the generated data will have using the conventional GAN. Conditional Generative Adversarial Networks (CGAN) is used to deal with this issue by using the label information in the network to direct the data generation process. [3]

The goal of the project is to show how CGAN and other alternative GANs can generate data conditioned on class labels, and compare their the performance with each other. We divide such data generating problem into 2 categories based the type of label associated with the data and we conquer each of the two kinds of problem.

The first problem is generating data with unstructured label. The label is categorical and is just simple class label. We will show how CGAN, ACGAN (Auxiliary Classifier GAN) [4] and InfoGAN (Information-theoretic extension to GAN) [5] generate data conditioned on the class labels and how GAN generates data without class labels. Their structures will be illustrated and their performance will be compared and analyzed. We train and test our model on MNIST and Fashion-MNIST dataset.

The second problem is generating data with structured label. For instance, labels can be texts (sentences or phrases). In this paper, we study a reverse image caption problem. From a caption of a picture, our trained CGAN is able to generate high-quality picture which corresponds to caption.

The first challenge of this project is there are many variants of GAN and the detailed mechanism and subtlety of GAN, CGAN and other variants are difficult to fully understand. For instance, how to formalize the optimization problems and how to use EM algorithm to solve them. To have deeper insights these GANs, we searched and studied many resources and papers. By reading and discussion, we fully understand the mechanism of GAN and its extensions, what improvement each GAN variant make and how they solve the problem.

The second challenge is the implementation of different GANs. There are no modules of GAN implemented in tensorflow 2 library, and almost all the GAN previously implemented are in tensorflow 1.x which are not well supported currently. This means that we need to implement GANs ourselves. Moreover, since the computation in GAN involves two nets (or more that two nets in its variants) and they interact with each other, there is not a simple method to build a GAN model like keras.Sequential. Therefore, we implement GANs from a rather bottom level. For instance, we write our own "train_one_step" functions, loss functions and so on.

The third challenge is the difficulty of training GANs.

It is well known that GANs are difficult to train and the training process can be very unstable. To deal with this challenge, we first try the initialization of the parameters of our model based on suggested parameters from the paper [3]. By observing the change of loss and accuracy during the training process, we tune the parameters accordingly. In the second problem, since the performance of the GAN with the suggested architecture is not so good and cannot be improved after we spend lots of time tuning the parameters, we explored and created a new architecture that work much better.

2. Summary of the Original Paper

2.1. Methodology of the Original Paper

The authors introduce the conditional version of generative adversarial nets. Both the data (x) and the label (y) are fed into CGAN. By introducing the label in the input of the network, which traditional GAN does not, they hope the net can learn to generate data conditioned on their labels.

2.2. Key Results of the Original Paper

The authors demonstrate the effectiveness of CGAN in two empirical experiments. One is on MNIST digit data set. They constructed CGAN which produces digit images conditioned on class labels. The results of the CGAN that they present are comparable with some other network based.

The other experiment is on MIR Flickr 25,000 dataset where the constructed CGAN can generate corresponding words conditioned on the image it receives.

3. Our Methodology

First, we believe that the problem of generating data based on label can be divided into: the case when the label is unstructured (e.g. categorical class label) and the case when the label is structured (e.g. texts, words). Therefore, for the first kind of problem, we will construct a CGAN which generates MNIST images conditioned on class labels. However, we will not reproduce the second experiment in the paper. This is because the second experiment that the original paper did was more related to image captioning and even though they deal with the problem using CGAN, many other methods can also be successfully applied in image captioning and can give better results, which the authors also admit to. Moreover, it seems GANs are more important in generating images than in other fields and our division of the kind of data generating problem is quite reasonable. So, in our second experiment, we choose to deal with reverse image caption problem, where we construct a CGAN to generate imaged based on captions.

Second, since there are more than one way to incorporating label information into GANs, we decide to try different extensions of GAN (i.e. CGAN, ACGAN, InfoGAN) and compare their performance in our first experiment. Conventional GAN is also implemented as a benchmark for these extensions of GAN.

Third, Frechet Inception Distance (FID) method is used to evaluate and compare the performance of the GANs. It is one of the most prevalent way to evaluate GANs and has been argued to have similar measurement characteristic like human. Besides, loss and accuracy are also used as metrics of evaluation.

3.1. Objectives and Technical Challenges

In the first experiment, we will construct GAN, CGAN, ACGAN and InfoGAN and train them on two datasets: MNIST and Fashion-MNIST. The goal is to make CGAN, ACGAN and InfoGAN learn to generate images conditioned on class labels, and make GAN learn generate images without label information. Then, we will compare and analyze their performance.

In the second experiment, we will construct CGAN and train it on dataset "Oxford-102 flower dataset". The goal is to make the CGAN learn to generate images based on texts.

As mentioned in the introduction section, there are three main challenges. First, we need to investigate many variants of GAN and the subtlety of GAN, CGAN and other variants takes time to understand. Second, we need to implement our GANs in tensorflow 2 with few related code currently available. And we have to implement GANs from a rather bottom and detailed level due to GANs' characteristics. Third, it is hard to train GANs, so we try to tune parameters and explore architecture.

4. GANs with Structured Labels

4.1. Implementation

Our work includes replicating the GAN, CGAN, ACGAN and InfoGAN (the later two are added) on MNIST and Fashion-MNIST dataset. Then, we build a new CGAN model to test its performance on reverse image captioning task.

Different GANs on MNIST and Fashion-MNIST

GAN

- **Framework:**

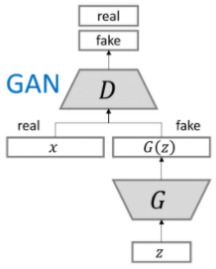

Figure 2: GAN Framework

- **Generator**

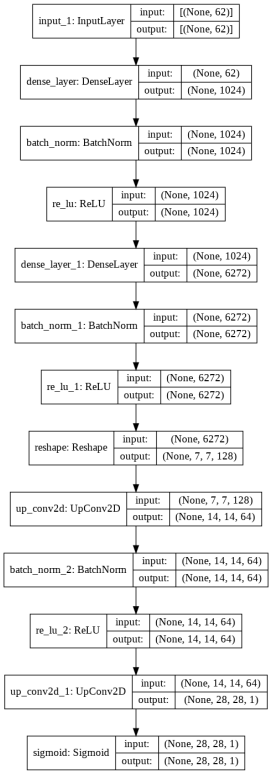

Figure 3: GAN Generator

- **Discriminator**

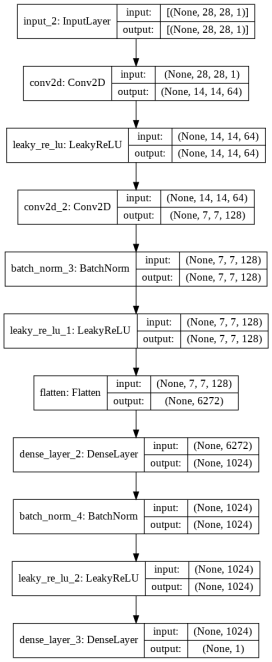

Figure 4: GAN Discriminator

- **Loss function:**
$\min_G \max_D V(D,G) = E_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + E_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$

CGAN

- **Framework:**

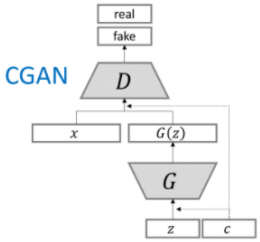

Figure 5: CGAN Framework

- **Features:**
· Log-Likelihood Estimates
· Make the model learn to generate data conditioned the on class labels by passing both data and labels during training
· Probabilistic one-to-many mapping is instantiated as a conditional predictive distribution, the input is taken to be conditioning variable
- **Generator**
Same as GAN Generator
- **Discriminator**
Same as GAN Discriminator
- **Loss function:**
$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))]$

ACGAN

- **Framework**:

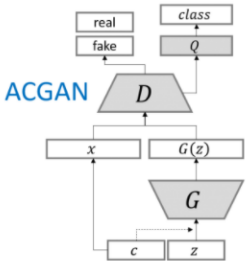

Figure 6: ACGAN Framework

- **Generator**
Same as GAN Generator
- **Discriminator**
Same as GAN Discriminator
- **Classifier**

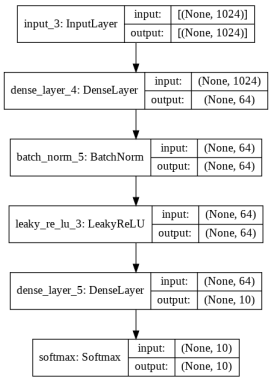

Figure 7: ACGAN Classifier

- **Loss function:**
  L_{s} : *log-likelihood of the correct source*
  L_c : *log-likelihood of the correct class*

$$
\begin{aligned}
L_S =& E\left[\log P\left(S=\text{ real }|X_{\text{real}}\right)\right] \\
&+ E\left[\log P\left(S=\text{ fake }|X_{\text{fake}}\right)\right] \\
L_C =& E\left[\log P\left(C=c|X_{\text{real}}\right)\right] \\
&+ E\left[\log P\left(C=c|X_{\text{fake}}\right)\right]
\end{aligned}
$$

Generator Loss $= \operatorname{argmax}(L_s + L_c)$
Discriminator Loss $= \operatorname{argmax}(L_s - L_c)$

InfoGAN

- **Framework:**

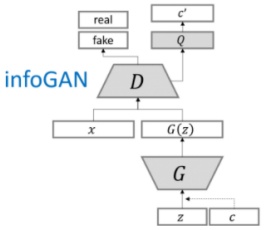

Figure 8: InfoGAN Framework

- **Key insights:**
  · A new term encourages high mutual information between generated samples and a small subset of latent variables c.
  · many domains naturally decompose into a set of semantically meaningful factors of variation
  · decompose the input noise vector into two parts:
  (i) z, which is treated as source of incompressible noise;
  (ii) c, which we will call the latent code and will target the salient structured semantic features of the data distribution.
  InfoGAN splits the Generator input into two parts:the traditional noise vector and a new "latent code" vector.
  The codes are then made meaningful by maximizing the Mutual Information between the code and the Generator output.

- **Generator**
  Same as GAN Generator
- **Discriminator**
  Same as GAN Discriminator
- **Classifier**
  Same as ACGAN Classifier
- **Loss function:**
  $\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c;G(z,c))$

$$
\begin{aligned}
I(c;G(z,c)) =& H(c) - H(c \mid G(z,c)) \\
=& E_{x\sim G(=,e)}\left[E_{c'\sim P(c|x)}\left[\log P\left(c' \mid x\right)\right]\right] \\
&+ H(c) \\
=& E_{x\sim G(+,c)}[\underbrace{D_{KL}(P(\cdot \mid x)\|Q(\cdot \mid x))}_{\geq 0} \\
&+ E_{e'\sim P(c|x)}\left[\log Q\left(c' \mid x\right)\right]] + H(c) \\
\geq& E_{x\sim G(z,c)}\left[E_{e'\sim P(e|x)}\left[\log Q\left(c' \mid x\right)\right]\right] \\
&+ H(c)
\end{aligned}
$$

Evaluation-FID FID is a measure of similarity between two datasets of images. It was shown to correlate well with human judgement of visual quality and is most often used to evaluate the quality of samples of Generative Adversarial Networks. FID is calculated by computing the Fréchet distance between two Gaussians fitted to feature representations of the Inception network.

The score was proposed as an improvement over the existing Inception Score, or IS. The inception score estimates the quality of a collection of synthetic images based on how well the top-performing image classification model Inception v3 classifies them as one of 1,000 known objects. The scores combine both the confidence of the conditional class predictions for each synthetic image (quality) and the integral of the marginal probability of the predicted classes (diversity).

The inception score does not capture how synthetic images compare to real images. The goal in developing the FID score was to evaluate synthetic images based on the statistics of a collection of synthetic images compared to the statistics of a collection of real images from the target domain.

Like the inception score, the FID score uses the inception v3 model. Specifically, the coding layer of the model (the last pooling layer prior to the output classification of images) is used to capture computer-vision-specific features of an input image. These activations are calculated for a collection of real and generated images.

The activations are summarized as a multivariate Gaussian by calculating the mean and covariance of the images. These statistics are then calculated for the activations across the collection of real and generated images.

The distance between these two distributions is then calculated using the Frechet distance, also called the Wasserstein-2 distance.

A lower FID indicates better-quality images; conversely, a higher score indicates a lower-quality image and the relationship may be linear.

The authors of the score show that lower FID scores correlate with better-quality images when systematic distor-

tions were applied such as the addition of random noise and blur.

4.2. Results: Different GANs comparison

- GAN

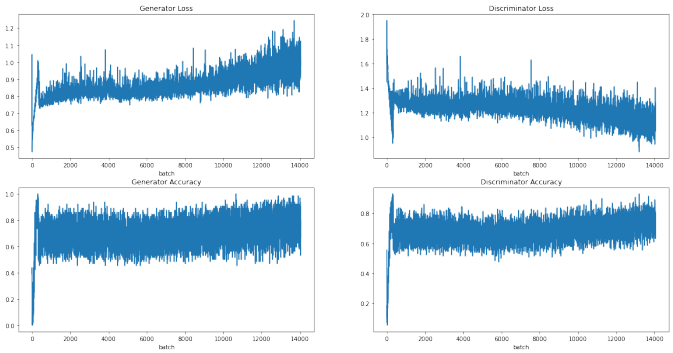

Figure 9: GAN Loss & Accuracy (MNIST)

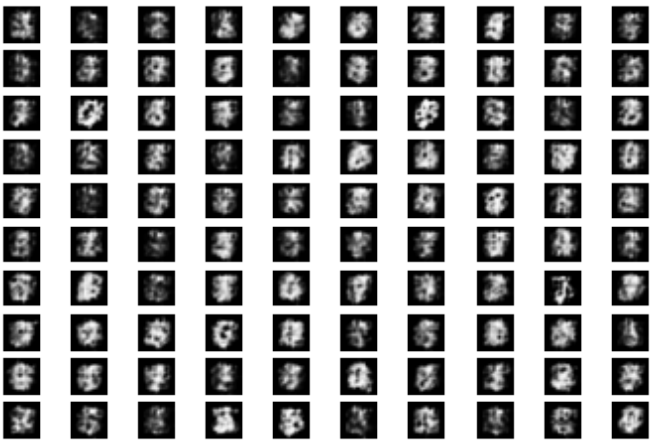

Figure 10: GAN sample generated image (MNIST)

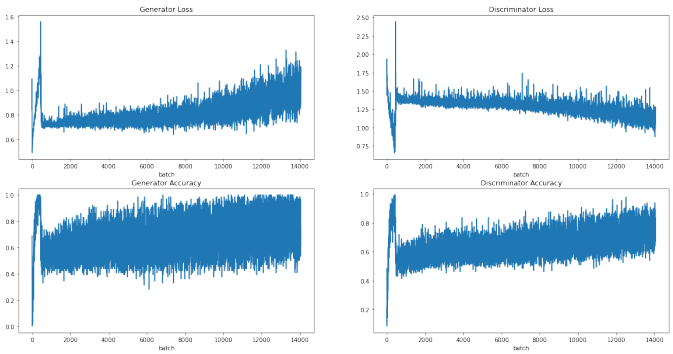

Figure 11: GAN Loss & Accuracy (Fashion-MNIST)

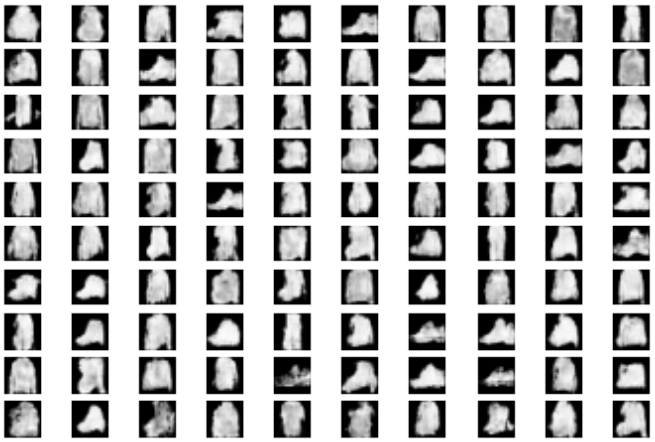

Figure 12: GAN sample generated image (Fashion-MNIST)

Comments:
We can find that the performance of GAN is pool. The image it generated are very ambiguous.

- CGAN

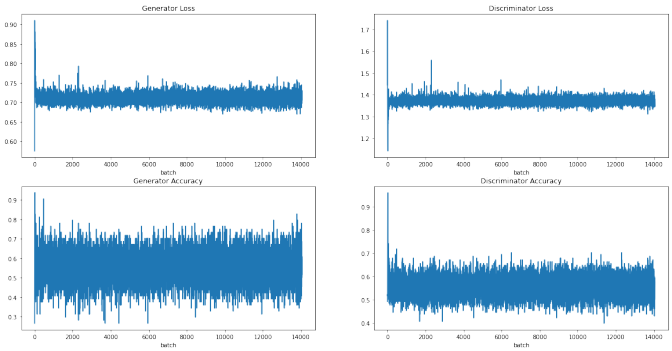

Figure 13: CGAN Loss & Accuracy (MNIST)

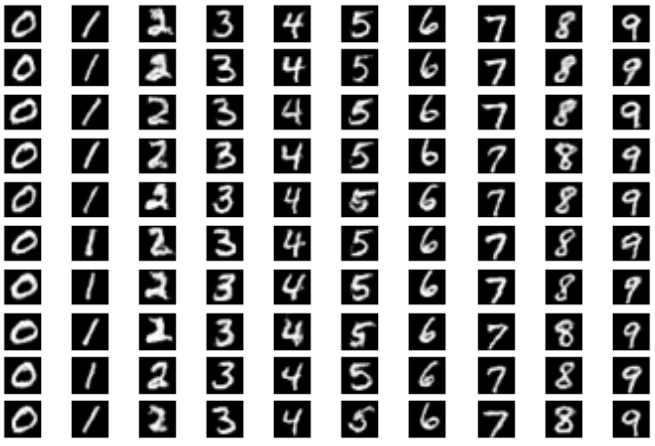

Figure 14: CGAN sample generated image (MNIST)

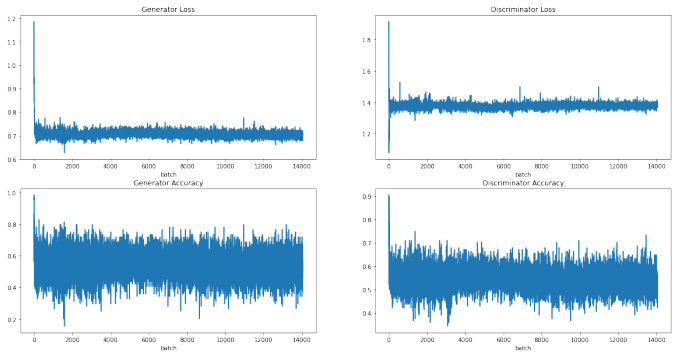

Figure 15: CGAN Loss & Accuracy (Fashion-MNIST)

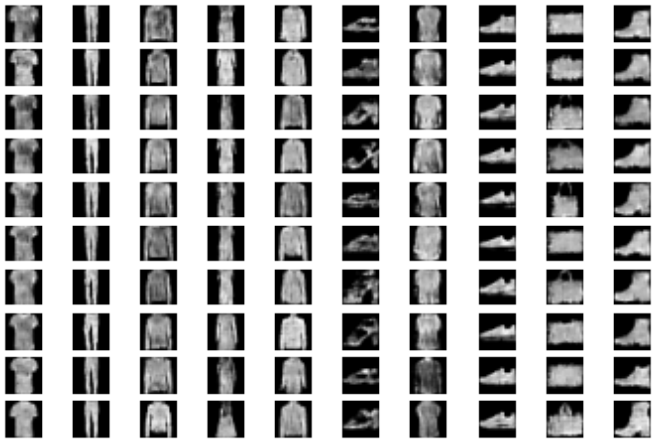

Figure 16: CGAN sample generated image(Fashion-MNIST)

Comments:
We can find that the performance of CGAN is good. The image it generated can be distinguishable.

- ACGAN

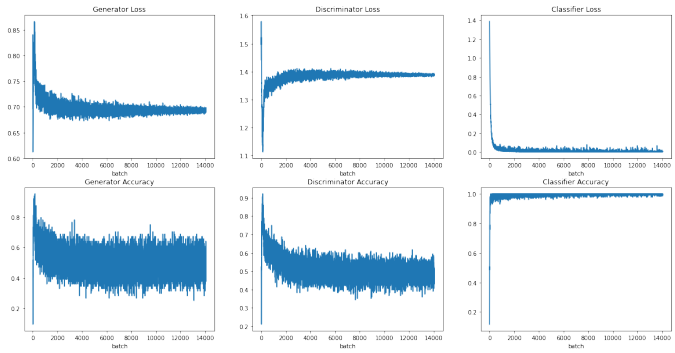

Figure 17: ACGAN Loss & Accuracy (MNIST)

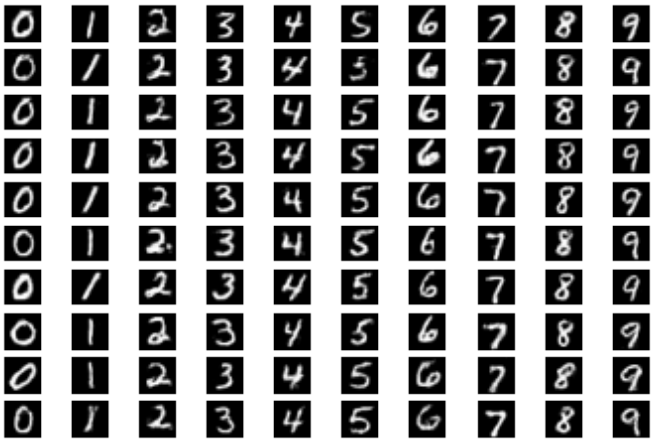

Figure 18: ACGAN sample generated image (MNIST)

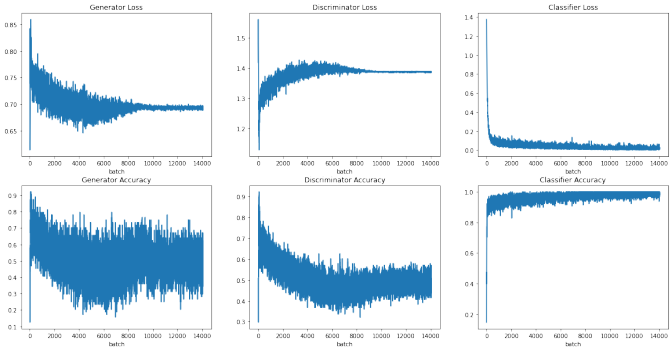

Figure 19: ACGAN Loss & Accuracy (Fashion-MNIST)

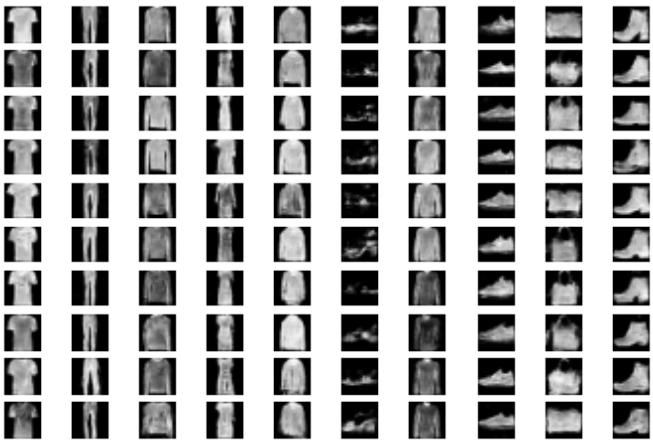

Figure 20: ACGAN sample generated image (Fashion-MNIST)

Comments:
We can find that the performance of ACGAN is also good. The image it generated can be distinguishable.

- InfoGAN

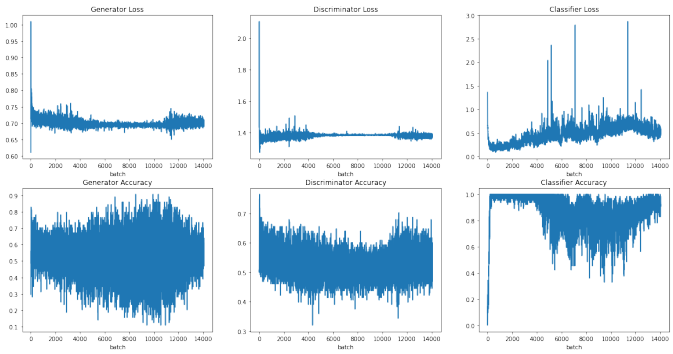

Figure 21: InfoGAN Loss & Accuracy (MNIST)

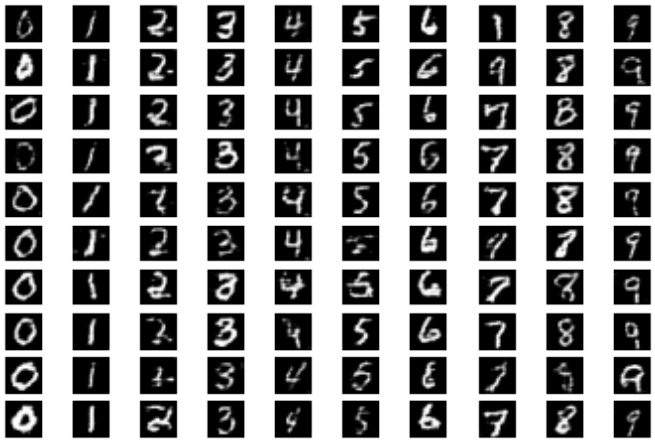

Figure 22: InfoGAN sample generated image (MNIST)

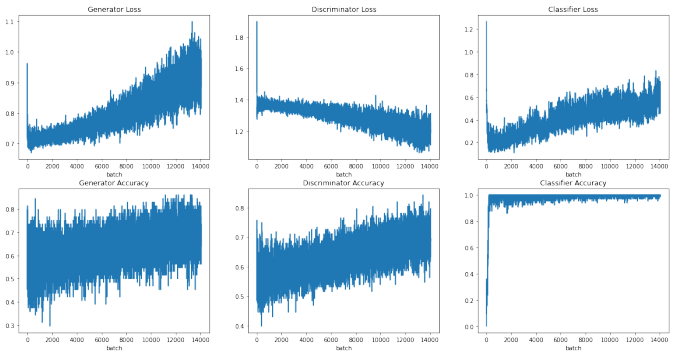

Figure 23: InfoGAN Loss & Accuracy (FashionMNIST)

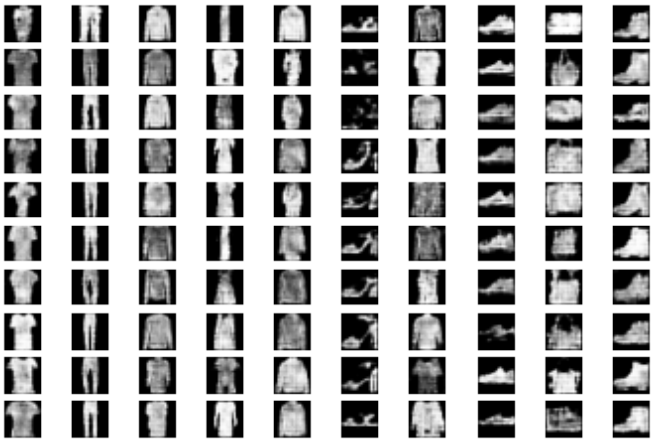

Figure 24: InfoGAN sample generated image (Fashion-MNIST)

Comments:
We can find that the performance of InfoGAN is acceptable. The image it generated can be distinguishable but some of them may be ambiguous and this may due to we have trained it for too many epochs.

- Comparison

label\Model	GAN	CGAN	ACGAN	InfoGAN
0	48.50104	24.22372	12.34759	11.98613
1	38.43417	8.262218	5.272202	5.519771
2	35.73007	16.45747	18.82452	14.41833
3	32.58208	18.9527	15.29742	10.69281
4	34.82875	13.38471	13.95656	11.5535
5	32.66027	17.88916	13.94423	11.93538
6	38.46155	21.35832	11.77081	13.75485
7	37.52359	15.50353	11.97365	8.398639
8	29.78755	15.41948	15.17358	11.5538
9	32.3888	16.91429	11.53984	10.03926
average	36.08979	16.83656	13.01004	10.98525

TABLE 1: Frechet Inception Distance (MNIST)

label\Model	GAN	CGAN	ACGAN	InfoGAN
0	69.99077	21.75427	15.5246	9.142492
1	101.3743	13.60992	9.339433	7.325413
2	51.76424	23.14966	15.95525	12.44956
3	74.28812	14.12675	14.00189	8.542314
4	40.94767	19.06063	12.96244	9.718835
5	120.8636	16.77815	22.49573	14.58932
6	48.99698	18.60875	16.63096	9.17438
7	120.689	13.31515	8.410337	9.635207
8	64.11165	26.78417	17.72144	15.63279
9	72.7509	17.19623	14.69308	13.33019
average	76.57773	18.43837	14.77352	10.95405

TABLE 2: Frechet Inception Distance (Fashion-MNIST)

Comments:
By compareing the FID scores of these four models in MNIST and Fashion-MNIST dataset, we can find that the performance rank is InfoGAN>ACGAM>CGAN>GAN (the lower average FID score means better performance).

We can also find that the original GAN's performance is much pooler than the other three model, while the other three don't differ very much on FID score. But we also should notice that though InfoGAN's FID score is the lowest in these four models, some of the images it generated can still be more ambiguous to recognize than those generated by CGAN or ACGAN. So, FID score still have some limitations and may need future improvement.

5. CGAN with Unstructured Conditions: Flower Generation with Captions

5.0.1. Introduction. In most cases, structured labels are used as conditions. In this project, we explored a much more challenging one: image generation with unstructured conditions.

We are interested in translating text in the form of single-sentence human-written descriptions directly into image pixels. For example, "this flower has petals that are yellow and has a ruffled stamen" and "this pink and yellow flower has a beautiful yellow center with many stamens". We developed a deep CGAN from scratch, which can translate text description into flower images.

Specifically, given a set of texts, we want to generate reasonable images with size 64x64x3 to illustrate the corresponding texts. Here we use Oxford-102 flower dataset and its paired texts as our training dataset. The dataset is available on the kaggle competition page: https://www.kaggle.com/c/datalab-cup3-reverse-image-caption-2020/data.

Figure 25: Text description: The petals of the flower are pink in color and have a yellow center

- 7296 images as training set, where each image comes with at most 10 texts. We randomly choose one text as the caption.
- For generation, each time we sampled 8 texts from the test set. We generated 8 64x64x3 images for each text for visualization.

5.0.2. CGAN Model Structure. Our model has three main parts:

- Text Encoder (blue)
- Generator Network (left)
- Discriminator network (right).

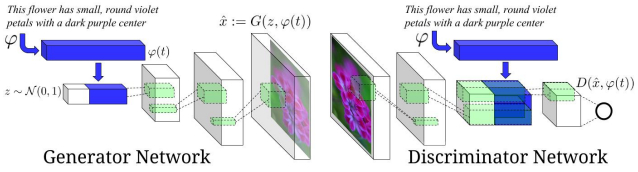

Figure 26: Framework of flower generation with text conditions

Text Encoder transforms words into high-dimension vectors using pretrained word vectors. In this task, we use 'glove-wiki-gigaword-100', which can be found at https://github.com/RaRe-Technologies/gensim-data. To be specific, for each flower image, we pick a caption which is a list of integers with maximum length of 20. Each integer stands for a word and can be transformed into a word using an id2word dictionary. Each word is embedded into a 100-dimension vector using the pretrained word vectors. Therefore, each text can be represented by a 100-dimension vector by calculating the mean of its word vectors. Anytime before a text t is fed into generative or discriminator network, it must pass through the text encoder, which is φ in the figure 26.

ID	Captions	ImagePath
6734	[[9, 2, 17, 9, 1, 6, 14, 13, 18, 3, 41, 8, 11,...	./102flowers/image_06734.jpg
6736	[[4, 1, 5, 12, 2, 3, 11, 31, 28, 68, 106, 132,...	./102flowers/image_06736.jpg
6737	[[9, 2, 27, 4, 1, 6, 14, 7, 12, 19, 5427, 5427...	./102flowers/image_06737.jpg
6738	[[9, 1, 5, 8, 54, 16, 38, 7, 12, 116, 325, 3, ...	./102flowers/image_06738.jpg
6739	[[4, 12, 1, 5, 29, 11, 19, 7, 26, 70, 5427, 54...	./102flowers/image_06739.jpg

Figure 27: Illustration of flower dataset

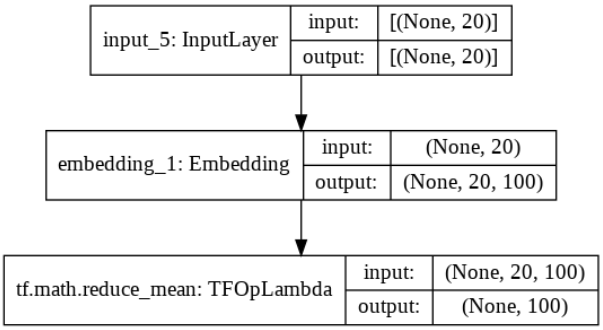

Figure 28: Text Encoder structure

Generator Network generates flower images given random Gaussian noises and text description of the flower. It hopes to generate images as real as possible to confuse the discriminator. As is described in 29, generator combines noises and embedded text information, and then passes them through dense layers and finally through CNN to generate images. In this task, we frequently applied batch normalization. Batch normalization is essential for make deep neural networks work without falling into mode collapse. Mode

collapse is the situation means generator creates samples with low diversity.

An important technical view in the generator is that instead of using convolution layers and pooling (downsample) as usual, GAN architectures are required to upsample input data in order to generate images. In the project, we learnt how to use the UpSampling2D and Conv2DTranspose Layers in Keras. The Upsampling layer has no weights and simply doubles the dimensions of input. The Transpose Convolutional layer is an inverse convolutional layer that can both upsample its input and learn how to fill in details during the training process. To put all into a nutshell, generator starts from abstract characters, expands them into larger tensors, and finally forms images. This process is totally in reverse to discriminator's behavior.

Notice that the final activation function is Sigmoid so that pixel values can be cast into $[0, 1]$.

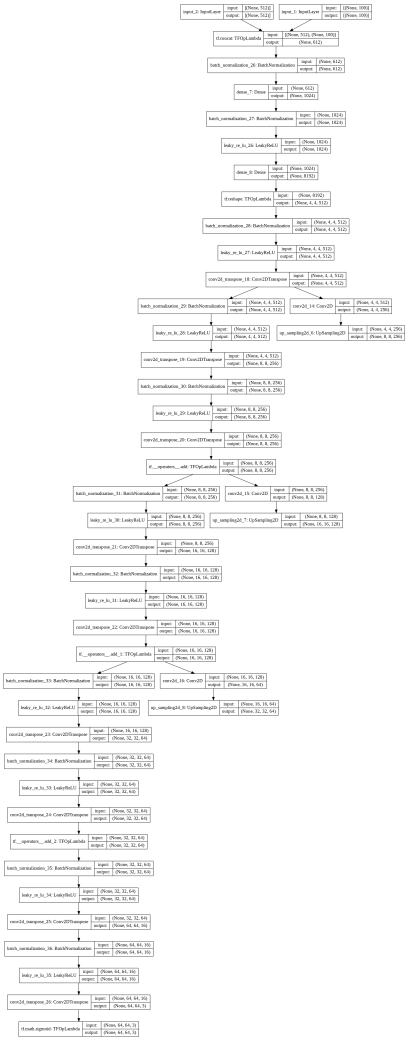

Figure 29: Generator structure

Discriminator Network As is stated above, discriminator is an "enemy" of generator. It starts from images and text description (labels), extracts abstract features, and comes to the final conclusion: whether the image is real with respect to the label.

An important lesson we learnt in the experiments is that Leaky ReLU is preferred in the discriminator. Different from the regular ReLU function, Leaky ReLU allows the pass of a small gradient signal for negative values. Therefore, it helps the gradients from the discriminator flows into the generator and prevents gradient vanishing problem. To be specific, instead of passing a gradient of zero in the back propagation for some units, it passes a small negative number.

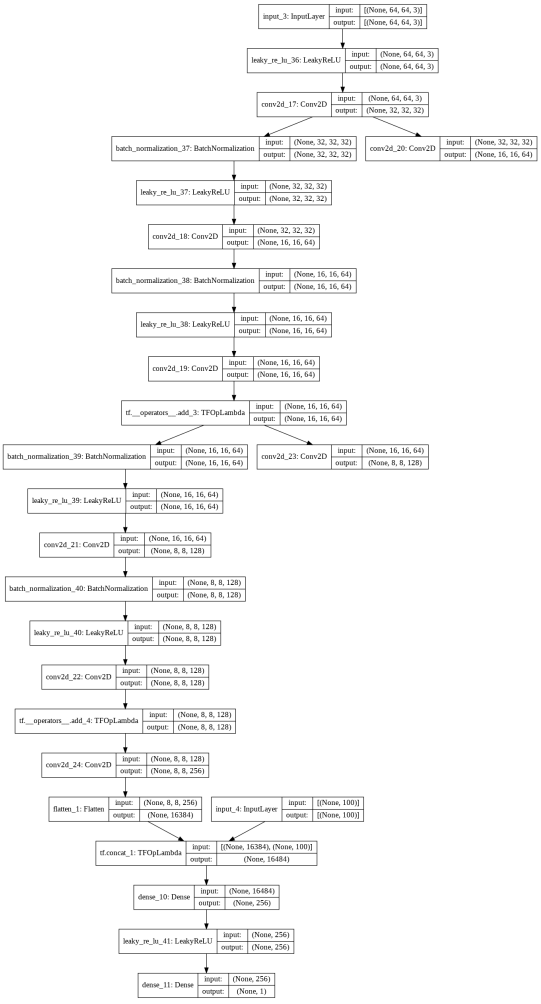

Figure 30: Discriminator structure

5.0.3. Training and Results. We ran 200 epochs with Tensorflow 2.0, costing 50 minutes in total. We plotted the loss and accuracy for both generator and discriminator 31, and showed the generated images after 200 epochs32 33.

Algorithm 1 Train Generator and Discriminator Networks with Text Conditions Alternately

for $epoch = 1 \rightarrow N$ **do**
  for $batch = 1 \rightarrow N//batchsize$ **do**
    Sample $RealImages$ and $Captions$
    $EmbeddedText = TextEncoder(Captions)$
    Generate $Noise$
    $FakeImages = Generator(EmbeddedText, Noise)$
    $RealPreds = Discriminator(RealImages, EmbeddedText)$
    $FakePreds = Discriminator(FakeImages, EmbeddedText)$
    Calculate loss of Discriminator
    Calculate loss of Generator
    Discriminator: Backward Propagation & update
    Generator: Backward Propagation & update
  end for
end for

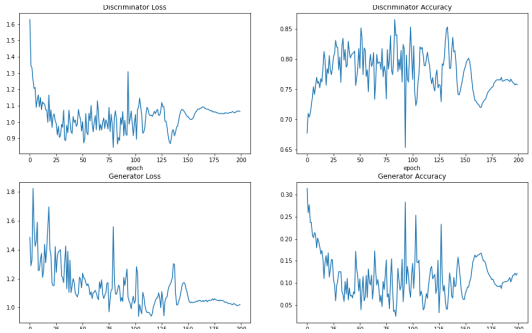

Figure 31: Loss and accuracy for both generator and discriminator

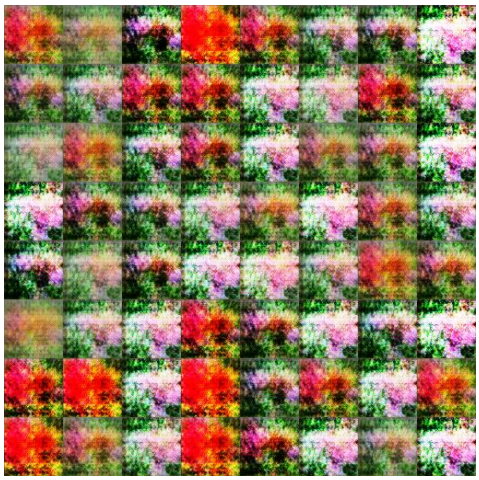

Figure 32: Generated flowers after 200 epochs - case 1

Figure 33: Generated flowers after 200 epochs - case 2

6. Comparison of the Results Between the Original Paper and Students' Project

The original paper presented the CGAN model and used it to generate MNIST digits, we replicated their work and also compared CGAN's result with other three GAN model(GAN, ACGAN and InfoGAN) by calculating FID score.

The second part of the paper was showing that CGAN can be used to generate tags for images. And we reversed this work by researching on how to applicate CGAN to generate images for given text.

7. Discussion of Insights Gained

7.1. Insights from Experiment with Structured Labels

During the first experiment, we tested four kinds of GANs and made a comparison. We can find that after considering the conditional probability to direct the model, the performance can be improved hugely than the original GAN. While ACGAN and InfoGAN are the variants of CGAN, they change the loss function and add a classifier class. The improvements is not too much when testing on MNIST and Fashion-MNIST and evaluating by FID score.

7.2. Insights from Experiment with Unstructured Conditions

The second experiment is generating colorful flower images with text description as conditions. This task is challenging as the 3-channel data is limited, and CGAN is hard to train.

We find that the **design of generator** is demanding and important. We initially applied dense layers without using CNNs, which cannot generate flower-like shapes. Therefore, CNNs is a must in this task. After that, we tried traditional

down-sampling layers, and the output 34 needs improvement. Finally, we realized that the generator should upsample (rather than downsample) by using UpSampling2D and Conv2DTranspose Layers in Keras. This is a wonderful process of learning from mistakes, gaining the experience of building a promising and reasonable generator.

What's more, we understood why **Leaky ReLU** is encouraged for the discriminator. We always hope the generator can see non-zero gradients rather than purely zero resulted by vanilla ReLU. Leaky ReLU prevents the deep networks from gradient vanishing.

Finally, we must emphasize the importance of word embedding using **pretrained word vectors**. Initially, we tried GRU and LSTM to embed text information, and updated the weights of text encoder with weights of generator and discriminator, which led to extremely unstable performance: sometimes it generated a flower, while sometimes loss exploded, and meaningless images are generated. Therefore, using pretrained word vectors to embed text information is preferred by CGAN.

All in all, training a CGAN on colorful images and complex conditions is never a simple story. It requires complete understanding of the model structures and hyperparameter tuning experiences. For further improvement of the results, we can enlarge the dataset size (which is small in our task) and try other variations of GAN (WGAN is a good choice).

Figure 34: Images generated without upsampling

8. Conclusion

In this project, we study two kinds of data generation conditioned on the class label problems: generating data conditioned on unstructured label (categorical label) or on structured label (e.g. texts). We study how to use conditional generative adversarial net (CGAN) and several other variants of GAN to deal with each problem. For the first problem, GAN, CGAN, ACGAN and InfoGAN are constructed and trained on two datasets: MNIST and Fashion-MNIST. The performance of the 4 GANs are compared and analyzed. We find that CGAN's performance is much better than the conventional GAN. InfoGAN has the best performance, and it doesn't differ too much from CGAN and ACGAN. All the three variants of GAN successfully learn to generate images given the label. For the second problem, we construct a new CGAN and train it on dataset "Oxford-102 flower dataset". We can find that the model can generate the images corresponding to the given text. But its result is still not so satisfactory and need some future improvements.

9. Acknowledgments

This project was developed in Tensorflow 2.3 framework, and we would like to thank Tensorflow developers. We also like to thank the professor and TAs for this course.

References

[1] Code for this project: https://github.com/ecbme4040/e4040-2021Spring-Project-AAAA-wj2309-sz2936-hz2700

[2] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative adversarial networks. Communications of the ACM, 63(11), pp.139-144., 2020.

[3] Mirza, M. and Osindero, S., Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784., 2014.

[4] Odena, A., Olah, C. and Shlens, J. Conditional image synthesis with auxiliary classifier gans. In International conference on machine learning (pp. 2642-2651). PMLR., July, 2017.

[5] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. arXiv preprint arXiv:1606.03657., 2016.

10. Appendix

10.1. Individual Student Contributions in Fractions

	UNI1	UNI2	UNI3
Last Name	Ji	Zheng	Zhou
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Reverse Image Caption	GANs	GANs
What I did 2	GANs Evaluation	GANs Evaluation	Reverse Image Caption
What I did 3	Report	Report	Report